Techniques for Proving Approximation Ratios in Scheduling

by

Peruvemba Sundaram Ravi

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Mathematics
in
Combinatorics and Optimization

Waterloo, Ontario, Canada, 2010

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Abstract

The problem of finding a schedule with the lowest makespan in the class of all flowtime-optimal schedules for parallel identical machines is an NP-hard problem. Several approximation algorithms have been suggested for this problem. We focus on algorithms that are fast and easy to implement, rather than on more involved algorithms that might provide tighter approximation bounds. A set of approaches for proving conjectured bounds on performance ratios for such algorithms is outlined. These approaches are used to examine Coffman and Sethi's conjecture for a worst-case bound on the ratio of the makespan of the schedule generated by the LD algorithm to the makespan of the optimal schedule. A significant reduction is achieved in the size of a hypothesised minimal counterexample to this conjecture.

Acknowledgements

I would like to acknowledge the assistance provided by my thesis adviser, Dr. Levent Tuncel. Levent's patience and willingness to accommodate my somewhat unusual situation made it possible to pursue this thesis on a part-time basis. Levent suggested that I work on the problem and the conjecture that form the main focus of this thesis. His continual detailed feedback and guidance made this thesis possible.

I would like to thank Dr. Jochen Konemann and Dr. Joseph Cheriyan for serving on my thesis committee and for taking the time to review this thesis. I would also like to thank Dr. Bill Cunningham, Dr. Stephen Vavasis, Dr. Bruce Richter, and Dr. Jeffrey Shallit, who taught the courses that I took at the University of Waterloo.

Above all, I would like to thank my family for their love, support and encouragement.

Dedication

I would like to dedicate this thesis to my parents.

Contents

Chapter 1

Introduction

Various performance criteria may be used to schedule n independent jobs on m parallel identical machines. These criteria often involve the minimization of some measure of time for the set of n jobs. The term makespan refers to the amount of time that elapses between the instant when the processing of the first job in a batch of jobs begins and the instant when the processing of the last job in the batch is completed. Makespan minimization tends to ensure the earliest possible completion date for a set of jobs. The flow time of a job is the amount of time that elapses between the instant when a job is released to the system and the instant when the processing of the job is completed. The mean flow time for a set of jobs is the average value of the flow time for that set of jobs. Minimization of mean flow time tends to minimize the average amount of time spent by a job in the system and therefore the mean level of work-in-process inventory in the system. Minimizing the makespan has been shown to be an NP-complete problem. Various versions of this problem have been studied by several researchers. Graham (1966) examines the problem of makespan minimization for a set of jobs with a partial order (precedence constraints) on a set of parallel identical machines. He shows that the ratio of the makespan of any schedule that satisfies these constraints to the makespan of the optimal schedule that satisfies these constraints cannot exceed $2-1/m$. He provides examples that show that this ratio can be achieved. Graham (1969) proves that if there are no precedence constraints and if jobs are sequenced based on the LPT (Longest Processing Time) rule, with the longest remaining task being scheduled next, the best possible bound for the ratio of the makespan of any such schedule to the optimal makespan is $4/3-1/(3m)$. Graham also considers the case in which each of the m largest jobs is assigned to a different machine and the remaining jobs are subsequently assigned in any order. In this case, the best possible bound for the makespan ratio is $3/2-1/(2m)$. A comprehensive survey of these and related results is contained in Graham (1972). Coffman and Sethi (1976b) obtain an improved bound for the LPT for situations in which the number of jobs is large. They show that, if at least k jobs are assigned to

each machine, the $4/3 - 1/(3m)$ bound can be replaced by a $(k+1)/k - 1/(mk)$ bound. Thus, if the number of jobs is very large, the ratio is significantly smaller than the ratio obtained by Graham, and it approaches 1 for large values of k.

The bin-packing problem may be regarded as the dual problem to the makespan minimization problem. In the bin-packing problem, a set of items of — in general — unequal sizes $\ell(T_i)$ must be packed into a set of m bins, each of a given capacity C. The objective of the bin-packing problem is to minimize the number of bins m used for the packing. In the FFD (First Fit Decreasing) algorithm (Johnson (1974) and Johnson et al. (1974), the bins are assigned indices 1 through m. The largest remaining item is assigned to the lowest-indexed bin into which it can fit. The FFD solution is the smallest value of m for which the algorithm is able to pack the entire set of items into m bins. Garey and Johnson (1981) and Coffman, Garey and Johnson (1983) have proposed various alternatives to the FFD algorithm. In these algorithms, items are ordered arbitrarily - not based on increasing or decreasing order of size. In the NF (Next Fit) algorithm, starting with the first item, items are assigned to the lowest-indexed bin in which they can fit. If an item does not fit in a bin, that bin is permanently closed. Thus, starting with the lowest-indexed bin, the bins are filled in succession. Another alternative is the FF (First Fit) algorithm, items are assigned to the lowest-indexed bin in which they can fit. However, a bin is closed only when the entire capacity of the bin has been used up. In the Best Fit (BF) algorithm, an item is assigned to the lowest-indexed bin in which it can fit and which has the *smallest* amount of capacity remaining after the assignment of the item. In the Worst Fit (WF) algorithm, an item is assigned to the lowest-indexed bin in which it can fit and which has the *largest* amount of capacity remaining after the assignment of the item.

Coffman, Garey, and Johnson (1978) propose an algorithm for makespan minimization, the Multifit algorithm, that is based on the FFD algorithm for the bin-packing problem. The authors obtain a solution for the makespan minimization problem by attempting to find the smallest bin capacity (makespan) for which a set of items (jobs) can be packed into m bins (machines). They develop upper and lower bounds for the capacity and subsequently use a binary search procedure to obtain FFD solutions to a series of bin-packing problems. The final suggested solution to the makespan minimization problem is the smallest capacity for which a feasible solution is obtained (for a given value of m) and the corresponding assignment of items to bins. Dosa (2000 and 2001) proposes generalized versions of the LPT and Multifit methods and provides numerical results that show that these versions often provide better solutions than the original version of these algorithms. Chang and Hwang(1999) extend the Multifit algorithm to a situation in which different processors have different starting times. Coffman, Garey, and Johnson (1978) provide a bound of 1.22 for the performance of the Multifit algorithm. Friesen (1984) and Yue, Kellerer and Yu(1988) provide alternative proofs for a bound of 1.2 for the Multifi algorithm. Friesen showed that the bound could not be less than 13/11. Yue (1990) and

Cao(1995) provide a tight bound of 13/11 for the performance of the Multifit algorithm.

Using the notation proposed by Graham, Lawler, Lenstra and Rinnooy Kan (1979), the makespan minimization problem for m parallel identical machines, where m is fixed, is denoted by $Pm \,||\, C_{max}$. The problem $Qm \,|r_i|\, C_{max}$ (m uniform machines, where m is fixed, release times, a makespan minimization objective) has been shown to be pseudopolynomially solvable by Lawler, Lenstra, Rinnooy Kan and Shmoys (1993). It follows that $Pm \,||\, C_{max}$ (makespan minimization on m parallel identical machines without release times and with fixed m) is also pseudopolynomially solvable. Vega and Leuker (1981) propose a PTAS (Polynomial Time Approximation Scheme) for the bin packing problem that runs in linear time. Hochbaum and Shmoys (1987) propose a PTAS for the makespan minimization problem for parallel identical machines. Ho and Wong (1995) propose an $O(2^n)$ algorithm, the two-machine optimal scheduling (TMO) algorithm (based on a lexicographic search procedure) to find the optimal solution for $P2 \,||\, C_{max}$. They use a simulation study to show that the average runtime for the TMO algorithm tends to be significantly better than exponential and is less than that required by the Multifit algorithm.

Besides makespan minimization, another objective examined by researchers is the minimization of mean flow time on a set of parallel identical machines. A schedule that minimizes mean flow time is termed a *flowtime-optimal schedule*. The mean flow time for a set of jobs on a set of parallel identical machines can be readily minimized by using the SPT (Shortest Processing Time) rule. This rule requires jobs to be assigned in nondecreasing order of processing times. If there are a total of m machines, each of the first m jobs is assigned to the first position on any of the m machines, each of the next m jobs is assigned to the second position on any of the m machines, and so on until every job has been assigned. The optimality of this rule follows from the inequalities that were proposed and proved by Hardy, Littlewood and Polya (1934). The optimality of this rule can be proved by using the following argument.

Notice that, for a given number of jobs, the minimization of the mean flow time is equivalent to the minimization of the total flow time. For a set of n jobs assigned to m machines, the total flow time is a weighted sum of flow times $\sum_{j=1,\ldots,n}((p_j)(n_j+1))$, where p_j denotes the processing time of job j and n_j refers to the number of jobs that are processed after job j on the same machine as job j. We assume that the processing times are strictly greater than zero and are arranged in nonincreasing order i.e. $p_j \geq p_{j+1}$ for $j = 1, 2, \ldots, n-1$. Let $N'' \equiv \{n''_1, n''_2, \ldots, n''_n\}$ and $N''' \equiv \{n'''_1, n'''_2, \ldots, n'''_n\}$, where N'' and N''' are feasible sets of values of n_j for $j = 1, \ldots, n$ that satisfy the following: $n''_j \leq n'''_j$ for $j = 1, \ldots, n$, and $n''_u < n'''_u$ for some value of u that satisfies $1 \leq u \leq n$. Note that a feasible set of values of n_j is a set of integers that corresponds to one or more feasible solutions to any instance of the problem with n jobs and m machines. Clearly, using the set of values N'' results in a weighted sum of flow times that is less than the weighted sum of flow times that results from using the set of values N'''.

Let $N_{opt} \equiv \left\{n_{1_{opt}}, n_{2_{opt}}, \ldots, n_{n_{opt}}\right\}$ denote a feasible set of weights n_j that results in a weighted sum of flow times that is less than or equal to that attained by any other feasible set of weights. For any given set of weights, the smallest weighted sum is attained by assigning weights to the processing times in nondecreasing order. Thus $n_{j_{opt}} \leq n_{(j+1)_{opt}}$ for $j = 1, \ldots, n-1$. It is evident that N_{opt} consists of m zeros, followed by m ones,..., followed by $n - m(\lceil \frac{n}{m} \rceil - 1)$ values that are equal to $\lceil \frac{n}{m} \rceil - 1$. Thus $n_{j_{opt}} = \left\lceil \frac{j_{opt}}{m} \right\rceil - 1$. This proves the optimality of the SPT rule.

The SPT rule generates a very large number of flowtime-optimal schedules. If we have a stream of jobs arriving in the system, the average number of jobs in the system (also referred to as the work-in-process inventory or WIP inventory) at any point in time is equal to the product of the arrival rate and the average amount of time that each job spends in the system. The average amount of time that each job spends in the system is the mean flow time. The objectives of minimizing WIP inventory (by minimizing the mean flow time) and ensuring the earliest possible completion date (by minimizing the makespan) can be combined by selecting the schedule with the smallest makespan among the class of all flowtime-optimal schedules. In this thesis, we term this problem the FM ('flowtime-makespan')problem. Using the notation proposed by Graham et al., the FM problem is denoted by $P \,||\, F_h(C_{max}/\Sigma C_i)$. This problem is known to be NP-hard (Bruno, Coffman and Sethi, 1974). The development of approximation algorithms for this problem

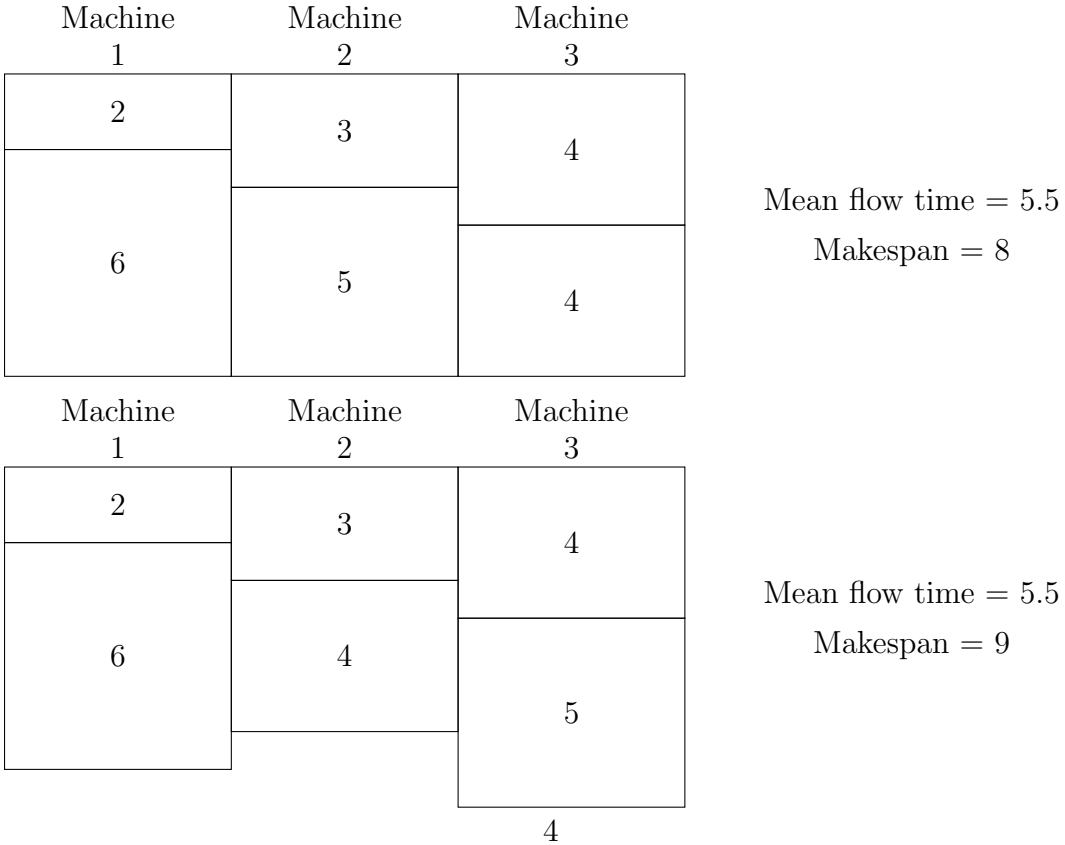

Figure 1: Two flowtime-optimal schedules with different makespans

and the development of worst-case performance bounds for these algorithms is therefore of considerable interest. Bruno, Coffman and Sethi obtain bounds on the ratio of the makespan of any flowtime-optimal schedule to the makespan of an LPT schedule. They show that this ratio lies between $(4m-1)/(5m-2)$ and $(2-1/m)$ and that these are the best possible bounds.

Conway, Maxwell and Miller (1967) develop the notion of ranks for the FM problem. A schedule is flowtime-optimal if jobs are assigned in decreasing order of *ranks*, with the jobs in the first rank being assigned last. If n is the number of jobs and m is the number of machines, we may assume that m divides n. (If it does not, we add $(\lceil n/m \rceil * m - n)$ jobs with zero processing times.) If we assume that the jobs are numbered in nonincreasing order of processing times, with job 1 having the largest processing time, the set of jobs belonging to rank r are the following: $(r-1)m+1, (r-1)m+2, \cdots, (r-1)m+m$. Any schedule in which all rank $r+1$ jobs are started before all rank r jobs (where $r = 1, 2, \cdots, (n/m)-1$), there is no idle time between successive jobs assigned to the same machine, and all rank n/m jobs start at time 0, is flowtime-optimal. Figure 1 consists of two figures that show two flowtime-optimal schedules for an instance of the FM problem with three machines and six jobs. Note that the vertical axis denotes time, with the upper left-hand corner of each figure corresponding to time equal to 0, and time increasing in each figure from top to bottom. The six jobs have processing times 2,3,4,4,5, and 6 respectively. The jobs with processing times 2,3, and 4 lie in the second rank. The jobs with processing times 4, 5, and 6 lie in the first rank. Any schedule in which each machine is assigned one job in the second rank, followed by one job in the first rank, is flowtime-optimal. Clearly, there are a total of or $(m!)^{\frac{n}{m}}$ or 36 such schedules. Each flowtime-optimal schedule has an average flowtime equal to 5.5. The first schedule in Figure 1 is a rectangular flowtime-optimal schedule obtained by assigning the jobs in the first rank largest-first after assigning the jobs in the second rank. To assign jobs in the first rank largest-first, the jobs are assigned in nonincreasing order of processing times to the earliest available machine. A machine is available if it does not already have a job in the second rank assigned to it. This largest-first assignment results in a makespan of 8. The second schedule in the figure is a flowtime-optimal schedule with a makespan of 9. The first schedule clearly has the lowest makespan among all flowtime-optimal schedules. For larger problem instances, optimal schedules cannot be found easily and approximation algorithms are used to construct good schedules.

Coffman and Sethi (1976a) propose two approximation algorithms for the FM problem. In LI scheduling, ranks are assigned in decreasing order of ranks, starting with the rank containing the m jobs with the smallest processing times. In LD scheduling, ranks are assigned in increasing order of ranks, starting with the rank containing the m jobs with the largest processing times. Jobs with the same rank are assigned largest-first onto distinct machines as they become available after executing the previous ranks. In LD scheduling,

the sequence thus obtained must be reversed and all jobs in the last rank must be set to the same starting time of zero to ensure that the schedule is flowtime-optimal. Coffman and Sethi show that, in LI scheduling, assigning only the jobs in rank 1 largest-first results in a makespan that is no more than 4/3 times the lowest makespan of any flowtime-optimal schedule. A complete LI schedule results in a worst-case bound of $(5m-4)/(4m-3)$.

No bound on the makespan ratio has been proved for the LD algorithm. Coffman and Sethi make the following conjecture for the makespan ratio for the LD algorithm.

Coffman-Sethi Conjecture: $t_{\mathbb{S}_{LD}}/t_{\mathbb{S}^*} \leq (5m-2)/(4m-1)$

where $t_{\mathbb{S}_{LD}}$ denotes the makespan of the LD schedule and $t_{\mathbb{S}^*}$ denotes the optimal makespan for any flowtime-optimal schedule.

To obtain a bound for the LI algorithm, Coffmann and Sethi use the notion of a profile to represent a schedule. The profile of a schedule after rank r is defined as the sorted set of completion times after rank r. They propose three measures for the comparison of any two schedules - the *variation*, the *relative distance*, and a measure denoted by δ_ℓ. For two schedules, the *variation* v_ℓ after rank ℓ refers to the gap between the largest completion time attained in either schedule and the smallest completion time attained in either schedule in rank ℓ. The *relative distance* x_ℓ between two profiles after rank ℓ is the largest difference between the corresponding elements of the two profiles after rank ℓ. For two schedules with profiles $a(\ell)$ and $b(\ell)$, if $x_\ell = b_j(\ell) - a_j(\ell)$, the measure δ_ℓ is defined as being equal to $\max[a_1(\ell), b_j(\ell)] - \min[a_j(\ell), b_m(\ell)]$.

Coffman and Sethi propose and prove three lemmas that provide bounds for these measures as a function of the rank. They subsequently use these lemmas to prove that merely assigning the rank with the largest processing times largest-first (after assigning the other ranks) ensures that the makespan is no more than $(4m-3)/(3m-2)$ times the lowest makespan among all flowtime-optimal schedules. They use a somewhat similar proof to show that complete LI scheduling (assigning all ranks largest-first, starting with the largest rank) results in a schedule with a makespan that is no more than $(5m-4)/(4m-3)$ times the smallest makespan among all flowtime-optimal schedules.

Coffman and Yannakakis (1984) study a more general version of this problem. They study the problem of permuting the elements within the columns of an $m \times n$ matrix so as to minimize its maximum row sum, where the term row sum refers to the sum of the elements in a row of the matrix. They propose a greedy approximation algorithm (Algorithm Row Sum or RS) for this problem. Their algorithm starts out by identifying the m largest elements in the matrix. The algorithm subsequently executes permutations

within each column so that, for $i = 1, \ldots, m$, the i_{th} smallest element is located in row i. Subsequently, the remaining elements of each column are sorted into ascending order using only those positions not already occupied by the m largest elements. The next step of the algorithm processes rows 2 through m in that order in the following manner. For row i, if the row sum for i is not larger than the maximum row sum for rows 1 through $i-1$, then no action is taken and RS proceeds to the next row. If the row sum for i is larger than the maximum row sum for rows 1 through $i-1$, and if row ℓ, $\ell < i$, has the smallest row sum among the first $i-1$ rows, then RS scans rows i and ℓ from right to left until corresponding elements in the same column of i and ℓ are found such that neither of them is among the m largest elements in the matrix and interchanging them would reduce the maximum row sum in the first i rows. If such a pair of elements is found, the two elements are interchanged and the process is reapplied to row i. This process continues until no such pair is found or the row sum for i no longer exceeds the maximum row sum for rows 1 through $i-1$. The algorithm is then applied to the remaining rows.

Coffman and Yannakakis show that the ratio of the maximum row sum obtained by applying the RS algorithm to the optimal maximum row sum cannot exceed 3/2 - 1/(2m) and that this bound is best possible. Notice that, for the special case in which, for $j = 1, \ldots, n$, every element of column j is less than or equal to every element of column $j+1$, the problem reduces to the makespan minimization problem for flowtime-optimal schedules. In this case, each row may be taken to refer to a machine and each column may be taken to refer to a rank.

Eck and Pinedo (1993) propose a new algorithm for the problem of finding the flowtime-optimal schedule with the smallest makespan. Their algorithm, the LPT* algorithm, is a modified version of Graham's LPT algorithm. Their algorithm requires the construction of a new problem instance by replacing every processing time in a rank by the difference between it and the smallest processing time in that rank. They consider the effect of applying the LPT rule to these differences for any two-machine problem instance. Arranging these differences according to LPT generates a sequence for the original problem that is, in general, not identical to the LPT sequence for the originial problem. This procedure is the LPT* algorithm. For the two-machine case, the authors obtain a worst-case bound of (28/27) on the makespan ratio for the LPT* algorithm.

The bin-packing and makespan minimization references cited previously show that it is possible to obtain extremely tight bounds (such as the 13/11 bound for the multifit algorithm) and PTASs for these problems. It is possible to construct algorithms for the FM problem that provide tighter bounds than the algorithms discussed above. Gupta and Ho (2001) build on the procedure developed by Ho and Wong for makespan minimization to develop three algorithms — an algorithm to find the optimal solution and two heuristic procedures — for the two-machine FM problem. Lin and Liao(2004) extend the procedures developed by Ho and Wong and by Gupta and Ho to construct a procedure to obtain the

optimal solution to the FM problem in $O(m!^{n/m})$ time. They use numerical experiments to show that the average runtime tends to be much less than is indicated by the exponential worst-case bound. Besides developing procedures to obtain an optimal solution, it might also be possible to construct PTASs for the FM problem. The focus of this thesis is *not* on the development of a PTAS or the development of more sophisticated heuristics that provide better approximation ratios. Instead, we focus on fast and simple algorithms like the LI and LD algorithms. The time taken by these algorithms is of the same order as the time required to sort a set of n processing times. These are thus extremely fast algorithms and are extremely simple to implement. We study the problem of proving a conjectured bound for such algorithms and focus specifically on attempting to prove Coffman and Sethi's conjectured bound for the LD algorithm.

The approach used to attempt to obtain a proof of Coffman and Sethi's conjectured bound is to identify properties of a minimal counterexample. In Chapter 2 of this thesis, the algorithms developed by Coffman and Sethi (LI and LD) and by Eck and Pinedo (LPT*) are formally defined. An improved version of LPT* and a new category of algorithms, the GAP algorithms (both of which were proposed by Huang and Tunçel (2004)), are also defined. In Chapter 3 of this thesis, worst-case examples are provided for the LI and LD algorithms. Note that the example provided for LD may not be a worst-case example if the Coffman-Sethi conjecture is false. In Chapter 4, the notion of minimality is defined and clarified. In Chapter 5, the three measures developed by Coffman and Sethi to study the LI scheduling algorithm are redefined for the LD scheduling algorithm. Lemmas are proposed that provide bounds for these measures as a function of the rank for LD scheduling. These lemmas and their proofs are similar to those proposed by Coffman and Sethi for LI scheduling. Subsequently, it is shown that, under certain assumptions, there always exists a minimal counterexample with integer processing times. It is also shown that, if the second rank is assigned largest-first, the ratio of makespan of the resulting schedule to the optimal makespan among all flowtime-optimal schedules cannot exceed 4/3. In Chapter 6, the two-machine, three-machine, and m-machine cases (for general m) are examined. A novel duality-based approach (proposed by Huang and Tunçel (2004)) is used to analyse the three-machine, three-rank case and the two-machine case. Huang and Tunçel (2004) used this approach to show that the conjecture holds for the three-machine, three-rank case. This approach is used to show that the conjecture holds for the two-machine case. Thus a hypothesised minimal counterexample to the conjecture must have at least four machines and three ranks, or three machines and four ranks. In Chapter 7, various properties are derived for a hypothesised minimal counterexample to the Coffman-Sethi conjecture with integer processing times. It is shown that a minimal counterexample to the conjecture has either three machines and no more than six ranks, or four or more machines and no more than five ranks. It is also shown that, if the conjecture is false, there exists a minimal counterexample to the conjecture with an LD makespan equal to $\mu_1 + \sum_{r=2}^{r=k} \lambda_r$.

Thus the key results obtained in this thesis are the following:
(i) The Coffman-Sethi conjecture holds for the two-machine case.
(ii) If the Coffman-Sethi conjecture is false, there exists a minimal counterexample with integer processing times and with either three machines and no more than six ranks, or four or more machines and no more than five ranks.
(iii) If the Coffman-Sethi conjecture is false, there exists a minimal counterexample with integer processing times and with an LD makespan equal to the sum of the smallest processing time in rank 1 and the largest processing time in each of the remaining $k-1$ ranks.

Chapter 2

Fast and Simple Algorithms for Problem FM

Let p_j denote the processing time of job j, where the jobs are numbered in nonincreasing order of processing times. Thus $p_j \geq p_{j+1}$. The set of jobs belonging to rank r are the following: $(r-1)m+1, (r-1)m+2, \cdots, (r-1)m+m$.

Any schedule in which all rank $r+1$ jobs are started before all rank r jobs (where $r = 1, 2, \cdots, (n/m)-1$) is said to satisfy the *rank restriction* or *rank constraint*.

Any schedule in which all rank $r+1$ jobs are started before all rank r jobs (where $r = 1, 2, \cdots, (n/m)-1$), there is no idle time between successive jobs assigned to the same machine, and all rank n/m jobs start at time 0, is termed a *flowtime-optimal schedule*. Let λ_r and μ_r denote the largest and smallest processing times respectively in rank r. Note that

$$\lambda_1 \geq \mu_1 \geq \lambda_2 \geq \mu_2 \geq \ldots \geq \lambda_{k-1} \geq \mu_{k-1} \geq \lambda_k \geq 0 \tag{2.1}$$

and for every $h, \ell \in \{1, 2, \ldots, k\}$ with $h < \ell$, we have

$$\sum_{r=h}^{\ell-1} \mu_r \geq \sum_{r=h+1}^{\ell} \lambda_r. \tag{2.2}$$

The profile of a schedule after rank r is defined as the sorted set of completion times on m machines after rank r. If the jobs in r ranks (out of a total of k) have been assigned to machines, and if the jobs in the remaining ranks have not yet been assigned to machines, the profile after rank r is termed the current profile. We let $a^{(r)} \in \mathbb{Z}^m : a_1^{(r)} \geq a_2^{(r)} \geq \cdots \geq a_m^{(r)}$ denote the current profile. Thus, $a_1^{(r)}$ denotes the largest completion time after rank r, $a_2^{(r)}$

denotes the second-largest completion time after rank r, ..., and $a_m^{(r)}$ denotes the smallest completion time after rank r. We may omit "(r)" when the rank is clear from the context.

The algorithms listed in this section differ in two ways — the set of processing times examined and the sequence in which ranks are examined. The first two algorithms listed in this chapter make no modification to the original set of processing times. The last four algorithms in this chapter examine a modified set of processing times. This modified set of processing times is constructed by subtracting the smallest processing time in each rank from every processing time in that rank. The sequence in which ranks are examined varies from one algorithm to the next. The set of ranks discussed in Chapter 1 is denoted by the labels $1, 2, \ldots, k$, where 1 denotes the rank containing the largest processing times and k denotes the rank containing the smallest processing times. The LI algorithm processes ranks in the order $k, k-1, \ldots, 1$ while the LD algorithm processes ranks in the order $1, 2, \ldots, k$. Each of the remaining algorithms - the LPT*, PIA, GAP-L_∞ and GAP-L_1 algorithms - examine ranks in a different order. We let $[r]$ denote the r_{th} rank examined by an algorithm. Thus [1] denotes the first rank examined by an algorithm, [2] denotes the second rank examined by the algorithm, ..., and $[k]$ denotes the last rank examined by the algorithm.

In each algorithm, the jobs in the next rank are assigned based on the Longest Processing Time criterion. Thus the longest remaining job is assigned to the machine that becomes available first. The general idea behind assigning jobs largest-first is to try to achieve a *rectangular* schedule. A rectangular schedule is one in which the completion time is the same on every machine. Clearly, if a rectangular schedule exists after the last rank, that schedule is an optimal schedule.

After an algorithm has processed the entire set of ranks, the modified processing times are replaced by the original processing times and the ranks are rearranged to ensure that the resulting schedule is flowtime-optimal. Note that this does not change the set of jobs assigned to a machine.

2.1 LI (Coffman and Sethi, 1976a)

The LI algorithm starts with rank k, the rank containing the m jobs with the lowest processing time, and works its way through ranks $k, k-1, \ldots, 2, 1$. The schedule generated by the algorithm is thus a flowtime-optimal schedule.

The algorithm works as follows: Schedule the ranks in the following order:

$$k, k-1, \ldots, 2, 1.$$

Let $a \in \mathbb{Z}^m$:

$$a_1 \geq a_2 \geq \cdots \geq a_m$$

denote the current profile. Schedule the jobs in the next rank so that the largest processing time is matched with a_m, second largest with a_{m-1}, . . . , and the smallest processing time is matched with a_1.

Of the six algorithms listed in this chapter, the LI algorithm is the only algorithm for which a tight bound (equal to $(5m-4)/(4m-3)$) has been proved in the literature. The algorithm moves from rank k to rank 1, with jobs within a rank being assigned based on Longest Processing Time.

2.2 LD (Coffman and Sethi, 1976a)

The LD algorithm starts with rank 1, the rank containing the m jobs with the largest processing time, and works its way through ranks $1, 2, \ldots, k-1, k$. The sequence of jobs on each machines is then reversed to make it a flowtime-optimal schedule.

The algorithm works as follows: Schedule the ranks in the following order:

$$1, 2, \ldots, k-1, k.$$

Let $a \in \mathbb{Z}^m$:

$$a_1 \geq a_2 \geq \cdots \geq a_m$$

denote the current profile. Schedule the jobs in the next rank so that the largest processing time is matched with a_m, second largest with a_{m-1}, . . . , and the smallest processing time is matched with a_1. After all the jobs are scheduled, reverse the schedule and left-justify it.

Like the LI algorithm, the LD algorithm tries to achieve a rectangular schedule by assigning jobs within a rank based on Longest Processing Time. The only difference is that the algorithm is applied in increasing order of ranks, starting with rank 1 and ending with rank k. Coffman and Sethi (1976a) proposed but did not prove a bound for the LD algorithm. The conjectured LD bound is slightly smaller than the LI bound. This is not surprising because the LD algorithm starts with the ranks that make the greatest contribution to makespan.

2.3 LPT* (Eck and Pinedo, 1993)

The smallest completion time on any machine after all jobs have been assigned cannot be less than the sum of the smallest processing times in each rank. Therefore the sum of the smallest processing times in each rank represents a lower bound on the makespan. The

LPT* algorithm seeks to minimize the extent to which the makespan exceeds this lower bound.

Recall that λ_r and μ_r denote the largest and the smallest processing time respectively in rank r. We define $\tilde{\lambda}_r := \lambda_r - \mu_r$ for $r \in \{1, 2, \ldots, k\}$. Recall that, if jobs are arranged in nonincreasing order of processing times, the jobs in rank ℓ are the following:
$(\ell - 1)m + 1, (\ell - 1)m + 2, \ldots, (\ell - 1)m + m$.
If job j' is in rank ℓ, we set the processing time of job j' to be equal to $p_{j'} - \min_{j \in \text{rank}(\ell)} \{p_j\}$.

For the new set of processing times, let $a \in \mathbb{Z}^m$:

$$a_1 \geq a_2 \geq \cdots \geq a_m$$

denote the current profile.

Schedule the ranks in the order

$$[1], [2], \ldots, [k-1], [k],$$

where

$$\tilde{\lambda}_{[1]} \geq \tilde{\lambda}_{[2]} \geq \cdots \geq \tilde{\lambda}_{[k]}.$$

Schedule the jobs in the next rank so that the largest modified processing time is matched with a_m, second largest with a_{m-1}, . . . , and the smallest processing time is matched with a_1. After all the jobs are scheduled, rearrange the ranks to ensure that the schedule is flowtime-optimal.

The LPT* algorithm recognizes the fact that the sum of the smallest processing times in each rank represents a constant component of the makespan and a lower bound on the makespan. The ranks are sorted based on the difference between the largest and the smallest processing time in each rank, with rank [1] corresponding to the largest difference and rank $[k]$ corresponding to the smallest difference. The algorithm is applied to the residual processing times (the difference between the original processing time of a job and the smallest processing time in that rank) rather than to the original processing times. The set of jobs in each rank are assigned based on Longest Processing Time being applied to the residual processing times, with the algorithm being applied starting with rank [1] and ending with rank $[k]$. After all the jobs are scheduled, the schedule is made flowtime-optimal by rearranging the ranks.

There are two key elements of weakness in the LPT* algorithm: (i) The ranks are ordered based on the difference between the largest and the smallest processing times in each rank. Thus information about all other processing times in each rank is not used to order the ranks. It is quite possible that rank $[r]$ has a profile that is almost flat, except for one very small or very large processing time, while rank $[h]$ (for some value of $h > r$) has a very uneven profile. (ii) The algorithm does not suggest a way of breaking ties for

ranks with the same value of $\tilde{\lambda}_{[r]}$. For a problem instance with $r = 1, 2, \ldots, \ell$ ranks, with every rank having the same value of $\tilde{\lambda}_{[r]}$, there could exist $\ell!$ LPT* schedules. Different ways of breaking ties could result in very different values of the makespan.

2.4 Pairwise Improvement Algorithm (PIA) (Huang and Tunçel, 2004)

Here we attach to LPT* an iterative, local improvement routine. If $m = 2$, we apply LPT* and stop. If $m \geq 3$, we apply LPT*. Then we find the machines i and i' in the LPT* schedule with the largest total processing time and the smallest total processing time respectively. We apply LPT* to the two-machine subproblem given by the jobs on the machines i and i'. Then we look for the machines with the largest and smallest processing times again (breaking the ties appropriately). If we can't find a new pair we stop. The ranks are then rearranged to ensure that the schedule is flowtime-optimal.

The motivation behind this improvement routine is to take advantage of the theoretical bound 28/27 proven for the LPT* algorithm for the two-machine problem (Eck and Pinedo, 1993). The LPT* algorithm provides a near-optimum solution for the two-machine problem, therefore it seems likely that applying LPT* to a two-machine subproblem will result in a significant improvement to the overall makespan. At any stage of the Pairwise Improvement Algorithm, if there exist m' machines, such that each of those machines has a total processing time equal to the largest total processing time, LPT* must be applied to m' two-machine subproblems in succession before any reduction in the makespan is achieved. Thus this algorithm takes more time than the LD, LI, and LPT* algorithms, but could provide a better solution. The algorithm may never converge and an upper bound must, therefore, be placed on the number of iterations.

2.5 GAP Algorithms (Huang and Tunçel, 2004)

As in the LPT* algorithm, if job j' is in rank ℓ, we set the processing time of job j' to be equal to $p_{j'} - \min_{j \in rank\ell} p_j$.

However, we do not sort the ranks based on values of $\tilde{\lambda}_{[r]}$. Instead, we try to utilize more detailed information within each rank. We define the *maximum adjacency gap of rank* ℓ as

$$\sigma_\ell := \max_{j \in \{(\ell-1)m+1, (\ell-1)m+2, \ldots, \ell m-1\}} \{p_j - p_{j+1}\},$$

where $\ell \in \{1, 2, \ldots, k\}$. Also, we define

$$\bar{\sigma}_\ell := \sum_{j=2}^{m} \left(p_{(\ell-1)m+1} - p_{(\ell-1)m+j} \right) = m\lambda_\ell - \sum_{j \in \text{rank}(\ell)} p_j,$$

where $\ell \in \{1, 2, \ldots, k\}$.

While LPT* uses the difference between the largest and the smallest processing time in a rank to measure unevenness of a rank, the GAP algorithms use the differences between adjacent members of an ordered set of processing times in a rank as measures of the unevenness of a rank, with these differences being used in different ways to sort ranks in the two GAP algorithms. The GAP algorithms assign the set of modified processing times within a rank based on Longest Processing Time, with the algorithm being applied starting with the most uneven rank. The GAP algorithms clearly use more detailed information than the LPT* algorithm and therefore might produce schedules with lower makespans.

2.5.1 GAP-L_∞ (Huang and Tunçel, 2004)

Use the same set of modified processing times as in the algorithm LPT*; but use the following ordering of ranks. Schedule the ranks from last to first in the order of ranks

$$[1], [2], \ldots, [k-1], [k],$$

where

$$\sigma_{[1]} \geq \sigma_{[2]} \geq \cdots \geq \sigma_{[k]}.$$

Once the jobs are assigned to the machines, reorder the ranks so that the final schedule is a flowtime-optimal schedule.

The measure of unevenness of a rank for the GAP-L_∞ algorithm is the largest difference between adjacent members of an ordered set of processing times in the rank.

2.5.2 GAP-L_1 (Huang and Tunçel, 2004)

Use the same set of modified processing times as in the algorithm LPT*; but use the following ordering of ranks. Schedule the ranks from last to first in the order of ranks

$$[1], [2], \ldots, [k-1], [k].$$

$$\bar{\sigma}_{[1]} \geq \bar{\sigma}_{[2]} \geq \cdots \geq \bar{\sigma}_{[k]}.$$

Once the jobs are assigned to the machines, reorder the ranks so that the final schedule is a flowtime-optimal schedule.

The measure of unevenness of a rank for the GAP-L_1 algorithm is the sum of the differences between the processing times of adjacent jobs in the rank.

Chapter 3

Worst-Case Examples

Worst-case examples have been proposed for the LI and LD algorithms. These worst-case examples will be discussed below.

For problem instances with a large number of ranks, a small number of ranks (consisting of the jobs with the largest processing times) will contribute a large proportion of the total completion time on each machine. It seems likely that, for such problem instances, removing the rank with the smallest processing times will result in only a small change in the optimal makespan and in the makespan of the schedules resulting from the application of these algorithms. Therefore, one may hypothesize that, for these algorithms, a worst-case example will consist of a relatively small number of ranks. On the other hand, a worst-case example must consist of more than two ranks. A worst-case example cannot have only one rank because, for a single rank, there exists only one possible value for the makespan. For two ranks, every algorithm listed in the previous chapter produces an optimal solution. Thus a worst-case example must consist of at least three ranks.

Further, the solutions produced by each of these algorithms are profile-dependent. The algorithms lose their effectiveness when they are applied to a fully rectangular schedule. One may hypothesize that, in each case, a worst-case example is one in which the algorithm results in a rectangular schedule after rank $k-1$, where k is the total number of ranks, while the optimal schedule is one which is rectangular after rank k.

It will be shown in subsequent chapters of this paper that, for the LI and LD algorithms, there will always exist a worst-case problem instance with integer processing times and with a rectangular optimal schedule. Each of the worst-case problem instances proposed below has three ranks, integer processing times, a rectangular schedule produced by the algorithm after the second rank, and a rectangular optimal schedule after the third rank. Note that the worst-case example for the LD algorithm is a conjectured worst-case example.

3.1 LI

The next example is presented in Coffman and Sethi (1976a). For $m \geq 2$, let $n := 3m$ and consider

$$p_j := \begin{cases} (j-1) & \text{for } j \in \{1, 2, \ldots, m\} \\ (j-2) & \text{for } j \in \{m+1, m+2, \ldots, 2m\} \\ (2m-2) & \text{for } j \in \{2m+1, 2m+2, \ldots, 3m-1\} \\ (3m-2) & \text{for } j = 3m. \end{cases}$$

The ratio of the objective value of the LI schedule to the optimal objective value is $\frac{5m-4}{4m-3}$.

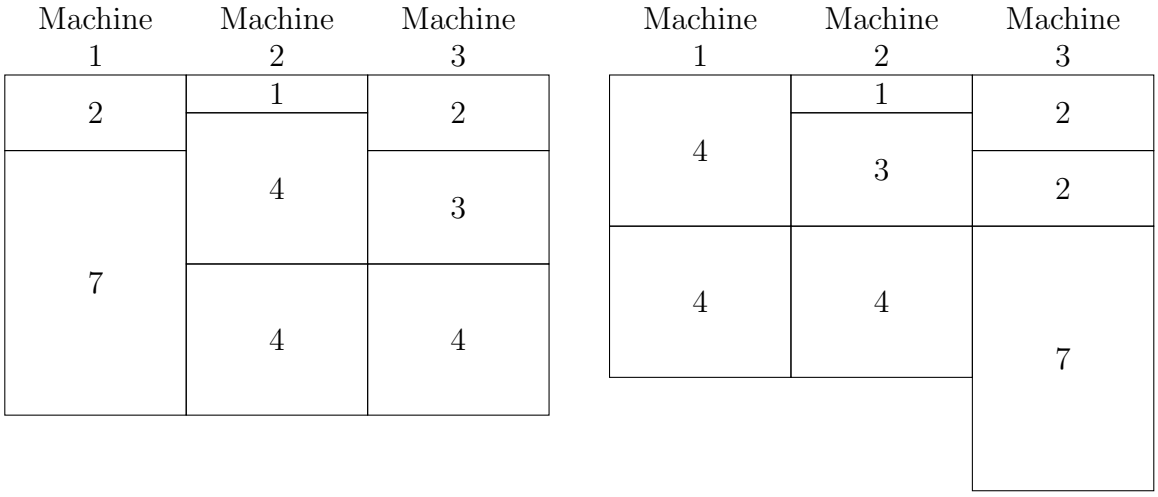

Figure 2: The optimal schedule and the LI schedule for a worst-case problem instance

Figure 2 consists of two figures. Note that the vertical axis denotes time, with the upper left-hand corner of each figure corresponding to time equal to 0, and time increasing in each figure from top to bottom. The figure on the left shows an optimal schedule for an instance of the FM problem with three machines and nine jobs. The nine jobs have processing times 0,1,2,2,3,4,4,4, and 7 respectively. The figure on the right shows the LI schedule for the same problem instance. The LI schedule is rectangular after rank 2, while the optimal schedule is rectangular after rank 1. Both schedules are flowtime-optimal schedules. One of the nine jobs has a processing time equal to zero. In both figures, the job with zero processing time is the first job assigned to machine 1. A sufficient number of jobs with zero processing time must be included in any problem instance to ensure that the total number of jobs is an integer multiple of the number of machines. If these jobs with zero processing time are not included in the problem instance, the set of jobs allocated to each rank would be different. This could, in general, result in schedules that are not flowtime-optimal.

3.2 LD

The following example is presented in Huang and Tunçel (2004).

For $m \geq 2$, let $n := 3m$ and consider

$$p_j := \begin{cases} 0 & \text{for } j \in \{1, 2, \ldots, m-1\} \\ m & \text{for } j = m \\ (j-1) & \text{for } j \in \{m+1, m+2, \ldots, 2m\} \\ (j-2) & \text{for } j \in \{2m+1, 2m+2, \ldots, 3m\}. \end{cases}$$

It is easy to show that the ratio of the objective value of the LD schedule to the optimal objective value is $\frac{5m-2}{4m-1}$.

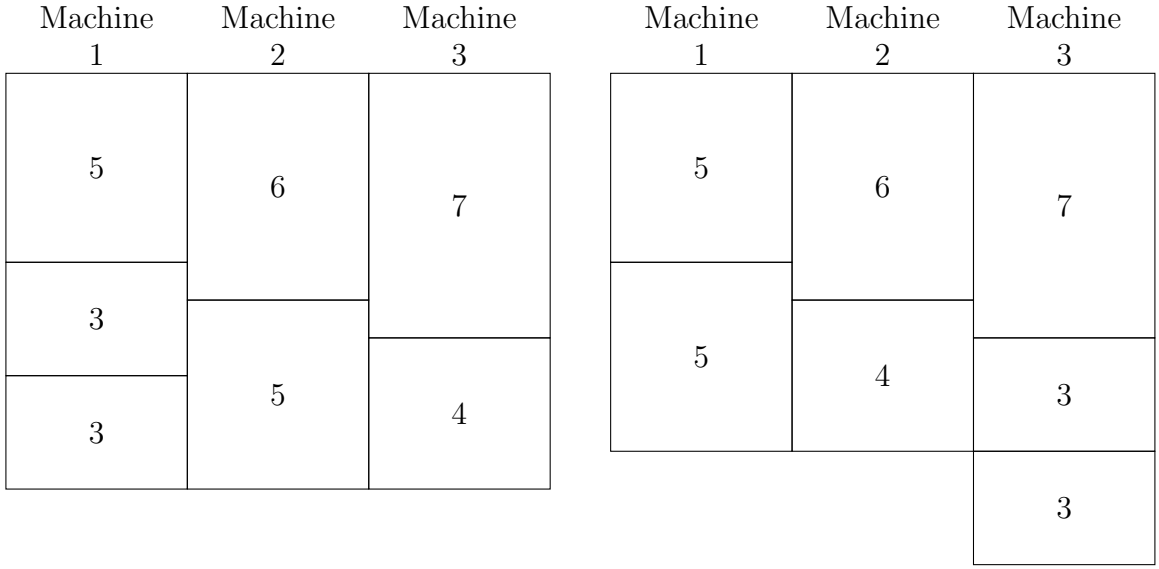

Figure 3: The optimal schedule and the LD schedule for a worst-case problem instance

Figure 3 consists of two figures. Note that the vertical axis denotes time, with the upper left-hand corner of each figure corresponding to time equal to 0, and time increasing in each figure from top to bottom. The figure on the left shows an optimal schedule for an instance of the FM problem with three machines and nine jobs. The nine jobs have processing times 0,0,3,3,4,5,5,6, and 7 respectively. The figure on the right shows the LD schedule for the same problem instance. The sequence of jobs assigned to each machine must be reversed to obtain flowtime-optimal schedules. This method of representing the schedules serves to highlight the fact that the LD algorithm starts with the jobs in the first

rank and processes jobs in increasng order of ranks. This method of representation also shows that the LD schedule is rectangular after rank 2. The optimal schedule is rectangular after rank 3. Two of the nine jobs have a processing time equal to zero. A sufficient number of jobs with zero processing time must be included in any problem instance to ensure that the total number of jobs is an integer multiple of the number of machines.

Chapter 4

A definition of minimality

Our general approach in this current attempt to obtain a proof of the Coffman-Sethi conjecture is to obtain a set of properties for a minimal counterexample to the conjecture. These properties are then used either to construct a counterexample or to prove that no such counterexample exists.

Observation: If an instance of the FM problem with m machines and k ranks has fewer than mk jobs, increasing the number of jobs to mk by including up to $m-1$ jobs with zero processing time does not result in any change in the total flowtime or the optimal makespan or the makespan of an LD schedule, though it changes the set of jobs allocated to each rank.

From this observation, it follows that, for the purpose of proving conjectures regarding worst-case makespan ratios for algorithms for the problem FM, only problem instances with mk jobs need to be considered. Jobs with zero processing time are unique in the sense that they can be assigned to any position on any machine without altering the total flowtime. For example, for a problem instance with 3 machines and 27 jobs, each with a processing time greater than zero, the smallest job must be the third job performed on either the first, second, or third machine. If the smallest job has a processing time equal to zero, it could be the first, second, or third job performed on the first, second, or third machine. Focusing on problem instances with mk jobs leads to a standardized method of allocating jobs to ranks. This makes it easier to analyze the performance of algorithms for the FM problem. Further, as shown later in this thesis, a worst-case problem instance for the LD algorithm always includes at least one job with zero processing time.

We define the ordered set of processing times P for a scheduling problem instance with m machines and k ranks to consist of elements equal to the processing times of these mk jobs arranged in nonincreasing order. We let $P(j)$ refer to the j_{th} element of P. Thus, $P(j) \geq P(j+1)$ for $j = 1, \ldots, mk-1$.

Consider two scheduling problem instances, each with m machines and k ranks. If P_1 is the set of processing times for the first problem instance and P_2 is the set of processing times for the second problem instance, we define P_2 to be $< P_1$ if and only if: (i) for some j', $1 \leq j' \leq mk$, $P_2(j') < P_1(j')$, and (ii) for all j, $1 \leq j \leq mk$, $P_2(j) \leq P_1(j)$. Thus the sets of processing times are lexicographically ordered.

For the problem of characterizing a minimal counterexample to the Coffman-Sethi conjecture, minimality will be defined based on the following criteria: the number of machines, the number of ranks, and the set of processing times. A minimal counterexample is defined to be a counterexample with k ranks, m machines, and a set of processing times P_1 for which there does not exist another counterexample with one of the following:
(i) number of ranks $< k$
(ii) number of ranks $= k$ and number of machines $< m$
(iii) k ranks, m machines and a set of processing times $P_2 < P_1$.

Thus, the notion of minimality is based on the number of ranks and the number of machines (in that order). For a given number of ranks and a given number of machines, minimality is based on the set of processing times.

This thesis proposes a set of techniques to characterise a hypothesised minimal counterexample and to reduce it to a relatively small size. The approach discussed in this thesis may be useful in the characterisation of minimal counterexamples to conjectured bounds for algorithms for some other constrained optimization problems. The general approach is to assume the existence of a minimal counterexample. We then subtract a vector from the vector of hypothesised parameters, subject to the condition that none of the constraints in the problem are violated. This will produce a new problem instance. The original counterexample was assumed to be a *minimal* counterexample, therefore the new problem instance must satisfy the conjecture. This may allow us to obtain a property, i.e. a constraint or several constraints that are satisfied by the original counterexample. By repeating this process and by suitable choices of vectors, we obtain a set of constraints (properties) and keep reducing the size of the minimal counterexample. For certain problems and conjectures, the size of the minimal counterexample may be reduced sufficiently to make it possible to analyse all possible cases and thus to either prove or disprove the conjecture. After reducing the size of the problem, a novel approach (obtained from Huang and Tunçel (2004)) inspired by the notion of LP duality may be used. For given values of m, the number of machines, and/or k, the number of ranks, this approach treats the parameters of problem FM (job processing times) as variables. The problem examined is that of determining the values of the processing times that result in the LD makespan being maximised for a given value of the optimal makespan. All possible relationships between processing times (subject to the rank restriction) are considered in the form of various cases. Thus, an exhaustive set of linear programs, each corresponding to a different case, is set up and solved. The solution to each linear program is checked to determine if

it violates the conjecture. In Chapter 6, this approach is used to show that the Coffman-Sethi conjecture is satisfied when $m = 2$, and when $m = 3$ and $k = 3$. After a reduction in the size of the problem, this duality-based approach may, in principle, be used to prove or disprove the existence of a counterexample to a conjectured bound on the makespan ratio for any algorithm for problem FM and potentially for other scheduling problems.

Later in this thesis, it will be shown that, if the Coffman-Sethi conjecture is false, there must exist a counterexample with integer processing times. Further, if the conjecture is false, there must exist a counterexample with integer processing times and a rectangular optimal schedule. Therefore, we define four types of problem instances, counterexamples and minimal counterexamples to the Coffman-Sethi conjecture. A problem instance of Type A is one that is not required to have either integer processing times or a rectangular optimal schedule. A problem instance of Type I is one with integer processing times. A problem instance of Type R is one with a rectangular optimal schedule. A problem instance of Type IR is one with integer processing times and a rectangular optimal schedule. A counterexample of a particular type (A, I, R, or IR) is a problem instance of that type that violates the Coffman-Sethi conjecture. A minimal counterexample of a particular type is a counterexample of that type for which there does not exist a smaller counterexample (based on the notion of minimality defined in this chapter) of the same type. It will be shown later in this thesis that, if the Coffman-Sethi conjecture is false, there exists a minimal counterexample of Type IR.

Chapter 5

Properties of flowtime-optimal schedules

Before examining the correctness of any particular conjecture, it might be useful to obtain general properties of flowtime-optimal schedules. Coffman and Sethi (1976) derive a set of properties that are appropriate for algorithms that examine ranks in decreasing order. In the first part of this chapter, similar properties (*Lemma* 5.1 and *Lemma* 5.2) are derived for algorithms (such as the LD algorithm) that examine ranks in increasing order. The next two lemmas show that, under certain conditions, it is appropriate to focus entirely on hypothesised counterexamples with integer-valued parameters. We subsequently examine a very simple algorithm for this problem, one that merely requires a largest-first assignment of jobs in the second rank. We show that this simple algorithm, the LD_0 algorithm has a makespan ratio with a bound equal to 4/3. This is also a bound for the LD algorithm. This analysis serves to demonstrate the fact that the use of integer-valued parameters can serve as a useful tool in developing and proving bounds for algorithms for this problem. The last three lemmas in this chapter begin the process of examining the LD algorithm. This process continues in the next two chapters of this thesis.

The choice of schedule determines the set of completion times (or the profile) for each rank associated with that schedule.

The profile after rank ℓ of Schedule $\mathbb{S}_a$ is denoted by $a(\ell) = (a_1(\ell), \cdots, a_m(\ell))$, where $a_i(\ell) \geq a_{i+1}(\ell)$ for $i = 1, 2, \cdots, m-1$. Note that $a_i(\ell)$ is the i_{th} largest completion time after rank ℓ.

For two schedules, the gap between the largest completion time attained in either schedule and the smallest completion time attained in either schedule for a given rank provides a measure of the lack of 'flatness' or lack of 'rectangularity' in the two profiles after that rank. We use the term *variation* between profiles to denote this lack of rectangularity

for a pair of profiles. Note that, if the gap between the largest and smallest completion time is small for two schedules after a given rank, both schedules are close to being rectangular after that rank. Conversely, if two schedules are approximately rectangular after a given rank, the largest completion time after that rank will be approximately equal for those two schedules because they must have the same average completion time per machine after that rank. Thus, two schedules that are approximately rectangular after a given rank will have a small gap between the largest and the smallest completion time for either schedule after that rank. Note that rectangularity of two schedules is a sufficient but not a necessary condition for the two schedules to have makespans that are roughly equal. Absence of rectangularity is a necessary but not a sufficient condition for two schedules to have different makespans.

Let $a(\ell)$ and $b(\ell)$ be profiles of schedules $\mathbb{S}_a$ and $\mathbb{S}_b$.

The variation v_ℓ in $a(\ell)$ and $b(\ell)$ is given by $v_\ell = \max\left[b_1(\ell), a_1(\ell)\right] - \min\left[b_m(\ell), a_m(\ell)\right]$

A large variation between two profiles indicates an absence of rectangularity for the two profiles but does not guarantee that the two profiles are very different from each other. The amount of information contained in the 'variation' measure is limited due to the fact that it depends only on the largest and the smallest completion times in the two schedules. This measure does not make any distinction between the processing times of the first schedule and the processing times of the second schedule. For a set of m machines, the variation can be computed by considering a set of $2m$ processing times (for the two schedules) and finding the range of values for this set of $2m$ processing times.

Therefore, to obtain a deeper understanding of the dissimilarity between the two schedules, we need more information than that provided by the variation measure. We need a second measure that makes a distinction between the processing times for the two schedules and considers the two sets of m processing times separately (without merging them into one set of $2m$ elements). This second measure should also ideally compare corresponding elements of the profiles associated with the two schedules, rather than merely looking at the largest and the smallest element.

This second measure is the 'relative distance' between two profiles. The relative distance is determined by calculating the differences in the corresponding elements of the two profiles after a given rank and by finding the largest value. Thus, we compute the difference between the i^{th} largest completion time in schedule $\mathbb{S}_b$ after rank ℓ and the i^{th} largest completion time in schedule $\mathbb{S}_a$ after rank ℓ.

The relative distance x_ℓ between profiles $a(\ell)$ and $b(\ell)$ is given by

$x_\ell = \max_{1 \leq i \leq m} |b_i(\ell) - a_i(\ell)|$

For example, consider a problem with three machines labelled Machine 1, Machine 2 and Machine 3, and nine jobs with processing times 9, 8, 7, 6, 5, 4, 3, 2, and 1 respectively.

In schedule $\mathbb{S}_a$, jobs with processing times 1, 5, and 7 are assigned to Machine 1, jobs with processing times 2, 4, and 8 are assigned to Machine 2, and jobs with processing times 3, 6, and 9 are assigned to Machine 3. In schedule $\mathbb{S}_b$, jobs with processing times 1, 4, and 7 are assigned to Machine 1, jobs with processing times 2, 6, and 8 are assigned to Machine 2, and jobs with processing times 3, 5, and 9 are assigned to Machine 3. Thus $a(2) = (9, 6, 6)$ and $b(2) = (8, 8, 5)$. Therefore the variation after the second rank $v_2 = \max[8, 9] - \min[5, 6] = 4$. The relative distance after the second rank $x_2 = \max[|8 - 9|, |8 - 6|, |5 - 6|] = 2$.

Note that two identical but very non-rectangular profiles will have a large variation but would have a relative distance equal to zero. A positive relative distance between two profiles for a given rank indicates that the corresponding elements in at least one position in the profiles (e.g. the 7^{th} largest element in either profile) are not identical. Due to the fact that the cumulative completion time across all machines after a given rank is the same for every schedule, the fact that the corresponding elements in one position are not identical implies that there exists at least one other position for which the corresponding elements in the two profiles are not identical.

The following lemma establishes relationships between the relative distance, the variation and the processing times. The left-hand side inequality in part (a) of the lemma establishes a relationship between the variation in two profiles after rank ℓ and the relative distance between them after rank $\ell + 1$. This relationship is based on the following underlying insight. If two flowtime optimal schedules have a low variation in profiles after rank ℓ, the two schedules are close to being rectangular after rank ℓ. Every rectangular schedule after rank ℓ has the same average completion time. Therefore, if the two schedules are close to being rectangular, they must be very similar after rank ℓ, and the relative distance between the two profiles after rank $\ell + 1$ cannot be large because it results mainly from the differing assignments of jobs to machines in rank $\ell + 1$.

Let λ_i denote the largest processing time in rank i and let μ_i denote the smallest processing time in rank i.

Lemma 5.1. *For a given problem instance, let $\mathbb{S}_a$ and $\mathbb{S}_b$ be two flowtime-optimal schedules with k ranks. Let $a(\ell)$ and $b(\ell)$ denote the profiles of $\mathbb{S}_a$ and $\mathbb{S}_b$ respectively after rank ℓ. Then:*
(a) For every $\ell \in \{1, 2, ..., k-1\}$, the variation v_ℓ between profiles $a(\ell)$ and $b(\ell)$ is less than or equal to the difference between the largest task processing time and the smallest task processing time assigned so far. The relative distance $x_{\ell+1}$ between profiles $a(\ell+1)$ and $b(\ell+1)$ is less than or equal to the variation in the previous rank v_ℓ. Thus,

$$x_{\ell+1} \leq v_\ell \leq \lambda_1 - \mu_\ell.$$

(b) For every $\ell \in \{1, 2, \ldots, k-1\}$,

$$x_\ell - x_h \leq \lambda_{h+1} - \mu_\ell \leq \min\left\{\mu_h - \mu_\ell, \lambda_{h+1} - \lambda_{\ell+1}\right\}.$$

Thus, the increase in relative distance from rank h to rank ℓ, where $h < \ell$, is less than or equal to the increase in the smallest task time from rank ℓ to rank h. It is also less than or equal to the increase in the largest task time from rank $\ell + 1$ to rank $h + 1$.

Proof. Proof of part (a):

The earliest and latest times at which a processor can become free after rank ℓ are given by $\sum_{r=1}^{\ell} \mu_r$ and $\sum_{r=1}^{\ell} \lambda_r$ respectively.

Hence

$$v_\ell = \max(b_1(\ell), a_1(\ell)) - \min(b_m(\ell), a_m(\ell)) \leq \left(\sum_{r=1}^{\ell} \lambda_r\right) - \left(\sum_{r=1}^{\ell} \mu_r\right) \leq \lambda_1 - \mu_\ell,$$

where the last inequality uses (2.1).

To prove the lower bound, we may assume $x_{\ell+1} = b_s(\ell + 1) - a_s(\ell + 1)$ for some $s \in 1, 2, \ldots, m$.

There are s jobs in schedule $\mathbb{S}_b$ that finish at or after $b_s(\ell + 1)$.

There are $m - s + 1$ jobs in schedule $\mathbb{S}_a$ that finish at or before $a_s(\ell + 1)$.

So, there is clearly at least one job j' that finishes at or before $a_s(\ell + 1)$ in $\mathbb{S}_a$ and at or after $b_s(\ell + 1)$ in $\mathbb{S}_b$.

Let $C_{j'}^{(a)}$ and $C_{j'}^{(b)}$ denote the completion times of job j' in schedules $\mathbb{S}_a$ and $\mathbb{S}_b$ respectively. Then, by the above observation,

$$C_{j'}^{(b)} - C_{j'}^{(a)} \geq b_s(\ell + 1) - a_s(\ell + 1) = x_{\ell+1}. \tag{5.1}$$

Moreover,

$$C_{j'}^{(a)} \geq a_m(\ell) + p_{j'} \tag{5.2}$$

$$C_{j'}^{(b)} \leq b_1(\ell) + p_{j'} \tag{5.3}$$

Therefore, we obtain

$$\begin{aligned} v_\ell &= \max\{b_1(\ell), a_1(\ell)\} - \min\{b_m(\ell), a_m(\ell)\} \\ &\geq b_1(\ell) - a_m(\ell) \\ &\geq C_{j'}^{(b)} - C_{j'}^{(a)} \text{ by (5.2) and (5.3)} \\ &\geq x_{\ell+1} \text{ by (5.1)} \end{aligned}$$

as claimed.

Proof of part (b):

Let $\ell > h$. Without loss of generality, we may assume $b_s(\ell) \geq a_s(\ell)$ and $x_\ell = b_s(\ell) - a_s(\ell)$.

$$\begin{aligned} \text{For every } i \in \{1, 2, \ldots, m\}, a_i(\ell) &\geq a_i(h) + \sum_{r=h+1}^{\ell} \mu_r \text{ , and} \\ b_i(\ell) &\leq b_i(h) + \sum_{r=h+1}^{\ell} \lambda_r. \end{aligned} \tag{5.4}$$

$$\begin{aligned} \text{Thus, } x_\ell - x_h &= (b_s(\ell) - a_s(\ell)) - \max_{1 \leq i \leq m} |b_i(h) - a_i(h)| \\ &\leq \max_{1 \leq i \leq m} |b_s(\ell) - b_s(h) - (a_i(\ell) - a_i(h))| \\ &\leq \max \left| \sum_{r=h+1}^{\ell} \lambda_r - \sum_{r=h+1}^{\ell} \mu_r \right| \\ &\leq \lambda_{h+1} - \mu_\ell \\ &\leq \min\{\mu_h - \mu_\ell, \lambda_{h+1} - \lambda_{\ell+1}\}, \end{aligned}$$

where the second inequality follows from (5.4) and the third inequality follows from (2.2).

□

The right-hand side inequality in part (a) of the above lemma develops an upper bound for the variation in two profiles after rank ℓ. The largest possible completion time for any schedule after rank ℓ is obtained by assigning the largest processing time for each rank to the same machine. On the other hand, the smallest completion time for any schedule after rank ℓ is obtained by assigning the smallest processing time for each rank to the same machine. Thus the difference between the largest completion time for any schedule after rank ℓ and the smallest completion time for any schedule after rank ℓ is bounded above by the sum of the differences between the largest and the smallest processing times for each rank (for ranks 1 to ℓ). However, the smallest processing time in any rank cannot be smaller than the largest processing time in the next rank.

$$\lambda_1 \geq \mu_1 \geq \lambda_2 \geq \mu_2 \geq \ldots \geq \lambda_{k-1} \geq \mu_{k-1} \geq \lambda_k \geq 0.$$

It follows that the sum of the differences is bounded above by the difference between the largest processing time in rank 1 and the smallest processing time in rank ℓ.

Part (b) of Lemma 5.1 is based on the following underlying intuition. If there is only a small reduction in the largest processing time from rank $h+1$ to rank $\ell+1$ (where $\ell > h$),

it follows that the processing times do not change much from ranks h to ℓ for any schedule and these processing times lie within a narrow range. A small relative distance between the profiles for two schedules $\mathbb{S}_a$ and $\mathbb{S}_b$ after rank h indicates that there exist only small differences between the corresponding elements of the two profiles after rank h. These differences will not undergo a large increase if we add a set of ranks ($h+1$ to ℓ) consisting of jobs with processing times that lie within a narrow range.

The key purpose served by parts (a) and (b) of Lemma 5.1 is to show that the largest contributions to the differences between the makespan for two flowtime-optimal schedules occurs in the earlier ranks, thus suggesting that the worst case ratio of makespan for a schedule obtained by examining the ranks in increasing order (from largest to smallest processing times) to the makespan for the optimal schedule is likely to occur for some small number of ranks. Thus this lemma suggests that the LD algorithm is likely to have a worst case for a small number of ranks.

Lemma 5.2. *Let ℓ be any rank, $1 \leq \ell \leq k$, and let s be such that $x_\ell = b_s(\ell) - a_s(\ell)$. Let $\delta_\ell = \max[a_1(\ell), b_s(\ell)] - \min[a_s(\ell), b_m(\ell)]$. Then $\delta_\ell \geq \frac{mx_\ell}{m-1}$.*

Proof. Let $\ell \in \{1, 2, \ldots, k\}$ and $s \in \{1, 2, \ldots, m\}$ be as above. Firstly, note that

$$\delta_\ell \geq a_1(\ell) - a_s(\ell) \geq a_i(\ell) - a_s(\ell), \forall i \in \{1, 2, \ldots, s-1\}, \tag{5.5}$$

$$\delta_\ell \geq b_s(\ell) - b_m(\ell) \geq b_s(\ell) - b_i(\ell), \forall i \in \{s+1, s+2, \ldots, m\}. \tag{5.6}$$

Secondly, we have

$$\begin{aligned} i < s \Rightarrow b_i(\ell) &\geq b_s(\ell) = a_s(\ell) + x_\ell, \\ i > s \Rightarrow a_i(\ell) &\leq a_s(\ell) = b_s(\ell) - x_\ell. \end{aligned} \tag{5.7}$$

Moreover,

$$\sum_{i=1}^{m} a_i(\ell) = \sum_{i=1}^{m} b_i(\ell). \tag{5.8}$$

Now, using the above facts, we obtain

$$\begin{aligned} x_\ell &= \sum_{i=1, i\neq s}^{m} (a_i(\ell) - b_i(\ell)) \\ &\leq \sum_{i=1}^{s-1} (a_i(\ell) - a_s(\ell) - x_\ell) + \sum_{i=s+1}^{m} (b_s(\ell) - x_\ell - b_i(\ell)) \\ &= \sum_{i=1}^{s-1} (a_i(\ell) - a_s(\ell)) + \sum_{i=s+1}^{m} (b_s(\ell) - b_i(\ell)) - (m-1)x_\ell \\ &\leq (s-1)\delta_\ell + (m-s)\delta_\ell - (m-1)x_\ell, \end{aligned} \tag{5.9}$$

where the first inequality uses (5.7) and the second inequality follows from (5.5) and (5.6). Therefore, $mx_\ell \leq (m-1)\delta_\ell$. □

Observation: Let $\mathbb{S}$ be a flowtime-optimal schedule that is constructed using the following two-step process: (i) Ranks are assigned in increasing order and the last rank (containing the smallest jobs) is assigned largest-first. (ii) The jobs assigned to each machine are then reversed to make the schedule flowtime-optimal. Note that the processing times in the last rank can be made arbitrarily small. Therefore the ratio of the makespan of $\mathbb{S}$ to the optimal makespan among all flowtime-optimal schedules could be arbitrarily close to this ratio for the schedule with the highest ratio of makespan to optimal makespan among all flowtime-optimal schedules.

Recall that a schedule in which all rank $r+1$ jobs are started before all rank r jobs (where $r = 1, 2, \cdots, (n/m) - 1$) is said to satisfy the *rank constraint*. It is evident that if, after subtracting a constant amount from each processing time in rank ℓ, the smallest processing time in rank ℓ is still greater than or equal to the largest processing time in rank $\ell + 1$, the assignment of jobs to ranks remains unchanged.

Claim 5.1. *For the LD algorithm for problem FM, construction of a new problem instance from an existing problem instance by subtracting $t_{constant}$ from the processing time of each job in a rank without violating the rank constraint will lead to a problem instance for which the LD algorithm yields a makespan equal to $t_{S_{LD}} - t_{constant}$.*

Proof. Let PO denote the original problem instance and let PR denote the new problem instance. The LD algorithm can, in general, produce multiple solutions with the same value of the maximum completion time after each rank. It is clear that, for any LD solution to PO, an LD solution to PR can be constructed by assigning each modified job to the same machine as the original job and, for every job that remains unchanged, by assigning the job to the same machine in PR to which it was assigned in PO. It is clear, using similar arguments, that for any LD solution to PR, an LD solution to PO can be constructed. It follows that the makespan of an LD solution to PR is equal to the makespan of the corresponding LD solution to $PO - t_{constant}$. □

Let OPT denote an algorithm that finds the optimal solution to any problem instance of the FM problem. (One example of such an algorithm is a complete enumeration procedure.)

Claim 5.2. *For the OPT algorithm for problem FM, construction of a new problem instance from an existing problem instance by subtracting $t_{constant}$ from the processing time of each job in a rank without violating the rank constraint will lead to a problem instance with an optimal makespan equal to $t_{\mathbb{S}^*} - t_{constant}$.*

Proof. Assume that this claim is invalid. Let PO denote the original problem instance and let PR denote the new problem instance. Let $t_{\mathbb{S}^*}$ denote the optimal makespan of PO. Let $t_{\mathbb{R}^*}$ denote the optimal makespan of PR. If every job in PR is assigned to the same slot to which the corresponding job is assigned in an optimal solution to PO, it is clear that the makespan of this solution for PR is equal to $t_{\mathbb{S}^*} - t_{constant}$. It follows that $t_{\mathbb{R}^*}$ is less than or equal to $t_{\mathbb{S}^*} - t_{constant}$. Assume that $t_{\mathbb{R}^*}$ is less than $t_{\mathbb{S}^*} - t_{constant}$. Now, assign every job in PO to the same slot to which the corresponding job is assigned in an optimal solution to PR. It is clear that the makespan of this solution for PO is equal to $t_{\mathbb{R}^*} + t_{constant}$. This is less than $(t_{\mathbb{S}^*} - t_{constant}) + t_{constant} = t_{\mathbb{S}^*}$. This violates the initial assumption that $t_{\mathbb{S}^*}$ was the optimal makespan of PO. Therefore the assumption that $t_{\mathbb{R}^*}$ is less than $t_{\mathbb{S}^*} - t_{constant}$ is invalid. It follows that $t_{\mathbb{R}^*}$ is equal to $t_{\mathbb{S}^*} - t_{constant}$. □

Corollary 5.1. *If $t_{constant} > 0$, construction of a new problem instance from an existing problem instance by subtracting $t_{constant}$ from the processing time of each job in a rank without violating the rank constraint will lead to an increase in the $t_{S_{LD}}/t_{S^*}$ ratio.*

Proof. This follows from *Claim* 5.1 and *Claim* 5.2. □

Most scheduling problems are considered in the context of the Turing machine model of computation and, as a result, the data are assumed to be drawn from the rationals. In our problem FM, this would mean that $p_j \in \mathbb{Q}, \forall j \in 1, 2, \ldots, n$. Below, we prove that, under certain assumptions, a counterexample to a conjectured makespan ratio with irrational processing times could exist only if there existed a counterexample with rational processing times.

In the following lemmas, the term continuous function is used to refer to a function that satisfies the epsilon-delta definition of continuity proposed by Weierstrass (1886).

Let $G \subseteq \mathbb{R}^n$. Consider functions of the form $h : G \to \mathbb{R}$. We define continuity for such functions as follows.

Definition 5.1. *h is* continuous *at $\overline{x} \in$ G if for every $\epsilon > 0$, there exists a $\delta > 0$ such that for all $x \in$ G, $\|x - \overline{x}\| < \delta \Rightarrow \|h(x) - h(\overline{x})\| < \epsilon$. Here $\|.\|$ refers to any norm on $\mathbb{R}^n$, unless otherwise specified. h is termed a* continuous function *if it is continuous at all points in* G.

The following result shows that $t_{\mathbb{S}^*}$ is a continuous function.

Let $p \in \mathbb{R}^n_+$ denote the set of processing times for an instance E of the problem FM. Let $p = (p_1, p_2, \ldots, p_n)$. Let t^*_p denote the value of $t_{\mathbb{S}^*}$ for E. Let $\overline{p} \in \mathbb{R}^n_+$ denote the set of processing times for an instance $\overline{E}$ of the problem FM. Let $t^*_{\overline{p}}$ denote the value of $t_{\mathbb{S}^*}$ for $\overline{E}$.

Claim 5.3. *The maximum value of* $\left|t_p^* - t_{\overline{p}}^*\right|$ *over* $\{\overline{p} \in \mathbb{R}_+^n : \|p - \overline{p}\|_\infty \leq \delta\}$ *is attained either by* $\overline{p} = p + \delta.e$ *or* $\overline{p} = (max(p_1 - \delta, 0), max(p_2 - \delta, 0), \ldots, max(p_n - \delta, 0))$*, where* $e \in \mathbb{R}_+^n$ *is the vector of all ones.*

Proof. Assume that the claim does not hold. Thus, the maximum value of $\left|t_p^* - t_{\overline{p}}^*\right|$ over $\{\overline{p} \in \mathbb{R}_+^n : \|p - \overline{p}\|_\infty \leq \delta\}$ is attained neither by $\overline{p} = p + \delta.e$ nor by $\overline{p} = (max(p_1 - \delta, 0), max(p_2 - \delta, 0), \ldots, max(p_n - \delta, 0))$. Therefore, there exists $p' \in \mathbb{R}_+^n$ such that the maximum value of $\left|t_p^* - t_{\overline{p}}^*\right|$ over $\{\overline{p} \in \mathbb{R}_+^n : \|p - \overline{p}\|_\infty \leq \delta\}$ is attained by $\overline{p} = p'$, where $p' = (p'_1, p'_2, \ldots, p'_n)$, and $\exists j_1, j_2$ such that $p'_{j_1} > max(p_{j_1} - \delta, 0)$ and $p'_{j_2} < p_{j_2} + \delta$. (Note that j_1 could be equal to j_2.)

Case 1: $t_p^* - t_{\overline{p}}^* \leq 0$

Consider the problem instance E' of the problem FM with m machines and the set of processing times p', and the problem instance E'' of the problem FM with m machines and the set of processing times $p + \delta.e$. Assign every job in E' to the same slot to which the corresponding job is assigned in the optimal solution to E''. Clearly, for flowtime-optimal schedules, the optimal makespan of E'' is greater than or equal to the makespan of this solution to E', and the makespan of this solution to E' is greater than or equal to the optimal makespan for E'. Therefore $t_p^* \leq t_{\overline{p}}^* \leq t_{E''}^*$, where $t_{E''}^*$ denotes the optimal makespan for E''. Therefore the maximum value of $\left|t_p^* - t_{\overline{p}}^*\right|$ over $\{\overline{p} \in \mathbb{R}_+^n : \|p - \overline{p}\|_\infty \leq \delta\}$ is attained by the set of processing times $p + \delta.e$. This contradicts the assumption that the claim does not hold.

Case 2: $t_p^* - t_{\overline{p}}^* > 0$

Consider the problem instance E' of the problem FM with m machines and the set of processing times p', and the problem instance E''' of the problem FM with m machines and the set of processing times $(max(p_1 - \delta, 0), max(p_2 - \delta, 0), \ldots, max(p_n - \delta, 0))$. Assign every job in E''' to the same slot to which the corresponding job is assigned in the optimal solution to E'. Clearly, for flowtime-optimal schedules, the makespan of this solution to E''' is less than or equal to the optimal makespan of E', and the makespan of this solution to E''' is greater than or equal to the optimal makespan for E'''. Therefore $t_p^* > t_{\overline{p}}^* \geq t_{E'''}^*$, where $t_{E'''}^*$ denotes the optimal makespan for E'''. Therefore the maximum value of $\left|t_p^* - t_{\overline{p}}^*\right|$ over $\{\overline{p} \in \mathbb{R}_+^n : \|p - \overline{p}\|_\infty \leq \delta\}$ is attained by the set of processing times $(max(p_1 - \delta, 0), max(p_2 - \delta, 0), \ldots, max(p_n - \delta, 0))$. This contradicts the assumption that the claim does not hold. □

Corollary 5.2. *The maximum value of* $\left|t_p^* - t_{\overline{p}}^*\right|$ *over* $\{\overline{p} \in \mathbb{R}_+^n : \|p - \overline{p}\|_\infty \leq \delta\}$ *is less than or equal to* $k.\delta$*.*

Proof. This follows from Claim 5.3 and the fact that every machine has either k jobs or $k - 1$ jobs assigned to it. □

Claim 5.4. *Let $t_{\mathbb{S}^*}$ denote the optimal objective value for any instance of the problem FM. Then $t_{\mathbb{S}^*}$ is a continuous function at every point in $\mathbb{R}_+^n$.*

Proof. Let $p \in \mathbb{R}_+^n$ denote the set of processing times for an instance E of the problem FM. Let t_p^* denote the value of $t_{\mathbb{S}^*}$ for E. Let k denote the number of ranks in E. Pick any value of $\epsilon > 0$. Set $\delta = \epsilon/k$. Let $\overline{p} \in \mathbb{R}_+^n$ denote the set of processing times for an instance $\overline{E}$ of the problem FM, where $\|p - \overline{p}\|_\infty < \delta$. Let $t_{\overline{p}}^*$ denote the value of $t_{\mathbb{S}^*}$ for $\overline{E}$. From Corollary 5.2, it follows that $\left|t_p^* - t_{\overline{p}}^*\right| < k.\delta = \epsilon$. This proves that $t_{\mathbb{S}^*}$ is a continuous function. □

The above claim is used in the proof of the following lemma. The lemma shows that, to prove or disprove a conjectured makespan ratio for the algorithms listed in Chapter 2 of this thesis, we can focus on problem instances in which all processing times are integers. This lemma will be needed to prove some of the results in this thesis.

Let k denote the number of ranks in an instance of problem FM, where FM refers to the problem of selecting the schedule with the smallest makespan among the class of all flowtime-optimal schedules.

An instance of the FM problem is defined by an integer m denoting the number of machines and a set of processing times $(p_1, p_2, \ldots, p_n)$. Therefore, the domain of FM is a subset of $\mathbb{Z}_+ \times \mathbb{R}_+^n$. Let $FM(m, n)$ denote the FM problem with a fixed value of m and a fixed value of n. An instance of $FM(m, n)$ is defined by $(m, n; p_1, p_2, \ldots, p_n)$. It follows that the domain of $FM(m, n)$ is a subset of $\mathbb{Z}_+^2 \times \mathbb{R}_+^n$.

Lemma 5.3. *For every algorithm ALG' for problem FM(m,n) that produces a feasible solution $\mathbb{S}_{ALG'}$ whose makespan $t_{\mathbb{S}_{ALG'}} : \mathrm{G} \to \mathbb{R}_+$ is a continuous function at every point in* G *and for a conjecture which states that the makespan is no greater than* $\mathrm{f}(m).t_{\mathbb{S}^*}$*, where* $\mathrm{f} : \mathbb{Z}_+ \to \mathbb{R}_+$*, the following must hold: If there exists a counterexample of Type A to a conjectured $t_{\mathbb{S}_{ALG'}}/t_{\mathbb{S}^*}$ ratio, then there exists a counterexample of Type I.*

Proof. Suppose there exists a counterexample to the conjectured ratio $t_{\mathbb{S}_{ALG'}}/t_{\mathbb{S}^*}$. For a contradiction, suppose that every such counterexample has one or more irrational processing times. Pick such a counterexample, call it EI. Let m denote the number of machines and let k denote the number of ranks in EI. Then, we may assume that the number of jobs is $n = m.k$.

Then, $t_{S_{ALG'}}$ associated with EI is greater than the conjectured upper bound on $t_{\mathbb{S}_{ALG'}}$ associated with EI and there exists $\epsilon > 0$ such that

$$t_{\mathbb{S}_{ALG'}} > f(m)t_{\mathbb{S}^*} + \epsilon.$$

Let p denote the vector of job processing times for EI. Let $t_{\mathbb{S}_{ALG'}}$ denote the makespan associated with the schedule for EI generated by ALG'. Let $t_{\mathbb{S}^*}$ denote the makespan of the optimal schedule associated with EI.

For $\overline{p} \in \mathbb{Q}_+^n$ and m machines, let $t_{\overline{\mathbb{S}}_{ALG'}}$ denote the makespan of the schedule generated by ALG' and let $t_{\overline{\mathbb{S}^*}}$ denote the makespan of the optimal schedule for this instance. Since $t_{\mathbb{S}_{ALG'}}$ is continuous, there exists $\delta_1 > 0$ such that for every $\overline{p} \in \mathbb{Q}_+^n$ with $\|p - \overline{p}\| < \delta_1$ we have

$$\left| t_{\overline{\mathbb{S}}_{ALG'}} - t_{\mathbb{S}_{ALG'}} \right| < \frac{\epsilon}{2}. \tag{5.10}$$

Since $t_{\mathbb{S}^*}$ is continuous by *Claim* 5.4, there exists $\delta_2 > 0$ such that for every $\overline{p} \in \mathbb{Q}_+^n$ with $\|p - \overline{p}\| < \delta_2$, we have

$$|t_{\overline{\mathbb{S}^*}} - t_{\mathbb{S}^*}| < \frac{\epsilon}{2f(m)} \tag{5.11}$$

Let $\delta_0 := \min\{\delta_1, \delta_2\} > 0$. By (5.10) and (5.11), for every $\overline{p}$ with $\|p - \overline{p}\| < \delta_0$, we have

$$\begin{aligned} t_{\overline{\mathbb{S}}_{ALG'}} - f(m)t_{\overline{\mathbb{S}^*}} &= (t_{\mathbb{S}_{ALG'}} - f(m)t_{\mathbb{S}*}) + (t_{\overline{\mathbb{S}}_{ALG'}} - t_{\mathbb{S}_{ALG'}}) + (f(m)t_{\mathbb{S}^*} - f(m)t_{\overline{\mathbb{S}^*}}) \\ &> \epsilon - \tfrac{\epsilon}{2} - \tfrac{\epsilon}{2} \text{ from (5.10) and (5.11)} \\ &= 0. \end{aligned}$$

Thus, we obtain a counterexample to the conjecture with rational processing times. Each processing time in this counterexample can be expressed as $p_j = u_j/v_j$, where $u_j, v_j \in \mathbb{Z}_+$. Construct a new problem instance EI with integer processing times by multiplying each processing time by $\prod_{j=1}^n v_j$. Multiplying processing times by a constant does not change makespan ratios. Therefore, EI is also a counterexample to the conjecture. □

Corollary 5.3. *For the FM problem and the LD algorithm, the following must hold: If there exists a counterexample EI to a conjectured $t_{\mathbb{S}_{LD}}/t_{\mathbb{S}^*}$ ratio, then there exists a counterexample EN with integer processing times.*

Proof. We show that the FM problem and the LD algorithm satisfy the conditions listed in *Lemma* 5.3. The LD algorithm maps values in $\mathbb{R}_+^n$ onto values in $\mathbb{R}_+$. We need to show that the makespan $t_{\mathbb{S}_{LD}}$ of the LD solution is a continuous function at every point in $\mathbb{R}_+^n$ and that the optimal makespan $t_{\mathbb{S}^*}$ is a continuous function at every point in $\mathbb{R}_+^n$. Let $t_{\mathbb{S}_{LD}}(p)$ denote the makespan of the LD solution at $p \in \mathbb{R}_+^n$. $t_{\mathbb{S}_{LD}}$ is *continuous* at $\overline{p} \in \mathbb{R}_+^n$ if for every $\epsilon > 0$, there exists a $\delta > 0$ such that for all $p \in \mathbb{R}_+^n$, $\|p - \overline{p}\| < \delta \Rightarrow |t_{\mathbb{S}_{LD}}(p) - t_{\mathbb{S}_{LD}}(\overline{p})| < \epsilon$. Let $\epsilon > 0$ be given. Let $\overline{p} \in \mathbb{R}_+^n$ be arbitrary. Choose $\delta := \epsilon/k$, where k denotes the number of ranks. Let $p \equiv (p(1), p(2), \ldots, p(n))$. Select p such that $\|p - \overline{p}\|_\infty < \delta$. Let $p' \equiv (p'(1), p'(2), \ldots, p'(n))$ and $p'' \equiv (p''(1), p''(2), \ldots, p''(n))$, where $p'(j) = p(j) + \delta$

for $j = 1, 2, \ldots, n$ and $p''(j) = max(p(j) - \delta, 0)$ for $j = 1, 2, \ldots, n$. The following four inequalities follow from *Claim* 5.1.

$$t_{\mathbb{S}_{LD}}(p') - t_{\mathbb{S}_{LD}}(\overline{p}) > 0 \tag{5.12}$$

$$t_{\mathbb{S}_{LD}}(p'') - t_{\mathbb{S}_{LD}}(\overline{p}) \leq 0 \tag{5.13}$$

$$t_{\mathbb{S}_{LD}}(p) - t_{\mathbb{S}_{LD}}(p') = -k\delta \tag{5.14}$$

$$t_{\mathbb{S}_{LD}}(p) - t_{\mathbb{S}_{LD}}(p'') \leq k\delta \tag{5.15}$$

From (5.12) and (5.14),

$$t_{\mathbb{S}_{LD}}(p) - t_{\mathbb{S}_{LD}}(\overline{p}) \geq -k\delta \tag{5.16}$$

From (5.13) and (5.15),

$$t_{\mathbb{S}_{LD}}(p) - t_{\mathbb{S}_{LD}}(\overline{p}) \leq k\delta \tag{5.17}$$

From (5.16) and (5.17),

$$|t_{\mathbb{S}_{LD}}(p) - t_{\mathbb{S}_{LD}}(\overline{p})| \leq k\delta$$

ϵ was selected to be greater than $k\delta$. It follows that

$$|t_{\mathbb{S}_{LD}}(p) - t_{\mathbb{S}_{LD}}(\overline{p})| \leq \epsilon.$$

From *Claim* 5.4, it follows that the optimal makespan $t_{\mathbb{S}^*}$ is a continuous function at every point in $\mathbb{R}^n_+$. This completes the proof of the corollary. □

Let us define an LD_0 *schedule* to be a flowtime-optimal schedule in which the second rank is assigned largest-first. An LD_0 schedule is constructed using the following four-step process: (i) Jobs in the first rank are assigned arbitrarily to machines. (ii) Jobs in the second rank are assigned largest-first. Thus the jobs are assigned in nonincreasing order of processing times to the earliest available machine. (iii) Jobs in each remaining rank are assigned arbitrarily to machines. (iv) The jobs assigned to each machine are reversed to make the schedule flowtime-optimal. A *worst-case* LD_0 *schedule* for a given set of tasks is an LD_0 schedule with the largest length among all LD_0 schedules for that set of tasks. A problem instance with a *worst-case* $t_{\mathbb{S}_{LD_0}}/t_{\mathbb{S}^*}$ ratio is one with the largest $t_{\mathbb{S}_{LD_0}}/t_{\mathbb{S}^*}$ ratio among all flowtime-optimal schedules. Let us define the $LD_{0_{worst}}$ schedule to be the LD_0 schedule with the largest makespan ratio among all LD_0 schedules for a given problem instance. Clearly, an $LD_{0_{worst}}$ schedule is constructed by assigning the second rank largest-first and, if machine i' has the largest completion time after rank 2, assigning a job with processing time λ_r to machine i' for $r = 3, \ldots, k$. From this definition, it follows that the worst-case makespan ratio of the $LD_{0_{worst}}$ algorithm is equal to the worst-case makespan ratio of the LD_0 algorithm. Let $t_{S_{LD_{0_{worst}}}}$ denote the makespan of the $LD_{0_{worst}}$ schedule.

Corollary 5.4. *For the FM problem and the $LD_{0_{worst}}$ algorithm, the following must hold: If there exists a counterexample EI to a conjectured $t_{S_{LD_{0_{worst}}}}/t_{\mathbb{S}^*}$ ratio, then there exists a counterexample EN with integer processing times.*

Proof. The proof of this corollary is essentially identical to the proof of the previous corollary. □

The following proposition is a generalization of the previous lemma to a larger class of constrained minimization problems. This class consists of all constrained minimization problems that satisfy the following conditions: (i) Every member of this class is a constrained minimization problem P with a vector of q positive real-valued parameters $\overline{k}$. (ii) For every member of this class, a vector p of n real-valued variables defines an instance PI of the problem P. (iii) The optimal objective function value $g(\mathbb{S}^*)$ is continuous for every member of this class.

Proposition 5.1. *Consider any constrained minimization problem P with a vector of q positive real-valued parameters $\overline{k}$. Let a vector p of n real-valued variables define an instance PI of the problem P. For any algorithm ALG for problem P that produces a feasible solution $\mathbb{S}_{ALG}$ whose objective function value $\mathrm{g}(\mathbb{S}_{ALG}) : \mathrm{G} \rightarrow \mathbb{R}_+$ is a continuous function at every point in G, where $\mathrm{G} \subseteq \mathbb{R}^n_+$, and for a conjecture that states that the objective function value is no greater than $\mathrm{f}(\overline{k}).\mathrm{g}(\mathbb{S}^*)$, where $\mathrm{f} : \mathbb{R}^q_+ \rightarrow \mathbb{R}_+$ and $\mathrm{g}(\mathbb{S}^*)$ denotes the optimal objective function value, and where $\mathrm{g}(\mathbb{S}^*)$ is continuous and $\mathrm{g}(\mathbb{S}^*)$ and $\mathrm{g}(\mathbb{S}_{ALG})$ are homogeneous of the same degree, the following must hold: If there exists a counterexample EI to a conjectured $\frac{\mathrm{g}(\mathbb{S}_{ALG})}{\mathrm{g}(\mathbb{S}^*)}$ ratio, then there exists a counterexample EN with integer processing times.*

Proof. Assume that there exists a counterexample EI with one or more of the n variables having values that are irrational numbers. Then, $g(\mathbb{S}_{ALG})$ associated with EI is greater than the conjectured upper bound on $g(S_{ALG})$ associated with EI and there exists $\epsilon > 0$ such that

$$g(S_{ALG}) > f(\overline{k}).g(\mathbb{S}^*) + \epsilon.$$

Let p denote the vector of variables for EI. Let $g(\mathbb{S}_{ALG})$ denote the value of the objective function associated with the solution for EI generated by ALG. Let $g(\mathbb{S}^*)$ denote the value of the objective function for the optimal solution to EI. Let ER be a problem instance with all n variables having values that are rational numbers. Let $\overline{p}$ denote the vector of variables for ER. Let $g(\overline{\mathbb{S}}_{ALG})$ denote the value of the objective function associated with the solution for ER generated by ALG. Let $g(\overline{\mathbb{S}^*})$ denote the value of the objective function for the

optimal solution to ER. Since $g(\mathbb{S}_{ALG})$ is continuous, we may pick $\delta_1 > 0$ and $\overline{p} \in \mathbb{Q}_+^n$ such that $\|p - \overline{p}\| < \delta_1$. This implies that

$$\left| g(\overline{\mathbb{S}}_{ALG}) - g(\mathbb{S}_{ALG}) \right| < \frac{\epsilon}{2}. \tag{5.18}$$

Since $g(\mathbb{S}^*)$ is continuous, we may choose $\delta_2 > 0$ such that $\|p - \overline{p}\| < \delta_2$

$$\Rightarrow \left| g(\overline{\mathbb{S}^*}) - g(\mathbb{S}^*) \right| < \frac{\epsilon}{2f(\overline{k})}. \tag{5.19}$$

Let $\delta_0 := \min\{\delta_1, \delta_2\}$. Then $\|p - \overline{p}\| < \delta_0$ implies that

$$\begin{aligned}
&g(\overline{\mathbb{S}}_{ALG}) - f(\overline{k})g(\overline{\mathbb{S}^*}) = \\
&(g(\mathbb{S}_{ALG}) - f(\overline{k})g(\mathbb{S}^*)) + (g(\overline{\mathbb{S}}_{ALG}) - g(\mathbb{S}_{ALG})) + (f(\overline{k})g(\mathbb{S}^*) - f(\overline{k})g(\overline{\mathbb{S}^*})) \\
&> \epsilon - \frac{\epsilon}{2} - \frac{\epsilon}{2} \text{ (from (5.18) and (5.19))} \\
&= 0.
\end{aligned}$$

ER is thus a counterexample to the conjecture with rational processing times. Each processing time in ER can be expressed as $p_j = u_j/v_j$, where $u_j, v_j \in \mathbb{Z}^+$. Construct a new problem instance EI with integer processing times by multiplying each processing time in ER by $\prod_{j=1}^{n} v_j$. Since $g(\overline{\mathbb{S}}_{ALG})$ and $g(\mathbb{S}^*)$ are homogeneous functions of the data with the same degree of homogeneity, multiplying processing times by a constant does not change the ratio of objective function values. Therefore, EI is also a counterexample to the conjecture. □

The following lemmas are useful because they make it possible to subsequently restrict our attention to rectangular schedules for an examination of the worst-case makespan ratio for the LD_0 and LD algorithms.

Proposition 5.2. *Increasing the processing times of one or more tasks in the first rank of a $LD_{0_{worst}}$ schedule, while leaving the remaining processing times unchanged, will result in a $LD_{0_{worst}}$ schedule with the same or higher length.*

Proof. Let $a(2)$ denote the profile of the $LD_{0_{worst}}$ schedule after rank 2. $a_1(2)$ is the length of the set of tasks upon completion of the second rank in an $LD_{0_{worst}}$ schedule. An $LD_{0_{worst}}$ schedule is obtained by assigning the tasks with the largest processing times in ranks 3 through k, where k is the last rank, to a machine with completion time $a_1(2)$ after rank 2. So, $t_{\mathbb{S}_{LD_{0_{worst}}}} = a_1(2) + \sum_{i=3}^{k} \lambda_j$, with $a_1(2) = \max_{i=1,\ldots,m}(\tau_{i,1} + \tau_{m-i+1,2})$, where $\tau_{i,j}$ refers to the i^{th} largest processing time in rank j. Leaving all $\tau_{i,2}$ values unchanged and increasing one or more $\tau_{i,1}$ values can only increase the value of $a_1(2)$ and thus of $t_{\mathbb{S}_{LD_{0_{worst}}}}$. □

A problem instance with a *worst-case* $t_{\mathbb{S}_{LD0_{worst}}}/t_{\mathbb{S}^*}$ ratio is one with the largest $t_{\mathbb{S}_{LD0_{worst}}}/t_{\mathbb{S}^*}$ ratio among all flowtime-optimal schedules.

Lemma 5.4. *For any problem instance P_A for the FM problem, there exists a problem instance P_B with a rectangular optimal schedule and a $t_{\mathbb{S}_{LD0_{worst}}}/t_{\mathbb{S}^*}$ ratio that is at least the makespan ratio for P_A.*

Proof. Consider the optimal schedule for P_A. Construct a new problem instance as follows: For every job in the first rank that is performed on a machine for which the completion time after rank k in the optimal schedule is less than the makespan of the optimal schedule, increase the processing time so that the completion time after rank k in the optimal schedule is equal to the makespan. Clearly, the optimal schedule for the new problem instance is a rectangular schedule with a makespan equal to the makespan of P_A. Also, the new problem instance continues to be a flowtime-optimal problem instance because the increase in processing times does not lead to a violation of the rank restrictions. From *Proposition* 5.2, the makespan of the $LD0_{worst}$ schedule for the new problem instance is at least the makespan of the $LD0_{worst}$ schedule for the original problem instance. Therefore, the new problem instance has a $t_{\mathbb{S}_{LD0_{worst}}}/t_{\mathbb{S}^*}$ ratio that is at least the makespan ratio for P_A. □

Corollary 5.5. *There always exists a problem instance with a worst-case $t_{\mathbb{S}_{LD0_{worst}}}/t_{\mathbb{S}^*}$ ratio and a rectangular optimal schedule.*

Lemma 5.5. *There exists a problem instance with a rectangular optimal schedule and a worst-case $t_{\mathbb{S}_{LD0_{worst}}}/t_{\mathbb{S}^*}$ ratio in which the following hold:*
(i) All the tasks in the second rank have a processing time equal to λ_3.
(ii) There exist one or more tasks in the first rank with a processing time equal to λ_3.
Thus, $\mu_1 = \lambda_2 = \mu_2 = \lambda_3$.

Proof. By *Corollary* 5.4, there always exists a problem instance with a rectangular optimal schedule and a worst-case $t_{\mathbb{S}_{LD0_{worst}}}/t_{\mathbb{S}^*}$ ratio. Let $\mathbb{S}^*$ denote this rectangular optimal schedule. Let the machines be labelled based on the completion times after rank 2 in the optimal schedule $\mathbb{S}^*$. Thus the machine with the largest completion time after rank 2 in $\mathbb{S}^*$ is labelled machine 1, the machine with the second-largest completion time after rank 2 in $\mathbb{S}^*$ is labelled machine 2, and the machine with the q^{th} largest processing time (for $q = 1, 2, ..., m$) is labelled machine q. Note that $a_i(2)$ denotes the i^{th} largest element of the $\mathbb{S}^*$ profile after rank 2. This labelling ensures that machine i (for $i = 1, 2, ..., m$) has completion time $a_i(2)$ after rank 2.

For machine $i = 1, 2, \cdots, m$: Set the processing time of the task in the second rank equal to λ_3 and the processing time of the task in the first rank equal to $a_i(2) - \lambda_3$. This

change will either increase or leave unchanged the processing time of each job in rank 1. Therefore, it will not violate the rank restriction. Note that $a_i(2)$ remains unchanged and therefore the value of $t_{\mathbb{S}^*}$ remains unchanged.

For the $LD_{0_{worst}}$ schedule, the set of changes in the task set described above is equivalent to: (i) Reassigning the tasks in rank 2 so that each task in rank 2 is placed after the same task in rank 1 as it was in the $\mathbb{S}^*$ schedule. (ii) After completing step (i): For machine $i = 1, 2, \cdots, m$ (labelled as indicated above): Set the processing time of the task in the second rank equal to λ_3 and the processing time of the task in the first rank equal to $a_i(2) - \lambda_3$.

Note that the resulting schedule is an $LD_{0_{worst}}$ schedule. An $LD_{0_{worst}}$ schedule is optimal for a two-rank system. The rearrangement of the tasks in step (i) can therefore only increase the value of $a_i(2)$. $t_{\mathbb{S}_{LD_{0_{worst}}}} = a_i(2) + \sum_{j=3}^{k} \lambda_j$. Therefore, the value of $t_{\mathbb{S}_{LD_{0_{worst}}}}$ can only increase as a result of the rearrangement in step (i). The subsequent changes in step (ii) do not cause any further change in $a_i(2)$ and therefore do not cause any further change in $t_{\mathbb{S}_{LD_{0_{worst}}}}$.

Thus the value of $t_{\mathbb{S}_{LD_{0_{worst}}}}$ can only increase as a result of steps (i) and (ii) while the value of $t_{\mathbb{S}^*}$ remains unchanged. Therefore, the ratio $t_{\mathbb{S}_{LD_{0_{worst}}}}/t_{\mathbb{S}^*}$ can only increase.

After completing steps (i) and (ii), if $a_m(1) > \lambda_3$, for $i = 1, 2, \cdots, m$: set $a_i(1)$ equal to $a_i(1) - (a_m(1) - \lambda_3)$. By *Claim* 5.1, this will reduce $t_{\mathbb{S}_{LD_{0_{worst}}}}$ and $t_{\mathbb{S}^*}$ by the same amount and will therefore increase the ratio $t_{\mathbb{S}_{LD_{0_{worst}}}}/t_{\mathbb{S}^*}$. This will result in a value of μ_1 equal to λ_3. □

The following lemma shows that $t_{\mathbb{S}_{LD_{0_{worst}}}}$ is a continuous function.

Lemma 5.6. *$t_{\mathbb{S}_{LD_{0_{worst}}}}$ is a continuous function at every point in its domain.*

Proof. Let $\tau_{i,h}$ refers to the i^{th} largest processing time in rank h.
$t_{\mathbb{S}_{LD_{0_{worst}}}} = \max_{i=1,\ldots,m} [\tau_{i,1} + \tau_{m-i,2}] + \sum_{r=3,\ldots,k} \lambda_r$. It is evident that $t_{\mathbb{S}_{LD_{0_{worst}}}}$ is a continuous function at every point in its domain. □

Coffman and Sethi (1976) proved that the ratio of the makespan of any schedule in which the rank containing the largest processing times was assigned largest-first to the makespan of the optimal schedule could not exceed $(4m - 3)/(3m - 2)$, and this bound could be achieved for $m = 2$ and for $m = 3$. The following theorem provides a similar bound for $LD_{0_{worst}}$ schedules. It shows that the worst-case ratio for $LD_{0_{worst}}$ schedules cannot exceed 4/3. While Coffman and Sethi suggest that their $(4m - 3)/(3m - 2)$ bound is unlikely to be achieved for $m > 3$, this bound can be achieved for all m. The proof below uses an approach that is different from the approach used by Coffman and Sethi.

Theorem 5.1. $1 \leq t_{\mathbb{S}_{LD0_{worst}}}/t_{\mathbb{S}^*} \leq 4/3$, *and the upper bound is achieved for all* $m \geq 3$.

Proof. Consider a problem instance PI that satisfies the conditions established in *Lemma* 5.5. All the tasks in the second rank of a flowtime-optimal schedule for this problem instance have a processing time equal to λ_3. Let $t_{\mathbb{S}}$ denote the makespan of $\mathbb{S}$, the $LD0_{worst}$ schedule for this problem instance. Let $t_{\mathbb{S}^*}$ denote the makespan of $\mathbb{S}^*$, the optimal schedule for this problem instance.

Now, construct a new problem instance PI' by removing the jobs in the second rank from PI. Consider the schedules obtained by removing the jobs in the second rank from $\mathbb{S}$ and $\mathbb{S}^*$. Coffman and Sethi (1976) show that, for any problem instance with a flowtime-optimal schedule, the makespan ratio is less than or equal to 3/2. It follows that

$$\begin{aligned}(t_{\mathbb{S}_{LD0_{worst}}} - \lambda_3)/(t_{\mathbb{S}^*} - \lambda_3) &\leq 3/2. \\ \Rightarrow t_{\mathbb{S}_{LD0_{worst}}}/t_{\mathbb{S}^*} &\leq 3/2 - \lambda_3/2t_{\mathbb{S}^*}\end{aligned}$$

Note that $\mathbb{S}^*$ is rectangular. Therefore, the machine with processing time μ_1 in rank 1 has a completion time at the end of rank k that is equal to the length of $\mathbb{S}^*$. An upper bound on $t_{\mathbb{S}^*}$ can be obtained by adding the largest processing time in ranks 2 through k to μ_1. Therefore,

$$\begin{aligned}t_{\mathbb{S}^*} &\leq \mu_1 + \textstyle\sum_{j=2}^{k} \lambda_j \\ \Rightarrow t_{\mathbb{S}^*} &\leq 3\lambda_3 + \textstyle\sum_{j=4}^{k} \lambda_j.\end{aligned}$$

This conversion can be done by removing the jobs in the second rank. It follows that

$$\begin{aligned}(t_{\mathbb{S}_{LD0_{worst}}} - \lambda_3)/(t_{\mathbb{S}^*} - \lambda_3) &\leq 3/2. \\ \Rightarrow t_{\mathbb{S}_{LD0_{worst}}}/t_{\mathbb{S}^*} &\leq 3/2 - \lambda_3/2t_{\mathbb{S}^*}.\end{aligned}$$

From *Lemma* 5.3 and *Lemma* 5.6, it follows that, if there exists a counterexample to the 4/3 bound, there exists a counterexample in which all processing times are integers. Also note that, if a counterexample to the 4/3 bound exists, a problem instance with a worst-case $t_{\mathbb{S}_{LD0_{worst}}}/t_{\mathbb{S}^*}$ ratio would be a counterexample. Let us assume that the 4/3 conjecture is false. For a problem instance with a worst-case $t_{\mathbb{S}_{LD0_{worst}}}/t_{\mathbb{S}^*}$ ratio and integer processing times,

$$\begin{aligned}\lambda_i &\leq \lambda_3 - i + 3 \text{ for } 4 \leq i \leq k. \\ \Rightarrow \sum_{j=4}^{k} \lambda_j &\leq (k-3)(2\lambda_3 - k + 2)/2. \\ \Rightarrow t_{\mathbb{S}^*} &\leq k\lambda_3 - \tfrac{1}{2}(k - \tfrac{5}{2})^2 + \tfrac{1}{8}.\end{aligned}$$

Assume that λ_3 is fixed and is set equal to a positive integer value. (Note that this is only a matter of scaling up or scaling down all processing times.) For $k \geq 3$, and assuming integer processing times, λ_3 must be greater than or equal to $k-2$. The right-hand side of the above inequality has a partial derivative with respect to k that is equal to $\lambda_3 - k + \frac{5}{2}$. It follows that, for $3 \leq k \leq \lambda_3 + 2$, the right-hand side has a positive partial derivative with respect to k. For $k = 3$, the right-hand side is equal to $3\lambda_3$. Therefore, for $k \geq 3$, the right-hand side is greater than or equal to $3\lambda_3$. It follows that

$$\begin{aligned} t_{\mathbb{S}_{LD0_{worst}}}/t_{\mathbb{S}^*} &\leq 3/2 - (1/2)(1/3). \\ \Rightarrow t_{\mathbb{S}_{LD0_{worst}}}/t_{\mathbb{S}^*} &\leq 4/3. \end{aligned}$$

However, this contradicts the assumption that there exists a counterexample to the 4/3 bound. It follows that no such counterexample exists and the bound is valid.

For any value of $m \geq 2$, the $t_{\mathbb{S}_{LD0_{worst}}}/t_{\mathbb{S}^*}$ ratio equals the upper bound for the following three-rank system:

$$\begin{aligned} p_i &= 2 \quad \text{for} \quad 1 \leq i \leq m-1, \\ p_i &= 1 \quad \text{for} \quad m \leq i \leq 2m+1, \\ p_i &= 0 \quad \text{for} \quad 2m+2 \leq i \leq 3m. \end{aligned}$$

This completes the proof of the lemma. □

Corollary 5.6. $1 \leq t_{\mathbb{S}_{LD_0}}/t_{\mathbb{S}^*} \leq 4/3$, *and the upper bound is achieved for all* $m \geq 3$.

Proof. This follows from the previous theorem and the definition of an $LD0_{worst}$ schedule. □

The results obtained above for the simple LD_0 algorithm provided insight into the approaches that could be used to prove bounds for other algorithms for problem FM. The rest of this chapter will focus the LD algorithm proposed by Coffman and Sethi (1976a).

Lemma 5.7. *An increase in one or more processing times of jobs in rank* r *for* $1 \leq r \leq k-1$ *(with no change in the remaining processing times, and subject to the rank constraint) does not result in a reduction in any element of the profile* $b(\ell)$ *of an LD schedule after rank* ℓ *for* $r \leq \ell \leq k$.

Proof. The lemma will be proved by induction on ℓ.

Let us assume that the lemma holds for ℓ' ranks, where $\ell' \in \{r, r+1, \ldots, s\}$.

Let $\tau_{i,h}$ refers to the i^{th} largest processing time in rank h.

The induction assumption states that an increase in processing times in rank r does not cause a reduction in $b_{m-i+1}(\ell')$ for $i = 1, 2, \ldots, m$. Note that the increase in processing times in rank r leaves $\tau_{i,\ell'+1}$ unchanged for $i = 1, 2, \ldots, m$.

The profile $b(\ell' + 1)$ after rank $\ell' + 1$ consists of the following m elements: $b_{m-i+1}(\ell') + \tau_{i,\ell'+1}$ for $i = 1, 2, \ldots, m$.

It follows that none of the elements of the profile $b(\ell' + 1)$ gets reduced as a result of the increase in processing times in rank r. Thus the theorem holds for rank $\ell' + 1$, for $\ell' \in \{r, r + 1, \ldots, s\}$. It follows that the theorem holds for rank ℓ' for $\ell' \in \{r, r + 1, \ldots, s, s + 1\}$.

The base case follows from the fact that the result holds trivially for $\ell' = r$. □

One important fact follows from the above theorem. An optimal flowtime-optimal schedule that is not rectangular can be made rectangular by increasing the lengths of all tasks in the first rank that are performed on machines that have a completion time after the last rank that is strictly less than the makespan. This result is stated and proved below.

Lemma 5.8. *There always exists a problem instance with a worst-case $t_{\mathbb{S}_{LD}}/t_{\mathbb{S}^*}$ ratio that has the following property: There exists a rectangular optimal schedule for this set of jobs. (Coffman and Sethi, 1976a)*

Proof. Consider the optimal schedule for any problem instance. For every job in the first rank that is performed on a machine for which the completion time after rank k in the optimal schedule is less than the makespan of the optimal schedule, increase the processing time so that the completion time after rank k in the optimal schedule is equal to the makespan. Clearly, the increase in processing times does not lead to a violation of the rank restriction for flowtime-optimal schedules. Also, the increase in processing times results in a rectangular optimal schedule. From *Lemma* 5.7, the increase in processing times does not affect the makespan of the optimal schedule but may increase the makespan of the LD schedule. Therefore, it will either increase or leave unchanged the $t_{\mathbb{S}_{LD}}/t_{\mathbb{S}^*}$ ratio. □

Note that, if the Coffman-Sethi conjecture is false, a problem instance with a worst-case $t_{\mathbb{S}_{LD}}/t_{\mathbb{S}^*}$ ratio would be a counterexample to the conjecture. This leads to the following corollary.

Corollary 5.7. *If the Coffman-Sethi conjecture is false, then there exists a minimal counterexample to the conjecture of Type R.*

This corollary states that, if the Coffman-Sethi conjecture is false, there always exists a minimal counterexample to the conjecture with a rectangular optimal schedule.

Rectangular worst-case schedules are interesting for the following reasons. They formed the basis of the derivation of the worst-case LI bound by Coffman and Sethi. Therefore they hold out the promise of being able to extend the approach used for an LI bound to study LD bounds. Further, the problem of finding the rectangular worst-case schedule is related to the partition problem. In the classical partition problem, the objective is to partition a set of integers into two groups, each of which has the same sum.

Rectangular schedules contain more information than non-rectangular schedules because they impose a restriction on the sets of processing times in each rank. The restriction is the following: If there are m machines and k ranks, the union of the k sets, each consisting of m processing times, can be partitioned into m subsets, each consisting of k processing times with the same sum. This restricts the set of potential worst cases that must be examined.

The following lemma shows that, if the Coffman-Sethi conjecture is false, every minimal counterexample to the conjecture of Type A or Type I has only two processing times in the last rank.

Lemma 5.9. *If the Coffman-Sethi conjecture is false, then every minimal counterexample of Type A or Type I with k ranks satisfies the following:*
(a) The processing time of jobs in the last rank are equal to either λ_k or μ_k.
(b) $\mu_k = 0$.
(c) For $1 \leq r \leq k-1$, $\mu_r = \lambda_{r+1}$. (Huang and Tunçel, 2004)

Proof. If the Coffman-Sethi conjecture is false, there exists a minimal counterexample of Type A. We begin by proving the lemma for minimal counterexamples of Type A. Part (a) can be readily proved by considering a minimal counterexample of Type A that does not satisfy this condition. All jobs in the last rank cannot have the same positive processing time; if they do have the same processing time, the last rank could be removed to obtain a problem instance with a larger $t_{\mathbb{S}_{LD}}/t_{\mathbb{S}^*}$ ratio. If j is the job (with processing time p_j) that finishes last in the LD schedule, we reduce the processing time of every job in the last rank with processing time greater than p_j to p_j, and we reduce the processing time of every job in the last rank with processing time less than p_j to μ_k. This has no impact on the length of the LD schedule and cannot result in an increase in the length of the optimal schedule. Thus the makespan ratio either stays the same or increases. The resulting problem instance is smaller than the original problem instance. This contradicts the assumption that the original problem instance was a minimal counterexample. It follows that the minimal counterexample satisfies part (a) of the lemma. To prove part (b), we assume that there exists a minimal counterexample of Type A with μ_k greater than 0. We

subtract μ_k from every processing time in rank k. This results in both $t_{\mathbb{S}_{LD}}$ and $t_{\mathbb{S}^*}$ being reduced by mu_k, thus resulting in an increase in the $t_{\mathbb{S}_{LD}}/t_{\mathbb{S}^*}$ ratio. The resulting problem instance is smaller than the original problem instance. This contradicts the assumption that the original problem instance was a minimal counterexample. It follows that the minimal counterexample satisfies part (b) of the lemma. To prove part (c)), we assume that there exists a minimal counterexample of Type A in which $\mu_h > \lambda_{h+1}$ for some value of h that satisfies $1 \leq h \leq k-1$. We subtract $\mu_h - \lambda_{h+1}$ from the processing time of every job in rank h. This results in both $t_{\mathbb{S}_{LD}}$ and $t_{\mathbb{S}^*}$ being reduced by $\mu_h - \lambda_{h+1}$, thus resulting in an increase in the $t_{\mathbb{S}_{LD}}/t_{\mathbb{S}^*}$ ratio. The resulting problem instance is smaller than the original problem instance. This contradicts the assumption that the original problem instance was a minimal counterexample. It follows that the minimal counterexample satisfies part (c) of the lemma.
From *Corollary* 5.3, if the Coffman-Sethi conjecture is false, there exists a minimal counterexample of Type I. The above proof also applies to minimal counterexamples of Type I. This follows from the fact that, for a problem instance of Type I, subtracting an integer from one or more processing times results in another problem instance of Type I. □

Chapter 6

Analysis of the 2-machine, 3-machine, m-machine (for general m), and 3-machine 3-rank cases, and a new proof technique

In this chapter, the approach used is to examine the validity of the Coffman-Sethi conjecture for various values of m, the number of machines. It is shown that, if the Coffman-Sethi conjecture is false, a minimal counterexample to the conjecture has at least three machines. Two approaches are used for the two-machine case. The first approach examines ranks in increasing order, starting with rank 1. The second approach examines ranks in decreasing order, starting with rank k. The second approach starts with the ranks contaning the largest jobs. It appears to be more effective and leads to a proof of the conjecture for the $m = 2$ case. For both the 2-machine case and the 3-machine, 3-rank case, an approach (obtained from Huang and Tunçel (2004)) inspired by the notion of LP duality is used. This approach treats the parameters of problem FM (job processing times) as variables. The problem of determining the values of the processing times that result in the LD makespan being maximised for a given value of the optimal makespan is formulated as a set of linear programs. (Here, *Lemma* 5.9 is utilized.) The solution to each linear program is checked to determine if it violates the conjecture. It is shown that the Coffman-Sethi conjecture holds for the $m = 2$ case and the $m = 3$, $k = 3$ case.

6.1 Analysis of the two-machine case

We now consider a situation in which the number of machines m is equal to 2.

If $\mu_j > \lambda_{j+1}$ for some value of j that satisfies $1 \leq j \leq k-1$, then $\mu_j - \lambda_{j+1}$ may be subtracted from every processing time in rank j to obtain a problem instance with a larger ratio of the LD makespan to the optimal makespan. It follows that, in a worst-case problem instance, $\mu_j = \lambda_{j+1}$ for $j = 1, 2, \cdots, k-1$.

6.1.1 First approach: Ranks examined in increasing order, starting with rank 1

Properties are developed for a minimal counterexample of Type R i.e. a minimal counterexample with a rectangular optimal schedule.

The completion times after rank 1 are λ_1 and μ_1. The completion times after rank 2 in the LD schedule are $\lambda_1 + \mu_2$ and $\mu_1 + \lambda_2$. Thus, the completion times after rank 2 in the LD schedule are $\lambda_1 + \lambda_3$ and $2\lambda_2$. The completion times after rank 3 are the following:

If $\lambda_1 + \lambda_3 \leq 2\lambda_2$, the completion times after rank 3 are $\lambda_1 + 2\lambda_3$ and $2\lambda_2 + \mu_3$. (6.1)

If $\lambda_1 + \lambda_3 \geq 2\lambda_2$, the completion times after rank 3 are $\lambda_1 + \lambda_3 + \mu_3$ and $2\lambda_2 + \lambda_3$. (6.2)

The first if condition listed above is labelled Condition 6.1 and the second if condition is labelled Condition 6.2. The second condition can be rewritten as: Condition 6.2: If $\lambda_1 - \lambda_2 \geq \lambda_2 - \lambda_3$, the completion times after rank 3 are $\lambda_1 + \lambda_3 + \lambda_4$ and $2\lambda_2 + \lambda_3$. If Condition 6.2 holds, the completion times after rank 4 are the following: If $\lambda_1 + \lambda_3 + \lambda_4 \leq 2\lambda_2 + \lambda_3$, the completion times after rank 4 are $\lambda_1 + \lambda_3 + 2\lambda_4$ and $2\lambda_2 + \lambda_3 + \lambda_5$. If $\lambda_1 + \lambda_3 + \lambda_4 \geq 2\lambda_2 + \lambda_3$, the completion times after rank 4 are $\lambda_1 + \lambda_3 + \lambda_4 + \lambda_5$ and $2\lambda_2 + \lambda_3 + \lambda_4$.

The second condition for rank 4 can be rewritten as: Condition 6.2.2: If $\lambda_1 - \lambda_2 \geq \lambda_2 - \lambda_4$, the completion times after rank 4 are $\lambda_1 + \lambda_3 + \lambda_4 + \lambda_5$ and $2\lambda_2 + \lambda_3 + \lambda_4$. Note that Condition 6.2.2 holds only if Condition 6.2 holds. Extending this line of reasoning further: Condition 6.2.$(k-2)$: If $\lambda_1 - \lambda_2 \geq \lambda_2 - \lambda_k$, the completion times after rank k are $\lambda_1 + \lambda_3 + \lambda_4 + \cdots + \lambda_k + \mu_k$ and $2\lambda_2 + \lambda_3 + \lambda_4 + \cdots + \lambda_k$. Note that Condition 6.2.$(k-2)$ holds only if Condition 6.2.r holds for $1 \leq r \leq k-3$. The makespan for an optimal rectangular schedule (with k ranks) $= \lambda_1/2 + \lambda_2 + \lambda_3 + \cdots + \lambda_{k-1} + \lambda_k$

We consider two subcases:
Case (i): $\lambda_1 \geq 2\lambda_2$. In this case the ratio of the LD makespan to the optimal makespan is $\frac{\lambda_1+\lambda_3+\lambda_4+\cdots+\lambda_k}{\lambda_1/2+\lambda_2+\lambda_3+\cdots+\lambda_{k-1}+\lambda_k}$. For $k > 3$, deleting every rank r for which $3 \leq r \leq k-1$ clearly results in an increase in this ratio. Therefore this ratio is maximized when $k = 3$. Therefore the ratio $= \frac{\lambda_1+\lambda_3}{\lambda_1/2+\lambda_2+\lambda_3}$. For a three-rank problem, a rectangular optimal schedule implies that one of the following holds: (a) $\lambda_1 = 2\lambda_3$, (b) $\lambda_1 + 2\lambda_3 = 2\lambda_2$, or (c) $\lambda_1 = 2\lambda_2$. For case (i), (a) and (b) cannot hold. Therefore $\lambda_1 = 2\lambda_2$. Therefore the ratio $= \frac{2\lambda_2+\lambda_3}{2\lambda_2+\lambda_3} = 1$. It follows that, in case (i), the LD schedule always provides an optimal solution.

Case (ii): $\lambda_1 < 2\lambda_2$. In this case the ratio of the LD makespan to the optimal makespan is $\frac{2\lambda_2+\lambda_3+\lambda_4+\cdots+\lambda_k}{\lambda_1/2+\lambda_2+\lambda_3+\cdots+\lambda_k}$. Clearly, this ratio is maximized by setting k equal to 3. Therefore the ratio $= \frac{2\lambda_2+\lambda_3}{\lambda_1/2+\lambda_2+\lambda_3}$. For a three-rank problem, a rectangular optimal schedule implies that one of the following holds: (a) $\lambda_1 = 2\lambda_3$, (b) $\lambda_1 + 2\lambda_3 = 2\lambda_2$, or (c) $\lambda_1 = 2\lambda_2$. Note that (b) and (c) cannot hold in case (ii). Therefore $\lambda_1 = 2\lambda_3$. The ratio $= \frac{2\lambda_2+\lambda_3}{\lambda_2+2\lambda_3} = 2 - 3/((\lambda_2/\lambda_3)+2)$. Note that $\lambda_1 = 2\lambda_3$ and $\lambda_1 + \lambda_3 \geq 2\lambda_2$. It follows that $\lambda_2/\lambda_3 \leq 3/2$. Clearly, the ratio is maximized by setting λ_2/λ_3 equal to 3/2. This gives a value of the ratio equal to $2 - 6/7 = 8/7$. This is equal to the value of the bound defined by the Coffman-Sethi conjecture.

Similar analyses can be performed by branching out on other conditions (such as Condition 1.1) and considering various cases.

6.1.2 Second approach: Ranks examined in decreasing order, starting with rank k

We consider a minimal counterexample of Type A i.e. a minimal counterexample that is not required to have either integer processing times or a rectangular optimal schedule.

Lemma 6.1. *: If the Coffman-Sethi conjecture is false, for $m = 2$ there exists a minimal counterexample of Type A with 3 ranks.*

Proof. In rank k, one of the two machines has a processing time of λ_k and the second machine has a processing time of 0. Clearly, the makespan is equal to the completion time after rank k on the machine with a processing time of λ_k. (If this is not the case, the last rank could be deleted to obtain a problem instance with the same or larger makespan ratio.) It follows that the two completion times on the two machines after rank $k-1$ in the LD schedule are $\geq t_{S_{LD}} - \lambda_k$. $\Rightarrow t_{S^*} \geq t_{S_{LD}} - \lambda_k + \lambda_k/2$. In a counterexample, $t_{S_{LD}}/t_{S^*} > 8/7 . \Rightarrow t_{S^*} < 7\lambda_k/2$. Clearly, $\mu_\ell \geq \lambda_k$ for $\ell < k$. $\Rightarrow t_{S^*} \geq k\lambda_k$. $\Rightarrow k \leq 3$.

For $k = 1$ and $k = 2$, the LD schedule is optimal. It follows that, if the Coffman-Sethi conjecture is false, there exists a minimal counterexample of Type A with 3 ranks.

□

The following analysis of the 2-machine case is based on an approach developed by Huang and Tunçel (2004). Huang and Tunçel's approach treats the parameters of problem FM (job processing times) as variables. For a given value of the optimal makespan, the problem of determining the values of the processing times that result in the LD makespan being maximised is set up as a set of linear programs. Each possible relationship between the processing times (subject to the rank restriction) results in a different linear program.

The solution to each linear program is checked to determine if it violates the Coffman-Sethi conjecture. Huang and Tunçel's approach is used to show that the Coffman-Sethi conjecture is not violated in the two-machine case.

Lemma 6.2. *The Coffman-Sethi conjecture holds for $m = 2$.*

Proof. Clearly, we only need to consider the $k = 3$ case.

There are two possible LD schedules:

LD schedule 1: Jobs with processing times $\lambda_1, \lambda_3, 0$ on machine 1, jobs with processing times $\lambda_2, \lambda_2, \lambda_3$ on machine 2.

LD schedule 2: Jobs with processing times $\lambda_1, \lambda_3, \lambda_3$ on machine 1, jobs with processing times $\lambda_2, \lambda_2, 0$ on machine 2.

For a makespan ratio > 1, the second and third ranks must not be the same in the LD schedule and the optimal schedule. There is only one possible optimal schedule: Jobs with processing times $\lambda_1, \lambda_2, 0$ on machine 1, jobs with processing times $\lambda_2, \lambda_3, \lambda_3$ on machine 2.

In each of the following cases, we set the optimal makespan equal to 1. The makespan ratio is then equal to the LD makespan. We seek to maximise the LD makespan in each case.

There are 4 possible values for the LD makespan, resulting in the following 4 cases.

Case 1: $t_{S_{LD}} = \lambda_1 + \lambda_3$. This will be true only if $\lambda_1 \geq 2\lambda_2 \Rightarrow \lambda_1 \geq 2\lambda_3 \Rightarrow t_{S^*} = \lambda_1 + \lambda_2$

This is clearly infeasible.

Case 2: $t_{S_{LD}} = 2\lambda_2 + \lambda_3$. This will be true only if $\lambda_1 \leq 2\lambda_2$ and $\lambda_1 + \lambda_3 \geq 2\lambda_2$

Case 2A: $\lambda_1 \leq 2\lambda_3$. So, $t_{S^*} = \lambda_2 + 2\lambda_3$

$Maximize\ 2\lambda_2 + \lambda_3$

subject to

$$\lambda_1 \leq 2\lambda_3$$
$$\lambda_2 + 2\lambda_3 = 1$$
$$2\lambda_2 \leq \lambda_1 + \lambda_3$$

This can be simplified as follows:

$Maximize\ 2\lambda_2 + \lambda_3$

subject to

$$2\lambda_2 \leq 3\lambda_3$$
$$\lambda_2 + 2\lambda_3 = 1$$

Solution: $\lambda_2 = 3/7, \lambda_3 = 2/7$, objective function $= 8/7$.

Case 2B: $\lambda_1 \geq 2\lambda_3 \Rightarrow t_{S^*} = \lambda_1 + \lambda_2$

$Maximize\ 2\lambda_2 + \lambda_3$

subject to

$$\lambda_1 + \lambda_2 = 1$$
$$\lambda_1 \geq 2\lambda_3$$
$$\lambda_1 \leq 2\lambda_2$$
$$2\lambda_2 \leq \lambda_1 + \lambda_3$$

This can be simplified as follows:

$Maximize\ 2\lambda_2 + \lambda_3$

subject to

$$\lambda_2 \geq 1/3$$
$$\lambda_2 + 2\lambda_3 \leq 1$$
$$3\lambda_2 \leq 1 + \lambda_3$$

Solution: $\lambda_2 = 3/7, \lambda_3 = 2/7$, objective function $= 8/7$.

Case 3: $t_{S_{LD}} = \lambda_1 + 2\lambda_3$. This will be true only if $\lambda_1 + \lambda_3 \leq 2\lambda_2$ and $\lambda_1 + 2\lambda_3 \geq 2\lambda_2$.

Case 3A: $\lambda_1 \leq 2\lambda_3 \Rightarrow t_{S^*} = \lambda_2 + 2\lambda_3$

$Maximize\ \lambda_1 + 2\lambda_3$

subject to

$$\lambda_1 + \lambda_3 \leq 2\lambda_2$$
$$\lambda_1 + 2\lambda_3 \geq 2\lambda_2$$
$$\lambda_2 + 2\lambda_3 = 1$$
$$2\lambda_1 \leq 2\lambda_3$$

The constraints can be replaced by the following equivalent set of constraints:

$\lambda_1 + 5\lambda_3 \leq 2$

$\lambda_1 + 6\lambda_3 \geq 2$

$2\lambda_1 \leq 2\lambda_3$

Solution: $\lambda_1 = 4/7, \lambda_2 = 3/7, \lambda_3 = 2/7$, objective function $= 8/7$.

Case 3B: $\lambda_1 \geq 2\lambda_3 \Rightarrow t_{S^*} = \lambda_1 + \lambda_2$

Maximize $\lambda_1 + 2\lambda_3$

subject to

$$\lambda_1 + \lambda_2 = 1$$
$$\lambda_1 \geq 2\lambda_3$$
$$\lambda_1 + \lambda_3 \leq 2\lambda_2$$
$$\lambda_1 + 2\lambda_3 \geq 2\lambda_2$$

This can be simplified as follows:

Maximize $\lambda_1 + 2\lambda_3$

subject to

$$\lambda_1 \geq 2\lambda_3$$
$$3\lambda_1 + \lambda_3 \leq 2$$
$$3\lambda_1 + 2\lambda_3 \geq 2$$

Solution: $\lambda_1 = 4/7, \lambda_2 = 3/7, \lambda_3 = 2/7$, objective function $= 8/7$.

Case 4: $t_{S_{LD}} = 2\lambda_2$

This will be true only if $2\lambda_2 \geq \lambda_1 + 2\lambda_3$

Case 4A: $\lambda_1 \leq 2\lambda_3 \Rightarrow t_{S^*} = \lambda_2 + 2\lambda_3$.

Maximize $2\lambda_2$

subject to

$$\lambda_1 \leq 2\lambda_3$$
$$\lambda_2 + 2\lambda_3 = 1$$
$$2\lambda_2 \geq \lambda_1 + 2\lambda_3$$

The constraints can be replaced by the following equivalent set of constraints:

$\lambda_1 + \lambda_2 \leq 1$

$-2\lambda_1 + 5\lambda_2 \geq 1$.

Solution: $\lambda_1 = 1/2, \lambda_2 = 1/2, \lambda_3 = 1/4$, objective function $= 1$.

Case 4B: $\lambda_1 \geq 2\lambda_3 \Rightarrow t_{S^*} = \lambda_1 + \lambda_2$

Maximize $2\lambda_2$

subject to

$$\lambda_1 + \lambda_2 = 1$$

$\lambda_1 \geq 2\lambda_3$

$\lambda_1 + 2\lambda_3 \leq 2\lambda_2$

This can be simplified as follows:

Maximize $2\lambda_2$

subject to

$\lambda_2 \geq \lambda_3$

$2\lambda_3 + \lambda_2 \leq 1$

$-2\lambda_3 + 3\lambda_2 \geq 1$

Solution: $\lambda_1 = 1/2, \lambda_2 = 1/2, \lambda_3 = 1/4$, objective function $= 1$. □

The second approach appears to be more promising. For $m > 2$, we will use the second approach.

6.2 Huang and Tunçel's analysis of the 3-machine, 3-rank case

The following analysis shows that the Coffman-Sethi conjecture holds for the $m = 3, k = 3$ case. It uses an approach similar to the approach used for the $m = 2$ case. In general, after reducing the size of a minimal counterexample to a scheduling conjecture to a relatively small size, it may be possible to use this approach to prove or disprove the conjecture.

From the previous lemma, $\mu_1 = \lambda_2$ and $\mu_2 = \lambda_3$ and $\mu_3 = 0$ and either $\alpha_3 = \lambda_3$ or $\alpha_3 = \mu_3$. For $1 \leq r \leq 3$, let λ_r, α_r and μ_r denote the processing times in rank r, where $\lambda_r \geq \alpha_r \geq \mu_r$.

The first two ranks of the LD schedule will look like

$$\begin{pmatrix} \lambda_1 & \lambda_3 \\ \alpha_1 & \alpha_2 \\ \lambda_2 & \lambda_2 \end{pmatrix}.$$

The third rank will fit depending on the length of processing times on the first two ranks. We will use case analysis to cover all possible LD schedules.

For all cases we will have $\lambda_1 \geq \alpha_1 \geq \lambda_2 \geq \alpha_2 \geq \lambda_3 > 0$. This results in the following 5 constraints.

$$\begin{array}{cccccccc} -\lambda_1 & & & +\alpha_1 & & \leq & 0 & (1) \\ & \lambda_2 & & -\alpha_1 & & \leq & 0 & (2) \\ & -\lambda_2 & & & +\alpha_2 & \leq & 0 & (3) \\ & & \lambda_3 & & -\alpha_2 & \leq & 0 & (4) \\ & & -\lambda_3 & & & \leq & 0 & (5) \end{array}$$

We begin by looking at the cases when $\alpha_3 = \lambda_3$.

Consider the case $\lambda_1 + \lambda_3 \geq \alpha_1 + \alpha_2 \geq 2\lambda_2$, then we have the constraints

$$\begin{array}{ccccccc} -\lambda_1 & & -\lambda_3 & +\alpha_1 & \alpha_2 & \leq & 0 \quad (6) \\ & 2\lambda_2 & & -\alpha_1 & -\alpha_2 & \leq & 0 \quad (7) \end{array}$$

Further, the LD schedule will look like

$$\begin{pmatrix} \lambda_1 & \lambda_3 & 0 \\ \alpha_1 & \alpha_2 & \lambda_3 \\ \lambda_2 & \lambda_2 & \lambda_3 \end{pmatrix}$$

$t_{S_{LD}} = \max(\lambda_1 + \lambda_3, \alpha_1 + \alpha_2 + \lambda_3)$.

If $t_{S*} = \lambda_1 + \lambda_3$, then t_{S*} is equal to a lower bound and $\frac{t_{S_{LD}}}{t_{S*}} = 1$. So we may assume $t_{S*} = \alpha_1 + \alpha_2 + \lambda_3$.

Consider an optimal configuration for this problem. For the third job, the machine with λ_1 in the first rank must have a job in the third rank with processing time $= 0$, otherwise that machine will have processing time $\geq \lambda_1 + 2\lambda_3 \geq \alpha_1 + \alpha_2 + \lambda_3 = t_{S_{LD}}$. Since the machine with λ_1 in the frst rank has a job with processing time $= 0$ in the last rank, the machine with α_1 in the first rank has a job with processing time $= \lambda_3$ in the third rank. In order for t_{S*} to be less than $t_{S_{LD}}$, the machine with α_1 in the first rank must have a job with processing time $= \lambda_3$ in the second rank as well. Thus the optimal configuration must be either

$$\begin{pmatrix} \lambda_1 & \alpha_2 & 0 \\ \alpha_1 & \lambda_3 & \lambda_3 \\ \lambda_2 & \lambda_2 & \lambda_3 \end{pmatrix} \text{ or } \begin{pmatrix} \lambda_1 & \lambda_2 & 0 \\ \alpha_1 & \lambda_3 & \lambda_3 \\ \lambda_2 & \alpha_2 & \lambda_3 \end{pmatrix}$$

We consider the case where the former is the optimal configuration. Suppose $t_{S*} = \lambda_1 + \alpha_2$ then we have $\lambda_1 + \alpha_2 \geq \alpha_1 + 2\lambda_3$ and $\lambda_1 + \alpha_2 \geq 2\lambda_2 + \lambda_3$.

This leads to two more constraints

$$\begin{array}{ccccccc} -\lambda_1 & & +2\lambda_3 & +\alpha_1 & -\alpha_2 & \leq & 0, \quad (8) \\ -\lambda_1 & +2\lambda_2 & +\lambda_3 & & -\alpha_2 & \leq & 0. \quad (9) \end{array}$$

Because the processing times of the jobs can be scaled to any number we let $t_{S*} = 1$. We add the constraint

$$\lambda_1 \qquad +\alpha_2 \;\leq\; 1. \quad (10)$$

We now consider the linear program

$$\begin{array}{ll} Max & t_{S_{LD}} = \alpha_1 + \alpha_2 + \lambda_3 \\ s.t. & constraints(1),\ (2)\ ...(10) \end{array}$$

The solution will give the worst case ratio of $\frac{t_{S_{LD}}}{t_{S*}}$ for this particular case.

Solving the LP yields objective value = 1.125 = $\frac{9}{8}$ when $[\lambda_1, \lambda_2, \lambda_3, \alpha_1, \alpha_2] = \frac{1}{8}$ [5, 3, 2, 4, 3].

We now suppose $t_{S*} = \alpha_1 + 2\lambda_3$. Constraints (8) - (10) are now replaced with

$$\begin{array}{cccccccc} \lambda_1 & & -2\lambda_3 & -\alpha_1 & +\alpha_2 & \leq & 0 & (8) \\ & 2\lambda_2 & -\lambda_3 & -\alpha_1 & & \leq & 0 & (9) \\ & & 2\lambda_3 & +\alpha_1 & & \leq & 1. & (10) \end{array}$$

This new LP also yields objective value = $\frac{9}{8}$ when $[\lambda_1, \lambda_2, \lambda_3, \alpha_1, \alpha_2] = \frac{1}{8}$ [5, 3, 2, 4, 3].

We now suppose $t_{S*} = 2\lambda_2 + \lambda_3$. Constraints (8) - (10) are replaced with

$$\begin{array}{cccccccc} \lambda_1 & -2\lambda_2 & -\lambda_3 & & +\alpha_2 & \leq & 0 & (8) \\ & -2\lambda_2 & +\lambda_3 & +\alpha_1 & & \leq & 0 & (9) \\ & 2\lambda_2 & +\lambda_3 & & & \leq & 1. & (10) \end{array}$$

Again the new LP yields objective value = $\frac{9}{8}$ when $[\lambda_1, \lambda_2, \lambda_3, \alpha_1, \alpha_2] = \frac{1}{8}$ [5, 3, 2, 4, 3].

We now consider the other possible configuration as the optimal configuration. We consider each possibility for t_{S*} by changing constraints (8)-(10) to accomodate each t_{S*}. For example, when $t_{S*} = \lambda_1 + \lambda_2$, constraints (8) - (10) are

$$\begin{array}{cccccccc} -\lambda_1 & -\lambda_2 & +2\lambda_3 & +\alpha_1 & & \leq & 0 & (8) \\ -\lambda_1 & & +\lambda_3 & & +\alpha_2 & \leq & 0 & (9) \\ \lambda_1 & +\lambda_2 & & & & \leq & 1. & (10) \end{array}$$

For all three LPs for this configuration, the optimal objective value is $\frac{9}{8}$ which is attained when $[\lambda_1, \lambda_2, \lambda_3, \alpha_1, \alpha_2] = \frac{1}{8}$ [5, 3, 2, 4, 3].

We have exhausted this case and now move to the case where $\lambda_1 + \lambda_3 \geq 2\lambda_2 \geq \alpha_1 + \alpha_2$ We change constraints (6) and (7) to reflect the inequalities related to this case.

We continue in this manner and set up and solve every possible case for the 3-machine, 3-rank problem. In every case, the makespan ratio is less than or equal to 13/11. This shows

that the Coffman-Sethi conjecture holds for the 3-machine, 3-rank problem. Further, an LD makespan of 13/11 is actually attained in some cases, thus showing that the Coffman-Sethi conjecture provides a tight bound for the 3-machine, 3-rank problem. If the size of a hypothesised minimal counterexample to the Coffman-Sethi conjecture can be shown the satisfy $m \leq 3$ and $k \leq 3$, then the above analysis would show that the Coffman-Sethi conjecture holds for all problem instances.

In the next chapter, we will continue to work on reducing the size of a hypothesised minimal counterexample to the Coffman-Sethi conjecture.

Chapter 7

Properties of a hypothesised minimal counterexample to the Coffman-Sethi conjecture

The results obtained in the previous chapter showed that a hypothesised minimal counterexample to the Coffman-Sethi conjecture would have at least 3 machines, and that a hypothesised counterexample could not have 3 machines and 3 ranks. The approach proposed in Chapter 4 was applied in Chapter 6 to problem instances in which job processing times were assumed to be real numbers. A large part of the analysis in Chapter 7 focuses on problem instances in which job processing times are integers. The approach used is to check if the algorithm and problem satisfy the conditions of *Lemma* 5.3. If the conditions of *Lemma* 5.3 are satisfied, we may focus on problem instances with integer-valued parameters. As discussed in Chapter 4, we assume the existence of a minimal counterexample. We then subtract an integer-valued vector from the vector of hypothesised parameters, subject to the conditions that none of the constraints in the problem are violated. This will produce a new problem instance. The original counterexample was a minimal counterexample, therefore the new problem instance must satisfy the conjecture. This may allow us to obtain a property, i.e., a constraint or several constraints that are satisfied by the original counterexample. By repeating this process and by suitable choices of the integer-valued vectors, we keep reducing the size of the minimal counterexample.

The first three lemmas in this chapter do not use the approach described above. *Lemmas* 7.4 to 7.11 and *Theorem* 7.1 use the approach described above. The final result obtained is that a hypothesised minimal counterexample to the Coffman-Sethi conjecture has only a small number of ranks.

After each rank, there exists a set of one or more machines with a completion time equal

to the largest completion time after that rank. Lemmas 7.1 to 7.3 characterize changes in this set of machines.

Let I_h denote the set of machines with the largest completion time after rank h in the LD schedule.

Lemma 7.1. *If the Coffman-Sethi conjecture is false, every minimal counterexample of Type A with k ranks must satisfy the following:* $\exists i' \in I_{k-1}$ *such that* $i' \notin I_k$.

Proof. Suppose that the Coffman-Sethi conjecture is false. Then there exists a minimal counterexample to the Coffman-Sethi conjecture of Type A with k ranks. For a contradiction to the lemma, assume that there exists an integer r that satisfies the following: r is the smallest integer greater than or equal to 1 for which $I_h \subseteq I_{h+1}$ for $r \leq h \leq k-1$. Let $i' \in I_r$. It follows that $i' \in I_{h+1}$ for $r \leq h \leq k-1$. Let $a_i(k)_{LD}$ denote the completion times after rank k in the LD schedule, where $a_i(k)_{LD} \geq a_{i+1}(k)_{LD}$ for $1 \leq i \leq m-1$. Similarly, let $a_i^*(k)$, $1 \leq i \leq m$, denote the completion times after rank k in an optimal schedule S^*, where $a_i^*(k) \geq a_{i+1}^*(k)$ for $1 \leq i \leq m-1$. Clearly,

$$t_{\mathbb{S}_{LD}} = a_1(k)_{LD} = a_1(r)_{LD} + \mu_{r+1} + \mu_{r+2} + \ldots + \mu_k.$$

Also,

$$t_{\mathbb{S}^*} = a_1^*(k) \geq a_1^*(r) + \mu_{r+1} + \mu_{r+2} + \ldots + \mu_k.$$

It follows that, for $r < k$,

$$\frac{a_1(r)_{LD}}{a_1^*(r)} \geq \frac{a_1(k)_{LD} - (\mu_{r+1} + \mu_{r+2} + \ldots + \mu_k)}{a_1^*(k) - (\mu_{r+1} + \mu_{r+2} + \ldots + \mu_k)} \geq \frac{t_{\mathbb{S}_{LD}}}{t_{\mathbb{S}^*}}.$$

So, an instance of FM constructed from the first rm jobs has at least as bad an approximation ratio as $\frac{t_{\mathbb{S}_{LD}}}{t_{\mathbb{S}^*}}$. This contradicts the assumption that the original counterexample was a minimal counterexample. Therefore, no such r exists and the lemma is true. □

Lemma 7.2. *If the Coffman-Sethi conjecture is false, then:*
(i) In a minimal counterexample of Type A with k ranks, if there exists $s \in \{2, 3, \ldots, k-1\}$ *such that* I_{s-1} *is not a subset of* I_s *and* $I_h \subseteq I_{h+1}$ *for* $s \leq h \leq k-2$, *then*

$$s > k - \frac{4m-1}{m-1} - 1.$$

(ii) In a minimal counterexample of Type I with k ranks, if there exists $s \in \{2, 3, \ldots, k-1\}$ *such that* I_{s-1} *is not a subset of* I_s *and* $I_h \subseteq I_{h+1}$ *for* $s \leq h \leq k-2$, *then*

$$(k-1-s)\left(1 - \frac{1}{2\mu_{k-1}}\right) + \frac{(k-1-s)^2}{2\mu_{k-1}} < \left(\frac{4m-1}{m-1}\right).$$

Proof. Suppose that the Coffman-Sethi conjecture is false. Then there exists a minimal counterexample to the Coffman-Sethi conjecture of Type A with k ranks. Assume that there exists an integer s that satisfies the following: Let s be the smallest integer greater than or equal to 2 for which I_{s-1} is not a subset of I_s and I_h is a subset of I_{h+1} for $s \leq h \leq k-2$. If $s = k-1$, the results clearly hold. Assume that $s \leq k-2$. We have

$$\begin{aligned} t_{\mathbb{S}_{LD}} &= a_1(s)_{LD} + \mu_{s+1} + \mu_{s+2} + \ldots + \mu_{k-1} + (a_1(k)_{LD} - a_1(k-1)_{LD}) \\ &\leq a_1(s)_{LD} + \mu_{s+1} + \mu_{s+2} + \ldots + \mu_{k-1} + \lambda_k. \end{aligned} \tag{7.1}$$

Also,

$$t_{\mathbb{S}^*} \geq a_1^*(s) + \mu_{s+1} + \mu_{s+2} + \ldots + \mu_k. \tag{7.2}$$

For a minimal counterexample,

$$\frac{a_1(s)_{LD}}{a_1^*(s)} < \frac{t_{\mathbb{S}_{LD}}}{t_{\mathbb{S}^*}}. \tag{7.3}$$

From (7.1), (7.2), and (7.3), we deduce that

$$\frac{t_{\mathbb{S}_{LD}}}{t_{\mathbb{S}^*}} < \frac{\mu_{s+1} + \mu_{s+2} + \ldots + \mu_{k-1} + \lambda_k}{\mu_{s+1} + \mu_{s+2} + \ldots + \mu_{k-1} + \mu_k}. \tag{7.4}$$

Also, for a counterexample to the conjecture,

$$\frac{t_{\mathbb{S}_{LD}}}{t_{\mathbb{S}^*}} > \frac{5m-2}{4m-1}. \tag{7.5}$$

Note that, by *Lemma* 5.9,

$$\lambda_k = \mu_{k-1} \text{ and } \mu_k = 0. \tag{7.6}$$

From (7.4), (7.5), and (7.6), we obtain

$$\frac{\mu_{k-1}}{\mu_{s+1} + \mu_{s+2} + \ldots + \mu_{k-1}} > \frac{m-1}{4m-1}. \tag{7.7}$$

The latter is equivalent to

$$\frac{(\mu_{s+1} + \mu_{s+2} + \ldots + \mu_{k-1})}{\mu_{k-1}} < \frac{(4m-1)}{(m-1)}. \tag{7.8}$$

In a minimal counterexample to the Coffman-Sethi conjecture, every job in a rank cannot have the same processing time. (If every job in a rank has the same processing

time, that rank can be removed to obtain a smaller counterexample.) It follows that

$$\mu_{s+1} > \mu_{s+2} > \ldots > \mu_{k-1}. \tag{7.9}$$

From (7.8) and (7.9), we have

$$(k-1-s) < \frac{(4m-1)}{(m-1)}. \tag{7.10}$$

Thus,

$$s > k - \frac{(4m-1)}{(m-1)} - 1. \tag{7.11}$$

If the processing times are integers, then we have

$$\mu_h \geq \mu_{h+1} + 1 \text{ for } s \leq h \leq k-2. \tag{7.12}$$

From (7.8) and (7.12),

$$\frac{(k-1-s)\mu_{k-1} + (\frac{(k-1-s)}{2})(0 + (k-2-s)(1))}{\mu_{k-1}} < \frac{(4m-1)}{(m-1)}.$$

Hence,

$$(k-1-s)(1 - \frac{1}{2\mu_{k-1}}) + \frac{((k-1-s)^2)}{(2\mu_{k-1})} < \frac{(4m-1)}{(m-1)}.$$

□

From the above lemma, it follows that, if the Coffman-Sethi conjecture is false, a minimal counterexample of Type A must satisfy the following:

For $m = 2, s > k-8$. For $m = 3, s \geq k-6$. For $m \geq 4, s \geq k-5$.

In the following lemma, s refers to the rank that satisfies the first condition established in the previous lemma.

Lemma 7.3. *If the Coffman-Sethi conjecture is false:*
In a minimal counterexample of Type A with k ranks,
if $\exists s$ *and* $\exists r$ *that satisfy* $2 \leq r \leq s-1$ *and* $2 \leq s \leq k-1$ *such that:*
(i)I_{s-1} *is not a subset of* I_s*, and*
(ii)$I_h \subseteq I_{h+1}$ *for* $s \leq h \leq k-2$*, and*
(iii)I_{r-1} *is not a subset of* I_r*, and*
(iv)$I_h \subseteq I_{h+1}$ *for* $r \leq h \leq s-2$
then

$$r > k - \frac{2(4m-1)}{m-1} - 2.$$

Proof. Suppose that the Coffman-Sethi conjecture is false. Then there exists a minimal counterexample to the Coffman-Sethi conjecture of Type A with k ranks. Assume that there exist integers s and r that satisfy the following: Let s be the smallest integer greater than or equal to 2 for which I_{s-1} is not a subset of I_s and I_h is a subset of I_{h+1} for $s \leq h \leq k-2$. Let r be the smallest integer greater than or equal to 2 for which I_{r-1} is not a subset of I_r and I_h is a subset of I_{h+1} for $r \leq h \leq s-2$. If $r = s-1$, the claim clearly holds. Assume that $r \leq s-2$.

$$\begin{aligned}
&t_{\mathbb{S}_{LD}} \\
&= a_1(r)_{LD} + \sum_{h=r+1}^{s+1} \mu_h + (a_1(s)_{LD} - a_1(s-1)_{LD}) + \sum_{h=s+1}^{k-1} \mu_h + (a_1(k)_{LD} - a_1(k-1)_{LD}) \\
&\leq a_1(r)_{LD} + \sum_{h=r+1}^{s-1} \mu_h + \lambda_s + \sum_{h=s+1}^{k-1} \mu_h + \lambda_k. \qquad (7.13)
\end{aligned}$$

Also,

$$t_{\mathbb{S}^*} \geq a_1^*(r) + \sum_{h=r+1}^{k} \mu_h. \qquad (7.14)$$

For a minimal counterexample,

$$\frac{a_1(r)_{LD}}{a_1^*(r)} < \frac{t_{\mathbb{S}_{LD}}}{t_{\mathbb{S}^*}} \text{ and } \frac{t_{\mathbb{S}_{LD}}}{t_{\mathbb{S}^*}} > \frac{(5m-2)}{(4m-1)}. \qquad (7.15)$$

From (7.13), (7.14), and (7.15),

$$\frac{\sum_{h=r+1}^{s-1} \mu_h + \sum_{h=s+1}^{k-1} \mu_h + (\lambda_s + \lambda_k)}{\sum_{h=r+1}^{s-1} \mu_h + \sum_{h=s+1}^{k-1} \mu_h + (\mu_s + \mu_k)} > \frac{(5m-2)}{(4m-1)}. \qquad (7.16)$$

$$\Rightarrow \frac{(\lambda_s + \lambda_k) - (\mu_s + \mu_k)}{\sum_{h=r+1}^{s-1} \mu_h + \sum_{h=s+1}^{k-1} \mu_h + (\mu_s + \mu_k)} > \frac{(m-1)}{(4m-1)}. \tag{7.17}$$

Note that

$$\lambda_s = \mu_{s-1} \text{ and } \lambda_k = \mu_{k-1}. \tag{7.18}$$

Also,

$$\mu_{r+1} > \mu_{r+2} > \ldots > \mu_{s-1} \text{ and } \mu_s > \mu_{s+1} > \ldots > \mu_{k-1} > \mu_k = 0. \tag{7.19}$$

From (7.17), (7.18), and (7.19),

$$\frac{(s-1-r)\mu_{s-1} + (k-s)\mu_{k-1}}{\mu_{s-1} + \mu_{k-1}} < \frac{4m-1}{m-1}. \tag{7.20}$$

There are two possibilities:

Case 1: $s-1-r \leq k-s$. In this case, $r > s - \frac{4m-1}{m-1} - 1 \Rightarrow r > k - \frac{2(4m-1)}{m-1} - 2$.

Case 2: $s-1-r > k-s$. Therefore $\frac{(s-1-r)+(k-s)}{2} < \frac{4m-1}{m-1} \Rightarrow r > k - \frac{2(4m-1)}{m-1} - 1$. □

From the above lemma, it follows that, if the Coffman-Sethi conjecture is false, a minimal counterexample of Type A must satisfy the following:

For $m = 2, r > k - 16$. For $m = 3, r > k - 13$. For $m = 4, r > k - 12$. For $m = 5, r \geq k - 11$. For $m = 6, r \geq k - 11$. For $m \geq 7, r \geq k - 10$.

If $t_{\mathbb{S}_{LD}}$ denotes the makespan of the LD schedule for a problem instance and $t_{\mathbb{S}^*}$ denotes the optimal makespan for the same problem instance, we use the term *makespan ratio* to refer to $t_{\mathbb{S}_{LD}}/t_{\mathbb{S}^*}$.

Lemma 7.4. *(i) If the Coffman-Sethi conjecture is false, then there exists a minimal counterexample to the conjecture of Type IR.*
(ii) If the Coffman-Sethi conjecture is false, any minimal counterexample to the conjecture of Type A, I, R, or IR must satisfy the following: At least one of the jobs in the last rank (rank k) has a processing time equal to 0.

Proof. Proof of part (i): From *Corollary* 5.7 and *Corollary* 5.3, it follows that, if the Coffman-Sethi conjecture is false, there exists a minimal counterexample to the conjecture of Type IR.
Proof of part (ii): For minimal counterexamples of Type A or Type I, part (ii) follows from

Lemma 5.9. To prove part (ii) for minimal counterexamples of Type R, assume that there exists such a minimal counterexample with every job in the last rank having a processing time greater than 0. Subtract μ_k from every processing time in the last rank to obtain a counterexample with a larger $t_{\mathbb{S}_{LD}}/t_{\mathbb{S}^*}$ ratio, thus contradicting the assumption that this was a minimal counterexample. Note that subtracting μ_k from every processing time in the last rank of a rectangular schedule results in a schedule that is still rectangular. To prove part (ii) for minimal counterexamples of Type IR, assume that there exists such a minimal counterexample with every job in the last rank having a processing time greater than 0. Subtract μ_k from every processing time in the last rank to obtain a counterexample with a larger $t_{\mathbb{S}_{LD}}/t_{\mathbb{S}^*}$ ratio, thus contradicting the assumption that this was a minimal counterexample. Note that subtracting μ_k from every processing time in the last rank leaves a set of processing times that are integers. Also, subtracting μ_k from every processing time in the last rank of a rectangular schedule results in a schedule that is still rectangular. □

We define a minimal counterexample of Type IR1 as follows. If the Coffman-Sethi conjecture is false, a minimal counterexample of Type IR1 is a minimal counterexample of Type IR that has an LD schedule with the following property: Every machine with a completion time after rank k equal to the makespan has a job with processing time = λ_k in rank k, where k denotes the number of ranks.

Lemma 7.5. *If the Coffman-Sethi conjecture is false, then there exists a minimal counterexample to the conjecture of Type IR1, and every minimal counterexample of Type IR is a minimal counterexample of Type IR1.*

Proof. Consider a minimal counterexample P1 of Type IR. By *Lemma* 7.4, such a counterexample exists if the Coffman-Sethi conjecture is false. Now apply the following two-step process. Step 1: Construct a new problem instance P2 as follows. Subtract 1 time unit from the processing time of every job in rank $k-1$. Also subtract 1 time unit from the processing time of every job in rank k that has a processing time of λ_k. Note that, for a minimal counterexample, all processing times in rank k are not equal, therefore there exist processing times in rank k that are less than λ_k. Leave these processing times unchanged. Note that the assignment of jobs in each rank to machines in the LD schedule remains unchanged. Clearly, if t_{S^*} denotes the optimal makespan of problem instance P1, problem P2 has an optimal makespan that is equal to $t_{S^*} - 1$. Step 2: For every job in rank 1 of the optimal schedule for P2 that is processed on a machine with a completion time after rank k that is less than the makespan, increase the processing time so that the completion time after rank k becomes equal to the makespan. This produces a problem instance P2R of Type IR. P1 is a minimal counterexample of Type IR. It follows that P2R has a smaller makespan ratio than P1. P2R will have an optimal makespan that is equal to the optimal makespan of P2. Therefore P2R has an optimal makespan that is equal to $t_{S^*} - 1$. It follows that the makespan of the LD schedule for problem instance P2R is less

than or equal to $t_{S_{LD}} - 2$, where $t_{S_{LD}}$ denotes the LD makespan of problem instance P1. From *Lemma* 5.7, the LD makespan of P2R cannot be less than the LD makespan of P2. Therefore the makespan of the LD schedule for problem instance P2 is less than or equal to the LD makespan for P1 -2. This implies that, in the LD schedule for P1, every machine with a completion time after rank k equal to the makespan has a job with processing time λ_k in rank k. Therefore P1 is a minimal counterexample of Type IR1. □

We define a problem instance, a counterexample and a minimal counterexample of Type I1 as follows. A problem instance of Type I1 is a problem instance of Type I that has an LD schedule with the following properties:
(i) It has only one machine i' with a completion time after rank k equal to the makespan.
(ii) Machine i' has a processing time equal to λ_{k-1} in rank $k-1$ and λ_k in rank k, where k denotes the number of ranks.
If the Coffman-Sethi conjecture is false, a counterexample to the conjecture of Type I1 is a counterexample of Type I that has an LD schedule with the following properties:
(i) It has only one machine i' with a completion time after rank k equal to the makespan.
(ii) Machine i' has a processing time equal to λ_{k-1} in rank $k-1$ and λ_k in rank k, where k denotes the number of ranks.

A minimal counterexample of Type I1 is a counterexample of Type I1 for which there does not exist a smaller counterexample of Type I1.

Lemma 7.6. *If the Coffman-Sethi conjecture is false, then there exists a minimal counterexample to the conjecture of Type I1.*

Proof. Consider a minimal counterexample P1 of Type IR1. By *Lemma* 7.5, such a counterexample exists if the Coffman-Sethi conjecture is false. Now construct a new problem instance P2 as follows. Subtract 1 time unit from the processing time of every job in rank $k-2$. Also subtract 1 time unit from the processing time of every job in rank $k-1$ that has a processing time of λ_{k-1}. Note that, for a minimal counterexample, all processing times in rank $k-1$ are not equal, therefore there exist processing times in rank $k-1$ that are less than λ_{k-1}. Leave these processing times unchanged. Note that the assignment of jobs in each rank to machines in the LD schedule remains unchanged. Problem instance P1 had an optimal rectangular schedule. Reducing the processing time of every job in rank $k-2$ by 1 leaves the rectangular property unchanged and results in a reduction of 1 in the optimal makespan. A further reduction of 1 in one or more, but not all, jobs in rank $k-1$ results in no further reduction in the optimal makespan. This follows from the fact that any reduction in the optimal makespan of a schedule that is initially rectangular requires the sum of processing times to be reduced by at least m. Thus, if t_{S^*} denotes the optimal makespan of problem instance P1, problem P2 has an optimal makespan equal to $t_{S^*} - 1$.

Now construct a problem instance P2R of Type IR by adding 1 unit to the processing time of every job in rank 1 that is performed on a machine with completion time after rank k in the optimal schedule that is less than the makespan. The optimal makespan of P2R is equal to the optimal makespan of P2. Let $t_{S_{LD}}$ denote the LD makespan of problem instance P1. P1 is a minimal counterexample of type IR. Therefore, the makespan of the LD schedule for problem instance P2R is less than or equal to the makespan of the LD schedule for P1 -2. From *Lemma* 5.7, the LD makespan of P2R cannot be less than the LD makespan of P2. Therefore the makespan of the LD schedule for problem instance P2 is less than or equal to the LD makespan for P1 -2. This implies that one of the following is true: (i) In the LD schedule for P1, there exists a machine i' with a completion time after rank k equal to the makespan and with a job with processing time λ_{k-1} in rank $k-1$. (If more than one such machine exists, arbitrarily select one of those machines and label it i'.) From the preceding lemma, this machine has a job with processing time λ_k in rank k. (ii) In the LD schedule for P1, there does not exist any machine with a completion time after rank k equal to the makespan and with a job with processing time λ_{k-1} in rank $k-1$. However, there must exist machines i' and i'', where i' is not equal to i'', machine i' and machine i'' have the same completion time after rank $k-1$, machine i' has a completion time after rank k equal to the makespan and a job with processing time less than λ_{k-1} in rank $k-1$, and machine i'' has a job with processing time equal to λ_{k-1} in rank $k-1$ and a completion time after rank k that is less than the makespan. From the previous lemma, machine i' has a job with processing time that is equal to λ_k in rank k. Machine i'' has a lower completion time after rank k and the same completion time after rank $k-1$ as machine i'. Therefore machine i'' has a job with processing time that is less than λ_k in rank k.

In case (ii), swap the jobs assigned to machines i' and i'' in rank k of the LD schedule and swap the labels assigned to those machines. The resulting schedule is a valid LD schedule. It follows that, for a minimal counterexample P1 of Type IR1, there exists an LD schedule with a machine i' with a completion time after rank k equal to the makespan, a job with processing time λ_k in rank k, and job with processing time λ_{k-1} in rank $k-1$.

Now construct a new counterexample P3 as follows. In the LD schedule for P1, for every machine $i \neq i'$ with a completion time after rank k equal to the makespan, delete the job in rank k. The makespan and the set of jobs assigned to i' in the LD schedule for P3 are the same as those in the LD schedule for P1. The makespan of the optimal schedule for P3 is less than or equal to the makespan of the optimal schedule for P1. However, the optimal schedule for P3 may not be rectangular. P3 is clearly a counterexample to the conjecture of Type I1.

Thus, there exists a counterexample to the conjecture of Type I1. It follows that there exists a minimal counterexample to the conjecture of Type I1. □

We define a problem instance, a counterexample and a minimal counterexample of Type I2 as follows. A problem instance of Type I2 is a problem instance of Type I1 with the following property: The machine i' with a completion time equal to the makespan has a processing time equal to λ_r in rank r for $2 \leq r \leq k$. If the Coffman-Sethi conjecture is false, a counterexample to the conjecture of Type I2 is a minimal counterexample of Type I1 with the following property: The machine i' with a completion time equal to the makespan has a processing time equal to λ_r in rank r for $2 \leq r \leq k$. A minimal counterexample of Type I2 is a counterexample of Type I2 for which there does not exist a smaller counterexample of Type I2.

Lemma 7.7. *If the Coffman-Sethi conjecture is false, there exists a minimal counterexample to the conjecture of Type I2.*

Proof. The statement is proved using induction. The induction assumption is that there exists a minimal counterexample P3′ of Type I1 that has an LD schedule with a machine i' that has a processing time equal to λ_r in rank r for $h \leq r \leq k$, where h is an integer greater than or equal to 3. Construct P3″ from P3′ by subtracting 1 time unit from the processing time of every job in rank $h-2$, and subtracting 1 time unit from the processing time of every job in rank $h-1$ that has a processing time of λ_{h-1}. Leave the remaining processing times unchanged. Note that P3″ has integer processing times and a single machine with the completion time equal to the makespan, and that machine has a job with processing time equal to the largest processing time in ranks k and $k-1$. If P3″ were a counterexample to the conjecture, it would be a counterexample of Type I1, thus contradicting the assumption that P3′ is a minimal counterexample of Type I1. It follows that P3″ satisfies the Coffman-Sethi conjecture. If t_{S^*} denotes the optimal makespan of P3′, problem P3″ has an optimal makespan that is less than or equal to $t_{S^*} - 1$. If $t_{S_{LD}}$ denotes the makespan of the LD schedule for P3′, the makespan of the LD schedule for P3″ must be less than or equal to $t_{S_{LD}} - 2$. Clearly, one of the following must hold: (i) In the LD schedule for P3′, machine i' has a job with processing time equal to λ_{h-1} in rank $h-1$. (ii) In the LD schedule for P3′, machine i' has a job with processing time less than λ_{h-1} in rank $h-1$. However, there exists a machine i'' with the same completion time after rank $h-1$ as machine i' and a job with processing time equal to λ_{h-1} in rank $h-1$ and a completion time after rank k that is less than the makespan. In this case, for ranks $h, h+1, \ldots, k$, swap the jobs assigned to machines i' and i''. After the swap of jobs is completed, swap the labels i' and i'' assigned to the two machines. Clearly, after the swap, the schedule continues to be an LD schedule. After the swap, machine i' is the only machine with a completion time after rank k equal to the makespan and machine i' has a processing time equal to λ_r in rank r for $h-1 \leq r \leq k$.

In both cases, there exists a counterexample P3′ with a machine i' that has a processing time equal to λ_r in rank r for $h-1 \leq r \leq k$. The base case of the induction corresponding

to the situation in which $h = k - 1$ follows from the previous lemma. This proves that there exists a counterexample to the conjecture of Type I2. It follows that there exists a minimal counterexample to the conjecture of Type I2. □

Lemma 7.8. *If the Coffman-Sethi conjecture is false, in a minimal counterexample of Type I2, the sole machine i' with a completion time after rank k equal to the makespan has a processing time equal to μ_1 in rank 1.*

Proof. If the Coffman-Sethi conjecture is false, it follows from the previous lemma that there exist one or more minimal counterexamples to the conjecture of Type I2. Let P3′ denote one of these minimal counterexamples. If, in the LD schedule for P3′, there exists only one machine with a processing time equal to λ_r in rank r for $2 \leq r \leq k$, the lemma clearly holds. Assume that there exists a set of two or more machines with a processing time equal to λ_r in rank r for $2 \leq r \leq k$. Let i_1 denote the machine with the largest processing time in rank 1 in this set of machines. If there exist two or more machines with the largest processing time in rank 1, arbitrarily select one of these machines to be machine i_1. Let $i_2 \neq i_1$ denote another machine with a processing time equal to λ_r in rank r for $2 \leq r \leq k$. Let $\mathbb{S}^*$ denote the optimal schedule for P3′. Find the machine i_3 in $\mathbb{S}^*$ with the same job assigned to it in rank 1 as is assigned to machine i_2 in the LD schedule. For rank $r = 2, 3, \ldots, k$, do the following:
Step 1: For machine i_3, if the processing time of the job assigned to rank r in $\mathbb{S}^*$ is λ_r, delete that job from $\mathbb{S}^*$ and from rank r of machine i_2 in P3′.
Step 2: For machine i_3, if the processing time of the job assigned to rank r in $\mathbb{S}^*$ is less than λ_r, find a machine i_4 in $\mathbb{S}^*$ with a processing time equal to λ_r in rank r. Delete the job with processing time equal to λ_r from the set of jobs assigned to machine i_4 in $\mathbb{S}^*$ and from rank r of machine i_2 in P3′. Transfer the job in rank r of $\mathbb{S}^*$ that was originally assigned to machine $i - 3$ from machine i_3 to machine i_4.
After completing steps 1 and 2 above for $r = 2, 3, \ldots, k$, do the following: Step 3: Delete the job assigned to machine i_2 in rank 1 of the LD schedule and to machine i_3 in rank 1 of $\mathbb{S}^*$.

Note that each step in this process will either reduce the makespan of $\mathbb{S}^*$ or will leave it unchanged. When the process is completed, all jobs that were originally assigned to i_3 in $\mathbb{S}^*$ would have been moved to another machine or deleted from $\mathbb{S}^*$. All the jobs that were assigned to machine i_2 in the LD schedule have been deleted. We now delete machine $i3$ from $\mathbb{S}^*$ and machine i_2 from the LD schedule. Clearly, the new problem instance is also a problem instance of Type I2. It has the same LD makespan as the original problem instance and an optimal makespan that is less than or equal to that of the original problem instance. The number of machines in the new problem instance is less than the number of machines in P3′. This contradicts the assumption that P3′ is a minimal counterexample of Type I2. It follows that, in the LD schedule for P3′, only one machine has a processing

time equal to λ_r in rank r for $r = 2, 3, \ldots, k$. From the previous lemma, this machine has a completion time after rank k that is equal to the makespan. From the mechanics of the LD algorithm, it is clear that this machine has a job with a processing time that is equal to μ_1 in rank 1. □

Lemma 7.9. *If the Coffman-Sethi conjecture is false, a minimal counterexample to the conjecture of Type I2 with k ranks has an optimal schedule that satisfies the following:*
(i) At least one of the machines with a completion time equal to the makespan has a processing time equal to λ_k in rank $k-1$ and a processing time equal to 0 in rank k, or
(ii) At least one of the machines with a completion time equal to the makespan has a processing time equal to λ_k in rank k, or
(iii) At least one of the machines with a completion time equal to the makespan has a processing time equal to 0 in rank k and a job in rank $k-1$ which, in the LD schedule for the original counterexample, has the same completion time after rank $k-1$ as the job assigned to rank $k-1$ on machine i'. i' refers to the only machine in the original counterexample with a completion time after rank k equal to the makespan.

Proof. From *Lemma* 7.7, if the Coffman-Sethi conjecture is false, there exists a minimal counterexample to the conjecture of Type I2. Note that the optimal schedule for this minimal counterexample is not guaranteed to be rectangular. Assume that, in the optimal schedule for the counterexample, every machine with a completion time equal to the makespan has a job with processing time greater than 0 in the last rank. In the LD schedule for the counterexample, there exists only one machine i' with a completion time equal to the makespan and that machine has a processing time equal to λ_k in the last rank, where k is the number of ranks. Subtract 1 from the processing time of every job in rank k that has a processing time greater than 0. Clearly, the assignment of jobs to machines in each rank of the optimal schedule and the LD schedule remain unchanged. λ_k gets reduced by 1, and this results in a reduction of 1 in the makespan of the LD schedule. The makespan of the optimal schedule also gets reduced by 1. This follows from the assumption that every machine with a completion time equal to the makespan has a job with processing time greater than 0 in the last rank. The new problem instance obtained after the reduction in processing times is a problem instance of Type I2. For the new problem instance, the ratio of the makespan of the LD schedule to the makespan of the optimal schedule is $(t_{S_{LD}} - 1)/(t_{S^*} - 1)$, where $t_{S_{LD}}$ denotes the makespan of the LD schedule and t_{S^*} denotes the makespan of the optimal schedule prior to the change in processing times. This contradicts the assumption that the original counterexample was a minimal counterexample of Type I2.
Therefore, in the optimal schedule for the minimal counterexample, at least one machine with a completion time equal to the makespan has a job with processing time equal to 0 in the last rank. Subtract 1 from the processing time of every job that, in the optimal schedule, is assigned to rank $k-1$ or rank k and has a completion time equal to the

makespan. If one of these jobs has a processing time equal to λ_k in rank $k-1$, subtract 1 from the processing time of every job in rank k that has a processing time equal to λ_k. The makespan of the optimal schedule gets reduced from t_{S^*} to $t_{S^*} - 1$.
If the new problem instance resulting from the reduction in processing times is not a problem instance of Type I2, it follows that the new problem instance cannot have an LD schedule in which only a single machine has a completion time after rank k equal to the makespan and in which the LD makespan is equal to $\mu_1 + \lambda_2 + \lambda_3 + \ldots + \lambda_k$, where μ_r denotes the smallest processing time and λ_r denotes the largest processing time in rank r. It follows that the new problem instance has an LD schedule with a makespan that is less than the LD makespan of the original problem instance. If the new problem instance resulting from the reduction in processing times is a problem instance of Type I2, it follows that, to ensure that the original problem instance was a minimal counterexample of Type I2, the makespan of the LD schedule of the new problem instance is less than the LD makespan of the original problem instance -1. In both cases, the LD makespan of the new problem instance is less than the LD makespan of the original problem instance. A reduction in the LD makespan occurs only if, in the optimal schedule for the original counterexample, one of the following holds:
(i) At least one of the machines with a completion time equal to the makespan has a processing time equal to λ_k in rank $k-1$ and a processing time equal to 0 in rank k, or
(ii) At least one of the machines with a completion time equal to the makespan has a processing time equal to λ_k in rank k, or
(ii) At least one of the machines with a completion time equal to the makespan has a processing time equal to 0 in rank k and a job in rank $k-1$ which, in the LD schedule for the original counterexample, has the same completion time after rank $k-1$ as the job assigned to rank $k-1$ on machine i'. □

Lemma 7.10. *If the Coffman-Sethi conjecture is false, a minimal counterexample to the conjecture of Type I2 with k ranks has an LD schedule that satisfies the following:*
(i) Every job in rank k has a processing time equal to either λ_k or 0.
(ii) Every job in rank $k-1$ with a processing time p that is greater than μ_{k-1} but less than λ_{k-1} satisfies one of the following: (a) It has a completion time after rank $k-1$ that is equal to the completion time after rank $k-1$ on machine i', or (b) It has a completion time after rank $k-1$ that is less than the completion time after rank $k-1$ on machine i', but there exists another job in rank $k-1$ with a processing time p that has a completion time after rank $k-1$ that is equal to the completion time after rank $k-1$ on machine i', where i' is the only machine with a completion time after rank k equal to the makespan.

Proof. Part (i) of the lemma can be proved in two ways. The first proof is the proof of *Lemma* 5.9. The second proof is the following. Consider a minimal counterexample of Type I2 that has k ranks. From *Lemma* 7.4, $\mu_k = 0$ for any minimal counterexample of Type I. Assume that there exists at least one job in rank k with a processing time

that is greater than 0 and less than λ_k. Subtract 1 from the processing time of every job in rank k that has a processing time that is greater than 0 and less than λ_k. From the preceding lemmas, a minimal counterexample of Type I2 has an LD schedule with only one machine with a completion time after rank k equal to the makespan, and that machine has a processing time of μ_1 in rank 1 and a processing time of λ_r in rank r for $2 \leq r \leq k$. Clearly, this machine has the same set of jobs assigned to it in the LD schedule even after the reduction in processing times in rank k. Thus the makespan of the LD schedule remains the same even after the reduction in processing times.

Clearly, the reduction in processing times leads either to no change in the optimal makespan or to a reduction of 1 unit in the optimal makespan. In both cases, the new problem instance with the reduced processing times clearly has a makespan ratio that is greater than or equal to the makespan ratio for the original counterexample. This contradicts the assumption that the original counterexample was a minimal counterexample.

It follows that there does not exist any job in rank k with a processing time that is greater than 0 and less than λ_k. This completes the proof of part (i) of the lemma.

Part (ii) of the lemma can be proved as follows. Consider a minimal counterexample to the conjecture of Type I2. Consider a job in rank $k-1$ of the LD schedule for the counterexample with a processing time p that is greater than μ_{k-1} but less than λ_{k-1}. If the job is processed on machine i'', the completion time after rank $k-2$ on machine i'' must be greater than or equal to the completion time after rank $k-2$ on machine i'. Reduce the processing time of the job by 1. This could result in a new problem instance of Type I2 or in a new problem instance that is not of Type I2. In both cases, the LD makespan of the new problem instance must be less than the LD makespan of the original problem instance. If neither condition (a) nor condition (b) hold, the reduction in processing time has no impact on the LD makespan. It follows that at least one of these conditions must hold. □

Lemma 7.11. *If the Coffman-Sethi conjecture is false, a minimal counterexample to the conjecture of Type I2 with k ranks has a set of jobs that satisfy the following:*

$$(5m-2)\lambda_1 < (8m-2)\lambda_2 - (m-1)(\lambda_3 + \lambda_4 + \ldots + \lambda_k).$$

Proof. Consider a minimal counterexample of Type I2 that has k ranks. From *Lemma* 7.8, it follows that $t_{S_{LD}} = \mu_1 + \lambda_2 + \lambda_3 + \ldots + \lambda_k$. Also, $t_{S^*} \geq \lambda_1 + \mu_2 + \mu_3 + \ldots + \mu_{k-2} + \mu_{k-1} + \mu_k$. μ_k is equal to zero and μ_h is equal to λ_{h+1} for $1 \leq h \leq k-1$. It follows that $\frac{t_{S_{LD}}}{t_{S^*}} \leq \frac{2\lambda_2+\lambda_3+\lambda_4+\ldots\lambda_k}{\lambda_1+\lambda_3+\lambda_4+\ldots+\lambda_{k-1}+\lambda_k}$. Also, $\frac{t_{S_{LD}}}{t_{S^*}} > \frac{5m-2}{4m-1}$.

This completes the proof of the lemma. □

Theorem 7.1. *If the Coffman-Sethi conjecture is false, then every minimal counterexample to the conjecture of Type IR or I or I2 satisfies* $t_{\mathbb{S}_{LD}}/t_{\mathbb{S}^*} < k/(k-1)$.

Proof. Proof for minimal counterexamples of Type IR:
Consider a minimal counterexample P1 of Type IR. Now apply the following two-step process. Step 1: Reduce each nonzero processing time by 1 to construct a new problem instance P2. Clearly, the assignment of jobs in each rank to machines can be kept unchanged for the new LD schedule. The schedule that was originally an optimal rectangular schedule for P1 will not be rectangular but will be optimal for P2 after the reduction in processing times. In the modified schedule, there are only $(k-1)$ jobs assigned to the one or more machines with a zero processing time job in the last rank. This implies that the new optimal makespan is equal to $t_{S^*} - (k-1)$. Step 2: For every job in rank 1 of the optimal schedule for P2 that is processed on a machine with a completion time after rank k that is less than the makespan, increase the processing time so that the completion time after rank k becomes equal to the makespan. This produces a problem instance P2R of Type IR. P2R will have an optimal makespan that is equal to the optimal makespan of P2. From *Lemma* 5.7, it follows that the LD makespan of P2R cannot be less than the LD makespan of P2.
The optimal makespan of P2R is equal to $t_{S^*} - (k-1)$. For the LD schedule for P2R, there are two possibilities: (i) $t_{S_{LD}}$ gets reduced to a value that is greater than or equal to $t_{S_{LD}} - (k-1)$. This results in a new problem instance with a makespan ratio that is at least $(t_{S_{LD}} - (k-1))/(t_{S^*} - (k-1))$. This new ratio is larger than $t_{S_{LD}}/t_{S^*}$, thus contradicting the assumption that the original problem instance was a minimal counterexample of Type IR. (ii) $t_{S_{LD}}$ gets reduced to $t_{S_{LD}} - k$. This results in a new problem instance with a makespan ratio $\geq (t_{S_{LD}} - k)/(t_{S^*} - (k-1))$. The original problem instance P1 was a minimal counterexample of Type IR. This implies that $(t_{S_{LD}} - k)/(t_{S^*} - (k-1))$ is less than $t_{S_{LD}}/t_{S^*}$. Therefore, $t_{S_{LD}}/t_{S^*} < (k/(k-1))$. This completes the proof for minimal counterexamples of Type IR.
Proof for minimal counterexamples of Type I:
Consider a minimal counterexample P1′ of Type I. Now reduce each nonzero processing time by 1 to construct a new problem instance P2′ with integer processing times. Clearly, the assignment of jobs in each rank to machines can be kept unchanged for the new LD schedule and the new optimal schedule. The new optimal makespan is less than or equal to $t_{S^*} - (k-1)$. For the new LD schedule, there are two possibilities: (i) $t_{S_{LD}}$ gets reduced to a value that is equal to $t_{S_{LD}} - (k-1)$. This results in a new problem instance with a makespan ratio that is at least $(t_{S_{LD}} - (k-1))/(t_{S^*} - (k-1))$. This new ratio is larger than $t_{S_{LD}}/t_{S^*}$, thus contradicting the assumption that the original problem instance was a minimal counterexample of Type I. (ii) $t_{S_{LD}}$ gets reduced to $t_{S_{LD}} - k$. This results in a new problem instance with a makespan ratio that is greater than or equal to $(t_{S_{LD}} - k)/(t_{S^*} - (k-1))$. The original problem instance P1′ was a minimal counterexample of Type I. This implies that $(t_{S_{LD}} - k)/(t_{S^*} - (k-1))$ is less than $t_{S_{LD}}/t_{S^*}$. Therefore, $t_{S_{LD}}/t_{S^*} < (k/(k-1))$. This completes the proof for minimal counterexamples of Type I.

Proof for minimal counterexamples of Type I2:
Consider a minimal counterexample P1″ of Type I2. Now reduce each nonzero processing time by 1 to construct a new problem instance P2″. Clearly, the assignment of jobs in each rank to machines can be kept unchanged for the new LD schedule and the new optimal schedule. The new problem instance P2″ is a problem instance of Type I2. The new optimal makespan is less than or equal to $t_{S^*} - (k-1)$. For the new LD schedule, there are two possibilities: (i) $t_{S_{LD}}$ gets reduced to a value that is equal to $t_{S_{LD}} - (k-1)$. This results in a new problem instance with a makespan ratio that is at least $(t_{S_{LD}} - (k-1))/(t_{S^*} - (k-1))$. This new ratio is larger than $t_{S_{LD}}/t_{S^*}$, thus contradicting the assumption that the original problem instance was a minimal counterexample of Type I2. (ii) $t_{S_{LD}}$ gets reduced to $t_{S_{LD}} - k$. This results in a new problem instance with a makespan ratio that is greater than or equal to $(t_{S_{LD}} - k)/(t_{S^*} - (k-1))$. The original problem instance P1′ was a minimal counterexample of Type I2. This implies that $(t_{S_{LD}} - k)/(t_{S^*} - (k-1))$ is less than $t_{S_{LD}}/t_{S^*}$. Therefore, $t_{S_{LD}}/t_{S^*} < (k/(k-1))$. This completes the proof for minimal counterexamples of Type I2. □

It follows that, if the Coffman-Sethi conjecture is false, a minimal counterexample of Type IR or I or I2 must satisfy the following: $k/(k-1) > (5m-2)/(4m-1)$
$\Rightarrow 1/(k-1) > (m-1)/(4m-1)$
$\Rightarrow k < (5m-2)/(m-1)$.
For $m = 2$, k must be less than or equal to 7. For $m = 3$, k must be less than or equal to 6. For $m \geq 4$, k must be less than or equal to 5. Thus, if the Coffman-Sethi conjecture is false, a minimal counterexample to the conjecture of Type IR or I or I2 has only a small number of ranks.

Chapter 8

Directions for future research

The most immediate potential direction for future research would be the application of the approaches outlined in this thesis to conjectured worst-case makespan ratios for other fast and simple algorithms for the FM problem. For the LI algorithm, it might be possible to obtain an alternative proof of the worst-case makespan ratio. For the LPT*, PIA and GAP algorithms, the first step would be to develop a conjecture for the makespan ratio. An attempt could then be made to prove or disprove these conjectures using techiques similar to those used for the LD algorithm.

It might also be possible to prove performance bounds for algorithms for other problems and algorithms, particularly other algorithms for scheduling and constrained minimisation. These include problems in related areas such as bin packing and makespan minimisation. If a loose upper bound is available, it might be possible to use a binary search procedure to construct a sequence of proofs that lead to a continual tightening of the upper bound. For any algorithm for the FM problem, the starting point could be a lower bound of 1 and an upper bound of 3/2, followed by a lower bound of 1 and an upper bound of 5/4, etc. For the LD algorithm, this procedure could have resulted in a set of proofs that showed that the upper bound lay between 9/8 and 5/4.

The novel approach of treating the parameters of the problem (job processing times in this case) as variables and subsequently setting up the performance ratio maximisation problem as a set of linear programs could also be potentially applied to other combinatorial optimization problems. Further, many reasonably simple algorithms and objective functions would satisfy the continuity requirements needed to discretize the problem. For most problems in combinatorial optimization, being able to focus entirely on integer or rational parameters should make it easier to determine a bound on the performance ratio.

Bibliography

Bruno, J., E. G. Coffman and R. Sethi, Algorithms for minimizing mean flow time, *INFORMATION PROCESSING '74: Processings of the IFIPS Conference*, 504–510, North-Holland Publishing Company, 1974.

Cao, F., Determining the performance ratio of algorithm MULTIFIT for scheduling, in *Nonconvex Optimization Applications, Volume 4: Minimax and applications*, Kluwer Academic Publishers, Dordrecht, 79–96, 1995.

Chang, S. Y. and H. C. Hwang, The worst-case analysis of the MULTIFIT algorithm for scheduling nonsimultaneous parallel machines, *Discrete Applied Mathematics*, 92(2–3), 135–147, 1999.

Coffman, E. G. M. R. Garey and D. S. Johnson, An application of bin-packing to multiprocessor scheduling. *SIAM Journal of Computing* 1, 1–17, 1978.

Coffman, E. G., M. R. Garey and D. S. Johnson, Dynamic bin packing, *SIAM Journal of Computing*, 12, 227–258, 1983.

Coffman, E. G. and R. Sethi, Algorithms minimizing mean flowtime: schedule-length properties, *Acta Informatica* 6, 1–14, 1976.

Coffman, E. G. and R. Sethi, A generalized bound on LPT sequencing, *Proceedings of the 1976 ACM SIGMETRICS Conference on Computer Performance Modeling Measurement and Evaluation*, 306-310, 1976.

Coffman, E. G. and M. Yannakakis, Permuting elements within columns of a matrix in order to minimize maximum row sum, *Mathematics of Operations Research*, 9(3), 384–390, 1984.

Conway, R. W., W. L. Maxwell and L. W. Miller, *Theory of Scheduling*, Addi-

son Wesley, MA, USA, 1967.

Dosa, G. Generalized multifit-type methods II, *Alkalmaz. Mat. Lapok*, 20(1) (2000), 91–111, 2000.

Dosa, G. Generalized multifit-type methods for scheduling parallel identical machines, *Pure Mathematics and Applications*, 12(3), 287–296, 2001.

Eck, B. T. and M. Pinedo, On the minimization of the makespan subject to flowtime optimality, *Operations Research*, 41(4), 1993.

Friesen, D. K., Tighter bounds for the MULTIFIT processor scheduling algorithm, *SIAM Journal of Computing* 13, 179–181, 1984.

Garey, M. R. and D. S. Johnson, Approximation algorithms for bin packing problems - a survey , in *Analysis and Design of Algorithms in Combinatorial Optimization*, G. Ausiello and M. Lucertini, (eds.), Springer-Verlag, New York, 147–172, 1981.

Graham, R. L., Bounds for certain multiprocessing anomalies, *Bell Systems Technical Journal*, 45(9), 1563–1581, 1966.

Graham, R. L., Bounds on multiprocessing timing anomalies, *SIAM Journal of Applied Mathematics*, 17(2), 416–429, 1969.

Graham, R. L., Bounds on multiprocessing anomalies and related packing algorithms, *Proceedings of the AFIPS 1972 Spring Joint Computer Conference*, 40, 1–14, AFIPS Press, Arlington, VA, 1972.

Graham, R. L., E. L. Lawler, J. K. Lenstra and A. H. G. Rinnooy Kan, Optimization and approximation in deterministic sequencing and scheduling: a survey, *Annals of Discrete Mathematics*, 4, 287–326, 1979.

Gupta, J. N. D. and J. C. Ho, Minimizing makespan subject to minimum flowtime on two identical parallel machines, *Computers and Operations Research*, 28, 705–717, 2001.

Hardy, G. H., J. E. Littlewood and G. Polya, *Inequalities*, Cambridge University Press, 1934.

Ho, J. C. and J. S. Wong, Makespan minimization for m parallel identical processors, *Naval Research Logistics*, 42, 935–948, 1995.

Hochbaum, D. S. and D. B. Shmoys, Using Dual Approximation Algorithms for Scheduling Problems: Theoretical and Practical Results, *Journal of the ACM*, 34(1), 144–162, 1987.

Huang, M. and L. Tunçel, Approximation Algorithms for Minimizing Makespan of Flow-Time Optimal Schedules in the Basic Parallel Identical Machine Model, Working paper, Dept. of Combinatorics and Optimization, Faculty of Mathematics, University of Waterloo, Waterloo, Canada, 2004.

Johnson, D. S., Near optimal bin packing algorithms, *Ph.D. thesis*, Mathematics Dept., MIT, Cambridge, MA, 1973.

Johnson, D. S., A. Demers, J. D. Ullman, M. R. Garey and R. L. Graham, Worst-case performance bounds for simple one-dimensional packing algorithms, *SIAM Journal of Computing*, 3, 299–325, 1974.

Lawler, E. L., J. K. Lenstra, A. H. G. Rinnooy Kan and D. B. Shmoys, Sequencing and scheduling: algorithms and complexity, *The Handbooks of Operations Research and Management Science, Volume IV: Logistics of Production and Inventory*, S.C. Graves, A.H.G. Rinnooy Kan, P. Zipkin, (eds.), North-Holland, 445–522, 1993.

Lin, C. H. and C. J. Liao, Makespan minimization subject to flowtime optimality on identical parallel machines, *Computers and Operations Research*, 31, 1655–1666, 2004.

de la Vega, W. F. and G. S. Lueker, Bin packing can be solved within $1 + \epsilon$ in linear time, *Combinatorica*, 1(4),349–355, 1981.

Weierstrass, K., Abhandlungen aus der Functionenlehre. Berlin: J. Springer, 97, 1886.

Yue, M. Y., On the exact upper bound for the multifit processor scheduling algorithm. *Annals of Operations Research*, 24, 1–4, 233–259, 1990.

Yue., M., H. Kellerer and Z. Yu, A simple proof of the inequality $R_M(MF(k)) \leq 1.2 + 1/2^k$ in multiprocessor scheduling, *Report No. 124, Institut fur Mathematik, Technische Universitat Graz*, 1–10, 1988.