	T/2	T	3T/2
P1	$\{t_{1,i} \mid i \varepsilon u\}$	$t_{1,3n+1}$	$\{t_{1,i} \mid i \varepsilon u\}$
P2	$t_{2,3n+1}$	$\{t_{2,i} \mid n+1 \le i \le 2n\}$	
P3	$\{t_{3,i} \mid 2n+1 \le i \le 3n\}$		$t_{3,3n+1}$

Figure 5.4.1

b) If S has no partition then all schedules for OS must have a finish time $> 3T/2$.

This is shown by contradiction. Assume that there is a schedule for OS with finish time $\le 3T/2$. Since $t_{1,3n+1} = t_{2,3n+1} = t_{3,3n+1} = T/2$, it follows that in this schedule job $3n+1$ must be being processed at all times. Since the schedule is nonpreemptive, there must be a processor j such that $t_{j,3n+1}$ begins at time $T/2$ and finishes at time T. For this processor, there is a set of jobs with $t_{j,i}$, $(j-1)n+1 \le i \le jn$ and

$$\sum_{(j-1)n+1}^{jn} t_{j,i} = T.$$

Since S has no partition, it follows that all the $T/2$ units of time preceding $t_{j,3n+1}$ on processor j cannot be used. Hence more than $T/2$ are needed after time T to complete the remaining tasks. Hence the finish time must be $> 3T/2$. This contradicts our assumption regarding the schedule. There is therefore no schedule with finish time $\le \tau = 3T/2$ when S has no partition. ∎

When $m > 3$ the proof of Lemma 5.4.1 can be strengthened to the case when each job has atmost 2 tasks.

Lemma 5.4.2 If LOFT is polynomial solvable for $m > 3$ then so also is PARTITION.

Proof (using only 2 tasks per job)

From the partition problem $S = \{a_1, a_2, \ldots, a_n\}$ the following open shop problem, OS, with $n+2$ jobs, $m = 4$ machines and all jobs having at most 2 nonzero tasks is constructed:

$$t_{1,i} = \varepsilon/n, \quad t_{2,i} = a_i, \quad t_{3,i} = t_{4,i} = 0; 1 \leq i \leq n$$
$$t_{1,n+1} = T/2, \quad t_{2,n+1} = t_{4,n+1} = 0, \quad t_{3,n+1} = T/2+\varepsilon$$
$$t_{1,n+2} = T/2, \quad t_{2,n+2} = t_{3,n+2} = 0, \quad t_{4,n+2} = T/2+\varepsilon$$

where $T = \sum_{1}^{n} a_i$, $\tau = T + \varepsilon$ and $0 < \varepsilon < 1$.

We show that the above open shop problem has a schedule with finish time $\leq T + \varepsilon$ iff S has a partition.

a) If S has a partition u then there is a schedule with finish time $T + \varepsilon$. Figure 5.4.2 presents such a schedule.

b) If S has no partition then all schedules for OS must have a finish time $> T + \varepsilon$.

This is shown by contradiction. Assume that there is a schedule for OS with finish time $\leq T + \varepsilon$. Since

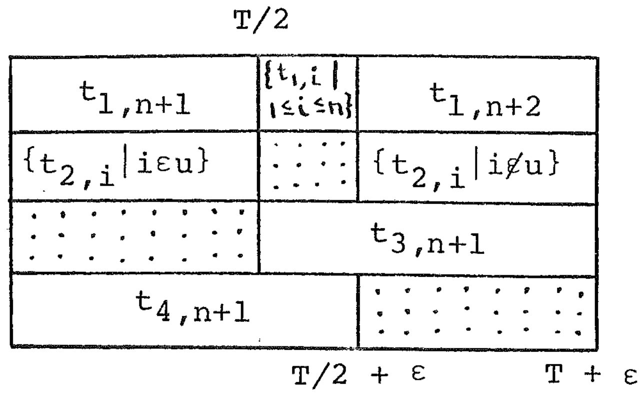


Figure 5.4.2

jobs n+1 and n+2 need a total time $T + \varepsilon$ they must be scheduled all the time and this will leave processor 1 free in the time interval $[T/2, T/2+\varepsilon]$. This is just enough time to process the n tasks $t_{1,i}$, $1 \leq i \leq n$. This means that all tasks $t_{2,i}$ that start their processing before time $T/2$ must terminate before time $T/2 + \varepsilon$ as otherwise for some job j $t_{1,j}$ and $t_{2,j}$ would be processed at the same time. Let u be the set of jobs that complete processing on processor 2 before time $T/2 + \varepsilon$. Then $\sum_{i \varepsilon s} t_{2,i} \leq T/2$ $\leq T/2 + \varepsilon$ as the a_i are integer. This implies that $\geq T/2$ is left for processing after time $T/2$. If the schedule is to finish at time $T + \varepsilon$ it must be the case that $\sum_{i \varepsilon u} t_{2,i} = T/2$. I.e. S has a partition. This contradicts the assumption. Hence when S has no partition there is no schedule with finish time $\leq \tau = T + \varepsilon$. ■

CHAPTER VI

FLOW SHOP AND JOB SHOP SCHEDULES

6.1 Introduction

In this Chapter we shall investigate optimal schedules for the following shop models:

a) Flow Shop

There are $n \geq 1$ jobs to be scheduled on $m \geq 1$ processors with the restriction that for every job i, the processing of task $j + 1$ cannot begin until the processing of task j is complete, $1 \leq j < m$. ■

b) Job Shop

There are $n \geq 1$ jobs to be scheduled on $m \geq 1$ processors with the restriction that the tasks for each job are ordered. The processing of a task cannot begin until the processing of all tasks preceding it have been completed. Several tasks may be specified for an individual processor. The notation $t_{j,i,k}$ will be used to indicate the kth task of job i. This task is to be processed on processor j. ■

For any schedule, S, we define the finish time, $f_i(S)$ of job i to be the earliest time at which all tasks of job i have been completed. The mean flow time, mft(S), is defined to be the quantity $\Sigma f_i(S)$. An optimal mean flow time (OMFT) schedule is a schedule which has the least mft amongst all possible schedules for the job set.

Several strategies for obtaining OFT (as defined in Chapter V) and OMFT schedules for flow shops and job shops have been advanced (see for example [5] and [7]). Branch and bound strategies for these problems are investigated in [18] and [24]. Despite all the research effort devoted to these problems there are no known efficient algorithms. In [2] and [11] it is shown that these problems are NP-Complete when one is restricted to nonpreemptive schedules. In section 6.2 we extend the NP-Completeness results of [2] and [11] for OFT nonpreemptive schedules. A more restricted version of the OFT nonpreemptive flow shop problem is also shown to be NP-Complete. In section 6.3 we obtain bounds comparing optimal and arbitrary active schedules for the flow shops and job shops. In this section we also present heuristics that result in schedules with a mft and finish time better than the worst active schedules. Finally, a comparison is made between the finish times of preemptive and nonpreemptive schedules.

Since NP-Complete problems are normally stated as language recognition problems, we restate the OFT problem as such:

<u>FOFT:</u> Given an m processor n job flow shop problem

with task times $t_{j,i}$, $1 \leq j \leq m$ and $1 \leq i \leq n$ does it have a schedule with finish time $\leq \tau$?

When it is necessary to distinguish between preemptive and nonpreemptive schedules we shall prefix FOFT with the type of schedule being considered .

JOFT: This is the same as FOFT except for a job shop.

6.2 Complexity of Preemptive and Nonpreemptive Scheduling

6.2.1 Flow Shop

OFT nonpreemptive schedules for the two processor (m = 2) flow shop can be obtained in O(n log n) time using Johnson's algorithm [7, p. 83]. For the case m = 2 one can easily show that an OFT preemptive schedule has the same finish time as an OFT nonpreemptive schedule. Hence, Johnson's algorithm also gives an OFT preemptive schedule. In this section we shall show that when m > 2 finding OFT preemptive and nonpreemptive flow shop schedules is NP-Complete. This is true even when the jobs are restricted to have at most two nonzero tasks each. This then gives us the simplest case of the flow shop problem that is both NP-Complete and for which no polynomial algorithm is known (note that when jobs have only 1 task per job, OFT schedules may be trivially obtained).

Theorem 6.2.1 FOFT with $m = 3$ and no job having more than two nonzero tasks is NP-Complete.

Proof The proof is presented as two lemmas. In Lemma 6.2.1 it is shown that Partition α Preemptive FOFT. Lemma 6.2.2 shows that if $P = NP$ then Preemptive FOFT is recognizable in polynomial time. The same proof also shows that nonpreemptive FOFT is also NP-Complete. ∎

Lemma 6.2.1 Partition α Preemptive FOFT with $m = 3$ and at most two nonzero tasks per job.

Proof From the partition problem $S = \{a_1, a_2, \ldots, a_n\}$ construct the following preemptive flow shop problem, FS, with $n+2$ jobs, $m = 3$ machines and at most 2 tasks per job:

$$t_{1,i} = a_i \; ; \; t_{2,i} = 0 \; ; \; t_{3,i} = a_i \; , \quad 1 \leq i \leq n$$

$$t_{1,n+1} = T/2 \; ; \; t_{2,n+1} = T \; ; \; t_{3,n+1} = 0$$

$$t_{1,n+2} = 0 \; ; \; t_{2,n+2} = T \; ; \; t_{3,n+2} = T/2$$

where $T = \sum_{1}^{n} a_i$ and $\tau = 2T$

We now show that the above flow shop problem has a preemptive schedule with finish time $\leq 2T$ iff S has a partition.

(a) If S has a partition u then there is a nonpreemptive schedule with finish time $2T$. One such schedule is shown in figure 6.2.1 .

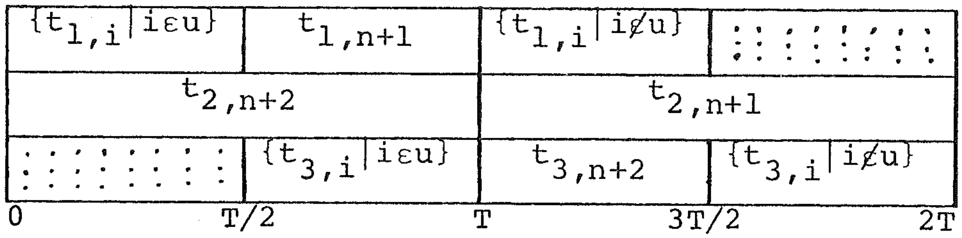


figure 6.2.1

(b) If S has no partition then all preemptive schedules for FS must have a finish time $> 2T$. This can be shown by contradiction. Assume that there is a preemptive schedule for FS with finish time $\leq 2T$. We make the following observations regarding this schedule: (i) task $t_{1,n+1}$ must finish by time T (as $t_{2,n+1} = T$ and cannot start until $t_{1,n+1}$ finishes)

(ii) task $t_{3,n+2}$ cannot start before T units of time have elapsed as $t_{2,n+2} = T$.

Observation (i) implies that only T/2 of the first T time units are free on machine one. Let V be the set of indices of tasks completed on machine 1 by time T (excluding task $t_{1,n+1}$). Then $\sum_{i \varepsilon V} t_{1,i} < T/2$ as S has no partition. Hence $\sum_{i \not\varepsilon V} t_{3,i} > T/2$. The processing of jobs not included in V cannot commence on machine 3 until after time $= T$ since their machine 1 processing is not completed until after T. This together with observation (ii) implies that the total amount of processing left for machine

3 at time T is $t_{3,n+2} + \sum_{i \notin V} t_{3,i} > T$. The schedule lenght must therefore be $> 2T$. ■

Corollary 6.2.1 Partition α Nonpreemptive FOFT with $m = 3$ and at most two nonzero tasks per job.

Proof The construction of Lemma 6.2.1 yields a flow shop problem that has a nonpreemptive schedule with finish time τ iff the corresponding partition problem has a partition. ■

Lemma 6.2.2 If $P = NP$ then preemptive FOFT is polynomial recognizable.

Proof One may easily construct a nondeterministic Turing machine that guesses a preemptive schedule and verifies that it is of lenght $\leq \tau$. In order for this Turing machine to be polynomial complexity, we must show that every flow shop problem has an optimal preemptive schedule with a polynomial number of preemptions. We show this by construction. Let R be an optimal preemptive schedule for a m processor n job flow shop problem. If on any processor, j, there is a job, k, such that between its preemption and next resumption on that processor, no task $t_{j,i}$, $i \neq k$ is completed then this preemption for job k can be eliminated without affecting the schedules for the other processors. I.e. if the schedule for processor i has a subsequence $(\ell_1=k, s_1, f_1)$ (ℓ_2, s_2, f_2) ... $(\ell_{r-1}, s_{r-1}, f_{r-1})$ $(\ell_r=k, s_r, f_r)$ $\ell_i \neq k$, $2 \leq i < r$ and none of the tasks on j for jobs $\ell_2, \ell_3, \ldots, \ell_{r-1}$ finishes in this interval, then the schedule may be modified to $(\ell_1=k, s_1, f_1+\Delta)$ $(\ell_2, s_2 + \Delta, f_2 + \Delta)$ ... $(\ell_{r-1}, s_{r-1}+ \Delta, f_r)$ where $\Delta = f_r - s_r > 0$. Repeating this preemption elimination process one will eventually obtain a preemptive schedule with the same finish time as R and having at most n^2 preemptions per processor. Hence, there is an optimal preemptive schedule with at most n^2m preemptions. ■

6.2.2 Job Shop

The preceding results trivially imply that a severely restricted form of the job shop problem for $m > 2$ is NP-Complete. For the job shop with $m = 2$, however, no polynomial time algorithm is known. In [7, p. 105] an $O(n \log n)$ algorithm to obtain OFT nonpreemptive schedules when $m = 2$ and the jobs are restricted to have at most two nonzero tasks is presented. For this case, OFT preemptive schedules may be similarly obtained. For the nonpreemptive case, it is known [2,11] that when $m = 2$ and the job mix consists of $n-1$ jobs with one nonzero task each and an additional job with three nonzero tasks then the problem is

NP-Complete. We extend this result to the case of preemptive schedules. We show that finding OFT preemptive schedules when $m = 2$ is NP-Complete even when the job mix contains only two jobs with three nonzero tasks. The proof is presented in the form of two lemmas.

<u>Lemma 6.2.3</u> Partition α Preemptive JOFT

<u>Proof</u> From the partition problem $S = \{a_1, a_2, \ldots, a_n\}$ construct the following job shop problem JS, with $n+2$ jobs, $m = 2$ processors and all jobs with 2 non zero tasks except for two jobs with 3 nonzero tasks.

Job 1 - n : $t_{2,i,1} = t_{1,i,2} = a_i$ for $1 \leq i \leq n$

Job n+1 : $t_{1,n+1,1} = t_{2,n+1,2} = T/2$

$t_{1,n+1,3} = 3T$

Job n+2 : $t_{2,n+2,1} = 3T$

$t_{1,n+2,2} = t_{2,n+2,3} = T/2$

where $T = \sum_{1}^{n} a_i$ and $\tau = 5T$

We now show that the above job shop problem JS has a preemptive schedule with finish time $\leq 5T$ iff S has a partition.

a) If S has a partition u then there is a schedule with finish time 5T. One such schedule is shown in figure 6.2.2 .

$t_{1,n+1,1}$	$\{t_{1,i,2} \mid i \in u\}$	$t_{1,n+3,3}$	$t_{1,n+2,2}$	$\{t_{1,i,2} \mid i \notin u\}$
$\{t_{2,i,1} \mid i \in u\}$	$t_{2,n+1,2}$	$t_{2,n+2,1}$	$\{t_{2,i,1} \mid i \notin u\}$	$t_{1,n+2,3}$
T/2	T	4T	9T/2	5T

Figure 6.2.2

b) If S has no partition then all schedules for JS must have a finish time > 5T.

This is shown by contradiction. Assume that there is a schedule for JS with finish time $\leq$ 5T .

Observe i) on processor 2, task $t_{2,n+1,2}$ must be completed before time 2T and $t_{2,n+2,1}$ before 4T. Therefore, before 4T only T/2 units are free for jobs 1 - n and

ii) on processor 1: $t_{1,n+1,3}$ cannot start until T and $t_{1,n+2,2}$ cannot start until 3T.

Hence, processor 1 is committed to processing $t_{1,n+1,3}$ and $t_{1,n+2,2}$ after time T. Since these two tasks together require 7T/2 units, at most T/2 units are free after time T for jobs 1-n. This in turn implies that there are at least T/2 units of free time on processor 1 before time T. Let $\overline{U}$ be the set of indices of jobs being processed in the time interval [0,T] on processor 1. Let $u = \{i \mid i \in \overline{U}$ and $i \leq n\}$. Since the schedule has a finish time of 5T, there can be no idle time on either of the processors.

Hence, $\sum_{i \in u} t_{1,i,2} \leq T/2$. But, the only jobs that can start on processor 1 are those for which $t_{2,i,1}$ has completed. Hence, $\sum_{i \in u} t_{2,i,1} = \sum_{i \in u} t_{1,i,2} \geq T/2$ must be completed by T on processor 2. By (i) this processor has at most $T/2$ units free before T. Consequently $\sum_{i \in u} t_{2,i,1} = T/2$. This contradicts the assumption that S has no partition. ■

Corollary 6.2.2 Partition α Nonpreemptive JOFT.

Proof Same as above. A simpler proof of this corollary may be found in [2,11]. ■

Lemma 6.2.4 If $P = NP$ then JOFT is polynomial recognizable.

Proof Similar to that for Lemma 6.2.2. Note that the bound on the number of preemptions will now be $n^2\ell$ where ℓ is the maximum number of nonzero tasks for any job. ■

6.3 Approximate Solutions

Since the problems of finding OFT and OMFT schedules for flow shops and job shops is NP-Complete (see [11] for NP-Completeness of OMFT) we turn our attention to obtaining schedules whose performance approximates that of optimal schedules. To begin with, we derive a bound for the ratio of worst and best schedules for the two performance measures being considered. We then present approximation algorithms that generate schedules with a worst case bound smaller than this. In examining "worst" schedules, we restrict ourselves to active schedules. An <u>active schedule</u> is a schedule in which at all times from start to finish some processor is busy (i.e. it is processing a task). For a given set of jobs and a schedule S we denote by $f(S)$ the finish time of S and by $mft(S)$ the mean flow time of S.

<u>Lemma 6.3.1</u> Let S^* be an OMFT schedule for an m processor n job flow (or job) shop problem. Let S be any active schedule for this problem. Then, $mft(S)/mft(S^*) \leq n$.

<u>Proof</u> For each job i define L_i to be the sum of its task times. Let $T = \sum_1^n L_i$. Without loss of generality we may assume that the jobs have been indexed so that $L_1 \geq L_2 \geq \ldots \geq L_n$. Let the order in which jobs finish in S be $i_1, i_2, \ldots, i_n$ and let f_{i_j} be the finish time of job i_j in S. Then,

$$f_{i_j} \leq \sum_{\ell=1}^{j} L_{i_\ell} \leq \sum_{i=1}^{j} L_i$$

$$\therefore mft(S) = \sum_{j=1}^{n} f_{i_j} \leq \sum_j \sum_{i=1}^{j} L_i$$

$$= \sum_{i=1}^{n} (n+1-i)L_i$$

$$\leq nT$$

For S^* we know that $f_i \geq L_i$ and so $mft(S^*) \geq \Sigma L_i = T$. Hence, $mft(S)/mft(S^*) \leq n$. ∎

Since rather crude approximations were used to obtain this bound, it is surprising that Johnson's OFT algorithm on 2 processors schieves this bound. I.e. there are OFT schedules S such that $mft(S)/mft(S^*)=n$. The next example gives an instance of a job mix for which this happens.

Example 6.3.1 Consider the 2 processor, n job flow shop problem with task times $t_{1,i} = \varepsilon$, $1 \leq i \leq n$, $t_{2,1} = k$ and $t_{2,i} = \delta$, $2 \leq i \leq n$. $\delta < \varepsilon << k/n^2$. One may easily verify that Johnson's algorithm generate the permutation schedule $S = (1,2,\ldots,n)$. The mean flow time of this schedule is

$$mft(S) = (k + \varepsilon) + (k + \varepsilon + \delta) + \ldots + (k + \varepsilon + (n-1)\delta)$$

$$= n(k + \varepsilon) + n(n-1)\delta/2$$

An OMFT schedule, S^*, for this job set is obtained by using the permutation $2,3,\ldots,n,1$. The mean flow time of this schedule (figure 6.3.1) is:

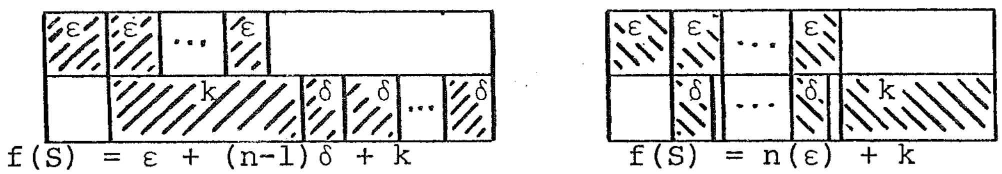
$$mft(S^*) = (\varepsilon + \delta) + (2\varepsilon + \delta) + \ldots + ((n-1)\varepsilon + \delta) + (n\varepsilon + k)$$

$$= n(n+1)\varepsilon/2 + n\delta + k$$

Hence $mft(S)/mft(S^*) = \dfrac{n(k + \varepsilon) + n(n-1)\delta/2}{n(n + 1)\varepsilon/2 + n\delta + k}$

As ε approached zero this bound becomes

$$\frac{mft(S)}{mft(S^*)} \simeq \frac{nk}{k} = n \quad ■$$



(a) OFT schedule from Johnson's algorithm (b) OMFT schedule

Figure 6.3.1

Note that this example also shows that the bound of n holds for OFT schedules even for the job shop and also when preemptive schedules are allowed.

A simple heuristic that results in schedules with a mft which in the worst case is closer to the optimal than the bound of Lemma 6.3.1 is obtained by processing jobs in order of nondecreasing L_i (L_i = sum of task times for job i). This heuristic will be referred to as SPT (see [7], p. 76).

<u>Lemma 6.3.2</u> Let S^* be an OMFT schedule for an m processor n job flow (or job) shop problem. Let S be a SPT schedule for this problem. Then, $mft(S)/mft(S^*) \leq m$.

<u>Proof</u> Let L_i be the sum of the task times for job i.

Without loss of generality we assume that $L_1 \leq L_2 \leq \ldots \leq L_n$. Let f_i be the finish time of job i using the SPT schedule S. Then, $f_i \leq \sum_1^i L_j$ and so

$$mft(S) = \sum_1^n f_i \leq \sum_{i=1}^n \sum_{j=1}^i L_j$$

Let $(i_1, i_2, \ldots, i_n)$ be the order in which jobs finish in the OMFT schedule S^*. For S^* we have,

$$f_{i_k} \geq \sum_{j=1}^k L_{i_j}/m \geq \sum_{j=1}^k L_j/m$$

and so $mft(S^*) \geq \sum_{k=1}^n \sum_{j=1}^k L_j/m$

Consequently, $mft(S)/mft(S^*) \leq m$. ■

Example 6.3.2 Consider the 2 processor, $2n$ job shop problem with task times $t_{1,i} = k$, $t_{2,i} = \varepsilon_1$ for $1 \leq i \leq n$ and $t_{1,i} = \varepsilon_2$, $t_{2,i=k}$ for $n+1 \leq i \leq 2n$, where $\varepsilon_1 < \varepsilon_2 << k/n^2$. One may easily verify that SPT algorithm generates the permutation schedule $s = (1, 2, \ldots, 2n)$. The mean flow time of this schedule is:

$$mft(S) = (k + \varepsilon_1) + (2k + \varepsilon_1) + \ldots + (nk + \varepsilon_1) + ((n + 1)k + \varepsilon_2) + \ldots + (2nk + \varepsilon_2)$$

$$= \sum_1^{2n} ik + n\varepsilon_1 + n\varepsilon_2$$

An OMFT schedule, S^*, for this job set is ob - tained by using the permutation $(n + 1, 1, n+2, 2, \ldots, n + i, i, \ldots, 2n, n)$. The mean flow time of this schedule (fig. 6.3.2) is

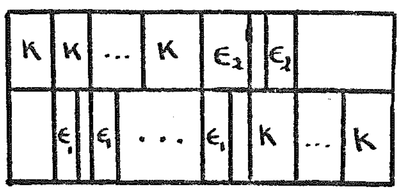
$$mft(S) = (k + \varepsilon_2) + (k + \varepsilon_2 + \varepsilon_1) + \dots + (nk + n\varepsilon_2) + (nk + n\varepsilon_2 + \varepsilon_1)$$

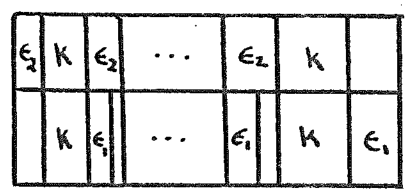
$$= 2 \sum^{n} ik + 2 \sum^{n} i\varepsilon_2 + n\varepsilon_1$$

$$\text{Hence} \quad mft(S)/mft(S^*) = \frac{\sum^{2n} ik + n\varepsilon_1 + n\varepsilon_2}{2 \sum^{n} ik + 2 \sum^{n} i\varepsilon_2 + n\varepsilon_1}$$

As ε_1 and ε_2 approach zero this bound becomes

$$\frac{mft(S)}{mft(S^*)} \simeq \frac{(2n)(2n+1)}{2n(n+1)} \simeq 2$$





(a) SPT Schedule (b) OMFT Schedule

Figure 6.3.2

This may be extended to any arbitrary m by using the following job set:

$$t_{1,i} = k\ ; \quad t_{2,i} = \varepsilon_1 \dots t_{m,i} = \varepsilon_1 \text{ for } 1 \le i \le n$$

$$t_{1,i} = \varepsilon_2 ; \quad t_{2,i} = k \quad \dots t_{m,i} = \varepsilon_2 \text{ for } n+1 \le i \le 2n$$

$$\vdots \qquad \vdots \qquad \vdots \qquad \vdots$$

$$t_{1,i} = \varepsilon_m ; \quad t_{2,i} = \varepsilon_m \dots t_{m,i} = k \quad (m-1)n+1 \le i \le mn$$

Where $\varepsilon_i < \varepsilon_{i+1}$ for $1 \le i < m$ and $\varepsilon_m << k/n^2$. This example show that the bound of Lemma 6.3.2 is best possible. ∎

Let us now turn our attention to the finish time properties of active schedule.

Lemma 6.3.3 Let S^* be an OFT schedule for a $m > 2$ processor n job flow (or job) shop problem. Let S be any active schedule for this problem. Then, $f(S)/f(S^*) \leq m$.

Proof Let T be the sum of task times for all n jobs. Then, clearly, for any active schedule S, $f(S) \leq T$. Also, for any schedule, S^* , we trivially have $f(S^*) \geq T/m$. So, $f(S)/f(S^*) \leq m$. ■

Once again, as in the case of Lemma 6.3.1 , the proof technique would seem to indicate that any "reasonable" heuristic would result in schedules with a worst case bound less than m. This unfortunately is not the case. We define by LPT the heuristic: schedule jobs in order of nondecreasing L_i. For LPT schedules the bound of Lemma 6.3.3 is tight. Note that this heuristic is similar to the one used by Graham [13] to schedule identical processors and in Chapter IV for scheduling on uniform processors. In both these earlier applications of the heuristic, LPT schedules had a worst case finish time at most a "small" constant times the optimal finish time. This is no longer the case for flow shop and job shop schedules.

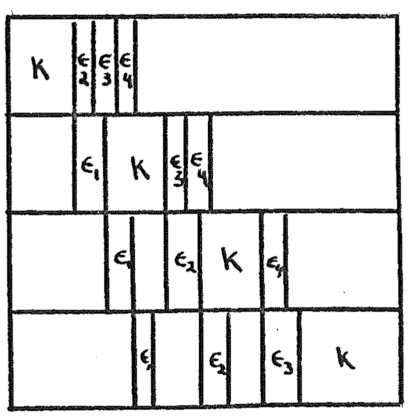
Example 6.3.3 Consider the m processor m job flow shop with task times $t_{i,i} = k$, $1 \leq i \leq m$ and $t_{j,i} = \varepsilon_i$, $1 \leq i \leq m$, $1 \leq j \leq m$ and $i \neq j$. $\varepsilon_i > \varepsilon_{i+1}$, $1 \leq i < m$, and $\varepsilon_1 << k$. $L_i = k + (m-1)\varepsilon_i$ and so $L_i > L_{i+1}$, $1 \leq i < m$. The LPT schedule, S, is the permutation schedule obtained by processing jobs in the order $(1,2,\ldots,n)$ on all processors. The finish time of S (figure 3.3(a)) is :

$$f(S) = mk + \sum_{i=1}^{m-1} \varepsilon_i$$

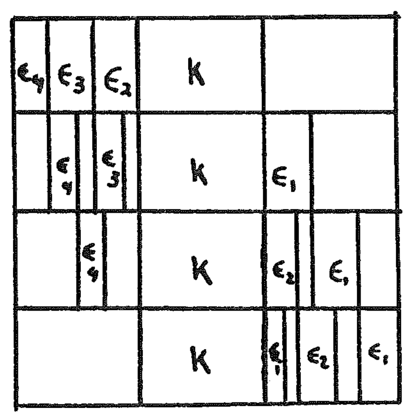
An optimal schedule, S* , is shown in figure 3.3(b). The finish time of S* is:

$$f(S^*) = \sum_{i=2}^{m} \varepsilon_i + k + (m-1)\varepsilon_1$$

Hence, $f(S)/f(S^*) = m$ for $\varepsilon_i << k$. ∎



(a) LPT Schedule



(b) OFT Schedule

Figure 6.3.3 Schedules for example 6.3.3 when m = 4.

The worst case bound of m for active schedules in a flow shop can be reduced to $\lceil m/2 \rceil$ by using the following heuristic: Divide the m processors into $\lceil m/2 \rceil$ groups, each group containing at most two processors. The processors in group i are the $2i$th and $2i+1$st ones. Johnson's $O(n \log n)$ algorithm is used to obtain optimal finish time schedules for each of these $\lceil m/2 \rceil$ groups of machines. These $\lceil m/2 \rceil$ optimal schedules are then concatenated to obtain a schedule for the original m processor flow shop. More formally, the algorithm H is:

```
Algorithm H
// Obtain a schedule for the n job m processor
   flow shop. The task times are t_{j,i}, 1 ≤ j ≤ m
   and 1 ≤ i ≤ n //
   for j=1 to ⌊m/2⌋ do
           R(j) ← permutation corresponding to
                  optimal schedule for the two
                  processors 2j-1,2j with task
                  times t_{k,i}, 2j-1 ≤ k ≤ 2j and
                  1 ≤ i ≤ n
   end
   if m is odd then R(⌈m/2⌉) ← permutation 1,2,...n
// the schedule for the m processor flow shop prob
   lem is: process jobs on processor i using the
```

permutation $R(\lfloor (i+1/2 \rfloor)$.//

<u>end</u> of algorithm H

Since an optimal two processor flow shop schedule can be obtained in time $O(n \log n)$, the total time needed by algorithm H is $O(mn \log n)$.

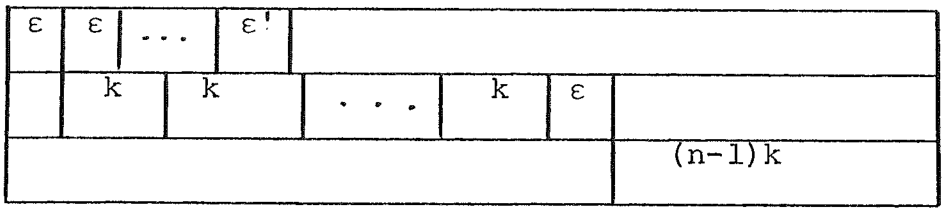
<u>Lemma 6.3.4</u> Let S be a schedule generated by algo - rithm H and let S^* be an OFT schedule. Then, $f(S)/f(S^*) \leq \lceil m/2 \rceil$.

<u>Proof</u> Let the lenght of each of the schedules $R(i)$, $1 \leq i \leq \lceil m/2 \rceil$ obtained in algorithm H be denoted by $f(R(i))$. Since each such schedule is optimal for its processor pair, it follows that $f(S^*) \geq \max\{f(R(i))\}$. In the worst case, the schedule, S , generated by algorithm H will have a finish time $f(S) \leq \Sigma f(R(i)) \leq \lceil m/2 \rceil \max\{f(R(i))\}$. Consequently, $f(S)/f(S^*) \leq \lceil m/2 \rceil$. ∎

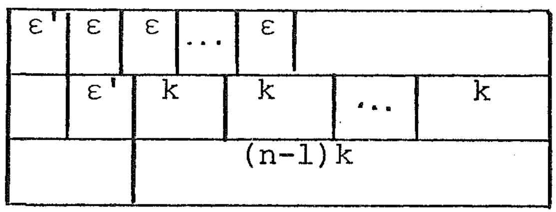
<u>Example 6.3.4</u> Using algorithm H on the n job flow shop problem with $m = 3$ and task times:

$$t_{1,i} = \varepsilon,\ t_{2,i} = k,\ t_{3,i} = 0, \quad 1 \leq i < n$$

and $t_{1,n} = \varepsilon',\ t_{2,n} = \varepsilon',\ t_{3,n} = (n-1)k; \varepsilon < \varepsilon' < k$ yields $R(1) = R(2) = 1,2,\ldots,n$. This gives the schedule, s, of figure 6.3.4(a) which has a finish time of $\varepsilon + \varepsilon' + 2k(n-1)$. Figure 6.3.4(b) shows an optimal schedule S^*. $f(S^*) = 2\varepsilon' + k(n-1)$ and $f(S)/f(S^*) \simeq 2 = \lceil 3/2 \rceil$. ∎



(a) Schedule S for algorithm H.



(b) Optimal schedüle S*

Figure 6.3.4

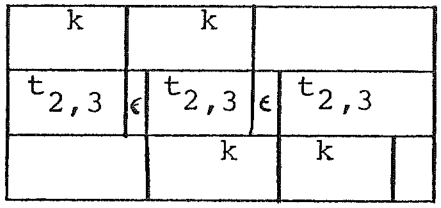
We close this section with a comparison of OFT preemptive and nonpreemptive schedules. If S_p^* snd S_n^* are OFT preemptive and nonpreemptive schedules respec - tively, then from the proof of Lemma 6.3.4 it follows that $f(S_n^*)/f(S_p^*) \leq \lceil m/2 \rceil$. The next example shows that this is a tight bound when $m = 3$ and $m = 4$.

<u>Example 6.3.5</u> Consider the 3 processor flow shop with 3 jobs and task times:

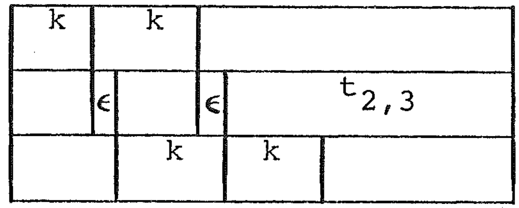
$$t_{1,i} = k \ , \quad t_{2,i} = \varepsilon \ , \quad t_{3,i} = k \qquad i = 1,2$$

and $t_{1,3} = 0$, $t_{2,3} = 3k$, $t_{3,3} = 0$

The OFT preemptive schedule S_p^* has $f(S_p^*) = 3k + 2\varepsilon$ while the OFT nonpreemptive schedule S_n^* has $f(S_n^*) = 5k + \varepsilon$, $f(S_n^*)/f(S_p^*) \simeq 5/3$ for $\varepsilon << k$.



(a) S_p^*



(b) S_n^*

Figure 6.3.5

Generalizing this to n-1 jobs with

$$t_{1,i} = k \ , \quad t_{2,i} = \varepsilon \ , \quad t_{3,i} = k \ , \qquad 1 \leq i \leq n-1$$

and 1 job with $t_{1,n} = t_{3,n} = 0$ and $t_{2,n} = nk$ we get $f(S_p^*)/f(S_n^*) \simeq 2 - 1/n$ which approximates $\lceil m/2 \rceil$ for large n. ∎

For the case of a job shop, we conclude from the proof of Lemma 6.3.3 that $f(S_n^*)/f(S_p^*) \leq m$. The next example shows that this is a tight bound for $m = 2$.

<u>Example 6.3.6</u> The two processor job shop problem with two jobs and task times:

$$t_{1,1,1} = k \ , \quad t_{2,1,2} = \varepsilon \ , \quad t_{1,1,3} = k$$

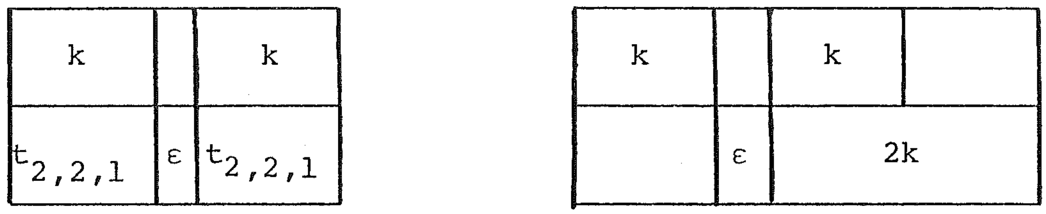
and $t_{2,2.1} = 2k$ $\qquad \varepsilon << k$

has S_p^* and S_n^* as in figure 6.3.6. $f(S_n^*)/f(S_p^*) = (3k + \varepsilon)/(2k + \varepsilon) = 3/2$ for $\varepsilon << k$. Generalizing to n-1 jobs with

$$t_{1,i,1} = k \ , \ t_{2,i,2} = \varepsilon \ , \ t_{1,i,3} = k \ , \quad 1 \leq i < n$$

and 1 job with $t_{2,n,1} = nk$ we get

$f(S_n^*)/f(S_p^*) = 2 - 1/n$. ∎



(a) S_p^* (b) S_n^*

Figure 6.3.6

CHAPTER VII

P-COMPLETE APPROXIMATE PROBLEMS

7.1 Introduction

Many P-Complete problems, especially the optimization problems,are of practical significance. Often, as in the case of the Knapsack problem [36], approximate solutions (i.e. feasible solutions that are guaranteed to be 'reasonably' close to the optimal) would be acceptable so long as they can be obtained 'quickly' (e.g. by an $O(n^k)$ algorithm for small k). Johnson [19] and Sahni [36] have studied some P-Complete problems with respect to obtaining 'good' (i.e. polynomial) approximate algorithms.

Additional results on approximation algorithms for P-Complete problems can be found in [3, 10, 13, 15, 17, 33, 34]. Some of these algorithms obtain ε-approximate solutions only for cerain values of ε (i.e., $\varepsilon \geq k$ for some k). For example, Graham's heuristic to sequence jobs on m identical processors is ε-approximate for $\varepsilon \geq (1/3)(1 - 1/m)$. Other approximation algorithms [15, 17, 34, 36] obtain ε-approximations for any $\varepsilon > 0$. An example is the $O(n/\varepsilon^2\ n \log n)$ algorithm of Ibarra and Kim [17] for the 0/1 Knapsack problem. General techniques for obtaining ε-approximate solutions for all

ε are presented by Sahni [34] and Horowitz and Sahni [15]. The techniques are applicable to certain types of P-Complete problems.

In this Chapter we shall look at some "natural" P-Complete problems and show that the corresponding approximation problem are also P-Complete. While one can easily construct "non-natural" problems with this property (for example: max : f(G) where $f(G) = 2/(1-\varepsilon)$ if graph G has a clique of size $\geq k$ and $f(G) = 1$ otherwise. If G has a clique of size $> k$ then all ε-approximate solutions have $f(G) > 1$. If G has no such clique then $f(G) = 1$. Hence the approximation problem is P-Complete for all ε ; it is interesting that this should be true for naturally occurring problems. Garey and Johnson [10] show that obtaining ε-approximate solutions to the chromatic number problem is P-Complete for all $\varepsilon < 1$. The problems we shall consider are: travelling salesperson; cycle covers; 0/1 integer programming; multicommodity network flows; quadratic assignment; general partition; k-MinCluster and generalized assignment. For these problems it will be shown that the ε-approximation problem is P-Complete for all values of ε . (One should note that in the case of a maximization problem ε is restricted to $0 < \varepsilon < 1$ as all feasible solutions are 1-approximate.)

While the ε-approximation problem for the general travelling salesperson problem is P-Complete for all ε Rosenkrantz and Stearns [33] have shown that when the edge weights satisfy the triangular inequality then ε-approximate solutions may be obtained in polynomial time for certain values of ε.

7.2 P-Complete Approximate Problems

In this section we look at some P-Complete problems and show that they have a polynomial time approximate algorithm iff $P = NP$. This would then imply that if $P \neq NP$, then any polynomial time approximation algorithm for these problems, heuristic or otherwise, must produce arbitrarily bad approximations on some inputs. The problems we look at are:

(i) Travelling Salesperson: Given an undirected (directed) complete graph $G(N,A)$ and a weighting function $w : A \to Z$ find an optimal tour (i.e. an optimal hamiltonian cycle). The optimality criteria we shall consider are:

(a) Minimize tour lenght (i.e. find the shortest hamiltonian cycle)

(b) Minimize mean arrival time at vertices. The arrival time is measured relative to a start vertex i_1 and the weights are interpreted as the time needed to go from vertex i to vertex j. If $i_1, i_2, \ldots, i_n$, $i_{n+1} = i_1$ is a tour (i.e. hamiltonian cycle) then the mean arrival time is measured by

$$(1/n) \sum_{k=2}^{n+1} \sum_{j=1}^{k} w(i_j, i_k) = (1/n) \sum_{j=1}^{n} (n+1-j) w(i_j, i_{j+1}) .$$

The objective is to find a tour $i_1, i_2, \ldots, i_n, i_1$ that minimizes $\sum_{j=1}^{n} (n+1-j)\, w(i_j, i_{j+1})$ (see [7], p.56).

(c) Minimize variance of arrival times. If $i_1, i_2, \ldots$ $\ldots, i_n, i_{n+1} = i_1$ is a tour starting at i_1 then the arrival time, Y_k at vertex i_k is

$$Y_k = \sum_{j=1}^{k-1} w(i_j, i_{j+1}) , \quad 1 < k \leq n+1$$

The mean arrival time $\overline{Y}$ (as defined in (b) above) is

$$\overline{Y} = \frac{1}{n} \sum_{k=2}^{n+1} Y_k = \frac{1}{n} \sum_{j=1}^{n} (n+1-j)\, w(i_j, i_{j+1})$$

We wish to obtain a tour that minimizes the quantity

$$\sigma = (1/n) \sum_{j=2}^{n+1} (Y_j - \overline{Y})^2$$

(ii) Undirected Edge Disjoint Cycle Cover: Given an undirected graph $G(N,A)$ find the minimum number of edge disjoint cycles needed to cover all the vertices of N (i.e. minimum number of cycles such that each vertex of G is on at least one cycle).

(iii) Directed Edge Disjoint Cycle Cover: Same as (ii) except that G is now a directed graph.

(iv) Undirected Vertex Disjoint Cycle Cover: This problem is the same as (ii) except that now, the cycles are constrained to be vertex disjoint.

(v) Directed Vertex Disjoint Cycle Covers: Same as (iv) except that G is now a directed graph.

(vi) 0-1 Integer Programming with one constraint.

(vii) Multicommodity Network Flows: We are given a transportation network [6] with source s_1 and sink s_2. The arcs of the network are labeled corresponding to the commodities that can be transported along them. Such a network is said to have a flow f iff f units of each commodity can be transported from source to sink. The problem here is to maximize f.

(viii) Quadratic Assignment ([12], p. 18):

$$\text{minimize} \quad f(x) = \sum_{\substack{i,j=1 \\ i \neq j}}^{n} \sum_{\substack{k,\ell=1 \\ k \neq \ell}}^{m} c_{i,j} d_{k,\ell} x_{i,k} x_{j,\ell}$$

$$\text{subject to (a)} \quad \sum_{k=1}^{m} x_{i,k} \leq 1, \quad 1 \leq i \leq n$$

$$\text{(b)} \quad \sum_{i=1}^{n} x_{i,k} = 1, \quad 1 \leq k \leq m$$

$$\text{and (c)} \quad x_{i,k} = 0,1 \quad \text{for all } i, k$$

$$\text{where } c_{i,j} \text{ and } d_{k,\ell} \geq 0, \quad 1 \leq i,j \leq n, \quad 1 \leq k,\ell \leq m$$

One situation in which a problem arises is that of optimally locating n plants, $1 \leq k \leq n$, at m, $1 \leq i \leq m$, possible sites. Thus, $x_{i,k} = 1$ iff plant k is to be located at site i. Condition (a) states that at most 1 plant can be located at any particular site. Condition (b) requires that every plant be assigned to exactly 1 site. If $c_{i,j}$ represents the cost of transporting 1 unit of goods from site i to site j and $d_{k,\ell}$ is the amount of goods to be transported from plant k to plant ℓ then $f(x)$ represents the cost of transporting all the goods between plants.

(ix) k-General Partition [31] : We are given a connected undirected graph $G(N,A)$, an edge weighting function $f: A \rightarrow Z$, a vertex weighting function $w: N \rightarrow Z$ a positive number W and an integer $k \geq 2$. The problem is to obtain k disjoint sets $S_1, S_2, \ldots, S_k$ such that:

(a) $\bigcup_{i=1}^{k} S_i = N$

(b) $S_i \cap S_j = \emptyset$ for $i \neq j$

(c) $\sum_{j \in S_i} w(j) \leq W$ for $1 \leq i \leq k$

and (d) $\sum_{i=1}^{k} \sum_{\substack{\{u,v\} \in A \\ u,v \in S_i}} f(u,v)$ is maximized

Partitioning problems of this type are encountered in the assignment of logic blocks to circuits cards in computer hardware design and in the assignment of computer information to physical blocks of storage[31].

(x) k-MinCluster: This is the document clustering problem with the minimization criteria (see Chapter II). Given an undirected graph $G(N,A)$, an integer $k \geq 3$, a weighting function $w: A \rightarrow Z$ find k disjoint sets $S_1, S_2, \ldots, S_k$ such that

$$\bigcup_{i=1}^{k} S_i = N \; ; \quad S_i \cap S_j = \emptyset \quad \text{for} \quad i \neq j$$

and $\displaystyle\sum_{i=1}^{k} \sum_{\substack{\{u,v\} \varepsilon A \\ u,v \varepsilon S_i}} w(u,v)$ is minimized

(xi) Generalized Assignment [38]:

$$\text{minimize} \quad \sum_{i \varepsilon I} \sum_{j \varepsilon J} c_{i,j} \, x_{i,j}$$

$$\text{subject to} \quad \sum_{j \varepsilon J} r_{i,j} \, x_{i,j} \leq b_i \quad \text{for all } i \varepsilon I$$

$$\sum_{i \varepsilon I} x_{i,j} = 1 \quad \text{for all } j \; \varepsilon \; J$$

$$x_{i,j} = 0,1$$

In this formulation $I = (1,2,\ldots,m)$ is a set of agent indices, $J = (1,2,\ldots,n)$ is a set of task indices, $c_{i,j}$ is the cost when agent i is assigned task j, $r_{i,j}$ is the resource required by agent i to perform task j and $b_i > 0$ is the amount of re - source available to agent i. The decision variable is 1 if agent i is assigned to task j , 0 otherwise. This problem arises in the following situations: assigning software development tasks to programmers, assigning jobs to computer networks, scheduling varia - ble lenght television commercials etc.

In order to prove some of our results we shall use the following known P-Complete problems:

a) Hamiltonian Cycle: Given an undirected (direc ted) graph $G(N,A)$ does it have a cycle containing each vertex exactly once, [21].

b) Multicommodity Flows: Given the transportation network of (vii) above does it have a flow of $f = 1$ [35].

c) k-Graph Colorability: Given an undirected graph $G(N,A)$ and a positive integer k do there exist disjoint subsets $S_1, S_2, \ldots, S_k$ such that $\bigcup_{i=1}^{k} S_i = N$ and if $\{i,j\} \varepsilon A$ then vertices i and j are in different sets [21].

<u>Theorem 7.3.1</u> The ε-approximation problem for (i)-(xi) above is P-Complete.

<u>Proof</u> For each of the problems (i) - (xi), it is easy to see that if $P = NP$ then the ε-approximation problem is polynomially solvable (as the exact solutions would then be obtainable in polynomial time). Consequently, we concern ourselves only with showing that if there is a polynomial time approximation algorithm for any of the problems listed above then $P = NP$. Our approach is to separate feasible solutions to a given problem in such a way that from a knowledge of the approximate solution one can solve exactly a known P-Complete problem.

(i) (a) Hamiltonian Cycle α ε-approximate Travelling Salesperson (Minimum length criteria).

Let $G(N,A)$ be any graph. Construct the graph

$G_1(V,E)$ such that $V = N$ and $E = \{(u,v) | u,v \varepsilon V\}$. Define the weighting function $w : E \rightarrow Z$ to be

$$w\{u,v\} = \begin{cases} 1 & \text{if } (u,v) \varepsilon A \\ k & \text{otherwise} \end{cases} .$$

Let $n = |N|$. For $k > 1$, the Travelling Salesperson problem on G_1 has a solution of length n iff G has a Hamiltonian cycle. Otherwise, all solutions to G_1 have length $\geq k + n - 1$. If we choose $k \geq (1 + \varepsilon)n$ then, the only solutions approximating a solution with value n (if there was a Hamiltonian cycle in G_1) also have length n. Consequently, if the ε-approximate solution has length $\leq (1 + \varepsilon)n$ then it must be of length n. If it has length $> (1 + \varepsilon)n$ then G has no Hamiltonian cycle.

(i) (b) Hamiltonian Cycle α ε-approximate Travelling Salesperson (minimum mean arrival time criteria)

We construct $G_1(V,E)$ as in (i)(a) above. Let the starting vertex $i_1 = 1$. It is easy to see that G_1 has a tour with mean arrival time $\leq (n + 1)/2$ iff G has a Hamiltonian cycle. If G has no Hamiltonian cycle then all tours in G_1 have mean arrival time $\geq k/n + (n - 1)/2$. Choosing $k > (1 + \varepsilon)n(n + 1)/2$ separates sufficiently these two solutions. The only solutions approximating $n(n + 1)/2$ also have value $n(n + 1)/2$. Consequently, if the ε-approximate solution has value $\leq (1 + \varepsilon)(n + 1)/2$ then it must be of value $(n + 1)/2$ and G has a Hamiltonian cycle. If the value is $> (1 + \varepsilon)(n + 1)/2$, G has no Hamiltonian cycle.

(i)(c) Hamiltonian Cycle α ε-approximate Travel - ling Salesperson (minimum variance criteria)

From the undirected graph $G(N,A)$ we obtain the undirected graph $G_1(N_1,A_1)$ with:

$$N_1 = N \cup \{\alpha,\beta,\gamma,\delta\}$$
$$A_1 = A \cup \{(r,\alpha),\ (\alpha,\beta),\ (\beta,\gamma),\ (\gamma,\delta)\}$$
$$\cup \{(\delta,z) \mid (r,z)\ \varepsilon\ A\}$$

where r is some arbitrary vertex in N. This cons - truction is shown in figure 7.3.1.

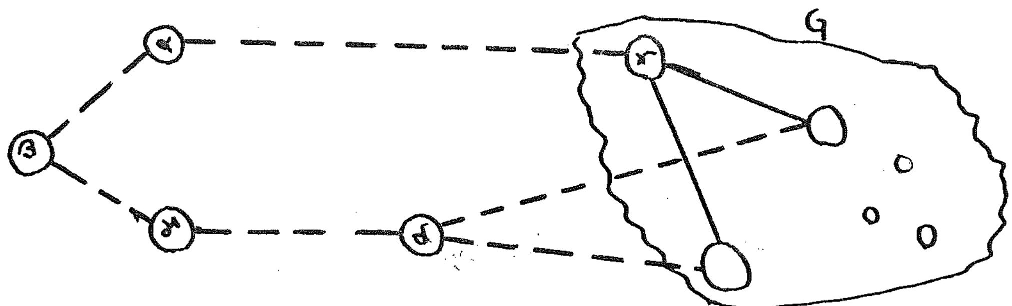


Figure 7.3.1 Construction of G_1 from G. Broken lines in G_1 are not in G.

From the construction, it is evident that G_1 has a Hamiltonian cycle iff G has such a cycle. From the graph G_1 we obtain the travelling salesperson problem $G_2(N_2,A_2)$ with $N_2 = N_1$, $A_2 = \{(i,j) \mid i \neq j,\ i,j \varepsilon N_2\}$ and weighting function $w : A_2 \rightarrow Z$ defined by

$$w(u,v) = \begin{cases} 1 & \text{if } (u,v) \varepsilon A_1 \\ k & \text{if } (u,v) \not\varepsilon A_1 \end{cases}$$

Lemma 7.3.1 obtains lower bounds on the variance (σ) of an optimal tour for G_2 .

Lemma 7.3.1 For $k > \sqrt{(1 + \varepsilon)(n)(n - 1)(n + 1)/3}$ and $\varepsilon > 0$ the complete graph G_2 has a tour, with starting vertex β , with a variance $\sigma \leq (n-1)(n+1)/12$ iff G_1 has a Hamiltonian cycle. If G_1 has no Hamiltonian cycle then the optimal tour for G_2 has $\sigma > (1+\varepsilon)(n-1)(n+1)/12$, $n = |N_2|$.

Proof If G_1 has a Hamiltonian cycle then this cycle is a valid tour in G_2 . All edges on this tour have weight 1 and:

$$\text{mean arrival time} = \overline{Y} = (n+1)/2$$

$$\begin{aligned} \sigma &= (1/n) \sum_{1}^{n} (i - \overline{Y})^2 \\ &= (1/n) \sum_{1}^{n} (i)^2 - \overline{Y}^2 \\ &= \frac{2n^2 + 3n + 1}{6} - \frac{(n + 1)^2}{4} \\ &= (n - 1)(n + 1)/12 \end{aligned}$$

If G_1 has no Hamiltonian cycle then every tour in G_2 must include at least one edge with weight $=k$. Let the optimal tour be $\beta = i_1, i_2, \ldots, i_n, i_{n+1} = \beta$. We have three cases:

case 1 $w(\beta, i_2) = 1$, $w(i_j, i_{j+1}) = k$ for some j, $1 < j \leq n$.

For this case we have $Y_2 = 1$ and $Y_{n+1} \geq k + n-1$.

If $\bar{Y} \geq k/2 + 1$ then $|Y_1 - \bar{Y}| \geq k/2$. If $\bar{Y} < k/2 + 1$ then $|Y_{n+1} - \bar{Y}| \geq k/2$.

In either case we have:

$$\sigma \geq (k/2)^2/n = k^2/(4n) > \frac{(1+\varepsilon)(n-1)(n+1)}{12}$$

for $k > \sqrt{(1+\varepsilon)n(n-1)(n+1)/3}$.

Case 2 $w(\beta,i_2) = k$ and all other edges have weight=1

Since all other edges on the tour have weight 1 it follows that $i_n = \alpha$ or $i_n = \gamma$ as (α,β) and (γ,β) are the only two edges in A_2 incident to β and having weight 1. Without loss of generality we may assume $i_n = \alpha$. Since γ is a vertex on the tour, the tour enters γ via some edge (u,γ) and leaves via another edge (γ,v) $u \neq v$ and $v \neq \beta$. Also, $u \neq \beta$ as $w(\beta,i_2) = k \neq 1$. From the construction of G_2 it is clear that the only edges in A_2 incident to γ with weight 1 are (β,γ) and (γ,δ). (γ,δ) is the only edge satisfying the requirements on u and v. Hence, the second edge on the tour incident to γ must have weight = k. Hence, there is no tour in G_2 with $w(\beta,i_2) = k$ and all other edges having a weight of 1.

Case 3 $w(\beta,i_2) = k$ and $w(i_j,i_{j+1}) = k$ for some j, $1 < j \leq n$.

Now, $Y_2 = k$ and $Y_{n+1} \geq 2k + n - 2$

If $\bar{Y} \geq 3k/2$ then $|Y_1 - \bar{Y}| \geq k/2$

If $\bar{Y} < 3k/2$ then $|Y_n - \bar{Y}| \geq k/2$

Hence, $\sigma > (1+\varepsilon)(n-1)(n+1)/12$ (see case 1)

This takes care of all possibilities when G_1 has no Hamiltonian cycle. ■

The reduction of (i)(c) now follows from arguments similar to those of (i)(a) and (b).

(ii)-(v) The proofs for (ii)-(v) are similar. We outline the proof for (iv).

(iv) Undirected Hamiltonian Cycle α ε-approximate Disjoint Cycle Cover

Given an Undirected graph $G(N,A)$ construct k copies $G_i(N_i,A_i)$ of this graph. Pick a vertex $v \in N$. Let $u^1, u^2, \ldots, u^d$ be the vertices adjacent to v in G (i.e. $(u^i,v) \in A$ $1 \leq i \leq d$). Define $H_j(V_j.E_j)$ to be the graph with $V_j = \bigcup_{i=1}^{k} N_i$ and

$$E_j = \bigcup_{i=1}^{k} A_i \bigcup \{(u_1^j, v_{i+1}) \mid 1 \leq i < k\} \bigcup \{(u_k^j, v_1)\}$$

Clearly, if G has a Hamiltonian Cycle then, for some j, H_j has a cycle cover containing exactly one cycle (as for some j (v,u^j) are adjacent in the Hamiltonian cycle and using the images of the edges of this cycle in the subgraphs G_i (except for the images of the edge (v,u^j)), together with the edges $\{(u_i^j, v_{i+1}) \mid 1 \leq i < k\} \bigcup \{(u_k^j, v_1)\}$ one obtains a Hamiltonian cycle for E_j). If G has no Hamiltonian cycle, then the subgraphs G_i each require at least two disjoint cycles to cover their nodes. Consequently, the disjoint cycle cover for j contains at least $k+1$ cycles, $1 \leq j \leq k$. For any ε, one may choose a suitable k such that from the approximate solutions to H_j, $1 \leq j \leq k$ one can decide whether or not G has a Hamiltonian cycle (i.e. choose $k > (1 + \varepsilon)$).

Note that the above proof also works for the case of edge disjoint cycle covers.

(vi) Just consider the reduction:

Sum of Subsets α ε-approximate 0-1 Integer Programming.

i.e. $\min \; 1 + k(m - \Sigma w_i \delta_i)$

subject to $\Sigma w_i \delta_i \leq m$

$\delta_i = 0, 1$

(The minimum = 1 iff there is a subset with sum $=m$ otherwise the minimum is $\geq 1 + k$)

(vii) Multicommodity flows α ε-approximate Multicommodity Flows.

In [35] it was shown that multicommodity flows with $f = 1$ was P-Complete. Given a multicommodity network N as in [35] we construct k copies of it and put them in parallel. Another network with a flow $f = 1$ is also coupled to the network as in figure 7.3.2

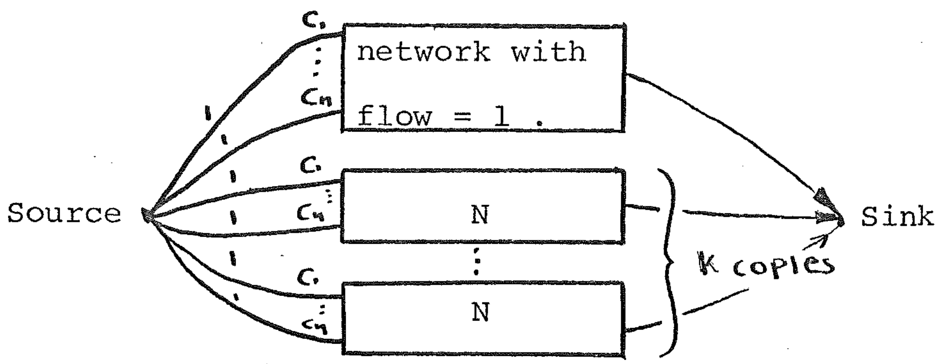


Figure 7.3.2

Clearly, the multicommodity network of figure 7.3.1 has a flow of $k + 1$ iff N has a flow of 1. If N does not have a flow of 1 then the maximum flow in the network of figure 7.3.1 is 1. Hence the approx - imation problem for multicommodity flows is P-Complete.

(viii) Hamiltonian Cycle α ε-approximate Quadra - tic Assignment

Let $G(N,A)$ be an undirected graph with $m = |N|$. The following Quadratic Assignment Problem (QAP) is constructed from G:

$$n = m$$

$$C_{i,j} = \begin{cases} 1 & \text{if } i = j + 1 \text{ and } i < m \\ & \text{or if } i = m \text{ and } j = 1 \\ 0 & \text{otherwise} \end{cases}$$

$$1 \leq i , j \leq n$$

$$d_{k,\ell} = \begin{cases} 1 & \text{if } (k,\ell) \varepsilon A \\ \omega & \text{otherwise} \end{cases} \qquad \text{for } 1 \leq k,\ell \leq m$$

The total cost, $f(\gamma)$, of an assignment, γ, of plants to locations is $\sum_{i=1}^{n} c_{i,j}\, d_{\gamma(i)\gamma(j)}$, $j = i \bmod m+1$ when $\gamma(i)$ is the index of the plant assigned to location i. If G has a Hamiltonian cycle $i_1, i_2, \ldots, i_n, i_1$ then the assignment $\gamma(j) = i_j$ has a cost $f(\gamma) = m$. In case G has no Hamiltonian cycle then at least one of the values $d_{\gamma(i),\gamma(i \bmod m+1)}$ must be ω and so the cost becomes $\geq m + \omega - 1$. Choosing $\omega > (1 + \varepsilon)m$ results in optimal solutions with a value of m if G has a Hamiltonian cycle and value $>(1+\varepsilon)m$ if G has no Hamiltonian cycle. Thus, from an ε-approximate solution, it can be determined whether or not G has a Hamiltonian cycle.

(ix) k-Partition α ε-approximate k-General Partition

We prove this for $k = 2$. The proof is similar for other values of k. From the 2-partition problem the following 2 General Partition problem is constructed: (see also figure 7.3.3)

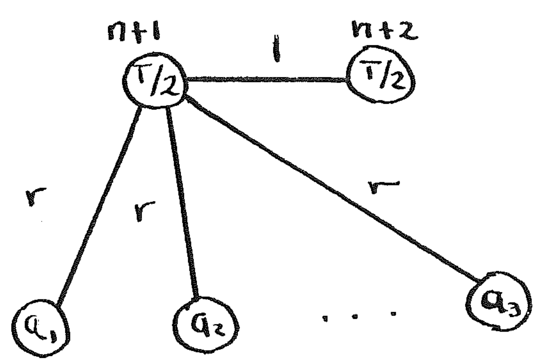


Figure 7.3.3: Numbers in vertices represent vertex weights, and on edges edges weights

$N = \{1,2,\ldots,n+2\}$

$A = \{(i,j) \mid 1 \leq i \leq n, \quad j = n+1\} \cup \{(n+1,n+2)\}$

$$f(u,v) = \begin{cases} r & \text{if } (u,v) \in A \text{ and } 1 \leq u \leq n \\ 1 & (u,v) = (n+1, n+2) \end{cases}$$

$$w(j) = \begin{cases} a_j & 1 \leq j \leq n \\ T/2 & j \geq n+1 \end{cases} \qquad \text{where} \quad T = \sum_{1}^{n} a_i$$

$W = T$

Clearly, there is a solution of value $\geq (nr)/2$ iff the multiset $\{a_1, a_2, \ldots, a_n\}$ has a partition. If there is no partition of this multiset, then the solution value is 1. A suitable choice for r yields the desired result.

(x) ℓ-Chromatic Number α ε-approximate ℓ-MinCluster.

Let $G(N,A)$ be an undirected graph. We may assume $\ell \geq 3$. The following ℓ-MinCluster problem $G_1(N_1,A)$ is constructed:

$N_1 = N$

$A_1 = \{(u,v) \mid u \neq v \text{ and } u,v \in N_1\}$

$$w(u,v) = \begin{cases} 1 & \text{if } (u,v) \notin A \\ k & \text{otherwise} \end{cases}$$

If G is ℓ-colorable then the ℓ-MinCluster problem has a solution with value $< n^2$. If G is not ℓ colorable then the minimum solution value is $\geq k$. Choosing $k > (1 + \varepsilon)n^2$ yields the desired result.

(xi) 2-Partition α ε-approximate Generalized Assignment.

From the partition problem $S = \{a_1, a_2, \ldots, a_n\}$ construct the following generalized assignment problem GS.

$$c_{1,i} = c_{2,i} = 1 \quad , \quad c_{3,i} = k \qquad \text{for} \quad 1 \leq i \leq n$$

$$r_{1,i} = r_{2,i} = r_{3,i} = a_i \qquad \text{for} \quad 1 \leq i \leq n$$

$$b_1 = b_2 = T/2 \quad , \quad b_3 = T$$

where $T = \Sigma\, a_i$

Clearly there is a solution of value n iff the multiset $\{a_1, a_2, \ldots, a_n\}$ has a partition. If there is no partition of this multiset, then the solution value $> k$. Choosing $k > (1 + \varepsilon)n$ yields the desired result.

This completes the proof of the theorem. ∎

BIBLIOGRAPHY

1. Bodin, L.D., "A Graph Theoretic Approach to the grouping of Ordering Data," Networks, 2, 1972, pp. 307-310.

2. Brucker, P., Lenstra, J.K. and Rinooy Kan, A.H.G., "Complexity of Machine Scheduling Problems," Mathematisch Centrum, Amsterdam, Technical Report BW 43/75, April 1975.

3. Bruno, J., Coffman, E.G.,Jr. and Sethi, R., "Scheduling Independent Tasks to Reduce Mean Finish-Time," CACM, Vol. 17, no. 7, July 1974, pp. 382-387.

4. Coffman, E.G.,Jr., and Denning, P.J., Operating Systems Theory, Prentice Hall, Inc., 1973.

5. Coffman, E.G.,Jr., "Computer and Job Shop Scheduling Theory," John Wiley and Sons, to appear.

6. Conover, W.J., "Practical Nonparametric Statistics," John Wiley and Sons Inc., 1971.

7. Conway, R.W., Maxwell, W.L. and Miller, L.W., "Theory of Scheduling," Addison Wesley, 1967.

8. Cook, S.A., "The complexity of Theorem-Proving Procedures," Conference Record of Third ACM Symposium on Theory of Computing, pp. 151-158, 1970.

9. Fishman, G.S., "Concepts and Methods in Discrete Event Digital Simulation," John Wiley and Sons , 1973.

10. Garey, M.R. and Johnson, D., "The Complexity of Near-Optimal Graph Coloring," Bell Laboratories, Technical Report, 1975.

11. Garey, M.R., Johnson, D. and Sethi, R., "Complexity of Flow Shop and Job Shop Scheduling," Pennsylvania State University, Technical Report #168, June 1975.

12. Garfinkel, R.S. and Nemhauser, G.L., Integer Programming, John Wiley and Sons, 1972.

13 Graham, R.L., "Bounds on Multiprocessing Timing Anomalies," SIAM Journal on Applied Mathematics, Vol. 17, No. 2, March 1969.

14 Hopcroft, J.E. and Karp, R.M., "A $n^{5/2}$ Algorithm for Maximum Matchings in Bipartite Graphs," 12th Annual IEEE Symposium on Switching and Automata Theory, 1971, pp. 122-125.

15 Horowitz, E. and Sahni, S., "Exact and Approximate Algorithms for Scheduling Non-Identical Processors," University of Minnesota, Computer Science Technical Report 74-22, September 1974.

16 Horvath, E. and Sethi, R. "Preemptive Schedules for Independent Tasks," Pennsylvania State University, Computer Science Technical Report 162, Feb. 1975.

17 Ibarra, O.H. and Kim, C.E., "Fast Approximation Algorithms for the Knapsack and Sum of Subset Problems," JACM, to appear.

18 Ignall, E. and Schrage, L., "Application of the Branch and Bound Technique to Some Flow Shop Scheduling Problems," Operation Research 13, No. 3, May 1965.

19. Johnson, D., "Approximation Algorithms for Combinatorial Problems," Conference Record of Fifth ACM Symposium on Theory of Computing, 1973.

20. Johnson, D.B. and Lafuente, J.M., "A Controlled Single Pass Classification Algorithm with Application to Multilevel Clustering," Information Science and Retrieval, Scientific Report #ISR-18, Oct., 1970.

21. Karp, R.M., "Reducibility Among Combinatorial Problems," in Complexity of Computer Computations, R.E. Miller and J.W. Thatcher, eds., Plenum Press, N.Y., 1972, pp. 85-104.

22. Knuth, D.E., "Art of Computer Programming," Vols. 1, 2 and 3, Addison Wesley.

23. Knuth, D.E., "A Terminological Proposal," SIGACT News, Jan. 1974, Vol. 6, No. 1, pp. 12-18. See also SIGACT News, April, 1974.

24. Kohler, W.H. and Steiglitz, K., "Exact, Approx - imate and Guaranteed Accuracy Algorithms for the Flow Shop Problem n/2/F/$\overline{F}$," JACM, Vol. 22, #1, Jan, 1975, pp. 106-114.

25. Lilliefors, H.W., "On the Kolmogorov-Smirnov Test for Normality with Mean and Variance Unknown," JASA, 62, 1967, pp. 339-402.

26. Lilliefors, H.W., "On the Kolmogorov-Smirnov Test for the Exponential Distribution with Mean Unknown ," JASA, 64, 1969, pp. 387-389.

27. Lindgren, B.W., "Statistical Theory," The MacMil - lian Company, New York, 1962.

28. Liu, C.L., "Introduction to Combinatorial Mathema- tics," McGraw Hill, 1968.

29. Liu, J.W.S. and Liu, C.L., "Bounds on Scheduling Algorithms for Heterogeneous Computing Systems," IFIP, August 1974, pp. 349-353.

30. Liu, J.W.S. and Yang, A., "Optimal Scheduling of Independent Tasks on Heterogeneous Computing Systems," 1974, ACM National Conference, pp.38-45.

31. Lukas, J.A., "Combinatorial Solution to the Part- itioning of General Graphs," IBM Journal of Re - search and Development, Vol. 19, No. 2, March 1975.

32. Miller, I. and Freund, J.E., "Probability and Statistics for Engineers," Prentice Hall, 1965.

33. Rosenkrantz, D.J., Stearns, R.E. and Lewis, P.M., "Approximate Algorithms for the Travelling Sales- person Problem," 15th Annual IEEE Symposium on on Switching and Automata Theory, 1974, pp.33-42.

34. Sahni, S., "Algorithms for Scheduling Independent Tasks," University of Minnesota, Technical Report #74-16, 1974.

35. Sahni, S., "Computationally Related Problems." SIAM Journal on Computing, Vol. 3, No. 4, Dec. 1974, pp. 262-279.

36. Sahni, S., "Approximate Algorithms for the 0/1 Knapsack Problem", JACM, Vol. 22, No. 1, pp. 115- 124, Jan. 1975.

37. Ullman, U.D., "Polynomial Complete Scheduling Problems," IBM Conference on Operating Systems, Oct. 1973.

38. Ross, G.T. and Soland, R.M., "A Branch and Bound Algorithm for the Generalized Assignment Problem" Mathematical Programming, Vol. 8, 1975,pp. 91-103.