Scheduling Algorithms for a Chain-like Task System

Chan, Chi-Lok[*], Gilbert, Young[†]
The Chinese University of Hong Kong

March 21, 1995

Abstract

The optimal allocation of a chain-like task system on the chain-like network computers was first presented by Bokhari with time complexity $O(m^3n)$[1], where m and n denote the number of modules and the number of processors respectively. Sheu and Chiang improved it and gave an $O(\min\{m,n\}m^2)$ algorithm[2]. Hsu had further developed a two phase approach with the worst case time complexity of $O(m+(m'-n)^2n)$[3] where m' denotes the number of the remaining modules after the merge phase. For all of these papers, none of them answers the decision version of this problem, that is whether there exists a schedule with schedule length less than a given deadline. In this paper, two algorithms are presented. The first one answers the decision problem and gives a feasible schedule if there exists one. It is an optimal algorithm with time complexity of $O(m)$. The second one finds an optimal schedule in $O(m+m'\log m'+m'^2-n^2)$ time using the first algorithm. It is shown that our approach is more efficient than Hsu's one if $m'-n=\omega(\sqrt{\log n})$. Furthermore, our algorithm returns an optimal schedule which uses the minimal number of processors.

Keywords: bottleneck processor, chain-like task system, completion time, contiguity constraint, feasible length-$\mathcal{K}$ schedule, layered graph, merged module, optimal schedule, schedule length, un-mergeable modules.

1 Introduction

The problem we investigate in this paper is the allocation of a chain-like task system on the chain-like network computers which was first presented by Bokhari [1]. A chain-like task system consists of m modules scheduled on n processors. Each module is associated with an execution time and each module communicates with its neighboring modules with a communication cost (Fig. 1).

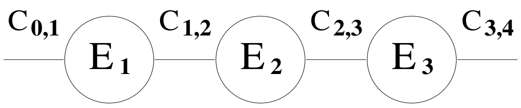


Fig. 1: A chain-like task system

*E-mail: clchan@cs.cuhk.hk
†E-mail: young@cs.cuhk.hk

A typical application of chain-like task system is pipelining system with an essentially unending series of data. The maximum rate of processing is now determined by the processor that takes the longest amount of time (schedule length) to perform its tasks (the bottleneck processor). The optimal allocation problem is to find a schedule in which the bottleneck processor has the minimal execution time. Bokhari had presented an algorithm with time complexity $O(m^3 n)$ [1]. Sheu and Chiang showed that a more general assignment problem can be solved with time complexity $O(\min\{m,n\}m^2)$ [2]. Hsu proposed an approach which is composed of two phases: merge phase and assignment phase[3]. The time complexity of the merge phase is $O(m)$ and the overall complexity is $O(m+(m'-n)^2 n)$ where m' denotes the number of the remaining modules after the merge phase $(m' \leq m)$. These three algorithms use a similar approach, a layered graph method. In this paper, we propose an $O(m)$ algorithm for answering whether there exists a schedule with schedule length no greater than a constant $\mathcal{K}$. Furthermore, we propose another approach for finding an optimal schedule in $O(m + m' \log m' + m'^2 - n^2)$ time.

The remainder of this paper is organized as follows: section 2 states the problem assumptions and notations used in this paper. In section 3, an optimal greedy algorithm for the decision problem is presented. Section 4 proposes another approach for finding an optimal schedule. Finally, we draw some concluding remarks in the last section.

2 Problem Assumptions and Notations Definition

1. A chain-like system consists of m modules, $t_1, t_2, \ldots, t_m$, scheduled on n processors. Execution time of module i is denoted as E_i and communication time of adjacent modules i and $i+1$ is denoted as $C_{i,i+1}$ $(0 \leq i \leq m)$. All of execution times and communication times are positive except that $C_{0,1}$ and $C_{m,m+1}$ equal zero.

2. Adjacent modules must be assigned to adjacent processors or the same processor, which is called contiguity constraint [1].

3. If adjacent modules are assigned to the same processor, their communication (intra-communication) time is assumed to be zero.

4. The elapsed time required to complete the execution and communication for a module (may be a sequence of adjacent modules) on a processor is called completion time. The completion time of a module i,denoted as CMP_i, is equal to $C_{i-1,i} + E_i + C_{i,i+1}$. The completion time of modules $i, i+1, \ldots, j$, denoted as $CMP_{i \oplus i+1 \oplus \cdots \oplus j}$, is equal to $C_{i-1,i} + \sum_{k=i}^{j} E_k + C_{j,j+1}$.

5. For a schedule S of m modules on n processors, if $t_j, t_{j+1}, \ldots, t_k$ are modules scheduled on processor i, $1 \leq j \leq k \leq m$ and $1 \leq i \leq n$, $CP_i(S)$ denotes the completion time of processor i, ie.

$$CP_i(S) = CMP_{j \oplus \cdots \oplus k}$$

6. The schedule length of a schedule, $\mathcal{L}(S)$, is defined as the maximum completion time of all processors in S, ie.

$$\mathcal{L}(S) = \max_i \{CP_i(S)\}$$

7. A schedule is said feasible if it satisfies the contiguity constraint. An optimal schedule, S°, is a feasible schedule with minimal schedule length, ie.

$$\mathcal{L}(S^\circ) \leq \mathcal{L}(\mathcal{S}')$$

where S' is any possible feasible schedule.

8. Two modules, t_i and t_{i+1} are mergeable if the completion time of the merged module is no greater than the maximum completion time of t_i and t_{i+1} (Fig. 2),

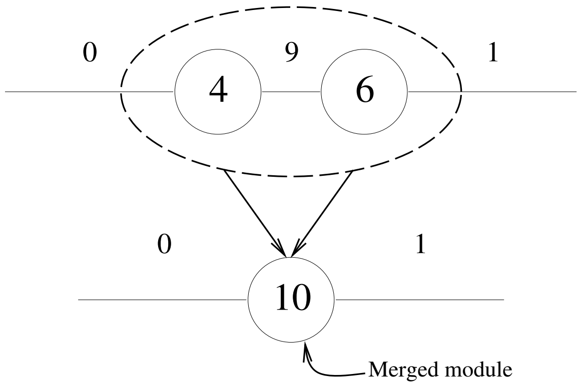


Fig. 2: Merge two modules.

ie.

$$CMP_{i \oplus i+1} \leq \max\{CMP_i, CMP_{i+1}\}$$

9. Two modules, t_i and t_{i+1} are un-mergeable if they are not mergeable, ie.

$$C_{i,i+1} < E_i + C_{i-1,i} \quad \text{and} \quad C_{i,i+1} < E_{i+1} + C_{i+1,i+2}$$

3 An Algorithm for the Decision Problem

In this section, we present an algorithm for answering whether there exists a schedule S such that the schedule length of S, $\mathcal{L}(S)$, is less than or equal to $\mathcal{K}$, a given constant (called a decision problem).

The following lemmas are useful for the proof of correctness of our algorithm.

Lemma 1 *For m' un-mergeable modules, $t_1, \ldots, t_{m'}$, in a chain-like task system, $1 \leq i \leq p \leq j \leq m'$.*

$$CMP_{i \oplus i+1 \oplus \cdots \oplus j} \geq CMP_{p \oplus p+1 \oplus \cdots \oplus j}$$

Proof: By induction on p, for any $1 \leq i \leq p \leq j \leq m'$,

For $p = i$,

$$CMP_{p \oplus p+1 \oplus \cdots \oplus j} = CMP_{i \oplus i+1 \oplus \cdots \oplus j}$$

ie. it is true for $p = i$
Assume it is true for $p = k < j$
ie. $CMP_{i \oplus i+1 \oplus \cdots \oplus j} \geq CMP_{k \oplus k+1 \oplus \cdots \oplus j}$
For $p = k + 1$,

$$
\begin{aligned}
CMP_{k+1\oplus\cdots\oplus j} &= CMP_{k\oplus\cdots\oplus j} - C_{k-1,k} - E_k + C_{k,k+1} \\
&< CMP_{k\oplus\cdots\oplus j} \qquad (C_{k,k+1} < C_{k-1,k} + E_k) \\
&\leq CMP_{i\oplus\cdots\oplus j} \qquad \text{(induction hypothesis)}
\end{aligned}
$$

ie. It is true for $p = k+1$
So, it is true for $i \leq p \leq j$ by induction.

□

Lemma 2 *For m' un-mergeable modules, $t_1, \ldots, t_{m'}$, in a chain-like task system, $1 \leq i \leq p \leq j \leq m'$.*

$$CMP_{i\oplus i+1\oplus\cdots\oplus j} \geq CMP_{i\oplus i+1\oplus\cdots\oplus p}$$

Proof: Similar to the proof of Lemma 1. □

Definition: A schedule is called a feasible length-$\mathcal{K}$ schedule if

$$\mathcal{L}(S) \ = \ max_{1\leq i\leq n}\{CP_i(S)\} \leq \mathcal{K}$$

where $\mathcal{K}$ is an integer constant.

The following algorithm decides whether there exists a feasible length-$\mathcal{K}$ schedule. The input modules are assumed un-mergeable. It schedules as many modules on a processor as possible successively and the completion time of each processor is no greater than $\mathcal{K}$. If all modules can be scheduled properly, it returns 'Yes' and the corresponding schedule; otherwise it returns 'No'.

Algorithm 1: (Greedy Algorithm)

Input: m' un-mergeable modules in a chain-like task system with their execution times and communication times, n processors and $\mathcal{K} \in \mathcal{Z}^+$

Output: 'Yes' and gives a feasible length-$\mathcal{K}$ schedule S if there exists one; otherwise, 'No'.

(1) $j = 1, CP = 0$ // j denotes the processor no.
(2) for i =1 to m' // for each module i,
(3)   if $CP + E_i + C_{i,i+1} - C_{i-1,i} > \mathcal{K}$ //if we cannot schedule t_i on p_j
(4)   if $j = n$ // no processor left
(5)   return 'No'
(6)   else
(7) $CP = C_{i-1,i} + E_i + C_{i,i+1}$ // $CP = CMP_i$
(8)   if $(CP > \mathcal{K})$ // schedule length $> \mathcal{K}$
(9)   return 'No'
(10) $j = j+1$
(11)   else
(12) $CP = CP + E_i + C_{i,i+1} - C_{i-1,i}$ // schedule t_i on p_j
(13) $S[i] = j$
(14) return 'Yes' and S

For the above algorithm, CP is the current completion time of processor j and S is an array such that $S[i] = j$ if t_i is scheduled on processor j.

Since each module is examined only once and there are m' modules (line 2), the time complexity of Algorithm 1 is $O(m')$. For a mergeable chain-like task system with m modules, $O(m)$ time is needed for merging [3] and $O(m')(m' \leq m)$ time is used for Algorithm 1, the overall time complexity is $O(m)$. The following example illustrates the above algorithm for a chain-like task system with 4 modules and $\mathcal{K}$ equals 25.

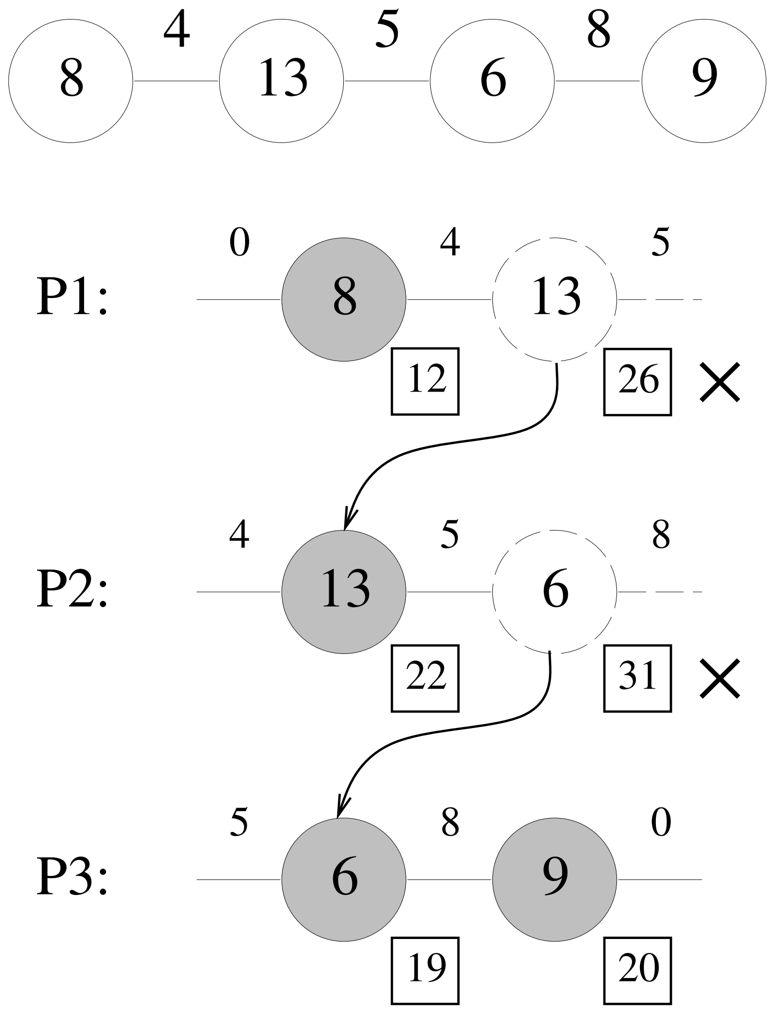


Figure 3: Example for Algorithm 1

Theorem 3 *For m' un-mergeable modules scheduled on n processors, Algorithm 1 returns a feasible length-$\mathcal{K}$ schedule S_n iff there exists a feasible length-$\mathcal{K}$ schedule S'_n for the m' modules on n processors.*

Proof:
Part 1: "$\Rightarrow$" part.

If Algorithm 1 answers 'Yes' and returns S_n, it is obvious that S_n is a feasible length-$\mathcal{K}$ schedule because we schedule t_j on processor i iff the completion time of processor i is less than or equal to $\mathcal{K}$, $1 \leq j \leq m'$, ie.

$$\max_{1 \leq i \leq n} \{CP_i(S_n)\} \leq \mathcal{K}$$

we can let

$$S'_n = S_n \qquad \text{and} \qquad \max_{1 \leq i \leq n}\{CP_i(S'_n)\} = \max_{1 \leq i \leq n}\{CP_i(S_n)\} \leq \mathcal{K}$$

Part 2: "$\Leftarrow$" part.
Suppose there exists a feasible length-$\mathcal{K}$ schedule S'_n and S_n is the schedule obtained from Algorithm 1 (suppose we do not know whether Algorithm 1 answers 'Yes' or 'No' and S_n may not be a feasible length-$\mathcal{K}$ schedule)
We have to prove that

$$\max_{1\le i\le n}\{CP_i(S'_n)\} \le \mathcal{K} \Rightarrow \max_{1\le i\le n}\{CP_i(S_n)\} \le \mathcal{K}$$

by induction on n, the number of processor.

For $n = 1$, there is only one processor so that all m' modules are scheduled on processor 1 in S'_1. By Algorithm 1, we always schedule as many modules as possible on each processor so that $S_1 = S'_1$ and

$$\max_{i=1}\{CP_i(S_1)\} = \max_{i=1}\{CP_i(S'_1)\} \le \mathcal{K}$$

So, it is true for $n = 1$.

Assume it is true for $n = k$, ie. if there exists a feasible length-$\mathcal{K}$ schedule S'_k, then Algorithm 1 returns a feasible length-$\mathcal{K}$ schedule S_k,

$$\max_{1\le i\le k}\{CP_i(S'_k)\} \le \mathcal{K} \Rightarrow \max_{1\le i\le k}\{CP_i(S_k)\} \le \mathcal{K}$$

For $n = k+1$, let S'_{k+1} is any feasible length-$\mathcal{K}$ schedule and S_{k+1} is the schedule returned by Algorithm 1. Firstly, for S'_{k+1}

$$\max_{1\le i\le k+1}\{CP_i(S'_{k+1})\} \le \mathcal{K}$$

We compare the modules scheduled on processor 1 in S'_{k+1} and S_{k+1}. Let n'_1 and n_1 be the number of modules scheduled on processor 1 in S'_{k+1} and S_{k+1} respectively,

Case 1: $n'_1 = n_1$

For processor 1, the schedules in S'_{k+1} and S_{k+1} are the same, ie.

$$CP_1(S_{k+1}) = CP_1(S'_{k+1}) \le \mathcal{K} \qquad \ldots \text{(i)}$$

Consider the modules scheduled on processor 2 to k+1 for both schedules, they schedule the same modules on k processors. By induction hypothesis,

$$\begin{aligned} & \max_{2\le i\le k+1}\{CP_i(S'_{k+1})\} \le \mathcal{K} \\ \Rightarrow\ & \max_{2\le i\le k+1}\{CP_i(S_{k+1})\} \le \mathcal{K} \qquad \ldots \text{(ii)} \end{aligned}$$

By (i) and (ii),

$$\max_{1\le i\le k+1}\{CP_i(S_{k+1})\} \le \mathcal{K}$$

Case 2: $n'_1 < n_1$

In this case, for S'_{k+1}, we can re-schedule some modules $(t_{n'_1+1}, t_{n'_1+2}, \ldots, t_{n_1})$ on processor 1 and remove these modules from some other processors. If there are some processors becoming no module scheduled on them after re-scheduling, we can relabel the processor so that the contiguity constraint is not violated. We call the resultant schedule S''_{k+1}. The completion time of processor 1 in S''_{k+1} is less than or equal to $\mathcal{K}$ because we can schedule n_1 modules on processor 1 by Algorithm 1 only if $CMP_{1\oplus 2\oplus\cdots\oplus n_1} \le \mathcal{K}$. For S''_{k+1}, compared with S'_{k+1}, the number of module remains in processor j, $2 \le j \le k+1$, is reduced (because some modules or none are removed). By Lemma 1, the completion time of processor j in S''_{k+1} is less than or equal to that in S'_{k+1}, ie.

$$CP_j(S''_{k+1}) \le CP_j(S'_{k+1}), 2 \le j \le k+1 \qquad \text{and}$$

$$\max_{2\le i\le k+1}\{CP_i(S''_{k+1})\} \le \max_{2\le i\le k+1}\{CP_i(S'_{k+1})\} \le \mathcal{K}$$

So, S''_{k+1} is also a feasible length-$\mathcal{K}$ schedule and the number of modules scheduled on processor 1 is equal to n_1. By case 1, S_{k+1} is a feasible length-$\mathcal{K}$ schedule as well.

Case 3: $n'_1 > n_1$

In this case, t_1 to t_{n_1} are scheduled on processor 1 in S_{k+1} and t_1 to $t_{n'_1}$ are scheduled on processor 1 in S'_{k+1} where $n'_1 > n_1$.

However, t_{n_1+1} cannot be scheduled on processor 1 by Algorithm 1 (line 3), so

$$CMP_{1\oplus\cdots\oplus n_1\oplus n_1+1} > \mathcal{K}$$

By Lemma 2, $n' \geq n_1 + 1$

$$CP_1(S'_{k+1}) = CMP_{1\oplus\cdots\oplus n'_1} \geq CMP_{1\oplus\cdots\oplus n_1\oplus n_1+1} > \mathcal{K}$$

which is contradictory to the fact that S'_{k+1} is a feasible length-$\mathcal{K}$ schedule. The case $n'_1 > n_1$ never occurs.

By case 1, 2 and 3, if S'_{k+1} is a feasible length-$\mathcal{K}$ schedule, S_{k+1} is a feasible length-$\mathcal{K}$ schedule as well, ie. it is true for $n = k + 1$.

By induction on n, Theorem 3 is true for any number of processor. □

4 An Algorithm for Optimal Schedule

In the previous section, we present an $O(m)$ algorithm for deciding the existence of a feasible length-$\mathcal{K}$ schedule. Now we can apply this algorithm to find an optimal schedule.

4.1 Algorithm

It is obvious that the schedule length of any schedule is equal to the longest completion time of the n processors, ie. the completion time of the bottleneck processor. We can construct a set, Λ', which contains all possible values of schedule length, By Hsu's result, we can only consider a chain-like task system with un-mergeable modules only [3] and

$$\Lambda' = \{CMP_{i\oplus\cdots\oplus j} : 1 \leq i \leq j \leq m'\}$$

By Hsu's result, for m' un-mergeable modules scheduled on n processors, if $m' \leq n$, we can obtain an optimal schedule by assigning one merged module to one processor [3]. On the other hand, if $m' > n$, it can be shown that ,for any optimal schedule, there exists a bottleneck processor such that at most $m' - n + 1$ modules have been scheduled on it. As a result, we can construct a set which covers all possible optimal schedule lengths

$$\Lambda = \{CMP_{i\oplus\cdots\oplus j} : 1 \leq i \leq j \leq m' \; and \; j - i \leq m' - n\}$$

where Λ covers all possible minimal schedule length and

$$|\Lambda| = \sum_{i=n}^{m'} i = \sum_{i=1}^{m'} i - \sum_{i=1}^{n-1} i = O(m'^2 - n^2)$$

Now, we can find the median of the elements in Λ and apply Algorithm 1 as a feasibility test. Then we can eliminate half of the elements in Λ. Repeat the steps until there is only one element in Λ and this is the optimal schedule length. The following algorithm summarize the above method.

Algorithm 2:

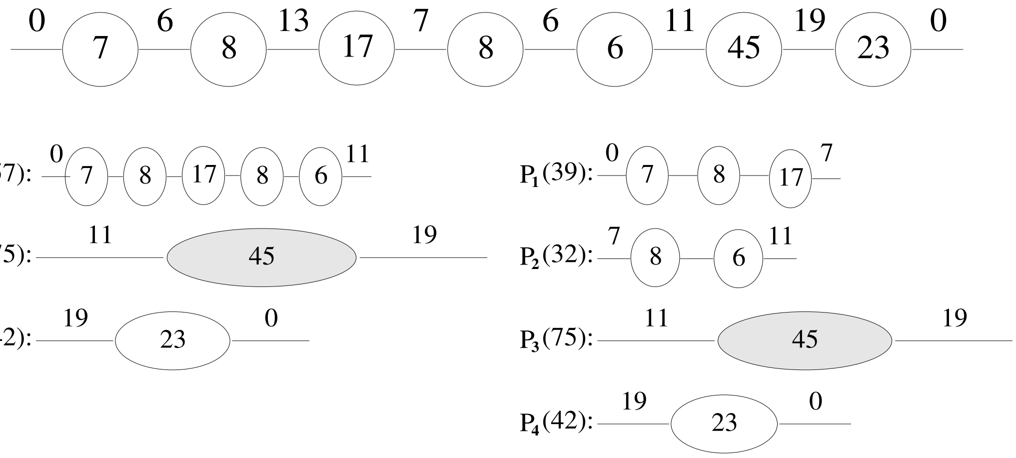
Input: m modules in a chain-like task system with their execution times and communication times, n processors

Output: An optimal schedule S.

(1) merge the m modules into m' un-mergeable modules by Hsu's algorithm
(2) if $m' \leq n$
(3)   return S where $S[i] = i, 1 \leq i \leq m'$
(4) construct a set Λ where
    $\Lambda = \{CMP_{i \oplus \cdots \oplus j} : 1 \leq i \leq j \leq m' \text{ and } j - i \leq m' - n\}$
(5) find the median of the elements in Λ
(6) apply Algorithm 1 with the median as the deadline, $\mathcal{K}$
(7) if Algorithm 1 returns 'Yes'
(8)   remove all elements which are larger than $\mathcal{K}$ in Λ
(9) otherwise
(10)   remove all elements which are no greater than $\mathcal{K}$ in Λ
(11) if $|\lambda| \neq 1$
(12)   repeat step 5
(13) return the corresponding schedule S

4.2 Number of Processor Used in the Optimal Schedule

The optimal schedule obtained by our algorithm, Algorithm 2, can be shown to use the minimal number of processors. By Hsu's algorithm [3], all n processors must be utilized. The following diagrams show the difference,



Optimal Schedule return by Searching Based Algorithm   Optimal Schedule return by Hsu's Algorithm

Figure 4: Example 1.

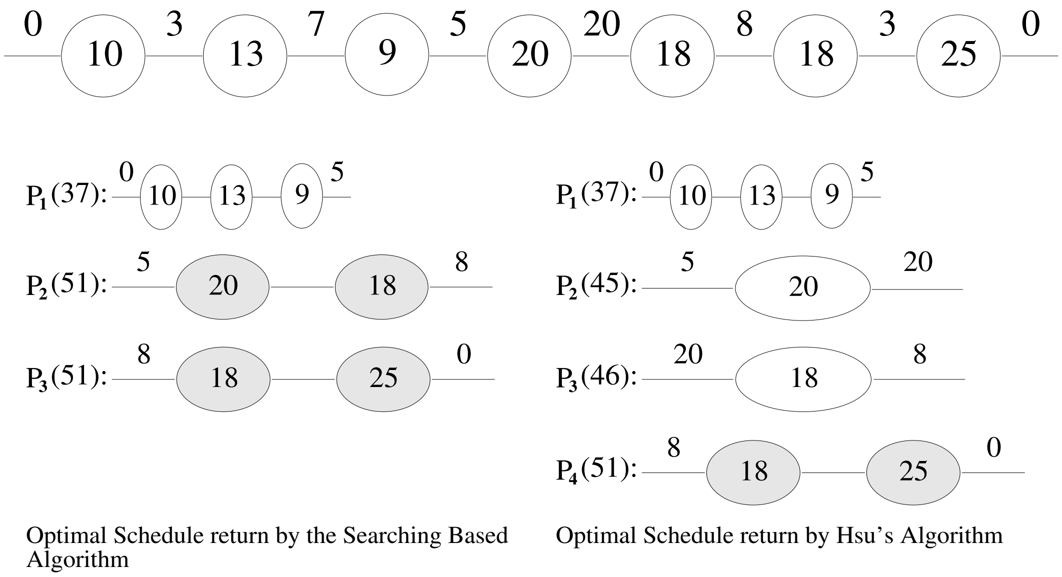


Figure 5: Example 2.

In the first example, there is one large merged module (the 6th module) so that more than $m' - n + 1$ modules can be scheduled on other processor ($p1$). In the second example, there exists an optimal schedule such that there are more than one bottleneck processors. In this case, our algorithm may give an optimal schedule using less than n processors.

4.3 Complexity Analysis

The time complexity of Algorithm 2 consists of several parts. Firstly, $O(m)$ time for merging the modules(line 1). For the construction of Λ (line 4), we need $O(m'^2 - n^2)$ time. For step 5 to 12, we can find the median of N elements in $O(N)$ [4][5][6]. The feasibility test, Algorithm 1, takes $O(m')$ time and we can remove the elements (line 8, 10) in $O(N)$ time. Let $X(N)$ be the time taken for step 5 to 12,

$$X(N) = X(\frac{N}{2}) + N + m'$$

$$X(m'^2 - n^2) = O(m' \log m' + m'^2 - n^2)$$

As a result, the total time complexity of Algorithm 2 is

$$T(m, n) = O(m + m' \log m' + m'^2 - n^2)$$

The time complexity of Hsu's algorithm is [3]

$$T'(m, n) = O(m + (m' - n)^2 n)$$

Comparison between the two algorithms

If $m' \leq n$, Algorithm 2 is identical to Hsu's algorithm and the time complexity is $O(m)$.

If $m' > n$, let $m' - n = f(n)$, $f(n) > 0$ for all n,

$$T(m, n) = O(m + m' \log m' + m' f(n))$$

$$T'(m, n) = O(m + n f^2(n))$$

Case 1: $f(n) = o(\sqrt{\log n}) \Rightarrow m' = O(n)$

$$T(m,n) = O(m + n\log n + nf(n)) = O(m + n\log n)$$
$$T'(m,n) = O(m + nf^2(n)) \le O(m + n\log n) = T(m,n)$$

Case 2: $O(\log n) = f(n) = \omega(\sqrt{\log n}) \Rightarrow m' = O(n)$

$$T(m,n) = O(m + n\log n + nf(n)) = O(m + n\log n)$$
$$T'(m,n) = O(m + nf^2(n)) \ge O(m + n\log n) = T(m,n)$$

Case 3: $O(n) = f(n) = \Omega(\log n) \Rightarrow m' = O(n)$

$$T(m,n) = O(m + n\log n + nf(n)) = O(m + nf(n))$$
$$T'(m,n) = O(m + nf^2(n)) \ge O(m + nf(n)) = T(m,n)$$

Case 4: $f(n) = \Omega(n) \Rightarrow m' = O(f(n))$

$$T(m,n) = O(m + f(n)\log f(n) + f^2(n)) = O(m + f^2(n))$$
$$T'(m,n) = O(m + nf^2(n)) \ge O(m + f^2(n)) = T(m,n)$$

For case 2, 3 and 4, $T'(m,n) \ge T(m,n)$. Combining these cases, $T(m,n) \le T'(m,n)$ if $f(n) = \omega(\sqrt{\log n})$. In other words, our algorithm is asymptotically better than Hsu's algorithm if $m - n = \omega(\sqrt{\log n})$.

5 Conclusion

In this paper, two algorithms are presented for answering the decision problem, and optimally assigning a chain-like task system on a chain-like network computers. The time complexity of the algorithm for the decision problem is $O(m)$ where m is the number of modules in the system. This is an optimal algorithm which returns a feasible schedule if there exists one. For optimal schedule, Hsu has improved the Sheu and Chiang's algorithm [2] and solved this problem by a two phase algorithm with time complexity $O(m + (m' - n)^2)$ if $m' > n$ where m' is the number of modules after the merge phase [3]. If $m' \le n$, the time complexity is $O(m)$. All of the existing algorithms use a layered graph approach. In this paper, we use a new approach which is based on a searching strategy to find an optimal schedule. The algorithm (Algorithm 2) presented in this paper is identical to Hsu's algorithm if $m' \le n$. For $m' > n$, the time complexity is $O(m + m'\log m' + m'^2 - n^2)$. In the case of $m' - n = \omega(\sqrt{\log n})$, our algorithm is more efficient than Hsu's algorithm. Furthermore, the optimal schedule obtained by our algorithm uses the minimal number of processors.

References

[1] Shahid H. Bokhari. Partitioning problems in parallel, pipelined, and distributed computing. *IEEE Transactions on Computers*, 37:48–57, January 1988.

[2] J.P. Sheu and Z.F. Chaing. Efficient allocation of chain-like task on chain-like network computers. *Information Processing Letters*, 36:241–245, 1990.

[3] C.C. Hsu. A two-phase approach for the optimal assignment of a chain-like task on a chain-like network computer. Technical report, National Taiwan Institute of Technology, 1993.

[4] Manuel Blum, Robert W. Floyd, Vaughan Pratt, Rivest Ronald L., and Robert E. Tarjan. Time bounds for selection. *Journal of Computer and System Science*, 7(4):448–461, 1973.

[5] C.A.R. Hoare. Algorithm 63(partition) and algorithm 65(find). *Communication of ACM*, 4(7):321–322, 1961.

[6] Robert W. Floyd and Ronald L. Rivest. Expected time bounds of selection. *Communication of ACM*, 18(3):165–172, 1975.