Res. Rep. Fac. Eng. Mie Univ., Vol. 26, pp. 51−68 (2001)

Original Paper

Code Scheduling for Multi-Processor Systems

Yoshikazu MATSUBARA*, Michio OYAMAGUCHI and Yoshikatsu OHTA
(Department of Information Engineering)

(Received September 17, 2001)

Abstract

This paper considers the scheduling problem of multi-processor systems under the condition that each processor is uniform and every instruction executes in one cycle. We propose two linear time scheduling algorithms which are respectively applicable to the two classes of dependence graphs, one of which consists of graphs with instructions of in-degree at most 2, and the other class of in-degree at most 3. We show that these algorithms produce optimal schedulings when these dependence graphs satisfy some more restricted conditions.

Key Words: Code Scheduling, Compiler Optimization, Multi-Processor

1 Introduction

Processors such as superscaler and VLIW machines can execute many instructions in parallel. In order to achieve high performance of these machines, code(instruction) scheduling by compiler plays an important role [1]. But, the scheduling problem is NP-complete in general [2]. So, it is expected to obtain heuristic algorithms producing a good approximating (or near optimal) solution to this problem. Dependences between instructions are represented by a directed acyclic graph (DAG) called a dependence graph: the vertices represent the instructions and the edges represent the dependences. The scheduling problem is the problem of finding out an optimal schedule for a dependence graph and a number of processors k. Our previous paper has proposed a heuristic scheduling algorithm which produces in linear time under the assumption that every dependence graph has only instructions of in-degree at most 2 (i.e., the number of the parents of each instruction is at most 2). The reason we restrict the number of the parents of each instruction to at most 2 is that where we use 3-address code as program's intermediate code and schedule its instructions before register allocation, in many cases the number of parents of each instruction is limited to at most 2 because no antidependence[1] induced by register redefinition occurs. In this algorithm, the set of vertices of a given DAG are divided into subsets $S_h(0 \leq h \leq H)$ of vertices with the critical path length (the height of the vertices)h (where H is the height of the DAG), and a local allocation method is proposed which schedules each subset $S_h(h > 0)$ using ordering the connected components of the subgraph consisting of S_h and S_{h-1} according to some priority. We have shown that this algorithm produces an optimal scheduling if $|S_h| \geq 2k - 1$ holds for every S_h, where $|S_h|$ is the number of vertices in S_h [5].

In this paper, we first extend this previous algorithm to the one in which a new local allocation method schedules each S_h by investigating the subgraph consisting of three subsets S_h, S_{h-1}, S_{h-2}. (Note that S_{h-2} is added and investigated.) We show that this new algorithm operates in linear time and produces an optimal solution if $|S_h| \geq \frac{3}{2}k$ for every S_h. This extends the previous result since $|S_h| \geq 2k - 1$ is relaxed to $|S_h| \geq \frac{3}{2}k$.

---

*Currently with Hitachi Corporation

[1]Antidependence is such a dependence between instructions that when some register is reused, the following instructions cannot write a new value into this register before all the preceding instructions referencing to this register's old value.

Next, we consider the case that the in-degree of every vertex in a given DAG is at most 3. Note that when a program to be scheduled handle array variables, the number of parents of some vertex(instruction) might be 3. In this paper, we propose a linear time heuristic algorithm which is applicable to a DAG in which the in-degree of every vertex is at most 3. And we show this algorithm gives an optimal scheduling if $|S_h| \geq 2k-1$ and when $k > 2$ ($|S_h| \geq 2k$ if $k = 2$) holds for every S_h.

2 Instruction Scheduling

For instruction scheduling, we must obey data dependences between instructions in order not to change the result of the program execution. These data dependences are represented by a DAG usually: instructions are represented by vertices and data dependences between instructions are represented by directed edges. For a DAG, we define the critical path length of an instruction by the maximum path length from this instruction to a leaf (i.e., the height of this instruction). It is well known that given a DAG and a number of processors k, the optimal scheduling problem to minimize the execution cycles is NP-complete in general [2].

In this paper, we consider the scheduling problem under the assumption that the following conditions (1)-(3) hold.

(1) The number of parents of every instruction is at most n (where $n = 2$ or $n = 3$).

(2) Every instruction completes its execution in the same number of cycles.

(3) All the processors are uniform, i.e., have the same function.

Even if we assume the conditions (1)-(3), the scheduling problem is still NP-complete in general [5]. Whether the scheduling problem in the case where the condition (2) and (3) hold and the number of processors is 3 is NP-complete or not is unsolved yet [4]. In the case of $k = 2$, polynomial algorithms have been known [6][7].

Henceforth, we consider a fixed DAG and assume that H is the maximum critical path length (i.e., the height of the DAG), S_h is the set of instructions whose critical path length is $h (0 \leq h \leq H)$, and k is the number of processors ($k \geq 2$).

[Notation] We denote the number of elements of a set S by $|S|$. For a real number r, we denote the minimum integer m satisfying $r \leq m$ by $\lceil r \rceil$ and the maximum integer l satisfying $l \leq r$ by $\lfloor r \rfloor$.

3 Scheduling of DAG in which every node has at most 2 parents

3.1 Allocation Algorithm

The scheduling algorithm proposed in this paper allocates instructions in the decreasing order of the length of critical path (i.e., in the order of $S_H, S_{H-1}, \cdots, S_0$) as shown at Fig.1 and uses two methods, called "local allocation method" and "extended local allocation method" to allocate instructions.

Allocation Algorithm A_0

(0) Initialization: $l_{H+1} := 0$; $h := H$; $t := 1$

(1)

- case of $l_{h+1} = 0$: $S'_h := S_h$.
- case of $l_{h+1} \neq 0$:

  Let $V \subseteq S_h$ be the set of elements which can be allocated at $k - l_{h+1}$ idle processors at time t. If $|V| \geq k - l_{h+1}$, choose $k - l_{h+1}$ elements from V arbitrarily and let the subset be V'. Otherwise, let $V' = V$. Allocate V' at time t. $S'_h := S_h - V'$; $t := t + 1$

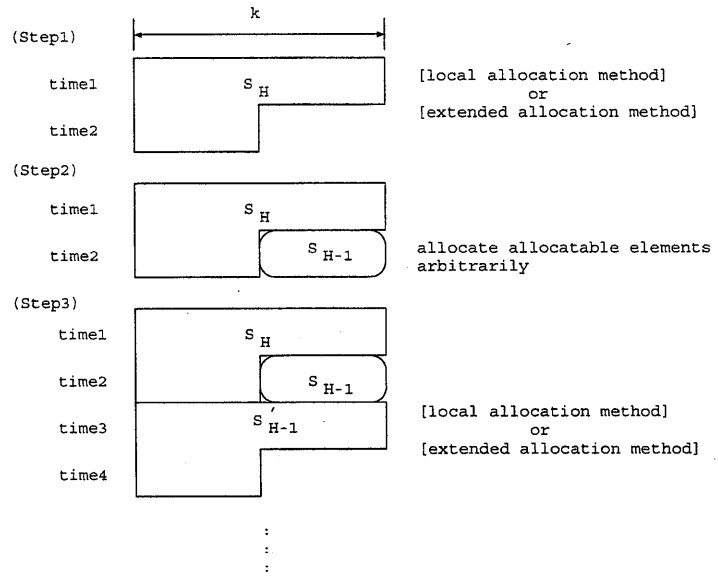

Figure 1: The algorithm for allocation

(2) If $S'_h \neq \emptyset$, allocate all the elements of S'_h as follows:
(2-1) case of $h > 0$:
$l_h := \mid S'_h \mid \mod k$.

- case of $l_h = 0$:
  Allocate all the elements of S'_h in an arbitrary order. $t := t + |S'_h|/k$
- case of $l_h \neq 0$:
  (a-1) case of $h = 1$ or $\exists i(h-2 \leq i \leq h)| S_i | < \frac{3}{2}k$ or $l_h + | S_{h-1} | \geq 2k$
  Allocate all the elements of S'_h by "local allocation method" (described later). $h := h - 1$; $t := t + \lceil |S'_h|/k \rceil - 1$
  (a-2) case of $h \geq 2$ and $\forall i(h-2 \leq i \leq h)| S_i | \geq \frac{3}{2}k$ and $l_h + | S_{h-1} | < 2k$
  Allocate all the elements of S'_h and S_{h-1} by "extended local allocation method" (described later). $h := h - 2$; $t := t + \lceil |S'_h|/k \rceil$

Return to (1).
(2-2) case of $h = 0$:
Allocate all the elements of S'_0 in an arbitrary order. □

3.1.1 Allocation S'_h by "local allocation method"

Let G' be the maximal subgraph consisting of the vertex set $S'_h \cup S_{h-1}$ of DAG: the edge set consisting of all the edges of DAG between S'_h and S_{h-1}. We change all the directed edges of G' into undirected and obtain all the connected components $C_1, \cdots, C_m$. We call elements of S'_h and S_{h-1} in C_i parents and children, respectively. We divide these connected components into two parts Π_1 and Π_2 as follows.

$\Pi_1 = \{C_i \mid$ the number of parents of $C_i \leq$ the number of children of $C_i, 1 \leq i \leq m\}$
$\Pi_2 = \{C_i \mid$ the number of parents of $C_i >$ the number of children of $C_i, 1 \leq i \leq m\}$

We allocate the elements of S'_h at time $t, t+1, \cdots, t + \lceil |S'_h|/k \rceil - 1$ as follows.

(1) Allocate all the elements of S'_h belonging to $C_i (1 \leq i \leq m)$ in the order Π_1 and then Π_2.

(2) For any connected component C_i whose parents need at least two processor times for their allocation, we first allocate a parent which has the most children. Then, we allocate a parent which is connected to the element immediately before allocated by length 2 (through one child). Continue this process until all the parents are allocated. Note that as C_i is a connected component, all the parents are chosen in this repeating process). □

Henceforth, we use the auxiliary functions *put*, *ko*, *oya* which are defined as follows.
put$((t_1,\cdots,t_n),X)$: function *put* takes as inputs a list of times $(t_1,\cdots,t_n)$ and an instruction set X and allocate all the elements of X allocatable in the time list $(t_1,\cdots,t_n)$ to the unallocated processors.
In case of $X \subseteq S_i$,

$ko(X) = \{y \in S_{i-1} \mid \exists x \in X.\ x$ is a predecessor (parent) of $y\}$

$oya(X) = \{y \in S_{i+1} \mid \exists x \in X.\ y$ is a predecessor (parent) of $x\}$

And let t and t' be the start and last time of allocation of S'_h, respectively.

3.1.2 Allocation of S'_h and S_{h-1} by 'extended local allocation method'

Make a list L of the elements of S'_h in the decreasing order of the out-degrees, where any the element whose out-degree is more than 5 is regarded as out-degree 5[2]. $l_{h-1} := (l_h + |S_{h-1}|) \bmod k$; $m := k - l_{h-1}$

(I) case of $k > 6l_h$.
We extract $5l_h$ elements from the tail of list L, divide this into 5 blocks with l_h elements. Let these blocks be $B_1, B_2, \cdots, B_5$ in this order. Choose one $B_i (1 \leq i \leq 5)$ satisfying $|ko(B_i)| \leq l_{h-1}$ and $|ko(ko(B_i))| \leq |S_{h-2}| - m$ and choose any m elements from $S_{h-2} - ko(ko(B_i))$ (Let this set be M).
put$(t', B_i \cup oya(M))$; put$((t,\cdots,t'-1), S'_h - B_i)$; put$(t'+1, ko(B_i))$;
put$((t', t'+1), (S_{h-1} - oya(M) - ko(B_i)))$

(II) case of $k \leq 6l_h$.
We extract $2l_h$ elements from the head of list L, divide this into 2 blocks with l_h elements. Let these two blocks be L_1 and L_2.

We extract $k - 2m$ elements from the head of L and let this set be L'[3].
Let $Ni = ko(Li), Pi = ko(Ni)$ $(i = 1, 2)$,
$X = N1 \cap N2$ ($| X |= x$), $Y = S_{h-1} - (N1 \cup N2)$. (Fig.2). Without the loss of generality, we assume that $| N1 | \leq | N2 |$

Select m elements of $P1 \cup P2 \subseteq S_{h-2}$ according to the following priorities 1-4 (Let this set be M).

1. The elements u satisfying that $oya(u) \subseteq X$. (Let this set be $X1$, $|X1| = x_1$).
2. The elements u satisfying that one parent of u is an element of X and the other is an element of $N1 \cup N2 - X$ (Let this set be $X2$, $|X2| = x_2$).
3. The elements u satisfying that $oya(u) \subseteq N1 \cup N2 - X$ (Let this set be B, $|B| = b$). Or the elements u satisfying that one parent of u is an element of X and the other is an elements of Y (Let this set be $X3$, $|X3| = x_3$).
4. $2x_1 + x_2$ elements chosen arbitrarily from the elements of $(P1 \cup P2) - (X1 \cup X2 \cup X3 \cup B)$. (Let this set be C, $|C| = c$). Here, if the number of the elements is less than $2x_1 + x_2$, let C be all the elements.

   In the case where $|X1 \cup \cdots \cup C| < m$, let $M = X1 \cup \cdots \cup C$.

According to the following cases, we apply one of the sub-allocation methods (II-1)-(II-6) described later.

- If $|M| = m$, then apply sub-method (II-1)

[2] We can make this list in linear time by using bucket-sort.
[3] $L' \supseteq L_1 \cup L_2$ holds (ref. Property 3.1(3)).

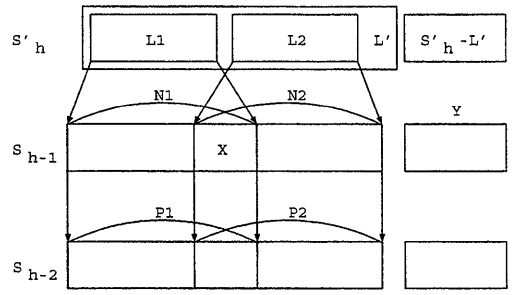

Figure 2: $Li, Ni, Pi (i = 1, 2), Y$

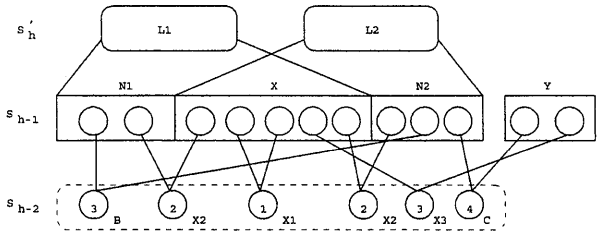

Figure 3: $X1, X2, X3, B, C$

- If $|M| < m$ and $x \geq m$, then apply sub-method (II-2)
- If $|M| < m$ and $x < m$ and
  - if $| N1 | \leq l_{h-1}$ and
    - if $| P1 | \leq | S_{h-2} | - m$, then apply sub-method (II-3)
    - if $| P1 | > | S_{h-2} | - m$, then apply sub-method (II-4)
  - if $| N1 | > l_{h-1}$ and
    - if $|Y| > \lfloor \frac{m}{2} \rfloor$, then apply sub-method (II-5)
    - if $|Y| \leq \lfloor \frac{m}{2} \rfloor$, then apply sub-method (II-6)

□

3.1.3 Sub-allocation methods (II-1)-(II-6)

- (II-1)
  put$(t, L' \cup oya(oya(M)))$;
  put$(t', oya(M))$;
  put$((t, \cdots, t')$,(all the remaining elements of S'_h));
  put$((t', t'+1)$, (all the remaining elements of S_{h-1})) (Fig. 4)
- (II-2)
  Choose $(m - x_1 - x_2 - x_3)$ elements from $S_{h-2} - (X1 \cup X2 \cup X3)$ arbitrarily (let this set be D).
  Let $M := X1 \cup X2 \cup X3 \cup D$. Apply the sub-allocation method (II-1).
- (II-3)
  Choose m elements from $S_{h-2} - ko(N1)$ arbitrarily (let this set be M).
  put$((t, \cdots, t'-1), S'_h - L1)$; put$(t', L1)$;
  put$(t'+1, N1)$; put$(t', oya(M))$;
  put$((t', t'+1)$, (all the remaining elements of S_{h-1}))

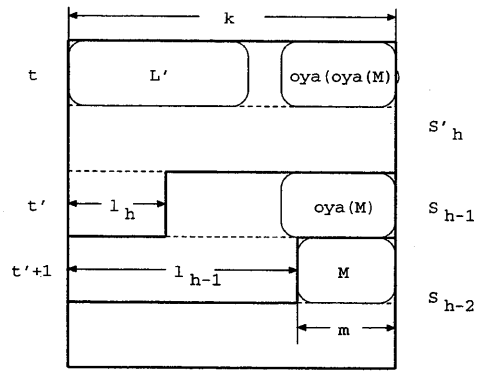

Figure 4: The extended local allocation method

- (II-4)
  Apply (II-3). Where $L1$ and $N1$ in (II-3) are replaced by $L2$ and $N2$, respectively.

- (II-5)
  Let $Z = \{z \in P1 \cup P2 \mid \mid oya(\{z\}) \cap Y \mid = 1\}$.
  Make a list of Y in the decreasing order of the out-degrees in the subgraph whose vertex set is $Y \cup Z$[4] and choose the top $\lfloor \frac{m}{2} \rfloor$ elements of the list. (Let this set be Y').
  Let $Z' = \{z' \in Z \mid \mid oya(\{z'\}) \cap Y' \mid = 1\}$.
  Choose m elements from Z' arbitrarily if possible. If we can choose only the $m-1$ elements(it happens in only the case where m is odd), we add any one element from $X1 \cup X2 \cup B$ to Z' in order to m elements. Let the set of the chosen m elements be M. Apply (II-1).

- (II-6)
  Let $Z = \{z \in P1 \cup P2 \mid |oya(\{z\}) \cap Y| = 1\}$ and choose m elements from Z arbitrarily. Let this set be M. Apply (II-1). □

In this algorithm, given an adjacent list of a DAG, it is obvious that the following problem (1)-(5) can be solved in linear time.

1. The problem of obtaining the set $S_H, S_{H-1}, \cdots, S_0$ (e.g., by depth-first-search).
2. The problem of obtaining the list $oya(\{v\})$ for every vertex v.
3. The problem of obtaining all the connected components for the subgraph of each $S'_h \cup S_{h-1}$ $(0 < h \leq H)$.
4. The problem of obtaining $ko(L_j)$, $ko(ko(L_j))$ and $oya(L_j)$, $1 \leq j \leq l$ of subsets $L_1, L_2, \cdots, L_l$ (where, if $j \neq j'$, $L_j \cap L_{j'} = \emptyset$) of each S_h $(0 < h \leq H)$ where l is some constant.
5. The problem of obtaining the set $X, Y, X1, X2, X3, B, C, M, Z, Y', Z'$ defined in "extended local allocation method" for each $S'_h, S_{h-1}, S_{h-2} (2 \leq h \leq H)$

By these facts, we can show that the time complexity of this algorithm is $O(n)$. (Where, n is the number of all instructions, i.e., the number of nodes of DAG). Note that the number of the edges are at most $2n$ because the DAG satisfies the assumption (1) in Section 2.

[4] we use bucket-sorting regarding all the outdgrees greater than 2 as 2.

3.2 The properties of the allocation algorithm

The "local allocation method" in this algorithm has been proposed in [5] and the following lemma holds.

Lemma 3.1 *([5], p.972, [Lemma 2]). Having allocated the last* $l_h(=|S'_h| mod\ k)$ *elements of* S'_h *at time* t *by "local allocation method", if* $|S'_h| \geq k+1$ *and* $|S_{h-1}| \geq k$, *we can allocate elements of* S_{h-1} *at all the unallocated* $k - l_h$ *processors at time* t. □

Using this lemma we have shown that the "local allocation method" produces an optimal scheduling if $|S_h| \geq 2k-1 (0 \leq h \leq H)$ in [5]. We show that the algorithm given in Section 3.1 produces an optimal scheduling if $|S_h| \geq \frac{3}{2}k (0 \leq h \leq H)$. Henceforth, we assume that $|S_h| \geq \frac{3}{2}k$ $(0 \leq h \leq H)$. And to show some properties of "extended local allocation method" allocating S'_h and S_{h-1}, we also assume the following condition (*0)

$$h \geq 2, l_h > 0, l_h + |S_{h-1}| < 2k \quad \cdots (*0)$$

and let t and t' be the start and last time of the allocation of S'_h, respectively.

Property 3.1

(1) $|S_{h-1}| = k + l_{h-1} - l_h$

(2) $l_{h-1} \geq \frac{1}{2}k + l_h$

(3) $k \geq 2l_h + 2m$

(Proof)

(1) Obvious from the definition of l_{h-1} and (*0).

(2) By (1), $l_h + \frac{3}{2}k \leq k + l_{h-1}$. Therefore, (2) holds.

(3) By $m = k - l_{h-1}$ and (2), $m \leq k - (\frac{1}{2}k + l_h) = \frac{1}{2}k - l_h$. Therefore, (3) holds.

[Notation] For $v \in S_h$, let $out(v)$ be the number of the out-edges from v to S_{h-1}. For a set of instructions L, let $out(L) = \sum_{v \in L} out(v)$.

Lemma 3.2 *Let* $k > 6l_h$ *and* $|S'_h| > k$. *For the allocation (I) of "extended local allocation method", there exists some* $i (1 \leq i \leq 5)$ *such that* $|ko(B_i)| \leq l_{h-1}$ *and* $|ko(ko(B_i))| \leq |S_{h-2}| - m$

(Proof) See Appendix 1.

The following Lemmas are concerned with the allocations (II-1)-(II-6) in the case of $k \leq 6l_h$.

Lemma 3.3 *Let* $k \leq 6l_h$ *and* $|S'_h| > k$. *We assume that the elements of* L' *are allocated at time* t. *Allocating the remaining unallocated elements of* S'_h *at time* $(t, t+1, \cdots, t')$ *in any order, we have* $out(L_{t'}) \leq l_{h-1}$ *for the set* $L_{t'}$ *of the elements allocated at time* t' *satisfying that* $|L_{t'}| = l_h, L_{t'} \subseteq S'_h$.

(Proof) See Appendix 2.

Lemma 3.4 $min(|N1|, |N2|) \leq l_{h-1}$ *or* $|Y| < (m - l_h) + |X|$

(Proof) Assuming that $|Y| \geq (m - l_h) + |X|$ holds, we have

$$\begin{aligned} |N1| + |N2| &= |S_{h-1}| + |X| - |Y| \\ &\leq |S_{h-1}| - (m - l_h) \\ &= 2l_{h-1} \quad \text{(by Property 3.1(1) and } m = k - l_{h-1}) \end{aligned}$$

Therefore, this lemma holds. □

Lemma 3.5 *Let $0 < h \leq H$. In the allocation of $S_H, S_{H-1}, \cdots, S_{h+1}$ in this algorithm, if there exists no idle processor except the last time and S'_h is defined, then $|S'_h| \geq k$ holds.*

(Proof) In the case of $h = H$, by $|S'_H| = |S_H| \geq \frac{3}{2}k$, this lemma holds. Assuming that this lemma holds for h satisfying $h > h'$, we prove that this lemma holds on h' by dividing into the following two cases.

Case(1): $S'_{h'+1}$ is not defined(i.e.,$S'_{h'+2}$ is defined and (a-2) is applied to $S'_{h'+2}$). In this case, by Property 3.1(2), $l_{h'+1} \geq \frac{1}{2}k + l_{h'+2} > \frac{1}{2}k$ holds. Therefore, the number of idle processors at the last time t' of the $S_{h'+1}$'s allocation is less than $\frac{1}{2}k$. Therefore, the number of instructions of $S_{h'}$ allocatable at time t' is also less than $\frac{1}{2}k$. So, by $|S_{h'}| \geq \frac{3}{2}k$ this lemma holds.

Case(2): $S'_{h'+1}$ is defined. In this case, $l_{h'+1} = 0$ or the allocation (a-1) is applied to $S'_{h'+1}$. In the former case, $|S'_{h'}| = |S_{h'}| \geq \frac{3}{2}k$, In the latter case, $l_{h'+1} + |S_{h'}| \geq 2k$.

So, by $|S'_{h'}| = |S_{h'}| - (k - l_{h'+1}) \geq k$, this lemma holds. □

Property 3.2

Let $k \leq 6l_h$. Then,

(1) $2l_h \geq m$

(2) $l_{h-1} \geq 2m$

(3) $k \geq 3m$

(Proof)

(1) By Property 3.1(3), $6l_h \geq k \geq 2l_h + 2m$. Therefore, (1) holds.

(2) By Property 3.1(3) and Property 3.2(1), we have $l_{h-1} = k - m \geq 2l_h + m \geq 2m$.

(3) Obvious from (1) and Property 3.1(3).

Lemma 3.6 *Assume that S'_h is defined and $|S'_h| > k$ where $1 \leq h \leq H$. If the allocation(a-2) is applied to S'_h, then elements of S_{h-1} are allocatable to all the remaining idle processors at the last time the allocation of S'_h terminates and elements of S_{h-2} at the last time the allocation of S_{h-1} terminates. Thus, no idle processors at the both times.*

(Proof)

We show that this lemma holds for all the cases (I) and (II-1)-(II-6) in "extended local allocation method".

(Proof of (I))

By Lemma 3.2, there exists $i (1 \leq i \leq 5)$ such that $|ko(B_i)| \leq l_{h-1}$ and $|ko(ko(B_i))| \leq |S_{h-2}| - m$. Hence, we can choose a set of the m elements $M \subseteq S_{h-2} - ko(ko(B_i))$ satisfying that

$$|B_i \cup oya(M)| \leq l_h + 2m \leq k \quad \text{(by Property 3.1(3))}$$

and

$$|ko(B_i) \cup M| \leq l_{h-1} + m = k$$

Therefore, we can allocate $B_i \cup oya(M)$ at time t' and $ko(B_i) \cup M$ at time $t' + 1$.

(Proof of (II-1)) In order to allocate one elements $q \in S_{h-2}$ at time $t' + 1$, we need to allocate $oya(oya(\{q\})) \subseteq S'_h$ at time t. If q belongs to any of $X1, X2, X3, B, C$, then $|oya(oya(\{q\})) - (L1 \cup L2)|$ (i.e., the number of elements except $L1 \cup L2$ needed to allocate) is given as follows.

$X1$	$X2$	$X3$	B	C
0	1	2	2	3

For the set M chosen in (a-2), let $|M| = m = x'_1+x'_2+x'_3+b'+c'$ where $x'_i \leq x_i (1 \leq i \leq 3), b' \leq b, c' \leq c$ are the numbers of elements chosen from the corresponding sets Xi, B, C. In order to allocate M at time $t'+1$, the number of elements of $oya(oya(M))$ in $S'_h - (L1 \cup L2)$ needed to allocate at t' are at most $0 \cdot x'_1 + x'_2 + 2x'_3 + 2b' + 3c' \leq 2m$ (by $c' \leq 2x'_1 + x'_2$). And by Property 3.1(3),$L' \supseteq L1 \cup L2$. Therefore, put$(t, L' \cup oya(oya(M)))$ can be completed.

Moreover, by Property 3.1(3) and $|oya(M)| \leq 2m$, put$(t', oya(M))$ can be completed and M is allocatable at time $t'+1$. Even if we choose any l_h elements from $S'_h - (L' \cup oya(oya(M)))$ at time t', Lemma 3.3 ensures that the $(k - l_h)$ elements of S_{h-1} including $oya(M)$ are allocatable at time t'. It follows that there exist no idle processors, as claimed.

(Proof of (II-2))

By $|X1 \cup X2 \cup X3| < m$, we have $x_1 + x_2 + x_3 < m$. Let the number of the elements of D be $d(= m-(x_1+x_2+x_3))$. By $|X| \leq 2x_1+x_2+x_3$ and $m \leq |X|$, we have $d \leq x_1, |M| = m = x_1+x_2+x_3+d$ and the number of the elements of $oya(oya(M))$ in $S'_h-(L1 \cup L2)$ is at most $0 \cdot x_1+x_2+2x_3+4d \leq 2m$ (by $d \leq x_1$) by a similar proof to that of Case (II-1). Therefore, by the same proof as the latter half that of (II-1), this lemma holds.

(Proof of (II-3))

By $| N1 | \leq l_{h-1}$, we can allocate at least $| S_{h-1} | - | N1 | (\geq k - l_h$ (by Property 3.1(1)) elements of S_{h-1} at time t', and m elements of $S_{h-2} - P1$ at time $t'+1$, since $|P1| \leq |S_{h-2}| - m$.

Therefore there exist no idle processors, as claimed.

(Proof of (II-4))

In this case, we first show that $| N2 | \leq l_{h-1}$. To the contrary, we assume that $| N2 | > l_{h-1}$. Then, we will have a contradiction.

Since $|N2| > 2m$ (by Property 3.2(2)) and $|oya(X1 \cup X2 \cup X3 \cup B \cup C) \cap N2| \leq 2x_1+2x_2+x_3+2b+c < 2m$ (by $|M| < m$), the children of some element of $N2 - X$ have not been chosen as the elements of C. Therefore, $c = 2x_1 + x_2$ holds by the definition of C, so that $x_1+x_2+x_3+b+c = 3x_1+2x_2+x_3+b < m$. And by $x \leq 2x_1 + x_2 + x_3$, we have

$$x + x_1 + x_2 + b < m \qquad (*1)$$

We define subset A', B', C' of S_{h-2} as follows.

$$\begin{aligned} A' &= \{v \in P1 \mid oya(\{v\}) \cap (N2 - X) \neq \emptyset\} \\ B' &= \{v \in P2 - P1 \mid oya(\{v\}) \subseteq (N2 - X)\} \\ C' &= (P2 - P1) - B' \end{aligned}$$

Let the numbers of the elements of A', B', C' be a', b', c', respectively. By the definition of $X2$ and B,

$$a' + b' \leq x_2 + b < m - x - x_1 \quad \text{(by (*1))}$$

By $|N2| \leq x + a' + 2b' + c'$,

$$2m < (x + a' + b') + b' + c' < (m - x_1) + b' + c'$$

Therefore

$$m < b' + c' = |P2 - P1| \quad \cdots \quad (*2)$$

But, by $|P1| > |S_{h-2}| - m$, $|P2 - P1| < m$ holds. This is a contradiction to (*2). Thus, $|N2| \leq l_{h-1}$, as claimed.

Next, we show that $|P2| \leq |S_{h-2}| - m$ holds. To the contrary, we assume that $|P2| > |S_{h-2}| - m$. By $|P1 \cap P2| \leq x_1 + x_2 + x_3 + b < m$, we have

$$\begin{aligned} |P1 \cup P2| &= |P1| + |P2| - |P1 \cap P2| \\ &> 2|S_{h-2}| - 3m \\ &\geq |S_{h-2}| + 3l_h \quad \text{(by } |S_{h-2}| \geq \frac{3}{2}k, \text{Property 3.1(3))} \end{aligned}$$

This is a contradiction to $|P1 \cup P2| \leq |S_{h-2}|$.

Therefore, $|\ N2\ | \leq l_{h-1}$ hold, so that $|\ P2\ | \leq |\ S_{h-2}\ | - m$, there exist no idle processors by a similar argument to that of the case (II-3) . (That is, in **(Proof of (II-3))**, we simply replace $L1$ and $N1$ by $L2$ and $N2$, respectively).

(Proof of (II-5))

If $m = 2n$ or $2n+1$ for some positive integer n, then we show that $|\ Z'\ | \geq 2n$ holds.

To the contrary, we assume that $|\ Z'\ | < 2n$. Let G be the DAG whose vertex set is $Y \cup Z$. Then, Y' has at most $2n-1$ edges(i.e., $out(Y') \leq 2n-1$). As we have chosen the elements of Y' in the decreasing order of the out-degrees from Y, every element of $Y - Y'$ has at most one edge in G.

Therefore, we have

$$\begin{aligned} |Z| &\leq 2n-1+|Y|-\lfloor\frac{m}{2}\rfloor = n-1+|Y| \\ &< \frac{1}{2}m+|Y| \\ &< \frac{3}{2}m-l_h+x \quad \text{(by Lemma 3.4)} \cdots (*3) \end{aligned}$$

$$|P1 \cup P2| = x_1+x_2+b+|Z| \quad \cdots \quad (*4)$$

$$|P1 \cup P2| \geq (|N1|+|N2|-x)-(x_1+x_2+b) \quad \cdots \quad (*5)$$

By (*4) and (*5), we have

$$\begin{aligned} x_1 + x_2 + b &\geq |N1|+|N2|-x-|Z|-(x_1+x_2+b) \\ &> 2l_{h-1}-x-(\frac{3}{2}m-l_h+x)-(x_1+x_2+b) \quad (\text{by } |N1|>l_{h-1}, |N2|>l_{h-1}, (*3)) \\ &> 2l_{h-1}-\frac{3}{2}m+l_h-2m \quad (\text{by } x<m, (*1)) \\ &\geq 4m-\frac{3}{2}m+\frac{1}{2}m-2m \quad \text{(by Properties 3.2(1) and (2))} \\ &= m \end{aligned}$$

Therefore, $x_1+x_2+b > m$. This is a contradiction to (*1).

Thus, $|\ Z'\ | \geq 2n$ holds, as claimed.

(1) Case of $m = 2n$:
In this case, $|\ Z'\ | \geq 2n = m$ holds. For each element of Z', one of the parents is in Y' ($|\ Y'\ | = \lfloor\frac{m}{2}\rfloor$) and the other is in $N1 \cup N2$, so that in order to allocate M at time $t'+1$, the number of the elements of $S'_h - L'$ belonging to $oya(oya(M))$ needed to allocate at time t is at most $2m$.

(2) Case of $m = 2n+1$:
In this case, $|\ Z'\ | \geq 2n = m-1$ holds. In the case of $|\ Z'\ | \geq m$, $|oya(oya(M)) - L'| \leq 2m$ holds by the same argument as that of (1). In the case of $|\ Z'\ | = m-1$, $|\ oya(oya(Z')) - L'\ | \leq 4n$ holds and for the element $d \in (X1 \cup X2 \cup B)$, $|\ oya(oya(\{d\})) - L'\ | \leq 2$ holds, so that $|\ oya(oya(M)) - L'\ | \leq 4n+2 = 2m$ holds. And the existence of $d \in (X1 \cup X2 \cup B)$ can be proved easily.

Therefore in the both cases (1) and (2), put$(t, L' \cup oya(oya(M)))$ can be completed.
And, by Property 3.1(3), so put$(t', oya(M))$. Thus, M can be allocated at time $t'+1$.
Moreover, from Lemma 3.3 there exist no idle processors at time t' and $t'+1$, as claimed.

(Proof of (II-6))

In this case, note that

$$\begin{aligned} |N1 \cup N2| &= |S_{h-1}| - |Y| \\ &\geq \frac{3}{2}k - \lfloor\frac{m}{2}\rfloor \\ &\geq 4m \quad \text{(by Property 3.2(3))} \end{aligned}$$

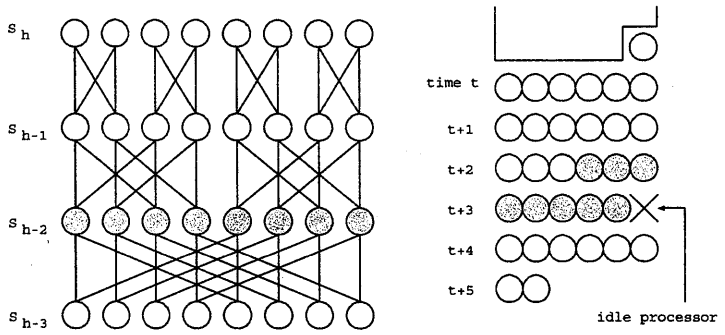

Figure 5: Example of a DAG with non-allocated processor time units in any allocation ($k = 6$).

Hence, $|P1 \cup P2| \geq 2m$. By this and (*4), we have

$$\begin{aligned} |Z| &> 2m - (x_1 + x_2 + b) \\ &> m \quad (\text{by } (*1)) \end{aligned}$$

Therefore, $|Z| > m$. By a similar argument to that of the case (II-5)-(1), $|oya(oya(M)) - L'| \leq 2m$ holds, so that there exist no idle processors. □

theorem 3.1 *For all* $h(0 \leq h \leq H)$, *if* $|\ S_h\ | \ \geq \frac{3}{2}k$, *this algorithm gives an optimal scheduling.*

(Proof) By showing that for any h $(0 < h \leq H)$, there exist no idle processors from the next of the start time of S_h's allocation (where if $l_{h+1} = 0$, from the start time) to the last time of this allocation, we prove the optimality.

In the case of $h = H$, $S'_h = S_h$ holds, so that $|S'_h| \geq \frac{3}{2}k$. If $l_h = 0$, then obviously there exist no idle processors. If $l_h \neq 0$ and the allocation (a-1) is applied, then by Lemma 3.1 there exist no idle processors from the start time of S'_h's allocation to the last time. If the allocation (a-2) is applied, then no idle processors by Lemma 3.6 .

In the case of $h < H$, we assume that there exist no idle processors at every time of the allocation of $S_H, S_{H-1}, \cdots, S_{h+1}$ by the induction hypothesis. If S'_h is defined, then by Lemma 3.5 $|S'_h| \geq k$ holds. Therefore by the same proof to that of the case of $h = H$, there exist no idle processors. If S'_h is not defined, then S'_{h+1} must be defined and the allocation (a-2) must be applied. Therefore, by Lemma 3.6 there exist no idle processors at every time of the allocation of S_h. □

When the condition of [Theorem 3.1] is not satisfied, we show an example of a DAG impossible to give a full scheduling (i.e., a scheduling without idle processor time units) in Fig. 5. In this example, $k = 6, |S_h| = 8 < \frac{3}{2} \cdot 6$ does not satisfy the condition.

Also, the example of Fig.2 in [5] is an example of the case of $k = 3$ which has no full scheduling (in this example $|S_h| = 4 < \frac{3}{2} \cdot 3$).

4 Scheduling of DAG in which every node has at most 3 parents

4.1 Allocation algorithm

In the following allocation algorithm, let G_h be the subgraph of a given DAG whose vertices consist of $S_h \cup S_{h-1} (0 < h \leq H)$.

Allocation algorithm A_1

(0) Initialization: $l_{H+1} := 0$; $t := 1$

(1) Allocate the elements of S_h in the order of $h = H, H-1, \ldots, 0$.

(1-1)
- Case of $l_{h+1} = 0$: $S'_h := S_h$; $U := \emptyset$.
- Case of $l_{h+1} \neq 0$: Choose elements of S_h allocatable to $k - l_{h+1}$ idle processors at time t as many as possible and allocate them at time t. Let U be the set of the allocated elements. $S'_h := S_h - U$; $t := t + 1$

(1-2) If $S'_h \neq \emptyset$, then allocate the elements of S'_h on and/or after time t as follows.

(a) Case of $h \neq 0$: $l_h := |S'_h| \mod k$; $m := 2k - 1$.

If $|S_{h-1}| > m$, then let $\tilde{S}_{h-1}$ be a subset of S_{h-1} satisfying the following condition: $|\tilde{S}_{h-1}| = m \wedge (\forall u \in U \exists s \in \tilde{S}_{h-1}.\ u \in oya(\{s\})) \wedge (|S_h| \geq m \Rightarrow |oya(\tilde{S}_{h-1})| \geq m)$. (It is obvious that we can choose such $\tilde{S}_{h-1}$.)

If $|S_{h-1}| \leq m$, then $\tilde{S}_{h-1} := S_{h-1}$. Let G'_h be the subgraph of G_h whose vertex set is $S'_h \cup \tilde{S}_{h-1}$.

- If $l_h = 0$ or $|S'_h| \leq k$, then allocate all the elements of S'_h in an arbitrary order.
- If $0 < l_h < \frac{1}{2}k$ and $|S'_h| > k$, then allocate all the elements of S'_h in the decreasing order of the out-degrees in G'_h. Here, we do bucket-sorting them by regarding all the out-degrees greater than 4 as 4.
- If $l_h \geq \frac{1}{2}k$ and $|S'_h| > k$, then let x be an element of S'_h with the smallest out-degree in G'_h. $S''_h := S'_h - \{x\}$. Choose $l_h - 1$ elements from S''_h arbitrarily and let V be their set. $T_{h-1} := \{s \in \tilde{S}_{h-1} |$ at least one of the parents of s belongs to $U \cup V\}$.

(a-1) Case of $|T_{h-1}| \geq k$: First, allocate the elements of V at time t. Next, for the subgraph G''_h of G'_h whose vertex set is $T_{h-1} \cup \{s \in (S'_h - V) \mid s$ is a parent of some element of $T_{h-1}\}$, allocate the parents of G''_h using "local allocation method" (ref. section 3.1). Finally, allocate all the unallocated elements of $S'_h - V$ in any order (See Fig. 6).

(a-2) Case of $|T_{h-1}| < k$: First, allocate the elements of $S''_h - V$. Next, allocate the elements of V and x. $t := t + \lfloor |S'_h|/k \rfloor$.

(b) If $h = 0$, then allocate the elements of S'_0 in any order. □

The time complexity of this algorithm is $O(n)$. Here, n is the total number of the instructions, i.e., the number of the vertices in DAG. Note that as DAG satisfies the condition (1) in Section 2, the total number of the edges is at most $3n$.

4.2 Properties of allocation algorithm A_1

For critical path length h if any of the following conditions ①, ②, ③ hold, then we say that our allocation algorithm enters a **special allocation stage** .

① $k = 4$, $l_h = 2$ ② $k = 3$, $l_h = 2$ ③ $k = 2$, $l_h = 1$

theorem 4.1 *If* $|S_h| \geq 2k - 1$ *for all* $h (0 \leq h \leq H)$, *and allocation algorithm* A_1 *never enter special allocation stages during the allocation, then* A_1 *produces an optimal scheduling.*

(Proof) We show the optimality by showing that Algorithm A_1 does not make idle processors except the last time of the S_0's allocation. That is, we prove the following proposition by induction.

Proposition Algorithm A_1 allocates $S_H, S_{H-1}, \cdots, S_h$ without making idle processors except the last time of the S_h's allocation for all $h (0 \leq h \leq H)$. □

(Proof of Proposition)

1. Basis: the case of $h = H$. Obvious.

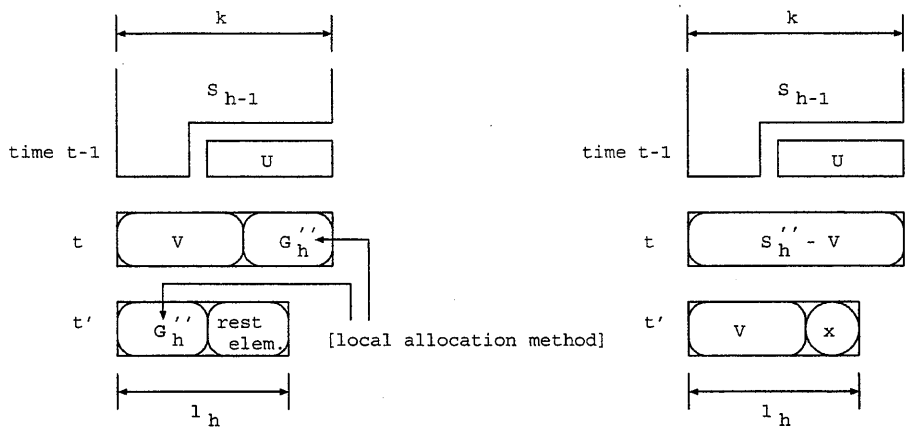

Figure 6: The method of allocation in the case of $l_h \geq \frac{1}{2}k$.
(Left fig. : $\mid T_{h-1} \mid \geq k$ Right fig. : $\mid T_{h-1} \mid < k$)

2. Induction step: By the induction hypothesis, we assume that for $h(0 < h \leq H)$, A_1 can allocate upto S_h without making idle processors. Consider the allocation of S_{h-1}.

   - If $l_h = 0$, then the proof is obvious.
   - If $l_h \neq 0$, then obviously $|S'_h| > k$. Let $|S'_h| = nk + l_h$ for some $n \geq 1$. Let t be the start time of the allocation of S'_h and $t'(= t+n)$ be the last time. We divide the proof into the following three cases.

   Case 1: $l_h < \frac{1}{2}k$
   Case 2: $l_h \geq \frac{1}{2}k$ and $\mid T_{h-1} \mid \ \geq k$
   Case 3: $l_h \geq \frac{1}{2}k$ and $\mid T_{h-1} \mid \ < k$

[Proof of Case 1]

As the number of parents of each instruction is at most 3,

$$out(S_h) \leq 6k - 3 \qquad \cdots \qquad (*)$$

Here, $out(S_h)$ is the total number of edges from S_h to $\tilde{S}_{h-1}$. If the number of instructions in $\tilde{S}_{h-1}$, each of which has at least one of the l_h elements in S'_h allocated at time t' as its parent, is less than or equal to $k + l_h - 1$, then at least $\mid \tilde{S}_{h-1} \mid -(k + l_h - 1) = k - l_h$ children of $\tilde{S}_{h-1}$ can be allocated at time t'. Therefore, in this case, there exist no idle processors at time t'.

So, consider the other case: the number of the children of $\tilde{S}_{h-1}$ dominated by the l_h elements at time t' is greater than or equal to $k + l_h$. In this case, $(k + l_h)/l_h > 3$ holds by $l_h < \frac{1}{2}k$. Therefore, every element in S_h allocated at time $t, \cdots, t' - 1$ has 4 children or more. Furthermore, the total number of out-edges of the $k - l_{h+1}$ elements of S_h allocated at time $t - 1$ is at least $k - l_{h+1}$, so that

$$\begin{aligned} out(S_h) &\geq (k - l_{h+1}) + 4nk + (k + l_h) \\ &\geq (2k - 1) - l_h + 3nk + (k + l_h) \quad (\text{by } (k - l_{h+1}) + nk \geq (2k - 1) - l_h) \\ &\geq 6k - 1 \quad (\text{by } n \geq 1) \end{aligned}$$

This is a contradiction to (*). Thus, the number of the children dominated by the l_h elements at time t' cannot be greater than or equal to $k + l_h$.

Therefore, we can allocate the elements of $\tilde{S}_{h-1}$ at time t' without making idle processors.

[Proof of Case 2]

Let $oya(G''_h)$ and $ko(G''_h)$ be the sets of the parents and the children in G''_h, respectively. By the construction of G''_h in Algorithm A_1, it is guaranteed for every element of $ko(G''_h)$, that the number of the parents in G''_h is at most 2. Hence, we can show this proposition by a similar proof to that of Lemma 2 in [5] p.972 (See Appendix 3.)

[Proof of Case 3] For x allocated at time t', $out(x) \leq 3$ holds. To the contrary, if we assume that $out(x) \geq 4$, then, since x is an element with the least out-degree in S'_h, every one of the other $k + l_h - 1$ or more elements in S'_h has 4 or more edges. Therefore the total number of the out-edges of the elements in S_h is

$$out(S_h) \geq out(S'_h) \geq 4(k + l_h) \quad \geq \quad 6k \quad (\text{by } l_h \geq \frac{1}{2}k)$$

This is a contradiction to (*).

Furthermore, by $|T_{h-1}| < k$, the number of the elements in $\tilde{S}_{h-1}$ dominated by V (i.e., each of which has some element of V as one of its parents) is at most $k-1$, so that the number of elements in $\tilde{S}_{h-1}$ dominated by $V \cup \{x\}$ is at most $k-1+3 = k+2$. That is, the number of the elements in $\tilde{S}_{h-1}$ allocatable at time t' is at most $| \tilde{S}_{h-1} | -(k+2) = k-3$, so that if $k - 3 \geq k - l_h$, that is, if $l_h \geq 3$ holds, then there exist no idle processors at time t'.

Thus, the remaining case is only that of $l_h = 2$ or 1 . Note that this **[Case 3]** satisfies $l_h \geq \frac{1}{2}k$. Hence, only the following three cases remain.

① $k = 4$, $l_h = 2$, ② $k = 3$, $l_h = 2$, ③ $k = 2$, $l_h = 1$

Note that these three cases are called the special allocation stages and we have assumed that algorithm A_1 never enter these stages. Hence, since these cases do not occur, we can allocate elements in S'_{h-1} to all the idle processors at time t'.

Therefore, for $h-1$, this proposition holds. □

4.3 Scheduling for special allocation stages

In this subsection, we describe how to schedule in the special allocation stages and extend [Theorem 4.1].

Case of ① $k = 4$, $l_h = 2$

- Case of $| S'_h | \geq 2k + l_h (= 10)$:
  Choose any two elements in S_{h-1}. Let W be the set of the elements. First, allocate all the parents of W from time t in any order, and then, allocate the remaining elements in S'_h to time t'.

  Because the total number of the parents of W is at most 6, the allocation of these elements is completed by time $t' - 1$. Therefore, W can be allocated at time t', so that there exist no idle processors.

- Case of $| S'_h | < 2k + l_h$:
  In this case, $| S'_h | = k + l_h = 6$, so that $| U | \geq (2k-1) - | S'_h | \geq 1$. Choose one element in S_{h-1} dominated by U arbitrarily and let this be x.

  (i) If x has 1 or 0 parent in S'_h, then choose one element, say y, in $S_{h-1} - \{x\}$ arbitrarily and let $W = \{x, y\}$.

  (ii) Otherwise (i.e., if x has two parents in S'_h), if there exists $y \in S_{h-1} - \{x\}$ such that y and x have at least one common parent in S'_h, then let $W = \{x, y\}$. Otherwise, choose two elements from $S_{h-1} - \{x\}$ arbitrarily and let this set be W.

  In either case, note that the number of the parents of W belonging to S'_h is at most 4, since $|S'_h| = 6$. Thus, allocate the parents of W at time t and then the remaining elements of S'_h to $t' = (t+1)$. Since W can be allocated at time t', there exist no idle processors.

Case of ② $k = 3$, $l_h = 2$ Because the number of parents of each element is at most 3, we can allocate all the parents of some element in S_{h-1} at time $t' - 1$, so that the element of S_{h-1} can be allocated at time t'. Thus, there exist no idle processors.

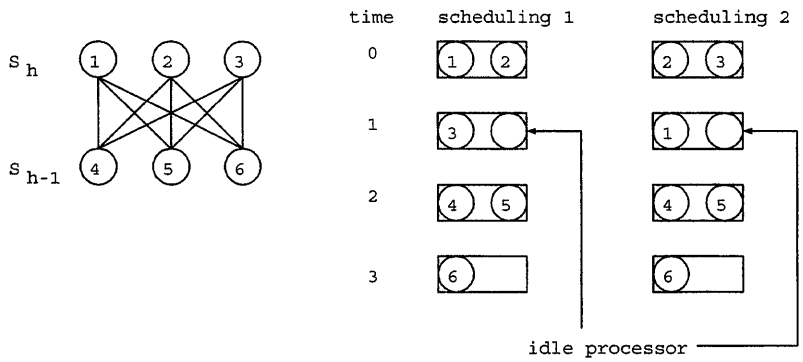

Figure 7: Example of a DAG with non-allocated processor time units in any allocation ($k = 2$).

Case of ③ $k = 2$, $l_h = 1$ In this case, there exists a DAG in which no elements in S_{h-1} are allocated at time t' (See Fig. 7), that is, there exists an idle processor at time t'.

However, if we change the condition $|S_h| \geq 2k - 1$ to $|S_h| \geq 2k = 4$, then obviously there exists elements allocatable at time t'.

By the above arguments, we obtain the following corollary of Theorem 4.1.

Corollary 4.1 *Let A_2 be the algorithm augmenting the allocation algorithm A_1 by the above algorithm in the case of ① and ②. If $k \geq 3$ and for all h $(0 \leq h \leq H)$ $\mid S_h \mid \geq 2k - 1$, then algorithm A_2 gives an optimal scheduling.*

Note that if $k = 2$, then we need to consider only the case of ③ or $l_h = 0$, so that if $|S_h| \geq 2k$, it is obvious that there exists a linear time optimal scheduling algorithm.

5 Conclusion

In this paper, we have first assumed that the number of parents of each instruction in a DAG is at most 2, and proposed a heuristic algorithm of scheduling in linear time under this assumption and shown that it produces an optimal scheduling if the number of vertices of S_h (the set of vertices with critical path length h) satisfies some restricted conditions. Next, we have assumed that the number of parents of each instruction in a DAG is at most 3, and proposed a heuristic algorithm of scheduling in linear time under this assumption, which has a similar property to that of the former algorithm.

References

[1] H. Komatsu, T. Shinriki, A. Koseki and Y. Fukazawa, "A Register Allocation Technique for Instruction-Level Parallel Architecture", Trans. of IPSJ, Vol.36, No .12, pp. 2819–2830, Dec. 1995.

[2] J.D. Ullman, "NP-Complete Scheduling Problems", J.Comput. Syst. Sci. 10, pp. 384–393, 1975.

[3] A.V. Aho, D. Ullman and R. Sethi, "Compilers Principles, Techniques, and Tools", Addison-Wesley, Reading, MA, 1986.

[4] M.R. Garey and D.S. Johnson, "Computers and Intractability: A Guide to the Theory of NP-Completeness", Freeman, San Francisco, 1991.

[5] Y. Matsubara, T. Hattori, M. Oyamaguchi and Y. Ohta, "Code Scheduling for Instruction-Level Parallel Processor", Trans. of IEICE, Vol.J80-D-I, No.12, pp. 971–974, 1997.

[6] E.G. Coffman and R.L. Graham, "Optimal scheduling for two-processor systems", Acta. Inf. 1, pp. 200–213, 1972.

[7] M. Fujii, T. Kasami and N. Ninomiya, "Optimal sequence of two equivalent processors", SIAM J. Appl. Math. 17, pp. 784–789, 1969.

A Proof of Lemma 3.2

(Proof) By $|S'_h| \geq k + l_h > 7l_h$, let $A = S'_h - \cup_{i=1}^{5} B_i$, so $|A| > 7l_h - 5l_h = 2l_h$. We first show that $out(B_1) \leq l_{h-1}$. To the contrary, we assume that $out(B_1) > l_{h-1}$. Then,

$$\begin{aligned} out(B_1)/l_h &> (l_{h-1}/l_h) \\ &\geq (\tfrac{1}{2}k + l_h)/l_h \quad \text{(by Property 3.1(2))} \\ &> 4 \quad \text{(by } k > 6l_h\text{)} \end{aligned}$$

Therefore, by $out(B_1) > 4l_h$, the out-degree of some element in B_1 is 5 or more, so that $out(A) \geq 5|A| \geq 5(k-4l_h)$. The out-degree of each element in $S_h - (A \cup B_1)$ is 1 or more, so we get the following inequality.

$$\begin{aligned} out(S_h) &= out(A) + out(B_1) + out(S_h - (A \cup B_1)) \\ &> 5|A| + l_{h-1} + |S_h| - (|A| + l_h) \\ &= 4|A| + l_{h-1} - l_h + |S_h| \\ &\geq 4(k - 4l_h) + l_{h-1} - l_h + \frac{3}{2}k \quad \text{(by } |S_h| \geq \frac{3}{2}k\text{)} \\ &= (5 + \frac{1}{2})k + l_{h-1} - 17l_h \\ &\geq 2(k + l_{h-1} - l_h) + (k - l_{h-1}) + 5(\frac{1}{2}k - 3l_h) \\ &> 2|S_{h-1}| \quad \text{(by Property 3.1(1), } k > l_{h-1}, k > 6l_h\text{)} \end{aligned}$$

This contradicts to the total number of the existing edges. Therefore, $out(B_1) \leq l_{h-1}$. By the same way, for every $i(2 \leq i \leq 5)$, $out(B_i) \leq l_{h-1}$ holds.

Next, for some $i(1 \leq i \leq 5)$, we show that $|ko(ko(B_i))| \leq |S_{h-2}| - m$. To the contrary, assuming that for every $i(1 \leq i \leq 5)$, $|ko(ko(B_i))| > |S_{h-2}| - m$, we show a contradiction. Let $B = \cup_{i=1}^{5} B_i$. Then, the following equality holds.

$out(S_h) = out(S_h - B) + \sum_{i=1}^{5} out(B_i)$

For each out-edge going out of $(S_h - B)$, the out-degree of the adjacent vertex in S_{h-1} is 1 or more and for all the out-edges going out of $B_i(1 \leq i \leq 5)$, the sum of the out-degrees of the adjacent vertices in S_{h-1} is $|ko(ko(B_i))|$ or more for each i. Because the number of the parents of each vertex in S_{h-1} is at most 2, $out(S_h - B) + \sum_{i=1}^{5} |ko(ko(B_i))|$ is limited to the double of the total number of the edges from S_{h-1} to S_{h-2}. That is, the following inequality holds.

$$out(S_h - B) + \sum_{i=1}^{5} |ko(ko(B_i))| \leq 4|S_{h-2}| \cdots (**)$$

However,

$$\begin{aligned} \text{the left hand side of } (**) &\geq (\frac{3}{2}k - 5l_h) + 5(|S_{h-2}| - m) \\ &\geq 4|S_{h-2}| + 3k - 5l_h - 5m \quad \text{(by } |S_{h-2}| \geq \frac{3}{2}k\text{)} \\ &= 4|S_{h-2}| - 2k + 5l_{h-1} - 5l_h \quad \text{(by } m = k - l_{h-1}\text{)} \\ &\geq 4|S_{h-2}| + \frac{1}{2}k \quad \text{(by Property 3.1(2))} \end{aligned}$$

This contradicts to (**). Therefore, this lemma holds. □

B Proof of Lemma 3.3

Let $L3$ be the set of the first l_h elements chosen from $S'_h - L'$ in the decreasing order of the out-degree. We first show that $out(L3) \leq l_{h-1}$. To the contrary, assuming $out(L3) > l_{h-1}$, we show a contradiction.

- Case of $m \geq l_h$: In this case, $k \geq 4l_h$ holds (by Property 3.1(3)).

$$\begin{aligned} out(L3)/l_h &> l_{h-1}/l_h \\ &\geq (\frac{1}{2}k + l_h)/l_h \quad \text{(by Property 3.1(2))} \\ &\geq 3 (\text{by } k \geq 4l_h) \end{aligned}$$

  Therefore, the out-degree of some element of $L3$ is greater than or equal to 4, But

$$\begin{aligned} out(S_h) &= out(L') + out(L3) + out(S_h - L' - L3) \\ &> 4(k-2m) + l_{h-1} + (|S_h| - (k-2m) - l_h) \\ &\geq (4+\frac{1}{2})k - 6m + l_{h-1} - l_h \quad (\text{by } |S_h| \geq \frac{3}{2}k) \\ &= (4+\frac{1}{2})k - 6(k - l_{h-1}) + l_{h-1} - l_h \\ &= 7l_{h-1} - \frac{3}{2}k - l_h \\ &\geq 2l_{h-1} + 5(\frac{1}{2}k + l_h) - \frac{3}{2}k - l_h \quad \text{(by Property 3.1(2))} \\ &= 2l_{h-1} + k + 4l_h \\ &= (2l_{h-1} + 2k - 2l_h) + (6l_h - k) \\ &\geq 2|S_{h-1}| \quad (\text{by } 6l_h \geq k, \text{ Property 3.1(1)}) \end{aligned}$$

  Therefore, $out(S_h) > 2|S_{h-1}|$. This is a contradiction.

- Case of $m < l_h$: In this case, $k > 4m$ (by Property 3.1(3)).

$$\begin{aligned} out(L3)/l_h &> l_{h-1}/l_h \\ &\geq \frac{1}{2} \cdot k/l_h + 1 \quad \text{(by Property 3.1(2))} \\ &\geq 2 \quad \text{(by Property 3.1(3))} \end{aligned}$$

  Therefore, the out-degree of some element of $L3$ is greater than or equal to 3, so

$$\begin{aligned} out(S_h) &> 3(k-2m) + l_{h-1} + (|S_h| - (k-2m) - l_h) \\ &\geq (3+\frac{1}{2})k + l_{h-1} - 4m - l_h \\ &= (2+\frac{1}{2})k + 2l_{h-1} - 3m - l_h (\text{by } k = m + l_{h-1}) \\ &= (2k + 2l_{h-1} - 2l_h) + (\frac{1}{2}k + l_h - 3m) \\ &> 2|S_{h-1}| \quad (\text{by } l_h > m, k > 4m) \end{aligned}$$

  This is a contradiction. Hence, we have $out(L3) \leq l_{h-1}$. By a similar argument, $out(L3) < 5l_h$ holds[5]. Therefore, $out(L_{t'}) \leq out(L3)$ holds, so this lemma holds. □

[5]We do bucket-sort under the assumption that the out-degree greater than 5 is regarded as 5, so that in order to show $out(L_{t'}) \leq out(L3)$ we need to guarantee that there exists at least one element in $L3$ whose out-degree is lesser than or equal to 4 by this inequality.

C Proof of Theorem 4.1 [Case 2]

If all the elements of $oya(G''_h)$ are allocated before or at time $t'-1$, then by $|\ T_{h-1}\ | \geq k$, we can obviously allocate $k - l_h$ elements in $\tilde{S}_{h-1}$ at time t'. So, we assume that at least one elements in $oya(G''_h)$ is allocated at time t'. Then, we can show this proposition by a similar proof to that of [5] using the following lemma in [5].

Lemma B.1([5],Lemma 1) For DAG whose vertex set is $D_h \cup D_{h-1}$ and whose edge set is $(\subseteq D_h \times D_{h-1})$, we assume the number of the parents $(\in D_h)$ of each child $(\in D_{h-1})$ is at most 2. Allocating the parents by 'local allocation method', for any connected component C of DAG, if $m(>0)$ parents in C are allocated at any time, at least $m-1$ children in C can be allocated at the next time. □

Let C_n be the connected component of G''_h whose parents are allocated at both times $t'-1$ and t'. Let p_n be the number of the parents in C_n and q_n be the number of the children. And, let p'_n and $p''_n (= p_n - p'_n)$ be the number of parents in C_n allocated at time $t'-1$ and t', respectively.

1. Case of $p_n \leq q_n$:
   By Lemma B.1, at least $p'_n - 1$ of the q_n children are allocatable at time t'.

   Since for any connected component C_i allocated before C_n, $p_i \leq q_i$ holds, the number of elements in $\tilde{S}_{h-1}$ allocatable at time t' is at least (the number of the elements of $oya(G''_h)$ allocated at time $t'-1$) $-1 = (k - l_h + 1) - 1 = k - l_h$.

2. Case of $p_n > q_n$:
   By Lemma B.1, at least $p'_n - 1$ of the q_n children are allocatable at time t'. So, the number of the children of C_n which cannot be allocated at time t' is at most

   $$q_n - (p'_n - 1) < p_n - (p'_n - 1) = p''_n + 1$$

   That is, the number of the children of C_n which cannot be allocated at time t' is at most p''_n(i.e., the number of the parents of C_n allocated at time t').

   And for each connected components C_i allocated after C_n was allocated, $p_i > q_i$ holds, so that the number of the children of C_i which cannot be allocated at time t' is at most p_i. Therefore, the number of the elements of T_{h-1}, each of which has as its parents one of the l_h elements of $oya(G''_h)$ allocated at t', is at most l_h (i.e., the number of $oya(G''_h)$ allocated at t'). Therefore, there exist at least $k - l_h$ elements of $\tilde{S}_{h-1}$ which can be allocated at t'.

By (1) and (2), we can allocate $k - l_h$ elements of $\tilde{S}_{h-1}$ to $k - l_h$ processors at t'.
Hence, there exists no idle processor at t'. □