Performance Evaluation of a Two-Processor Scheduling Method for Acyclic SWITCH-less Program Nets

Qi-Wei GE†, Member, Chen LI††, Nonmember, and Mitsuru NAKATA†, Member

SUMMARY This paper investigates the usefulness of a new priority list for two-processor scheduling problem of program nets. Firstly, we discuss the weakness of a previously proposed priority list and then introduce a new priority list. Through simulation experiment we show that the new priority list is better than the previous one and can generate the same length of schedules as GA scheduling, which implies the new priority list can generate approximately optimal schedules.

**key words:** multiprocessor scheduling, program net, dynamic priority list, optimal schedule, GA scheduling

1. Introduction

Multiprocessor scheduling for program net is generally, given with processors and a program net, to determine the optimal schedule, GA scheduling. This problem is generally intractable and has been known as NP-hard problem even for a simple case. Through simulation experiment we show that the new priority list can generate approximately optimal schedules.

A program net is a graph representation of programs, consisting of AND-node, OR-node and SWITCH-node, which represent arithmetic/logic, data merge and context switch operations respectively [3], [4]. For acyclic SWITCH-less program nets, optimal list scheduling and GA scheduling methods have been proposed [5]. An optimal two-processor scheduling for acyclic SWITCH-less program nets has also been proposed, in which each AND-node possesses two input edges [6]. Further, an extended hybrid priority list has been proposed in order to adapt to the nets whose AND-nodes possess two input edges [7].

In this paper, we deal with nonpreemptive two-processor scheduling for acyclic SWITCH-less program nets with unity node firing time and with at most two input edges for each AND-node. Firstly, we discuss the weakness of a previously proposed list [7] and then introduce a new list. Through simulation experiment by the previous and the new lists as well as GA, we show the new list is better than the previous one and can generate the same length of schedules as GA.

2. Preliminary

A program net (net for short) is denoted by $PN = (N, E)$, where $N$ is a set of nodes consisting of AND-node $\odot$, OR-node $\triangleleft$ and SWITCH-node $\nabla\circ$ and $E$ is a set of directed edges between nodes. A token represents a single datum and token distribution $d = (d_1, \cdots, d_n)$ expresses token numbers on each edge at time $\tau$. The detailed description for program nets is referred to [8]. In this paper, we suppose (i) node firing times are identical (say $1$); and (ii) the program nets are self-cleaning. That is, there is no token remained on each edge after firing [8]. Figure 1 shows the node firing rules of program nets.

**Definition 1.** Let $PN$ be a SWITCH-less net with zero initial token distribution $d^0 = 0$.

(i) $z \in N$ is *firable* and denoted $d^*\text{-}firable$ with respect to $d^*$, iff (a) for AND-node $z$, each its input edge $e$ satisfies $d^*_e \geq 1$; (b) for OR-node $z$, one of its input edges $e$ satisfies $d^*_e \geq 1$.

(ii) Given with processors, firing completion time of $PN$ is the time taken in completing all the firings of nodes of $PN$ by the processors.

(iii) Let $F^*(z)$ denote the maximum firable times of node $z$ at $\tau$: (a) for AND-node $z$, $F^*(z) = \min\{d^*_e, d^*_z\}$; (b) for OR-node $z$, $F^*(z) = d^*_z + d^*_e$, where $e_1$ (and $e_2$) is the input edge(s) of $z$; and $F(\tau)$ denote the set of firable nodes at time $\tau$. And let $t(z)$ denote the total firing times of node $z$ till the termination of the program net’s firing.

Manuscript received September 6, 2004.
Manuscript revised January 4, 2005.
Final manuscript received February 18, 2005.
†The authors are with the Faculty of Education, Yamaguchi University, Yamaguchi-shi, 753-8513 Japan.
††The author is with the Graduate School, Yamaguchi University, Yamaguchi-shi, 753-8512 Japan.
a) E-mail: qgw@inf.edu.yamaguchi-u.ac.jp
DOI: 10.1093/ietfec/e88-a.6.1502

Fig. 1 The firing rules of SWITCH-less program nets.
Definition 2. Let $z_1$ and $z_2$ be two nodes of $PN$.

(i) If there is a directed path from node $z_1$ to node $z_2$ then $z_1$ is predecessor of $z_2$, and $z_2$ is successor of $z_1$; If $(z_1, z_2) \in E$ then $z_1$ is immediate predecessor of $z_2$, and $z_2$ is immediate successor of $z_1$.

(ii) The sets of predecessors and successors of node $z$ are denoted as $Pre(z)$ and $Suc(z)$, and the sets of immediate predecessors and successors of node $z$ are denoted as $IP(z)$ and $IS(z)$ respectively.

Definition 3. Let $Dis(z)$ be the maximum distance from $z$ to termination node $t$ by taking into account the edge numbers. $Z_m^f$ denotes the set of firable nodes that have the longest distance, $D'_m = \max\{Dis(z) | z \in F(\tau)\}$, to the termination node:

$$Z_m^f = \{z^* \in F(\tau) \cap Dis(z^*_m) = D'_m\}.$$

And $Z_{m-i}^f$ denotes the set of firable nodes that have distance $i$ shorter than that of $Z_m^f$, denoted by $D'_{m-i} = D'_m - i$:

$$Z_{m-i}^f = \{z^* \in F(\tau) \cap Dis(z^*_m) = D'_{m-i}\}.$$

There has been proposed a hybrid priority list concatenated from 4 priority lists, $L=L_s \cdot L_w \cdot L_a \cdot L_{all}$ [7], which are defined as follows:

(i) $L_s = \psi_s(\tau)$ and $\psi_s(\tau)$ satisfies:

$$
\psi_s(\tau) = \begin{cases} 
\phi : & \text{if } F(\tau) \geq 2 \\
\phi : & \text{otherwise}
\end{cases}
$$

where $e$ is the input edge of $\tau$;

(ii) $L_w = \psi_w(\tau) \psi_w(\tau)$ and $o_i$ is such an OR-node that its two input nodes, $x_i$ and $y_i$, satisfy $IS(x_i) = IS(y_i) = [o_i]$. And the order of the OR-nodes satisfies $Dis(o_i) > Dis(o_j)$ if $i < j$ and $\psi_w(\tau)$ satisfies:

$$
\psi_w(\tau) = \begin{cases} 
\phi : & \text{if } F(\tau) \geq 2 \\
\phi : & \text{otherwise}
\end{cases}
$$

(iii) $L_a = \psi_a(\tau) \psi_a(\tau)$ and $a_i$ is such an AND-node that its two input nodes, $x_i$ and $y_i$, satisfy $IS(x_i) = IS(y_i) = [a_i]$. And the order of the AND-nodes satisfies $Dis(a_i) > Dis(a_j)$ if $i < j$ and $\psi_a(\tau)$ satisfies:

$$
\psi_a(\tau) = \begin{cases} 
\phi : & \text{if } F(\tau) \geq 2 \\
\phi : & \text{otherwise}
\end{cases}
$$

(iv) $L_{all} = z_1 \ldots z_n$ includes all the nodes satisfying $Dis(z_i) \geq Dis(z_j)$ if $i < j$.

3. A New Hybrid Priority List

In Ref. [7], the following factors have not been taken into account in constructing the previous list $L$: (i) to shrink the distances between the single firable node with the longest distance and other firable nodes; (ii) to guarantee the higher priority for the nodes that can fire more times at $\tau$; and (iii) the nodes that can increase the firable nodes at next time epoch $\tau+1$ should be priorly fired. To improve these, a new priority list has been proposed as follows [9]:

Definition 4: $L' = L_{MaxDis} \cdot L_{f1} \cdot L_{f2} \cdot L_{all}$ is a priority list to schedule acyclic SWITCH-less program nets, where

(i) $L_{MaxDis} = \{ z | (z \in Z_m^f) \land (|Z_m^f| = 1) \land (Z_m^f = \emptyset) \land (f^\tau(z) = 1) \}$

(ii) $L'_{f2} = z_2^1 \ldots z_2^2 \ldots z_2^k$. Here $f^\tau(z_2^k) \geq 2$ holds for each node $z_2^k$ in $L'_{f2}$ and the order of the nodes satisfies $f^\tau(z_2^k) \geq f^\tau(z_2^j)$ if $i < j$. When $f^\tau(z_2^k) = f^\tau(z_2^j)$, $Dis(z_i) > Dis(z_j)$ holds if $i < j$;

(iii) $L'_{all} = \{ \phi | \delta(z_1) \phi(z_2) \cdot \ldots \cdot \phi(z_n) ; \text{otherwise}; \text{where the order of } \delta(z_i) \text{ satisfies } Dis(z_i) > Dis(z_j) \text{ if } i < j, \text{and } \delta(z_i) \text{ is as follows:} \}

when $z_i$ is AND-node:

$$
\delta(z_i) = \begin{cases} 
\phi & : z_2^i \in F(\tau) \lor (IP(z_i) \cap F(\tau) = \emptyset) \lor \\
\delta(z_i) & : \text{other otherwise}
\end{cases}
$$

In the above, $x$ satisfies $\forall z \in IP(z_i)(d^\tau(z_j, z_i) > 0) \lor (f^\tau(z_j) \geq 2)$ and $\sigma(Y)$ is a firable node sequence $y_1 \ldots y_i \ldots y_j \ldots$ of node set $Y$, which satisfies $Dis(y_i) > Dis(y_j)$ if $i < j$. In addition, $d^\tau(z_j, z_i)$ denote the number of token on edge $(z_j, z_i) \in E$.

when $z_i$ is OR-node:

$$
\delta(z_i) = \begin{cases} 
\phi & : (z_2^i \in F(\tau)) \lor (IP(z_i) \cap F(\tau) = \emptyset) \lor \\
\delta(z_i) & : \text{otherwise}
\end{cases}
$$

In the above, $x$ satisfies $\forall z \in IP(z_i)(d^\tau(z_j, z_i) > 0) \lor (f^\tau(z_j) \geq 2)$ and $\sigma(Y)$ is a firable node sequence $y_1 \ldots y_i \ldots y_j \ldots$ of node set $Y$, which satisfies $Dis(y_i) > Dis(y_j)$ if $i < j$. In addition, $d^\tau(z_j, z_i)$ denote the number of token on edge $(z_j, z_i) \in E$.

List $L'$ has been generated to optimal schedules for the program nets whose AND-node possess single input edge [9]. So it is important to verify if it is better than $L$ for the nets whose AND-nodes possess at most two input
4. Experimental Results and Discussions

Genetic algorithm was proposed to study the adaptive process of nature systems [10] and has been known as an effective way to do scheduling of program nets [7]. So it is reasonable to do the evaluation of $L^*$ by applying GA scheduling with the following steps [10].

1) Initialization: Generate initial populations $\{pop_i\}$ that represent here as priority lists.
2) Evaluation: Evaluate each $pop_i$ by fitness function that measures if $pop_i$ leads to a good result (short firing completion time of $PN$).
3) Selection: Select parents by examining the fitness value in order to generate the populations of the next generation.
4) Genetic operations: Generate child populations of the next generation from selected parents by using genetic operations of crossover and mutation.
5) Repeat steps 2), 3) and 4) until termination conditions are satisfied.

We have used the same procedures and also the same parameters, such as population number of one generation, fitness function, probabilities of genetic operations and so on, as done in [7]. In the following, we are to give the experimental results and the discussions.

Our experiments have been done for 400 randomly generated general acyclic SWITCH-less program nets as used in [7], which are classified into two classes as follows: (1) Class I includes 200 acyclic SWITCH-less nets and is divided further into 10 groups with node numbers 20, 40 and so forth; (2) Class II includes also 200 acyclic SWITCH-less nets and is similarly divided into 10 groups with OR-node numbers 5, 10 and so forth. The reason to fix OR-node number is that generally total firing times of node are dependent on node number as well as OR-node number, of which the latter effects $f(z_i)$ much remarkably [5]. Note that the node firing times of the nets are set to 2.

(A) Experimental results for Class I
Table 1 shows the experimental results for 200 nets of Class I. Average firing completion times, $T_L^*$ and $T_{GA}$, are shown for each group by using $L$, $L^*$ and GA scheduling respectively. The following characteristics can be found from Table 1.

1) Except the groups written in italic, $T_{L^*}$ is equal to $T_L^*$ for 4 groups, but is not for the left, and the difference of total averages between $T_L^*$ and $T_{L^*}$ is $1.6(=917.8–916.2)$. The maximal value of difference normalized by the schedule length among the nets is 3.4%.
2) $T_{L^*}$ is completely equal to $T_{GA}$ for all groups. More detailedly, for each program net of the 200 the firing completion time $T_{L^*}$ is exactly the same as $T_{GA}$.

(B) Experimental results for Class II
Table 2 shows the experimental results for Class II. Similarly as for Class I, we can find the followings from Table 2:

1) $T_L^*$ is equal to $T_{L^*}$ for 4 groups, and the difference of total averages between $T_L^*$ and $T_{L^*}$ is $22.4(=11627.8–11605.4)$. The maximal value of difference normalized by the schedule length among the nets is also 3.4%.
2) $T_{L^*}$ is also completely equal to $T_{GA}$ for all groups. And $T_{L^*}$ is exactly the same as $T_{GA}$ for each program net of this class.

From the above experimental results, it is obvious that $L^*$ can generate better schedules than $L$ and further all the schedules of $L^*$ are of the same length as GA scheduling’s. Since GA scheduling has been known to generate almost optimal schedules for the objective program nets of this paper [7], the new list $L^*$ may probably provide optimal schedules to the program nets whose AND-nodes may have two input edges.

5. Concluding Remarks

We have evaluated the performance of a priority list that is supposed to generate better schedules than the previously proposed one. Pointing out the weakness of the previous priority list $L$, we first have introduced a new priority list $L^*$. Doing the simulation experiment by $L^*$ as well as $L$ and GA
(1) the new priority list can generate shorter schedule than previous list;

(2) the new priority list can give completely the same firing completion time as GA scheduling.

Therefore, the new priority list $L^*$ may most probably generate optimal schedules for the general acyclic program nets (whose AND-nodes may have two input edges). As the future work, we try to do the theoretical proof of the optimality of $L^*$ for general acyclic SWITCH-less program nets.

Acknowledgment

This research is partly supported by Grant-in-Aid for Exploratory Research 14655143 of the Ministry of Education, Culture, Sports, Science and Technology (MEXT) of Japan.

References


[7] Q.W. Ge, “A two-processor scheduling method for a class of pro-

### Table 1 Experimental data for Class I.

<table>
<thead>
<tr>
<th>PN</th>
<th>Net conditions</th>
<th>Average scheduling results</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Fixed node number</td>
<td>Average edge number</td>
</tr>
<tr>
<td>$P_{NI}^1$-$P_{NI}^{20}$</td>
<td>20</td>
<td>27</td>
</tr>
<tr>
<td>$P_{NI}^{21}$-$P_{NI}^{40}$</td>
<td>40</td>
<td>56</td>
</tr>
<tr>
<td>$P_{NI}^{41}$-$P_{NI}^{60}$</td>
<td>60</td>
<td>85</td>
</tr>
<tr>
<td>$P_{NI}^{61}$-$P_{NI}^{80}$</td>
<td>80</td>
<td>114</td>
</tr>
<tr>
<td>$P_{NI}^{81}$-$P_{NI}^{100}$</td>
<td>100</td>
<td>143</td>
</tr>
<tr>
<td>$P_{NI}^{101}$-$P_{NI}^{120}$</td>
<td>120</td>
<td>171</td>
</tr>
<tr>
<td>$P_{NI}^{121}$-$P_{NI}^{140}$</td>
<td>140</td>
<td>200</td>
</tr>
<tr>
<td>$P_{NI}^{141}$-$P_{NI}^{160}$</td>
<td>160</td>
<td>230</td>
</tr>
<tr>
<td>$P_{NI}^{161}$-$P_{NI}^{180}$</td>
<td>180</td>
<td>258</td>
</tr>
<tr>
<td>$P_{NI}^{181}$-$P_{NI}^{200}$</td>
<td>200</td>
<td>287</td>
</tr>
<tr>
<td><strong>AVE.</strong></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Table 2 Experimental data for Class II.

<table>
<thead>
<tr>
<th>PN</th>
<th>Net conditions</th>
<th>Average scheduling results</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Fixed OR-node number</td>
<td>Average node number</td>
</tr>
<tr>
<td>$P_{NI}^{1}$-$P_{NI}^{20}$</td>
<td>5</td>
<td>38</td>
</tr>
<tr>
<td>$P_{NI}^{21}$-$P_{NI}^{40}$</td>
<td>10</td>
<td>41</td>
</tr>
<tr>
<td>$P_{NI}^{41}$-$P_{NI}^{60}$</td>
<td>15</td>
<td>62</td>
</tr>
<tr>
<td>$P_{NI}^{61}$-$P_{NI}^{80}$</td>
<td>20</td>
<td>82</td>
</tr>
<tr>
<td>$P_{NI}^{81}$-$P_{NI}^{100}$</td>
<td>25</td>
<td>95</td>
</tr>
<tr>
<td>$P_{NI}^{101}$-$P_{NI}^{120}$</td>
<td>30</td>
<td>133</td>
</tr>
<tr>
<td>$P_{NI}^{121}$-$P_{NI}^{140}$</td>
<td>35</td>
<td>174</td>
</tr>
<tr>
<td>$P_{NI}^{141}$-$P_{NI}^{160}$</td>
<td>40</td>
<td>198</td>
</tr>
<tr>
<td>$P_{NI}^{161}$-$P_{NI}^{180}$</td>
<td>45</td>
<td>259</td>
</tr>
<tr>
<td>$P_{NI}^{181}$-$P_{NI}^{200}$</td>
<td>50</td>
<td>295</td>
</tr>
<tr>
<td><strong>AVE.</strong></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

