Deadlock Detection Based on Resource Allocation Graph

Qinqin Ni
College of Information Engineering
Capital Normal University
Beijing, China
niqiqin0113@163.com

Weizhen Sun
College of Information Engineering
Capital Normal University
Beijing, China
sunweizhen@mail.cnu.edu.cn

Sen Ma
College of Information Engineering
Capital Normal University
Beijing, China
askc037@126.com

Abstract—Deadlock occurs randomly and is difficult to detect, it always has a negative impact on the effective execution of operating system. This paper uses the principle of adjacency matrix, path matrix and strongly-connected component of simple directed graph in graph theory, gives a model of detecting deadlock by exploring strongly-connected component from resource allocation graph. The experiment shows that it can detect resources and processes involved in deadlock effectively by this detection method. The paper provides a new idea for the research of operating system algorithms, and a new way for auxiliary teaching and practical engineering.

Keywords—resource allocation graph; deadlock detection; strongly-connected component

I. INTRODUCTION

Operating system has developed from single program to multiprogramming, which improved the system of resource utilization. However, if the synchronous relationship of parallel process execution is improper, it can produce a serious problem: deadlock [1]. Deadlock refers to a specific condition when two or more processes are each waiting for another to release a resource, or more than two processes are waiting for resources in a circular chain [2]. Deadlock is a common problem in multiprocessing where many processes share a specific type of mutually exclusive resource known as a software, or files, devices. The resource allocation graph of operating system is a simple directed graph [3, 4, 5], which can express the allocation state of system resources in some time. By identifying if there is more than one point in the strongly-connected component of resource allocation graph, it can detect deadlock and restore from deadlock rapidly [5, 6].

II. EXPLORE STRONGLY-CONNECTED COMPONENT FROM RESOURCE ALLOCATION GRAPH

It is helpful to deadlock detection and recover by using directed graph to express the resource request [5]. Resource allocation graph is a kind of simple directed graph, which can express the states of resource allocation in an operating system [6, 7]. The graph has two kinds of nodes: processes, shown as circles, and resources, shown as squares. An arc from a resource node (square) to a process node (circle) means that the resource has previously been requested by, granted to, and is currently held by that process. In Fig. 1 (a), resource R is currently assigned to process A. An arc from a process to a resource means that the process is currently blocked waiting for that resource. In Fig. 1 (b), process B is waiting for resource S. In Fig. 1 (c) we see a deadlock: process C is waiting for resource T, which is currently held by process D. Process D is not about to release resource T because it is waiting for resource U, held by C. Both processes will wait forever [9, 10, 11].

![Figure 1. Resource allocation graphs](image)

(a) Holding a resource (b) Requesting a resource (c) Deadlock

Deadlock detection is a classical graph theory problem. Connectivity is a most important problem in graph, and its application fields are extensive. This paper introduces the adjacency matrix, power of adjacency matrix and path matrix [6].

Definition 1 Suppose that $G = (V, E)$ is a simple graph where $V = \{v_1, v_2, \ldots, v_n\}$. Suppose that the vertices of $G$ are listed arbitrarily as $v_1, v_2, \ldots, v_n$. The adjacency matrix $A$ of $G$, with respect to this listing of the vertices, is an $n \times n$ zero-one matrix with $1$ as its $(i,j)$th entry when $v_i$ and $v_j$ are adjacent, and $0$ as its $(i,j)$th entry when they are not adjacent. In order words, if its adjacency matrix is $A = [a_{ij}]$, then

$$a_{ij} = \begin{cases} 1 & (v_i, v_j) \in E \\ 0 & (v_i, v_j) \notin E \end{cases}$$  \hspace{1cm} (1)

Definition 2 Matrix $A^2$ is the second power of adjacency matrix $A$, with its elements shown as Formula (2)
Definition 3 Suppose that \( G = (V, E) \) is a simple graph where \( |V| = n \). Path matrix \( P \) can be defined an \( n \times n \) matrix, with its elements shown as Formula (3)

\[
p_{i,j} = \begin{cases} 1 & \text{there is at least one path from } v_i \text{ to } v_j \\ 0 & \text{there is no path from } v_i \text{ to } v_j \end{cases}
\]  

Definition 4 Suppose that \( G = (V, E) \) is a simple directed graph, \( P \) is the path matrix of graph \( G \), \( P^T \) is the transpose of matrix \( P \). Define a matrix \( P \times P^T \). Then the strongly-connected component to find out if this system is deadlocked, and if so, which processes and resources are involved.

III. DEADLOCK DETECTION BY EXPLORING STRONGLY-CONNECTED COMPONENT OF GRAPH

Consider a system with seven processes \( P = \{P_1, P_2, \cdots, P_7\} \), and six resources \( R = \{R_1, R_2, \cdots, R_6\} \). The state of which resources are currently owned and which ones are currently being requested is as follows:

1) Process \( P_1 \) holds resource \( R_2 \) and wants \( R_1 \).
2) Process \( P_2 \) holds nothing but wants resource \( R_1 \).
3) Process \( P_3 \) holds resources \( R_4 \) and \( R_1 \) and wants \( R_2 \).
4) Process \( P_4 \) holds resource \( R_5 \) and wants \( R_1 \) and \( R_3 \).
5) Process \( P_5 \) holds resource \( R_6 \) and wants \( R_4 \).
6) Process \( P_6 \) holds nothing and wants resource \( R_3 \).
7) Process \( P_7 \) holds resource \( R_2 \) and wants \( R_5 \).

We can construct resource allocation graph, and explore the strongly-connected component to find out if this system is deadlocked, and if so, which processes and resources are involved.

A. Construct Original Resource Allocation Graph

We can construct the resource graph by the distribution of the resource allocation, just as Fig. 2 shown.

![Figure 2. A resource graph](image)

B. Construct Adjacency Matrix of Graph

According to the resources allocation graph and definition 1, it is easy to get the adjacency matrix of Figure 2, just as matrix (4) shows.

\[
P = \begin{bmatrix}
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{bmatrix}
\]

Let \( n \) be a nonnegative integer and \( G \) a simple directed graph. The length of basic path or the loop is less than \( n \). If \((i,j)\) is not zero, \( v_i \) and \( v_j \) is accessible [12, 13].

<table>
<thead>
<tr>
<th>TABLE I. ARITHMETIC OF MATLAB PROGRAM</th>
</tr>
</thead>
<tbody>
<tr>
<td>[RR,SS] = size(A); % get the size of matrix A</td>
</tr>
<tr>
<td>B = A + A^2 + A^3 + A^4 + A^5; % compute matrix B</td>
</tr>
<tr>
<td>+A6 + A7 + A8 + A9 + A10 + A11 + A12 + A13;</td>
</tr>
<tr>
<td>[R,S] = size(B); % get the size of matrix B</td>
</tr>
<tr>
<td>P = zeros(R,S); % initialize path matrix P</td>
</tr>
<tr>
<td>For i = 1:R; % row loop</td>
</tr>
<tr>
<td>For j = 1:S; % column loop</td>
</tr>
<tr>
<td>if (B(i,j) &gt;= 1) % according to Definition 3, construct path matrix P</td>
</tr>
<tr>
<td>P(i,j) = 1;</td>
</tr>
<tr>
<td>End;</td>
</tr>
<tr>
<td>End;</td>
</tr>
<tr>
<td>PT = P^T; % compute the transpose of matrix P</td>
</tr>
<tr>
<td>Res = P.*PT; % the matrix of strongly-connected component</td>
</tr>
</tbody>
</table>

According to definition 3 and definition 4, the program is shown in Table I. Compute by MATLAB, it is easy to get the path matrix \( P \) and construct the matrix \( P \times P^T \) as shown in matrix (5)
C. Determine Processes and Resources Involved in Deadlock

According to $P \times P^T$ as matrix (5) shows, there are two strongly-connected components: they are constructed by $P_1, P_1, R_1, R_1$ and $P_2, P_2, R_2, R_2, R_2, R_2$ respectively. These processes and resources are involved in deadlock. The original resource graph contains two cycles shown as Figure 3, which can be seen by visual inspection. This proof computing strongly-connected component in resource allocation graph can detect deadlock [14].

In Fig. 4, squares express resources and circles express processes. Users can use the toolbar on the left, and drag the process and resource to the operation area in the middle of the picture. In the operation area, users can select any resource or process to display the related line which can connect the resource and process. When the resource allocation graph is completed, press the execution button, the program will produce the information of adjacency matrix and path matrix in the message-table on the right. And through exploring strongly-connected component, the state of resource allocation will be shown in the status bar below; also the resources and processes which are involved in deadlock will be given, just as Figure 4 shows.

IV. PROGRAM REALIZATIONS

According to the algorithm described as the above, the program of using strongly-connected component to detect deadlock in resource allocation graph based on Java language has a friendly user interface, and complete function, just as Figure 4 shows. And the important code of the program is shown in Table II.

![Figure 3. Cycles constructed by processes and resources involved in deadlock](image)

**TABLE II. PROGRAM OF JAVA**

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>public int[][] operatePPT(int[][] P) {  // compute the product of matrix P and transpose of matrix P</td>
<td></td>
</tr>
<tr>
<td>int[][] PT = new int[row][row]; // the transpose of matrix P</td>
<td></td>
</tr>
<tr>
<td>int[][] C = new int[row][row]; // row loop</td>
<td></td>
</tr>
<tr>
<td>for (int i = 0; i &lt; row; i++) {  // column loop</td>
<td></td>
</tr>
<tr>
<td>for (int j = 0; j &lt; row; j++) { // compute PT, the transpose of matrix P</td>
<td></td>
</tr>
<tr>
<td>PT[i][j] = P[i][j];</td>
<td></td>
</tr>
<tr>
<td>return C;  // return the result</td>
<td></td>
</tr>
</tbody>
</table>

In Fig. 4, squares express resources and circles express processes. Users can use the toolbar on the left, and drag the process and resource to the operation area in the middle of the picture. In the operation area, users can select any resource or process to display the related line which can connect the resource and process. When the resource allocation graph is completed, press the execution button, the program will produce the information of adjacency matrix and path matrix in the message-table on the right. And through exploring strongly-connected component, the state of resource allocation will be shown in the status bar below; also the resources and processes which are involved in deadlock will be given, just as Figure 4 shows.

![Figure 4. User interface of the program](image)

V. CONCLUSION

Graph theory of discrete mathematics plays an important role in computer science such as operating system [15]. Simple directed graph can be used to express the state of resource allocation in operating system, which is helpful to deadlock detection and correction. According to the program, it can prove that it can detect resources and processes involved in deadlock by exploring the strongly-connected component in resource allocation graph. This can provide a new way for the research of operating system algorithm. The program of deadlock detection based on Java can become teaching aids of operating system and some sections of discrete mathematics, which can help students to understand the abstract concept of course. This method also can be used for practical system after it been improved appropriately, which can provide a new method for practical engineering.
REFERENCES


