

CS 442 Introduction to Parallel Processing

Homework 2

Erik Erhardt

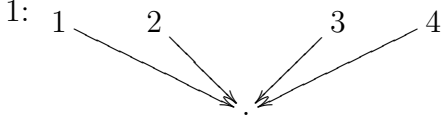
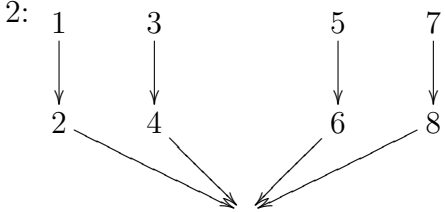
February 20, 2006

GGKK 3.2 *Task graphs.*

1. *Maximum degree of concurrency.*
2. *Critical path length.*
3. *Maximum speed-up over 1 process.*
4. *Minimum number of processes for max speed-up.*
5. *Maximum speed-up if number of processes were 2, 4, 8.*

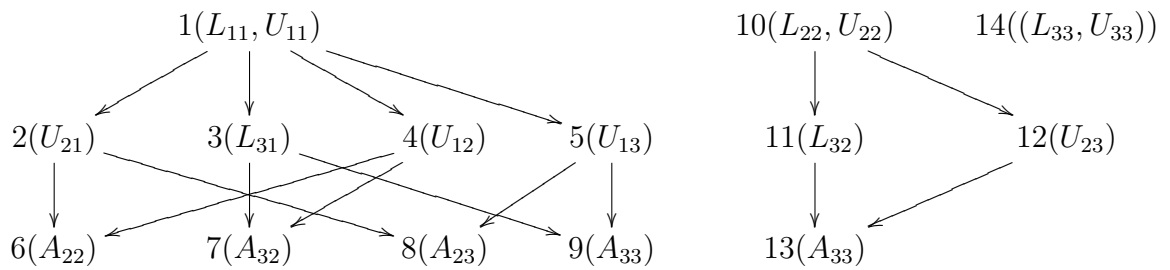
Question	Task Graph			
	(a)	(b)	(c)	(d)
1.	8	8	8	2
2.	4	4	7	8
3. serial	15	15	14	15
3. parallel	4	4	7	8
3. speed-up	3.75	3.75	2	1.875
4.	8	8	8	2
	(parallel/speed-up)			
5. 2 processes	8/1.875	8/1.875	8/1.75	2/1.875
5. 4 processes	5/3	5/3	7/2	2/1.875
5. 8 processes	8/3.75	8/3.75	8/2	2/1.875

GGKK 3.3 Average degree of concurrency and critical-path length for matrix multiplication decomposition.

Figure	Average degree of concurrency	Critical path length
3.10	4	1: 
3.11	4	2: 

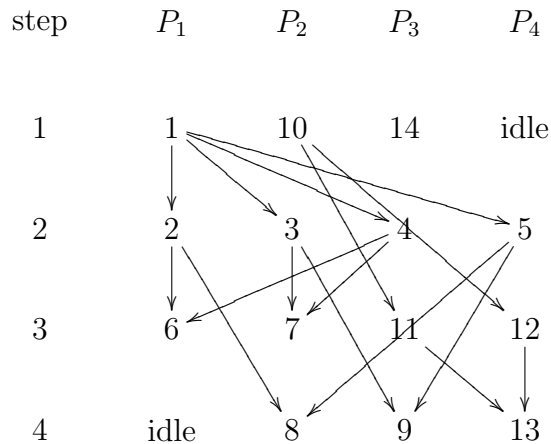
GGKK 3.6 Enumerate all critical paths in LU decomposition in Figure 3.27.

1. Graph task dependency.



2. Enumeration.

Enumeration of 11 critical paths		
1,2,6	10,11,13	14
1,2,8	10,12,13	
1,3,7		
1,3,9		
1,4,6		
1,4,7		
1,5,8		
1,5,9		

GGKK 3.8 *Efficient mapping of the task-dependency onto four processes.***1.** *Efficient mapping onto four processes.***2.** *Description.*

Step 1. 1, 10, 12 must occur to start, with one idle processor. 1 has four children, each with two dependencies. 10 has two children, having one dependency together. 14 has no dependencies.

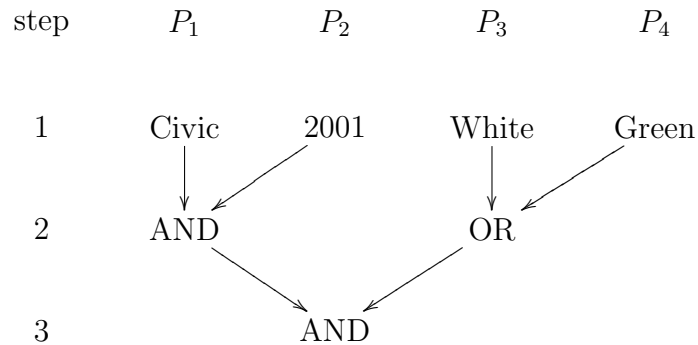
Step 2. Do 2, 3, 4, 5 since they each have the most dependencies, at two each.

Step 3. Do 11, 12 since they have a dependency, and do 6, 7 to complete two paths.

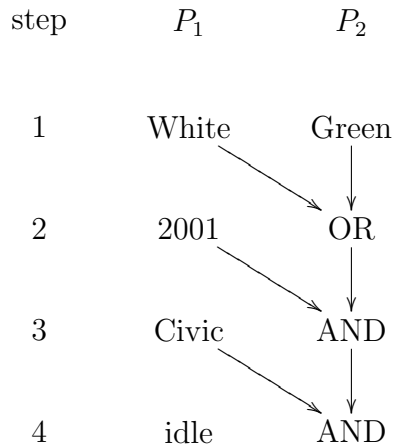
Step 4. Do 8, 9 to complete two paths, and 13 to complete other path.

GGKK 3.14 *Efficient mapping the decomposition of the database query problem (Figures 3.2 and 3.3), and what is the maximum number of processes to use in each case..*

Since the databases in the example are small, I would use the binary tree decomposition in Figure 3.2 over the one in 3.3, with one process per database, for 4 processes.



If I had to use the asymmetric tree, only 2 processes would be required (provided each task takes a comparable amount of time).

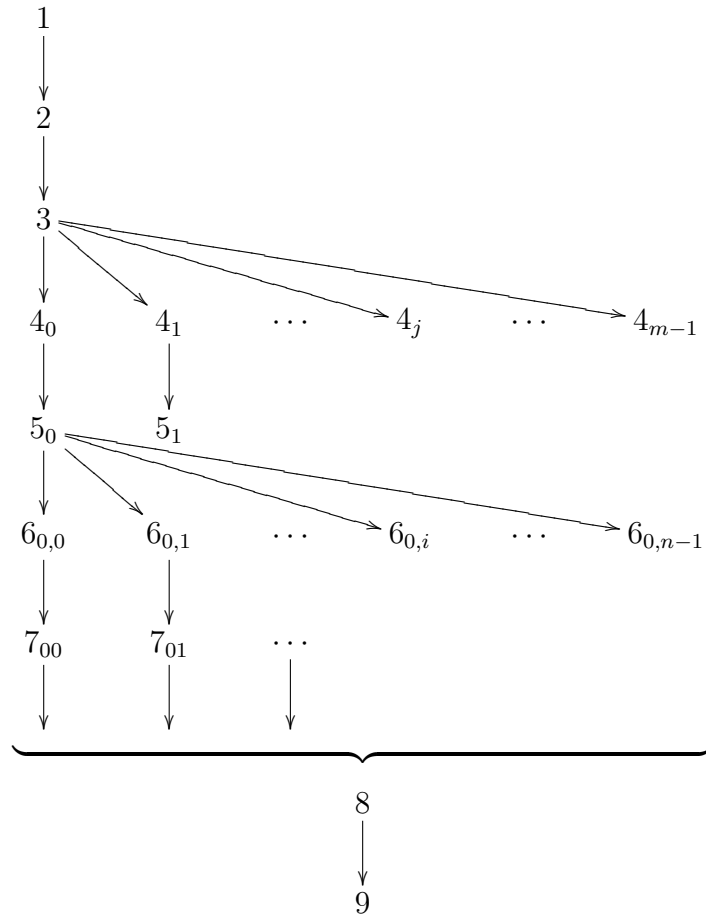


GGKK 3.15 *Task-dependency graph and interaction graph of Algorithm 3.4.*

First, correct line 7 of the algorithm to replace 2^j with k .

1. *Task dependencies (drawn to also show dependencies in the 7_{ji}).*

Allowing only task 7 to be decomposed, ignore the multiple 4's, as those are for interactions.



The interactions are the 7_{ji} having interactions over the j for each i .

GGKK 3.19 *Bucket-sort.*

a. *Decompose on partitioning the input data.*

1. Assign each process $\frac{n}{p}$ values from the input array A .
2. Each process assigns each of its value's indices to one of r buckets.
3. All-to-one gather of indices to processor 0.

b. *Decompose on output data.*

1. Assign each process $\frac{r}{p}$ buckets.
2. Each process reads the entire array A , assigning indices to its buckets.
3. All-to-one gather of buckets to processor 0.

GGKK 3.21 *Running times 1, 2, 3, 4, 5, 5, 10, on two processes.*

Best			Worst		
Strategy	P_1	P_2	Strategy	P_1	P_2
Big	10	5	Small	1	2
to		5	to	4	3
Small	4	3	Big	5	5
balanced	1	2	extra	10	
	15	15 =30		20	10 =30