APPD
TD n°6 - Scheduling (2)

Etienne Mauffret Anthony Dugois

Part 1

Task graph scheduling

Consider scheduling the following task graph:

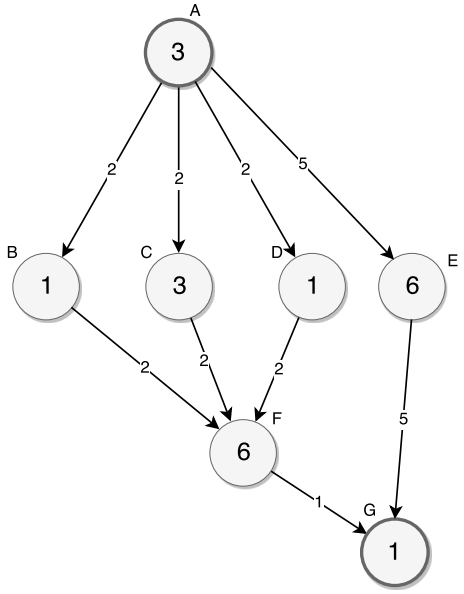

Figure 1: A task graph.

1.1

Scheduling without communications

We first disregard the labelling of the edges and assume communications to come for free.

Question 1

a) Compute the bottom level for each node.

b) Schedule the task graph on 3 processors using a list heuristic. What is the makespan of our schedule? Is it optimal?

Solution 1

a)

A	B	C	D	E	F	G
13	8	10	8	7	7	1

b)

	A	B	C	D	E	F	G
Processor	1	1	2	3	1	2	2
Start time	0	3	3	3	4	6	12

The makespan is 13. We cannot do better because G must be done after F, which must be done after C, which must be done after A, and this path has a total execution time of 13.

1.2 Critical path scheduling

From now on we will consider the communication costs which have to be accounted for when two adjacent tasks are scheduled on different processors.

Question 2

a) How communications should be taken into consideration when computing the bottom level?

b) Compute the bottom level for each node.

c) Schedule the task graph on 3 processors using the critical path heuristic. What is the makespan of our schedule?

Solution 2

a) Communication costs must be added to bottom levels.

b)

A	B	C	D	E	F	G
20	11	13	11	12	8	1

c) Let us recall the critical path heuristic. A task becomes ready when its predecessors have been completed. A processor becomes *busy* when it is affected a task (even if this task does not start immediately), and becomes *idle* again when this task completes. Whenever a task is finished, ready tasks are affected on *idle* processors; we choose the ready task whose bottom level is the largest (we break ties arbitrarily).

This gives the following schedule.

	A	B	C	D	E	F	G
Processor	1	3	1	1	2	1	1
Start time	0	5	3	6	8	8	19

The makespan is 20.

1.3 Modified critical path scheduling

Sometimes, it is worth waiting to schedule a task on a busy processor rather than using the first processor available.

Question 3

a) Which wrong decision, made in the previous section, would be avoided by using this new heuristic?

b) Using this approach, propose a new schedule for our task graph with 3 processors. What is its makespan?

Solution 3

a) We could have affected E on the first processor at time 6, to avoid the communication cost. We also could have affected G on the same processor than E.

b) Instead of affecting each ready task on an *idle* processor, we affect it on the processor that is able to start it at the earliest time.

	A	B	C	D	E	F	G
Processor	1	2	1	3	1	2	1
Start time	0	5	3	5	6	8	15

The makespan is 16.

1.4 Clustering

List heuristics are simple but not very efficient. In order to increase the performance of distributed systems, many clustering algorithms have been developed. By analysing the graph, Sarkar's algorithm removes the communication cost of some edges by forcing the neighbouring tasks to be executed on the same processor. Clustering algorithms may vary in how they decide which edge to remove (or which cluster to merge).

Question 4

a) Apply the clustering algorithms you saw in class to the example and schedule the execution of the clusters you obtain on an infinity of processors. What is the makespan of the schedules?

b) Are the schedules optimal?

Solution 4

a) **Kim and Browne's Linear Clustering.** At each step, find the longest path, turn it into a cluster, and check that the estimated parallel time decreases.

(i) Initially, each individual node is a cluster, and all nodes are *unvisited*. Thus, the cluster set is $\mathcal{C}_0 = \{\{A\},\{B\},\{C\},\{D\},\{E\},\{F\},\{G\}\}$. Let us compute the top/bottom levels of each node:

	A	B	C	D	E	F	G
tl	0	5	5	5	8	10	19
bl	20	11	13	11	12	8	1
$tl + bl$	20	16	18	16	20	18	20

Therefore, the estimated parallel time is $EPT(\mathcal{C}_0) = 20$, and the longest path is AEG (these nodes are marked as *visited*).

(ii) Let us turn AEG into a cluster. The new cluster set is $\mathcal{C}_1 = \{\{A,E,G\},\{B\},\{C\},\{D\},\{F\}\}$. Let us compute the top/bottom levels of each node:

	A	B	C	D	E	F	G
tl	0	5	5	5	3	10	17
bl	18	11	13	11	7	8	1
$tl + bl$	18	16	18	16	10	18	18

Therefore, the estimated parallel time is $EPT(\mathcal{C}_1) = 18 \leq EPT(\mathcal{C}_0)$. The longest path among *unvisited* nodes is CF (these nodes are also marked as *visited*).

(iii) Let us turn CF into a cluster. The new cluster set is $\mathcal{C}_2 = \{\{A,E,G\},\{B\},\{C,F\},\{D\}\}$. Let us compute the top/bottom levels of each node:

	A	B	C	D	E	F	G
tl	0	5	5	5	3	8	15
bl	16	11	11	11	7	8	1
$tl + bl$	16	16	16	16	10	16	16

Therefore, the estimated parallel time is $EPT(\mathcal{C}_2) = 16 \leq EPT(\mathcal{C}_1)$. There is no path among *unvisited* nodes, so we stop here.

Now we assign each cluster on an individual processor (without forgetting dependencies and communication costs). As we have 4 clusters, we use 4 processors.

	A	B	C	D	E	F	G
Processor	1	2	3	4	1	3	1
Start time	0	5	5	5	2	8	15

The makespan is 16.

Sarkar's Greedy Clustering. Sort edges by non-increasing communication cost; consider each edge one by one in-order, and try to allocate both endpoints to the same cluster. If the estimated parallel time decreases (or is equal), the new cluster is valid; otherwise, we ignore it and go to the next edge.

(i) Initially, each individual node is a cluster. Thus, the cluster set is $\mathcal{C}_0 = \{\{A\},\{B\},\{C\},\{D\},\{E\},\{F\},\{G\}\}$. Let us compute the top/bottom levels of each node:

	A	B	C	D	E	F	G
tl	0	5	5	5	8	10	19
bl	20	11	13	11	12	8	1
$tl + bl$	20	16	18	16	20	18	20

Therefore, the estimated parallel time is $EPT(\mathcal{C}_0) = 20$. Now we sort the edges:

$$\{(A,E),(E,G),(A,B),(A,C),(A,D),(B,F),(C,F),(D,F),(F,G)\}.$$

(ii) Consider (A,E). Cluster set would become $\mathcal{C}_1 = \{\{A,E\},\{B\},\{C\},\{D\},\{F\},\{G\}\}$ and corresponding estimated parallel time would be 18, which is lower than $EPT(\mathcal{C}_0) = 20$. The new cluster $\mathcal{C}_1$ is valid.

(iii) Consider (E, G). Cluster set would become $\mathcal{C}_2 = \{\{A, E, G\}, \{B\}, \{C\}, \{D\}, \{F\}\}$ and corresponding estimated parallel time would be 18, which is equal to $EPT(\mathcal{C}_1) = 18$. The new cluster $\mathcal{C}_2$ is valid.

(iv) Consider (A, B). Cluster set would become $\mathcal{C}_3 = \{\{A, B, E, G\}, \{C\}, \{D\}, \{F\}\}$, and corresponding estimated parallel time would be 18, which is equal to $EPT(\mathcal{C}_2) = 18$. The new cluster $\mathcal{C}_3$ is valid.

(v) Consider (A, C). Cluster set would become $\mathcal{C}_4 = \{\{A, C, B, E, G\}, \{D\}, \{F\}\}$, and corresponding estimated parallel time would be 17, which is lower than $EPT(\mathcal{C}_3) = 18$. The new cluster $\mathcal{C}_4$ is valid.

(vi) Consider (A, D). Cluster set would become $\mathcal{C}_5 = \{\{A, C, B, D, E, G\}, \{F\}\}$, and corresponding estimated parallel time would be 18, which is greater than $EPT(\mathcal{C}_4) = 17$. The new cluster $\mathcal{C}_5$ is not valid.

(vii) In the same manner, we discard (B, F) and (C, F).

(viii) Consider (D, F). Cluster set would become $\mathcal{C}_5 = \{\{A, C, B, E, G\}, \{D, F\}\}$, and corresponding estimated parallel time would be 17, which is equal to $EPT(\mathcal{C}_4) = 17$. The new cluster $\mathcal{C}_5$ is valid.

(ix) We discard (F, G).

Now we assign each cluster on an individual processor (without forgetting dependencies and communication costs). As we have 2 clusters, we use 2 processors.

	A	B	C	D	E	F	G
Processor	1	1	1	2	1	2	1
Start time	0	6	3	5	7	9	16

The makespan is 17.

Dominant Sequence Clustering. At each step, find a longest path, find the edge with the largest weight on this path, and try to allocate both endpoints to the same cluster. If the estimated parallel time decreases, the new cluster set is valid; otherwise ignore it, and choose another edge.

(i) Initially, all edges are *unvisited*. The cluster set is $\mathcal{C}_0 = \{\{A\}, \{B\}, \{C\}, \{D\}, \{E\}, \{F\}, \{G\}\}$. Let us compute the top/bottom levels of each node:

	A	B	C	D	E	F	G
tl	0	5	5	5	8	10	19
bl	20	11	13	11	12	8	1
$tl + bl$	20	16	18	16	20	18	20

Therefore, the estimated parallel time is $EPT(\mathcal{C}_0) = 20$. A longest path is AEG.

(ii) Choose edge (A, E). Cluster set would become $\mathcal{C}_1 = \{\{A, E\}, \{B\}, \{C\}, \{D\}, \{F\}, \{G\}\}$ and corresponding estimated parallel time would be 18, which is lower than $EPT(\mathcal{C}_0) = 20$. The new cluster $\mathcal{C}_1$ is valid. Now a longest path is ACFG.

(iii) Choose edge (A, C). Cluster set would become $\mathcal{C}_2 = \{\{A, C, E\}, \{B\}, \{D\}, \{F\}, \{G\}\}$ and corresponding estimated parallel time would be 18, which is equal to $EPT(\mathcal{C}_1) = 18$. The new cluster $\mathcal{C}_2$ is valid. Now a longest path is ACEG.

(iv) Choose edge (E, G). Cluster set would become $\mathcal{C}_3 = \{\{A, C, E, G\}, \{B\}, \{D\}, \{F\}\}$ and corresponding estimated parallel time would be 16, which is lower than $EPT(\mathcal{C}_2) = 18$. The new cluster $\mathcal{C}_3$ is valid. Now a longest path is ABFG.

(v) Choose edge (A, B). Cluster set would become $\mathcal{C}_4 = \{\{A, B, C, E, G\}, \{D\}, \{F\}\}$ and corresponding estimated parallel time would be 17, which is greater than $EPT(\mathcal{C}_3) = 16$. The new cluster $\mathcal{C}_4$ is not valid.

(vi) Choose edge (B, F). Cluster set would become $\mathcal{C}_4 = \{\{A, C, E, G\}, \{B, F\}, \{D\}\}$ and corresponding estimated parallel time would be 16, which is equal to $EPT(\mathcal{C}_3) = 16$. The new cluster $\mathcal{C}_4$ is valid. Now a longest path is ACFG.

(vii) Any *unvisited* edge of ACFG is a bad choice.

(viii) Try another longest path: ADFG. Edge (A, D) is a bad choice. Choose edge (D, F). Cluster set would become $\mathcal{C}_5 = \{\{A, C, E, G\}, \{D, B, F\}\}$ and corresponding estimated parallel time would be 16, which is equal to $EPT(\mathcal{C}_4) = 16$. The new cluster $\mathcal{C}_5$ is valid.

Now we assign each cluster on an individual processor (without forgetting dependencies and communication costs). As we have 2 clusters, we use 2 processors. The makespan is 16.

b) The optimal schedule gives a makespan of 16.

Part 2

List Scheduling anomalies

Consider the DAG in Figure 2 where a label X (w) means that task X has weight w. For instance A has weight 8.

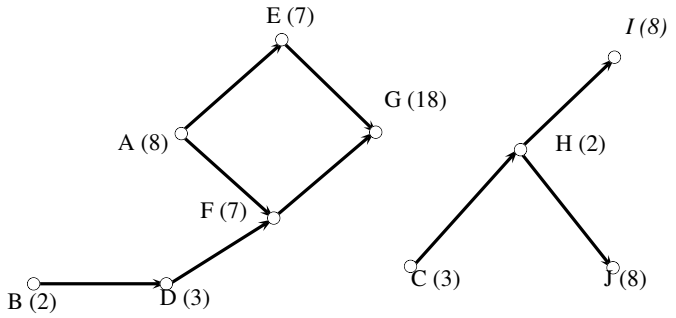

Figure 2: A DAG to reveal anomalies of list scheduling.

Question 5

a) What is the makespan achieved by critical path list scheduling with 2 processors? Is it optimal?

b) Assume that each task weight is decreased by one unit (now A has weight 7, B has weight 1, and so on). Show that the makespan achieved by critical path list scheduling increases. Show that, somewhat shockingly, it is impossible to get a lower makespan than before with a list scheduling algorithm.

c) Going back to original task weights (see Figure 2), assume that we have 3 processors. Show that the makespan achieved by critical path list scheduling increases. Show that the makespan achieved by any list scheduling algorithm, shockingly again, increases.

Solution 5

a) First we compute the bottom levels:

A	B	C	D	E	F	G	H	I	J
33	30	13	28	25	25	18	10	8	8

Now we can schedule using the critical path list scheduling heuristic:

	A	B	C	D	E	F	G	H	I	J
Processor	1	2	2	2	1	2	1	2	2	2
Start time	0	0	5	2	8	8	15	15	17	25

The makespan is 33, which is optimal, as it is the length of the critical path AEG.

b) First we compute the bottom levels:

A	B	C	D	E	F	G	H	I	J
30	26	10	25	23	23	17	8	7	7

Now we can schedule using the critical path list scheduling heuristic:

	A	B	C	D	E	F	G	H	I	J
Processor	1	2	2	2	1	1	1	2	2	2
Start time	0	0	3	1	7	13	19	5	6	13

The makespan is 36.

Suppose there exists a list-scheduling heuristic that gives a makespan lower than or equal to 32.

Then E and F must start at most at time 9, because G depends on them and has a duration of 17 time units ($9 + 6 + 17 = 32$). But they cannot start before A is done, so A must complete at most at time 9.

Moreover, while A is running, the only tasks that can be processed are B, D, C and H, and they all can be done on the same processor in 6 time units.

Therefore, there will necessarily be a processor that is free after the completion of H and before the completion of A. At this time, the only ready tasks are I and J. So I or J will start before the completion of A. But they cannot start before H, which cannot start before C, so I (or J) will complete at time 10 in the best case.

Hence, at least one task among E and F cannot start at or before time 9, and the makespan will necessarily be greater than 32.

c) We take our original bottom levels:

A	B	C	D	E	F	G	H	I	J
33	30	13	28	25	25	18	10	8	8

Now we can schedule using the critical path list scheduling heuristic:

	A	B	C	D	E	F	G	H	I	J
Processor	1	2	3	2	1	2	1	3	2	3
Start time	0	0	0	2	8	13	20	3	5	5

The makespan is 38.

The only ready tasks at time 0 are A, B and C. We have 3 processors, and as we use a list-scheduling algorithm (that does not let a processor idle when at least one task is ready) we will start A, B and C at time 0.

The first processor to be idle again is the one that executes B (it completes before A and C), and the only ready task at this time is D, so we have no choice.

Then the processor to be idle first will be the one that executes C, and the only ready task at this time is H, so we have no choice.

Then the two processors on which we scheduled B and C will finish at time 5, and the ready tasks are I and J, which have the same execution time (so it does not matter where we put them).

Then the processor that executes A will finish, and the ready tasks are E and F, which have the same execution time; it does not matter if we choose E or F.

Then the processors that execute I and J will finish. The only ready task is E or F (the one that we did not choose). So we have no choice.

Overall, either we have no choice, or our choice does not matter. Therefore, G will necessarily start at time 20 if we use a list-scheduling algorithm.

Part 3

Approximation result

Recall that any list schedule is a $\left(2 - \frac{1}{p}\right)$-approximation algorithm on a platform with p processors. We now consider a task graph with communication costs, where $c_{\max} = \max_{T,T' \in V} c(T, T')$ is the maximum communication cost, and $w_{\min} = \min_{T \in V} w(T)$ is the minimum computation cost, and we consider the modified critical path (MCP) list-scheduling heuristic.

Question 6

a) Build the worst-case example for MCP you can think of, i.e., where the ratio between the makespan returned by the MCP heuristic and the optimal makespan is as large as possible. Do you think MCP is still a $(2 - 1/p)$-approximation algorithm?

b) We now consider that $c_{\max} \leq w_{\min}$. Propose a worst-case example satisfying this constraint.

c) When $c_{\max} \leq w_{\min}$, can you prove that MCP is an approximation algorithm? What is the approximation ratio? Otherwise, prove that there is no constant approximation ratio for MCP.

d) Derive an approximation ratio (or inapproximability result) for MCP in the general case.

Solution 6

a)