B10615029 王孝宇

Project 2 Constrained Scheduling →

使用方法為計算出每個 Node 在 ASAP 與 ALAP 下的時間點與 slack，當開始排程時，先從 level 1 開始做，將此時有機會做到的 Node 丟到 queue 中，每次只選出一個點來做(slack=0 優先度最高)，而要是我下一次進入的 Node 比這次時間的 Node 還要多時（一定倍數的比例），便增加我的 Resource 數量，避免一次全部做增加大量 Resource；且很早進入 queue 的 Node(slack 寬裕)，會設定他的 waitTime，達到一定的等待時間則會讓他的優先度達到最高，避免他 starvation。

Project 3 Latency Constrained Scheduling →

使用方法為 Force-Directed；計算每個點的 Self-Force，與影響他的 Predecessor/Successor-Force；做出每一個點的最好選擇，但是這個由哪個點開始計算就很重要，因此我做了由 Name 的大小來做（stl::string deafult）跟由子孫的多寡來排程做比較。

測資	Time Restrict
pdf 上的 sample02.blif	4、5、10、20、30
aoi_t481.blif	10、15、20、25、30
B10615029.blif(自己所想)	8、10、12、14、16
aoi_big2.blif	100、150

我們用這些測資來看看在時間限制不同下，各種排程方式下，所花的時間(gprof)與結果：

- sample02.blif
  - Constrained Scheduling MR-LCS

Restrict Time	AND	OR	NOT	Cost time	Actually use time
4	2	1	1	0.00	4
5	1	1	1	0.00	5
10	1	1	1	0.00	5
20	1	1	1	0.00	5
30	1	1	1	0.00	5

  - Latency Constrained Scheduling default

Restrict Time	AND	OR	NOT	Cost time	Actually use time
4	2	1	1	0.00	4
5	1	1	1	0.00	5
10	1	1	1	0.00	5
20	1	1	1	0.00	5
30	1	1	1	0.00	5

  - Latency Constrained Scheduling Successor big first

Restrict Time	AND	OR	NOT	Cost time	Actually use time
4	2	1	1	0.00	4
5	2	1	1	0.00	5
10	1	1	1	0.00	5
20	1	1	1	0.00	5
30	1	1	1	0.00	5

- Latency Constrained Scheduling Successor small first

Restrict Time	AND	OR	NOT	Cost time	Actually use time
4	2	2	1	0.00	4
5	2	2	1	0.00	4
10	2	1	1	0.00	9
20	1	1	1	0.00	19
30	1	1	1	0.00	29

測試資料 1 可以發現，在 MR-LCS 和依照 string 資料結構，選點順序是正常的，不過當我選點時改變使用 Sucessor 多寡來取決時，少得先做時在 slack 空間越大時，會拖到後面做，導致在使用時間上會拖到最後一刻；而在多的點先做時，他在考慮時就就會先提前做完，效果比較好，但是就最終結果來看，MR-LCS 和 LS 差不多。

aoi_t481.blif

- Constrained Scheduling MR-LCS

Restrict Time	AND	OR	NOT	Cost time	Actually use time
10	296	20	13	0.00	10
15	198	20	3	0.00	15
20	129	20	3	0.00	20
25	129	9	3	0.01	25
30	129	5	3	0.02	27

■ Latency Constrained Scheduling default

Restrict Time	AND	OR	NOT	Cost time	Actually use time
10	137	19	14	0.00	10
15	86	18	12	0.02	15
20	78	17	11	0.02	20
25	78	18	10	0.03	25
30	78	17	11	0.03	30

■ Latency Constrained Scheduling Successor big first

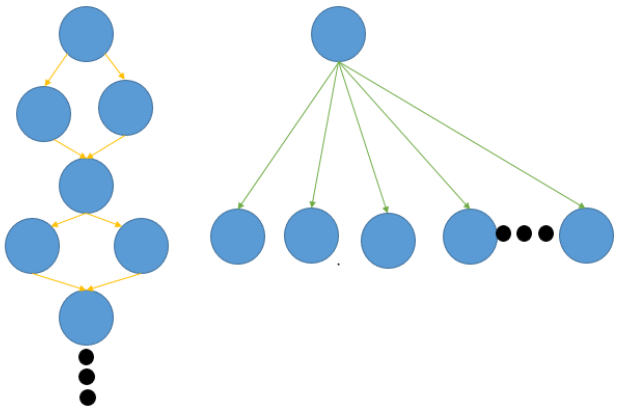
Restrict Time	AND	OR	NOT	Cost time	Actually use time
10	175	25	10	0.37	10
15	171	19	10	0.37	15
20	171	20	8	0.42	20
25	154	19	7	0.43	25
30	139	19	5	0.39	30

■ Latency Constrained Scheduling Successor small first

Restrict Time	AND	OR	NOT	Cost time	Actually use time
10	280	27	16	0.00	9
15	202	18	16	0.40	15
20	135	13	16	0.38	20
25	84	7	16	0.39	25
30	91	8	16	0.42	30

測試資料 2，因為資料量上升因為，FS 演算法為 O(N^3)，所以可以看到花費時間比 MR-LCS 高出一些，不過那也只限於我的選點是電腦自動決定，當我是用 Successor 大小來選點時，因為要花額外時間來計算各個點有多少 Successor，所以 cost 會花上些時間，不過在排程上呢，限制時間提高上，各點的 Resourse 都有下降的趨勢，FS 最後大多都可以將 Resource 限制到將近 100 個，不過與 MR-LCS 相比，表現卻無太大區別。

- B10615029.blif
  - 此測資圖形→



- Constrained Scheduling MR-LCS

Restrict Time	AND	OR	NOT	Cost time	Actually use time
8	2	2	1	0.00	8
10	2	2	1	0.00	10
12	1	2	1	0.00	12
14	1	2	1	0.00	12
16	1	2	1	0.00	12

Latency Constrained Scheduling default

Restrict Time	AND	OR	NOT	Cost time	Actually use time
8	2	5	1	0.00	8
10	3	2	1	0.00	9
12	3	1	1	0.00	11
14	1	1	1	0.00	13
16	1	1	1	0.00	15

Latency Constrained Scheduling Successor big first

Restrict Time	AND	OR	NOT	Cost time	Actually use time
8	2	3	1	0.00	8
10	2	2	1	0.00	10
12	2	1	1	0.00	11
14	1	1	1	0.00	12
16	1	1	1	0.00	16

Latency Constrained Scheduling Successor small first

Restrict Time	AND	OR	NOT	Cost time	Actually use time
8	2	3	1	0.00	8
10	2	2	1	0.00	10
12	2	1	1	0.00	11
14	2	1	1	0.00	11
16	2	1	1	0.00	11

此筆測資有一條路是直向發展，有一條路是橫向發展，要是你的一開始選點沒選好，會導致後面需要大量加上 Resource，而我在 MR-LCS，做 waittime 跟 Resource+1 就是為了要預防此狀況，所以可以看出他的 resource 控制的很好；而在 FS 上當我使用 Successor 少得先做時，時間可以控制那個時間點下，最少的 Resource；不過卻無法限制到最少 Resource，時間比預期快但是你的 Resource 還是用比較多，有點可惜;但此測資來看來講 FS 都可以在一方面達到好的效益。

➢ aoi_big2.blif

Restrict 100

Method	AND	OR	NOT	Cost time	Actually use time
MR-LCS	328	58	3	1s	91
LS	719	127	9	30s	100
LS-Big	887	181	10	1.7min	100
LS-Small	153	342	9	1.6min	99

Restrict 150

Method	AND	OR	NOT	Cost time	Actually use time
MR-LCS	328	58	3	1s	91
LS	724	122	9	45s	150
LS-Big	884	184	10	2.4min	150
LS-Small	120	287	9	2.6min	149

當我們使用大的測資時，可以明顯看到 FS 演算法跟 MR-LCS 演算法時間大幅差距，不過結果卻沒有預期的比 MR-LCS 好，甚至 MR-LCS 表現的還要更加好，只有 Preccessor 小的優先做表現的比 MR-LCS 好，但也花了很多時間。

Summary→

整體來看，當我的限值時間往上爬時，每筆演算法都可以有更進一步的縮減，且 FS 演算法在爬升上有更高的回報，不過也僅限於 Resourcess 由小大大先做，其他上表現差勁；MR-LCS 到一定程度後爬升坡度便緩緩前進，甚至不動，但是 MR-LCS 表現不會比 LS 還要差 ，雖然有可能是我的 MR-LCS 有針對一些狀況去做特別處理，當我要做的 Resource 大量增加時，我的 Resource 也會做出對應，呈現指數成長，且也不會讓能做的 Node 等太多時間，所以測資出來結果都不會遜色於 FS，甚至優於他們；FS 在 cost time 上會高出 MR-LCS 許多，不過做出來的結果卻沒有更好；在選點上，我做了三種變化，另外兩種依照 Successor 來選擇，他所花費的時間會是更是原本 FS 的數倍以上，但是預期得到結果在 Successor 小的做優先在大多情況會比較優秀，甚至在大測資比 MR-LCS 還可以達到更好的結果；不過 Successor 大的做優先在一些測資上，會看到比原本還要差的結果；在這個 FS 選點上結果，雖然 Successor 小的做優先情況大多表現得比原本 FS 好，但是因此花更多時間，不過卻有機會比 MR-LCS 拿到更好結果。