

How to avoid wasting parallel program performance

András Z. Salamon^{1,2}

¹Computing Laboratory, University of Oxford

²Oxford-Man Institute of Quantitative Finance

07 April 2009



cheesy image of hourglass



cheesy image of lightbulb



cheesy image of lit lightbulb



cheesy image of lots of computers



Matlab Parallel Computing Toolbox

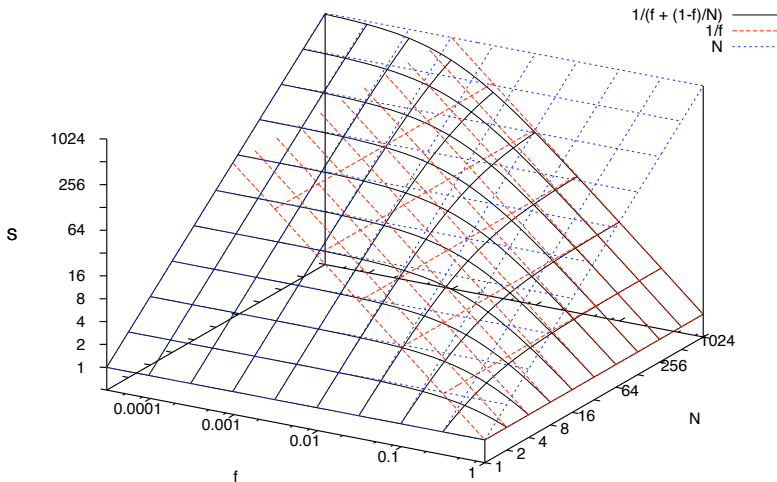
Mathematica 7

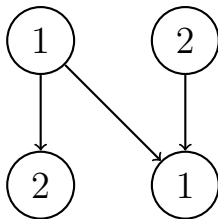
Khronos OpenCL (Apple, Nvidia, AMD, Intel)

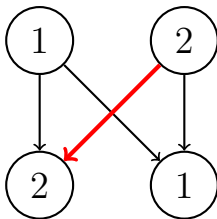
Amdahl's Law



speedup S , processors N , sequential fraction f







Activity network



Activity network



1

Activity network

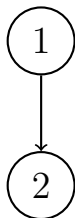


1

$$a[t+1,i] = a[t,i] + \text{sqrt}(a[t,i-1]**2+a[t,i]+a[t,i+1]**2);$$

...

Activity network

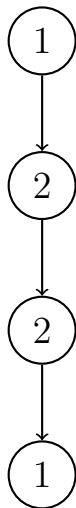


$$a[t+1,i] = a[t,i] + \text{sqrt}(a[t,i-1]**2+a[t,i]+a[t,i+1]**2);$$

...

$$c[t+1,i] = a[t+1,i] + \dots$$

Activity network



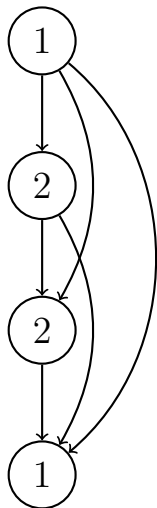
Problem

Series-parallelisation

4/3 conjecture

Conclusion

Activity network



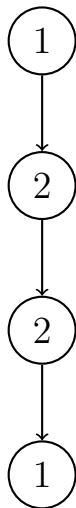
Problem

Series-parallelisation

4/3 conjecture

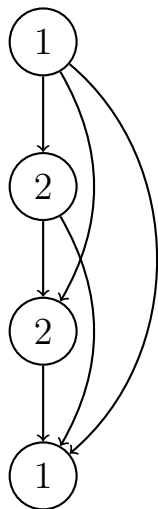
Conclusion

Activity network



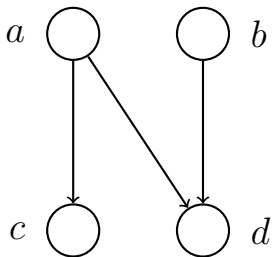
transitive reduction

Activity network



transitive closure = partial order

Not series-parallel

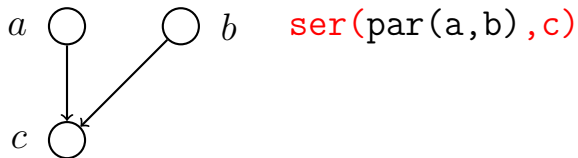


Series-parallel

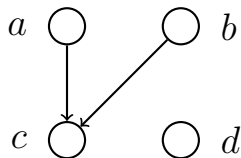


$a \circ \quad \circ b$ `par(a,b)`

Series-parallel

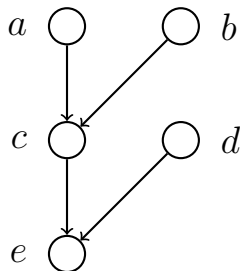


Series-parallel



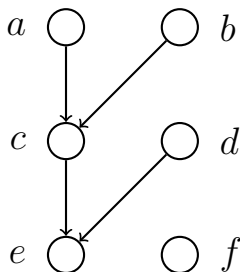
$\text{par}(\text{ser}(\text{par}(a, b), c), d)$

Series-parallel



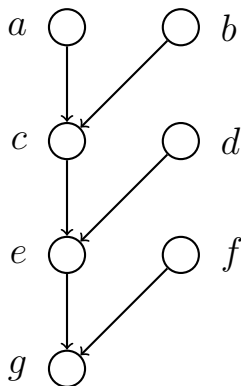
$\text{ser}(\text{par}(\text{ser}(\text{par}(a, b), c), d), e)$

Series-parallel



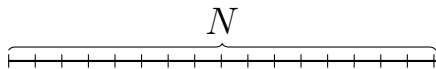
$\text{par}(\text{ser}(\text{par}(\text{ser}(\text{par}(a, b), c), d), e), f)$

Series-parallel



$\text{ser}(\text{par}(\text{ser}(\text{par}(\text{ser}(\text{par}(a, b), c), d), e), f), g)$

1D Flow Model



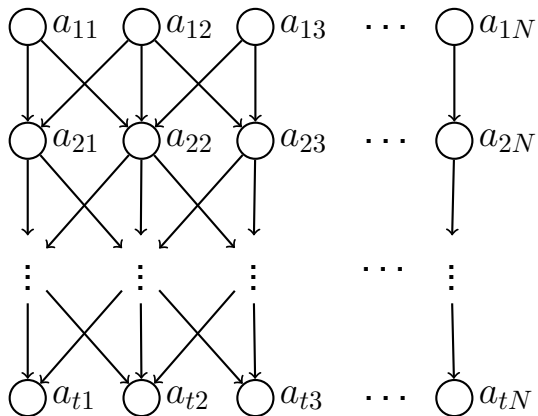
time 1: $a_{11}, a_{12}, a_{13}, \dots, a_{1N}$

time 2: $a_{21}, a_{22}, a_{23}, \dots, a_{2N}$

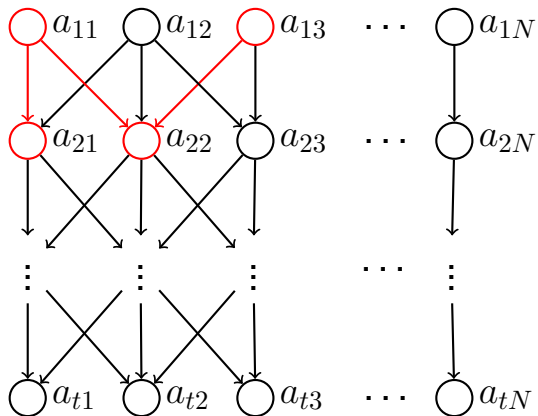
\vdots

time t : $a_{t1}, a_{t2}, a_{t3}, \dots, a_{tN}$

Neighbour synchronization



Neighbour synchronization



for

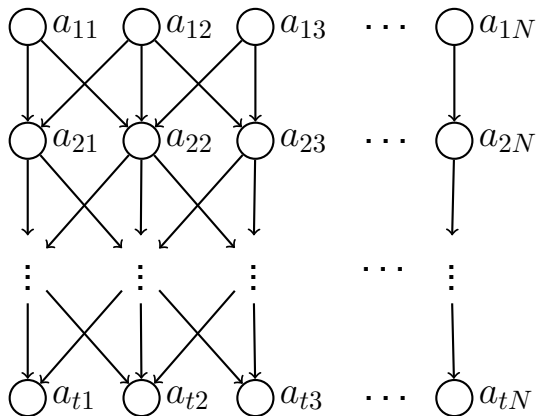
parfor

parfor

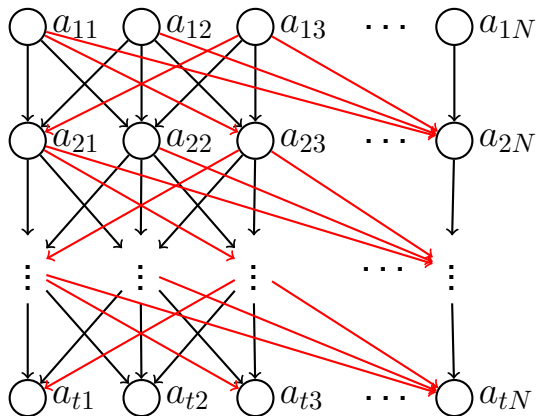


```
for  $i = 1 : t$   
    parfor  $j = 1 : N$   
        % compute  $a_{ij}$   
    end  
end
```

parfor



parfor



$L(G)$



Let $d_G(x)$ be the depth of vertex x in activity network G .

Observation $\{x \mid d_G(x) = i\}$ is independent set.

For an activity network G let $L(G)$ be the unique activity network with weighted vertices $V(G)$ and edges E' so that for $x, y \in V(G)$

- ▶ $d_{L(G)}(x) = d_G(x)$, and
- ▶ $d_{L(G)}(x) < d_{L(G)}(y) \Leftrightarrow (x, y) \in E'$.

$L(G)$



$T(x)$ is the weight of activity x .

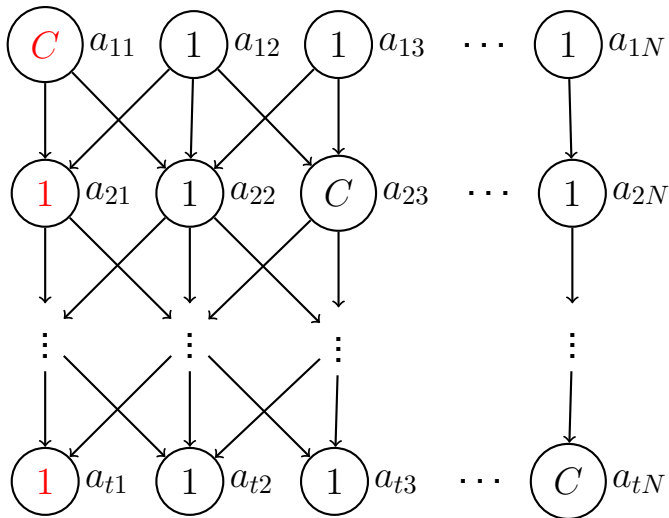
$T(G)$ is the makespan of activity network G , i.e. maximum of $\sum_{x \in C} T(x)$ over all chains C in G .

Observation $T(L(G))/T(G) \geq 1$. (“slowdown”)
Proposition

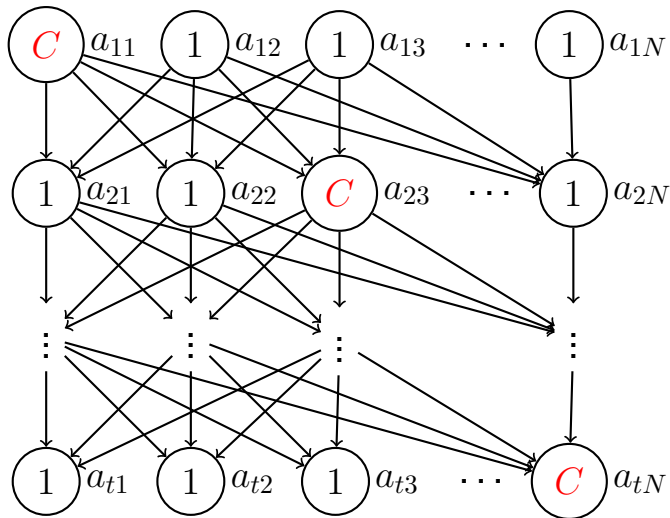
$$\frac{T(L(G))}{T(G)} \leq \frac{\max_{x \in V(G)} T(x)}{\min_{x \in V(G)} T(x)}.$$

Observation This bound is (essentially) tight.

$T(G)$



$T(L(G))$



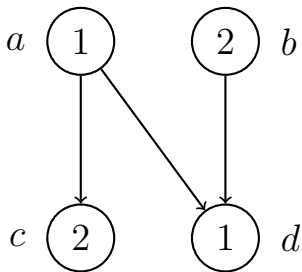
$$\begin{aligned}\frac{T(L(G))}{T(G)} &= \frac{Ct}{C + t - 1} \\ &\leq C(1 + (C - 1)/N)^{-1} \\ &\rightarrow C \text{ as } N \rightarrow \infty\end{aligned}$$

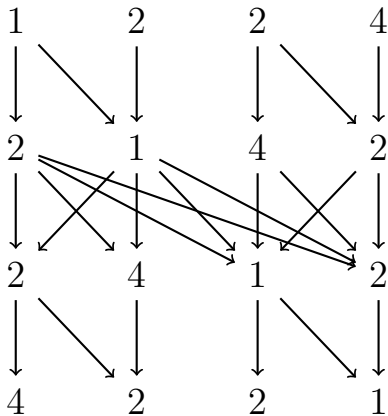
Add precedence constraints to an activity network G to obtain series-parallel G' , ideally $T(G') < T(L(G))$.

Conjecture There is constant σ such that for any activity network G there is G' with slowdown at most σ .

Proved for $|V(G)| \leq 6$ and $\sigma = 4/3$.

$$\sigma \geq 4/3$$





$$T(G) = 9$$

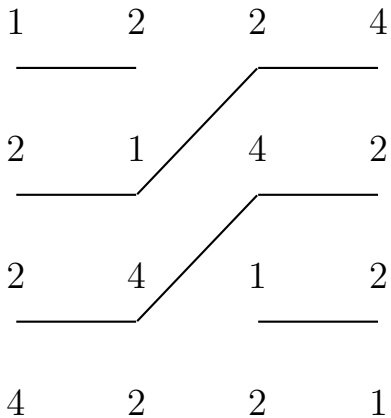
1 2 2 4

2 1 4 2

2 4 1 2

4 2 2 1

$$\frac{T(L(G))}{T(G)} = 16/9 > 4/3$$



$$\frac{T(G')}{T(G)} = 12/9 = 4/3$$

σ -BSP

Input: activity network G

Question: find G' with slowdown σ .

Conjecture $4/3$ -BSP is NP-hard.

Different to most NP-hard problems: edge modification problems typically count number of edges added, optimization does not usually modify structure, how to capture SAT?

Take-home message

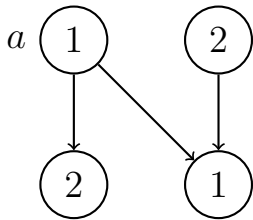


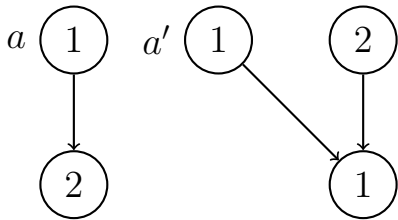
- ▶ series-parallel constructs force implicit precedence constraints
- ▶ G to G' results in slowdown
- ▶ simple algorithm $G' = L(G)$: unbounded slowdown
- ▶ bounded slowdown: NP-hard?
- ▶ **slowdown $4/3$ is inherent**

What to do?



- ▶ disprove conjectures, accept $4/3$ slowdown
- ▶ accept yet another NP-hard problem, **on top of** scheduling, register allocation, optimal code partitioning, . . .
- ▶ activities similar size: network, cache miss
- ▶ better languages: CCS/CSP difficult
- ▶ duplicate activities





Further work



- ▶ language features for neighbour synch?
 - ▶ approximation scheme for slowdown?
 - ▶ is small amount of duplication sufficient?
-