Asynchronous distributed algorithms

Keith Briggs

Keith.Briggs@bt.com

more.btexact.com/people/briggsk2

Bristol Engineering Maths 2003 Dec 02 1100

TYPESET 2003 NOVEMBER 28 17:06 IN PDFLATEX ON A LINUX SYSTEM

Outline

- Motivation for asynchronous distributed algorithms (ADAs)
- Simulation techniques
- Some real examples
- Future work

Models of computing

- Single CPU + RAM
- Multiple CPU + RAM
- Cluster (Beowulf, mosix)
- ZetaGrid
- Asynchronous distributed algorithms (ADAs)
- internet
- . . .

Can we unify these?

Asynchronous distributed model

- Network of identical nodes, with message queue
- Each knows only its neighbours
- Each performs the same subalgorithm
- Each runs asynchronously wrt neighbours
- Protocol: a finite set of pre-specified messages
- Indefinite delay before reply to message

Theme: atheism

Cooperation of nodes

Required: to perform some useful global actions:

- ★ Reboot system
- ★ Detect node failures
- ★ Count total number of nodes
- ★ Name nodes and elect leader
- ★ Build spanning trees
- ★ Find shortest paths
- ★ Compute and optimize network flows

Simulating an ADA on one processor

In decreasing order of weight:

- unix processes
- kernel threads
- threads in python, java etc.
- other tricks

Python threads

```
import threading

class Node:

   def init(links):
     message_queue=[]
     # ...

   def run():
     while 1:
       # ...

   def send(target):
     # ...

   def receive(source):
     # ...

nodes=[Node([2,3]),Node([3]),Node([1])]

for node in nodes:
  Thread(node.run).start()
```

Counting all nodes

- All nodes *asleep* except 0, who is *awake* and sends to all neighbours
  - if receiver awake: return ‘reject’
  - if receiver asleep:
    - ▷ *wake up and relay message to neighbours*
    - ▷ *return number of nodes from relay replies*
    - ▷ *receiver returns sum*$+1$ *to requester*

Shortest path routing to node 0

Asynchronous Bellman-Ford algorithm:

$$x(i) \leftarrow \min_{j \,\in\, \text{neighbours of node } i} x(j)+d(j)$$

where:

- $x(i)$ is node i's current estimate of the shortest path to node 0
- $d(j)$ is the distance to node j (one hop)

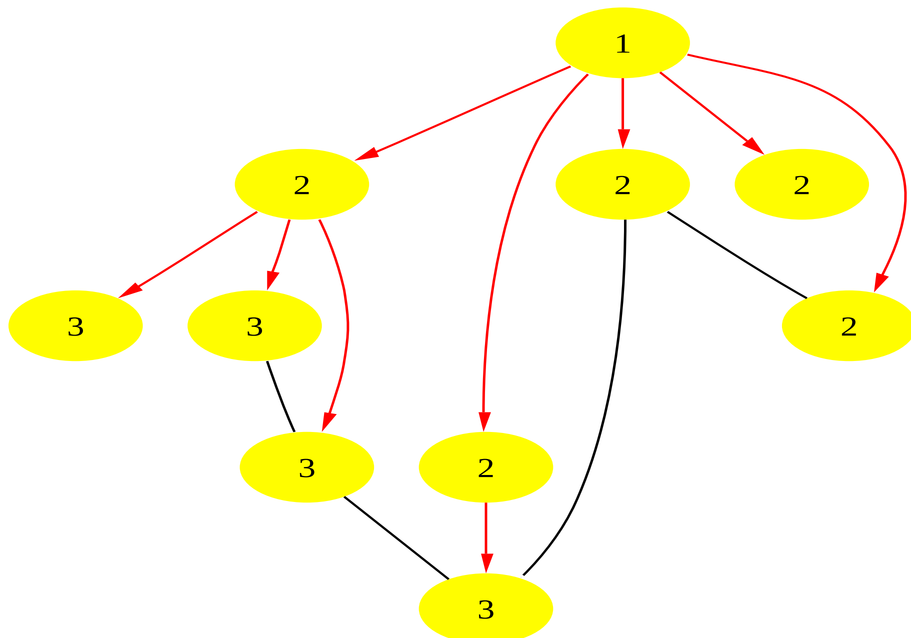
Termination?

Building a spanning tree

Root node has weight 1

`while 1:`

- node sends its weight to neighbours
- if receiver is unweighted, adopt sender's weight+1
- else if receiver's weight $>$ sender's weight$+1$
  - *receiver adopts new parent*



Layering

optimization, flows, . . .
counting; spanning tree
routing
reboot; failure detection
adjacency

Graph centre - definitions

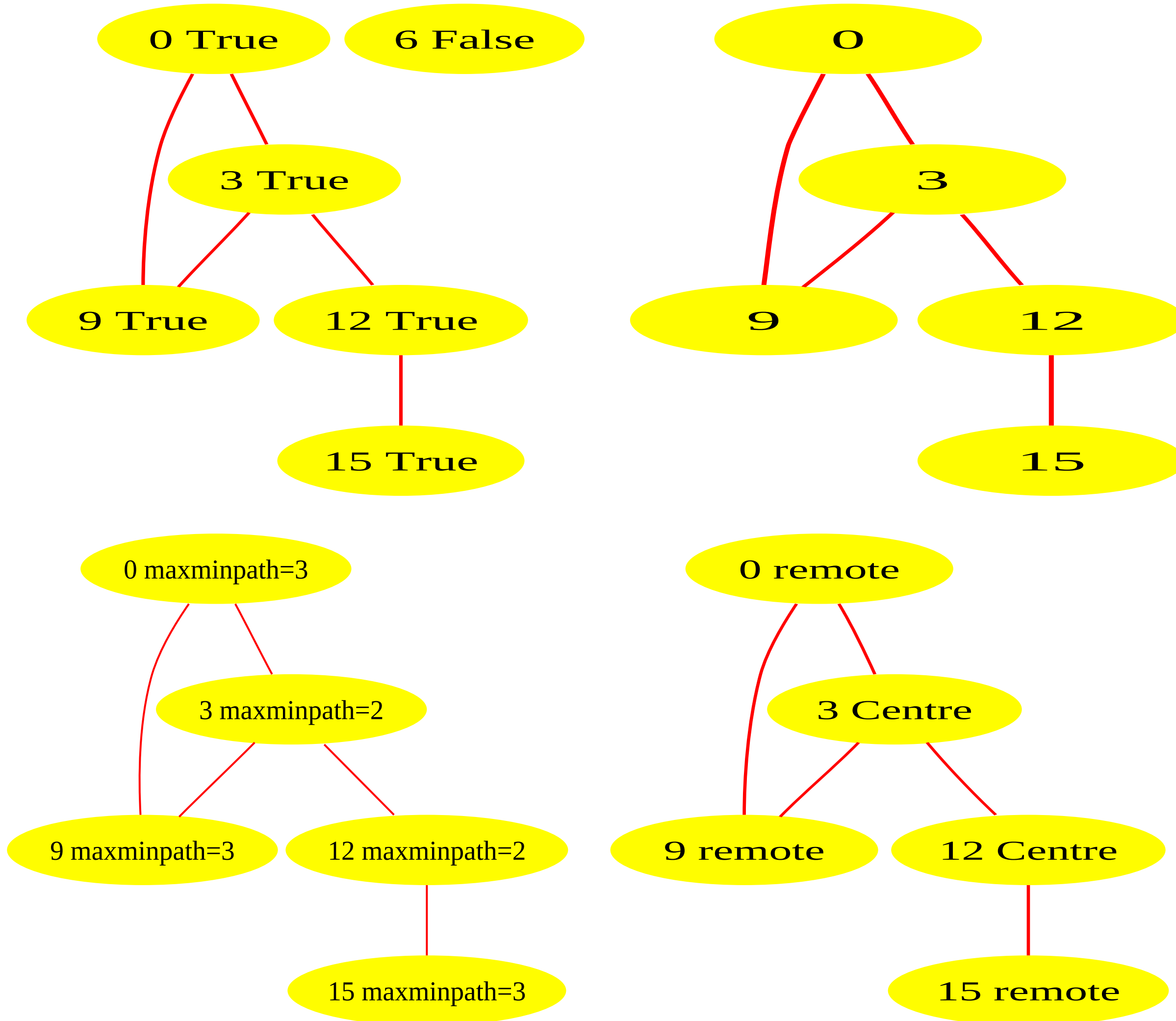
- $D = d(x, y) =$ distance matrix of graph G
- The eccentricity of a vertex x in G and the radius $\rho(G)$ are defined as $e(x) = \max_{y \in V} d(x, y)$ and $\rho(G) = \min_{x \in V} e(x)$
- The centre of G is the set

$$C(G) = \{ \, x \in V \mid e(x) = \rho(G) \, \}.$$

  $C(G)$ is the solution of the *emergency facility local problem*
- The status $d(x)$ of a vertex and the status $\sigma(G)$ of the graph G are defined as $d(x) = \sum_{y \in V} d(x, y)$ and $\sigma(G) = \min d(x)$
- The median is the solution of the *service facility location problem*. The median of G is the set

$$M(G) = \{ \, x \in V \mid d(x) = \sigma(G) \, \}$$

Graph centre - algorithm



top left: connectivity; top right: connected partition

bottom left: eccentricity; bottom right: centre

Electrical circuits: theory

- ⋆ Digraph G with r_k the resistance of edge k
- ⋆ Problem 1: translate graph topology (known only locally) to circuit equations
- ⋆ Problem 2: solve these equations
- ⋆ Apply to the circuit:
  - Kirchhoff's current law (KCL)
  - Kirchhoff's voltage law (KVL)
  - Ohm's Law (ΩL)
- ⋆ Let v be the voltage vector and i the current vector (in edge space)

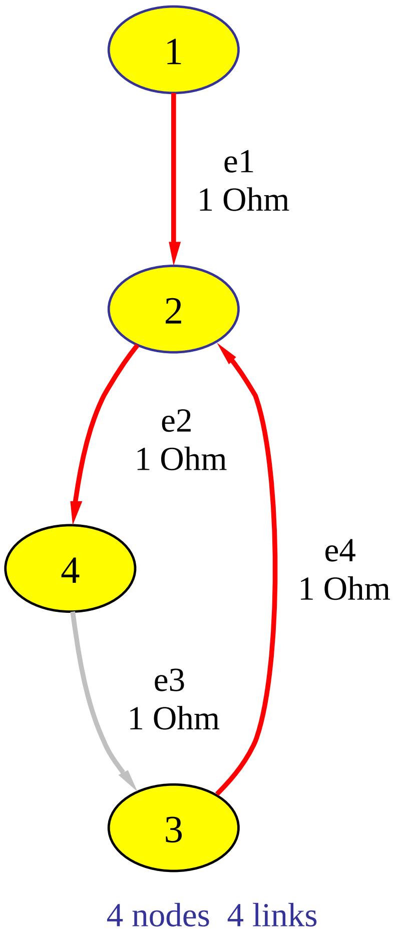
Electrical circuits: more theory

- A is the adjacency matrix and D is the degree matrix
- Find *incidence matrix* B from $BB^{\mathsf{T}} = L = D - A$
- Then KCL is $Bi = 0$
- Build a spanning tree T
- Edges in T are *branches*, other edges are *chords*
- Each chord has a *fundamental cycle* (FC)
- C: matrix with one column for each edge, with elements being the coefficients of the corresponding FC in the edge space (only chords are really needed)
- Then KVL is $C^{\mathsf{T}}v = 0$
- ΩL is $v_k = i_k r_k$

Electrical circuits: the solution!

- $i = Yv$, where Y is the *conductance matrix*
- $Y = -C\,C^{+R}$, $R = \text{diag}\,(r_1, r_2, \dots)$
- C^{+R} is the *weighted Moore-Penrose pseudo-inverse* of C with weight R. If $R = W^{\mathsf{T}}W$, then $C^{+R} = (WC)^{+}\,W^{\mathsf{T}^{-1}}$
- $C^{+R}CC = C$ and $(RCC^{+R})^T = RCC^{+R}$
- I have developed an algorithm for incremental computation of C^{+R}, which can be applied as the columns of C are found by remote nodes

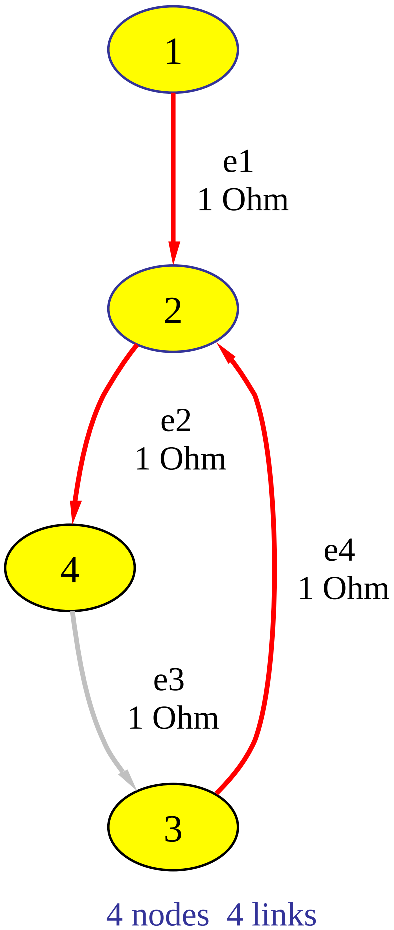
Electrical circuits: example



$$A = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{bmatrix}$$

$$B = \begin{bmatrix} 1 & 0 & 0 & 0 \\ -1 & 1 & 0 & -1 \\ 0 & 0 & -1 & 1 \\ 0 & -1 & 1 & 0 \end{bmatrix}$$

Electrical circuits: example continued



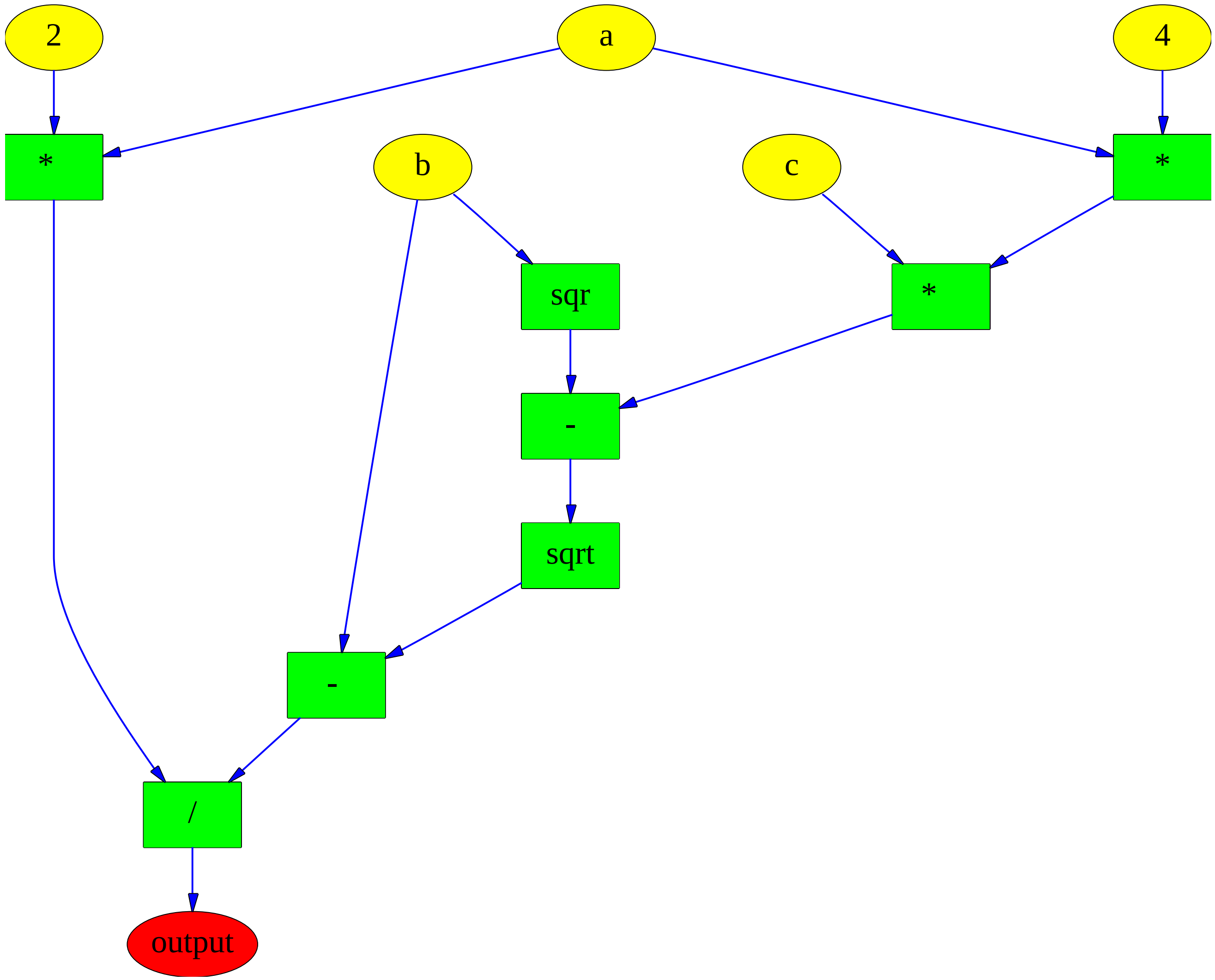
$$C = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

$$Y = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & -1/3 & -1/3 & -1/3 \\ 0 & -1/3 & -1/3 & -1/3 \\ 0 & -1/3 & -1/3 & -1/3 \end{bmatrix}$$

Distributed linear algebra

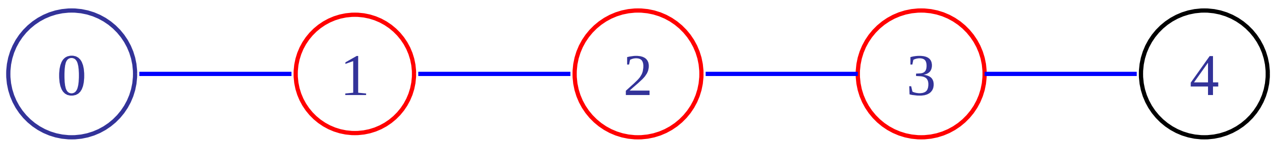
- Example problem: compute matrix A^{-1}, when elements A_{ij} are received in random order and at random times from distant nodes
- Assume messages are updates $A_{ij} \to A_{ij}+\alpha$ to an initially zero matrix
- Require A^{-1} to be correct at all times
- Generically A is singular at most times, but we can use the Moore-Penrose pseudo-inverse $A^\dagger$
  - $AA^\dagger A = A$
  - $A^\dagger AA^\dagger = A^\dagger$
  - $(AA^\dagger)^T = AA^\dagger$
  - $(A^\dagger A)^T = A^\dagger A$
- Update formulas are available which require storage only of the current A and $A^\dagger$
- cf. stream computing model

Exact real arithmetic example



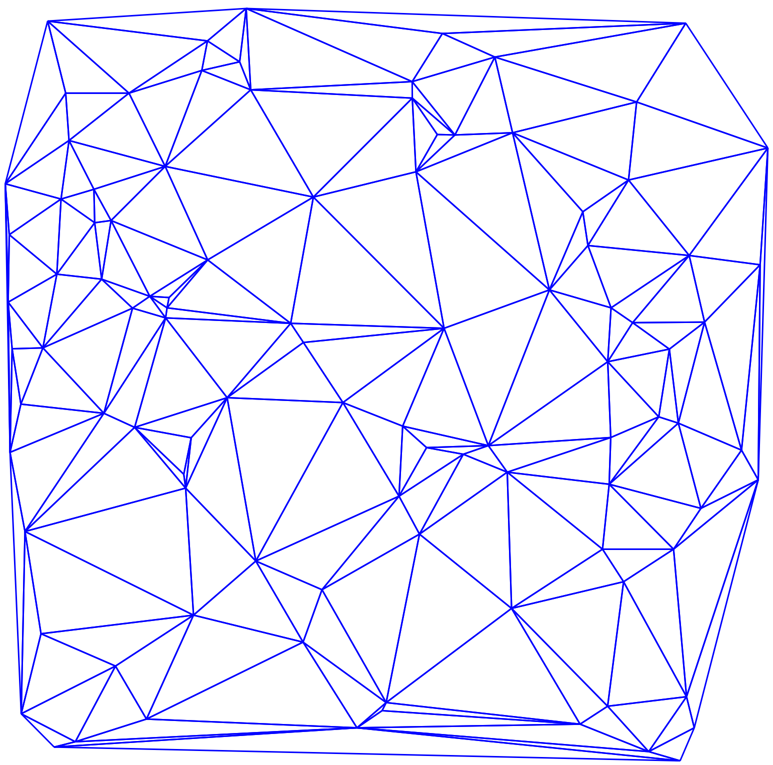
more.btexact.com/people/briggsk2/XR.html

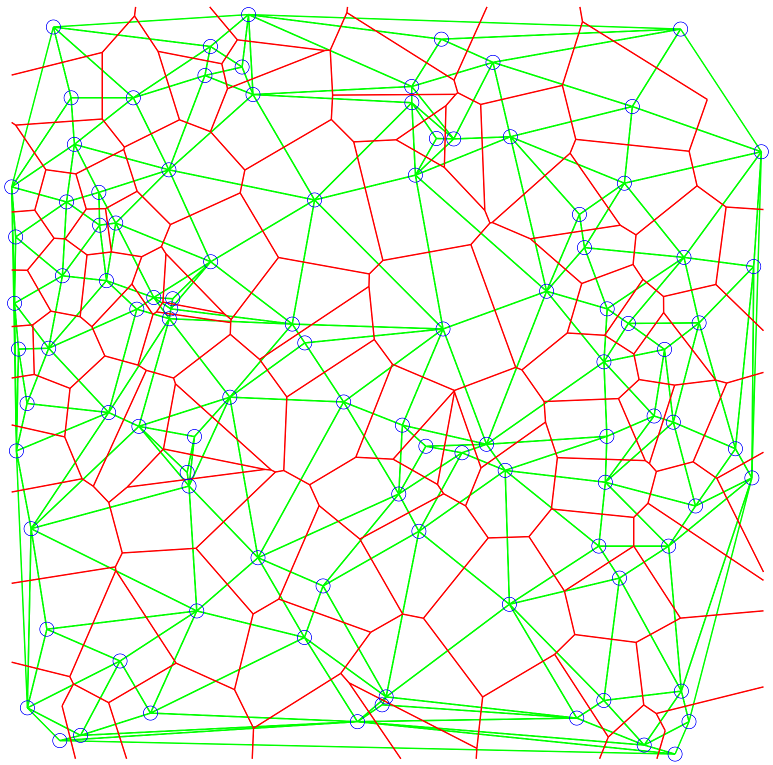
Dynamical processes on graphs

- coupled dynamical systems
- diffusion processes $\frac{d}{dt}\, u = L\, u$
- discrete Green's functions
- example
  - $\Gamma =$ 

  - $G(\Gamma) = \begin{bmatrix} 3 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 3 \end{bmatrix} /2$

Delaunay triangles





Can we compute this in a distributed fashion?

Ideas for future work

- ★ dynamic topology
- ★ characterize convergence rates
- ★ nontermination
- ★ computational complexity issues
- ★ distributed optimization
- ★ distributed control of network flows
- ★ . . .

References

N Lynch Distributed algorithms
ISBN 1558603484 Morgan Kauffman 1996

D P Bertsekas & J N Tsitsiklis Parallel and distributed computation: Numerical Methods
ISBN 1886529019 Athena Scientific 1997

G Tel Introduction to distributed algorithms
ISBN 0521794838 CUP 2000

V Barbosa An introduction to distributed algorithms
ISBN 0262024128 MIT Press 1996

sodium:/home/kbriggs/ada-bristol-talk/ada.tex