Multistage Process Models and Grid Computation

Michael C. Ferris
University of Wisconsin

joint with Christos Maravelias

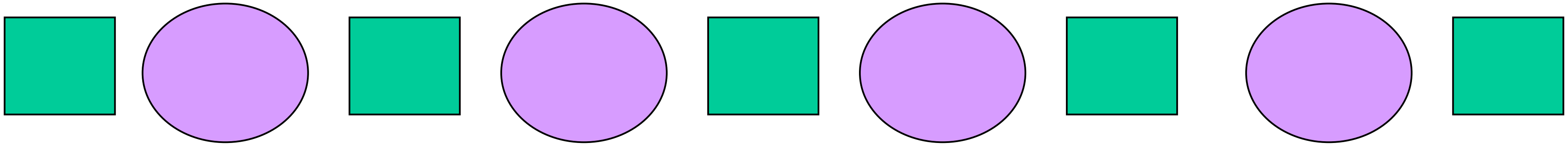
(AFOSR, NSF, AChS)

I'm depressed!

- Last year I told you about GAMS/grid on Condor
- Even better now!
  - Able to do directed runtime output switching
  - Simpler mechanisms for collecting jobs
- Condor is bigger and better
- Paper at www.cs.wisc.edu/~ferris

Typical Application for GAMS

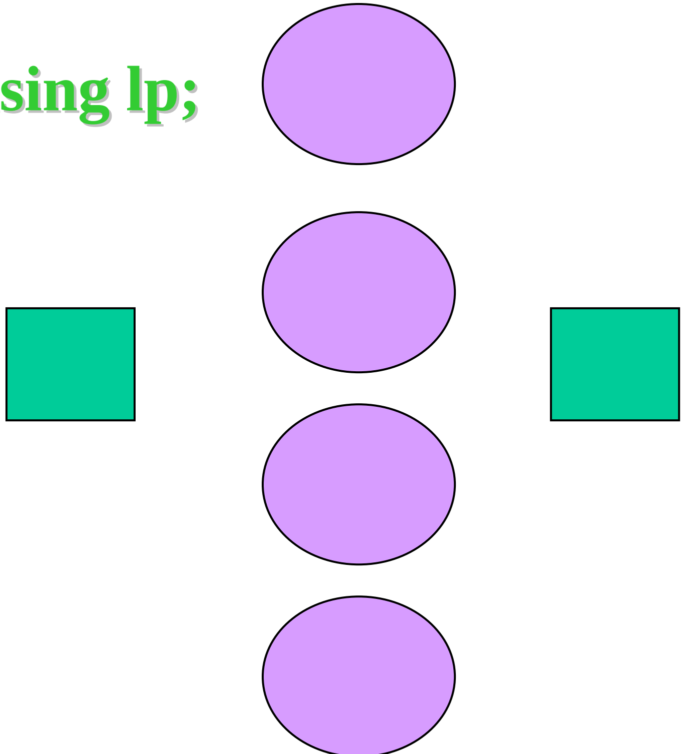
```
loop(s,
   b(j) = dem(s,j)
   solve transport min z using lp;
   report(s) = z.l;
) ;
```



Typical Application for GAMS

Need notion of a handle

```
loop(s,
   b(j) = dem(s,j)
   solve transport min z using lp;
   report(s) = z.l;
) ;
```



Typical Application for GAMS/grid

```
transport.solvelink = 3;          // turn on grid option
loop(s,
   b(j) = dem(s,j)
   solve transport min z using lp;
   h(s) = transport.handle );    // save instance handle

repeat
   loop(s$handlecollect(h(s)),
      report(s) = z.l;
      h(s) = 0 ) ;    // indicate that we have loaded the solution
   display$sleep(card(h)*0.2) 'was sleeping for some time';
until card(h) = 0 or timeelapsed > 10;
```

Why used only by my buddies?

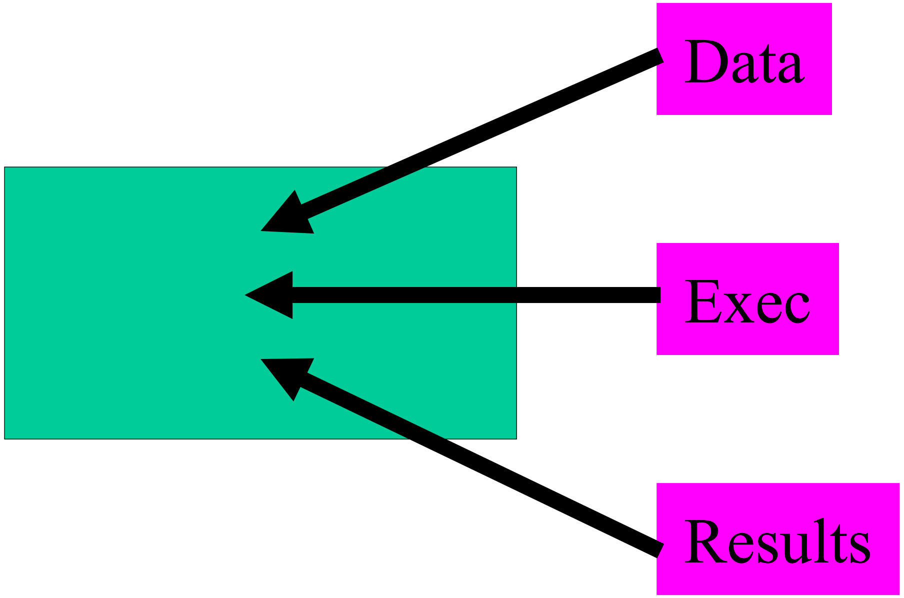
- Entry cost to parallel computing is high
  - Accounts at supercomputer site
  - Source code changes – debugging hard
  - Wait for 2 days for job to start
  - Install Condor
- Good news - diminishing – 4 proc laptops
  - No change at all to GAMS source
  - Can use already – relies on OS not grid tools
- Is this true of your parallel application?

Worker setup cost

- 'Free' for background process
- Easy on Sun-Grid since 'shared FS'
- Condor-Grid much larger, has no SFS
  - Worker set up installs GAMS
  - Design has 1 task per worker
- Good news – MW/GAMS
  - 1 worker, many tasks

Worker / task

- Local copy of gams needed
  - Zip file, job dir
  - Mimic environment
- Problem instance
- Start flag
- End flag
- Trigger file
  - Updates



Shortcomings

- Iterative schemes update small amount of model "data"
- As convergence occurs models become easier to solve (great start point)
- Model regeneration time is longer than solution time!
- Fix: use MW and gams_submit

Problems are hard

- Embarrassingly parallel applications are not transformative
- Naïve parallelism usually not effective
  - Hamming distance decomposition
  - Important variable decomposition
  - dumptree = 400 option better
  - B&B, LP & fix, Dantzig-Wolfe decomposition possible using GAMS/grid but not trivial
- Good news – domain knowledge critical

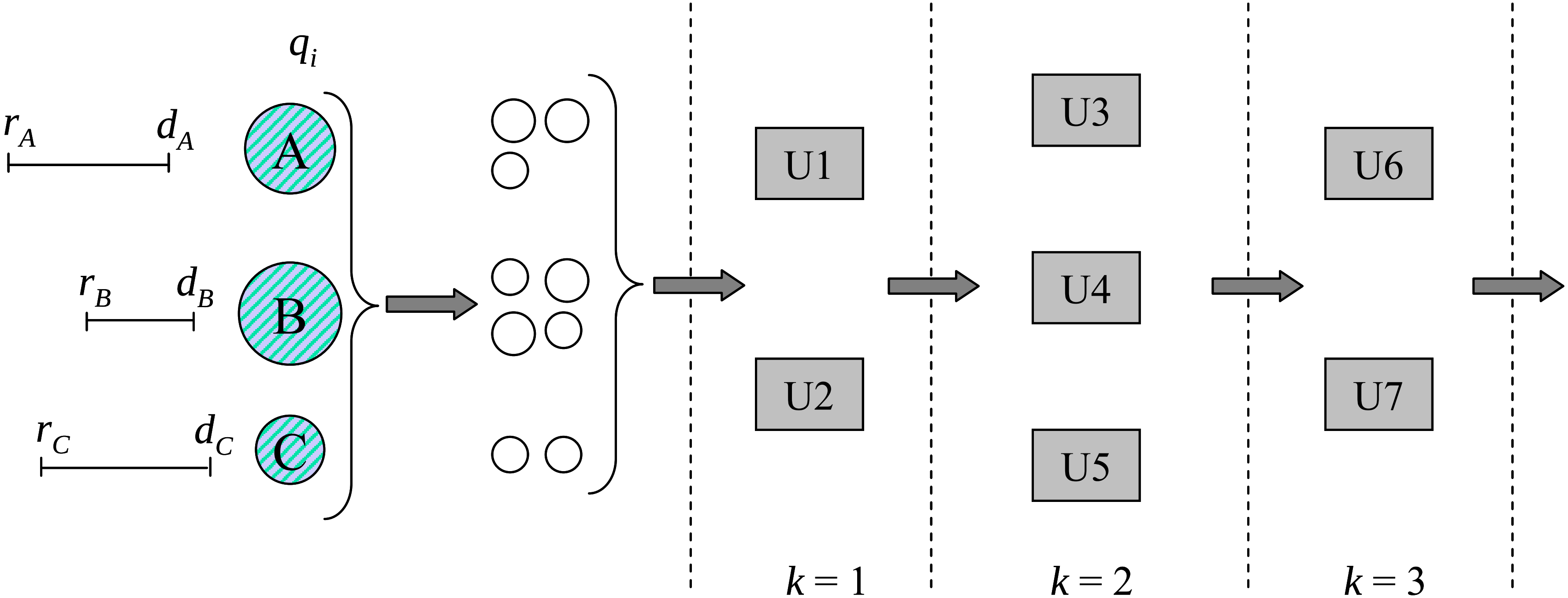
Batching and scheduling

Orders: $i \in I = \{A, B, C\}$

Batches

Stages: $k \in K = \{1, 2, 3\}$

Units: $j \in J = J(1) \cup J(2) \cup J(3)$



$k = 1$ $k = 2$ $k = 3$

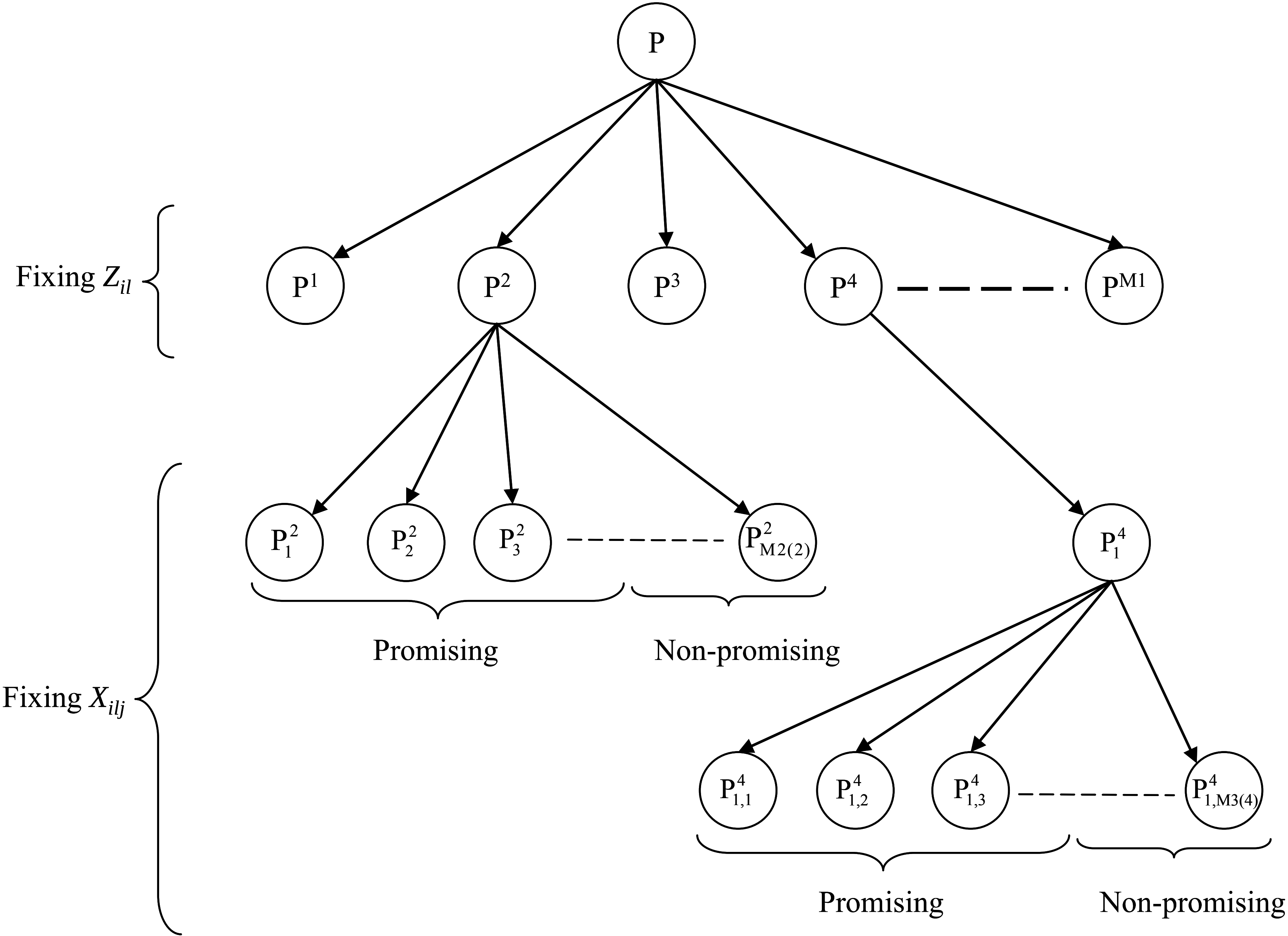
$J(1) = \{U1, U2\}$ $J(2) = \{U3, U4, U5\}$ $J(3) = \{U6, U7\}$

Heirarchical MIP

Determine

- the number and size of batches required to meet each order (batching decision),
- the assignment of batches to processing units at each stage,
- the sequencing of assigned batches in each processing unit,

in order to minimize the time necessary to meet all orders.


P
Fixing Z_{il}
P^{1}
P^{2}
P^{3}
P^{4}
P^{M1}
Fixing X_{ilj}
P^{2}_{1}
P^{2}_{2}
P^{2}_{3}
$P^{2}_{M2(2)}$
P^{4}_{1}
Promising
Non-promising
$P^{4}_{1,1}$
$P^{4}_{1,2}$
$P^{4}_{1,3}$
$P^{4}_{1,M3(4)}$
Promising
Non-promising

Results

- Models parameterized by q, nonstandard option file used for CPLEX
- Model 1: optimality proof in 17 secs
- Model 2:
  - CPLEX (9.0) fails to prove optimality in 2hrs
  - dumptree=400 optimality proof in 2 hrs, but 13 hrs of computing done
  - Interprocessor communication, "good heuristic", reduces optimality proof to 21 mins
  - Domain partitioning, optimality proof in 7.5 mins
  - CPLEX (10.2) optimality proof in 8 mins

Model 3

- CPLEX 10.2 fails after 2 hrs, …
- dumptree, dynamic repartitioning with 1 hr time limit – filled disk
- Domain partitioning (2 levels) followed by dumptree – 12 days CPU without lower bound update
- Domain partitioning (3 levels) followed by dumptree (1 hr) – 9 hrs wall clock time
- Domain partitioning (4 levels) – 12 hours wall clock time

Model 4

- Even harder
- Domain partitioning
  - 745 problems left at level 2
  - One subproblem partitioned into 28886
  - 29 left after 1 hr
- 12 hrs wall clock provided 126 CPU days
- Time-constrained application fails

Optimal Transmission Switching

- Change topology of electrical network
- How to choose optimally?
- Similar type MIP, similar solution strategy - 3 days wall clock time
- Application requires "overnight" turnaround
- "Time constrained" optimization (as opposed to "real-time") via grid

Conclusions

- Grid systems available (e.g. Condor, IBM, SUN)
- Grid computing convenient via simple language extensions to modeling languages
- Can experiment with coarse grain parallel approaches for solving difficult problems
- Exploiting underlying structure and model knowledge key for "larger, faster" solution
- Please use it!