Optimal Crane Scheduling

Samid Hoda, John Hooker
Latife Genc Kaya, Ben Peterson

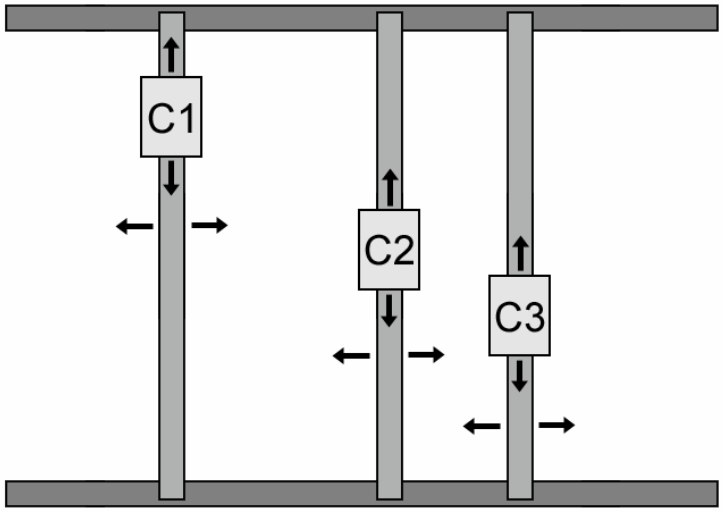
Carnegie Mellon University

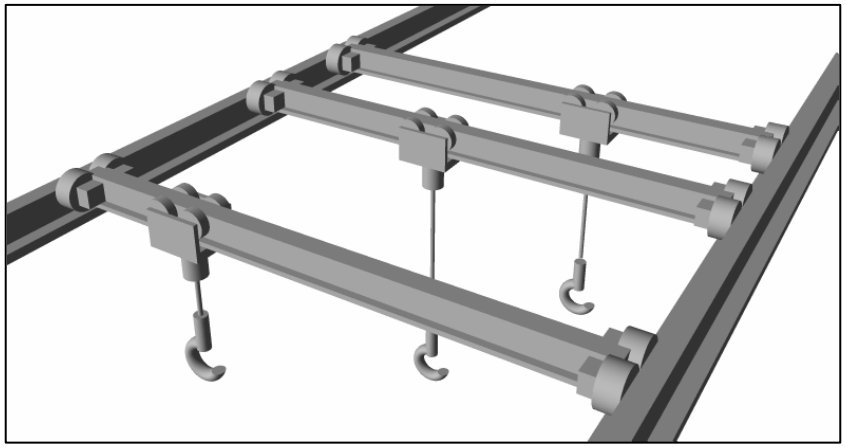
Iiro Harjunkoski

ABB Corporate Research

Problem

- Track-mounted cranes move materials & equipment.

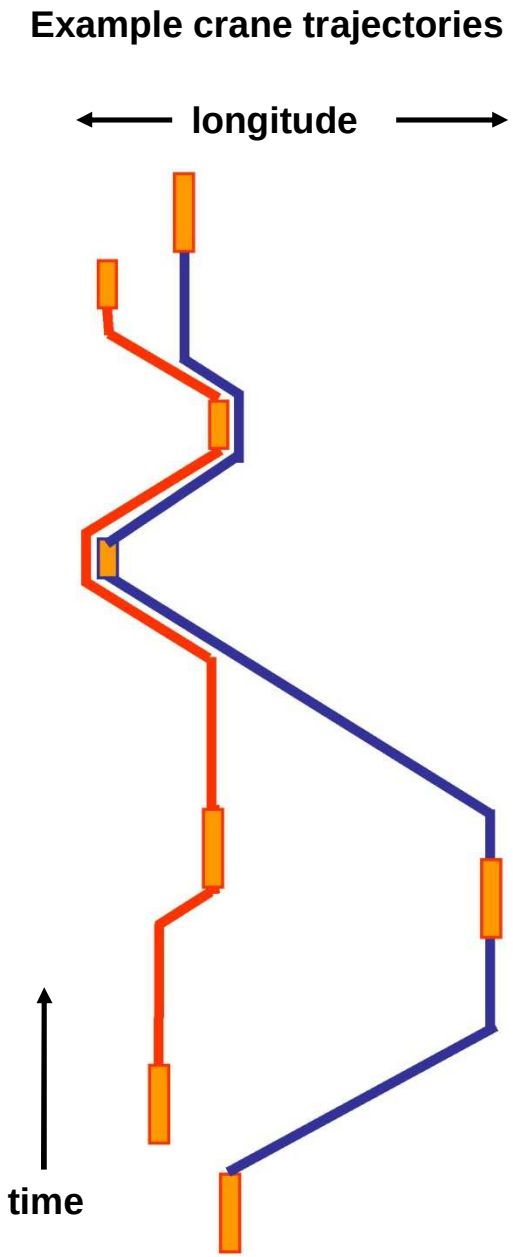




  - We focus on longitudinal moves (left-right).
  - Cranes must not cross paths.

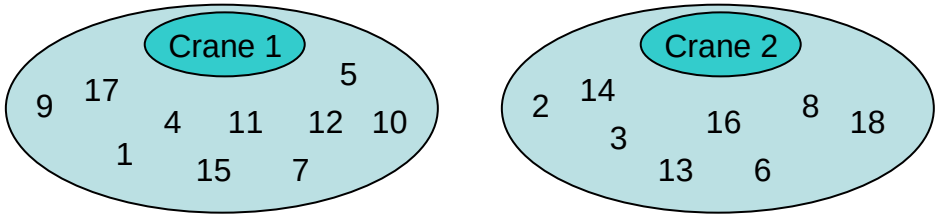
Problem

- Schedule cranes to transfer material between locations.
  - 50-300 jobs with 1-8 tasks each.
  - Origin/destination and stop times for each task.
  - Time windows and priorities.
  - Precedence constraints.
- Goal:
  - Follow production schedule as closely as possible.



Problem

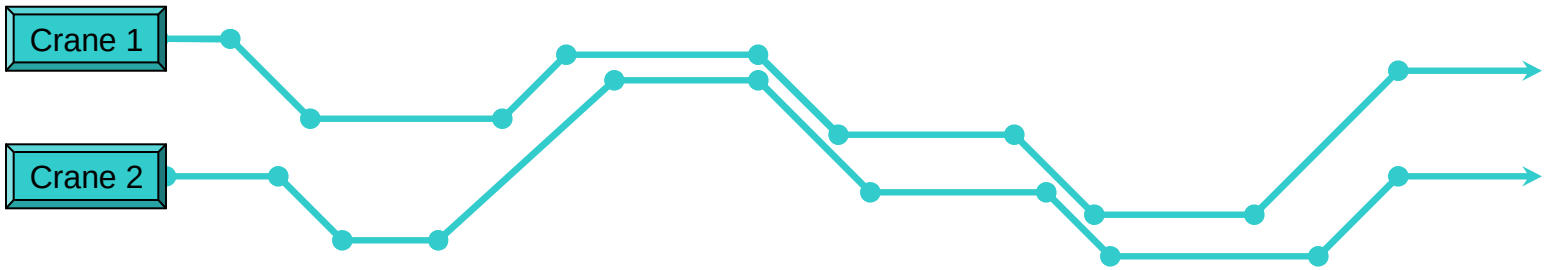
- Three problem levels.
  - Assign jobs to cranes.



  - Sequence jobs on each crane.

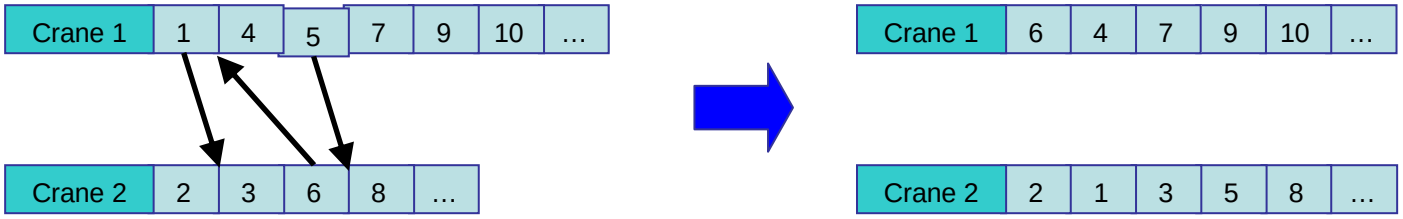
Crane 1	1	4	5	7	9	10	11	12	15	17	…
Crane 2	2	3	6	8	13	14	16	18	…		

  - Plan crane trajectories.
    - Time granularity ≈ 10 sec over ~24 hrs



Algorithm 1: Two-phase search

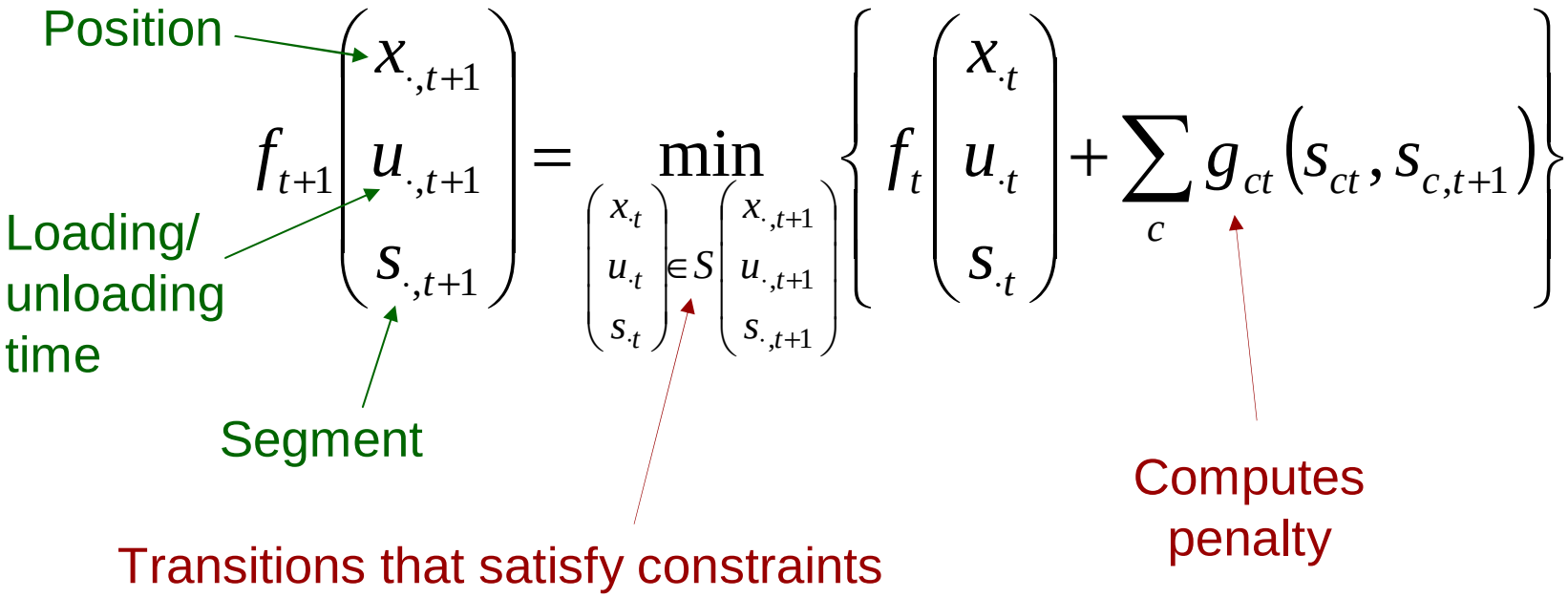
- For 2-crane problems.
- Phase 1
  - Assign and sequence with local search.



- Phase 2
  - Compute **optimal** trajectory with dynamic programming.

Algorithm 1: Two-phase search

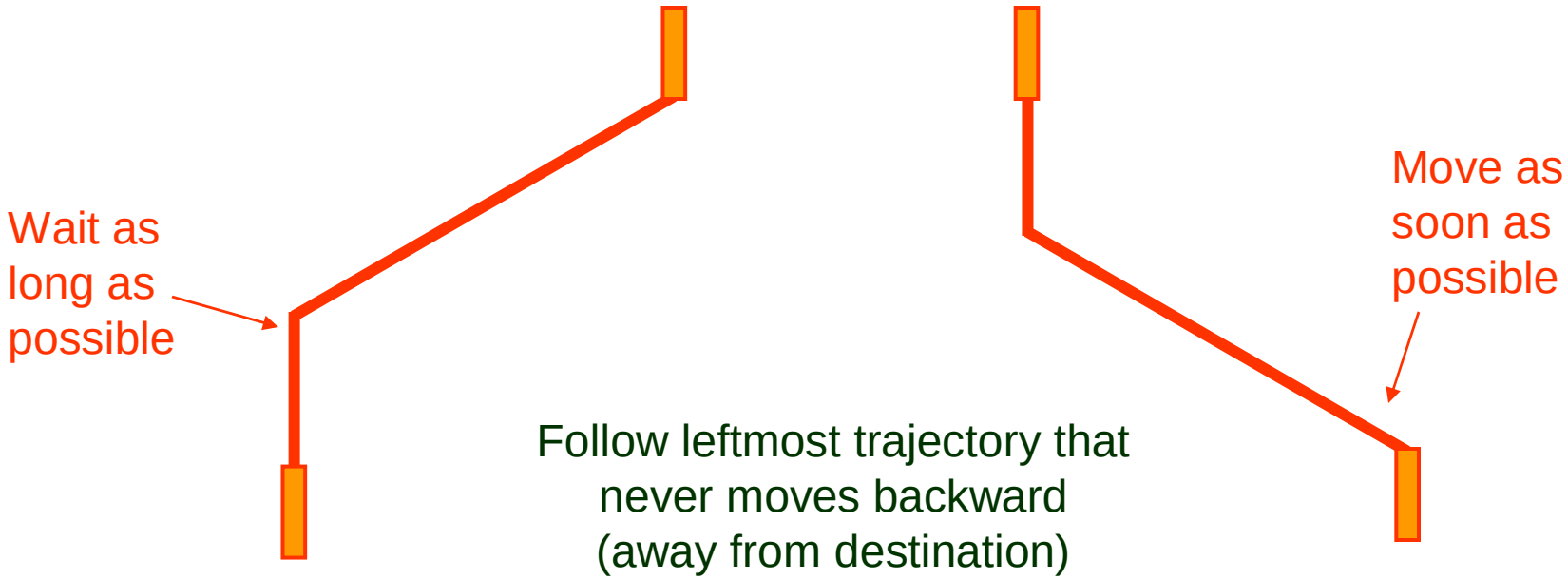
- Basic DP recursion:



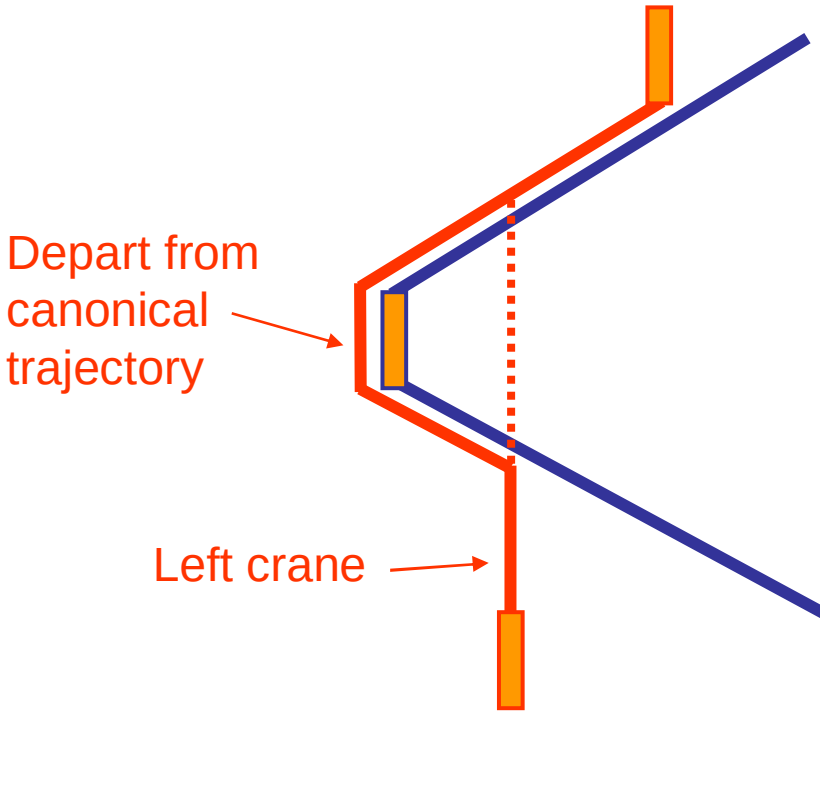
$$f_{t+1}\begin{pmatrix} x_{\cdot,t+1} \\ u_{\cdot,t+1} \\ s_{\cdot,t+1} \end{pmatrix} = \min_{\begin{pmatrix} x_{\cdot t} \\ u_{\cdot t} \\ s_{\cdot t} \end{pmatrix} \in S \begin{pmatrix} x_{\cdot,t+1} \\ u_{\cdot,t+1} \\ s_{\cdot,t+1} \end{pmatrix}} \left\{ f_t \begin{pmatrix} x_{\cdot t} \\ u_{\cdot t} \\ s_{\cdot t} \end{pmatrix} + \sum_c g_{ct}\left(s_{ct}, s_{c,t+1}\right) \right\}$$

State Space Reduction

- Only certain trajectories must be considered.
  - *Canonical* trajectory for the left crane:



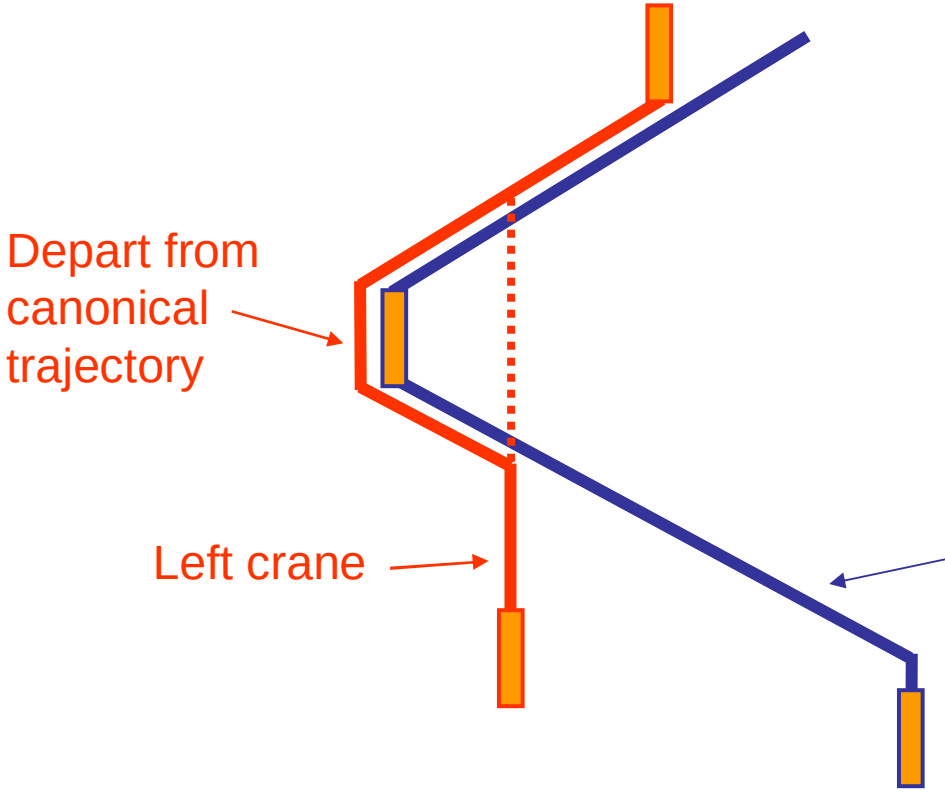
State Space Reduction



At each moment, follow canonical trajectory or right crane's trajectory, whichever is further to the left

State Space Reduction

– Properties of the minimal trajectory:



• The left crane never stops *en route* unless it is adjacent to the right crane.

• The left crane never moves backward unless it is adjacent to the right crane.

State Space Reduction

- **Theorem:** Some optimal pair of trajectories are minimal with respect to each other.

State Space Reduction

- Represent processing-time states as an **interval** of states.
  - Exploit the fact that cranes are processing (and therefore at rest) most of the time.
  - Store these states in a 2-dimensional circular queue data structure that persists through time.

State Space Reduction

Compute these costs when a task pair for which both tasks are at their stop locations appears in the state space.

c_{41}				
c_{31}				
c_{21}				
c_{11}	c_{12}	c_{13}	c_{14}	c_{15}

c_{ij} = cost-to-go when the left crane has been processing *i* time units and the right crane has been processing *j* time units.

State Space Reduction

Compute these costs in the next period. No need to update existing costs.

Data structure is a 2-dimensional circular queue.

c_{12}	c_{13}	c_{14}	c_{15}	c_{11}
c_{42}				c_{41}
c_{32}				c_{31}
c_{22}	c_{23}	c_{24}	c_{25}	c_{21}

c_{ij} = cost-to-go when the left crane has been processing i time units and the right crane has been processing j time units.

State Space Reduction

And similarly in the next period.

c_{23}	c_{24}	c_{25}	c_{21}	c_{22}
c_{13}	c_{14}	c_{15}	c_{11}	c_{12}
c_{43}			c_{41}	c_{42}
c_{33}	c_{34}	c_{35}	c_{31}	c_{32}

c_{ij} = cost-to-go when the left crane has been processing *i* time units and the right crane has been processing *j* time units.

State Space Reduction

These costs do not correspond to separate states.

After this point the table is no longer needed and memory can be released.

c_{34}	c_{35}	c_{31}	c_{32}	c_{33}
c_{24}	c_{25}	c_{21}	c_{22}	c_{23}
c_{14}	c_{15}	c_{11}	c_{12}	c_{13}
c_{44}	c_{45}	c_{41}	c_{42}	c_{43}

c_{ij} = cost-to-go when the left crane has been processing i time units and the right crane has been processing j time units.

Computational Results

- New data structure reduced computation time an order of magnitude.
  - Computation time is sensitive to width of time windows.
  - Can now solve 60-job problem with wide time windows (~1 hour)
    - Wide windows are necessary for feasible solution.

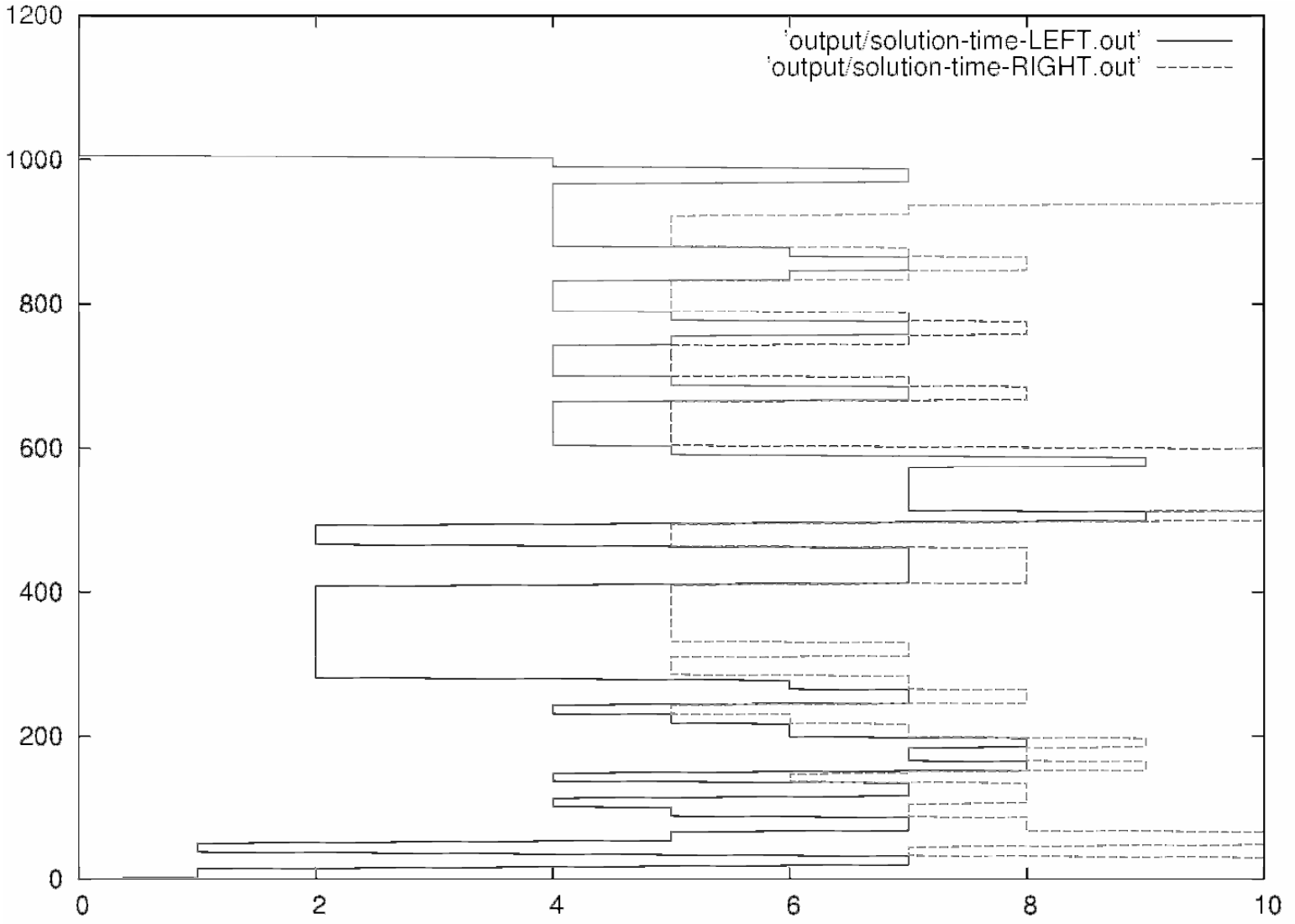
ABB

PJM

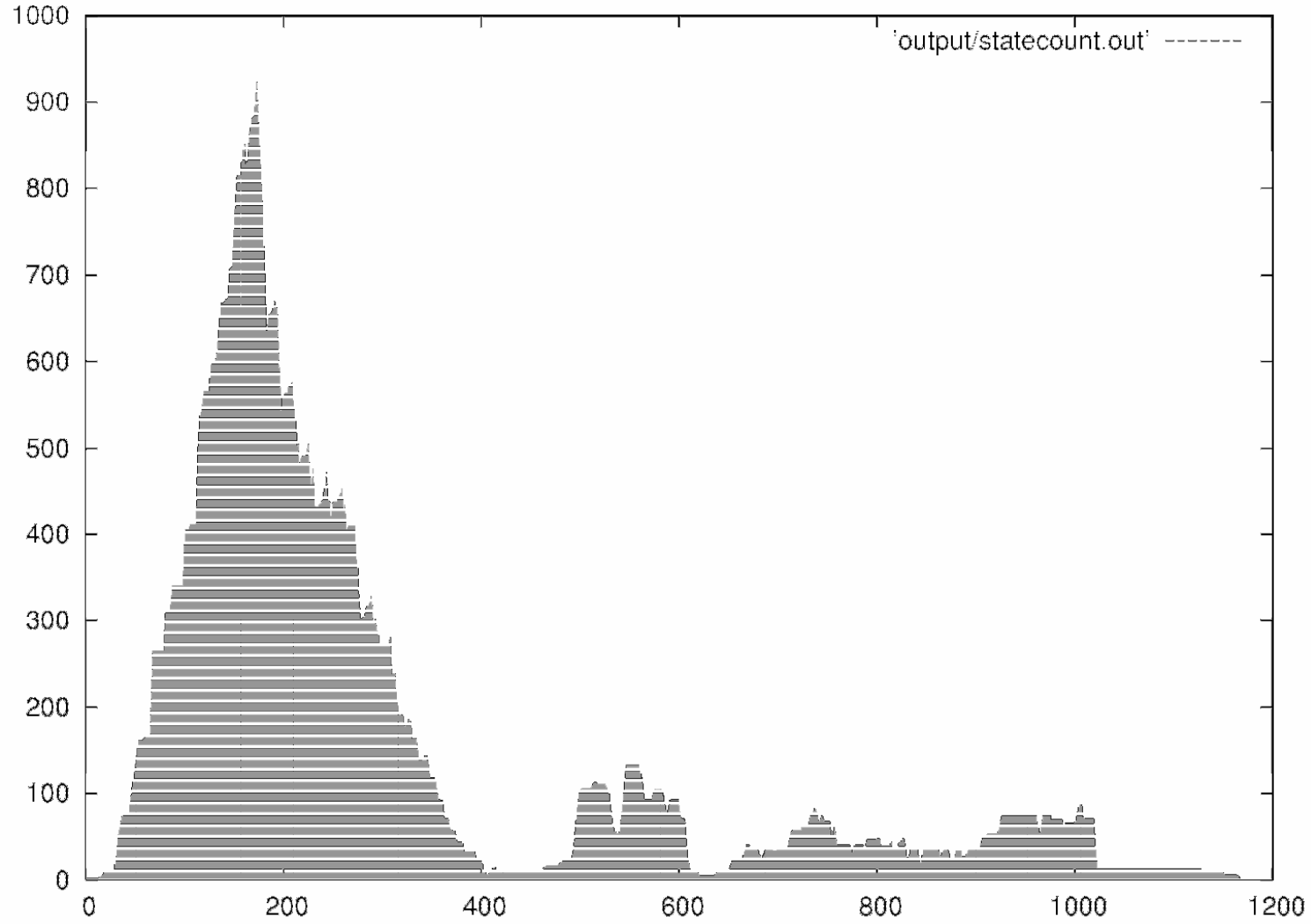
Effect of New Data Structure

Jobs	Time window (min)	Avg # states		Peak # states		Time (s)	
		Old	New	Old	New	Old	New
10	25	3,224	139	9,477	465	158	20
20	35	3,200	144	22,204	927	826	86
30	35	3,204	216	22,204	940	1,438	150

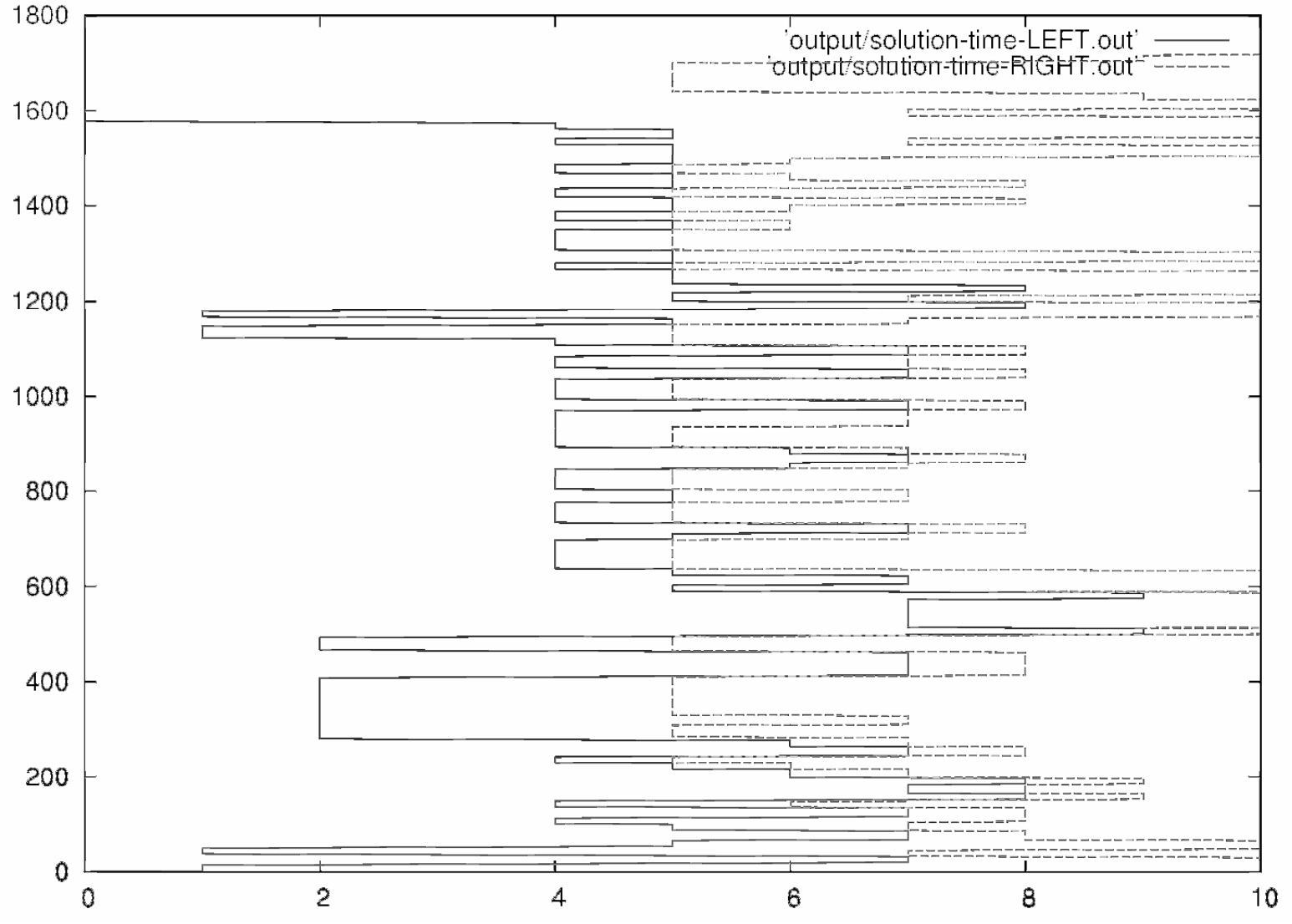
Solution of 20-job Problem



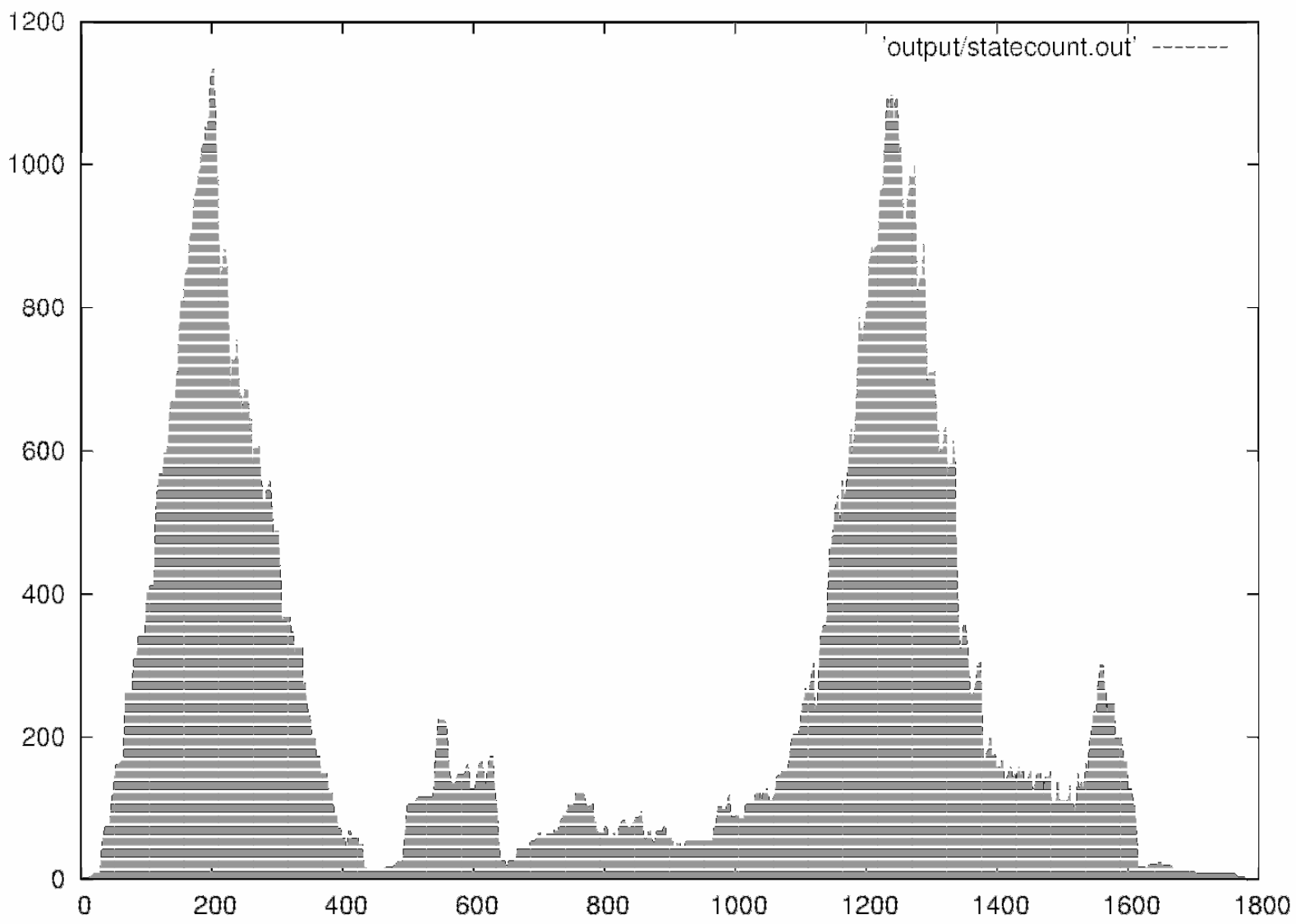
State Space Size



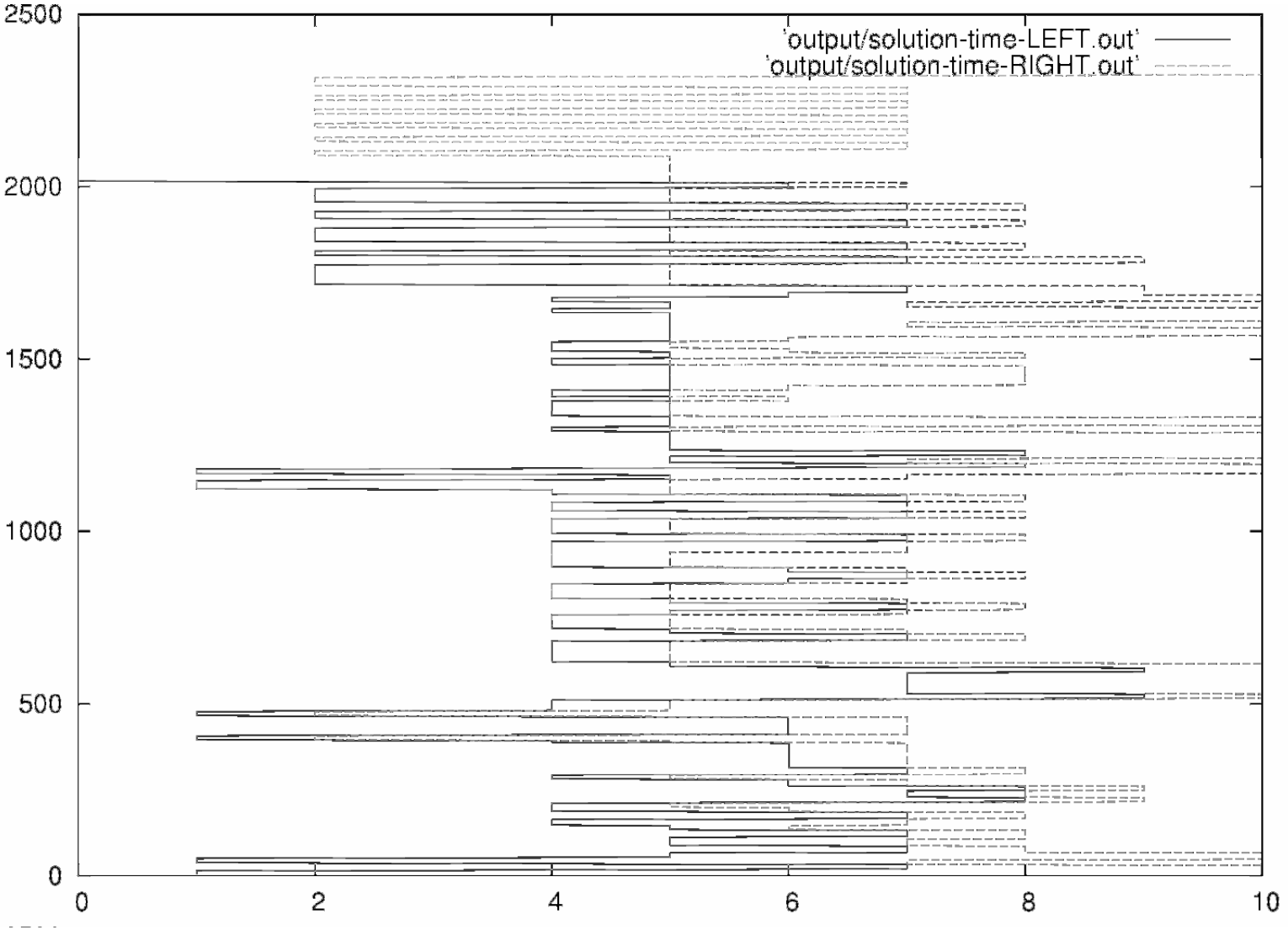
Solution of 40-job Problem



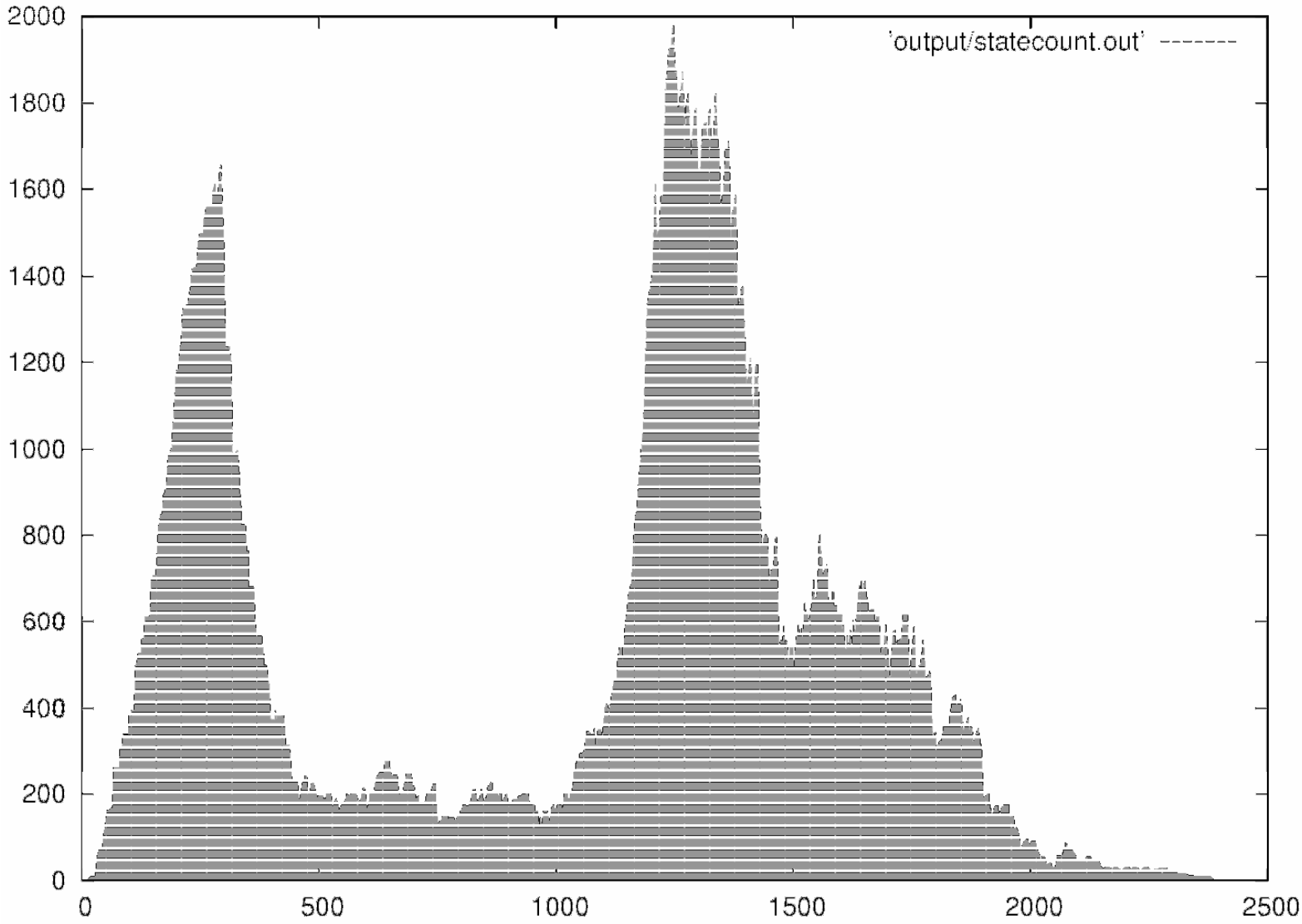
State Space Size



Solution of 60-job Problem



State Space Size

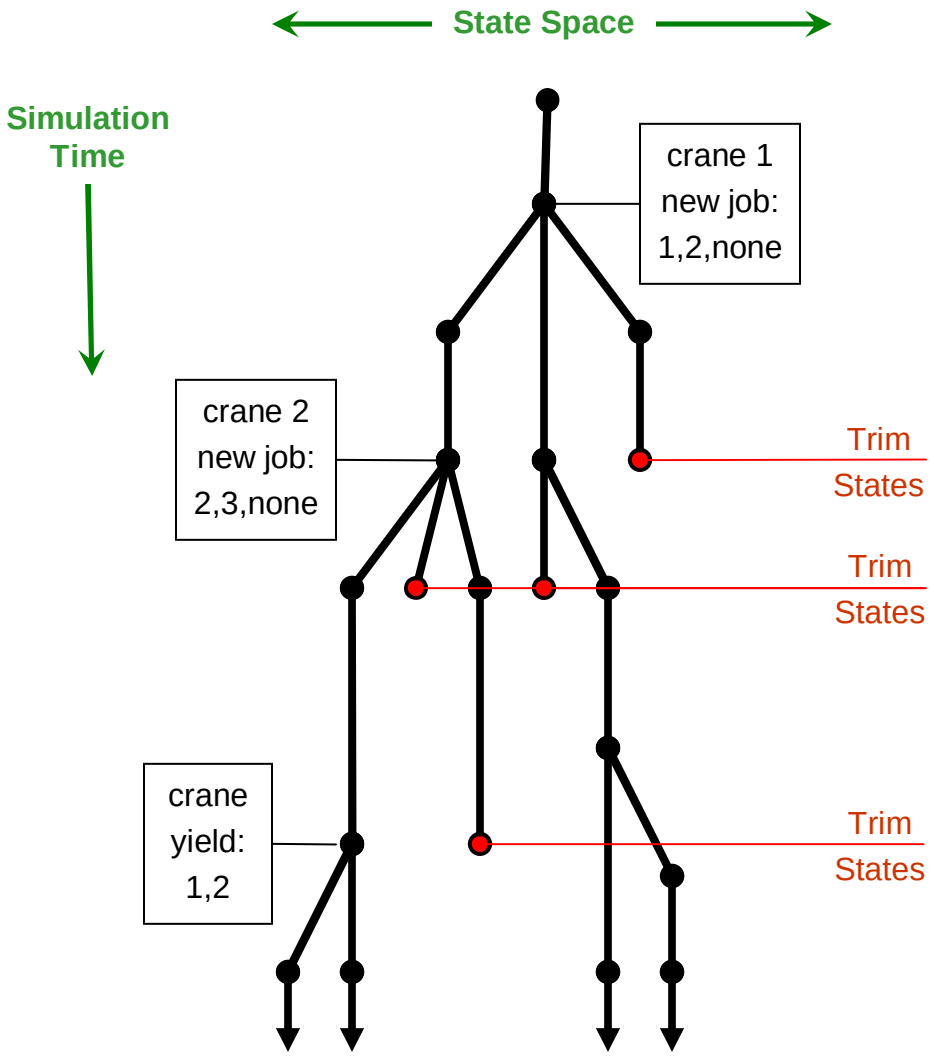


Computation Time for one DP

Jobs	Time window (min)	Computation Time per DP (s)
10	40	6.8
20	40	7.5
30	40	15.8
40	40	16.7
50	40	18.8
60	55	48.1

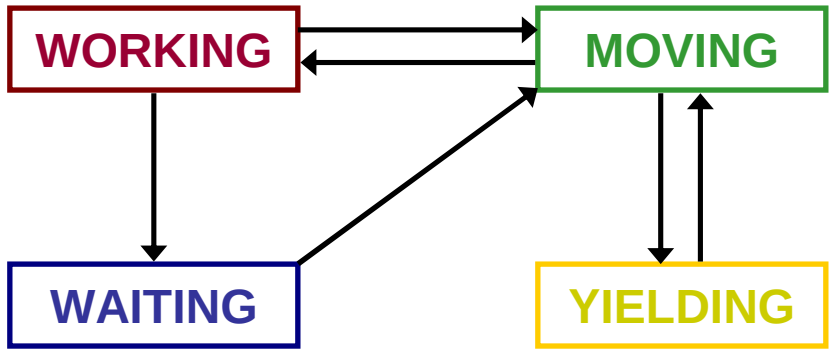
Algorithm 2: Inexact DP

- Crane simulation.
  - Splits at decision points.
- State trimming:
  - Eliminates inferior states
- Applies to > 2 cranes.



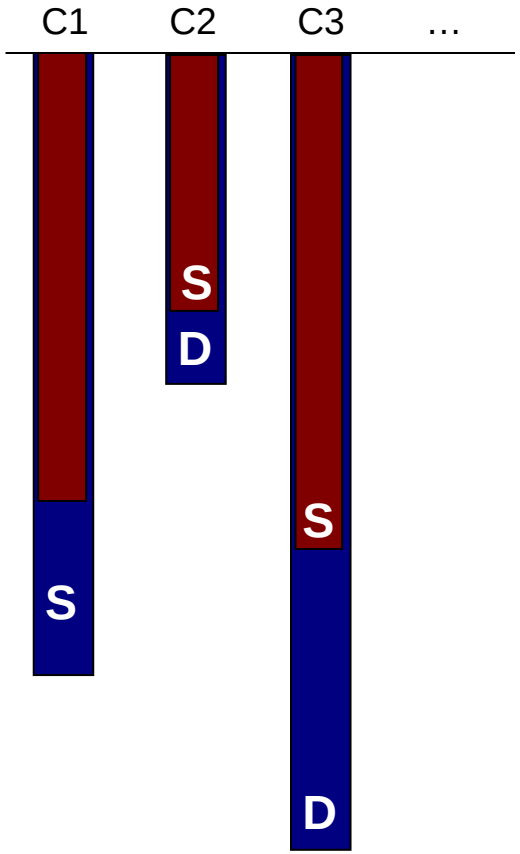
Algorithm 2: Inexact DP

- Discrete-event simulation
  - Continuous time
  - 4 crane actions
  - State transition only when an action is completed
- Cranes will work and move until:
  - Time to pick up new job.
    - Path splits for each job choice.
  - Cranes interfere.
    - Path splits for each yielding choice.



Algorithm 2: Inexact DP

- Exact state trimming by domination.
  - State *D* dominates state *S* if
    - Obj fcn value of *D* ≤ Obj fcn value of *S*
    - {jobs completed in *D*} ⊂ {jobs completed in *S*}
    - For all cranes *c*,
      - *c* working on same job in *D* & *S* or *c* is waiting in *D*, and
      - progress on *c*'s job in *D* ≥ progress on *c*'s job in *S*
- Inexact state trimming by size limit
  - Drop states with higher obj fcn values

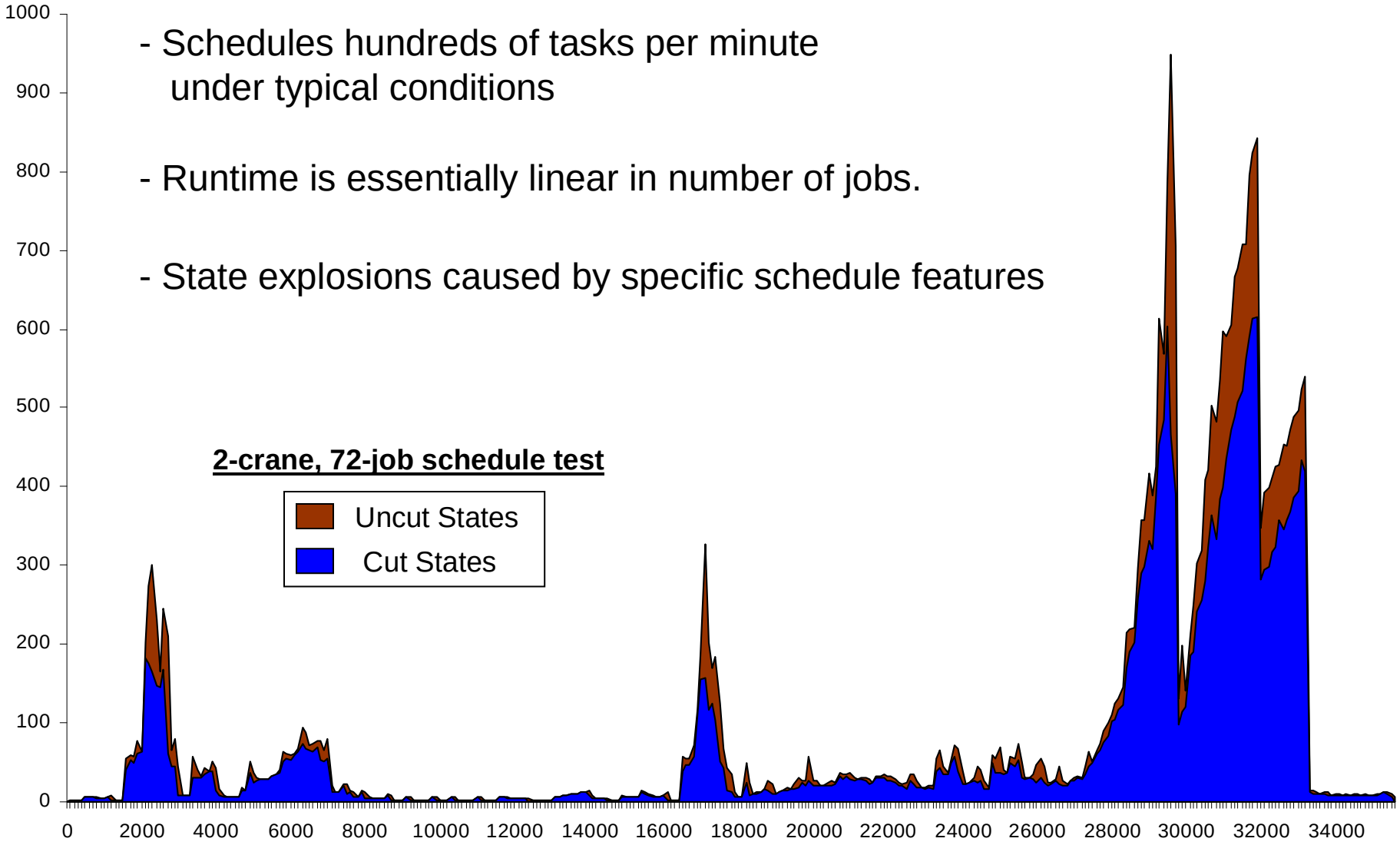


Computational Results

- Schedules hundreds of tasks per minute
  under typical conditions

- Runtime is essentially linear in number of jobs.

- State explosions caused by specific schedule features



Computational Results

- Inexact DP algorithm:
  - Meets runtime goals.
  - Can accommodate multiples cranes.
- State space size:
  - Still sensitive to time window width.
  - Inexact pruning required for ~1 hour windows.
- Unfinished business:
  - Compare solution quality with Algorithm 1.
  - Solve additional problems.