Crane Scheduling for ABB: Progress Report

Ionuț Aron
Latife Genç
John Hooker

March 2006

Problem

- Schedule 2 cranes to transfer material between locations in a copper processing facility.
  - Some 300 jobs, each with a time window and priority.
  - Precedence relations between jobs.
  - A job may require several stops.
  - Cranes run on the same track.
- Objective: minimize total penalty
  - Penalties reflect deviation from desired start times or completion times.

Problem

- Three problems in one:
  - Assign jobs to cranes.
  - Find ordering of jobs on each crane.
  - Find space-time trajectory of each crane.
    - Crane scheduling problems are coupled since the cranes must not cross one another.

Two-phase Algorithm

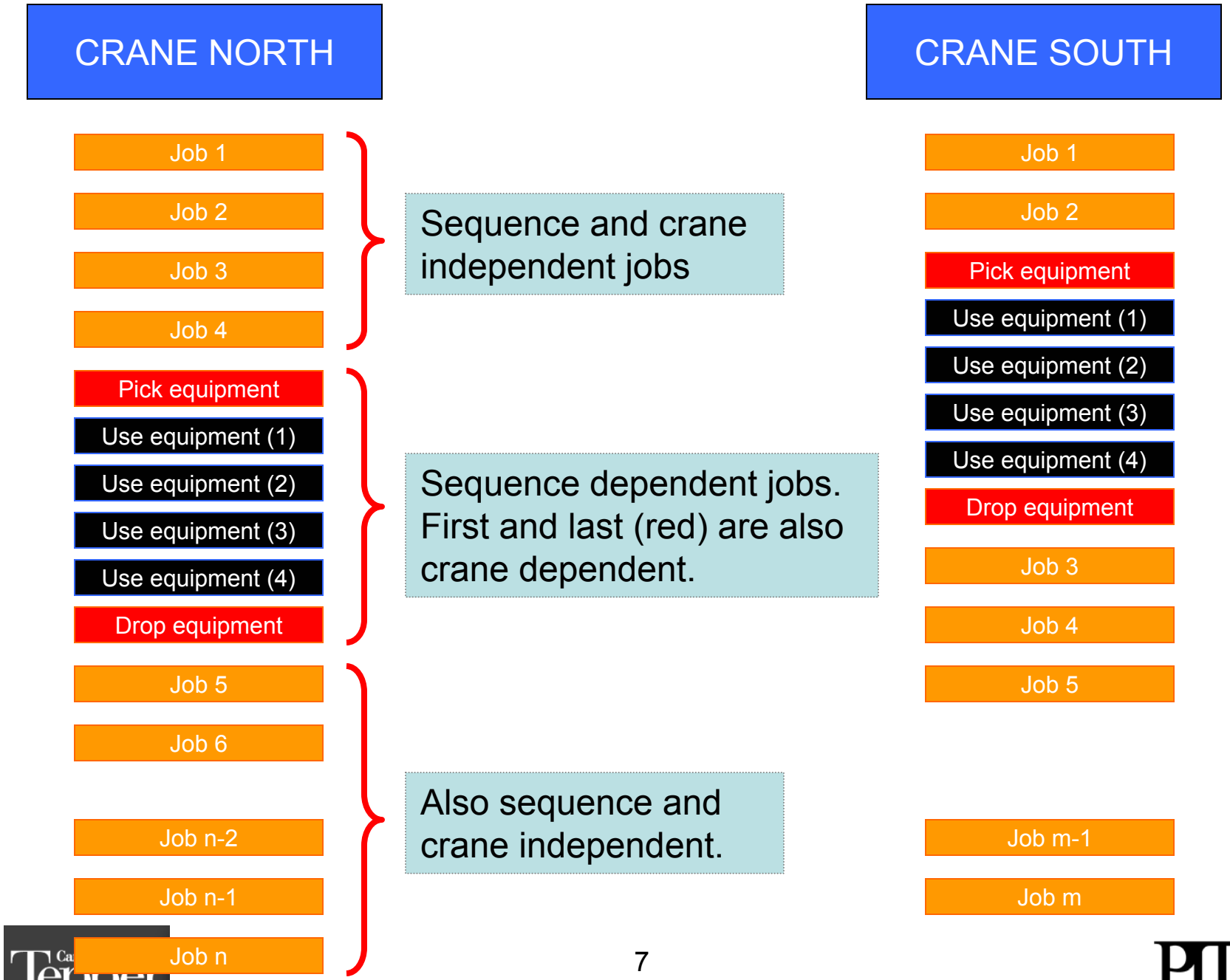
- Phase 1: Local search
  - Assign jobs to cranes
  - Sequence jobs on each crane
  - Solve simultaneously by tabu-like local search.
- Phase 2: Dynamic programming (DP)
  - Find optimal space-time trajectory for the cranes.
  - Solve for the two cranes simultaneously.

Background

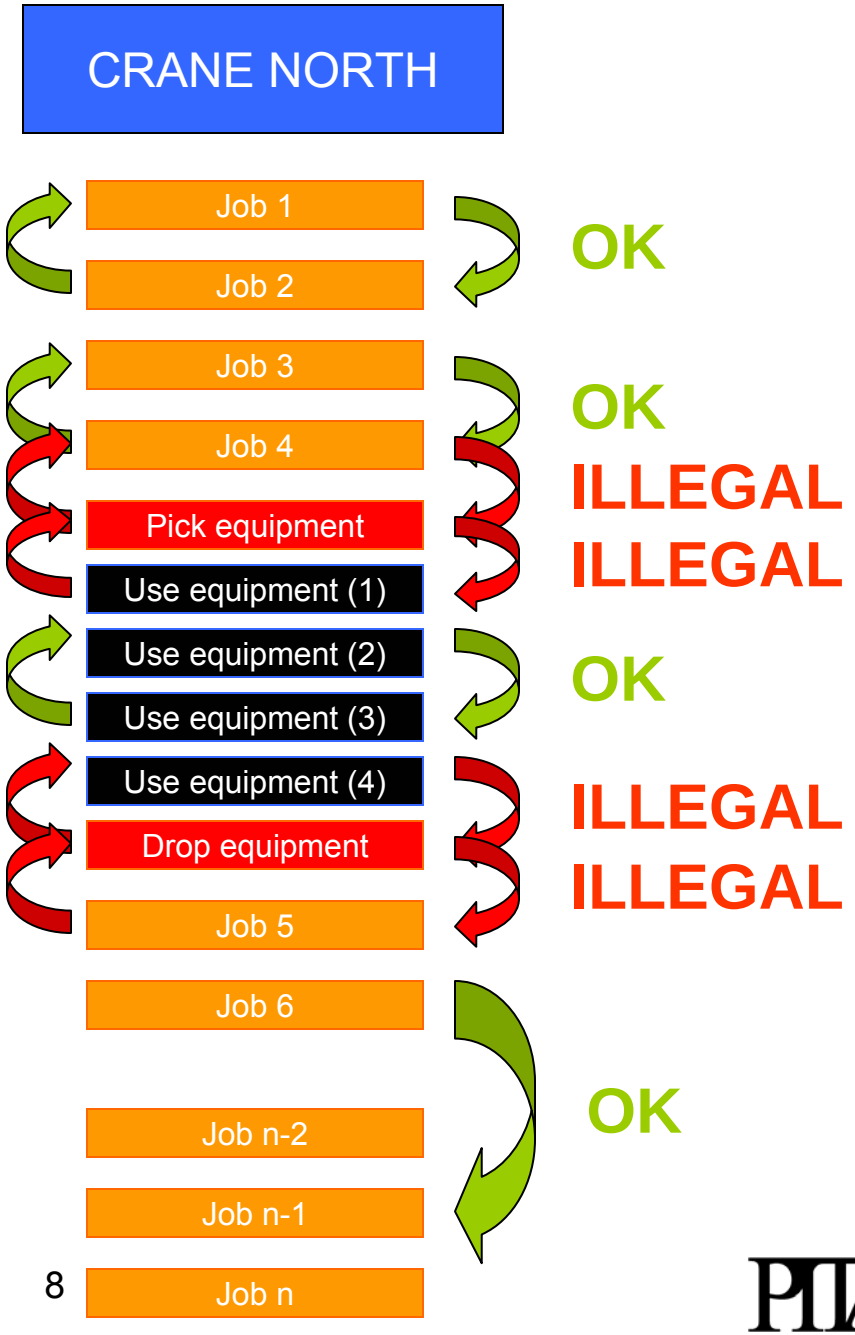
- Jobs must be assigned to 2 cranes s.t:
  - respect precedence rules
  - respect deadlines
  - ignore feasibility of assignments
    - i.e. one or both cranes may fail to execute the assigned jobs
    - feasibility is later ensured by dynamic programming
- Solution method: local search

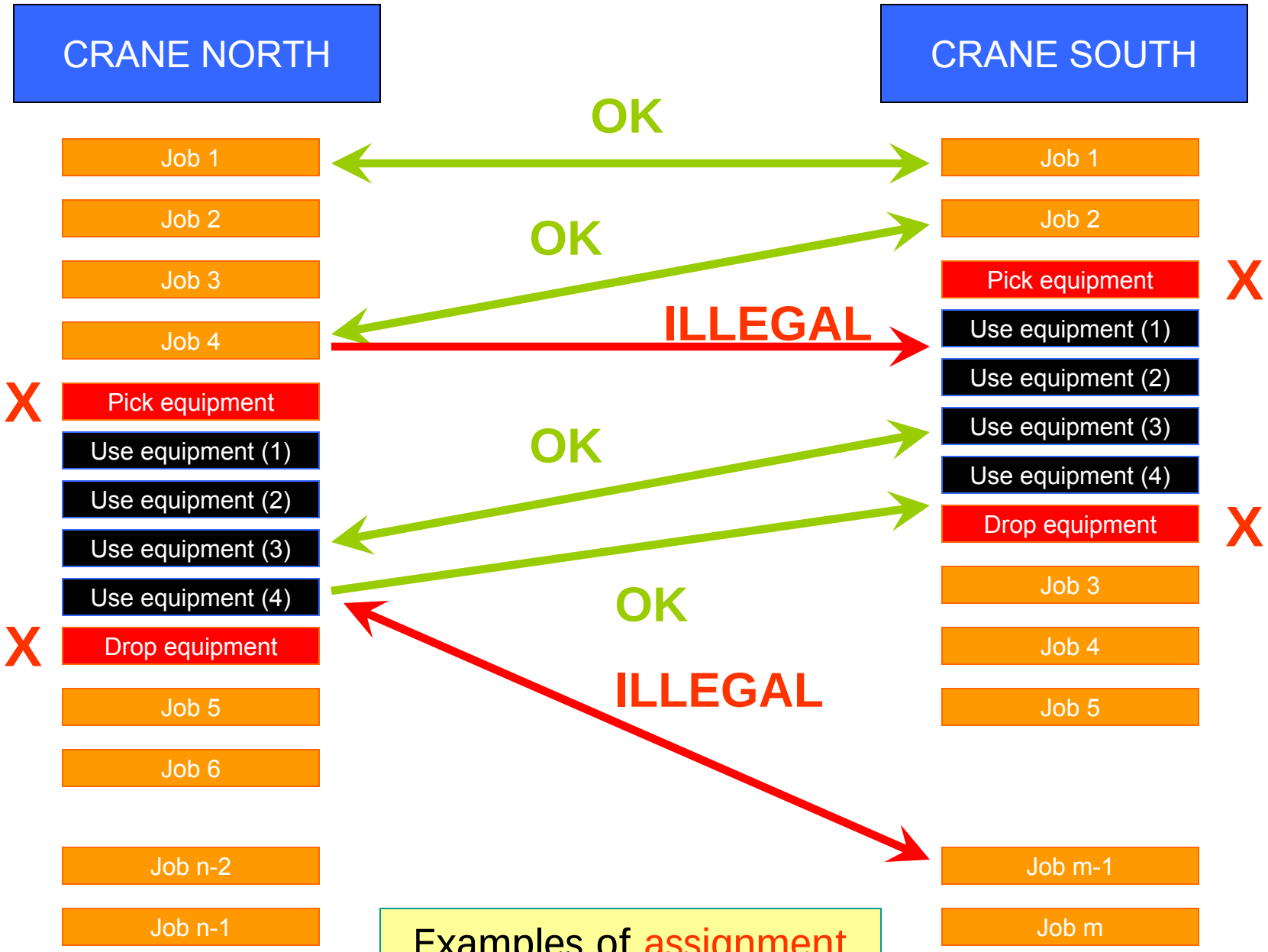
Local Search

- Neighborhood is defined by two types of moves.
  - Change assignment
    - Move a job to the other crane
    - Swap two jobs between cranes.
  - Change sequence
    - Move a job to a different position.
    - Swap two jobs.
- Evaluation of moves
  - Approximate evaluation function used for most moves.
  - Check best moves with DP (phase 2).

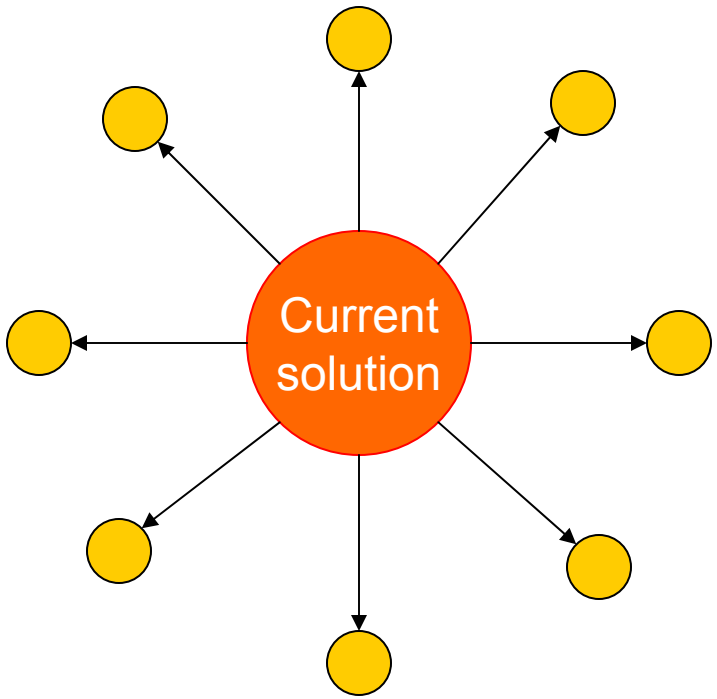

CRANE NORTH
CRANE SOUTH
Job 1
Job 2
Job 3
Job 4
Sequence and crane independent jobs
Pick equipment
Use equipment (1)
Use equipment (2)
Use equipment (3)
Use equipment (4)
Drop equipment
Sequence dependent jobs. First and last (red) are also crane dependent.
Job 5
Job 6
Job n-2
Job n-1
Job n
Also sequence and crane independent.
Job 1
Job 2
Pick equipment
Use equipment (1)
Use equipment (2)
Use equipment (3)
Use equipment (4)
Drop equipment
Job 3
Job 4
Job 5
Job m-1
Job m

Examples of sequencing moves (on the same crane)




CRANE NORTH
CRANE SOUTH
OK
Job 1
Job 1
OK
Job 2
Job 2
Job 3
Pick equipment
X
ILLEGAL
Job 4
Use equipment (1)
Use equipment (2)
X
Pick equipment
OK
Use equipment (3)
Use equipment (1)
Use equipment (4)
Use equipment (2)
Drop equipment
X
Use equipment (3)
Job 3
OK
Use equipment (4)
X
Drop equipment
Job 4
ILLEGAL
Job 5
Job 5
Job 6
Job n-2
Job m-1
Job n-1
Job m
Job n
Examples of assignment moves (between cranes)

Algorithm



1. Start with a current assignment and sequence for each crane

2. Generate all possible "neighboring" assignments and sequences as described by the rules for local moves

3. Perform a quick evaluation of the objective function for each of these neighbors

4. For the most promising neighbors, call the dynamic programming module (expensive!) to determine whether the assignment is actually feasible!

Dynamic Programming

- Find optimal space-time trajectory for each crane.
  - Sequence of jobs on each crane is given.
  - Cranes may not cross.
  - Minimize sum of penalties.

Dynamic Programming

Dynamic Programming

- Each job consists of one or more “segments.”
  - Order of segments within a job is fixed.
- Each segment consists of loading, movement to another position, unloading.
- Given for each segment:
  - Loading and unloading positions.
  - Time required to load, unload.
  - Min time for crane movement.

Dynamic Programming

- Main issue: state space explosion.
- State variables:
  - Position of cranes 1 and 2 on track.
  - How long each crane has been loading/unloading.
  - Current segment in process for each crane
    - Negative number if on the way to load the segment.
    - Positive number if loading, unloading, or on the way to unload.

Dynamic Programming

- State space reduction:

DP Algorithm

- discrete time t: 0 – N (final time)
- initial conditions:
  - time and cost=0
  - cranes are at terminal positions
  - crane job state s_i=-1
- Iterate:
  - find feasible states for time t+1
  - update cost for each feasible state: record completion time when last segment of a job is delivered
- stop at time t=N