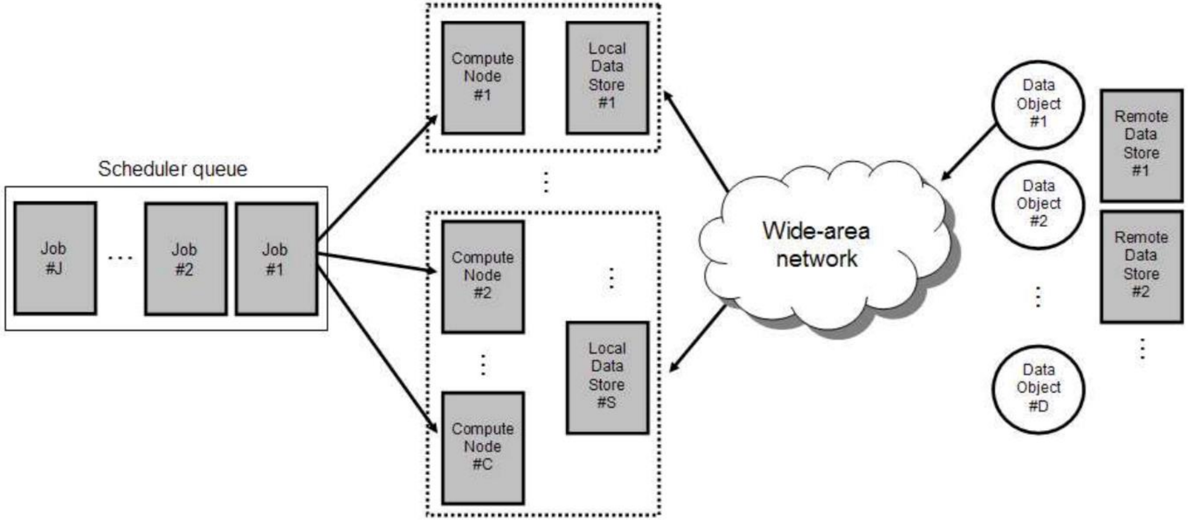
REDWOOD Job Scheduling Optimization

Oct. 2nd, 2024

Shengyu Feng, Jaehyung Kim (CMU)

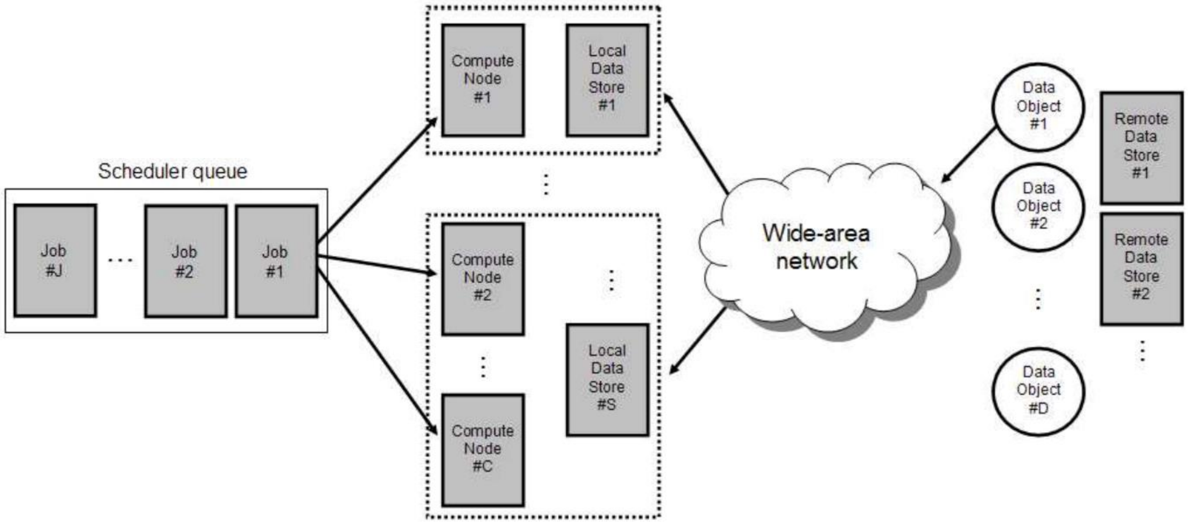
Problem Setup

- **Goal**: minimizing makespan (*i.e.*, total time to finish all jobs)
  - <u>Two terms</u>: (1) computing time & (2) data downloading time



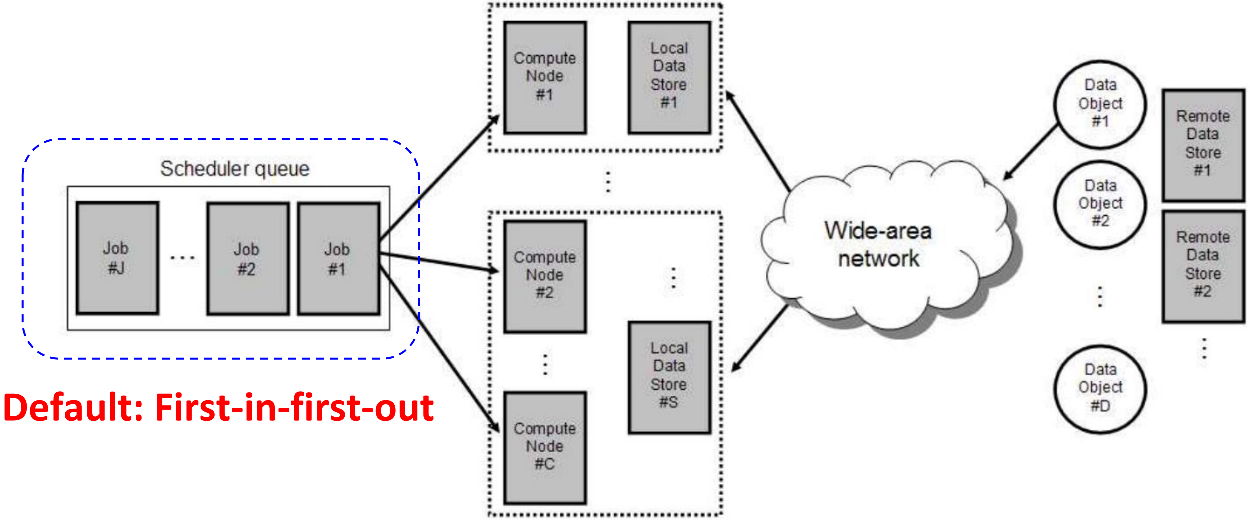
Problem Setup

- **Variables**: 1) job schedule, 2) job assignment, 3) data assignment



Problem Setup

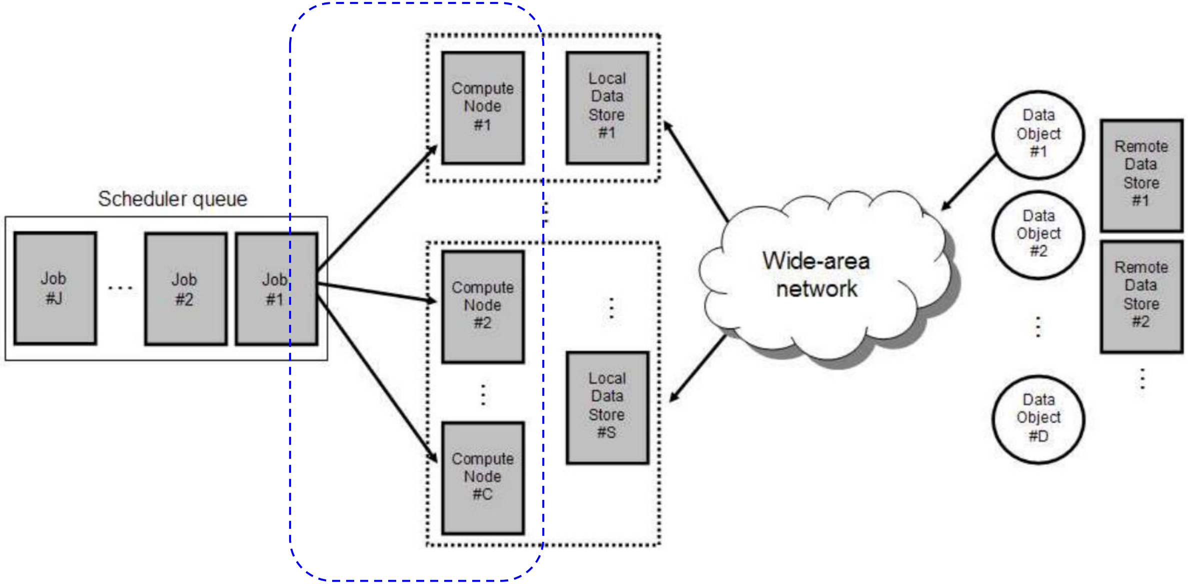
- **Variables**: 1) job schedule, 2) job assignment, 3) data assignment
  - i.e., how the assigned jobs should be computed in order?



Default: First-in-first-out

Problem Setup

- **Variables**: 1) job schedule, 2) job assignment, 3) data assignment
  - i.e., which CN (compute node) computes i-th job



Problem Setup

- **Variables**: 1) job schedule, 2) job assignment, 3) data assignment
  - i.e., which SN (local storage node) saves i-th data object

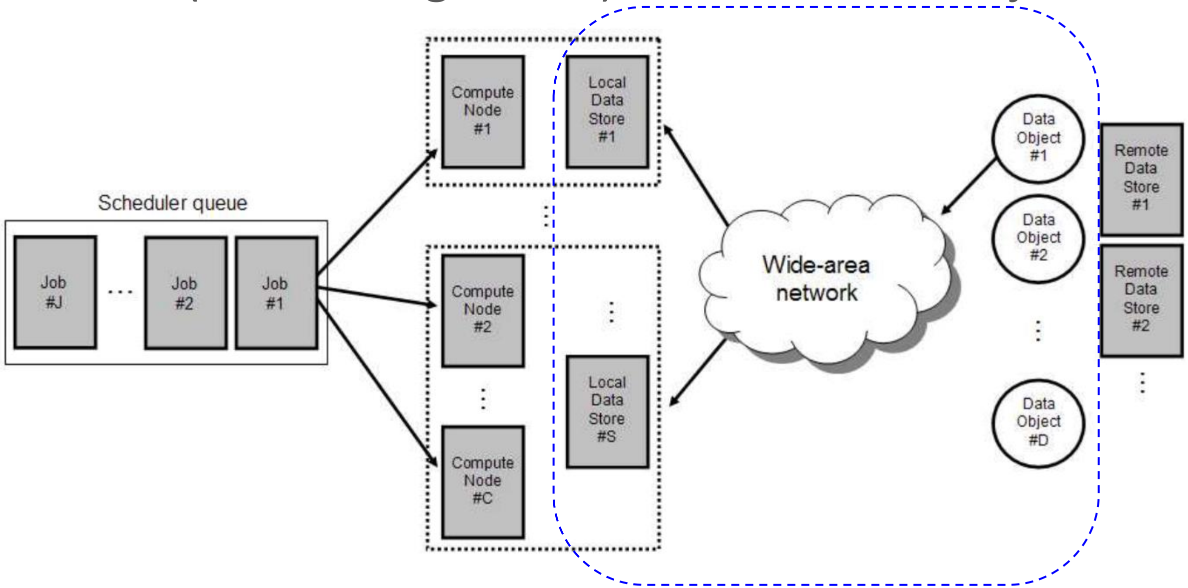
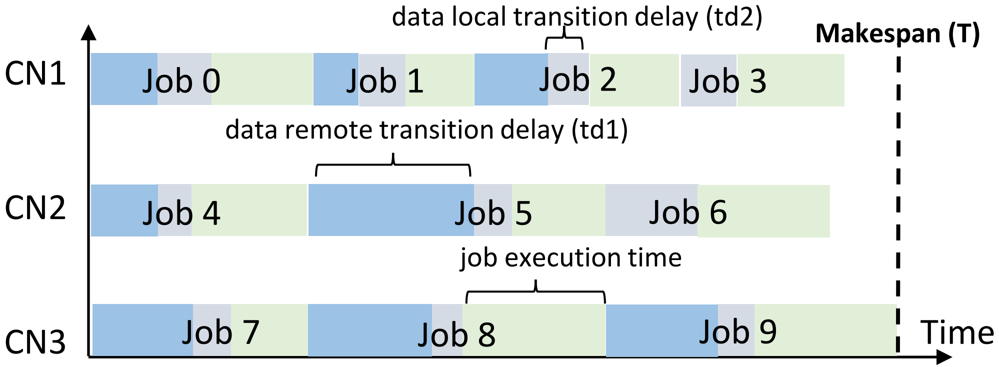


Illustration of Problem

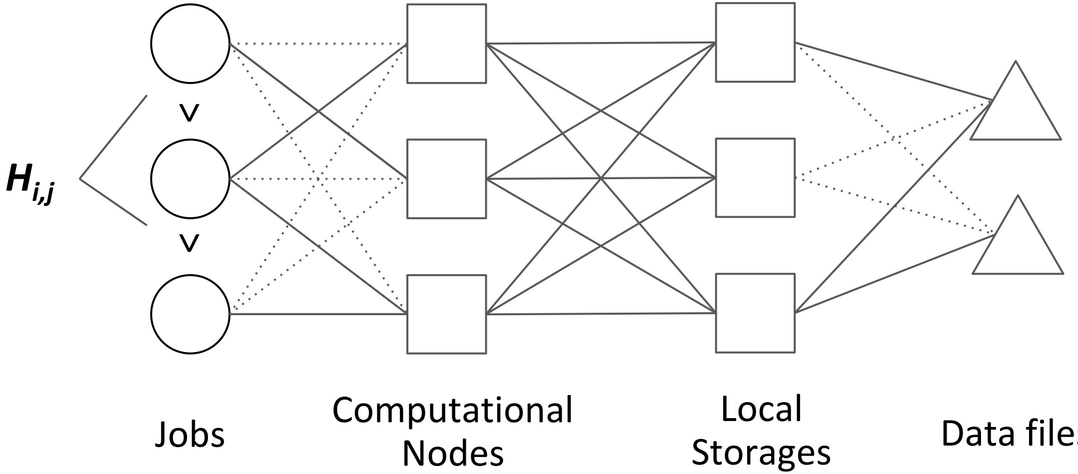
- Assumption: 3 computational nodes, 10 jobs



Notations

Optimization variables

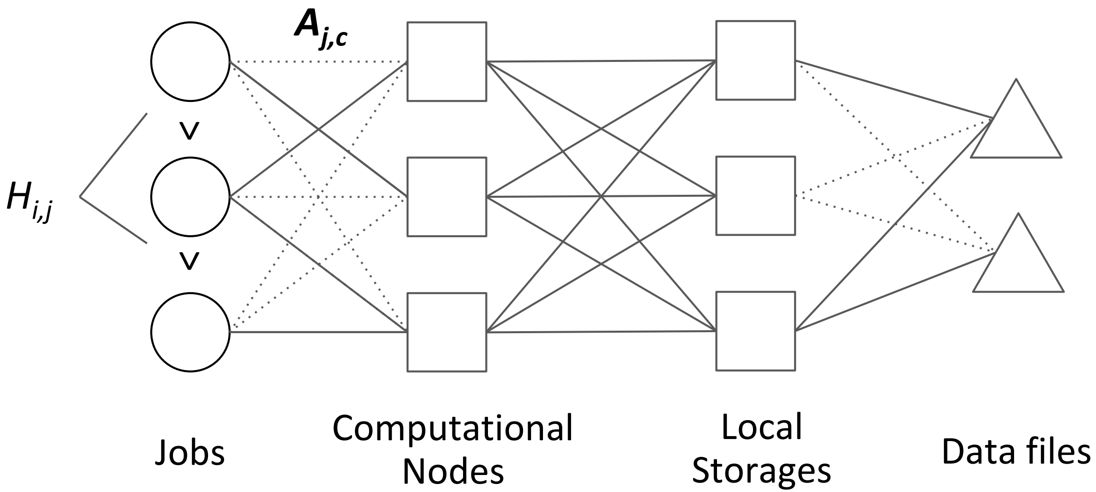
- $H_{i,j} \in \{0,1\}$: ***job i*** is scheduled before ***job j*** if it is 1



Notations

Optimization variables

- $H_{i,j}$∈{0,1}: ***job i*** is scheduled before ***job j*** if it is 1
- $A_{j,c}$∈{0,1}: ***job j*** is assigned to computational ***node c*** if it is 1

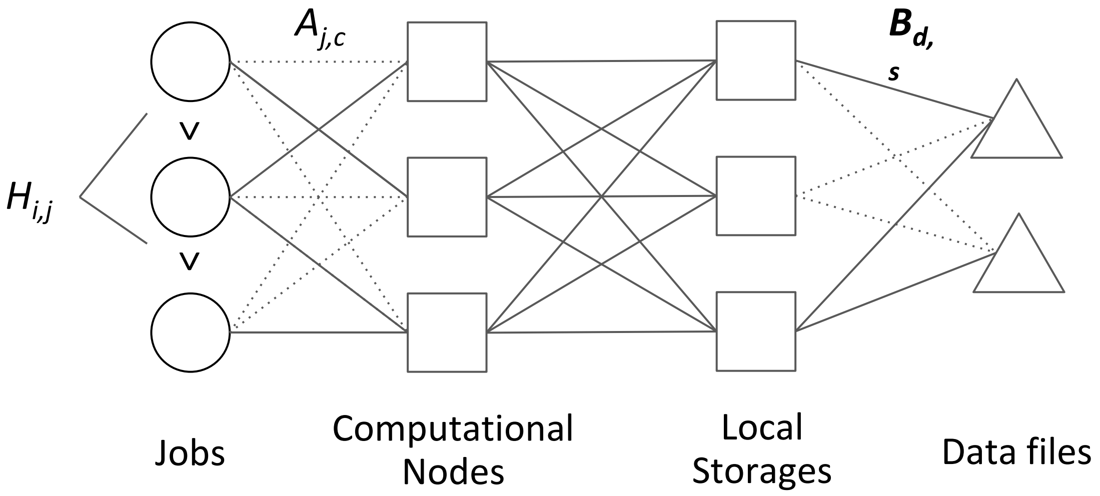


+ Some constraints (e.g., unique assignment)

Notations

Optimization variables

- $H_{i,j} \in \{0,1\}$: ***job i*** is scheduled before ***job j*** if it is 1
- $A_{j,c} \in \{0,1\}$: ***job j*** is assigned to computational ***node c*** if it is 1
- $B_{d,s} \in \{0,1\}$: ***data file d*** is assigned to ***local storage s*** if its is 1



+ Some constraints (e.g., unique assignment)

Our solution: AlterMILP

- **Idea: Alternating optimization** by fixing one variable as constant
  - If **variables are splitted** ($A_{j,c}$ vs. $H_{i,j}$, $B_{d,s}$), then problem **becomes MILP again**

$$w_d = \sum_{s=1}^{S} td_1(d,s) B_{d,s}, \quad \forall d \in [D]; \tag{5}$$

$$l_j = \max_{d \in O_j} \left(\max\{w_d, f_j\} + \sum_{s=1}^{S} \sum_{c=1}^{C} td_2(d,s,c) A_{j,c} B_{d,s} \right), \quad \forall j \in [J]; \tag{6}$$

$$f_j \geq V(H_{i,j}(A_{j,c} + A_{i,c} - 1) - 1) + (l_i + e_i), \ \forall i \neq j,\ i,j \in [J], c \in [C] \tag{7}$$

$$e_j = \sum_{c=1}^{C} exe(j,c) A_{j,c}, \quad \forall j \in [J]; \tag{8}$$

$$T \geq l_j + e_j, \quad \forall j \in [J]; \tag{9}$$

$$H_{i,j} \in \{0,1\}, \quad \forall i \neq j,\ i,j \in [J]; \tag{10}$$

$$A_{j,c} \in \{0,1\},\ f_j, l_j, e_j \geq 0, \quad \forall j \in [J],\ c \in [C]; \tag{11}$$

$$B_{d,s} \in \{0,1\},\ w_d \geq 0, V >> 0, \quad \forall d \in [D],\ s \in [S]. \tag{12}$$

Our solution: AlterMILP

- Also, **constraints are splitted** (but same) to ease the optimization

$$\min \quad T \qquad (1)$$
minimize makespan

$$\text{s.t.} \quad H_{i,j} + H_{j,i} = 1, \quad \forall i \neq j,\ i,j \in [J]; \qquad (2)$$
Order is unique

$$\sum_{c=1}^{C} A_{j,c} = 1, \quad \forall j \in [J]; \qquad (3)$$
Job assignment is unique

$$\sum_{s=1}^{S} B_{d,s} = 1, \quad \forall d \in [D]; \qquad (4)$$
Data assignment is unique

$$w_d = \sum_{s=1}^{S} td_1(d,s) B_{d,s}, \quad \forall d \in [D]; \qquad (5)$$
First downloading from remote resource

$$l_j^1 = \max_{d \in O_j} \left(w_d + \sum_{s=1}^{S} \sum_{c=1}^{C} td_2(d,s,c) A_{j,c} B_{d,s} \right), \quad \forall j \in [J]; \qquad (6)$$

$$l_j^2 = \max_{d \in O_j} \left(f_j + \sum_{s=1}^{S} \sum_{c=1}^{C} td_2(d,s,c) A_{j,c} B_{d,s} \right), \quad \forall j \in [J]; \qquad (7)$$

$$l_j = \max \{ l_j^1, l_j^2 \}; \qquad (8)$$
Beginning of execution

$$f_j \geq V\big(H_{i,j}(A_{j,c} + A_{i,c} - 1) - 1\big) + (l_i + e_i), \quad \forall i \neq j,\ i,j \in [J], c \in [C] \qquad (9)$$
Precedence

$$e_j = \sum_{c=1}^{C} exe(j,c) A_{j,c}, \quad \forall j \in [J]; \qquad (10)$$
Execution time

$$T \geq l_j + e_j, \quad \forall j \in [J]; \qquad (11)$$
Total makespan

$$H_{i,j} \in \{0,1\}, \quad \forall i \neq j,\ i,j \in [J]; \qquad (12)$$

$$A_{j,c} \in \{0,1\},\ f_j, l_j, e_j \geq 0, \quad \forall j \in [J],\ c \in [C]; \qquad (13)$$

$$B_{d,s} \in \{0,1\},\ w_d \geq 0, V >> 0, \quad \forall d \in [D],\ s \in [S]. \qquad (14)$$

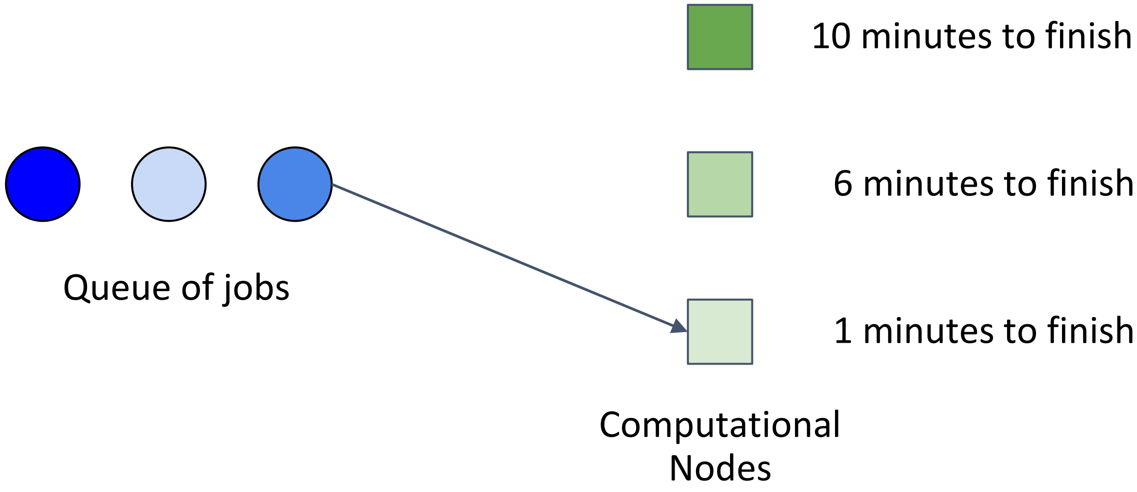
Summary of Related Works

- **Considerable baselines**
  - Two categories: *independent* optimization & *joint* optimization

Baselines	
Random	
MinTrans	independent
MinExe	independent
Greedy	independent
Ensemble Greedy	independent
JDS-HNN	joint
Genetic alg.	joint
DIANA scheduling	joint
MIQP	joint

Baselines: Independent Optimization

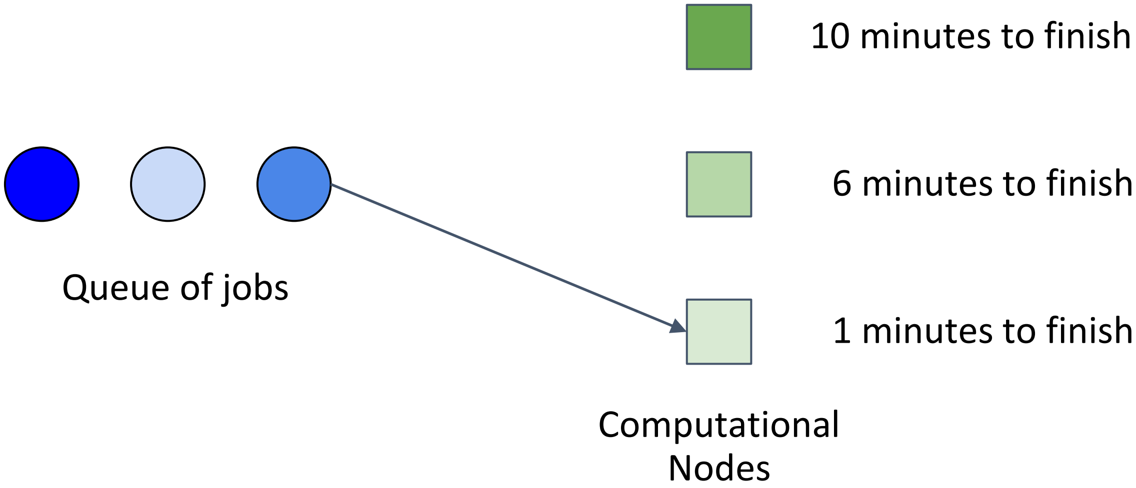
- **Greedy[1]:** allocate job to next available computational node
  - Random data assignment & job scheduling



[1] Park and Kim., Chameleon: a resource scheduler in a data grid environment., IEEE International Symposium on Cluster Computing and the Grid 2003

Baselines: Independent Optimization

- **Ensemble Greedy[1]:** Run the greedy algorithm multiples times with different job order in the queue
  - No longer real-time, but benefit from multiple trials



[1] Park and Kim., Chameleon: a resource scheduler in a data grid environment., IEEE International Symposium on Cluster Computing and the Grid 2003

Jar of Stone Method

Each time move a stone from the highest jar to the lowest jar to balance the storage

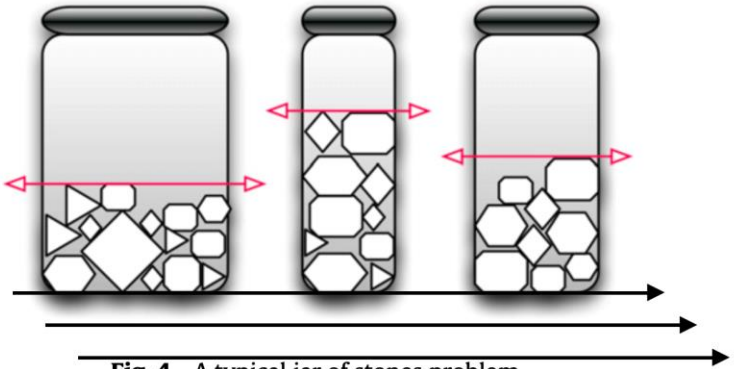
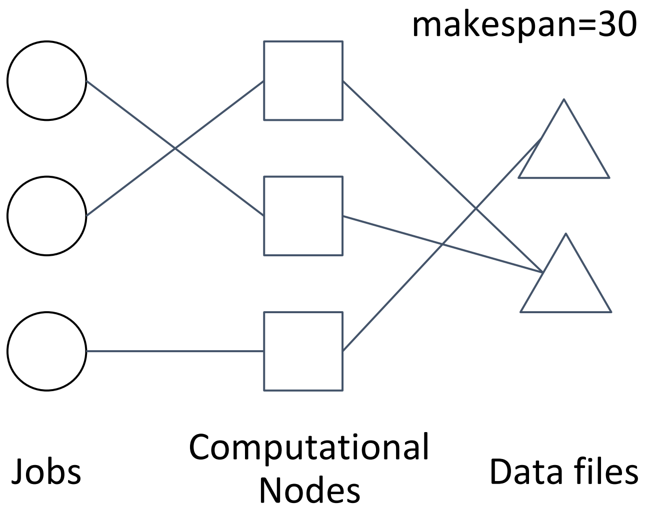


Fig. 4. A typical jar of stones problem.

Baselines: Joint Optimization

- **JDS-HNN[1]**
  - Iterating (1) generating new candidate solution via local greedy search
  - (2) Evaluating the candidate and update the best solution

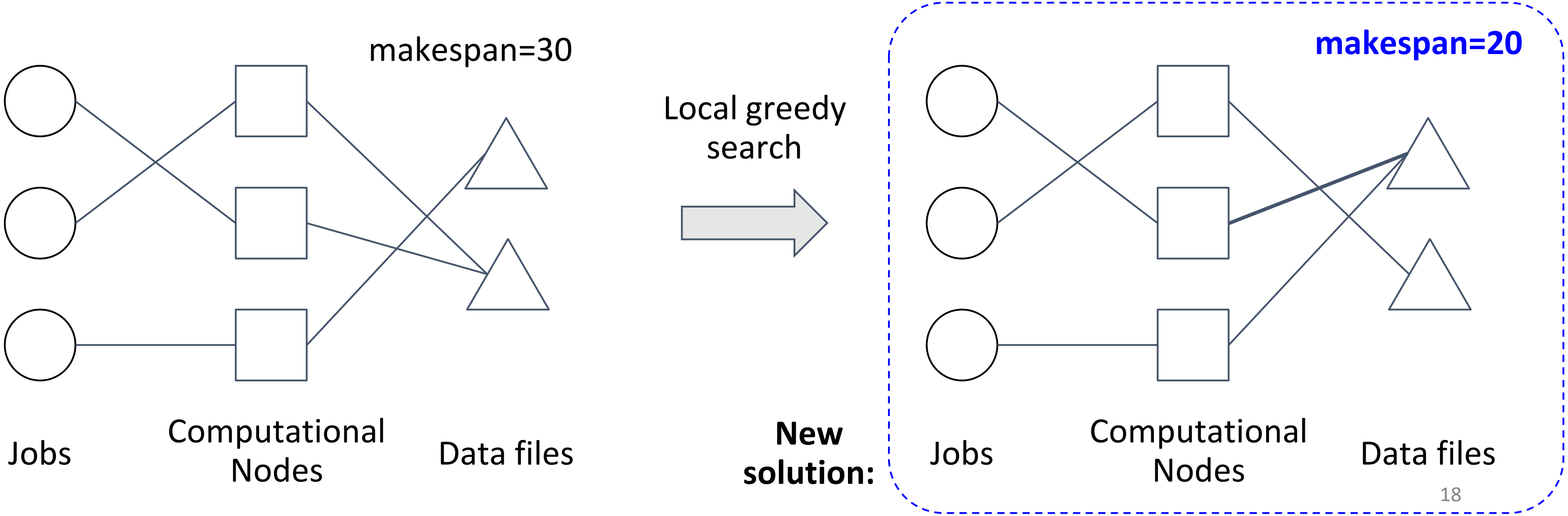
Current solution:



[1] Taheir et al., Hopfield Neural Network for simultaneously job scheduling and data replication in grids, Future Generation Computer Systems 2013

Baselines: Joint Optimization

- **JDS-HNN**
  - Iterating (1) generating new candidate solution via local greedy search
  - (2) Evaluating the candidate and update the best solution



Experimental Setups

Setups**: Simulated environment** (e.g., cloud computing)[1,2]

1. *Computational Nodes*: number of computational nodes, computational efficiency (job size/time)
2. *Data storages*: number of local storages and remote storages
3. *Data files*: number of data files and their sizes
4. *Jobs*: number of jobs and the data files they need

[1] Taheri et al., Hopfield neural network for simultaneous job scheduling and data replication in grids., 2013
[2] Casas et al., A balanced scheduler with data reuse and replication for scientific workflows in cloud computing systems., 2017

Experimental Setups (Parameters)

- **Computational Nodes, Data storages, Data objects, Jobs**
  - Small: 10, 10, 20, 10
  - Medium: 20, 20, 100, 50
  - Large: 50, 50, 300, 100

Baselines	Small	Medium	Large
Random	2903	21052	23221

Results: Comparison with Baselines

Current algorithm (BCD MILP) outperforms other baselines (under same time)

Baselines	Small	Medium	Large
Random	2903	21052	23221
MinTrans	2819	19227	18924
MinExe	2215	9262	8564
Greedy	2278	11304	10371
Ensemble Greedy	1781	10079	9431.3
JDS-HNN	1914	10221	8951
Genetic alg.	1875	12122	13222
MIQP	2453	N/A	N/A
DIANA scheduling	1736	63021	121050
AlterMILP (Ours)	**1707**	**8714**	**7912**