LECTURE 21

Scheduling in Distributed Systems

General

- Scheduling refers to assigning a resource and a start time end to a task
- Much of scheduling research has been done in Operations Research Community e.g Job Shop, Flow shop scheduling etc.
- Scheduling is often an overloaded term in Grids.
- A related term is mapping that assigns a resource to a task but not the start time.

Systems taxonomy

- Parallel Systems
- Distributed Systems
- Dedicated Systems
- Shared Systems
  - Time Shared e.g. aludra
  - Space Shared e.g. HPCC cluster
- Homogeneous Systems
- Heterogeneous Systems

Scheduling Regimens

- Online/Dynamic Scheduling
- Offline/Static Scheduling
- Resource level Scheduling
- Application level Scheduling

Applications taxonomy

- Bag of tasks – Independent tasks
- Workflows – dependent tasks
  - Generally Directed Acyclic Graphs (DAGs)
- Performance criteria
  - Completion time (makespan), reliability etc.

Scheduling Bag of Tasks on Dedicated Systems

- Min-Min
- Max-Min
- Sufferage

Min-Min Heuristic

- For each task determine its minimum completion time over all machines
- Over all tasks find the minimum completion time
- Assign the task to the machine that gives this completion time
- Iterate till all the tasks are scheduled

Example of Min-Min

	T1	T2	T3
M1	140	20	60
M2	100	100	70

Stage 1:

T1-M2 = 100

T2-M1 = 20

T3-M1 = 60

Assign T2 to M1

	T1	T3
M1	160	80
M2	100	70

Stage 2:

T1-M2 = 100

T3-M2 = 70

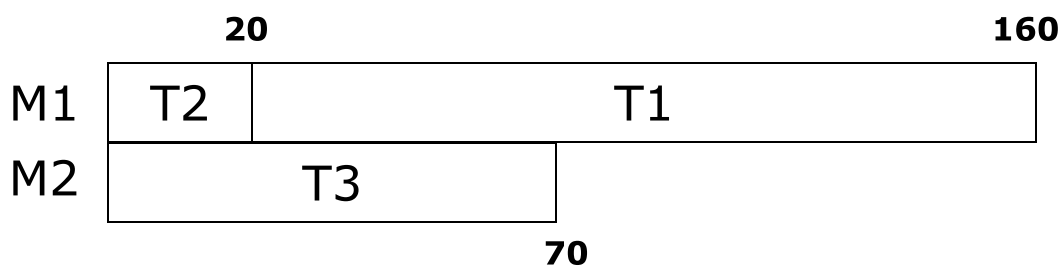
Assign T3 to M2

	T1
M1	160
M2	170

Stage 3:

T1-M1 = 160

Assign T1 to M1



Max-Min Heuristic

- For each task determine its minimum completion time over all machines
- Over all tasks find the maximum completion time
- Assign the task to the machine that gives this completion time
- Iterate till all the tasks are scheduled

Example of Max-Min

	T1	T2	T3
M1	140	20	60
M2	100	100	70

Stage 1:

T1-M2 = 100

T2-M1 = 20

T3-M1 = 60

Assign T1 to M2

	T2	T3
M1	20	60
M2	200	170

Stage 2:

T2-M1 = 20

T3-M1 = 60

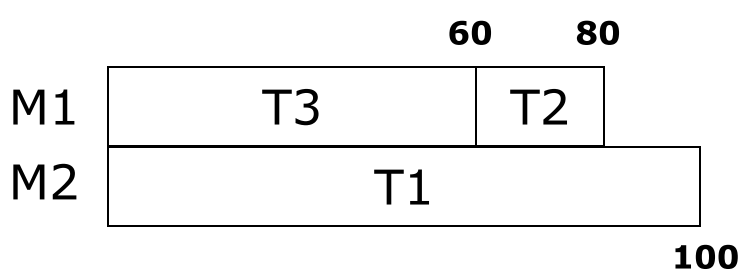
Assign T3 to M1

	T2
M1	80
M2	200

Stage 3:

T2-M1 = 80

Assign T2 to M1



Sufferage Heuristic

- For each task determine the difference between its minimum and second minimum completion time over all machines (sufferage)
- Over all tasks find the maximum sufferage
- Assign the task to the machine that gives this sufferage
- Iterate till all the tasks are scheduled

Example of Sufferage

	T1	T2	T3
M1	140	20	60
M2	100	100	70

Stage 1:

T1 = 40

T2 = 80

T3 = 10

Assign T2 to M1

	T1	T3
M1	160	80
M2	100	70

Stage 2:

T1 = 60

T3 = 10

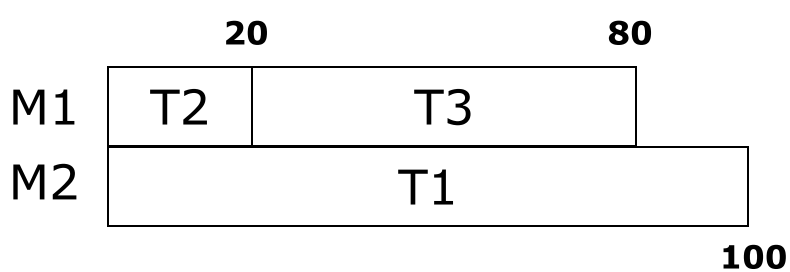
Assign T1 to M2

	T3
M1	80
M2	170

Stage 3:

T3 = 90

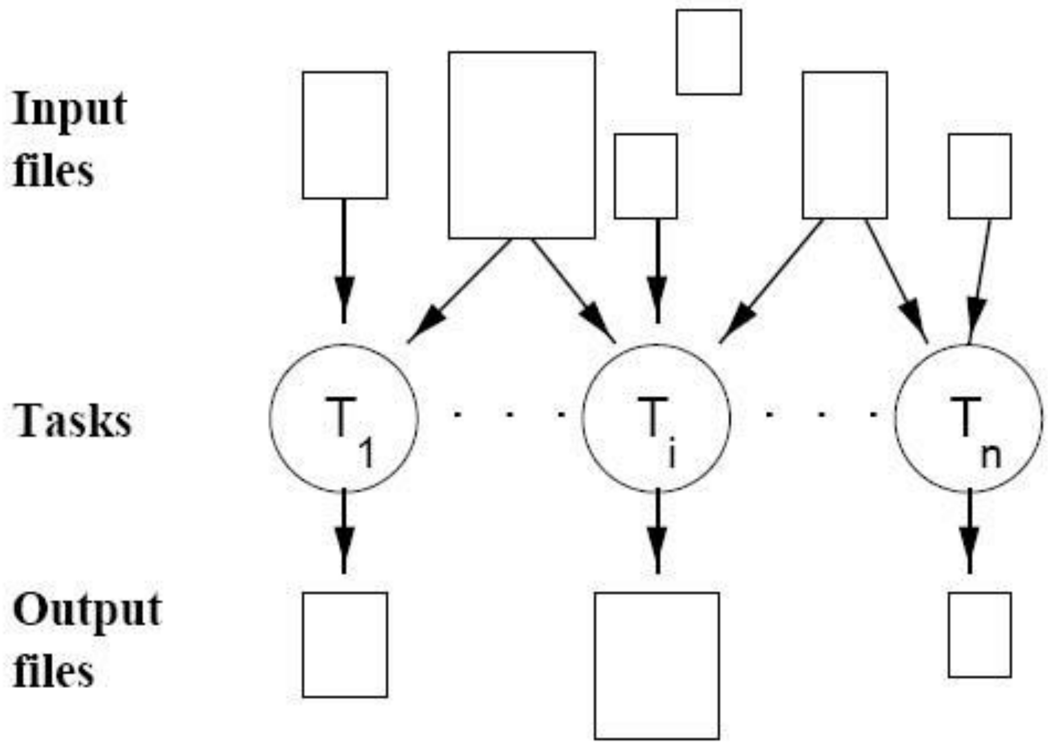
Assign T3 to M1



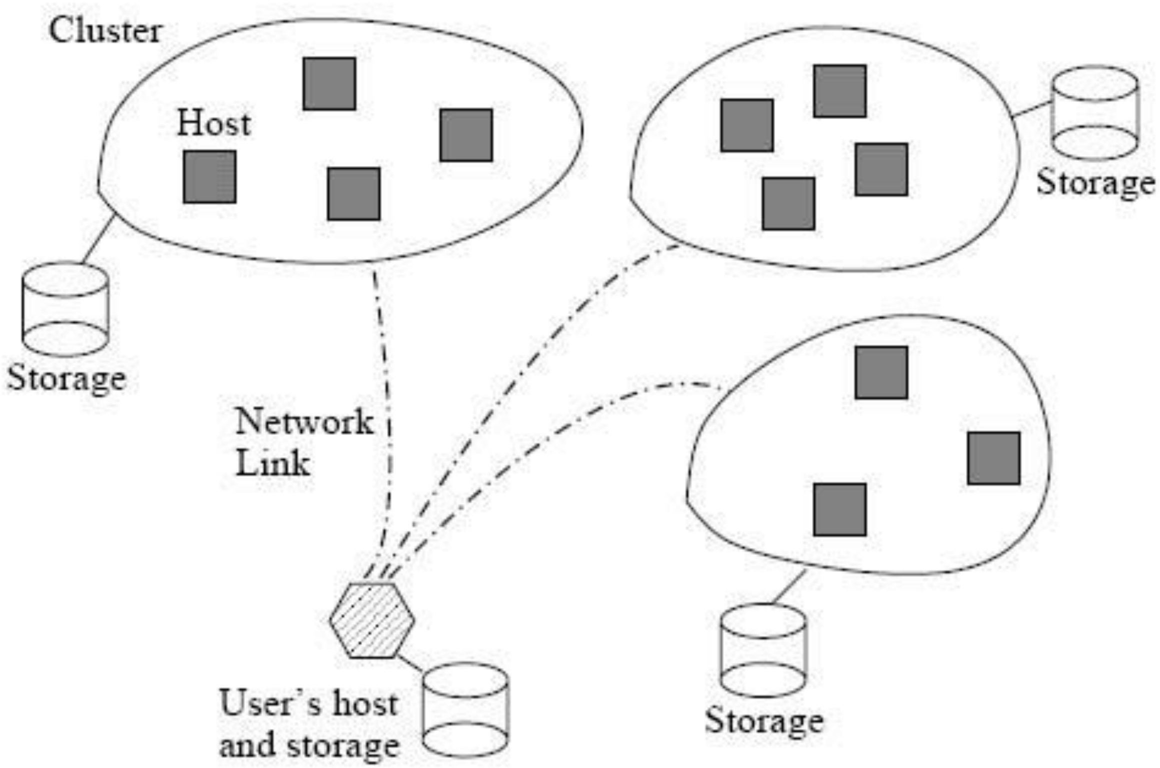
Grid Environments

- Time-shared resources
- Heterogeneous resources
- Tasks require input files that might be shared
- Data transfer times are important

Application Model



System Model

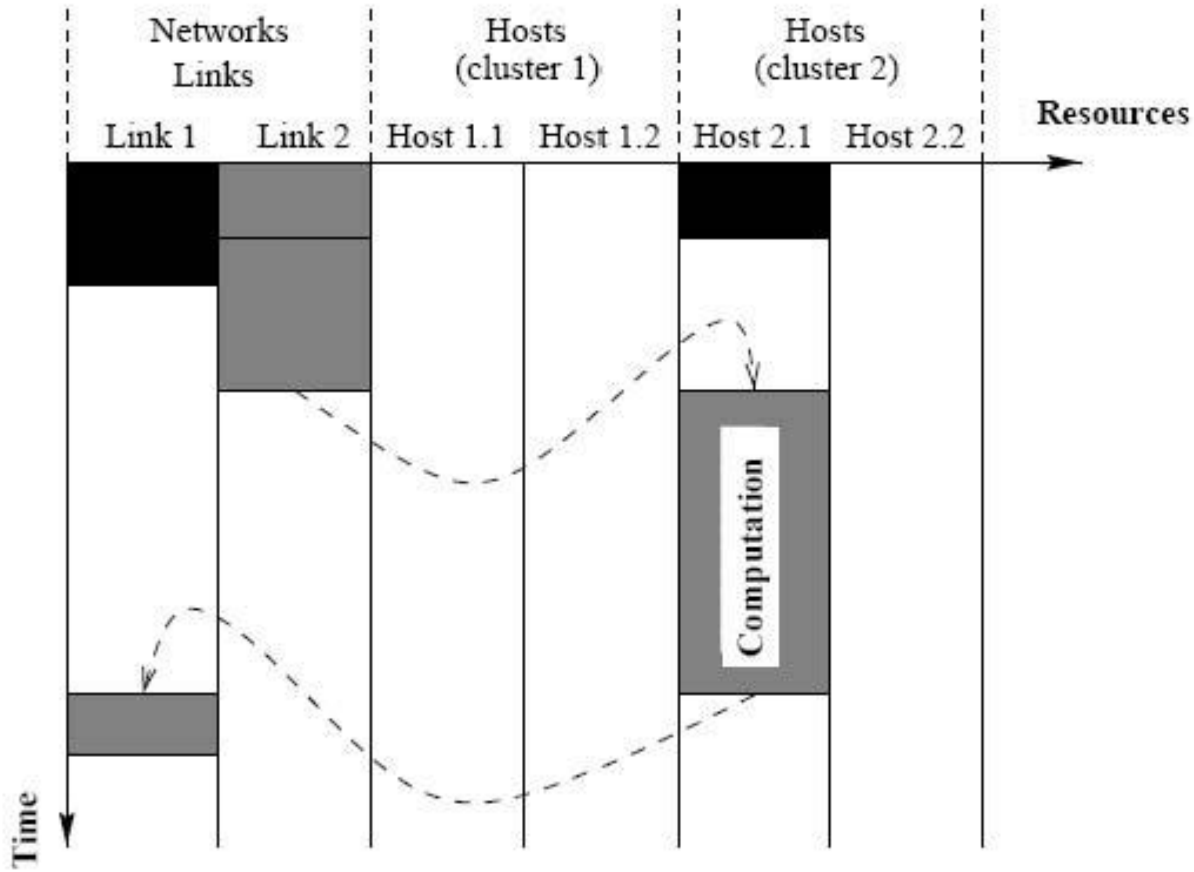


Scheduling Heuristic

Schedule()

1. Compute the next Scheduling event
2. Create a Gantt Chart G
3. For each computation and file transfer underway
    1. Compute an estimate of its completion time
    2. Update the Gantt Chart G
4. Select a subset of tasks that have not started execution: T
5. Until each host has been assigned enough work
    1. Heuristically assign tasks to hosts
6. Convert G into a plan

Sample Gantt Chart



Possible Variations

Schedule()

1. Compute the next Scheduling event
2. Create a Gantt Chart G
3. For each computation and file transfer underway
    1. Compute an estimate of its completion time
    2. Update the Gantt Chart G
4. Select a subset of tasks that have not started execution: T
5. Until each host has been assigned enough work
    1. Heuristically assign tasks to hosts
6. Convert G into a plan

XSufferage

- Tasks may have little intra-cluster Completion time (CT) variation and large inter-cluster CT variation.
- Cluster-level MCT for Sufferage.

		T1
C1	H1	5
	H2	6
C2	H3	30
	H4	32

		T1	T1
C1	H1	5	5
	H2	6	
C2	H3	30	30
	H4	32	

Simulations

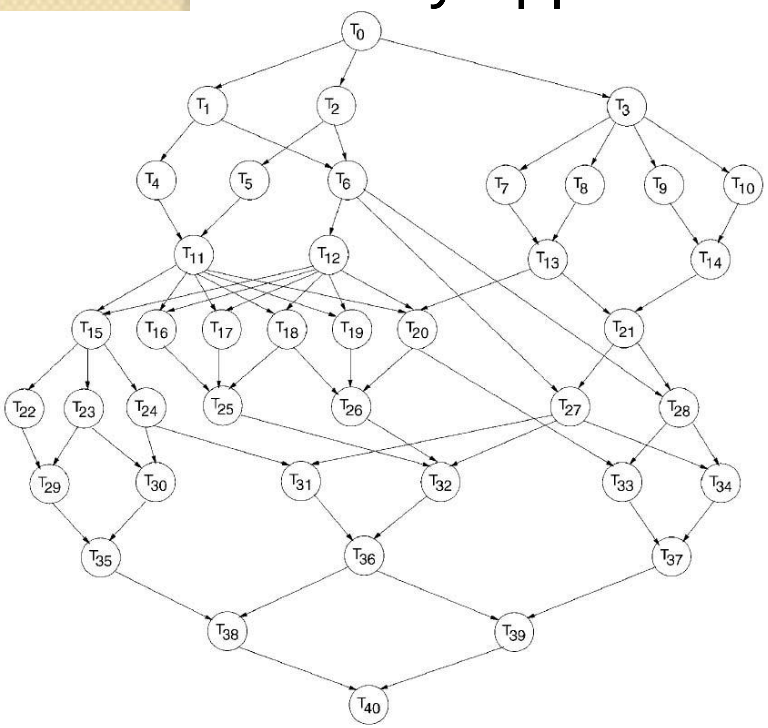
- 1000 Simulated Grids, U(2,12)xU(2,32)
- 2000 applications, U(2,10)xU(20x1000)
- Task runtime U(100,300)
- Large Input File, U(400,100000) KBytes
- Small Input File, 1 KB
- One Output File, 10 KB
- Background load on the host machines and network links based on NWS traces
- Results over 1000 random Grid/application pairs.

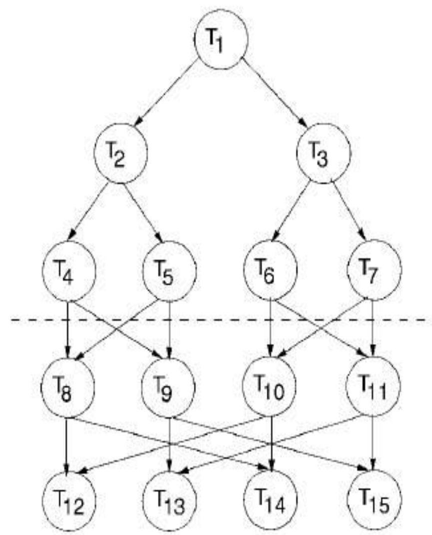
Results

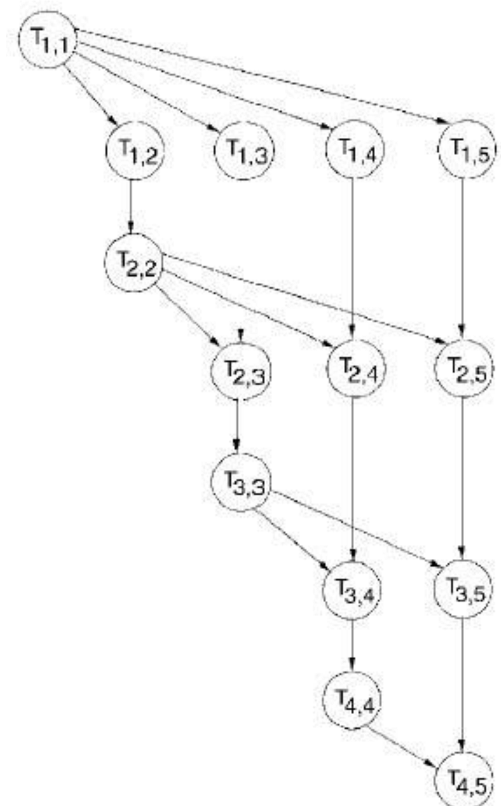
	Geometric mean (sec)	Average Degradation from Best (%)	Average Rank
Max-min	2390	17.3	3.1
Min-min	2452	21.2	3.0
Sufferage	2329	14.1	2.8
XSufferage	2174	6.2	1.8

Scheduling Task Graphs

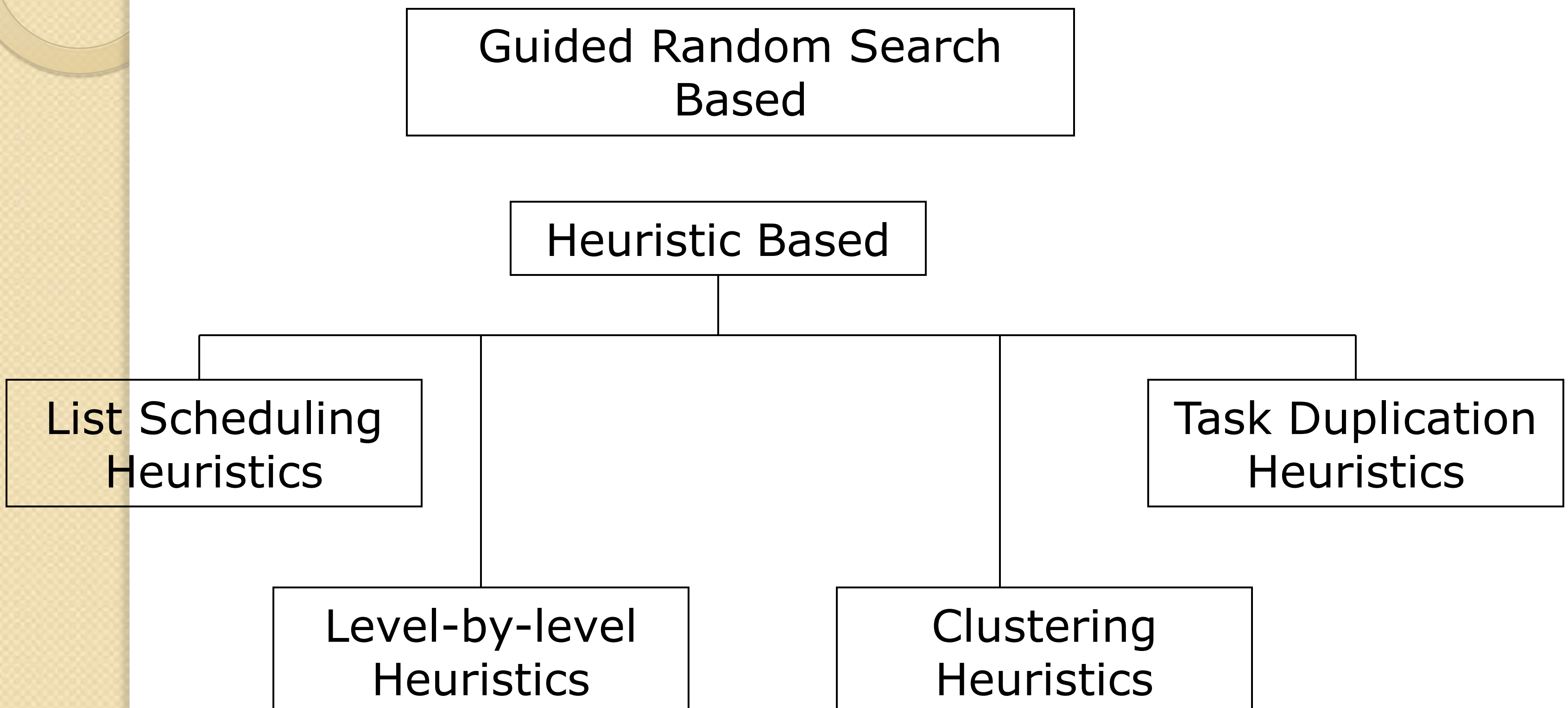
- Task Graphs have dependencies between the tasks in the Application
- Scheduling methods for bag of task applications cannot be directly applied







Graph Scheduling Algorithms

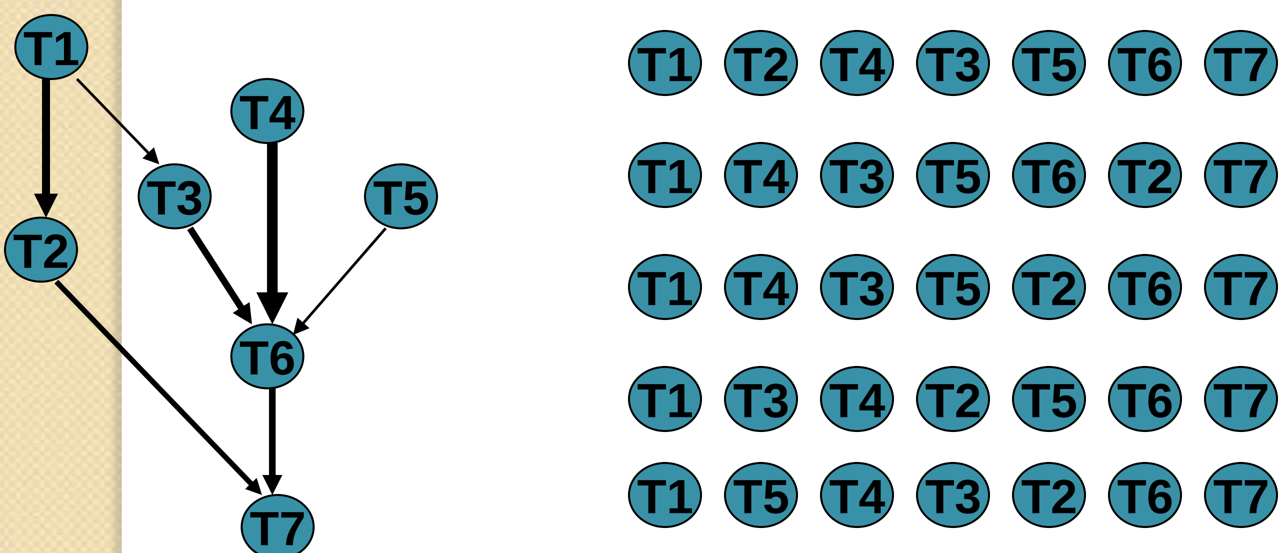


Guided Random Search Based

- Genetic Algorithms
  - A chromosome is an ordering of tasks
  - A rule is required to convert it to a schedule
- Simulated Annealing
- Local Search Techniques, taboo etc

List Scheduling Heuristics

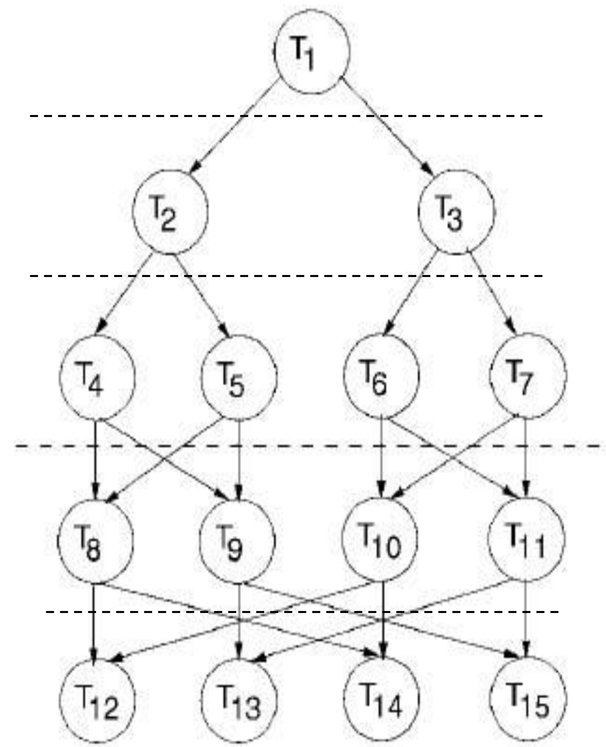
- An ordered list of tasks is constructed by assigning priority to each task
- Tasks are selected on priority order and scheduled in order to minimize a predefined cost function
- Tasks have to be in a topologically sorted order



Level by Level Scheduling

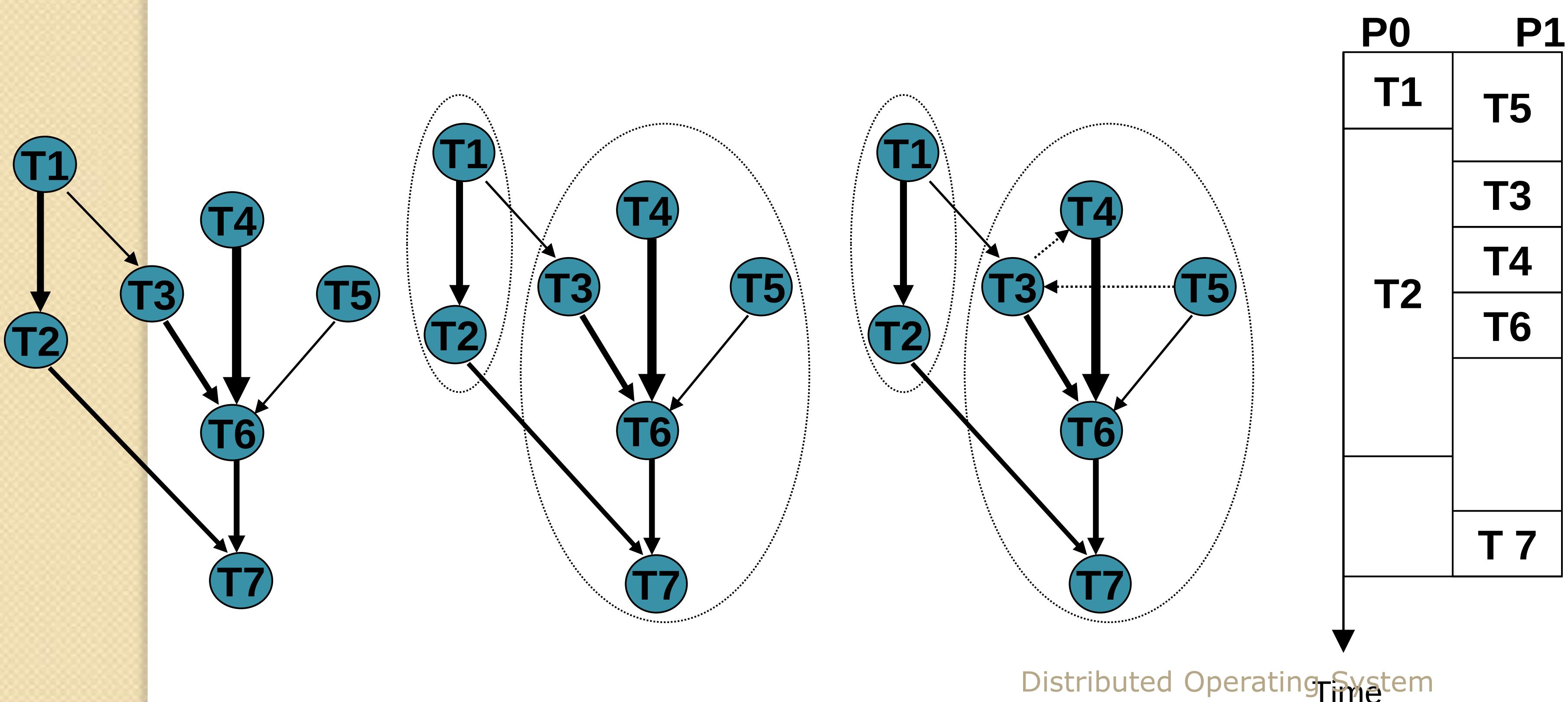
- Partition a DAG into multiple levels such that task in each level are independent.

- Apply Min-Min, Max-Min or other heuristics to tasks at each level



Clustering Heuristics

- Clustering heuristics cluster tasks together
- Tasks in the same cluster are scheduled on the same processor



Heterogeneous Earliest Finish Time

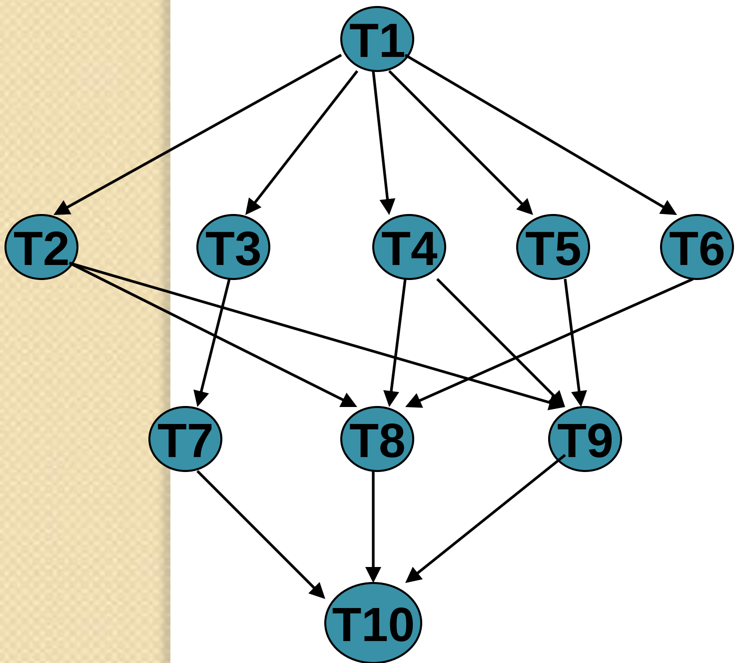
- List scheduling based heuristic
- Do a bottom up traversal of the graph and assign ranks to each task

$$rank_u(n_i) = \overline{w}_i + \max_{n_j \in succ(n_i)} (\overline{c}_{i,j} + rank_u(n_j))$$

$$rank_u(n_{exit}) = \overline{w}_{exit}$$

HEFT- contd

- Compute rank for all tasks in the graph
- Sort the tasks by non-increasing rank values (ensures topological sort)
- While there are unscheduled tasks
  - Select the first task in the list
  - Assign the task to the processor/machine that minimizes its completion time using insertion based scheduling
- endWhile

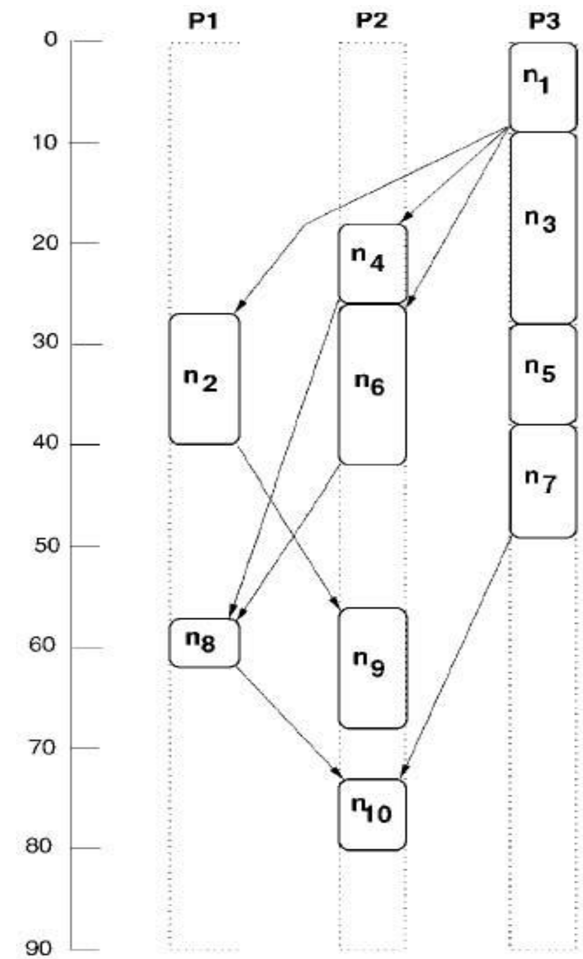


	Priority
T1	108
T2	77
T3	80
T4	80
T5	69
T6	63.33
T7	42.667
T8	35.667
T9	44.333
T10	14.667

HEFT Order

T1 T3 T4 T2 T5 T6 T7 T8 T9 T10

HEFT Schedule



Critical Path on a Processor (CPOP)

- Upward ranking

$$rank_u(n_i) = \overline{w}_i + \max_{n_j \in succ(n_i)} (\overline{c}_{i,j} + rank_u(n_j))$$

$$rank_u(n_{exit}) = \overline{w}_{exit}$$

- Downward ranking

$$rank_d(n_i) = \overline{w}_i + \max_{n_j \in pred(n_i)} (\overline{c}_{i,j} + \overline{w}_j + rank_d(n_j))$$

$$rank_d(n_{entry}) = 0$$

$$priority(n_i) = rank_u(n_i) + rank_d(n_i)$$

CPOP

$|CP| = priority(n_{entry})$

$SET_{CP} = \{n_{entry}\}$

$n_k \leftarrow n_{entry}$

While n_k is not the exit task do

Select n_j where $((n_j \in succ(n_k)$ and $priority(n_j) == |CP|$

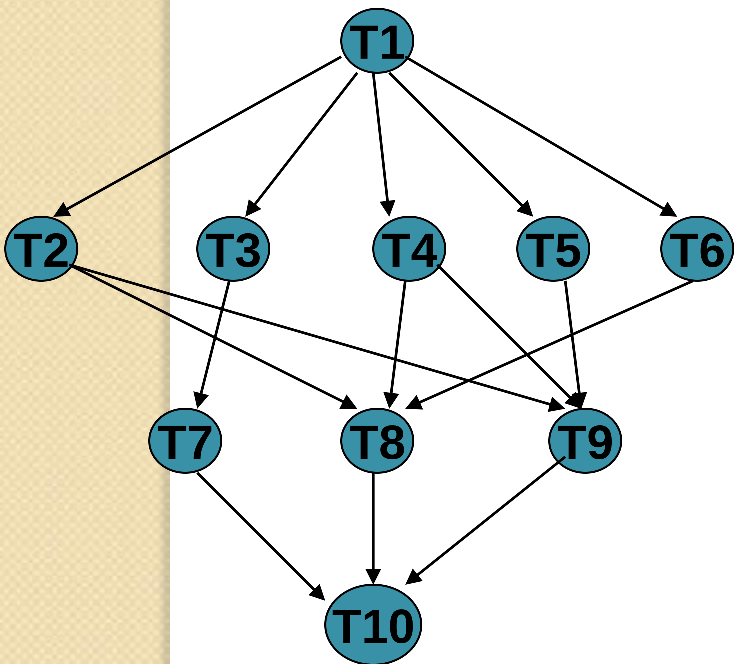
$SET_{CP} = \{SET_{CP} \cup n_j\}$

$n_k \leftarrow n_j$

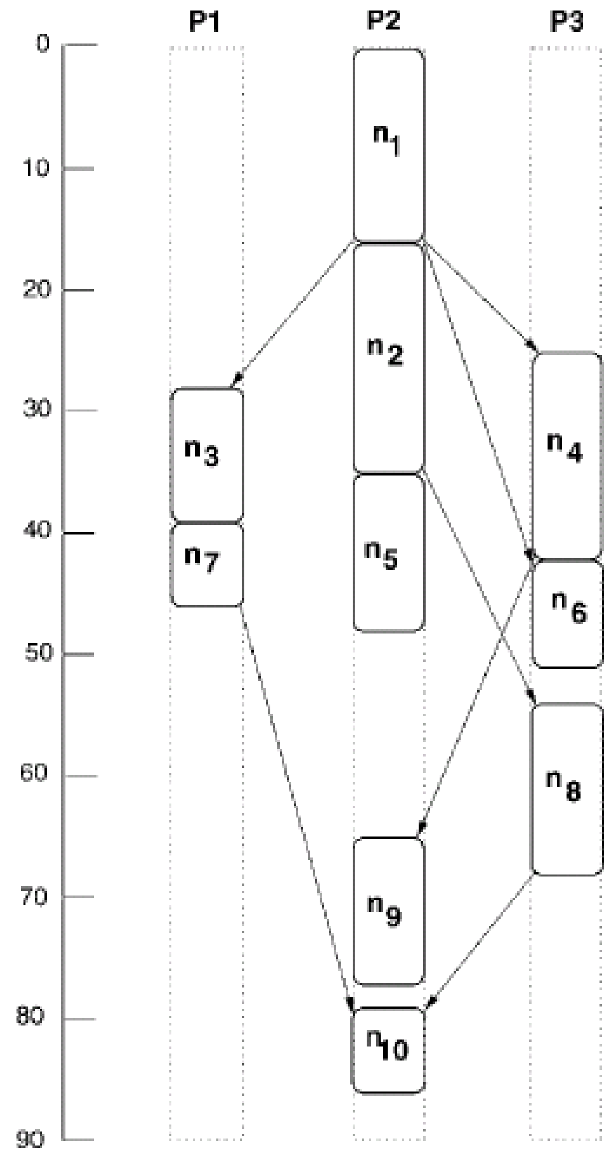
endWhile

Select the Critical Path processor p_{cp} that minimizes the sum of runtimes of tasks on the critical path

Go through the task list in priority order, assign tasks in SET_{CP} to p_{cp} and other tasks to any processor that minimizes its finish time



	Priority
T1	108
T2	108
T3	105
T4	102
T5	93
T6	90.333
T7	105
T8	102.334
T9	108
T10	108



Conclusions

- Heuristics for scheduling independent and dependent tasks on distributed systems
- Rescheduling in order to adapt to dynamic Grid conditions
- Partitioning in case of task graphs
- Future lecture (Nov 27th) on resource provisioning for applications.

ASSIGNMENT

- Q: Explain various scheduling algorithms in distributed system.