EECS 571 “Principles of Real-Time Embedded Systems”

Lecture Note #8: Task Assignment and Scheduling on Multiprocessor Systems

Kang G. Shin
EECS Department
University of Michigan

What Have We Done So Far?

- Scheduling a set of tasks with various constraints on a *single* processor.
- What should we do if schedulability condition for the given task set can't be met?
- Question: which tasks should be assigned to which processors and why?
- Ideally, *combined* task assignment and scheduling is desirable, but this is *very hard*.
- *Common approach*: assign tasks and then schedule them on each processor.

Task Assignment

- What should we consider for task assignment ?

- NP-Complete $\Rightarrow$ Use heuristics
  Examples:

- *Utilization-balancing algorithm*: assign tasks one-by-one selecting the least utilized processor

$$\frac{\sum_{i=1}^{p}(u_i^B)^2}{\sum_{i=1}^{p}(u_i^*)^2} \leq \frac{9}{8}$$

where u_i^* = P_i's utilization under an optimal alg. that minimizes $\sum$ utilization^2

u_i^B = P_i's utilization under best-fit alg.

Kang Shin (kgshin@eecs.umich.edu)

Next-fit alg for RM scheduling

- Homogeneous multiprocessor systems

- There are m *classes* of tasks such that

  - o T_i belongs to class $j < m$ if $2^{\frac{1}{j+1}} - 1 < \frac{e_i}{p_i} \leq 2^{\frac{1}{j}} - 1$.

  - o T_i belongs to class m otherwise.

- Each class of tasks are assigned to a corresponding set of processors.

Example

There are $m = 4$ task classes

Class	Untilization bound
C_1	(0.41, 1.00]
C_2	(0.26, 0.41]
C_3	(0.19, 0.26]
C_4	(0.00, 0.19]

Task set

	T_1	T_2	T_3	T_4	T_5	T_6	T_7
e_i	5	7	3	1	10	16	1
P_i	10	21	22	24	30	40	50
$u(i)$	0.50	0.33	0.14	0.04	0.33	0.40	0.02
Class	C_1	C_2	C_4	C_4	C_2	C_2	C_4
Processor	1	2	4	4	2	5	4

	T_8	T_9	T_{10}	T_{11}
e_i	3	9	17	21
P_i	55	70	90	95
$u(i)$	0.05	0.13	0.19	0.22
Class	C_4	C_4	C_4	C_3
Processor	4	4	4	3

Bin-packing assignment for EDF

- Same assumptions on tasks and processors as Next-fit alg.

- Task set is EDF-schedulable if $U \leq 1$.

- Assign tasks such that $U \leq 1$ for all processors.

Myopic offline scheduling alg

- Can consider resources other than CPU

- *Given:* set of tasks, their arrival times, execution times, deadlines.

- *Allocation tree:*

  - o Root: null allocation

  - o Node: an assignment and scheduling of a subset of tasks.

  - o child node: parent's allocation + a task

  - o leaf: "complete" allocation.

  - o How many levels for an n-task system?

  - o A level-i node means?

  - o Very expensive to generate a complete allocation tree $\Rightarrow$ heuristics.

Combined Assignment and Scheduling

- *Static* (offline) assignment of periodic and/or critical tasks: myopic scheduling, B&B alg.

- *Dynamic* (online) load sharing of aperiodics and/or non-criticals
    - Bidding
    - Focused addressing
    - Drafting
    - Buddy

Offline Allocation of Periodics

IEEE Trans. on Software Engineering, vol. 23, no. 12,

pp. 745–758, Dec. 1997

- *Task allocation*: combined task assignment and scheduling of periodics

  - Derive an "optimal" assignment that yields feasible schedules for all processors. How ?

- Main features:

  - Inter-task communications $\Rightarrow$ precedence constraints hence task structure.

  - Tasks are periodic and time-critical $\Rightarrow$ allocation objective function.

- Want allocation x of communicating periodic tasks in a *heterogeneous* distributed system that minimizes *system hazard*, $\Theta(x)$, or maximum *normalized task response time*.

System Model

- Tasks $T = \{T_i : i = 1, 2, \ldots, m\}$;
  Heterogeneous PNs
  $N = \{N_k : k = 1, 2, \ldots, n\}$.

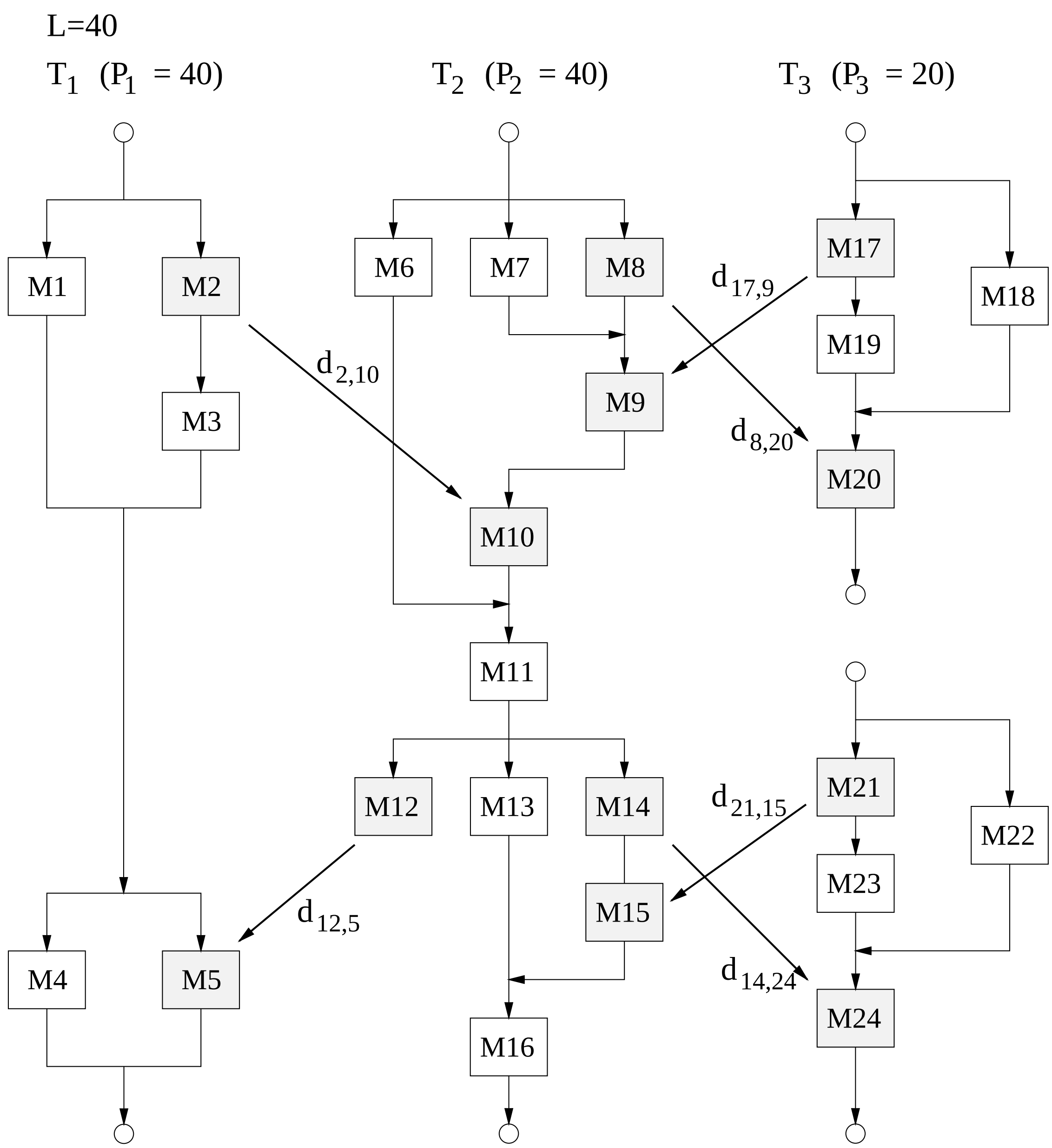
- Allocation constraints:

  - o Co-location of T_i and T_j on *same* PN.
  - o Location of T_i and T_j on *different* PNs.
  - o Location of T_i on a *special* PN.

- Task invocations and release times, precedence constraints, planning cycle.

- Execution times of computation and communication modules.

Example Task Graph



Kang Shin (kgshin@eecs.umich.edu)

Problem Formulation

- Normalized task response time of the v-th invocation of T_i:

$$\overline{c}_{iv} = \frac{c_{iv} - r_{iv}}{d_{iv} - r_{iv}}$$

- System hazard under allocation x:

$$\Theta^x = \max_{T_i \in T} \overline{c}_{iv}$$

- *Problem*: find an optimal x^* that minimizes the system hazard.

- Both $\overline{c}_{iv}$ and Θ^x depend on:

  - o how tasks are assigned under x and
  - o how assigned tasks are scheduled on each PN.

Task Allocation Algorithm

- Consists of two branch-and-bound algorithms, one for assignment (B&BA) and the other for scheduling (B&BS).

- Same as traversing a tree

- Vertex = allocation

- Complete allocation = leaf node; B&BS alg

- Partial allocation = intermediate vertex; B&BS is too expensive, so compute and use a *lower bound* of optimal system hazard

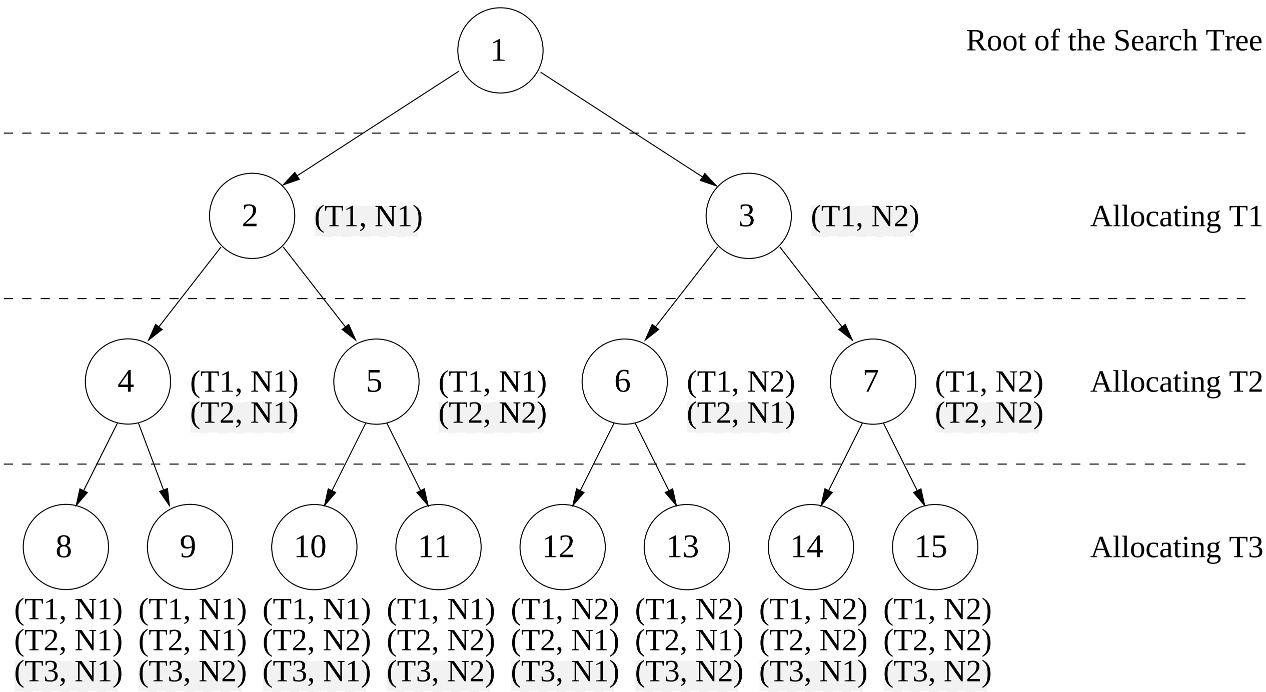
Branch and Bound Algorithm

```
let active set $A$ = $\{Root\}$
let vertex cost $\Theta(Root)$ = 0
let best solution cost, $\Theta_{min}$ = $\infty$

while true do
    let $V_{best}$ = minimum cost vertex in $A$
    if $V_{best}$ is a leaf vertex then
        prune all vertices $V \in A$ except $V_{best}$
        return $V_{best}$ as optimal solution
    else
        generate (task assignments of) all children of $V_{best}$
        remove $V_{best}$ from active set $A$
        for each child $x$ of $V_{best}$ do
            if assignment constraints in set $AC$ are not satisfied then prune $x$
            else
                compute vertex cost $\Theta(x)$
                add $x$ to active set $A$
                if $x$ is a leaf vertex then
                    if $\Theta(x) < \Theta_{min}$ then
                        $\Theta_{min} = \Theta(x)$
                        prune all vertices $V \in A$ for which $V \neq x$ and $\Theta(V) \geq \Theta_{min}$
                    else prune $x$
                end if
            end if
        end for
    end if
end while
```

Kang Shin (kgshin@eecs.umich.edu)

Search Tree



Kang Shin (kgshin@eecs.umich.edu)

B&BA Algorithm

- A terminal vertex or complete assignment: B&BS alg based on dominance properties.

- For each non-terminal vertex or partial assignment x:
  - o B&BS is too expensive
  - o As long as a lower-bound, Θ_{lb}^{x} of the optimal cost for x is used, B&BA will find an optimal assignment.
  - o Θ_{lb}^{x} is obtained by relaxing task invocation times, precedence constraints, etc.

Computing Lower-Bound Vertex Cost

1. Compute the minimum computational load imposed on each processor by tasks already assigned to PNs at search vertex x.

2. Estimate the minimum additional load to be imposed on each PN due to those tasks not yet assigned at x.

3. Schedule the combined load at each PN and compute the system hazard. We ensure that the system hazard of the resulting schedule is a lower bound on the system hazard of any leaf vertex descending from x, i.e., it represents $\Theta(x) = \Theta_{lb}(x)$.

B&BS Algorithms

Scheduling tasks w.r.t. Θ for a given complete assignment is NP-Hard $\Rightarrow$ *Dominance properties* are derived to guide search for an optimal schedule.

- Preemptions which do not reduce Θ must be disallowed.

- A PN is not allowed to idle when there are ready (uncompleted) modules on the PN.

- Always advantageous to reduce the completion time of a task without increasing others'.

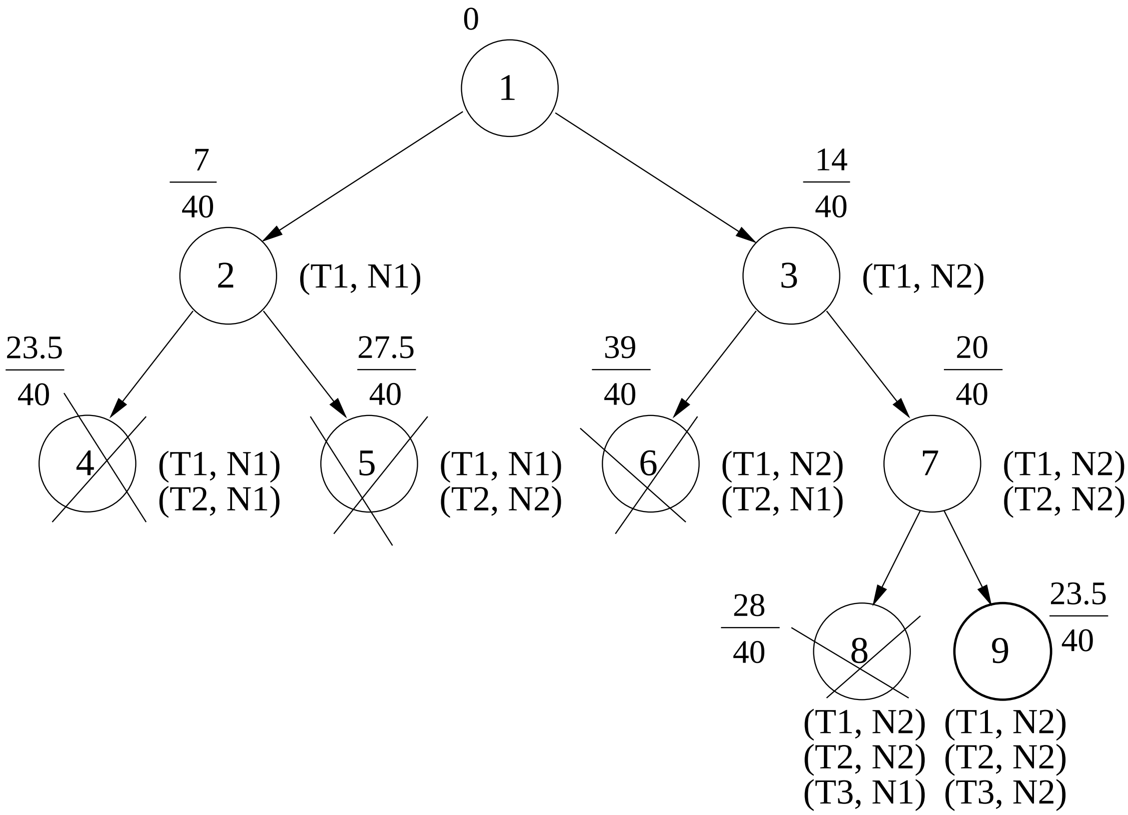
Example

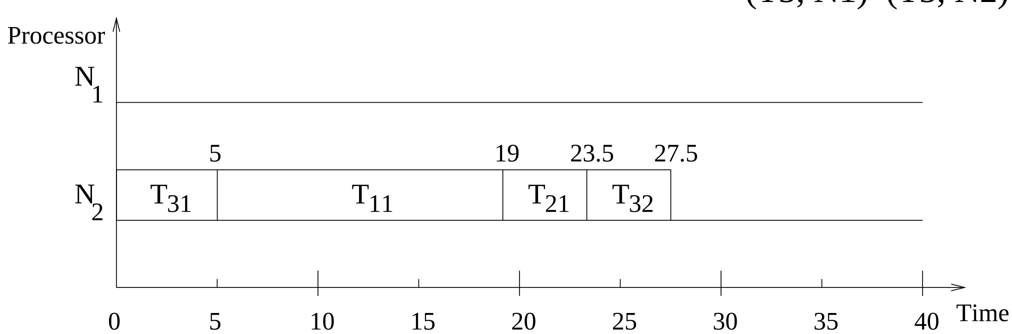
The same as before: 3 tasks and 2 PNs

Module execution times on N_1

M_j	e_{j1}	M_j	e_{j1}	M_j	e_{j1}
M_1	4	M_9	1:3	M_{17}	1:3
M_2	1:4	M_{10}	1:4	M_{18}	1
M_3	2	M_{11}	1	M_{19}	2
M_4	2	M_{12}	2:4	M_{20}	0:1
M_5	2:6	M_{13}	2	M_{21}	1:3
M_6	2	M_{14}	0:2	M_{22}	1
M_7	1	M_{15}	2:3	M_{23}	2
M_8	1:2	M_{16}	3	M_{24}	1:2

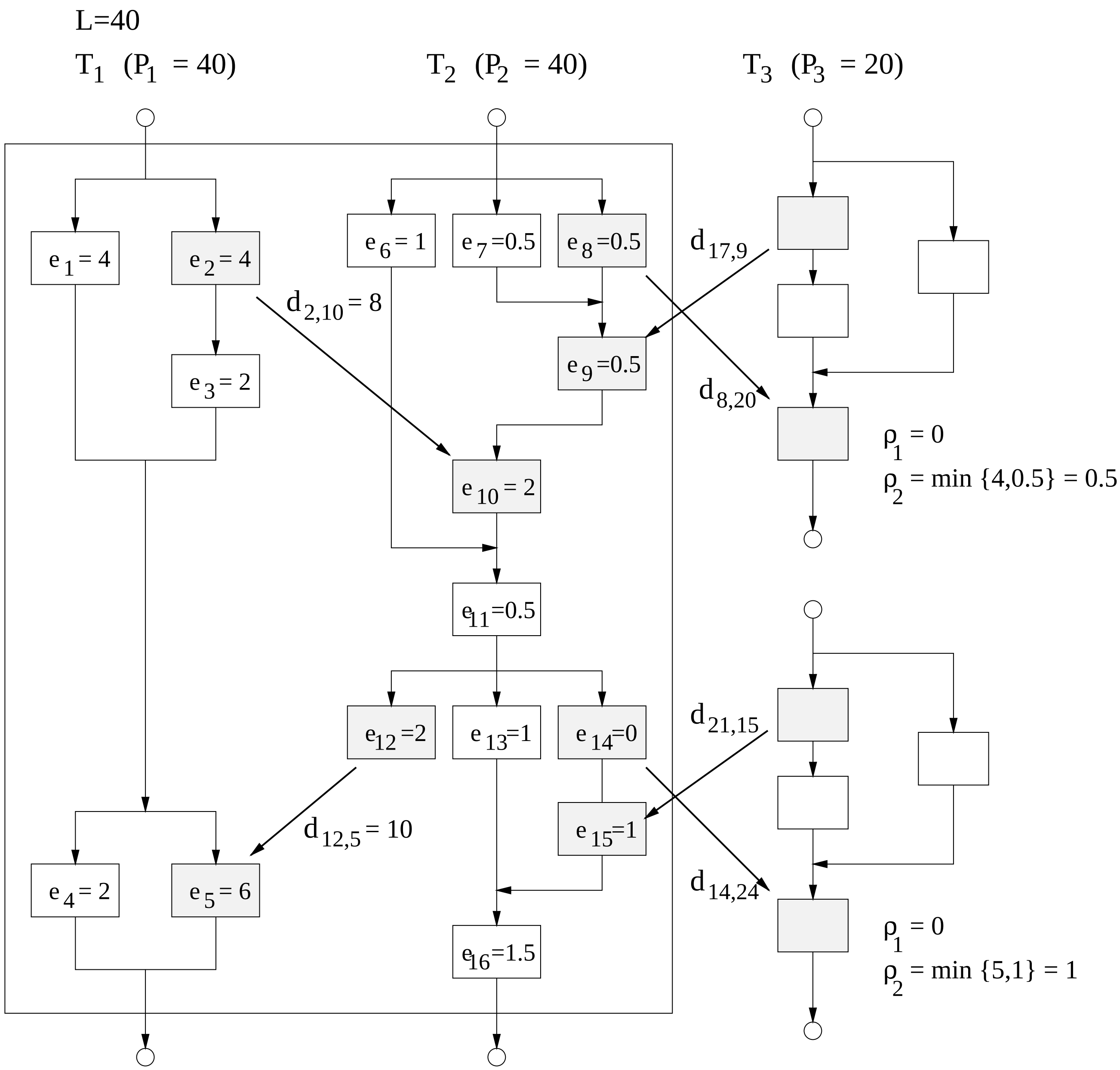
Search Tree Generated and Optimal Schedule





Kang Shin (kgshin@eecs.umich.edu)

Task Graph at Vertex 5



Kang Shin (kgshin@eecs.umich.edu)

Computational Experiences

Tasks	Expanded Vertices	Total Space	% Expanded
6	18	4096	0.43
8	65	65536	0.1
10	95	1048576	0.01
12	133	16777216	0.0008
14	274	268435456	0.0000004

Nodes	Expanded Vertices	Coef of Variation	Total Space	% Expanded
2	16	0.35	256	6.17
4	37	0.8	65536	0.06
6	38	0.8	1679616	0.002