**Hardware Emulators scheduling problem**

Hardware emulator is a machine that can accept specific software modeling language called RTL (which describes logical circuits, which in turn becomes cpu), and imitate the described design in hardware. It is not really important to know more details than this for the problem description, that is why I will stop here ☺, links for better explanations: <http://en.wikipedia.org/wiki/Hardware_emulation>, <http://en.wikipedia.org/wiki/Register-transfer_level>.

Cost of one emulator machine is around $0.5M.

There is also assumed cost of “wasted emulator time” (when the board was not used, but could be used), from numbers that I remember are around $0.5K per hour, but I am not sure about the exact value.

For our problem, emulator can be seen as a machine that have five boards.

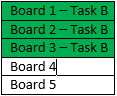
|  |
| --- |
| Board 1 |
| Board 2 |
| Board 3 |
| Board 4 |
| Board 5 |

When a user want to run some task, his task can use 1 or more boards.

If his task uses more than 5 boards it means that his task will occupy more than one emulator.

The task can occupy only sequential boards. For example if task A needs 3 boards it can occupy boards 1,2,3 or 2,3,4, but it cannot occupy 1,3,4. So for example if our current state is that we have a task that runs on emulator on board 3.

|  |
| --- |
| Board 1 |
| Board 2 |
| Board 3 – Task A |
| Board 4 |
| Board 5 |

and now a new task comes in, that need 3 boards, it cannot run in the same emulator unit as the previous task, because of the constraint that all boards must be occupied sequentially. So we will use two emulators.

|  |
| --- |
| Board 1 |
| Board 2 |
| Board 3 – Task A |
| Board 4 |
| Board 5 |

But of course if task A was on board 1, we could use only one emulator:

|  |
| --- |
| Board 1 – Task A |
| Board 2 – Task B |
| Board 3 – Task B |
| Board 4 – Task B |
| Board 5 |

For some of the tasks there is a constraint that their first board must be with some specific index. An example of this would be that task C must use 2 Boards and start from Board 2

|  |
| --- |
| Board 1 |
| Board 2 – Task C |
| Board 3 – Task C |
| Board 4 |
| Board 5 |

When employee requests time to use the machine he also states for how long he needs to use it. He states the needed duration in hours.

A process of requesting time slot is an iterative on-going process, when at some point employee 1 asks to run a task for X hour, then employee 2, and so on.

The goal is to maximize number of tasks that will run at each period of time.

**General Problem Description – Multiprocessor task scheduling**

We can look at the Emulator problem as a multiprocessor task (job) scheduling problem (also referred as parallel machine scheduling problem), where each board is a processor, and a task can run simultaneously on multiple boards. The constraint that some jobs can start only from set of boards that has specific index value, classify the problem as machines (processors) eligibility constraints problem.

\*Machine and processor are used interchangeably. Task and job also used interchangeably.

**Deterministic problem description**

In deterministic problem description we look at the current environment, free and occupied processors (boards), jobs that are already in process and jobs that still need to be scheduled, and try to optimize the objective function.

We have N jobs, and M parallel machines.

Job characteristics

* pj – Process time
* sizej – Size of job j
* aj – Availability of job, earliest time it can start running.
* dj – End time of job, the latest time at which the job must finish running
* emj – set of eligibile machines that the first board of the task can start on. We will include here size constraint, i.e. not to include machines that the size of the job is bigger than number of possible consecutive machines, if we have 15 consecutive machines, and our job needs to run on 6 machines, machines 10-15 will not be included in this set.

Optimality Criteria

The optimality criteria that we are looking for is to minimize the makespan. Which is to minimize the completion time of the last job. This criteria is chosen since it has a strong relation to processors utilization criteria. We can describe the total *allocated capacity* as the (number of machines multiply the time they are used). The actual *needed capacity* is . So we can look at utilization as , needed capacity is fixed, and the allocated capacity is co-related with makespan. So we can see that minimizing the makespan will minimize the allocated capacity and hence minimize processors utilization.

**Mip formulation**

Parameters

T – Maximum time for which our model will try to find optimality, i.e. t = 1 ... T.

Indexes:

j = 1 … N

m = 1 … M

t = 1 … T

Decision variables:

xjmt – Equals 1 if job j starts the process on machine m, at time t, 0 otherwise.

Cmax – Completion time of the last job.

Subject to

Constraint (1) forces that all the jobs will start process at some period of time, and on the processor that they can start work on. (2) Enforces that at time t, and on board m only one job j can run. (3) Takes care that all jobs will end before their due date. (4) Ensures that all jobs will not start running before their availability time. (5) Sets the Cmax equal to completion of the last job.