Name - Krishna Biswakarma.

Roll: 24CS60R71

Date: 24/10/24

Subject: ADA.

① a) P-processor scheduling problem

$T = \{T_1, T_2, T_3, \ldots, T_n\}$.

$t_1\ t_2\ \ldots\ t_n$

Goal: Find the minimum processing time.

Let;

Total execution time on processor 'K' be d_K.

Total time across all processor = D

where $D = \max(d_1, d_2, \ldots)$

Now, to show P-PSP is an optimization problem we need to show it is NP.

A problem is NP iff,

i) If can be verified in polynomial time (optimization).

ii) The decision problem can be solved non-deterministically in polynomial time.

Decision problem: Can the set of tasks be scheduled on 'p' processors such that total time on longest processor is less than or equal to the given time.

$$\max(d_1, d_2, \ldots, d_p) \leq D.$$

Optimization problem: If the sol" exists (assignment of tasks to processors), for each processor K execution time d_K.

$$d_K = \sum_{i \in T_K} t_i$$

Calculating sum of task on each processor and finding max can be done in $O(n)$ time.

∴ Optimization problem is in NP.

b) Showing
prove it

In case of

Items
Weight
Value
Capacity

Our goal
the tota

Consider

For each
Knapsack
Task ex
Time Bo

Now our
be part
on each

Correct
whethe

∵ Kn
it to

b) Showing reducibility of p-PNP from 0/1 Knapsack to prove it is a NP-Hard problem.

In case of 0/1 Knapsack.

Items: $I_1, I_2, \ldots, I_n$

Weight: $w_1, w_2, \ldots, w_n \longrightarrow w[\,]$

Value: $v_1, v_2, \ldots, v_n \longrightarrow v[\,]$.

Capacity of 'C'.

Our goal is to maximize the total value of the knapsack s.t. the total weight of selected items $\leq C$.

Consider p-PSP with $p = 2$ processors.

For each task t_i in PSP, corresponds to an item in 0/1 Knapsack.

Task execution time: $t_i = w_i$ (weight of ith item).

Time Bound $D = C$ (Knapsack capacity)

Now our task is to decide whether the set of task T can be partitioned across 2 processors such that total time on each processor does not exceed D.

Corresponds to the knapsack problem of determining whether a subset of items can have total weight $\leq C$.

∴ Knapsack problem is NP Hard and we can reduce it to p-PSP hence p-PSP is also NP-Hard.

c) Algo approximation:

One common approximation for P-PSP-

i) LSA
ii) Sort tasks in desc order of execution time $t_1, t_2, \ldots, t_n$
iii) Assign each task to the processor with the current load.

(i) Approximation Bound:

This algo has approx bound of 1. i.e. its solution obtained by LSA is atmost twice the optimal sol^n

$$D_{approx} \leq 2 \cdot D_{opt}$$

Complexity.

Sorting $\rightarrow O(n \log n)$.

Assigning each task to a process $\rightarrow O(n \log p)$

We can use priority queue to keep track of current load on each processor.

$\therefore T(n) = O(n \log n + n \log p)$.

d) The basic idea is to perform Binary search over feasible value of D and for each candidate 'D', check whether it is feasible the assign the tasks to processor s.t. no processor exceeds this load.

Steps:

1. Binary Search
- The L.B. of D is the longest single task time. $\max(t_1, t_2, \ldots, t_n)$.
- The U.B of D is the sum of tasks $\sum_{i=1}^{n} t_i$

2. Feasibility check:

For each candidate D try to assign tasks to processor using strategy. Assign each each task to the current processors and of adding the tasks to group D assign to new processors.

If you can ass
processor exce

g.s logic.

If the candid
'd'

If its not feas

Complexity.

Each feasibl
through the
g.s ta

Hence over

If you can assign all tasks within p processor and the processor exceeds the time limit D then its feasible.

B-S Logic.

If the candidate is feasible such the lower half for 'd'

If its not feasible such that the upper half for 'd'.

Complexity.

Each feasibility take O(n) because we can iterate through the task once and assign it to the processor.

B-S takes $O(\log \Sigma t_i)$

Hence overall TC $O(\log \Sigma t_i)$