Homework 3

Thomas Kim tsk389 51835

David Munoz dam2989 51840

CS331 Algorithms and Complexity

Problem Q1. Prove each of the following heuristics to be ideal or not ideal for scheduling jobs on a machine while minimizing:

$$\sum_{i=1}^{n} w_i C_i$$

1. Smallest time first.
2. Most important first.
3. Maximum $\frac{w(i)}{t(i)}$.

Suppose jobs are expressed as tuples (w_k, t_k) where w_k is the weight of the job and t_k is the time that job takes.
Claim: Smallest time first is **not** ideal.

Proof. Suppose the following set of jobs is scheduled using smallest time first.
$\{j_1 = (1, 2), j_2 = (3, 3)\}$
The smallest time first heuristic would schedule j_1 before j_2.
Thus, j_1 would finish at time 2 and j_2 would finish at time 5.
This results in a total cost of 17.
Suppose j_2 is scheduled before j_1.
j_2 would finish at time 3 and j_1 would finish at time 5.
This results in a total cost of 14.
$14 < 17 \implies$ the smallest time first heuristic is not ideal. □

Claim: Most important first is **not** ideal.

Proof. Suppose the following set of jobs is scheduled using most important first.
$\{j_1 = (2, 3), j_2 = (1, 1)\}$ The most important first heuristic would schedule j_2 before j_1.
Thus, j_1 finishes at time 3 and j_2 finishes at time 4.
The total cost is therefore 10
Suppose j_2 was scheduled before j_1.
j_1 would finish at time 4 and j_2 would finish at time 1.
The total cost is therefore 9.
$9 < 10 \implies$ the most important first heuristic is not ideal. □

Claim: Maximum $\frac{w(i)}{t(i)}$ first **is** ideal.

Proof. Let the queue of jobs be represented by a set of jobs $\{j_1, j_2, \ldots, j_n\}$ where j_k is scheduled before j_c iff $k < c$.
Suppose there exists an inversion between adjacent jobs in the scheduling queue.
This means that for some k, $\frac{w_k}{t_k} < \frac{w_{k+1}}{t_{k+1}}$.
This means that $w_k t_{k+1} < w_{k+1} t_k$. **(1)**
Base case: The inversion exists between jobs j_1 and j_2.
$w_1 t_1 + w_2(t_1 + t_2) > w_2 t_2 + w_1(t_1 + t_2)$

$w_1t_1 + w_2t_1 + w_2t_2 > w_2t_2 + w_1t_1 + w_1t_2$
$w_2t_1 > w_1t_2$ is true by equation (1)
Induction hypothesis:
$w_{n+1}\left(\sum_{i=1}^{n+1} t_i\right) + w_{n+2}\left(\sum_{i=1}^{n+2} t_i\right) > w_{n+2}\left(\sum_{i=1}^{n} t_i + t_{n+2}\right) + w_{n+1}\left(\sum_{i=1}^{n+2} t_i\right)$
$w_{n+2}\left(\sum_{i=1}^{n+2} t_i\right) - w_{n+2}\left(\sum_{i=1}^{n} t_i + t_{n+2}\right) > w_{n+1}\left(\sum_{i=1}^{n+2} t_i\right) - w_{n+1}\left(\sum_{i=1}^{n+1} t_i\right)$
$w_{n+2}\left(\sum_{i=1}^{n+2} t_i - \sum_{i=1}^{n} t_i - t_{n+2}\right) > w_{n+1}\left(\sum_{i=1}^{n+2} t_i - \sum_{i=1}^{n+1} t_i\right)$
$w_{n+2}\left(t_{n+1} + t_{n+2} - t_{n+2}\right) > w_{n+1}t_{n+2}$
$w_{n+2}t_{n+1} > w_{n+1}t_{n+2}$
This is true by equation (1).

□

Problem Q2(a). Briefly explain which steps of Dijkstra's algorithm fail when a graph can have negative weight edges.

Trivial cheating answer:
Suppose some connected graph has a negative edge. The lowest-cost path between any two nodes is to infinitely cycle along the negative edge.
Dijkstra's algorithm only explores each node once, and therefore can never find the lowest cost path.
Actual answer:
Dijkstra's algorithm relies on a breadth-first search using a priority queue and stops execution when the goal is reached.
The stopping of execution when the goal has been reached relies on the invariant that the current path is the shortest path found so far, and that any other paths will either increase or stay the same in cost if explored.
However, negative-weight edges can result in the cost of a path being explored to decrease, violating this invariant.
This problem can be trivially solved by simply running BFS until all vertices are explored.

Problem Q2(b). Given a set of 4-tuples (s_i, d_i, p_i, q_i), where each entry corresponds to source, destination, departure time, arrival time respectively, calculate the plan that arrives at d_i as early as possible, while allowing for at least 1 hour for each connecting flight.

Problem Q3(a). Give an algorithm to find the minimum-product spanning tree.

```
1: Let $G(V, E)$ be the graph in question
2: for $\forall e \in E$ do
3:     $weight(e) = ln(weight(e))$
4: end for
5: Run Prim's algorithm on the new graph $G'$
```

Problem Q3(b). Prove that if the weights of an undirected graph are unique, there exists a unique minimum spanning tree.

Problem Q3(c). Let $\mathcal{T}_G$ be the set of minimum spanning trees for some graph G. Let T be a spanning tree for G. Prove that:

$$\forall e \in T, e \in T' \in \mathcal{T}_G \implies T \in \mathcal{T}_G$$