CS F422 Parallel Computing

Project – Exercises

Exercise III

Consider the algorithm design technique "Branch-and-Bound":

- *When it is difficult to design an algorithm that constructs the optimal solution to an input instance of a problem, one can fall back on "searching" for solutions; searching is done by trying out permutations / combinations of components of a solution until one of the permutations / combinations turns out to be the optimal solution*.
  - For instance, if the problem is to find an optimal schedule of jobs on machines in a shop, one can "assign" jobs to each machine and keep track of the current best solution.
- *To ensure that the search process is systematic, one starts out with an empty solution and construct a tree where branches (or edges) correspond to decisions and states (or vertices) correspond to (partial) solutions.*
  - For instance, in the job scheduling problem: one can work by assigning jobs to one machine at a time: each job assigned to a machine results in a different branch and each state (i.e. a partial solution) is a subset of jobs which have been assigned (to some machine or other).
- *The tree is traversed - as it is constructed - in a depth-first-order i.e. one path at a time;*
  - For instance, in the scheduling problem, a path would assign one job at a time (to some machine)
- *If a path leads to a dead-end i.e. a state that is (internally) inconsistent, then we backtrack i.e. go back to a "previous" decision and try an alternative path:*
  - This case may happen in the scheduling problem if multiple jobs are assigned to the same machine to be executed at the same time.
- *Each (full or partial) solution has a measure that is being optimized.*
  - For instance, in the scheduling problem, the measure could be the **maximum makespan** across all machines. Makespan is the time taken by a machine to complete all the jobs assigned to it. The problem is to ***minimize the maximum makespan.***
- *A feasible solution is a full solution not necessarily optimal.*
  - For instance, in the scheduling problem, a feasible solution includes assignment of all jobs (to some machine or other).
- *The measure of the best feasible solution so far is tracked. When the measure of any partial solution becomes worse than that of the known best feasible solution we can prune that part of the tree i.e. the measure of the known best feasible solution can be used as a bound on the solutions and paths can be pruned if the bound is exceeded. This also results in backtracking.*
  - For instance, in the scheduling problem, the makespan of the known best schedule is used as a bound i.e. partial schedules with makespan larger than the known best makespan need not be traversed (i.e. constructed) further.
- *An optimal solution may be found only after the entire tree has been traversed (or parts of it have been pruned and the rest traversed) unless there is a test for optimality.*
  - For instance, in the scheduling problem, there is no test for optimality. [Although 100% utilization would be optimal, a particular instance may not admit 100% utilization – e.g. jobs may have specific start times.]

[*read* **Goodrich and Tamassia, Algorithm Design** *for more details if you are not familiar with the* *technique*.]

A parallel algorithm for branch-and-bound would explore (i.e. construct) multiple paths in parallel. As soon as a path leads to a solution whose measure exceeds the known best measure exploration of that path can be stopped.

Design and implement a template parallel algorithm that schedules path computations given p nodes, with c cores per node. Note that scheduling has to be dynamic because paths may evolve at uneven rates.

Demonstrate your template with full solution for at least two example problems and at least one of which exhibits uneven rates of growth. Design your own interface for assigning computation and data to your tasks. Your interface has to be simpler and at a higher level than that of MPI.

Your implementation must use MPI for message passing and either PThreads or OpenMP for shared memory computations.

Example problems:

- Travelling Salesperson Problem
- A simple 2-player board game (more complex than Tic-Tac-Toe and much simpler than Chess)
- MAX-SAT (Given a Boolean formula in conjunctive normal form, find an assignment that satisfies the maximum number of clauses.)
- Facility Location Problem