## Convenience changes for algorithm:

1. Now that we have a better understanding of what arguments will be used where, should something like depth be put into the constructor, and should we remove the heuristics from the base algorithm class?

We could make something extending Boundable called GenericBoundableAlgorithm which has all the heuristics code, where Generic is meant to be inherited from for all normal algorithms.

Move Arborist and LowerBound from algorithm into

1. Change name from start to run because it will be blocking?
2. If we are passing schedules in to the algorithm now, couldn’t we drop the processors argument and just check the number of processors in the schedule?
3. Remove depth argument from default boundable? We could reason that we could allow any given algorithm implementation to choose the best time for it to hand off a partial schedule.
4. Get rid of onSolutionFound
5. Remove isComplete
6. Simplify MultiAlgorithmNotifier dummy
7. Simplify terrible schedule dummy
8. ~~USE COMPARE AND SWAP AAAAAA~~
9. ~~Move globalBest into MultiAlgorithmNotifier, change to MultiAlgorithmCommunicator? Makes changes of 9., and 8. easy~~

# Things to do:

* ~~Convert DFS to boundable~~
  + ~~Make sure it uses the notifier to call relevant methods~~
* Write Tiered
  + ~~Write thread method (method called in each thread)~~
  + ~~Write onSolutionFound~~
  + ~~Write notifyPartialSolution~~
  + ~~Finally, write Start.~~
* Fixes and edits
  + ~~Change name of start to run~~
  + ~~Drop processors argument~~
  + Drop depth argument
  + ~~Fix algorithm tests (ie make DFS runnable as it was beforehand)~~
  + Add comments tieredAlgorithm

# Separate DFS Clean-up:

Separate into methods:

-placeLast – placing the last node in the schedule

-getEarliestTimes – Get times on each processor for placing node

-calculateNextNodes – calculate the list of next available nodes when adding the given