Ch. 12 – Coping with limitations of algorithm power

* Tackling difficult combinatorial problems (NP-hard problems)
  + Strategy that guarantee to find a solution but may not be in polynomial time.
  + Strategy that find a sub-optimal solution in polynomial time.
* Exact solution strategies
  + Exhaustive search
  + Dp: dynamic programming (everyone hates it, sooooo fuckin difficult)
  + Backtracking (wut the hell is dat? Never heard of it)
    - Eliminates some unnecessary cases from consideration
    - Yields solutions in reasonable time for many instances but worst case is still exponential
  + Branch-and-bound
    - Further refines the backtracking idea for optimisation problems

12.1 Backtracking

* Construct the state-space tree
  + Nodes: partial solutions
  + Edges: choices in extending partial solutions
* Explore the state space tree using depth-first search
* “prune” non-promising nodes

12.2 Branch-and-Bound

* An enhancement of backtracking
* Applicable to optimisation problems
* For each node of a state-space tree, compute a bound on the value of the objective function for all descendants of the node
* Uses the bound for
  + Ruling out certain nodes as “non-promising” to prune the tree – if a node’s bound is not better than the best solution seen so far
  + Guiding the search through state-space

12.3 Approximation Algorithms for NP-hard Problems

* NP—hard problems: problems that are as hard as or harder than NP—Complete problems.
* Accuracy ratio of an approximate solution Sa
  + R(Sa) = f(Sa)/f(S\*) for minimisation
  + R(Sa) = f(S\*)/f(Sa) for maximisation
  + Where f(Sa) and f(S\*) are values of the objectives function f for the approximate solution Sa and actual optimal solution S\*
* Performance ratio of the algorithm A
  + The lowest upper bound of R(Sa) on all instances
* Theorem: if P != NP there exists no approximation algorithm for TSP with a finite performance ratio.

12.4 Algorithms for Solving Non-linear Equations

* Bisection: two guesses f(a) and f(b) must have different sign
* Newton’s Method: make sure that the initial guess cannot be too far away from the solution; otherwise it will be diverge.
* Regula falsi: similar to bisection but uses x-intercept instead of mid-pt. It converges faster than bisection but slower than Newton’s method.