2.098/15.093J: Recitation 9

Xuan Vinh Doan,

November 10, 2004

1 Heuristic Methods

Integer programming problems in general are hard problems, which cannot be easily solved to optimality. Heuristic methods allow us to obtain good feasible solutions with a reasonable amount of computational time. Local search heuristics are widely used. General framework for local search is as follows.

1. Start with a feasible solution.
2. Identify the neighbourhood.
3. Move to a 'better' feasible solution within the neighbourhood.

Generally, the local search heuristics will result in a local optimum, which is sometimes the global optimum. The movement in step 3 is method dependent and maybe problem dependent.

1.1 Simulated Annealing

Neighbours are selected randomly with probability p_n, where $\sum_{n\in N} p_n = 1$. The decision to move depends on the objective cost and the annealing probability:

$$p_{xy} = \begin{cases} p_y(x)e^{-\frac{c(y)-c(x)}{T}} & c(y) > c(x) \\ p_y(x) & \end{cases}$$

The temperature schedule T is selected based on some rules, such as $T(t) = \frac{C}{\log t}$ or $T(t) = Ca^t$, where $a < 1$.

1.2 Other methods

There are many local search heuristics out there (some are beter than other for some specific problems). Among them are tabu search, genetic algorithm (a little bit different approach). Ant colony optimization is another one.

1. Tabu search elements: tabu list (list of moves or characters of moves that have been used), intensification (finer neighbourhood), and diversification (coarser neighborhood).
2. Genetic algorithm elements: chromosome (representation of solution), selection, crossover, and mutation.
3. Ant colony optimization: pheromone trail and heuristic desirability.

2 Dynamic Programming

2.1 Dynamic Programming Elements

1. State x_k, the most important element
2. Control $u_k \in U(x_k)$
3. Randomness w_k
4. Dynamic $x_{k+1} = f_k(x_k, u_k, w_k)$
5. Additive cost $E_W \left(g_N(x_N) + \sum_{i=1}^{N-1} g_k(x_k, u_k, w_k) \right)$

2.2 Bellman's Principle of Optimality

Here we want to find the total minimum cost over a period of N stages in the future. The cost of the final stage is $g_N(x_N)$. At a stage k with the state x_k, we need to decide on how much control u_k we need so that the total cost from that stage to the final one is optimal. The recursion formula is then:

$$J_k(x_k) = \min_{u_k \in U(x_k)} \{ E_{w_k} \left(g_k(x_k, u_k, w_k) + J_{k+1}(f_k(x_k, u_k, w_k)) \right) \}$$

The total minmum cost is $J_0(x_0)$, which can be calculated backwards recursively.
In general, a recursive formula needs two elements:

1. Initial condition $f_0 = f$
2. Recursive formula $f_{k+1} = g_k(f_k)$

2.3 Examples

Knapsack problem
Two recursive formulations with different state definitions:

1. $F(w)$ is the optimal cost with w as the weight limit, then we need to find $F(W)$, where:
   (a) $F(w) = 0$ if $w < \min_i w_i$
   (b) $F(w) = \max_{i=\overline{1,m}} \{F(w - w_i) + p_i\}$ if $w \geq \min_i w_i$
2. $F_i(w)$ is the optimal cost with w as the weight limit while using only the first i items, then we need to find $F_m(W)$, where:
   (a) $F_0(w) = 0$ if $w \geq 0$, $F_0(w) = -\infty$ if $w < 0$
   (b) $F_{i+1}(w) = \max \{F_i(w), F_i(w - w_{i+1}) + p_{i+1}\}$ if $w > 0$