<3. Shortest Path Problem and Other Applications>

- 목차
    - 자원 할당 문제
        - 문제 정의
        - DP formulation
    - knapsack Problem
    - Production Control
        - 목적
        - 설명
        - Integer Programming
        - DP formulations
        - 제안 3.2
        - DP revision
    - Product Control with Backlogging
        - 제안 3.3
        - DP formulation
    - Capacity Expansion
        - 목적
        - 설명.
        - Integer Programming
        - DP formulation

* Resource Allocation 문제 정의

- 정수 K개 만큼 자원 이용 가능, 자원은 N개의 상품으로 할당.
- n번째 상품을 x_n 단위로 생산하려면 $C_n(x_n)$의 자원이 필요함.
- x_n 단위 상품 생산했을 때 $P_n(x_n)$ 만큼 수익을 얻음.
- 개개별의 상품들은 B 단위 이상으로 생성할 수 없음.
- 자원을 잘 할당해서 $\Sigma P_n(x_n)$을 최대화하는게 목표
- 다음과 같이 나타냄.

$$\max \; P_1(x_1) + P_2(x_2) + \cdots + P_n(x_n)$$

$$\text{s.t. } C_1(x_1) + C_2(x_2) + \cdots + C_n(x_n) \le K$$

$$0 \le x_n \le B$$

* Resource Allocation – DP로 표현하기

- $n = N$, $V_N(y_N) = \max\limits_{\forall x_N,\, C_N(x_N) \le y_N,\, 0 \le x_N \le B} P_N(x_N)$

$1 \le n \le N-1$, $\max\limits_{\forall x_n,\, C_n(x_n) \le y_n,\, 0 \le x_n \le B} P_n(x_n) + V_{n+1}(y_n - C_n(x_n))$

- y_n : 1부터 n-1 까지 거치고 남은 자원의 양
- $V_n^*(y_n)$: n부터 N 까지 자원이 y_n 만큼 할당하여 얻을 수 있는 최대 수익.
-

* 만약 모든 자원들이 다 소모되어야 한다면?

- boundary condition 에 대해서, $V_N(y_N) = P_N(x_N)$, when $C_N(x_N) = y_N$ and $0 \le x_n \le B$
- otherwise, $V_N(y_N) = -\infty$

* Knapsack problem.

- return 함수와 consumption 함수가 선형인 resource allocation 문제.
- Notation은 다음과 같음.
    - k : 컨테이너의 부피
    - N : 상품 종류의 개수
    - C_n, P_n : n번째 상품의 부피 & 가치 (C_n은 양의 정수, $1 \le n \le N$)
- 컨테이너 안의 상품들의 전체 가치를 최대화하는 것이 목표.
- Integer Programming
    - $\max \; p_1 x_1 + \cdots + p_N x_N$

      s.t. $C_0 x_0 + C_1 x_1 + \cdots + C_N x_N = k$, $x_n \ge 0$, $\forall n = 1, \ldots, N$
- DP로 표현하면 다음과 같음.
    - Bellman Equation : $V(j) = \max\limits_{\forall n,\, C_n \le j} \{ P_n + V(j - C_n) \}$, $\forall j = 1, \ldots, k$
    - j : 현재 남은 부피, $j = 0, 1, \ldots, k$
    - $V^*(j)$: 현재 남은 부피 j에서 가능한 최대 가치

* Production Control의 목적

- 다중 기간 동안 최소 비용으로 수요 충족

* Production Control 설명

- 수요(d_t)는 $N-1$ 기간 동안 발생하며, 이전의 생산량으로 충족될 수 있음. (백로깅 ×)
- $t=1$ 일 땐 재고는 0 이며, $t=N$일 때 재고는 0이 되어야 함.
- $C_t(x)$: t 기간동안 x 단위 생산비용.
- $h_t(y)$: t 기간 동안 y 단위 재고 유지비용.
- t시기 초기에, x_t와 I_t가 결정됨.

* Production control 수리적 표현

- $\min \sum_{t=1}^{N-1} \{ C_t(x_t) + h_t(I_t) \}$

x_t 만큼 생산하는데 드는 비용

I_t 만큼 유지하는데 드는 비용

$I_1 = I_N = 0$, I_t : 정수 $(t = 2, 3, \dots)$

$I_t + x_t = d_t + I_{t+1}$, $\forall t = 1, 2, 3, \dots$, x_t 는 정수

처음과 마지막에는 재고가 없어야 함.

* Production control - DP로 표현

- t부터 N-1 까지는 현 시점의 I_t (재고 수준)에 따라 달라지며, 0~t-1은 영향을 주지 않음.
- $V^*_t(I_t)$는 t~N-1 까지의 수요를 충족하는 최소비용이며, $h_t(I_t)$와 $C_t(x_t)$를 포함하고 있음.
- $t = N$, $V_N(I_N) = \begin{cases} 0 & \text{if } I_N = 0 \\ \infty & \text{if } I_N > 0 \end{cases}$

이러면 최악인거지...

$t < N$, $V_t(I_t)$

$$= \min_{\underline{d_t \le I_t + x_t \le d_t + d_{t+1} + \cdots + d_{N-1}}} h_t(I_t) + C_t(x_t) + V_{t+1}(\underbrace{I_t + x_t - d_t}_{I_{t+1}})$$

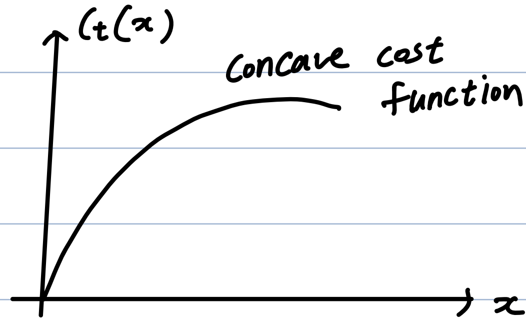
* Production Control - 제안 3.2

- 오목함수 형태의 생산·유지 비용에 대해서,

  $\underbrace{I_t x_t}$ = 0을 만족시키는 optimal production plan이 적어도 하나 존재.

  I_t : t시기에 재고 상황

  x_t : t시기에 생산한 재고량.



- 위 제안은 재고량이 0일 때만 생산이 발생함을 설명.
- $x_{i+1} = x_{i+2} = \cdots = x_j = 0$ 일 때, → i+1 부터 j 까지는 생산하지 않음.

  i 시기에 생산된 수량은 i~j-1 까지의 수요와 같아야 함.

  ↳ I_t가 0이 될 때 생산을 한다는 가정이 있으므로,

  $x_i = d_i + d_{i+1} + \cdots + d_{j-1}$.
- $d_{ij} = \sum_{k=i}^{j-1} d_k$
- 위 제안을 통해, DP 공식을 아래와 같이 수정 가능.
  - C_{ij} : i부터 j-1 까지 발생한 비용.
  - $C_{ij} = \underbrace{h_i(0)}_{\text{i시기엔 재고X}} + \underbrace{C_i(d_{ij})}_{\text{그래서 i시기에 } d_{ij}\text{를 생산.}} + \sum_{k=i+1}^{j-1} h_k(d_{kj})$ → i시기가 지나면 $d_{(i+1)j}$가 남고, 이를 유지해야 함.

* Production Control - DP 재구성.

- C_{ij} : i부터 j-1 까지 발생한 모든 비용 → 생산비용 + 유지비용.
- $C_{ij} = h_i(0) + C_i(d_{ij}) + \sum_{k=i+1}^{j-1} h_k(d_{kj})$로 정의할 때,

$$\begin{cases} i<N \text{ 일 때} & V_i = \min_{i<j\le N} C_{ij} + V_j \\ i=N \text{ 일 때} & V_N = 0 \end{cases}$$

- V_i^*는 $I_i = 0$이라 주어질 때 i부터 N까지 수요를 충족시키는 최소비용.
- V_1은 최적 생산 계획의 비용.

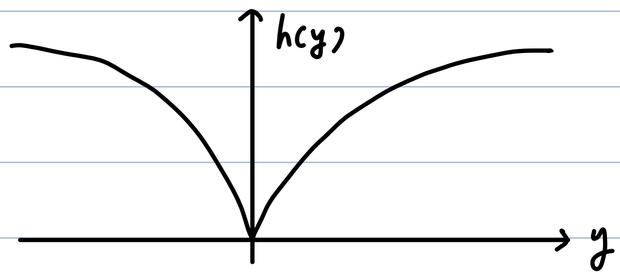
* Backlogging을 고려할 때 Product Control

- 백로깅을 고려하면, h_t는 유지비용 + 부족 비용으로 정의.
- DP로 표현하면 다음과 같음.

$t<N$ 일 때 $V_t(I_t) = \min\limits_{0 \le x_t \le -I_t + d_t + \cdots d_{N-1}} h_t(I_t) + C_t(x_t) + V_{t+1}(I_t + x_t - d_t)$

$t=N$ 일 때 $V_N(I_N) = \begin{cases} 0 & \text{if } I_N = 0 \\ \infty & \text{if } I_N > 0 \end{cases}$

- Concave cost function은 다음과 같음.

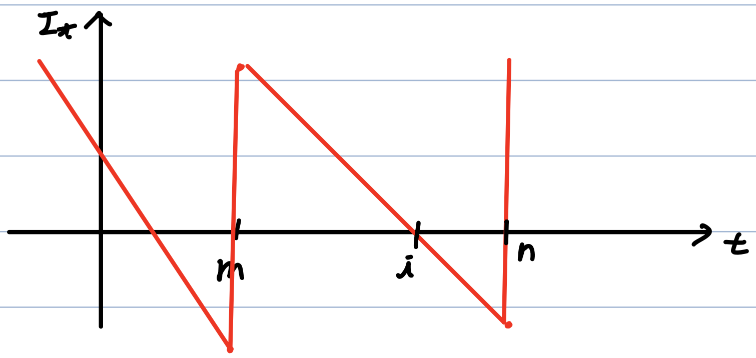


$h_t(y)$는 holding cost

$h_t(-y)$는 shortage cost.

• 제안 3.3

- 모든 비용이 concave 하고 $x_m, x_n > 0$ $(m>n)$, $I_i = 0$ $(m < i \le n)$ 할 때, 최적 생산 계획이 적어도 하나 존재한다.



- 위의 제안은 $i<j<N$ 이고 $I_i = I_j = 0$ 일 때, i 시기 때부터 $j-1$ 까지 $d_{ij} = \sum_{k=i}^{j-1} d_k$ 만큼 생산해야함.

* Backlogging 고려했을 때 DP 공식 구성.

- $C_{ij}(n) = h_i(0) + C_n(d_{ij}) + \sum_{k=i+1}^{n} h_k(-d_{ik}) + \sum_{k=n+1}^{j-1} h_k(d_{kj})$
  - i부터 j까지 발생한 모든 비용들의 총합.
  - i부터 j까지 발생한 모든 수요 d_{ij}는 n시기에 생산된 x_n에 의해 충족됨

- $i = N$일 때, $V_N = 0$

  $i < N$일 때, $V_i = \min_{i<j\le n} C_{ij}^* + V_j$

  - V_i^* : $I_i = 0$일 때 i부터 N까지의 수요들을 만족하는 최소 비용.
  - C_{ij}^* ; $\min_{n|i\le n\le j} C_{ij}(n)$ 이며, i부터 j까지 가장 싼 transition
  - V_1 : 최적 생산 계획에서의 비용.

    ↳ $t=1$일 때부터 최적으로 생산할 수 있으므로.

* Capacity Expansion - 목적

- 증가하는 수요를 감당하기 위한 최소 총 비용 기반 시설 확장 시기 및 규모를 계획

* Capacity Expansion 설명

- N년 연속 기간동안 전기수요는 d_t로 정의, $\forall t = 1, \dots, N$, $(d_1 < d_2 < \dots < d_N)$
- $t=1$에서의 초기 용량 x_1으로부터 확장 전략 결정.
- 모든 비용은 용량과 발전량이 높을 수록 덜 증가
- 수요를 충족시키면서 비용 최소화하는것이 목적.
- x_t : t 시기 초의 기존 발전소 용량.
- u_t : t 시기 초의 용량 확장 정도 (매년 건설할수 있는 최대 용량은 제한 있음).

* Capacity Expansion 수리적 표현

- $\min \sum_{t=1}^{N} \{ c(u_t) + f(x_t + u_t) + v(y_t) \}$
  - c : 건설 비용, f : 유지비용, v : 생산비용, (c, f, v 다 오목함수)
  - $y_t \geq d_t$

    $y_t \leq A \times (x_t + u_t)$ (x_t : 현재 공간, u_t : 확장하는 공간)

    $x_t + u_t = x_{t+1}$

    $0 \leq u_t \leq u_{max,t}$ maximum capacity to be constructed at year t.

* Capacity Expansion - DP 공식

- $t = N$일 때, $V_N(x_N) = c_N\left(\max\left(\frac{d_N}{A} - x_N, 0\right)\right) + f\left(x_N + \max\left(\frac{d_N}{A} - x_N, 0\right)\right) + v(d_N)$

  $t < N$일 때, $V_t(x_t) = \min_{\max\left(\frac{d_t}{A} - x_t, 0\right) \leq u_t \leq u_{max,t}} \left\{ c(u_t) + f(x_t + u_t) + v(d_t) + V_{t+1}(x_t + u_t) \right\}$
  - $V_1(x_1)$을 찾으면 최적의 확장 전략 가능.