Smart Technology - AIR

Optimization

What is Optimization?

- Optimization, Mathematical Optimization, Mathematical Programming
  - 주어진 제약을 만족하는 가능해 중에서, 의사결정자의 목적을 가장 극대화할 수 있는 해를 찾는 문제

- 최적화 문제의 예
  - 콜센터 직원의 스케줄 (휴가 계획, 균등한 야근 조율, 시간대별 수요 등)
  - 운송 계획 수립 (방문 목적지의 수, 운행 거리, 가용 시간 등)
  - 투자 포트폴리오 (가용한 예산, 투자 대상의 위험율 등)

- Data 기반의 방법론
  - 최적의 클러스터 찾기 (중심점부터 각 객체 까지의 거리 최소화)
  - 예측 모델의 최적의 Parameter 찾기

What is Operations Research?

- 경영과학, 운영과학 (Operations Research: OR) 은 복잡한 운영체계를 설계하고 운영하기 위한 수리적 모델 기반의 학문
- 2차 세계대전 기간에 군사 운영의 문제를 해결하기 위한 방법으로 크게 발전
  - 정량적인 기법을 이용하여 한정된 군 자원을 가장 효과적으로 사용하기 위한 방법
- 2차 세계 대전 이후, 많은 기업들이 시장 경쟁력 향상을 위해 OR을 활용함
- OR은 완벽한 정보를 갖고 있지 않은 상황에서 의사결정을 위한 합리적인 근거를 제시함 (시스템의 행동을 묘사하고, 이해하고, 예측하는 과학)

Mathematical Programming

- Mathematical Programming (Mathematical Optimization)
  - One of the most powerful techniques used in OR
  - Sometimes both terms are uses interchangeably

- 최적화 수리 모델의 구성요소
  - 의사결정변수(Decision variables): amounts of either inputs or outputs
  - 파라미터(Parameters): numerical values
  - 목적함수(Objective Function): mathematical statement of profit or cost for a given solution
  - 제약식(Constraints): limitations that restrict the available alternatives

- 최적화 수리모델의 일반적 형태

$$
\begin{array}{lr}
\textit{min} \text{ or } \textit{max} & f(x_1, \ldots, x_n) \\
s.t. & g_i(x_1, \ldots, x_n) \begin{Bmatrix} \leq \\ = \\ \geq \end{Bmatrix} b_i \\
 & x \in X
\end{array}
$$

Mathematical Programming

- A solution is an assignment of values to variables.
- A feasible solution is an assignment of values to variables such that all the constraints are satisfied.
- The objective function value of a solution is obtained by evaluating the objective function at the given solution.
- An optimal solution (assuming minimization) is one whose objective function value is less than or equal to that of all other feasible solutions.

Mathematical Programming

- Mathematical Programming (Mathematical Optimization)

$$
\begin{array}{lll}
\min \text{ or } \max & f(x_1, \ldots, x_n) & \\
\text{s.t.} & g_i(x_1, \ldots, x_n) \left\{ \begin{array}{c} \leq \\ = \\ \geq \end{array} \right\} b_i & \\
& x \in X &
\end{array}
$$

  - Deterministic optimization: parameters are known value
  - Stochastic optimization(SO): some parameters are random variable
  - Robust optimization(RO): some parameters are unknown but within a set

  - There are some other probabilistic OR models including Markov chain, queuing theory and simulation. They are not optimization technique, rather, they determine measures of performance of a system

Mathematical Programming

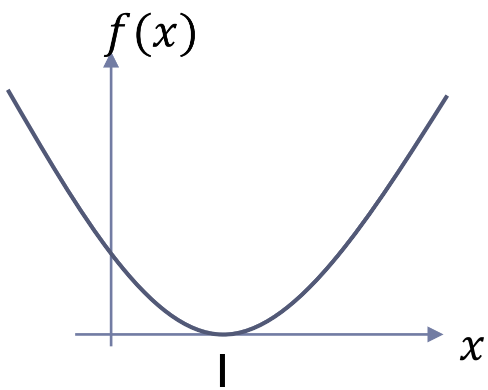
- Mathematical Programming (Mathematical Optimization)

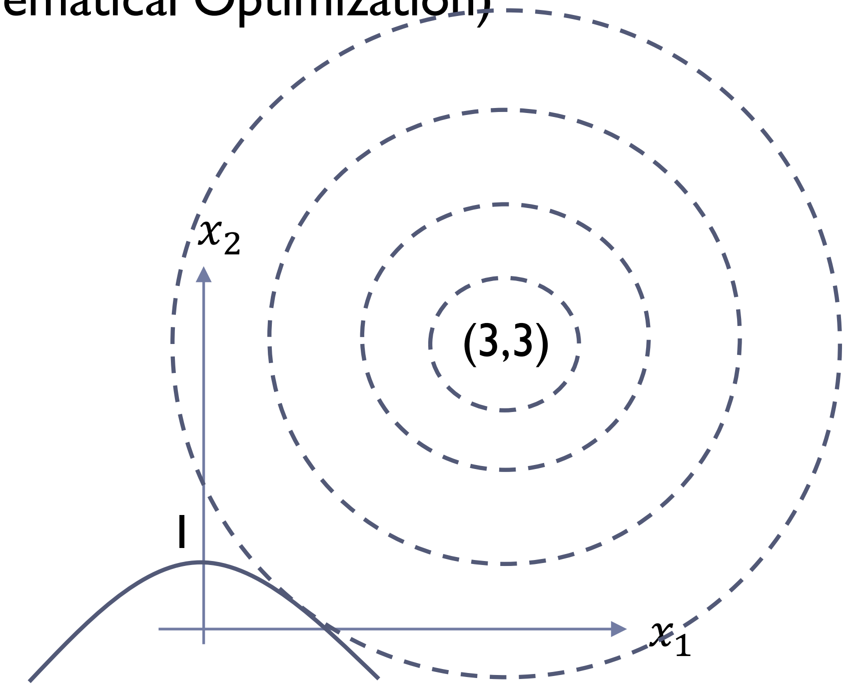
$$
\begin{array}{lr}
\min \text{ or } \max & f(x_1, \ldots, x_n) \\
s.t. & g_i(x_1, \ldots, x_n) \left\{ \begin{array}{c} \leq \\ = \\ \geq \end{array} \right\} b_i \\
 & x \in X
\end{array}
$$

- Linear Programming (LP) is an optimization technique for a system of linear constraints and a linear objective function
- Integer Programming (IP) is an optimization technique in which some of the variables are restricted to be integers
- Nonlinear Programming (NLP) is an optimization technique in which some of the constraints or objective function are nonlinear

Mathematical Programming

- Mathematical Programming (Mathematical Optimization)
  - Unconstrained vs. Constrained





$$\min_x f(x) = x^2 - 2x + 1$$

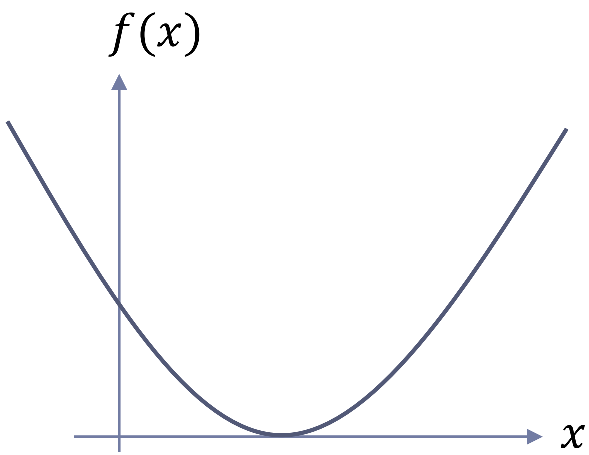
$$\min_x f(x) = (x_1 - 3)^2 + (x_2 - 3)^2$$

$$s.t.\ \ g(x) = x_1^2 + 3x_2 - 3 = 0$$

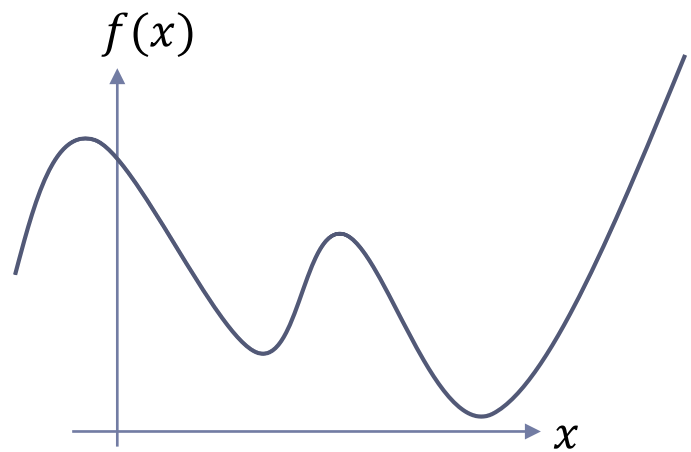
*Gradient: direction of fastest increase

Mathematical Programming

- Mathematical Programming (Mathematical Optimization)
  - Convex vs. Non-convex



Convex Optimization



Non-convex Optimization

* Gradient descent algorithm

$$x_{k+1} = x_k - r\nabla f\left(x_k\right)$$

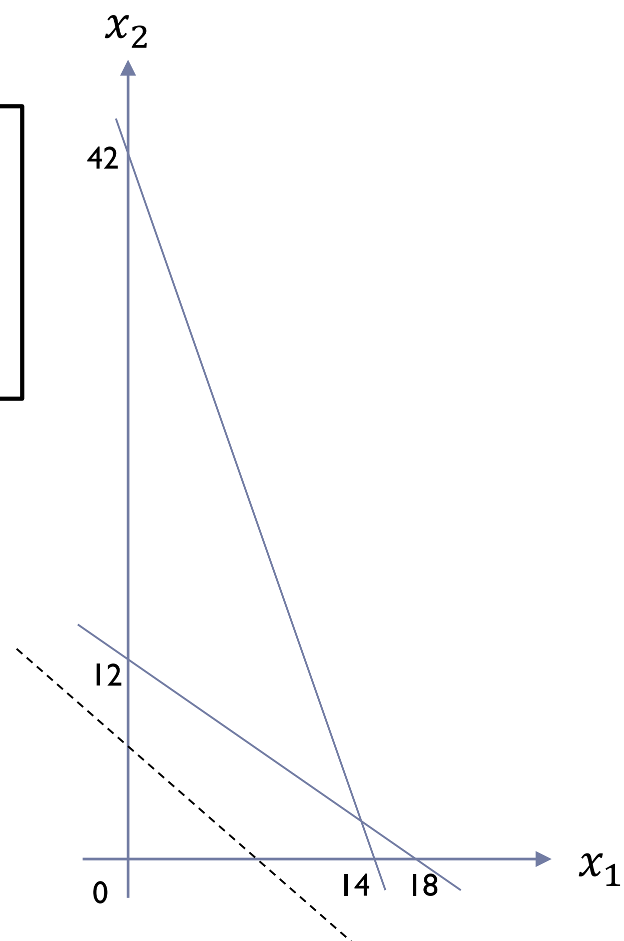
Mathematical Programming

- Mathematical Programming (Mathematical Optimization)
  - Convex vs. Non-convex

$$
\begin{array}{ll}
\textit{min} \text{ or } \textit{max} & f(x_1, \ldots, x_n) \\
\textit{s.t.} & g_i(x_1, \ldots, x_n) \begin{Bmatrix} \le \\ = \\ \ge \end{Bmatrix} b_i \\
& x \in X
\end{array}
$$

$$\max 150x_1 + 195\,x_2$$

$$
\begin{aligned}
\text{s.t. } 100x_1 + 150x_2 &\le 1800 \\
1.5x_1 + 0.5x_2 &\le 21 \\
x_1, x_2 &\ge 0
\end{aligned}
$$

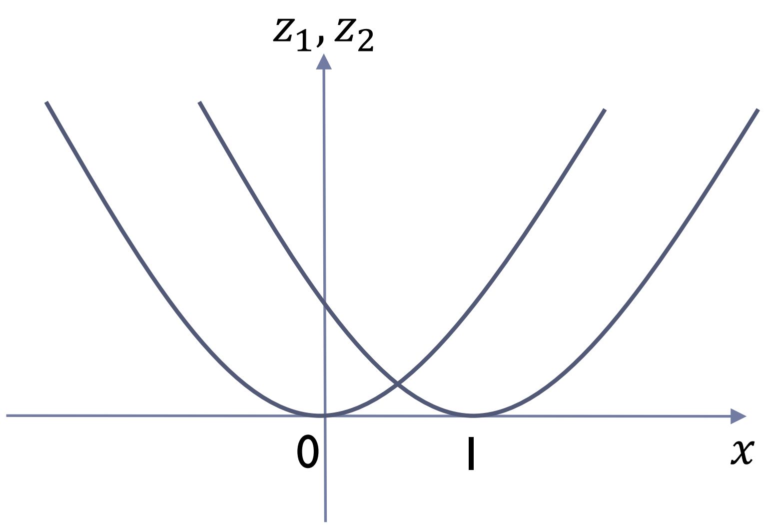


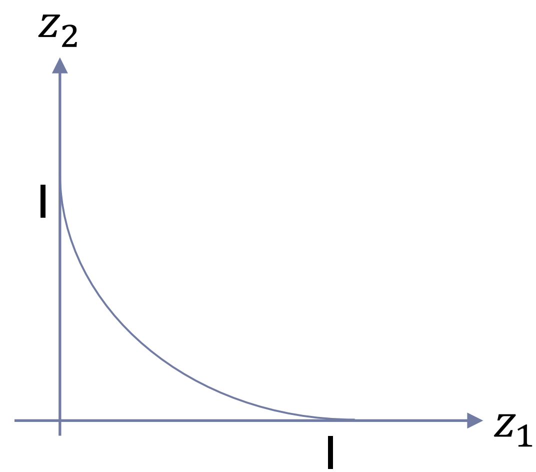
Mathematical Programming

- Mathematical Programming (Mathematical Optimization)
  - Single objective vs. Multi-objective

$$\min_x z_1 = x^2 - 2x + 1$$

$$\min_x z_2 = x^2$$





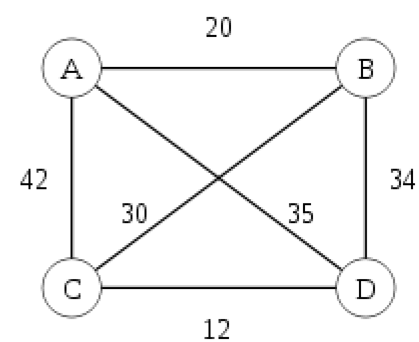
Objective Space

* Efficient frontier or Pareto optimal

Mathematical Programming

- Mathematical Programming (Mathematical Optimization)
  - Example) Traveling salesman problem: find the shortest route that visits each city and return to the original city
  - Decision variables: $x_{ij} = \begin{cases} 1 & \text{if city j is reached from city i} \\ 0 & \text{Otherwise} \end{cases}$

$$\begin{aligned}
\text{minimize} \quad & \sum_{(i,j)\in\mathcal{A}} c_{ij}x_{ij}, & (1.1)\\
\text{subject to} \quad & \sum_{i\in\mathcal{V}:(i,j)\in\mathcal{A}} x_{ij} = 1 \quad \forall j \in \mathcal{V}, & (1.2)\\
& \sum_{j\in\mathcal{V}:(i,j)\in\mathcal{A}} x_{ij} = 1 \quad \forall i \in \mathcal{V}, & (1.3)\\
& \{(i,j) \in \mathcal{A} : x_{ij} = 1\} & \\
& \quad \text{do not contain subtours,} & (1.4)\\
& x_{ij} \in \{0,1\} \quad \forall (i,j) \in \mathcal{A}, & (1.5)
\end{aligned}$$



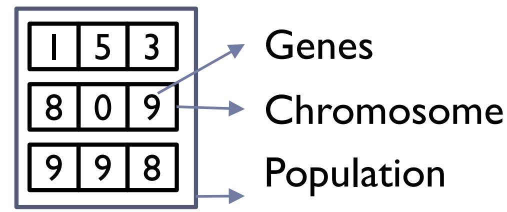
  - Nearest Neighbor Heuristic : Fine out the shortest edge
  - Metaheuristic: Tabu search, Simulated Annealing, Genetic Algorithm, Ant Colony Optimization, Particle Swarm Optimization

Mathematical Programming

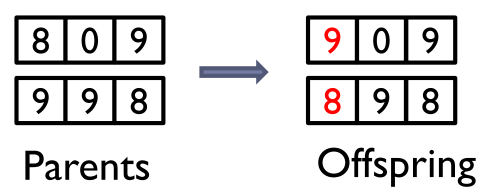
- The procedure of a generic GA

Step 1: Set $t = 1$. Randomly generate N solutions to form the first population, P_1. Evaluate the fitness of solutions in P_1.
Step 2: *Crossover*: Generate an offspring population Q_t as follows:
  2.1. Choose two solutions $\mathbf{x}$ and $\mathbf{y}$ from P_t based on the fitness values.
  2.2. Using a crossover operator, generate offspring and add them to Q_t.
Step 3: *Mutation:* Mutate each solution $\mathbf{x} \in Q_t$ with a predefined mutation rate.
Step 4: *Fitness assignment:* Evaluate and assign a fitness value to each solution $\mathbf{x} \in Q_t$ based on its objective function value and infeasibility.
Step 5: *Selection:* Select N solutions from Q_t based on their fitness and copy them to P_{t+1}.
Step 6: If the stopping criterion is satisfied, terminate the search and return to the current population, else, set $t = t + 1$ go to Step 2.

Choose 3 and make 20
{1,5,6,8,3,7,3,5,9,9,8,0}



Fitness: Difference from 20
f(1,5,3) = 11
f(8,0,9) = 3 → higher probability
f(9,9,8) = 6



LP Example

- 총과 버터 만들기
  - 목적: 가능한 많은 돈(매출) 벌기 – 버터 1톤 $150, 총 1개 $195
  - 의사결정변수: 수익을 최대화할 수 있는 총과 버터의 양
  - 제약식: 예산 $1800/month, 1톤의 버터 생산에 $100, 1개의 총 생산에 $150
    공간 21cubic 미터, 버터 1.5 cubic, 총 0.5 cubic

- 선형계획법 (Linear Programming)
  - 목적함수와 제약식이 모두 선형함수로 표현
  - 의사결정변수는 실수 구간에서 정의

$$
\begin{array}{ll}
\min \text{ or } \max & f(x_1, \ldots, x_n) \\
s.t. & g_i(x_1, \ldots, x_n) \left\{ \begin{array}{c} \leq \\ = \\ \geq \end{array} \right\} b_i \\
 & x \in X
\end{array}
$$

$$
\begin{array}{ll}
\min \ or \ \max \ c^T X & \\
\text{s.t.} & AX = b \\
 & X \geq 0
\end{array}
$$

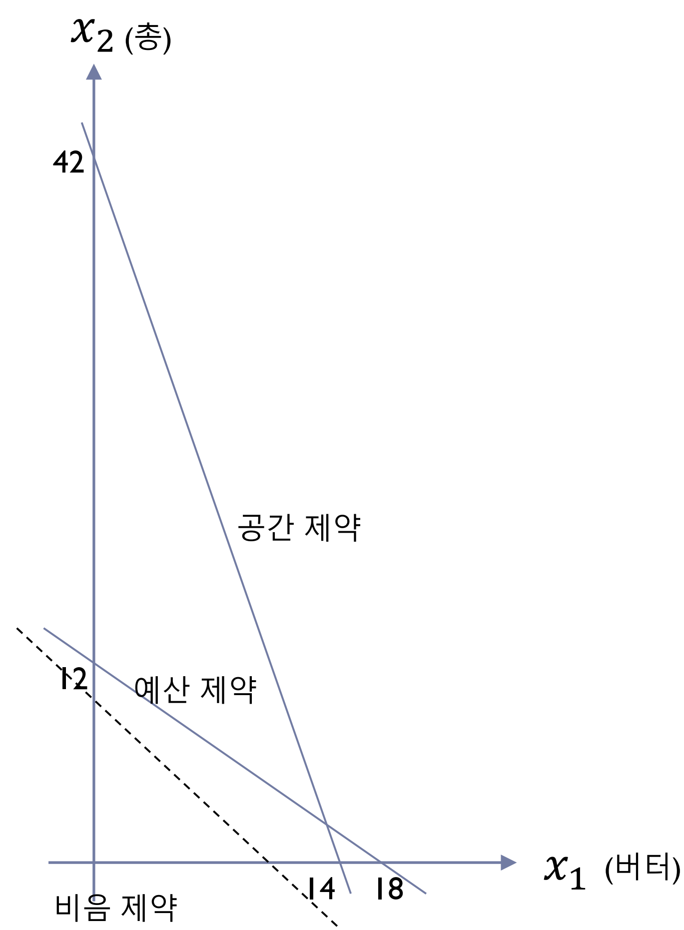
LP Example

- 총과 버터 만들기
  - 의사결정변수: 수익을 최대화할 수 있는 총과 버터의 양
    - x_1: 버터의 양
    - x_2: 총의 수
  - 목적: 가능한 많은 돈 벌기 – 버터 1톤 $150, 총 1개 $195
    - $\max 150x_1 + 195\,x_2$
  - 제약식: 예산 $1800/month, 1톤의 버터 생산에 $100, 1개의 총 생산에 $150
    공간 21cubic 미터, 버터 1.5 cubic, 총 0.5 cubic
    - 예산 제약: $100x_1 + 150x_2 \leq 1800$
    - 공간 제약: $1.5x_1 + 0.5x_2 \leq 21$
    - 비음 제약: $x_1, x_2 \geq 0$

LP Example

- LP 문제

$$\max 150x_1 + 195\,x_2$$
$$\text{s.t. } 100x_1 + 150x_2 \le 1800$$
$$1.5x_1 + 0.5x_2 \le 21$$
$$x_1, x_2 \ge 0$$



- Feasible region
- Level set

$$150x_1 + 195x_2 = z$$

  - (0,10), (13,0): z= 1950

- Simplex Method

LP 모델 구성하기

- 초기 데이터 설정
  - 사용된 공간: E7 =SUMPRODUCT(B4:C4,B7:C7)
  - 사용된 예산: E8 =SUMPRODUCT(B4:C4,B8:C8)
  - 총 매출: B1 =SUMPRODUCT(B4:C4,B9:C9)

  - Feasible Solution (1,1) 테스트

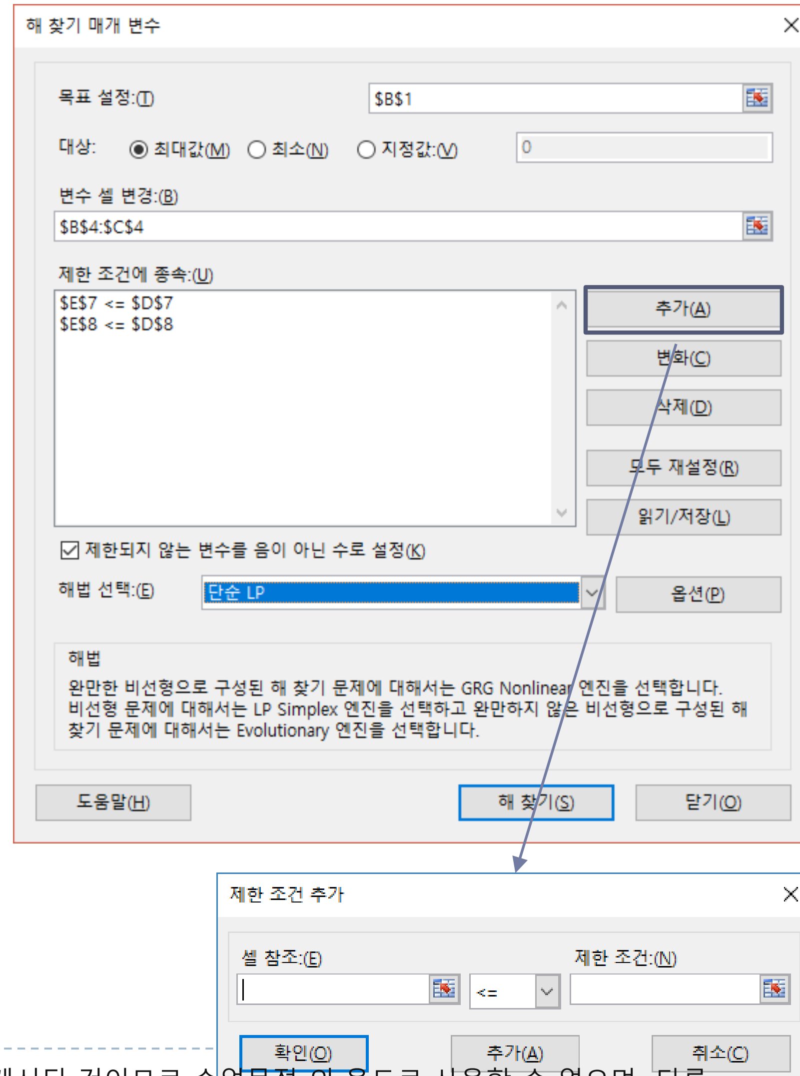
	A	B	C	D	E	F
1	Revenue	0				
2						
3		Guns	Butter			
4	Amount					
5						
6		Guns	Butter	Limit	Used	
7	Storage	0.5	1.5	21	0	
8	Price	150	100	1800	0	
9	Revenue	195	150			
10						

	A	B	C	D	E	F
1	Revenue	345				
2						
3		Guns	Butter			
4	Amount	1	1			
5						
6		Guns	Butter	Limit	Used	
7	Storage	0.5	1.5	21	2	
8	Price	150	100	1800	250	
9	Revenue	195	150			
10						

LP 모델 구성하기

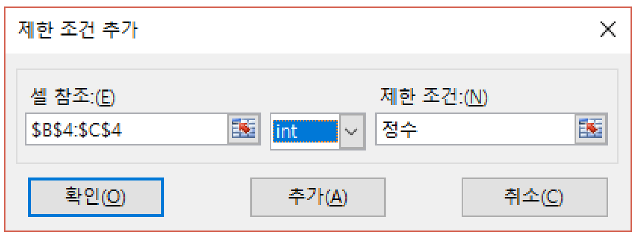
- Solver 설정
  - 목표설정: 목적함수 값
  - 대상: Min or Max
  - 변수셀: 의사결정 변수
  - 제약조건에 종속: 제약식
  - 제한되지 않는 변수를 음이 아닌 수로 설정: 비음조건
  - 해법선택: Simplex LP

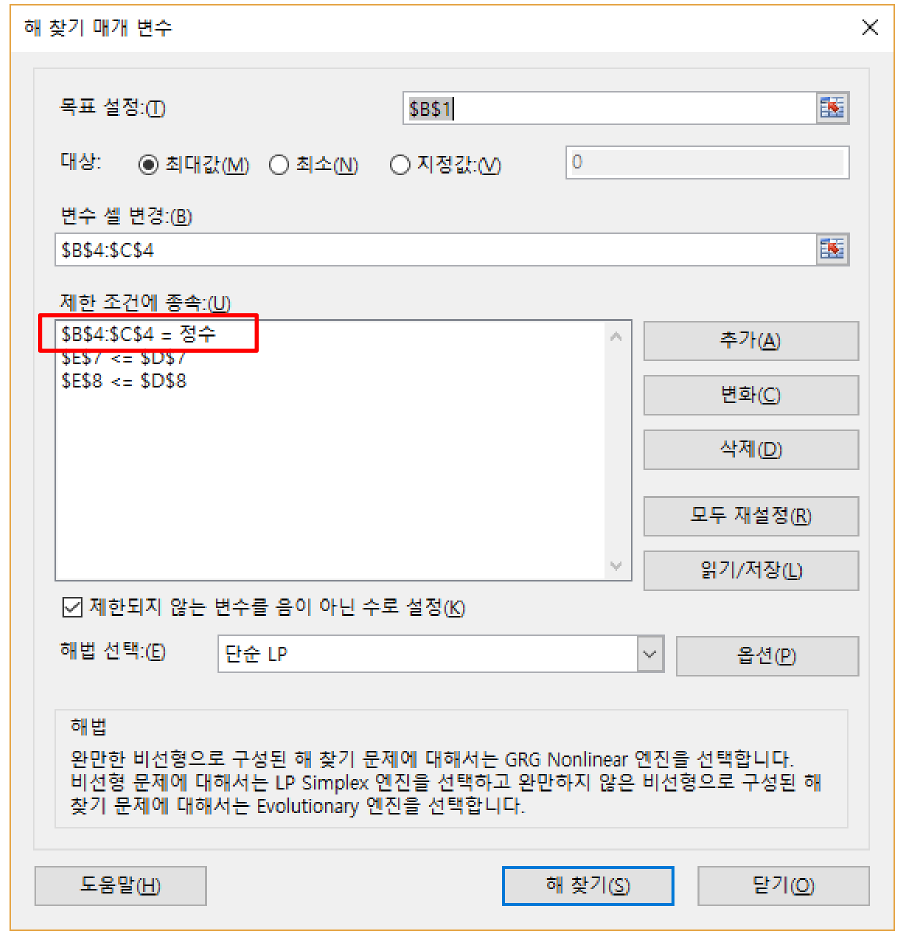
	A	B	C	D	E	F
1	Revenue	345				
2						
3		Guns	Butter			
4	Amount	1	1			
5						
6		Guns	Butter	Limit	Used	
7	Storage	0.5	1.5	21	2	
8	Price	150	100	1800	250	
9	Revenue	195	150			
10						



Integer 제약

- 정수해 찾기
  - LP는 실수값을 결정해줌
  - 3.43개의 종?
  - 정수해 찾기 제약 추가





(참고) 옵션에 '정수 제한 조건 무시'는 체크되지 않아야 함

Integer 제약

- 최적해 확인하기

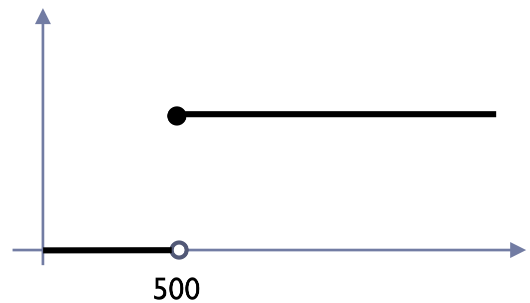
- Mixed Integer Linear Program (MILP)
  - Branch and Bound algorithm
  - 최적해가 아닐수도..

	A	B	C	D	E	F
1	Revenue	2597.143				
2						
3		Guns	Butter			
4	Amount	3.428571	12.85714			
5						
6		Guns	Butter	Limit	Used	
7	Storage	0.5	1.5	21	21	
8	Price	150	100	1800	1800	
9	Revenue	195	150			
10						
11						

	A	B	C	D	E	F
1	Revenue	2580				
2						
3		Guns	Butter			
4	Amount	4	12			
5						
6		Guns	Butter	Limit	Used	
7	Storage	0.5	1.5	21	20	
8	Price	150	100	1800	1800	
9	Revenue	195	150			
10						

Non-linear 함수

- 최적해 확인하기
  - 총 5개 이상이면 $500 추가 수입
  - B2 =SUMPRODUCT(B4:C4,B9:C9) +IF(B4>=5, 500,0)



  - GRG (Generalized reduced gradient) 비선형
  - Evolutionary : 의사결정변수의 bound의 지정 필요
    - $B$4:$C$4 <= 25

	A	B	C	D	E
1	Revenue	3020			
2					
3		Guns	Butter		
4	Amount	6	9		
5					
6		Guns	Butter	Limit	Used
7	Storage	0.5	1.5	21	16.5
8	Price	150	100	1800	1800
9	Revenue	195	150		
10					