Problem 1

Maximin Optimization Problems

$$f(x) := \min_{y \in \mathcal{Y}} g(x, y).$$

求函数 f(x) 的最大值点

个人理解：
x 理解为 solution（问题的解决方案）
y 理解为 scenario（问题的可能情况）

求在最坏的情况下表现最好的方案
因此，Minimax Optimization Problem 也叫做 robust design

进化算法中要对不同的解进行比较，都是按照以下思路进行
对于一个固定的 x，要找到使 g(x,y) 尽可能小的 y
对于一个固定的 y，要找到使 g(x,y) 尽可能大的 x

Problem 1

Maximin Optimization Problems

$$f(x) := \min_{y \in \mathcal{Y}} g(x, y).$$

求函数 f(x) 的最大值点

Black-Box Minimax Problem

也叫 Derivative-Free problem。不知道函数 g 的具体形式，只能询问有限次某个点的值，并且不能询问也不知道函数的斜率。

Problem 1

Maximin Optimization Problems

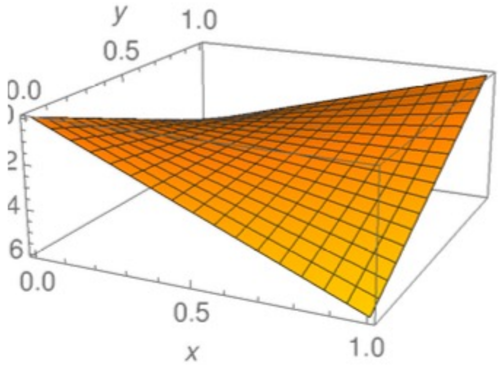
$$f(x) := \min_{y \in \mathcal{Y}} g(x, y).$$

求函数 f(x) 的最大值点

The Bilinear Problem

$$\textsc{Bilinear}(x, y) := |y|(|x| - \beta n) - \alpha n|x|,$$

对于特殊的 Bilinear 函数，求其最大值点



Bilinear 函数图像

Algorithm 1

对于函数 g(x,y) 定义支配关系 $\succeq_g$ ， 如果有 $(x_1, y_1) \succeq_g (x_2, y_2)$ 则满足：

$$g(x_1, y_2) \geq g(x_1, y_1) \geq g(x_2, y_1).$$

在该关系下最大的点就是 f(x) 的最大值点

Algorithm 1

Algorithm 2 Pairwise Dominance CoEA (PD-CoEA)

Require: Min-max-objective function $g : \{0,1\}^n \times \{0,1\}^n \to \mathbb{R}$.
Require: Population size $\lambda \in \mathbb{N}$ and mutation rate $\chi \in (0, n]$

1: **for** $i \in [\lambda]$ **do**
2:     Sample $P_0(i) \sim \mathrm{Unif}(\{0,1\}^n)$
3:     Sample $Q_0(i) \sim \mathrm{Unif}(\{0,1\}^n)$
4: **end for**
5: **for** $t \in \mathbb{N}$ until termination criterion met **do**
6:     **for** $i \in [\lambda]$ **do**
7:         Sample $(x_1, y_1) \sim \mathrm{Unif}(P_t \times Q_t)$
8:         Sample $(x_2, y_2) \sim \mathrm{Unif}(P_t \times Q_t)$
9:         **if** $(x_1, y_1) \succeq_g (x_2, y_2)$ **then**
10:             $(x, y) := (x_1, y_1)$
11:         **else**
12:             $(x, y) := (x_2, y_2)$
13:         **end if**
14:         Obtain x' by flipping each bit in x with prob. χ/n.
15:         Obtain y' by flipping each bit in y with prob. χ/n.
16:         Set $P_{t+1}(i) := x'$ and $Q_{t+1}(i) := y'$.
17:     **end for**
18: **end for**

1-4 随机生成初始种群

7-13 从种群中随机选出两个个体，选择在支配关系上更大的那个个体

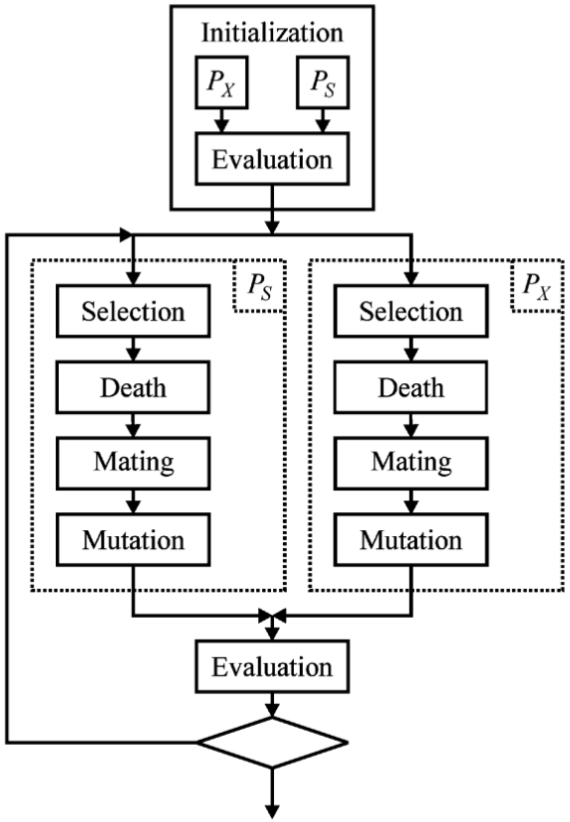
14-16 对该个体进行变异操作，加入下一代的种群中

Algorithm 2

J. W. Herrmann. A Genetic Algorithm for Minimax Optimization Problems. In Angeline et. al., editors, Proc. 1999 Congress on Evolutionary Computation, pages 1099-1103, 1997.

Parallel Coevolutionary Algorithm

1. 随机生成初始种群 P_1 P_2
2. 根据 P_2 种群对 P_1 中的个体进行估值
3. 依据 P_1 种群对 P_2 中的个体进行估值
4. 对 P_1 进行变异和交换操作，生成新的 P_1 种群
5. 对 P_2 进行变异和交换操作，生成新的 P_2 种群
6. 回到步骤 2



Algorithm 2

J. W. Herrmann. A Genetic Algorithm for Minimax Optimization Problems. In Angeline et. al., editors, Proc. 1999 Congress on Evolutionary Computation, pages 1099-1103, 1997.

Parallel Coevolutionary Algorithm

1. 随机生成初始种群 P_1 P_2
2. 根据 P_2 种群对 P_1 中的个体进行估值
3. 依据 P_1 种群对 P_2 中的个体进行估值
4. 对 P_1 进行变异和交换操作，生成新的 P_1 种群
5. 对 P_2 进行变异和交换操作，生成新的 P_2 种群
6. 回到步骤 2

P_1 P_2 ：代表 x, y 对应的种群

估值：对 P_1 中个体 x， 计算 g(x,y) 的最大值作为估计值，其 中 y 属于 P_2
对 P_2 中个体 y， 计算 g(x,y) 的最小值作为估计值，其中 x 属于 P_1

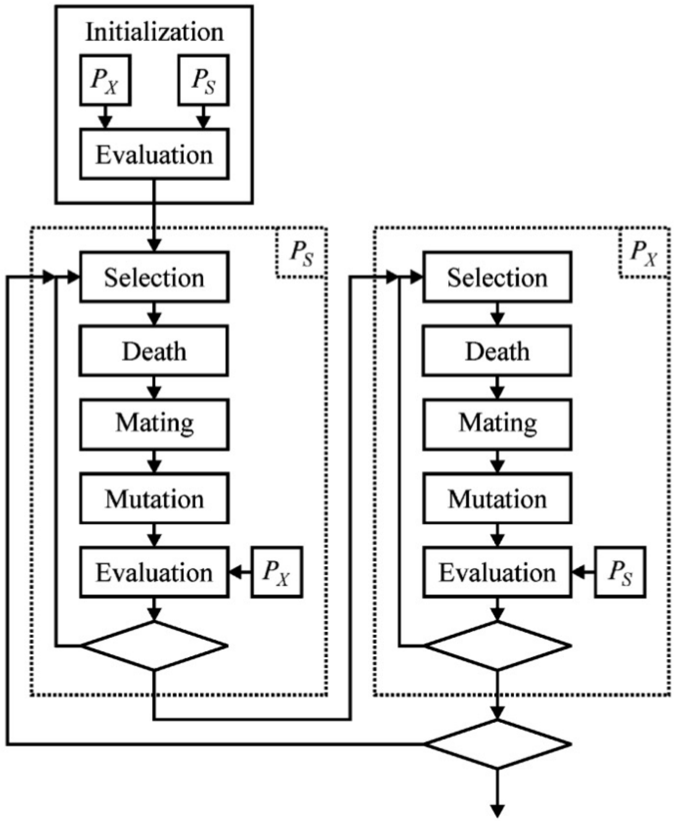
具体方案：一般采用 Tournament selection 选择出 k 个个体，然后这些个体每一位都有 p 的概率发生变异。在新个体选出最优的取代最劣的旧个体产生下一代的种群。

Algorithm 3

H.J.C.Barbosa, “A genetic algorithm for min-max problems,” inProc. 1st Int. Conf. Evol. Comput. and Applicat., 1996, pp. 99–109.

Alternating Coevolutionary Algorithm

1. 随机生成初始种群 P_1 P_2
2. 根据 P_2 种群对 P_1 中的个体进行估值
3. 对 P_1 进行变异和交换操作，生成新的 P_1 种群
4. 依据新的 P_1 种群对 P_2 中的个体进行估值
5. 对 P_2 进行变异和交换操作，生成新的 P_2 种群
6. 回到步骤 2



Algorithm 3

H.J.C.Barbosa, “A genetic algorithm for min-max problems,” inProc. 1st Int. Conf. Evol. Comput. and Applicat., 1996, pp. 99–109.

Alternating Coevolutionary Algorithm

1. 随机生成初始种群 P_1 P_2
2. 根据 P_2 种群对 P_1 中的个体进行估值
3. 对 P_1 进行变异和交换操作，生成新的 P_1 种群
4. 依据新的 P_1 种群对 P_2 中的个体进行估值
5. 对 P_2 进行变异和交换操作，生成新的 P_2 种群
6. 回到步骤 2

需要函数满足 symmetric 条件：

$$\min_{x\in X}\max_{s\in S} f(x,s) = \max_{s\in S}\min_{x\in X} f(x,s)$$

不满足时该算法结果会不收敛

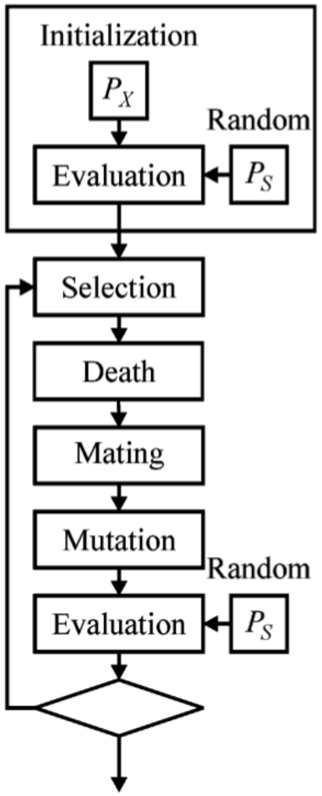
Bilinear 函数满足 symmetric 条件

Algorithm 4

A. M. Cramer, S. D. Sudhoff and E. L. Zivi, "Evolutionary Algorithms for Minimax Problems in Robust Design," in IEEE Transactions on Evolutionary Computation, vol. 13, no. 2, pp. 444-453, April 2009

Aging Sampled Genetic Algorithm

1. 初始化 x 对应种群 P_x
2. 随机一个 predator（y）加入 P_y，并计算 g(x,y) 作为适应度
3. 在 P_x 中随机选择个体，并进行交配和变异
4. 将新个体替换 P_x 中适应度低的个体，产生新种群 P_x
5. 随机一个 predator 加入 P_y，更新适应度。对于个体 x，P_y 中最小的 g(x,y) 作为适应度



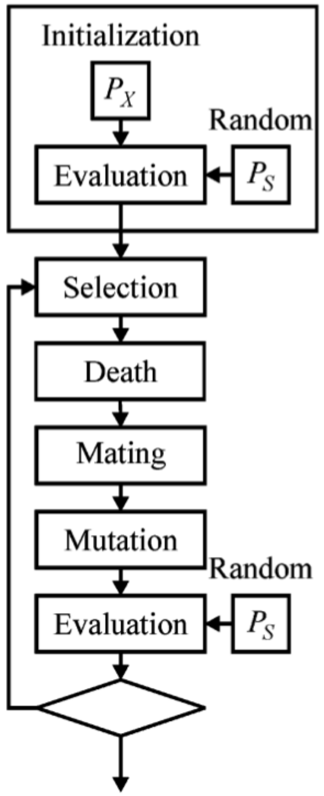
算法的表现在很多问题里比 CoEA 要好，但需要 g(x,y) 函数比较光滑，否则可能找不到最优解

Algorithm 4

A. M. Cramer, S. D. Sudhoff and E. L. Zivi, "Evolutionary Algorithms for Minimax Problems in Robust Design," in IEEE Transactions on Evolutionary Computation, vol. 13, no. 2, pp. 444-453, April 2009

Aging Sampled Genetic Algorithm

1. 初始化种群 P_x， 种群中个体是 x
2. 随机一个 y 加入 P_y，并计算 g(x,y) 作为适应度
3. 在 P_x 中随机选择个体 x，并进行交配和变异得到新个体 x^*
4. 将 x^* 替换 P_x 中适应度低的个体，产生新种群 P_x
5. 随机一个 y 加入 P_y，更新适应度。对于个体 x， P_y 中最小的 g(x,y) 作为适应度。
6. 重复步骤 2-6



同时论文也说明了目前很难知道在一个问题中，普通进化算法和合作进化算法的优劣

Algorithm 5

Xin Qiu et al. 2017. A New Differential Evolution Algorithm for Minimax Opti-mization in Robust Design. IEEE Transactions on Cybernetics (2017)

MINIMAX DIFFERENTIAL EVOLUTION ALGORTHM

1. 初始化种群 P_0，种群中的个体是二元组 (x, y)
2. 计算种群中个体的适应度 $\mathrm{g}(x, y)$
3. 采用 Bottom-Boosting Scheme 调整当前种群
4. 采用差分进化的策略产生新个体的 x^*，并随机生成 y^*
5. 将得到的新个体 (x^*, y^*) 取代种群 P 中适应度低的个体
6. 重复步骤 2-6

Differential Evolution Strategy

$\mathrm{X}_{\mathrm{i,j}}$ 表示当前种群的第 i 个体的第 j 维度，U 为生成的新个体，F 为缩放参数，C 为 crossover 概率参数

1. Mutation: $\mathrm{V} = \mathrm{X}_\mathrm{i} + \mathrm{F}(\mathrm{X}_j - \mathrm{X}_k)$ 其中 i, j, k 随机生成
2. Crossover: 如果 $\mathrm{rand}(0,1) < \mathrm{C}$， $\mathrm{U}_\mathrm{j} = \mathrm{V}_\mathrm{j}$ ；
   否则 $\mathrm{U}_\mathrm{j} = \mathrm{X}_{\mathrm{i,j}}$

Bottom-Boosting Scheme

1. 根据种群中个体的适应度 $\mathrm{g}(x, y)$ ，建立小根堆
2. 根据差分进化的策略，更新堆顶个体的 y，产生新的个体 (x, y^*)
3. 如果 $\mathrm{g}(x, y) < g(x, y^*)$（代表更坏情况），将 (x, y^*) 取代堆顶个体，同时调整堆的结构

Algorithm 6

Abdullah Al-Dujaili, Shashank Srikant, Erik Hemberg, and Una-May O' Reilly. 2019. On the application of Danskin' s theorem to derivative-free minimax problems. AIP Conference Proceedings 2070, 1 (Feb. 2019)

Algorithm 4 Reckless

Input:

T: number of iterations,

v: number of function evaluations (FEs) per iteration

$s \in (0, 0.5]$: budget allocation for descent direction

```
x_0 ~ U(X)
y_0 ~ U(Y)
x_* ← x_0
y_* ← y_0
for t = 1 to T do
    y_t ← arg max_{y∈Y} L(x_{t−1}, y) by ES with restarts and (1 − s)v FEs.
    if L(x_{t−1}, y_t) < L(x_*, y_*) then                ▷ best solution
        x_* ← x_{t−1}
        y_* ← y_t
    end if
    x_t ← arg min_{x∈X} L(x, y_t) by ES with sv FEs.
    if (x_t − x_{t−1})^T (x_{t−1} − x_{t−2}) ≤ 0 then    ▷ restart condition
        x_t ~ U(X)
        y_t ~ U(Y)
    end if
end for
return x_*, y_*
```

Algorithm 3 A Simplified Example of Evolution Strategy (ES)

Input:

η : learning rate

σ : perturbation standard deviation,

λ : number of perturbations (population size)

T : number of iterations (generations),

$f : \mathcal{X} \to \mathbb{R}$: fitness function

```
μ_0 ~ U(X)
for t = 0 to T do
    for i = 1 to λ do
        ε_i ~ N(0, I)
        f_i ← f(μ_t + σ ε_i)
    end for
    μ_{t+1} ← μ_t + η (1/(λσ)) Σ_{i=1}^{λ} f_i ε_i
end for
```

Algorithm 6

Abdullah Al-Dujaili, Shashank Srikant, Erik Hemberg, and Una-May O’ Reilly. 2019. On the application of Danskin’ s theorem to derivative-free minimax problems. AIP Conference Proceedings 2070, 1 (Feb. 2019)

Algorithm 3 A Simplified Example of Evolution Strategy (ES)

Input:

η : learning rate

σ : perturbation standard deviation,

λ : number of perturbations (population size)

T : number of iterations (generations),

$f : \mathcal{X} \to \mathbb{R}$: fitness function

$\boldsymbol{\mu}_0 \sim \mathcal{U}(\mathcal{X})$
for $t = 0$ **to** T **do**
  for $i = 1$ **to** λ **do**
    $\boldsymbol{\epsilon}_i \sim \mathcal{N}(0, I)$
    $f_i \leftarrow f(\boldsymbol{\mu}_t + \sigma \boldsymbol{\epsilon}_i)$
  end for
  $\boldsymbol{\mu}_{t+1} \leftarrow \boldsymbol{\mu}_t + \eta \frac{1}{\lambda\sigma} \sum_{i=1}^{\lambda} f_i \boldsymbol{\epsilon}_i$
end for

Danskin’ s theorem:

THEOREM 3.1 (MADRY ET AL. [23]). *Let* $\mathbf{y}^*$ *be such that* $\mathbf{y}^* \in \mathcal{Y}$ *and is a maximizer for* $\max_{\mathbf{y}} \mathcal{L}(\mathbf{x}, \mathbf{y})$. *Then, as long as it is nonzero,* $-\nabla_{\mathbf{x}} \mathcal{L}(\mathbf{x}, \mathbf{y}^*)$ *is a descent direction for* $\max_{\mathbf{y}} \mathcal{L}(\mathbf{x}, \mathbf{y})$.

使用蒙特卡洛方法，估计函数的梯度

$$-\nabla_{\mathbf{x}} \mathcal{L}(\mathbf{x}, \mathbf{y}^*) = -\frac{1}{\sigma\lambda} \sum_{i=1}^{\lambda} \mathcal{L}(\mathbf{x} + \sigma\epsilon_i, \mathbf{y}^*)\epsilon_i$$