ECE 580: Homework 5

Rahul Deshmukh

April 29, 2020

Exercise 1

In my Canonical GA, I am using the following parameter settings:

- Number of bits used to represent each variable: 10 (resolution=0.0098)

- Population size: 40

- Number of iterations: 30

- Probability for cross-over: 0.9

- Probability of Mutation: 0.01

- Selection Method: tournament selection method-2

After carrying out several trials, I obtain an optimal function value of $1.7928e - 05$ for the optimal solution as $x^* = \begin{bmatrix} -0.0049 & -0.0049 \end{bmatrix}^T$. The plot for best, average, and the worst objective function values in the population for every generation is at Figure 1.
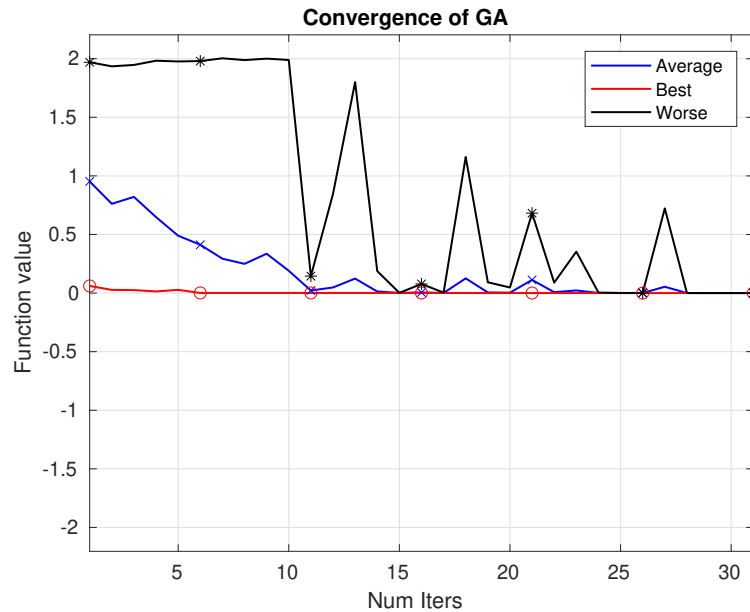

Figure 1: Plot of Average, Best and Worse function values for GA

For main file for GA refer to Listing 1 at page 19. The fitness function can be refered at Listing 2 at page 20 & Listing 3 at page 20. The encoding and decoding functions can be found at Listing 4 at page 21 & Listing 5 at page 21 respectively. The function for Tournament selection is at Listing 7 at page 22. The function for crossover, mutation and elitism can be refered at Listing 8 at page 22, Listing 9 at page 23 & Listing 10 at page 23 respectively.

Exercise 2

In my Real-Number GA, I am using the following parameter settings:

- Population size: 40

- Number of iterations: 30

- Probability for cross-over: 0.9

- Crossover-Method: Convex Combination

- Probability of Mutation: 0.01

- Selection Method: tournament selection method-2

After carrying out several trials, I obtain an optimal function value of $1.7333e - 11$ for the optimal solution as $x^* = 1.0e - 05 * \begin{bmatrix} 0.0195 & 0.8318 \end{bmatrix}^T$. The plot for best, average, and the worst objective function values in the population for every generation is at Figure 2.
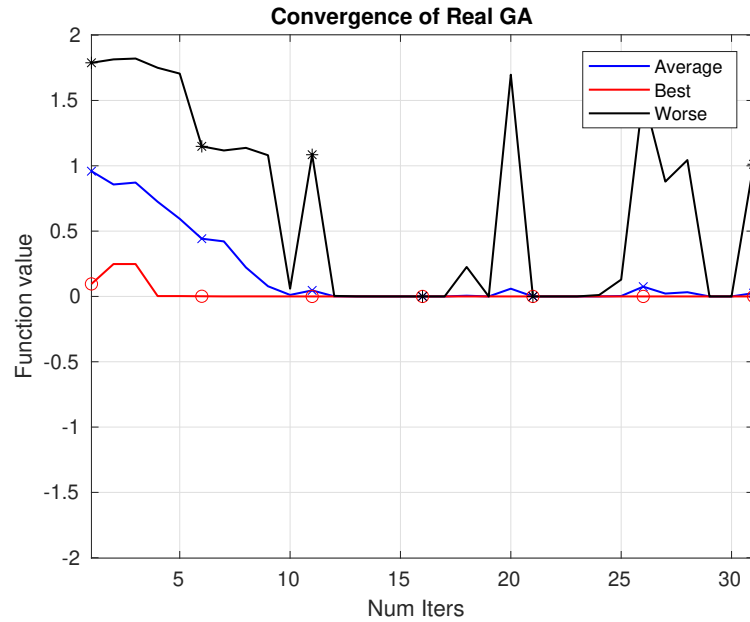

Figure 2: Plot of Average, Best and Worse function values for GA

For main file for GA refer to Listing 12 at page 24. The fitness function can be refered at Listing 2 at page 20 & Listing 3 at page 20. The function for Tournament selection is at Listing 7 at page 22. The function for crossover, mutation and elitism can be refered at Listing 13 at page 25, Listing 14 at page 26 & Listing 10 at page 23 respectively.

Exercise 3

The given LP problem is:

$$\begin{aligned}
\text{maximize} \quad & -4x_1 - 3x_2 \\
\text{subject to} \quad & 5x_1 + x_2 \geq 11 \\
& 2x_1 + x_2 \geq 8 \\
& x_1 + 2x_2 \geq 7 \\
& x_1, x_2 \geq 0
\end{aligned}$$

We first convert the above problem to standard form:

$$\begin{aligned}
\text{minimize} \quad & 4x_1 + 3x_2 \\
\text{subject to} \quad & 5x_1 + x_2 - x_3 = 11 \\
& 2x_1 + x_2 - x_4 = 8 \\
& x_1 + 2x_2 - x_5 = 7 \\
& x_1, x_2, x_3, x_4, x_5 \geq 0
\end{aligned}$$

We then solve the above problem using Two-phase simplex method. The computations (with pivot elements in boxes) are as follows:

Phase 1:

$$\begin{aligned}
\begin{bmatrix} \boldsymbol{A} & \boldsymbol{I} & \boldsymbol{b} \\ \boldsymbol{0}^T & \boldsymbol{1}^T & 0 \end{bmatrix} &=
\begin{bmatrix}
5 & 1 & -1 & 0 & 0 & 1 & 0 & 0 & 11 \\
2 & 1 & 0 & -1 & 0 & 0 & 1 & 0 & 8 \\
1 & 2 & 0 & 0 & -1 & 0 & 0 & 1 & 7 \\
0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0
\end{bmatrix} \\
&=
\begin{bmatrix}
\boxed{5} & 1 & -1 & 0 & 0 & 1 & 0 & 0 & 11 \\
2 & 1 & 0 & -1 & 0 & 0 & 1 & 0 & 8 \\
1 & 2 & 0 & 0 & -1 & 0 & 0 & 1 & 7 \\
-8 & -4 & 1 & 1 & 1 & 0 & 0 & 0 & -26
\end{bmatrix} \\
&=
\begin{bmatrix}
1 & 1/5 & -1/5 & 0 & 0 & 1/5 & 0 & 0 & 11/5 \\
0 & 3/5 & 2/5 & -1 & 0 & -2/5 & 1 & 0 & 18/5 \\
0 & \boxed{9/5} & 1/5 & 0 & -1 & -1/5 & 0 & 1 & 24/5 \\
0 & -12/5 & -3/5 & 1 & 1 & 8/5 & 0 & 0 & -42/5
\end{bmatrix} \\
&=
\begin{bmatrix}
1 & 0 & -2/9 & 0 & 1/9 & 2/9 & 0 & -1/9 & 5/3 \\
0 & 0 & \boxed{1/3} & -1 & 1/3 & -1/3 & 1 & -1/3 & 2 \\
0 & 1 & 1/9 & 0 & -5/9 & -1/9 & 0 & 5/9 & 8/3 \\
0 & 0 & -1/3 & 1 & -1/3 & 4/3 & 0 & 4/3 & -2
\end{bmatrix} \\
&=
\begin{bmatrix}
1 & 0 & 0 & -2/3 & 1/3 & 0 & 2/3 & -1/3 & 3 \\
0 & 0 & 1 & -3 & 1 & -1 & 3 & -1 & 6 \\
0 & 1 & 0 & 1/3 & -2/3 & 0 & -1/3 & 2/3 & 2 \\
0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0
\end{bmatrix}
\end{aligned}$$

3

Phase 2:

$$\begin{bmatrix} \boldsymbol{A} & \boldsymbol{b} \\ \boldsymbol{c}^T & 0 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & -2/3 & 1/3 & 3 \\ 0 & 0 & 1 & -3 & 1 & 6 \\ 0 & 1 & 0 & 1/3 & -2/3 & 2 \\ 4 & 3 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$= \begin{bmatrix} 1 & 0 & 0 & -2/3 & 1/3 & 3 \\ 0 & 0 & 1 & -3 & 1 & 6 \\ 0 & 1 & 0 & 1/3 & -2/3 & 2 \\ 0 & 0 & 0 & 5/3 & 2/3 & -18 \end{bmatrix}$$

The optimal solution is given by $x_1^* = 3, x_2^* = 2$ with maximum function value as $-4x_1^* - 3x_2^* = -18$

For MATLAB function for this problem refer to Listing 16 at page 26 & Listing 17 at page 28 and the call to the function can be referred at Listing 15 at page 26 with corresponding output at Listing 18 at page 29.

Exercise 4

The dual problem is given by:

$$\text{maximize} \quad \boldsymbol{\lambda}^T \boldsymbol{b}$$
$$\text{subject to} \quad \boldsymbol{\lambda}^T \boldsymbol{A} \leq \boldsymbol{c}^T$$

that is:

$$\text{maximize} \quad 11\lambda_1 + 8\lambda_2 + 7\lambda_3$$
$$\text{subject to} \quad \begin{bmatrix} \lambda_1 & \lambda_2 & \lambda_3 \end{bmatrix}^T \begin{bmatrix} 5 & 1 \\ 2 & 1 \\ 1 & 2 \end{bmatrix} \leq \begin{bmatrix} 4 & 3 \end{bmatrix}$$
$$\lambda_1, \lambda_2, \lambda_3 \geq 0$$

Since we have already obtained the optimal BFS $\boldsymbol{x}^* = \begin{bmatrix} 3 & 2 & 0 & 0 & 0 \end{bmatrix}^T$ corresponding to the optimal basis $\boldsymbol{B} = \begin{bmatrix} 5 & 1 & -1 \\ 2 & 1 & 0 \\ 1 & 2 & 0 \end{bmatrix}$ and cost coefficients $\boldsymbol{c_B} = \begin{bmatrix} 4 & 3 & 0 \end{bmatrix}^T$. From theorem of duality, we have

$$\boldsymbol{\lambda}^T \boldsymbol{b} = \boldsymbol{c}^T \boldsymbol{x} = \boldsymbol{c_B}^T \boldsymbol{B}^{-1} \boldsymbol{b}$$
$$\Rightarrow \boldsymbol{\lambda}^T = \boldsymbol{c_B}^T \boldsymbol{B}^{-1}$$
$$= \begin{bmatrix} 4 & 3 & 0 \end{bmatrix}^T \begin{bmatrix} 5 & 1 & -1 \\ 2 & 1 & 0 \\ 1 & 2 & 0 \end{bmatrix}^{-1}$$
$$= \frac{1}{-3} \begin{bmatrix} 4 & 3 & 0 \end{bmatrix}^T \begin{bmatrix} 0 & -2 & 1 \\ 0 & 1 & -2 \\ 3 & -9 & 3 \end{bmatrix}^T$$
$$\boldsymbol{\lambda}^{*T} = \begin{bmatrix} 0 & 5/3 & 2/3 \end{bmatrix}$$
$$\boldsymbol{\lambda}^{*T} \boldsymbol{b} = 18$$

Exercise 5

We have to solve the following constrained optimization problem:

$$\text{maximize} \quad f(x) = 4x_1 + x_2^2$$
$$\text{eqv minimize} \quad -f(x) = -4x_1 - x_2^2$$
$$\text{subject to} \quad h_1(x) = x_1^2 + x_2^2 - 9 = 0$$

$$\text{lagrangian} \quad l(\boldsymbol{x}, \lambda_1) = -4x_1 - x_2^2 + \lambda_1(x_1^2 + x_2^2 - 9)$$

Using FONC:

$$\nabla_{\boldsymbol{x}} l(\boldsymbol{x}, \lambda_1) = \begin{bmatrix} -4 & -2x_2 \end{bmatrix} + \lambda_1 \begin{bmatrix} 2x_1 & 2x_2 \end{bmatrix} = \boldsymbol{0}^T$$
$$\Rightarrow \begin{bmatrix} -4 + 2\lambda_1 x_1 \\ 2x_2(\lambda_1 - 1) \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$
$$\nabla_{\lambda_1} l(\boldsymbol{x}, \lambda_1) = x_1^2 + x_2^2 - 9 = 0$$

The solution to the above set of equations is:

$$\boldsymbol{x}_a^* = \begin{bmatrix} x_1 \\ x_2 \\ \lambda_1 \end{bmatrix} = \begin{bmatrix} \pm 3 \\ 0 \\ \pm\frac{2}{3} \end{bmatrix}, \quad \boldsymbol{x}_b^* = \begin{bmatrix} x_1 \\ x_2 \\ \lambda_1 \end{bmatrix} = \begin{bmatrix} 2 \\ \pm\sqrt{5} \\ 1 \end{bmatrix}$$

Computing the Hessian of lagrangian:

$$L(\boldsymbol{x}, \lambda_1) = 2 \begin{bmatrix} \lambda_1 & 0 \\ 0 & (\lambda_1 - 1) \end{bmatrix}$$

$$\Rightarrow L(\boldsymbol{x}_b^*, \lambda_1^* = 1) = 2 \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} \succcurlyeq 0 \Rightarrow \boldsymbol{x}_b^* \text{ is a possible minimizer of } -f(\boldsymbol{x})$$

$$L(\boldsymbol{x}_{a2}^*, \lambda_1^* = -\frac{2}{3}) = 2 \begin{bmatrix} -\frac{2}{3} & 0 \\ 0 & -\frac{5}{3} \end{bmatrix} \preccurlyeq 0 \Rightarrow \boldsymbol{x}_b^* \text{ is a possible maximizer of } -f(\boldsymbol{x})$$

For SONC we need to find the Tangent Space $T(\boldsymbol{x}^*)$:

$$T(\boldsymbol{x}_b^*) = \{y : \nabla_{\boldsymbol{x}} h_1^T(x)y = 0\}$$
$$= \{y : \begin{bmatrix} 2x_1^* & 2x_2^* \end{bmatrix} y = 0\}$$
$$= \{y : \begin{bmatrix} 2 & \pm\sqrt{5} \end{bmatrix} y = 0\}$$
$$= \{y : y = t \begin{bmatrix} 1 \\ \mp\frac{\sqrt{5}}{2} \end{bmatrix}, \quad t \in \mathbb{R} - \{0\}\}$$

$$y^T L(\boldsymbol{x}_b^*) y = t^2 \begin{bmatrix} 1 & \mp\frac{\sqrt{5}}{2} \end{bmatrix} 2 \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} 1 \\ \mp\frac{\sqrt{5}}{2} \end{bmatrix}$$
$$= 2t^2 > 0 \quad \forall \quad t \in \mathbb{R} - \{0\}$$

Therefore $\boldsymbol{x}_b^* = \begin{bmatrix} 2 \\ \pm\sqrt{5} \\ 1 \end{bmatrix}$ is a strict minimizer of $-f(x)$ and thus is a maximizer of $f(x)$

Exercise 6

We have to solve the following optimization problem:

$$\text{maximize} \quad r(x) = \frac{18x_1^2 - 8x_1x_2 + 12x_2^2}{2x_1^2 + 2x_2^2}$$

Suppose the above problem has a solution x^* which maximizes the function. Then $x = tx^*, t \in \mathbb{R} - \{0\}$ can be another possible solution. Therefore we convert the above problem to a constrained maximization problem with unique solution by enforcing the denominator to be 1. The converted problem is given by :

$$\text{maximize} \quad f(x) = 18x_1^2 - 8x_1x_2 + 12x_2^2$$
$$\text{subject to} \quad h(x) = 1 - (2x_1^2 + 2x_2^2) = 0$$

Converting to minimization problem:

$$\text{minimize} \quad -f(x) = -\frac{1}{2}\boldsymbol{x}^T \begin{bmatrix} 36 & -8 \\ -8 & 24 \end{bmatrix} \boldsymbol{x}$$

$$\text{subject to} \quad h(x) = 1 - \frac{1}{2}\boldsymbol{x}^T \begin{bmatrix} 4 & 0 \\ 0 & 4 \end{bmatrix} \boldsymbol{x}$$

$$\text{lagrangian} \quad l(x, \lambda) = -\frac{1}{2}\boldsymbol{x}^T \begin{bmatrix} 36 & -8 \\ -8 & 24 \end{bmatrix} \boldsymbol{x} + \lambda(1 - \frac{1}{2}\boldsymbol{x}^T \begin{bmatrix} 4 & 0 \\ 0 & 4 \end{bmatrix} \boldsymbol{x})$$

FONC:

$$\nabla_\lambda l(\boldsymbol{x}, \lambda) = 2x_1^2 + 2x_2^2 - 1 = 0$$

$$\nabla_{\boldsymbol{x}} l(\boldsymbol{x}, \lambda) = \begin{bmatrix} -36 & 8 \\ 8 & -24 \end{bmatrix} \boldsymbol{x} - \lambda(\begin{bmatrix} 4 & 0 \\ 0 & 4 \end{bmatrix} \boldsymbol{x}) = \boldsymbol{0}$$

$$\Rightarrow \begin{bmatrix} 4 & 0 \\ 0 & 4 \end{bmatrix}^{-1} \begin{bmatrix} -36 & +8 \\ +8 & -24 \end{bmatrix} \boldsymbol{x} = \lambda\boldsymbol{x}$$

$$\Rightarrow \begin{bmatrix} -9 & +2 \\ +2 & -6 \end{bmatrix} \boldsymbol{x} = \lambda\boldsymbol{x}$$

$$\Rightarrow det(\begin{bmatrix} -9 & +2 \\ +2 & -6 \end{bmatrix} - \lambda\boldsymbol{I}_2) = 0 \Rightarrow \lambda = -10, -5$$

$$\boldsymbol{x} = \alpha \begin{bmatrix} -2 \\ 1 \end{bmatrix}, \beta \begin{bmatrix} 1 \\ 2 \end{bmatrix}$$

The solution to the above equations is:

$$\boldsymbol{x}_a^* = \begin{bmatrix} x_1 \\ x_2 \\ \lambda \end{bmatrix} = \begin{bmatrix} \mp\frac{2}{\sqrt{10}} \\ \pm\frac{1}{\sqrt{10}} \\ -10 \end{bmatrix}, \quad \boldsymbol{x}_b^* = \begin{bmatrix} x_1 \\ x_2 \\ \lambda \end{bmatrix} = \begin{bmatrix} \pm\frac{1}{\sqrt{10}} \\ \pm\frac{2}{\sqrt{10}} \\ -5 \end{bmatrix}$$

SONC:

$$L(\boldsymbol{x}, \lambda) = \begin{bmatrix} -36 & 8 \\ 8 & -24 \end{bmatrix} - \lambda \begin{bmatrix} 4 & 0 \\ 0 & 4 \end{bmatrix}$$

$$= \left(\begin{bmatrix} -9 - \lambda & +2 \\ +2 & -6 - \lambda \end{bmatrix} \right) \begin{bmatrix} 4 & 0 \\ 0 & 4 \end{bmatrix}$$

$$\Rightarrow L(\boldsymbol{x}, \lambda = -10) = \begin{bmatrix} 4 & 8 \\ 8 & 16 \end{bmatrix} \succcurlyeq 0 \text{ is a possible minimizer of } - f(x)$$

$$\Rightarrow L(\boldsymbol{x}, \lambda = -5) = \begin{bmatrix} -16 & +8 \\ +8 & -4 \end{bmatrix} \preccurlyeq 0 \text{ is a possible maximizer of } - f(x)$$

SOSC: We need to find the Tangent Space $T(x^*)$

$$T(\boldsymbol{x}_a^*) = \{y : \nabla_{\boldsymbol{x}} h_1^T(x) y = 0\}$$

$$= \{y : \begin{bmatrix} 4 & 0 \\ 0 & 4 \end{bmatrix} \begin{bmatrix} \mp \frac{2}{\sqrt{10}} \\ \pm \frac{1}{\sqrt{10}} \end{bmatrix} y = 0\}$$

$$= \{y : y = t \begin{bmatrix} 1 \\ 2 \end{bmatrix}, \quad t \in \mathbb{R} - \{0\}\}$$

$$\Rightarrow y^T L(\boldsymbol{x}_a^*) y = 100t^2 > 0 \quad \forall \quad t \in \mathbb{R} - \{0\}$$

Therefore the solution corresponding to $\lambda = -10$, $\boldsymbol{x}_a^* = \begin{bmatrix} \mp \frac{2}{\sqrt{10}} \\ \pm \frac{1}{\sqrt{10}} \end{bmatrix}$ is a strict minimizer of $-f(x)$ and thus is a maximizer of $f(x)$.

Therefore all possible maximizers of

$$r(x) = \frac{18x_1^2 - 8x_1 x_2 + 12x_2^2}{2x_1^2 + 2x_2^2}$$

are of the form $\boldsymbol{x}^* = t \begin{bmatrix} \mp \frac{2}{\sqrt{10}} \\ \pm \frac{1}{\sqrt{10}} \end{bmatrix} \quad \forall \quad t \in \mathbb{R} - \{0\}$

Exercise 7

We need to find the extremizers for problems (a)-(c)

Problem (a)

The given problem is:

$$f(\boldsymbol{x}) = x_1^2 + x_2^2 - 2x_1 - 10x_2 + 26$$

$$\text{subject to} \quad g(\boldsymbol{x}) = \begin{bmatrix} \frac{x_2}{5} - x_1^2 \\ 5x_1 + \frac{x_2}{2} - 5 \end{bmatrix} \le \boldsymbol{0}$$

The gradients are given by:

$$\nabla f(x) = \begin{bmatrix} 2x_1 - 2 \\ 2x_2 - 10 \end{bmatrix}$$

$$\nabla g(x) = \begin{bmatrix} -2x_1 & 1/5 \\ 5 & 1/2 \end{bmatrix}$$

$$F(x) = \begin{bmatrix} 2 & 0 \\ 0 & 0 \end{bmatrix}$$

$$G_1(x) = \begin{bmatrix} -2 & 0 \\ 0 & 0 \end{bmatrix}$$

$$G_2(x) = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$$

The lagrangian is given by:

$$l(x, \lambda, \mu) = f(x) + \lambda^T h(x) + \mu^T g(x)$$

$$\nabla_x l(x, \lambda, \mu) = \begin{bmatrix} 2x_1 - 2 \\ 2x_2 - 10 \end{bmatrix} + \begin{bmatrix} -2x_1 & 1/5 \\ 5 & 1/2 \end{bmatrix}^T \begin{bmatrix} \mu_1 \\ \mu_2 \end{bmatrix}$$

$$L(x, \lambda, \mu) = F(x) + \sum_{i=1}^{m} \lambda_i H_i(x) + \sum_{i=1}^{p} \mu_i G_i(x)$$

$$= \begin{bmatrix} 2 & 0 \\ 0 & 0 \end{bmatrix} + \mu_1 \begin{bmatrix} -2 & 0 \\ 0 & 0 \end{bmatrix} + \mu_2 \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$$

$$= \begin{bmatrix} 2 & 0 \\ 0 & 0 \end{bmatrix} + \mu_1 \begin{bmatrix} -2 & 0 \\ 0 & 0 \end{bmatrix}$$

FONC: Using KKT:-

Case 1: $\mu_1 = 0, \mu_2 = 0$

$$\nabla_x l(x, \lambda, \mu) = \begin{bmatrix} 2x_1 - 2 \\ 2x_2 - 10 \end{bmatrix} = 0$$

$$\text{subject to} \quad g_1(x) = \frac{x_2}{5} - x_1^2 \le 0$$

$$g_2(x) = 5x_1 + \frac{x_2}{2} - 5 \le 0$$

8

The solution to the above system of equations is:

$$\boldsymbol{x}^* = \begin{bmatrix} 1 \\ 5 \end{bmatrix}$$

$$g_1(x) = 0 \leq 0 \quad \checkmark$$

$$g_2(x) = 5 + 5/2 \leq 0 \quad \textcolor{red}{✗}$$

Therefore the above solution is infeasible

Case 2: $\mu_1 = 0, \mu_2 > 0$

$$\nabla_x l(x, \lambda, \mu) = \begin{bmatrix} 2x_1 - 2 \\ 2x_2 - 10 \end{bmatrix} + \begin{bmatrix} -2x_1 & 1/5 \\ 5 & 1/2 \end{bmatrix}^T \begin{bmatrix} 0 \\ \mu_2 \end{bmatrix} = 0$$

$$\text{subject to} \quad g_1(x) = \frac{x_2}{5} - x_1^2 \leq 0$$

$$g_2(x) = 5x_1 + \frac{x_2}{2} - 5 = 0$$

The solution to the above system of equations is:

$$\boldsymbol{x}^* = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 51/101 \\ 500/101 \end{bmatrix}$$

$$\mu_2^* = 20/101$$

$$g_1(x) = \frac{x_2}{5} - x_1^2 = (100 * 101 - 51^2)/101^2 \leq 0 \quad \textcolor{red}{✗}$$

There is no feasible solution.

Case 3: $\mu_1 > 0, \mu_2 > 0$

$$\nabla_x l(x, \lambda, \mu) = \begin{bmatrix} 2x_1 - 2 \\ 2x_2 - 10 \end{bmatrix} + \begin{bmatrix} -2x_1 & 1/5 \\ 5 & 1/2 \end{bmatrix}^T \begin{bmatrix} \mu_1 \\ \mu_2 \end{bmatrix} = 0$$

$$\text{subject to} \quad g_1(x) = \frac{x_2}{5} - x_1^2 = 0$$

$$g_2(x) = 5x_1 + \frac{x_2}{2} - 5 = 0$$

The solution to the above system of equations is:

$$\boldsymbol{x}^* = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} -1 \pm \sqrt{3} \\ 20 \mp \sqrt{3} \end{bmatrix}$$

$$\mu_1^* = \frac{304 - 202\sqrt{3}}{-2\sqrt{3}} \quad \checkmark$$

$$\mu_2^* = (-300 + 200\sqrt{3} - 2\mu_1)/5 > 0 \quad \checkmark$$

SONC:

$$L(\boldsymbol{x}^*, \mu^*) = \begin{bmatrix} 2 & 0 \\ 0 & 0 \end{bmatrix} + \mu_1 \begin{bmatrix} -2 & 0 \\ 0 & 0 \end{bmatrix}$$

$$T(\boldsymbol{x}^*) = \{y : \begin{bmatrix} -2x_1 & 1/5 \\ 5 & 1/2 \end{bmatrix} y = 0\}$$

$$= \{y : \begin{bmatrix} -2(-1 \pm \sqrt{3}) & 1/5 \\ 5 & 1/2 \end{bmatrix} y = 0\}$$

$$= \{y : y = \boldsymbol{0}\}$$

$$\Rightarrow y^T L(\boldsymbol{x}^*, \mu^*) y = 0 \geq 0$$

Thereofore $\boldsymbol{x}^* = \begin{bmatrix} -1 \pm \sqrt{3} \\ 20 \mp \sqrt{3} \end{bmatrix}$ is an extremizer.

SOSC:

$$\tilde{T}(\boldsymbol{x}^*) = \{y : \begin{bmatrix} -2x_1 & 1/5 \\ 5 & 1/2 \end{bmatrix} y = 0\}$$

$$= \{y : \begin{bmatrix} -2(-1 \pm \sqrt{3}) & 1/5 \\ 5 & 1/2 \end{bmatrix} y = 0\}$$

$$= \{y : y = \boldsymbol{0}\}$$

$$\Rightarrow y^T L(\boldsymbol{x}^*, \mu^*) y = 0 > 0 \quad \textcolor{red}{\boldsymbol{X}}$$

As $\boldsymbol{x}^*$ does not satisfy SONC, it is not a minimizer. It is a saddle point.

Case 4: $\mu_1 > 0, \mu_2 = 0$

$$\nabla_x l(x, \lambda, \mu) = \begin{bmatrix} 2x_1 - 2 \\ 2x_2 - 10 \end{bmatrix} + \begin{bmatrix} -2x_1 & 1/5 \\ 5 & 1/2 \end{bmatrix}^T \begin{bmatrix} \mu_1 \\ 0 \end{bmatrix} = 0$$

$$\text{subject to} \quad g_1(x) = \frac{x_2}{5} - x_1^2 = 0$$

$$g_2(x) = 5x_1 + \frac{x_2}{2} - 5 \leq 0$$

The solution to the above system of equations is:

$$\boldsymbol{x}^* = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} -48 + 10\sqrt{23} \\ 23020 - 4800\sqrt{23} \end{bmatrix}$$

$$\mu_1^* = \frac{52 + 10\sqrt{23}}{4} > 0 \quad \textcolor{blue}{\checkmark}$$

$$g_2(x) = 5x_1 + \frac{x_2}{2} - 5 = 11265 - 2350\sqrt{23} \leq 0 \quad \textcolor{blue}{\checkmark}$$

Therefore $\boldsymbol{x}^* = \begin{bmatrix} -48 + 10\sqrt{23} \\ 23020 - 4800\sqrt{23} \end{bmatrix}$ is a possible extremizer as it satisfies the KKT conditions.

SONC:

$$L(\boldsymbol{x}^*, \mu^*) = \begin{bmatrix} 2 & 0 \\ 0 & 0 \end{bmatrix} + \mu_1 \begin{bmatrix} -2 & 0 \\ 0 & 0 \end{bmatrix}$$

$$T(\boldsymbol{x}^*) = \{y : \begin{bmatrix} -2x_1 & 1/5 \\ 5 & 1/2 \end{bmatrix} y = 0\}$$

$$= \{y : \begin{bmatrix} -2(-48 + 10\sqrt{23}) & 1/5 \\ 5 & 1/2 \end{bmatrix} y = 0\}$$

$$= \{y : y = \boldsymbol{0}\}$$

$$\Rightarrow y^T L(\boldsymbol{x}^*, \mu^*) y = 0 \geq 0$$

Therefore $\boldsymbol{x}^*$ is an extremizer as it satisfies SONC.

SOSC:

$$\tilde{T}(\boldsymbol{x}^*) = \{y : \begin{bmatrix} -2x_1 & 1/5 \end{bmatrix} y = 0\}$$

$$= \{y : y = t \begin{bmatrix} 1 \\ 10x_1 \end{bmatrix} \quad \forall \quad t \in \mathbb{R} - \{0\}\}$$

$$\Rightarrow y^T L(\boldsymbol{x}^*, \mu^*) y = 50x_1(1 + x_1)t^2$$

$$= 50 * (-48 + 10\sqrt{23})(1 - 48 + 10\sqrt{23})t^2 < 0 \quad \forall \quad t \in \mathbb{R} - \{0\}\}$$

Therefore $\boldsymbol{x}^* = \begin{bmatrix} -48 + 10\sqrt{23} \\ 23020 - 4800\sqrt{23} \end{bmatrix}$ is a strict maximizer.

Problem (b)

$$f(\boldsymbol{x}) = x_1^2 + x_2^2$$

$$\text{subject to} \quad g(\boldsymbol{x}) = \begin{bmatrix} -x_1 \\ -x_2 \\ -x_1 - x_2 + 5 \end{bmatrix} \leq \mathbf{0}$$

The gradients are given by:

$$\nabla f(x) = \begin{bmatrix} 2x_1 \\ 2x_2 \end{bmatrix}$$

$$\nabla g(x) = \begin{bmatrix} -1 & 0 \\ 0 & -1 \\ -1 & -1 \end{bmatrix}$$

$$F(x) = \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix}$$

$$G_1(x) = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$$

$$G_2(x) = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$$

$$G_3(x) = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$$

The lagrangian is given by:

$$l(x, \lambda, \mu) = f(x) + \lambda^T h(x) + \mu^T g(x)$$

$$\nabla_x l(x, \lambda, \mu) = \begin{bmatrix} 2x_1 \\ 2x_2 \end{bmatrix} + \begin{bmatrix} -1 & 0 \\ 0 & -1 \\ -1 & -1 \end{bmatrix}^T \begin{bmatrix} \mu_1 \\ \mu_2 \\ \mu_3 \end{bmatrix}$$

$$L(x, \lambda, \mu) = F(x) + \sum_{i=1}^{m} \lambda_i H_i(x) + \sum_{i=1}^{p} \mu_i G_i(x)$$

$$= \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix} + \mu_1 \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} + \mu_2 \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} + \mu_3 \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$$

$$= \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix} \succ 0 \quad \text{in all of } \mathbb{R}^2$$

Thus as $L(x, \lambda, \mu)$ is PD in all of $\mathbb{R}^2$, the SONC is always satisfied for any solution satisfying KKT.

FONC: Using KKT:-

<u>Case 1:</u> $\mu_1 > 0, \mu_2 > 0, \mu_3 > 0$

$$\nabla_x l(x, \lambda, \mu) = \begin{bmatrix} 2x_1 \\ 2x_2 \end{bmatrix} + \begin{bmatrix} -1 & 0 \\ 0 & -1 \\ -1 & -1 \end{bmatrix}^T \begin{bmatrix} \mu_1 \\ \mu_2 \\ \mu_3 \end{bmatrix}$$

$$\text{subject to} \quad g_1(x) = -x_1 = 0$$
$$g_2(x) = -x_2 = 0$$
$$g_3(x) = -x_1 - x_2 + 5 = 0$$

There is no feasible solution to the above system of equations.

<u>Case 2:</u> $\mu_1 > 0, \mu_2 > 0, \mu_3 = 0$

$$\nabla_x l(x, \lambda, \mu) = \begin{bmatrix} 2x_1 \\ 2x_2 \end{bmatrix} + \begin{bmatrix} -1 & 0 \\ 0 & -1 \\ -1 & -1 \end{bmatrix}^T \begin{bmatrix} \mu_1 \\ \mu_2 \\ 0 \end{bmatrix}$$

$$\text{subject to} \quad g_1(x) = -x_1 = 0$$
$$g_2(x) = -x_2 = 0$$
$$g_3(x) = -x_1 - x_2 + 5 = 5 \le 0 \quad \textcolor{red}{✗}$$

There is no feasible solution to the above system of equations.

<u>Case 3:</u> $\mu_1 > 0, \mu_2 = 0, \mu_3 > 0$

$$\nabla_x l(x, \lambda, \mu) = \begin{bmatrix} 2x_1 \\ 2x_2 \end{bmatrix} + \begin{bmatrix} -1 & 0 \\ 0 & -1 \\ -1 & -1 \end{bmatrix}^T \begin{bmatrix} \mu_1 \\ 0 \\ \mu_3 \end{bmatrix}$$

$$\text{subject to} \quad g_1(x) = -x_1 = 0$$
$$g_2(x) = -x_2 \le 0$$
$$g_3(x) = -x_1 - x_2 + 5 = 0$$

The solution to the above set of equations is :

$$x^* = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 5 \end{bmatrix}$$
$$\mu_3^* = 10$$
$$\mu_1^* = -10 > 0 \quad \textcolor{red}{✗}$$

There is no feasible solution.

<u>Case 4:</u> $\mu_1 = 0, \mu_2 > 0, \mu_3 > 0$

$$\nabla_x l(x, \lambda, \mu) = \begin{bmatrix} 2x_1 \\ 2x_2 \end{bmatrix} + \begin{bmatrix} -1 & 0 \\ 0 & -1 \\ -1 & -1 \end{bmatrix}^T \begin{bmatrix} 0 \\ \mu_2 \\ \mu_3 \end{bmatrix}$$

$$\text{subject to} \quad g_1(x) = -x_1 \le 0$$
$$g_2(x) = -x_2 = 0$$
$$g_3(x) = -x_1 - x_2 + 5 = 0$$

The solution to the above set of equations is :

$$\boldsymbol{x}^* = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 5 \\ 0 \end{bmatrix}$$

$$\mu_3^* = 10$$

$$\mu_2^* = -10 > 0 \quad \textcolor{red}{\boldsymbol{X}}$$

There is no feasible solution.

Case 5: $\mu_1 > 0, \mu_2 = 0, \mu_3 = 0$

$$\nabla_x l(x, \lambda, \mu) = \begin{bmatrix} 2x_1 \\ 2x_2 \end{bmatrix} + \begin{bmatrix} -1 & 0 \\ 0 & -1 \\ -1 & -1 \end{bmatrix}^T \begin{bmatrix} \mu_1 \\ 0 \\ 0 \end{bmatrix}$$

$$\text{subject to} \quad g_1(x) = -x_1 = 0$$

$$g_2(x) = -x_2 \le 0$$

$$g_3(x) = -x_1 - x_2 + 5 \le 0$$

The solution to the above set of equations is :

$$\boldsymbol{x}^* = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

$$g_2(x) = -x_2 = 0 \le 0 \quad \textcolor{blue}{\checkmark}$$

$$g_3(x) = -x_1 - x_2 + 5 = 5 \le 0 \quad \textcolor{red}{\boldsymbol{X}}$$

$$\mu_1^* = 0 > 0 \quad \textcolor{red}{\boldsymbol{X}}$$

There is no feasible solution.

Case 6: $\mu_1 = 0, \mu_2 > 0, \mu_3 = 0$

$$\nabla_x l(x, \lambda, \mu) = \begin{bmatrix} 2x_1 \\ 2x_2 \end{bmatrix} + \begin{bmatrix} -1 & 0 \\ 0 & -1 \\ -1 & -1 \end{bmatrix}^T \begin{bmatrix} 0 \\ \mu_2 \\ 0 \end{bmatrix}$$

$$\text{subject to} \quad g_1(x) = -x_1 \le 0$$

$$g_2(x) = -x_2 = 0$$

$$g_3(x) = -x_1 - x_2 + 5 \le 0$$

The solution to the above set of equations is :

$$\boldsymbol{x}^* = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

$$g_1(x) = -x_1 = 0 \le 0 \quad \textcolor{blue}{\checkmark}$$

$$g_3(x) = -x_1 - x_2 + 5 = 5 \le 0 \quad \textcolor{red}{\boldsymbol{X}}$$

$$\mu_2^* = 0 > 0 \quad \textcolor{red}{\boldsymbol{X}}$$

There is no feasible solution.

<u>Case 7:</u> $\mu_1 = 0, \mu_2 = 0, \mu_3 = 0$

$$\nabla_x l(x, \lambda, \mu) = \begin{bmatrix} 2x_1 \\ 2x_2 \end{bmatrix} + \begin{bmatrix} -1 & 0 \\ 0 & -1 \\ -1 & -1 \end{bmatrix}^T \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

$$\text{subject to} \quad g_1(x) = -x_1 \leq 0$$
$$g_2(x) = -x_2 \leq 0$$
$$g_3(x) = -x_1 - x_2 + 5 \leq 0$$

The solution to the above set of equations is :

$$x^* = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$
$$g_1(x) = -x_1 = 0 \leq 0 \quad \checkmark$$
$$g_2(x) = -x_2 = 0 \leq 0 \quad \checkmark$$
$$g_3(x) = -x_1 - x_2 + 5 = 5 \leq 0 \quad \textcolor{red}{\times}$$

There is no feasible solution.

<u>Case 8:</u> $\mu_1 = 0, \mu_2 = 0, \mu_3 > 0$

$$\nabla_x l(x, \lambda, \mu) = \begin{bmatrix} 2x_1 \\ 2x_2 \end{bmatrix} + \begin{bmatrix} -1 & 0 \\ 0 & -1 \\ -1 & -1 \end{bmatrix}^T \begin{bmatrix} 0 \\ 0 \\ \mu_3 \end{bmatrix}$$

$$\text{subject to} \quad g_1(x) = -x_1 \leq 0$$
$$g_2(x) = -x_2 \leq 0$$
$$g_3(x) = -x_1 - x_2 + 5 = 0$$

The solution to the above set of equations is :

$$x^* = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 5/2 \\ 5/2 \end{bmatrix}, \mu_3^* = 5$$
$$g_1(x) = -x_1 = -5/2 \leq 0 \quad \checkmark$$
$$g_2(x) = -x_2 = -5/2 \leq 0 \quad \checkmark$$

Therefore $x^* = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 5/2 \\ 5/2 \end{bmatrix}$ is a possible extremizer. Also since $L(x, \lambda, \mu) \succ 0$ for all of $\mathbb{R}^2$, thus x^* a strict minimizer.

Problem (c)

$$f(\boldsymbol{x}) = x_1^2 + 6x_1x_2 - 4x_1 - 2x_2$$

$$\text{subject to} \quad g(\boldsymbol{x}) = \begin{bmatrix} x_1^2 + 2x_2 - 1 \\ 2x_1 - 2x_2 - 1 \end{bmatrix} \leq \mathbf{0}$$

The gradients are given by:

$$\nabla f(x) = \begin{bmatrix} 2x_1 + 6x_2 - 4 \\ 6x_1 - 2 \end{bmatrix}$$

$$\nabla g(x) = \begin{bmatrix} 2x_1 & 2 \\ 2 & -2 \end{bmatrix}$$

$$F(x) = \begin{bmatrix} 2 & 6 \\ 6 & 0 \end{bmatrix}$$

$$G_1(x) = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}$$

$$G_2(x) = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$$

The lagrangian is given by:

$$l(x, \lambda, \mu) = f(x) + \lambda^T h(x) + \mu^T g(x)$$

$$\nabla_x l(x, \lambda, \mu) = \begin{bmatrix} 2x_1 + 6x_2 - 4 \\ 6x_1 - 2 \end{bmatrix} + \begin{bmatrix} 2x_1 & 2 \\ 2 & -2 \end{bmatrix}^T \begin{bmatrix} \mu_1 \\ \mu_2 \end{bmatrix}$$

$$L(x, \lambda, \mu) = F(x) + \sum_{i=1}^{m} \lambda_i H_i(x) + \sum_{i=1}^{p} \mu_i G_i(x)$$

$$= \begin{bmatrix} 2 & 6 \\ 6 & 0 \end{bmatrix} + \mu_1 \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} + \mu_2 \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$$

$$= \begin{bmatrix} 2 & 6 \\ 6 & 0 \end{bmatrix} + \mu_1 \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}$$

FONC: Using KKT:-

Case 1: $\mu_1 = 0, \mu_2 = 0$

$$\nabla_x l(x, \lambda, \mu) = \begin{bmatrix} 2x_1 + 6x_2 - 4 \\ 6x_1 - 2 \end{bmatrix} + \begin{bmatrix} 2x_1 & 2 \\ 2 & -2 \end{bmatrix}^T \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

$$\text{subject to} \quad g_1(x) = x_1^2 + 2x_2 - 1 \leq 0$$

$$g_2(x) = 2x_1 - 2x_2 - 1 \leq 0$$

The solution to the above set of equations is :

$$\boldsymbol{x}^* = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 1/3 \\ 5/9 \end{bmatrix}$$

$$g_1(x) = x_1^2 + 2x_2 - 1 = 2/9 \leq 0 \quad \textcolor{red}{✗}$$

$$g_2(x) = 2x_1 - 2x_2 - 1 = -13/9 \leq 0 \quad \textcolor{blue}{✓}$$

There is no feasible solution.

Case 2: $\mu_1 > 0, \mu_2 = 0$

$$\nabla_x l(x, \lambda, \mu) = \begin{bmatrix} 2x_1 + 6x_2 - 4 \\ 6x_1 - 2 \end{bmatrix} + \begin{bmatrix} 2x_1 & 2 \\ 2 & -2 \end{bmatrix}^T \begin{bmatrix} \mu_1 \\ 0 \end{bmatrix}$$

$$\text{subject to} \quad g_1(x) = x_1^2 + 2x_2 - 1 = 0$$

$$g_2(x) = 2x_1 - 2x_2 - 1 \leq 0$$

The solution to the above set of equations gives:

$$x_1 = \frac{1 - \mu_1}{3}$$

$$x_2 = \frac{2}{3} - \frac{1}{9}(1 - \mu_1^2)$$

$$g_1(x) = x_1^2 + 2x_2 - 1 = 0$$

$$\Rightarrow 3\mu_1^2 - 2\mu_1 + 2 = 0$$

$$\Rightarrow \mu_1^* = \frac{2 \pm \sqrt{4 - 24}}{2} \notin \mathbb{R}$$

There is no feasible solution.

Case 3: $\mu_1 > 0, \mu_2 > 0$

$$\nabla_x l(x, \lambda, \mu) = \begin{bmatrix} 2x_1 + 6x_2 - 4 \\ 6x_1 - 2 \end{bmatrix} + \begin{bmatrix} 2x_1 & 2 \\ 2 & -2 \end{bmatrix}^T \begin{bmatrix} \mu_1 \\ \mu_2 \end{bmatrix}$$

$$\text{subject to} \quad g_1(x) = x_1^2 + 2x_2 - 1 = 0$$

$$g_2(x) = 2x_1 - 2x_2 - 1 = 0$$

The solution to the above set of equations gives :

$$x^* = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} -1 \pm \sqrt{3} \\ -3/2 \pm \sqrt{3} \end{bmatrix}$$

$$\mu_1^* = \pm \frac{1}{\sqrt{3}}(23 \mp 14\sqrt{3}) > 0 \quad \textcolor{red}{\times}$$

$$\mu_2^* = \mu_1 + (-4 \pm 3\sqrt{3})$$

There is no feasible solution.

Case 4: $\mu_1 = 0, \mu_2 > 0$

$$\nabla_x l(x, \lambda, \mu) = \begin{bmatrix} 2x_1 + 6x_2 - 4 \\ 6x_1 - 2 \end{bmatrix} + \begin{bmatrix} 2x_1 & 2 \\ 2 & -2 \end{bmatrix}^T \begin{bmatrix} 0 \\ \mu_2 \end{bmatrix}$$

$$\text{subject to} \quad g_1(x) = x_1^2 + 2x_2 - 1 \leq 0$$

$$g_2(x) = 2x_1 - 2x_2 - 1 = 0$$

The solution to the above set of equations is :

$$x^* = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 9/14 \\ 1/7 \end{bmatrix}$$

$$\mu_2^* = 13/14 > 0 \quad \textcolor{blue}{\checkmark}$$

$$g_1(x) = x_1^2 + 2x_2 - 1 = -59/196 \leq 0 \quad \textcolor{blue}{\checkmark}$$

Therefore $\boldsymbol{x}^* = \begin{bmatrix} 9/14 \\ 1/7 \end{bmatrix}$ is a possible extremizer.

SONC:

$$L(\boldsymbol{x}^*, \mu^*) = \begin{bmatrix} 2 & 6 \\ 6 & 0 \end{bmatrix}$$

$$T(\boldsymbol{x}^*) = \{y : \begin{bmatrix} 2x_1 & 2 \\ 2 & -2 \end{bmatrix} y = \boldsymbol{0}\}$$

$$= \{y : \begin{bmatrix} 9/7 & 2 \\ 2 & -2 \end{bmatrix} y = \boldsymbol{0}\}$$

$$= \{y : y = \boldsymbol{0}\}$$

$$\Rightarrow y^T L(\boldsymbol{x}^*, \mu^*) y = 0 \geq 0$$

Thus $\boldsymbol{x}^*$ satisfies SONC and therefore is an extremizer.

SOSC:

$$\tilde{T}(\boldsymbol{x}^*) = \{y : \begin{bmatrix} 2 & -2 \end{bmatrix} y = 0\}$$

$$= \{y : y = t \begin{bmatrix} 1 \\ 1 \end{bmatrix} \quad \forall \quad t \in \mathbb{R} - \{0\}\}$$

$$\Rightarrow y^T L(\boldsymbol{x}^*, \mu^*) y = 14t^2 > 0 \quad \forall \quad t \in \mathbb{R} - \{0\}$$

Therefore $\boldsymbol{x}^* = \begin{bmatrix} 9/14 \\ 1/7 \end{bmatrix}$ is a strict minimizer.

MATLAB Code

Canonical GA Code

Listing 1: Canonical GA Main Code

```matlab
1  % ECE 580 HW5: Problem 1
2  % Rahul Deshmukh
3  % deshmuk5@purdue.edu
4  clc; clear all; close all;
5  format short;
6  %% Include paths
7  addpath('../OptimModule/optimizers/global/GA/');
8  save_dir = './pix/';
9  %% set seed
10 rng(7); % tried 0:10, 7 gave the smallest result, can jump between global and ...
       local min
11 %% Canonical GA problem setup
12 lb = -5*ones(1,2);
13 ub = 5*ones(1,2);
14 Num_vars = length(lb);
15 bits = 10;
16 coded_lens = bits*ones(1,Num_vars);
17 resolution = (ub-lb)./(2.^coded_lens-1)
18
19 %% GA: solver params
20 N_pop = 40; % cant be odd integer
21 p_xover = 0.9;
22 p_mut = 0.01;
23 Niters = 30;
24 selection_method = 'tournament_method2';
25
26 %% GA starts
27
28 % intialize collectors
29 best_f = [];
30 av_f = [];
31 worse_f = [];
32
33 % choose type of selector
34 if strcmp(selection_method, 'roulette')
35     selection = @(x,f) roulette(x,f);
36 elseif strcmp(selection_method, 'tournament_method1')
37     selection = @(x,f) tournament_selection(x,f,1);
38 elseif strcmp(selection_method, 'tournament_method2')
39     selection = @(x,f) tournament_selection(x,f,2);
40 end
41
42 % draw initial population
43 X = rand(N_pop, Num_vars);
44 % scale to domain
45 X = (X.*(ub-lb) + lb);
46 % discretize to resolution
47 X = floor((X - lb)./resolution).*resolution + lb;
48
49 %encode X
50 parents = encode(X, lb, ub, coded_lens, resolution);
```

```matlab
51  % evaluate fitness of parents
52  f_parent = -1*fitness_griewank(parents, lb, coded_lens, resolution);
53  [best_f, av_f, worse_f] = log_f(f_parent, best_f, av_f, worse_f);
54  for i=1:Niters
55      % generate mating pool using selection
56      mating_pool = selection(parents, f_parent);
57      %perform crossover
58      parents = two_point_crossover(mating_pool, p_xover);
59      %perform mutation
60      parents = mutation(parents, p_mut);
61      %perform elitism
62      parents = elitism(parents, f_parent);
63      %evaluate fitness of offspring
64      f_parent = -1*fitness_griewank(parents, lb, coded_lens, resolution);
65      [best_f, av_f, worse_f] = log_f(f_parent, best_f, av_f, worse_f);
66  end
67  % find the best offspring
68  [f_star, k_star] = max(f_parent);
69  fprintf(strcat('best fval: \t',num2str(-1*f_star)))
70  x_star_coded = parents(k_star,:);
71  x_star = decode(x_star_coded, lb, coded_lens, resolution)
72
73  %% Convergence Plotting
74  fig1 = figure(1);
75  hold on; grid on;
76  x = 1:Niters+1;
77  h1 =plot(x,-1*av_f,'-b','LineWidth',1);
78  h2 = plot(x,-1*best_f,'-r','LineWidth',1);
79  h3 = plot(x,-1*worse_f,'-k', 'LineWidth',1);
80  v = 1:5:Niters+1;
81  plot(x(v),-1*av_f(v),'bx');
82  plot(x(v),-1*best_f(v),'ro');
83  plot(x(v),-1*worse_f(v),'k*');
84  legend([h1,h2,h3],{'Average','Best','Worse'},'Location','northeast');
85  hold off;
86  box('on');
87  xlabel('Num Iters'); ylabel('Function value');
88  xlim([1, Niters+1])
89  ylim(max(abs(worse_f))*(1.1)*[-1,1]);
90  title('Convergence of GA');
91  saveas(fig1,strcat(save_dir,'ga_canon_conv'),'epsc');
```

Listing 2: Fitness function

```matlab
1  function f = fitness_griewank(X_coded, lb, code_lens, resolution)
2  X = decode(X_coded, lb, code_lens, resolution);
3  f = griewank_fun(X');
4  end
```

Listing 3: Griewank function

```matlab
1  function y = griewank_fun(X_swarm,min_bool)
2  %      d dimensionalfriewank function
3  switch nargin
4      case 1
5          min_bool=1;
```

```matlab
 6  end
 7
 8  [x_dim ,Nswarm] = size(X_swarm);
 9  y = zeros(Nswarm,1);
10  for k=1:Nswarm
11      sum = 0;
12      prod = 1;
13      x = X_swarm(:,k);
14      for i=1:x_dim
15          x_i = x(i);
16          sum = sum + x_i^2/4000;
17          prod = prod * cos(x_i/sqrt(i));
18      end
19      y(k) = sum - prod +1;
20  end
21  if ¬min_bool
22      y = -1*y;
23  end
24  end
```

Listing 4: Encoding function

```matlab
 1  function X_coded = encode(X, lb, ub, code_lens, resolution)
 2  [N_pop,¬] = size(X);
 3  L = sum(code_lens);
 4  cumsum_code_lens = [0, cumsum(code_lens)];
 5  X_coded = zeros(N_pop,L);
 6  Num_var = length(lb);
 7  % convert discretized X to integers for encoding
 8  X = round((X - lb)./resolution);
 9  for i = 1:N_pop
10      x = X(i,:);
11      x_coded = zeros(1,L);
12      for j = 1:Num_var
13          xj = x(j);
14          x_coded(cumsum_code_lens(j) + 1 : cumsum_code_lens(j+1)) = de2bi(xj ...
                 ,code_lens(j));
15      end
16      X_coded(i,:) = x_coded;
17  end
```

Listing 5: Decoding function

```matlab
 1  function X = decode(X_coded, lb, code_lens, resolution)
 2  [N_pop,¬] = size(X_coded);
 3  L = sum(code_lens);
 4  cumsum_code_lens = [0, cumsum(code_lens)];
 5  Num_var = length(lb);
 6  X = zeros(N_pop,Num_var);
 7  for i=1:N_pop
 8      x_coded = X_coded(i,:);
 9      x = zeros(1,Num_var);
10      for j=1:Num_var
11          xj_coded = x_coded(cumsum_code_lens(j) + 1 : cumsum_code_lens(j+1));
12          x(j) = resolution(j)*bi2de(xj_coded);
13      end
```

21

```
14     X(i,:) = x + lb;
15  end
16  end
```

Listing 6: Roulette-wheel selection function

```
1  function mating_pool = roulette(parent, f_parent)
2  [N_pop,¬] = size(parent);
3  f_min = min(f_parent);
4  f = f_parent – f_min;
5  F = sum(f);
6  p = f/F;
7  q = cumsum(p);
8  rand_nums = rand(N_pop,1);
9  mating_idx = zeros(N_pop,1);
10 temp = q' –rand_nums;
11 for k=1:N_pop
12    mating_idx(k) = find(temp(k,:) > 0, 1);
13 end
14 mating_pool = parent(mating_idx, :);
15 end
```

Listing 7: Tournament selection function

```
1  function mating_pool = tournament_selection(parent, f_parent, method)
2  [N_pop,¬] = size(parent);
3  mating_idx = zeros(N_pop,1);
4  if method == 1
5     a = randi([1, N_pop], 1, N_pop) ;
6     b = randi([1, N_pop], 1, N_pop) ;
7     fa = f_parent(a);
8     fb = f_parent(b);
9     for k=1:N_pop
10        if fa(k)>fb(k)
11           mating_idx(k) = a(k);
12        else
13           mating_idx(k) = b(k);
14        end
15     end
16 elseif method == 2
17     a = randi([1, N_pop], 1, N_pop);
18     fa = f_parent(a);
19     for k=1:N_pop
20        if fa(k)>f_parent(k)
21           mating_idx(k) = a(k);
22        else
23           mating_idx(k) = k;
24        end
25     end
26 end
27 mating_pool = parent(mating_idx, :);
28 end
```

Listing 8: Cross-over function

```matlab
1  function offspring = two_point_crossover(mating_pool, p_xover)
2
3  [N_pop,L] = size(mating_pool);
4  % shuffle parents
5  mating_pool = mating_pool(randperm(N_pop),:);
6  % generate rand nums for deciding if do crossover?
7  rand_nums = rand(1,round(N_pop/2));
8  do_xover = rand_nums > (1-p_xover);
9  offspring = zeros(N_pop, L);
10
11 for k = 1:round(N_pop/2)
12     parents = mating_pool([2*k-1 ,2*k],:);
13     if do_xover(k)
14         % find crossover point
15         xover_pt = randi(L,1);
16         % switch genes
17         offspring(2*k-1,:)= [parents(1,1:xover_pt), parents(2,xover_pt+1:end)];
18         offspring(2*k,:) = [parents(2,1:xover_pt), parents(1,xover_pt+1:end)];
19     else
20         offspring([2*k-1, 2*k],:) = parents;
21     end
22 end
23
24 end
```

Listing 9: Mutation function

```matlab
1  function mutated = mutation(parents, p_mut)
2  [N_pop,¬] = size(parents);
3  rand_nums = rand(N_pop,1);
4  do_mut_idx = find(rand_nums < p_mut);
5  mutated = parents;
6  % complement each bit in parent
7  mutated(do_mut_idx,:) = 1-parents(do_mut_idx,:);
8  end
```

Listing 10: Elitism function

```matlab
1  function new_pop = elitism(pop, fitness)
2  new_pop = pop;
3  temp_fit = fitness;
4  [¬, max_fit_idx] = max(temp_fit);
5  temp_fit(max_fit_idx) = min(temp_fit);
6  [¬, other_max_fit_idx] = max(temp_fit);
7  new_pop([1,2],:) = pop([max_fit_idx, other_max_fit_idx],:);
8  end
```

Listing 11: Logging function

```matlab
1  function [best_f, av_f, worse_f] = log_f(f_parent, best_f, av_f, worse_f)
2  best_f = [best_f, max(f_parent)];
3  av_f = [av_f, mean(f_parent)];
4  worse_f = [worse_f, min(f_parent)];
5  end
```

Real GA Code

```matlab
1  % ECE 580 HW5: Problem 2
2  % Rahul Deshmukh
3  % deshmuk5@purdue.edu
4  clc; clear all; close all;
5  format short;
6  %% Include paths
7  addpath('../OptimModule/optimizers/global/GA/');
8  addpath('../OptimModule/optimizers/global/GA/Real_Num_GA/');
9  save_dir = './pix/';
10 %% set seed
11 rng(6); % tried 0:10 ,6 was the best and always converges to global min
12 %% Real GA problem setup
13 lb = -5*ones(1,2);
14 ub = 5*ones(1,2);
15 Num_vars = length(lb);
16
17 %% GA: solver params
18 N_pop = 40; % cant be odd integer
19 p_xover = 0.9;
20 p_mut = 0.01;
21 Niters = 30;
22 selection_method = 'tournament_method2';
23 xover_method = 'conv_combo';
24
25 %% GA starts
26
27 % intialize collectors
28 best_f = [];
29 av_f = [];
30 worse_f = [];
31
32 % choose type of selector
33 if strcmp(selection_method, 'roulette')
34     selection = @(x,f) roulette(x,f);
35 elseif strcmp(selection_method, 'tournament_method1')
36     selection = @(x,f) tournament_selection(x,f,1);
37 elseif strcmp(selection_method, 'tournament_method2')
38     selection = @(x,f) tournament_selection(x,f,2);
39 end
40
41 % draw initial population
42 X = rand(N_pop, Num_vars);
43 % scale to domain
44 X = (X.*(ub-lb) + lb);
45
46 parents = X;
47 % evaluate fitness of parents
48 f_parent = -1*griewank_fun(parents');
49 [best_f, av_f, worse_f] = log_f(f_parent, best_f, av_f, worse_f);
50 for i=1:Niters
51     % generate mating pool using selection
52     mating_pool = selection(parents, f_parent);
53     %perform crossover
54     parents = crossover(mating_pool, p_xover, xover_method);
```

```matlab
55      %perform mutation
56      parents = mutation(parents, p_mut, lb, ub);
57      %perform elitism
58      parents = elitism(parents, f_parent);
59      %evaluate fitness of offspring
60      f_parent = -1*griewank_fun(parents');
61      [best_f, av_f, worse_f] = log_f(f_parent, best_f, av_f, worse_f);
62  end
63  % find the best offspring
64  [f_star, k_star] = max(f_parent);
65  fprintf(strcat('best fval: \t',num2str(-1*f_star)))
66  x_star = parents(k_star,:)
67
68  %% Convergence Plotting
69  fig1 = figure(1);
70  hold on; grid on;
71  x = 1:Niters+1;
72  h1 =plot(x,-1*av_f,'-b','LineWidth',1);
73  h2 = plot(x,-1*best_f,'-r','LineWidth',1);
74  h3 = plot(x,-1*worse_f,'-k', 'LineWidth',1);
75  v = 1:5:Niters+1;
76  plot(x(v),-1*av_f(v),'bx');
77  plot(x(v),-1*best_f(v),'ro');
78  plot(x(v),-1*worse_f(v),'k*');
79  legend([h1,h2,h3],{'Average','Best','Worse'},'Location','northeast');
80  hold off;
81  box('on');
82  xlabel('Num Iters'); ylabel('Function value');
83  xlim([1, Niters+1])
84  ylim(max(abs(worse_f))*(1.1)*[-1,1]);
85  title('Convergence of Real GA');
86  saveas(fig1,strcat(save_dir,'ga_real_conv'),'epsc');
```

Listing 13: Cross-over function

```matlab
1   function offspring = crossover(mating_pool,p_xover, method)
2   switch nargin
3       case 2
4           method='av';
5   end
6   %shuffle mating pool
7   [Npop,¬] =  size(mating_pool);
8   mating_pool = mating_pool(randperm(Npop), :);
9   rand_num = rand(round(Npop/2),1);
10  do_xover = rand_num < p_xover;
11  do_xover_idx = find(do_xover);
12
13  alpha = rand(length(do_xover_idx),1);
14  offspring = mating_pool;
15  if strcmp(method,'av')
16      for i = 1:length(do_xover_idx)
17          k = do_xover_idx(i);
18          parents = mating_pool([2*k-1 2*k],:);
19          offspring([2*k-1 2*k], :) = [sum(parents, 1)/2.0;
20                                       parents((alpha(i)> 0.5)+1,:)];
21      end
22  elseif strcmp(method,'conv_combo')
23      for i = 1:length(do_xover_idx)
```

```
24          k = do_xover_idx(i);
25          parents = mating_pool([2*k-1 2*k],:);
26          offspring([2*k-1 2*k], :) = [alpha(i), 1-alpha(i);
27                                       1-alpha(i), alpha(i)]*parents;
28      end
29  else
30      error('Method for crossover not implemented');
31  end
32
33  end
```

Listing 14: Mutation function

```
1  function mutated = mutation(parents, p_mut, lb, ub)
2  [N_pop,¬] = size(parents);
3  rand_nums = rand(N_pop,1);
4  do_mut_idx = find(rand_nums < p_mut);
5  mutated = parents;
6  alpha = rand(length(do_mut_idx),1);
7  w = rand(length(do_mut_idx),length(lb));
8  % scale and translate w to domain
9  w = w.*(ub-lb) + lb;
10 mutated(do_mut_idx,:) = parents(do_mut_idx,:).*alpha + w.*(1-alpha);
11 end
```

Linear programming Code

Listing 15: Linprog Main Code

```
1  % ECE 580 HW5: Problem 3
2  % Rahul Deshmukh
3  % deshmuk5@purdue.edu
4  clc; clear all;
5  format rat;
6  %% include paths
7  addpath('../OptimModule/optimizers/linprog/');
8  %%
9  verbose=0;
10 c = [4; 3];
11 A = [-5, -1;
12      -2, -1;
13      -1, -2];
14 b = [-11; -8; -7];
15 Aeq = [];
16 beq = [];
17 LB=[];
18 UB=[];
19 [x_str, fval] = mylinprog(c,A,b,Aeq,beq,LB,UB,verbose)
```

Listing 16: Two Phase Simplex

```
1  function [x_str, fval] = mylinprog(c,A,b,Aeq,beq,LB,UB,verbose)
2  % Two Phase Simplex LP solver
```

```matlab
 3  % the problem is not given in std form
 4  % Given: min c'*x
 5  % st:    Aeq*x = beq
 6  %         Ax <= b
 7  %         0 <= x  or LB <= x <= UB
 8  if nargin==7
 9      verbose=0; %by default
10  end
11  % transform variables: z = x - LB if LB is not 0
12  if ¬isempty(LB)
13  %      error('Not implemented');
14      b = b - A*LB;
15      if ¬isempty(Aeq)
16          beq = beq - Aeq*LB;
17      end
18  end
19  % stack ineqaulity conditions Iz <= UB-LB
20  if ¬isempty(UB)
21      A = [A;eye(length(UB))];
22      b = [b; UB-LB];
23  end
24  %% Convert to std form using slack and surplus variables st b_std>=0
25  A_std =[]; b_std = []; c_std =[];
26  A_std = [A_std; A, eye(size(A,1))];
27  b_std = [b_std;abs(b)];
28  c_std = [c_std;c;zeros(size(A,1),1)];
29  % make inequality b's positive
30  neg_b_idx = find(b < 0);
31  A_std(neg_b_idx,:) = -1*A_std(neg_b_idx,:);
32  % stack equality equations
33  if ¬isempty(Aeq)
34      neg_b_idx = find(beq < 0);
35      Aeq(neg_b_idx,:) = -1*Aeq(neg_b_idx,:);
36      A_std = [A_std; Aeq, zeros(size(Aeq,1),size(A,1))];
37      b_std = [b_std; abs(beq)];
38  end
39  %% Solve std form using two-phase simplex method
40  if verbose
41      fprintf('\\text{Phase 1:}&\\nonumber\\\\\n%%\n');
42  end
43  % Phase1: Find initial basis using artificial variables
44  A1 = [A_std, eye(size(A_std,1))];
45  c1 = [zeros(size(A_std,2),1);ones(size(A_std,1),1)];
46  basis_idx = size(A_std,2) + (1:size(A_std,1));
47  [¬, ¬, basis_idx_p1, tab]= simplex(A1,b_std,c1, basis_idx, verbose);
48  if verbose
49      fprintf('\\text{Phase 2:}&\\nonumber\\\\\n%%\n') ;
50  end
51  % Phase2: Find optimal solution
52  A2 = tab(1:end-1,1:size(A_std,2));
53  b2 = tab(1:end-1,end);
54  [x_str_p2, fval,¬, ¬] = simplex(A2, b2, c_std, basis_idx_p1, verbose);
55  x_str = x_str_p2(1:length(c));
56  if ¬isempty(LB)
57      % transform from z_str to x_str, x_str= z_str + LB
58      x_str = x_str + LB;
59      fval = fval + c'*LB;
60  end
61  fprintf('\n** Optimum Solution found using mylinprog **\n');
```

```
62  if verbose
63      display(x_str);
64      display(fval);
65  end
66  end
```

Listing 17: Simplex Method

```
 1  function [x_str, fval, basis_idx, tab]= simplex(A,b,c, basis_idx, verbose)
 2  % simplex method to solve LP in std form:
 3  % Given: min c^Tx
 4  % st:     Ax = b
 5  %          x≥0
 6  % basis_idx the indices are in order of std cartesian basis 1...n
 7  tol = 1e-6;
 8  tab = [A, b;
 9         c',0];
10  if verbose
11      print_tab(tab);
12  end
13
14  % make cost coeffs zero for basis idx
15  for i=1:length(basis_idx)
16  %     r = find(tab(1:end-1,basis_idx(i))>0);
17      tab(end,:) = tab(end,:) - tab(end,basis_idx(i))*tab(i,:);
18  end
19
20  % till all cost coeffs are non-negative do:
21  while ¬isempty(find(tab(end,1:end-1)< 0))
22  % choose pivot column
23      [¬,p] = min(tab(end,1:end-1));
24  % find pivot element
25      % using only positive a_p's
26      pos_ap_idx = find(tab(1:end-1,p)>0);
27      [¬,q_idx] = min(tab(pos_ap_idx,end)./tab(pos_ap_idx, p));
28      q = pos_ap_idx(q_idx);
29      % update basis idx
30      basis_idx(q) = p;
31  % print tab with pivot element
32      if verbose
33          print_tab(tab,[q,p]);
34      end
35      % make pivot 1
36      tab(q,:) = tab(q,:)/tab(q,p);
37  % make pivot column
38      for r=1:size(tab,1)
39          if r≠q
40              tab(r,:) = tab(r,:) - tab(r,p)*tab(q,:);
41          end
42      end
43      tab(tab>-tol & tab<tol) = 0;
44  end
45  if verbose
46      print_tab(tab)
47  end
48  %return x_str and basis_idx
49  x_str = zeros(length(c),1);
50  x_str(basis_idx) = tab(1:end-1,end);
```

```
51  fval= c'*x_str;
52  end
53
54  function print_tab(tab, pivot)
55  if nargin==1
56      pivot = 0;
57  end
58  % display(tab)
59  % print latex bmatrix
60  [n_row, n_col] = size(tab);
61  fprintf('&=\\begin{bmatrix}\n');
62  for r=1:n_row
63      for c=1:n_col-1
64          if r==pivot(1) && c== pivot(2)
65              % this is the pivot
66              fprintf('\\fbox{{\\color{red}%s}}&\t',strtrim(rats(tab(r,c))));
67          else
68              fprintf('%s&\t',strtrim( rats(tab(r,c)) ));
69          end
70      end
71      fprintf('%s\\\\\n',strtrim( rats(tab(r,c+1)) ));
72  end
73  fprintf('\\end{bmatrix}\\nonumber\n%%\n');
74  end
```

Listing 18: LP output

```
1    ** Optimum Solution found using mylinprog **
2
3   x_str =
4
5           3
6           2
7
8
9   fval =
10
11          18
```