SOA HW1

Fredrik Sitje
950314-8232

September 2023

Problem 1.1, Penalty Method

In this problem, we shall use the penalty method (see pp. 30-33 in the course book) to find the minimum of the function

$$f(x_1, x_2) = (x_1 - 1)^2 + 2(x_2 - 2)^2,$$

subject to the constraint

$$g(x_1, x_2) = x_1^2 + x_2^2 - 1 \leq 0.$$

In the following set define the function $f_p(\mathbf{x}; \mu)$, compute analytically the gradient $\nabla f_p(\mathbf{x}; \mu)$, find analytically the unconstrained minimum (i.e. for $\mu = 0$) of the function and write a Matlab program for solving the unconstrained problem of finding the minimum of $f_p(\mathbf{x}; \mu)$ using the method of *gradient descent*.

The function $f_p(x_1, x_2; \mu)$ is defined as

$$f_p(x_1, x_2; \mu) = f(x_1, x_2) + p(x_1, x_2; \mu) \tag{1}$$

$$p(\mathbf{x}; \mu) = \mu \left(\sum_{i=1}^{m} (\max\{g_i(\mathbf{x}, 0)\})^2 + \sum_{i=1}^{k} (h_i(\mathbf{x}))^2 \right). \tag{2}$$

Since there is only one, two dimensional inequality constraint the penalty function $p(\mathbf{x}; \mu)$ reduces to

$$p(x_1, x_2; \mu) = \mu(\max\{x_1^2 + x_2^2 - 1, 0\})^2. \tag{3}$$

The function $f_p(x_1, x_2; \mu)$ can thus be defined as follows

$$f_p(x_1, x_2; \mu) = \begin{cases} (x_1 - 1)^2 + 2(x_2 - 2)^2 + \mu(x_1^2 + x_2^2 - 1)^2, & \text{if } x_1^2 + x_2^2 \geq 1 \\ (x_1 - 1)^2 + 2(x_2 - 2)^2 & \text{else.} \end{cases} \tag{4}$$

Now to analytically compute the gradient $\nabla f_p(x_1, x_2; \mu)$ we must distinguish between the two cases. Starting with the case $x_1^2 + x_2^2 \geq 1$ for which the gradient looks like this

$$\frac{\partial f_p}{\partial x_1} = 2(x_1 - 1) + 4\mu(x_1^2 + x_2^2 - 1)x_1 \tag{5}$$

$$\frac{\partial f_p}{\partial x_2} = 4(x_2 - 2) + 4\mu(x_1^2 + x_2^2 - 1)x_2. \tag{6}$$

For the case $x_1^2 + x_2^2 < 1$ the gradient is expressed as

$$\frac{\partial f_p}{\partial x_1} = 2(x_1 - 1) \tag{7}$$

$$\frac{\partial f_p}{\partial x_2} = 4(x_2 - 2). \tag{8}$$

One can now analytically find the minimum of the unconstrained function $f_p(x_1, x_2; \mu = 0)$ by setting both equations (7) and (8) equal to zero and find their roots. This method provides us with

$$\frac{\partial f_p(\mu = 0)}{\partial x_1} = 2(x_1 - 1) \stackrel{!}{=} 0 \rightarrow x_1 = 1 \tag{9}$$

$$\frac{\partial f_p(\mu = 0)}{\partial x_2} = 4(x_2 - 2) \stackrel{!}{=} 0 \rightarrow x_2 = 2. \tag{10}$$

Table 1: The minimum of $f_p(x_1, x_2; \mu)$ obtained for different values of μ. As μ increases, $\mathbf{x}^*(\mu)$ seems to approach the a minimum of $f(x_1, x_2)$ subject to the constraint $g(x_1, x_2) \leq 0$.

μ	$(\mathbf{x}_1^*, \mathbf{x}_2^*)(\mu)$
0	(1,2)
1	(0.4338, 1.2102)
10	(0.3314, 0.9955)
100	(0.3137, 0.9553)
500	(0.3120, 0.9512)
100	(0.3118, 0.9507)

Thus one will find a minimum for the unconstrained problem ($\mu = 0$) at point $P(1, 2)$ with the function value of $f_p(1, 2; \mu = 0) = 0$. This minima is further used as a starting point to evaluate the constrained problem $f_p(x_1, x_2; \mu)$ with a penalty method using gradient descent in Matlab. For the gradient descent we chose the suitable parameter values for step size $\eta = 0.0001$, threshold $T = 10^{-6}$, and a sequence of μ values as 1, 10, 100, 500, 1000. The derivatives were hard coded. The results are portrayed both in the table (1) and in figure (1). Both indicate a convergence

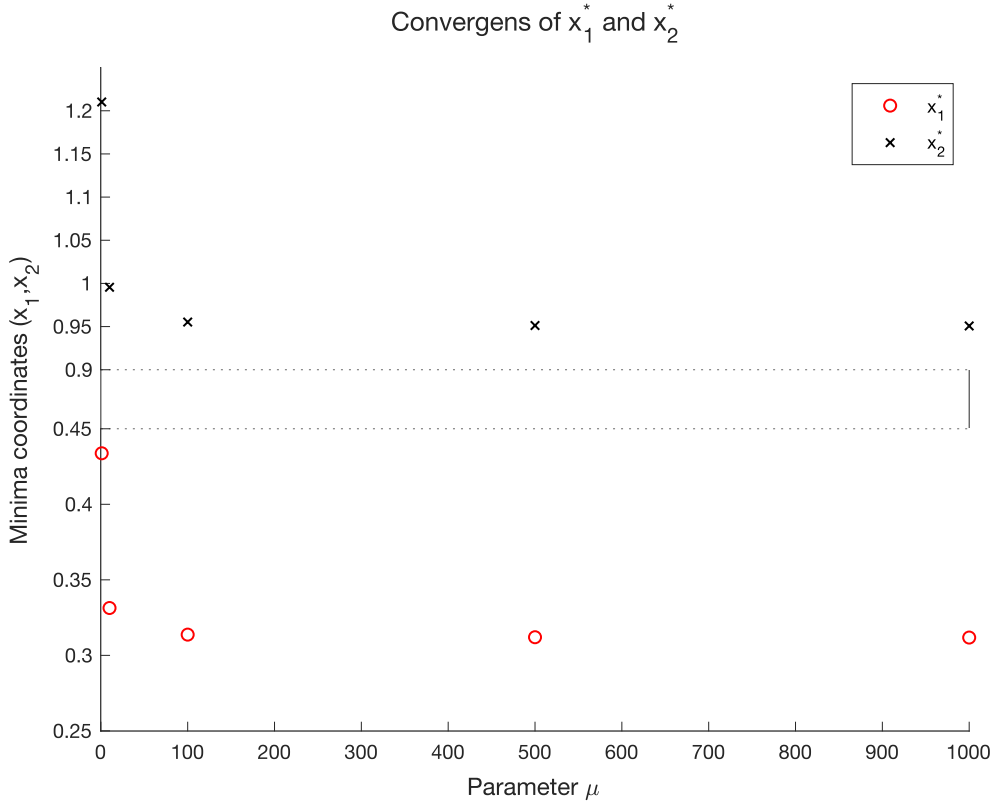

Figure 1: Here one can see the values for $\mathbf{x}^*(\mu)$ as a function of μ. With growing μ, $\mathbf{x}^*$ some convergent behaviour arises until the state where $(x_1, x_2) = (0.3118, 0.9507)$ for $\mu = 1000$. The other points in the figure have a value μ which is chosen to be an element of either 1, 10, 100, 500 or 1000.

of $\mathbf{x}^*(\mu)$ for higher values of μ. This is due to the fact that gradient decent tries to minimise the objective function $f_p(x_1, x_2; \mu)$ in which the inequality constraint gains more weight as the value of μ rises. In other terms, the optimisation algorithm becomes more aggressive on enforcing the constraint. So an increased value of μ leads to a higher significance of the penalty term all the while the algorithm tries to minimise the whole objective function $f_p(x_1, x_2; \mu)$. And thus one can see a convergence pattern in figure (1). While toying with the code in Matlab it is evident that there is dependence of the value of μ on the initial starting point (x_1, x_2). There seems to be a region of feasibility for the values of (x_1, x_2) with corresponding values of μ in which the optimisation algorithm still works. This region tends to shrink as μ rises.

Problem 1.2, Constrained optimisation

a) An analytical method for optimisation under inequality constraints

Use the analytical method described on pp. 29-30 in the course book to determine the global minimum $(x_1^*, x_2^*)^T$ (as well as the corresponding function value) of the function

$$f(x_1, x_2) = 4x_1^2 - x_1x_2 + 4x_2^2 - 6x_2$$

on the (closed) set S, shown in the figure. The corners of the triangle are located at (0,0), (0,1) and (1,1).

$$f(x_1, x_2) = 4x_1^2 - x_1x_2 + 4x_2^2 - 6x_2 \tag{11}$$

$$\frac{\partial f}{\partial x_1} = 8x_1 - x_2 \stackrel{!}{=} 0 \rightarrow x_2 = 8x_1 \tag{12}$$

$$\frac{\partial f}{\partial x_2} = 8x_2 - x_1 - 6 \stackrel{!}{=} 0 \rightarrow x_1 = \frac{2}{21} \tag{13}$$

$$\boxed{x_1 = \frac{2}{21}, x_2 = \frac{16}{21}} \tag{14}$$

At the point $P_1(2/21, 16/21)$ there is a optima with the value $f(2/21, 16/21) = -2.14$.

Since the partial derivative of f exists everywhere on **S**, all that remains to do is to consider the boundary $\partial\mathbf{S}$, which can be divided into three parts.

A. $x_1 = 0, 0 < x_2 < 1$

B. $0 < x_1 < 1, x_2 = 1$

C. $0 < x_1 < 1, 0 < x_2 < 1$

For A. it follows

$$f(0, x_2) = 4x_2^2 - 6x_2 \tag{15}$$

$$\frac{\partial f}{\partial x_2} = 8x_2 - 6 \stackrel{!}{=} 0 \tag{16}$$

$$\boxed{x_1 = 0, x_2 = \frac{3}{4}} \tag{17}$$

At the point $P_2(0, 3/4)$ there is a optima with the value $f(0, 3/4) = -2.25$.

For B. it follows

$$f(x_1, 1) = 4x_1^2 - x_1 - 2 \tag{18}$$

$$\frac{\partial f}{\partial x_1} = 8x_1 - 1 \stackrel{!}{=} 0 \tag{19}$$

$$\boxed{x_1 = \frac{1}{8}, x_2 = 1} \tag{20}$$

At the point $P_3(1/8, 1)$ there is a optima with the value $f(0, 3/4) = -2.06$.

For C. it follows

$$x_1 = x_2 \tag{21}$$

$$f(x_1, x_1) = 4x_1^2 - x_1^2 - 4x_1^2 - 6x_1 \tag{22}$$

$$\frac{\partial f}{\partial x_1} = 8x_1 - 2x_1 + 8x_1 - 6 \stackrel{!}{=} 0 \tag{23}$$

$$\boxed{x_1 = \frac{3}{7}, x_2 = \frac{3}{7}} \tag{24}$$

At the point $P_4(3/7, 3/7)$ there is a optima with the value $f(3/7, 3/7) = -1.29$.

Now we have to consider the corner:

$P_5(0,0)$: $f(0,0) = 0$

$P_6(0,1)$: $f(0,1) = -2$

$P_7(1,1)$: $f(1,1) = 1$

Thus there are seven points to consider. The global minimum is in point $P_2(0, 3/4)$ with a function value of $f(0, 3/4) = -2.25$.

b) The Lagrangian multiplier method

Use the Lagrange multiplier method described on pp. 25-28 in the course book to determine the minimum $(x_1^*, x_2^*)^T$ (as well as the corresponding function value) of the function $f(x_1, x_2) = 15 + 2x_1 + 3x_2$ subject to the constraint $h(x_1, x_2) = x_1^2 + x_1x_2 + x_2^2 - 21 = 0$.

$$f(x_1, x_2) = 15 + 2x_1 + 3x_2 \tag{25}$$

$$\text{Constraint: } h(x_1, x_2) = x_1^2 + x_1x_2 + x_2^2 - 21 = 0 \tag{26}$$

$$L(x_1, x_2, \lambda) = f(x_1, x_2) + \lambda h(x_1, x_2) \tag{27}$$

Now, lets set up the necessary equations:

$$\frac{\partial L}{\partial x_1} = 2 + \lambda(2x_1 + x_2) = 0 \tag{28}$$

$$\frac{\partial L}{\partial x_2} = 3 + \lambda(x_1 + 2x_2) = 0 \tag{29}$$

$$\frac{\partial L}{\partial \lambda} = x_1^2 + x_1x_2 + x_2^2 - 21 = 0 \tag{30}$$

From equation (28) it follows that

$$\lambda = -\frac{2}{(2x_1 + x_2)} \tag{31}$$

From equations (29) and (31) it follows that

$$3 - \frac{2}{(2x_1 + x_2)}(x_1 + 2x_2) = 0 \tag{32}$$

$$3 = \frac{2x_1 + 4x_2}{2x_1 + x_2} = 1 + \frac{3x_2}{2x_1 + x_2} \tag{33}$$

$$2 = \frac{3x_2}{2x_1 + x_2} \tag{34}$$

$$4x_1 + 2x_2 = 3x_2 \tag{35}$$

$$\boxed{x_1 = \frac{1}{4}x_2} \tag{36}$$

From equations (30) and (36) it follows that

$$(\frac{1}{4}x_2)^2 + \frac{1}{4}x_2x_2 + x_2^2 - 21 = 0 \tag{37}$$

$$\frac{1}{16}x_2^2 + \frac{4}{16}x_2^2 + \frac{16}{16}x_2^2 = 21 \tag{38}$$

$$\frac{1}{16}x_2^2 = 1 \tag{39}$$

$$\boxed{x_2 = \pm 4} \tag{40}$$

This result in combination with equation (36) leads to

$$x_1 = \pm 1 \tag{41}$$

and

$$x_2 = \pm 4 \tag{42}$$

Now one has to check which of the four potential pairs full fill the constraint function. Let's start off by setting $x_1 = 1$ in the constraint function

$$f_{const}(x_1 = 1) = x_2^2 + x_2 - 20 \stackrel{!}{=} 0 \tag{43}$$
$$\rightarrow x_2 = 4 \tag{44}$$

Now lets check for $x_1 = -1$

$$f_{const}(x_1 = -1) = -x_2^2 + x_2 - 20 \stackrel{!}{=} 0 \tag{45}$$
$$\rightarrow x_2 = -4 \tag{46}$$

So there are actually two points to consider

$$P_1(1, 4) \Rightarrow f(1, 4) = 15 + 2 + 12 = 29 \tag{47}$$
$$P_2(-1, -4) \Rightarrow f(-1, -4) = 15 - 2 - 12 = 1 \tag{48}$$

So at the point $P_2(-1, -4)$ we find the minima with a function value of $f(-1, -4) = 1$.

Problem 1.3, Basic genetic algorithm program

In this problem, you will implement a genetic algorithm (GA) for finding the minimum of the function

$$g(x_1, x_2) = (1.5 - x_1 + x_1x_2)^2 + (2.25 - x_1 + x_1x_2^2)^2 + (2.625 - x_1 + x_1x_2^3)^2$$

in the range [-5,5] (for both x_1 and x_2). You must use skeleton files for Matlab provided for this problem. Also do some parameter search for the mutation rate. For this, run batches of 100 runs for each different mutation probability p_{mut}. You tabulate the median performance (= fitness values) over the 100 runs, as a function of the mutation probability p_{mut}. Use at least 10 different values of p_{mut} in the range of [0,1], including $p_{\text{mut}} = 0$ and $p_{\text{mut}} = 0.02$. Since the GA will only provide a numerical solution of the true minimum, one has to guess the true values of x_1 and x_2. Take this guess and prove analytically that the point $(x_1^*, x_2^*)^T$ actually is a stationary point of the function g. (No need to prove that this is a minimum). Do the derivations of g by hand and include intermediate steps!

In table 2 one can see the results of evaluated coordinates for x_1 and x_2, as well as the function value for $g(x_1, x_2)$. The chosen parameters are stated in the table, and were somewhat chosen at random. The values seem to be around $x_1 \approx 3$ and $x_2 \approx 0.5$ which can be used to make an elaborated guess of the minima location point $P_{\min}(x_1, x_2)$.

Table 2: The output of ten measurements for the EA algorithm with the parameters where changes were allowed. tournamentSize = 8, tournamentProbability = 0.75, crossoverProbability = 0.8, mutationProbability = 0.03 and numberOfGenerations = 2000. The value for $g(x_1, x_2)$ is derived from the fitness value = $\frac{1}{g(x_1,x_2)+1}$. Fortunately there are several measurements where $g(x_1, x_2)$ is almost zero, which corresponds to the highest fitness value. From this one could already make the elaborated guess that for $x_1 = 3$ and $x_2 = 0.5$ the function $g(x_1, x_2)$ has a root.

x_1	x_2	$g(x_1, x_2)$
2.9687	0.4922	0.0002
3.0018	0.5005	$6.0 \cdot 10^{-7}$
3.1250	0.5294	0.0022
3.0078	0.5019	$9.7 \cdot 10^{-6}$
3.0078	0.5019	$9.7 \cdot 10^{-6}$
3.7500	0.6386	0.0438
2.4999	0.3515	0.0727
3.7500	0.6386	0.0438
2.9687	0.4922	0.0002

Next it was possible to do a parameter search concerning the mutation rate P_{mut}. For this a set of mutation rates were chosen and after 100 runs per mutation rate a median fitness value was calculated. The results are shown in table 3. The interval of P_{mut} was chosen because this is usually the region for which the mutations are seldom enough to not be too harmful to the chromosomes as a whole and in the region where mutations happen at such a rate that it seems to worsen the chromosomes. If the mutation probability goes in higher regions of P_{mut} like portrayed in figure 2 then it starts to be counter productive since chromosomes with high fitness values are exposed to high probability of mutating their chromosome to a worse one. This can be understood while looking at table 3 which shows the median fitness values for different mutation probabilities. To a certain degree surprisingly the mutation probability $P_{\text{mut}} = 0.6$ does show a very high median fitness value. Most probably this shows that this genetic algorithm is very robust or the problem in question is an easy one to solve for this kind of algorithm. One of the possible downsides to such a high mutation probability may be high computational cost since the algorithm could tend to converge slower than with a lower mutation probability.

The median fitness values tells us for each mutation probability what the fitness value at 50% of all 100 runs is. Sadly this does not indicate if the maximum potential fitness value of 1 is reached in the remaining runs. One of the key takeaways of table 3 and figure 2 should be that mutation are beneficial for genetic optimisation algorithms. One can see the enormous increase from having no mutation to just one in 100 chance of mutating a gene. This is explained by the fact that genetic algorithms can get stuck in local minima, but with just a little bit of random tweaking of the genome by the mutations the local minima can be overcome.

Table 3: In this table the median fitness values of each 100 runs per mutation probability P_{mut} is shown. The mutation probabilities were chosen in the region of 0 to 0.1 and from 0.6 to 0.99.

P_{mut}	**Median fitness value**
0	0.6578
0.01	0.9489
0.02	0.9580
0.03	0.9931
0.04	0.9931
0.06	0.9978
0.08	0.9998
0.1	1.0
0.6	0.9932
0.7	0.9755
0.8	0.9465
0.9	0.8164
0.99	0.7303

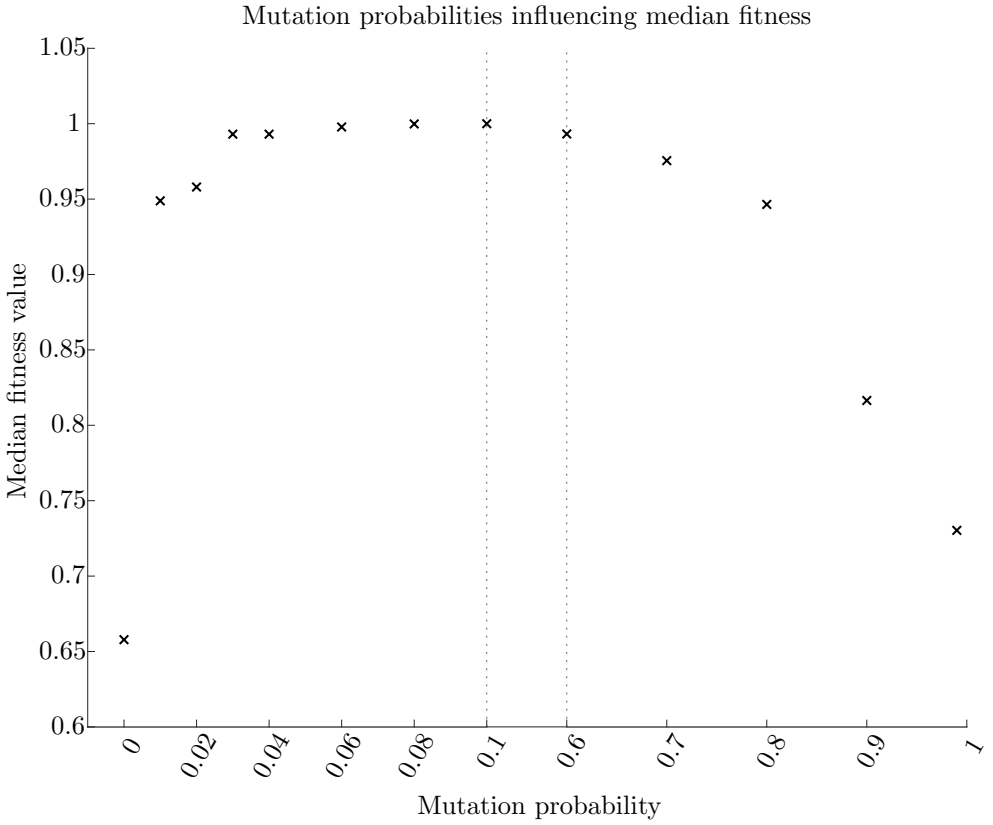

Figure 2: In this figure the median fitness values of each 100 runs per mutation probability P_{mut} is shown. The mutation probabilities were chosen in the region of 0 to 0.1 and 0.6 to 0.99 and the fitness values reach from 0.6578 for $P_{\text{mut}} = 0$ to 1.0 for $P_{\text{mut}} = 0.1$. Also one can see a somewhat negative convex relationship between the median fitness value and the mutation probability. This is due to the fact that for zero and low mutation rates the algorithm gets stuck in local minima and for high mutation rates the good chromosomes have a high probability to get worse, since their genes get changed every round.

The values for the elaborated guess of the roots for the optimum are already stated in table 2 and are $x_1 = 3, x_2 = 0.5$. It remains to prove that it actually is a stationary point, so that it can be confirmed that it actually is an optimum. It can be proven by taking the derivative of $g(x_1, x_2)$ with respect to both x_1 and x_2, and then put in $x_1 = 3, x_2 = 0.5$. Both derivatives must equal zero in order to be sure, that the found point $P(x_1, x_2)$ actually is a stationary point. The derivatives

are calculated analytically with the power rule and the chain rule.

$$\frac{\partial g}{\partial x_1} = 2(1.5 - x_1 + x_1 x_2)(x_2 - 1) + 2(2.25 - x_1 + x_1 x_2^2)(x_2^2 - 1) + 2(2.625 - x_1 + x_1 x_2^3)(x_2^3 - 1) \tag{49}$$

$$\frac{\partial g}{\partial x_2} = 2(1.5 - x_1 + x_1 x_2)x_1 + 2(2.25 - x_1 + x_1 x_2^2)(2x_1 x_1) + 2(2.625 - x_1 + x_1 x_2^3)(3x_1 x_2^2) \tag{50}$$

The equations evaluated for $x_1 = 3$ and $x_2 = 0.5$ are

$$\frac{\partial g(x_1, x_2)}{\partial x_1} = -(\underbrace{1.5 - 3 + 3 \cdot 0.5}_{0}) - 1.5(\underbrace{2.25 - 3 + 3 \cdot 0.5^2}_{0}) - 1.75(\underbrace{2.625 - 3 + 3 \cdot 0.5^3}_{0}) \tag{51}$$

$$\frac{\partial g(x_1, x_2)}{\partial x_2} = 6(\underbrace{1.5 - 3 + 3 \cdot 0.5}_{0}) + 6(\underbrace{2.25 - 3 + 3 \cdot 0.5^2}_{0}) + 6.75(\underbrace{2.625 - 3 + 3 \cdot 0.5^3}_{0}) \tag{52}$$

So as one can see the elaborated guess was the right choice the point $P(x_1 = 3, x_2 = 0.5)$ is stationary. This shows the usefulness of genetic algorithms in such a task.