University of Dundee

School of Science and Engineering

The Factory Location Problem

Author:
Thomas Senior 160011739

December 17, 2019

Contents

1 Introduction **2**

2 Method **4**
2.1 BFGS 4
2.2 Golden section search method with inexact search 5

3 Result **5**
3.1 Sensitive analysis 5

4 Conclusion **7**

References **8**

Appendix **8**
BFGS code with golden section inexact search 8
Objective function 10
Differential of objective function 10
Data 11

1 Introduction

In this project we will consider the Factory Location Problem, with the aim of identifying the optimal location to minimise the cost of transporting mass to customers. There are five main customers at different locations, measured in km, and they require various products of differing mass, given in tonnes. We will make several key assumptions. First, that the distributions cost is prepositional to the straight-line distance from the customer to the factory, which translates to having a straight road to each customer in real life. Second, that the distribution cost is proportional to the mass distributed to the customer. The final major assumption is that our smaller customers are distributed evenly around the area, such that there is no certain area with a high concentration of smaller customers that we would want to prioritize.
We aim to solve the location problem using a Broyden-Fletcher-Goldfarb-Shannon (BFGS) method with inexact golden section line search to find the optimal solution. We will also consider what would happen to this optimal location if the mass sold to the five main customers were to change by up to 5%. Based on this sensitivity analysis we would expect to find an area denoting the optimal location that would not necessarily be circular, instead favoring the customer with the higher mass. The location of our five major customers is stated below in the table;

Customers	**Locations (x,y)(in km)**	**total mass (in tonnes)**
1	(22,9)	23
2	(12,39)	15
3	(53,50)	25
4	(94,21)	18
5	(50,18)	14

We can see from plotting these five customers on the graph below that we would expect the optimal location to be near (50,30). We use these coordinates as an initial starting point when running our algorithm;

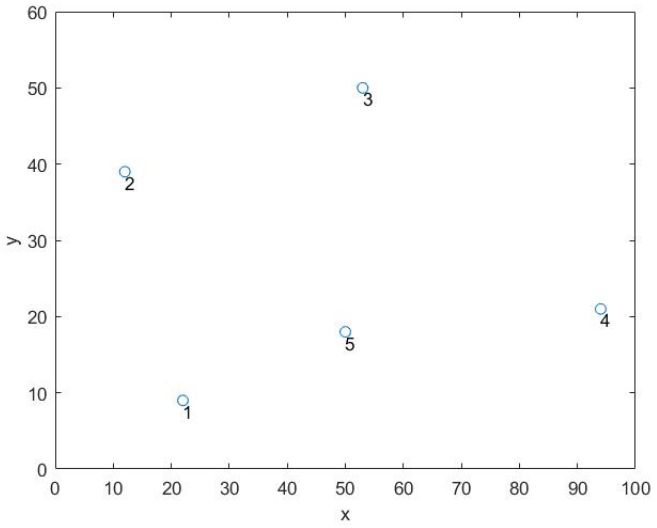
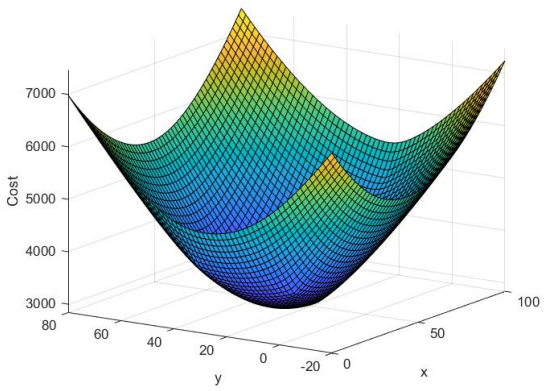


Figure 1: Customer locations

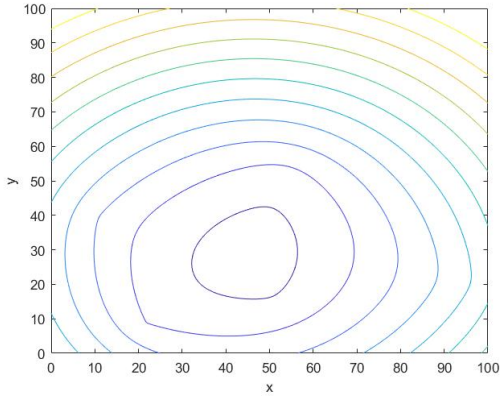
Using our previously stated assumptions we derive the objective function for this problem to be;

$$C(\mathbf{x}) = c\sum_{i=1}^{5} k_i||\mathbf{x} - \mathbf{a}|| \tag{1}$$

Where $\mathbf{x} = (x, y)$ is the location of the factory that we are trying to optimise, $\mathbf{a} = (a_i, b_i)$ is the location of the ith customer with k_i being the mass required by that customer. We have c as the cost of transporting 1 tonne by 1 km, and the objective function $C(\mathbf{x})$ that we are trying to minimise, as this is the cost annually of locating the factory at the location $\mathbf{x}$. Plotting this function on Matlab we produce a 3D graph showing the bowl shape of the function, along with its contour plot showing that the minimum cost is achieved near $[45, 30]$;



(a) 3D plot of the function



(b) Contour plot of function

The graph demonstrates that our function is a convex function [Borwein and Lewis]. The definition of a convex function of two variables is that if for all $\mathbf{x}, \mathbf{y} \in \mathbb{R}$ and for all $\lambda \in (0, 1)$ the following must hold;

$$f(\lambda\mathbf{x} + (1-\lambda)\mathbf{y}) \leq \lambda f(\mathbf{x}) + (1-\lambda)f(\mathbf{y}) \tag{2}$$

If we then consider our equation we get that we must have for our function being convex;

$$c\sum_{i=1}^{5} k_i||\lambda\mathbf{x} + (1-\lambda)\mathbf{y} - \mathbf{a}|| \leq c\lambda\sum_{i=1}^{5} k_i||\mathbf{x} - \mathbf{a}|| + c(1-\lambda)\sum_{i=1}^{5} k_i||\mathbf{y} - \mathbf{a}|| \tag{3}$$

We can start by eliminating c and only consider for i=1 because, if proven for i=1, the results will hold for all other values of i and the function overall;

$$k_1||\lambda\mathbf{x} + (1-\lambda)\mathbf{y} - \mathbf{a}|| \leq \lambda k_1||\mathbf{x} - \mathbf{a}|| + (1-\lambda)k_1||\mathbf{y} - \mathbf{a}|| \tag{4}$$

$$||\lambda\mathbf{x} + (1-\lambda)\mathbf{y} - \mathbf{a}|| \leq \lambda||\mathbf{x} - \mathbf{a}|| + (1-\lambda)||\mathbf{y} - \mathbf{a}|| \tag{5}$$

This is true via the triangle inequality, which shows that our objective function is made up of five convex functions and is therefore also convex itself. Since our objective function is convex, every local minimum is also a global minimum. Plotting the two derivatives of the function against each other and equal to zero (see below), we see that they cross at one point only, from which we conclude that there is only one stationary point and can therefore assume that our optimisation function is strictly convex with, at most, one global minimum. Given that our algorithm goes to the minimiser, we can assume that the starting vector for the BFGS method is irrelevant because it will always tend to the minimiser, given enough iterations;

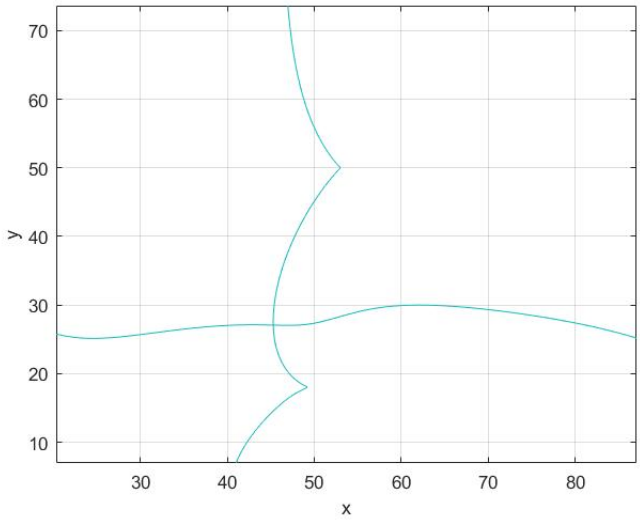


Figure 3: Intersection of the first derivatives equal to zero

2 Method

2.1 BFGS

The method we will be using to solve the objective function stated in the previous section is the BFGS method with inexact golden section line search. The BFGS method with serial variables is as follows;
a) Determine a starting vector $x^{(0)}$ and set $H^{(0)}$ and set $r = 0$ in which $H^{(r)}$ is a constructed matrix approximation for $G(\mathbf{x}^{(r)})^{-1}$.

b) We compute the search direction $\mathbf{s}^{(r)}$ by;

$$\mathbf{s}^{(r)} = -H^{(r)} \bigtriangledown f(\mathbf{x}^{(r)}). \tag{6}$$

c) We then determine the value for t that minimises $f(\mathbf{x}^{(r)} + t\mathbf{s}^{(r)})$ along the line $\mathbf{x}(t) = \mathbf{x}^{(r)} + t\mathbf{s}^{(r)}$, in which we will be using the golden section search method. We will use Wolfe conditions, explained on the next page. We will stop if our desired accuracy for our $\mathbf{x}^{(r)}$ is met, or is clearly not converging.

d) If our desired accuracy has not been met, we will update the approximation for the inverse Hessian matrix $H^{(r)}$ as;

$$H^{(r+1)} = H^{(r)} + \left(1 + \frac{\mathbf{w}^T \Delta g^{(r)}}{\mathbf{v}^T \Delta g^{(r)}}\right) \frac{\mathbf{v}\mathbf{v}^T}{\mathbf{v}^T \Delta g^{(r)}} - \frac{\mathbf{v}\mathbf{w}^T + \mathbf{w}\mathbf{v}^T}{\mathbf{v}^T \Delta g^{(r)}} \tag{7}$$

With;

$$\mathbf{v} = \Delta^{(r)} = \mathbf{x}^{(r+1)} - \mathbf{x}^{(r)} \tag{8}$$

$$\mathbf{w} = H^{(r)} \Delta g^{(r)} = H^{(r)} \left(\bigtriangledown f(\mathbf{x}^{(r+1)}) - \bigtriangledown f(\mathbf{x}^{(r)}) \right) \tag{9}$$

Then set $r = r + 1$ and repeat from step (b) until our desired accuracy is met, or until the stopping criteria exceeds a certain amount of iterations.

2.2 Golden section search method with inexact search

The golden section search method with inexact search used in the BFGS method above is as follows;
a) First let $t_0 = a, t_1 = b - \gamma(b-a), t_2 = a + \gamma(b-a)$ and $t_3 = b$, where $\gamma = \frac{1}{2}(-1+\sqrt{5}) \approx 0.618034$.

bi) Then if $f(\mathbf{x}^{(r)} + t_1\mathbf{s}^{(r)} \leq f(\mathbf{x}^{(r)} + t_2\mathbf{s}^{(r)})$, the local line minimiser is in $[t_0, t_2]$ and we define the new points as $t_0, t_2 - \gamma(t_2 - t_0), t_1$ and t_2.

bii) Otherwise, if $f(\mathbf{x}^{(r)} + t_1\mathbf{s}^{(r)} > f(\mathbf{x}^{(r)} + t_2\mathbf{s}^{(r)})$ then the local line minimiser would be in $[t_1, t_3]$ and we set the new points as $t_1, t_2, t_1 - \gamma(t_3 - t_1)$ and t_3.

c) We relabel the points as t_0, t_1, t_2 and t_3 and repeat from step b) until the Wolfe conditions are met. The Wolfe conditions are the conditions under which, when met, will give us a t value that is a sufficient reduction of the line function and meets a curvature condition preventing t from being very small. This allows us to complete the total algorithm much faster. The conditions are stated below;

$$f(\mathbf{x}^{(r)}) - f(\mathbf{x}^{(r)} + t\mathbf{s}^{(r)}) \geq -c_1 t\mathbf{s}^{(r)T} \bigtriangledown f(\mathbf{x}^{(r)}) \tag{10}$$

$$\mathbf{s}^{(r)T} \bigtriangledown f(\mathbf{x}^{(r)} + t\mathbf{s}^{(r)}) \geq c_2\mathbf{s}^{(r)T} \bigtriangledown f(\mathbf{x}^{(r)}) \tag{11}$$

With $0 < c_1 < c_2 < 1$ once the conditions are met we take that t value and continue with the BFGS method. This approach will improve the efficiency of the algorithm.

3 Result

Using the BFGS method with gold section inexact search we find that the minimiser vector for our objective function is $\mathbf{x} = [45.285227.0746]$, to four decimal places. This level of precision translates to an accuracy of 10 cm. Given that our distance from the factory is expressed in km, we can conclude that the result is acceptably accurate. The minimiser gives us a minimum cost of $\$2839.87 \times c$. The value of c is not given, but affects only the cost and not the location of the factory. We took the starting vector to be [50,30], from which our algorithm reached the minimiser in four iterations. We also tested the algorithm with starting points of [1000,1000] and [10000,10000], which both also reached the same minimiser. Plotting the optimal location with the five customers we can see that customer 4 is not as valuable. This finding likely results from the relatively low mass of the product required by customer 4, and it is more cost efficient to be nearer to customers who require more mass, such as customers 1 and 3;

3.1 Sensitive analysis

We considered how changing the mass of the products by up to 5% changed the location of the factory. The full list of results is presented at the end. In this section we will assess the most important results from the data collected. The first thing to note is that changing all the masses by the same percentage has no effect on the location of the factory, but does affect the cost of transportation. This is because the ratio between the masses is unchanged, so there is no need to move the factory to a different location, but changing the mass being moved directly affects the cost.
Next we consider the most extreme effects. The table below depicts the results of each customer's mass being increased and decreased by 5%, with the original, optimal location denoted as $\mathbf{x}^* = [45.285227.0746]$. We exclude the new cost because we are only interested in how changing the mass affects the factory location;
We can see from the table above that customer 3 has the greatest influence on the location of the factory when the mass of their product is increased or decreased by 5%. Customer 2, in contrast, had the smallest effect on

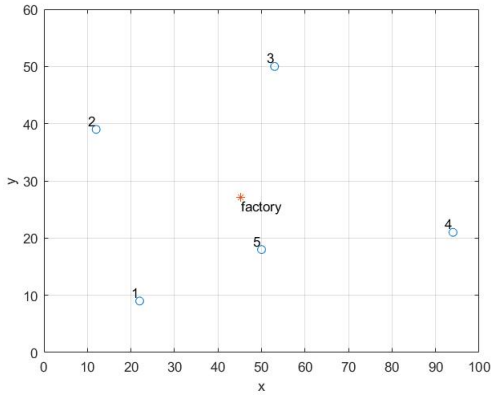


Figure 4: Optimal factory location with the five major customers

Customers	New factory location	Mass for customer i	Distance from x*
1	(44.9059, 26.6674)	24.15	0.4419
1	(45.6611, 27.5165)	21.85	0.4072
2	(44.9791, 27.2371)	15.75	0.3062
2	(45.5860, 26.9063)	14.25	0.3007
3	(45.4523, 27.8214)	26.25	0.7468
3	(45.1434, 26.3678)	23.75	0.7068
4	(45.6617, 26.9933)	18.9	0.3764
4	(44.8994, 27.1538)	17.1	0.3859
5	(45.4342, 26.6879)	14.7	0.38676
5	(45.1438, 27.4637)	13.3	0.3891

optimal factory location. These findings are somewhat surprising because customer 5 is very close and their product has the smallest mass, so therefore might be assumed to influence factory location the least. The small effect for customer 2 may be due to the fact that they are located on the opposite side of the factory to customer 4, hence moving the factory closer to customer 2 would increase the cost of delivering to customer 4. From this table we can conclude that the two most valuable customer are customers 3 and 1, because they have the greatest influence on the optimal location of the factory.

Increasing the mass of customer 4's product by 5% increases the x-axis value of factory location by the most. Considering which percentage increases or decreases have the biggest influence on factory location, we can get an approximation for the most extreme location changes in both the x- and y-axis. The table of these extreme values is given below;

Customers mass +5%	Customers mass -5%	New factory location	Distance from x*
3,4,5	1,2	(46.5527, 27.6400)	1.2674
1,2	3,4,5	(43.7995, 26.5836)	1.4858
2,3	1,4,5	(45.0675, 28.9379)	1.8633
1,4,5	2,3	(45.6589, 25.3701)	1.7045

We can see that this approach has a much greater influence on optimal factory location than if we change only one customer's mass by 5%. This gives an indication of the most extreme cases, with each location being more

then 1 km from the original, optimal location. Plotting these four points around the original location we find an approximate area that would be ideal. Such an approach better translates to a real-world scenario, in which it is often not possible to position a factory precisely at the optimal location. As these are the results for extreme

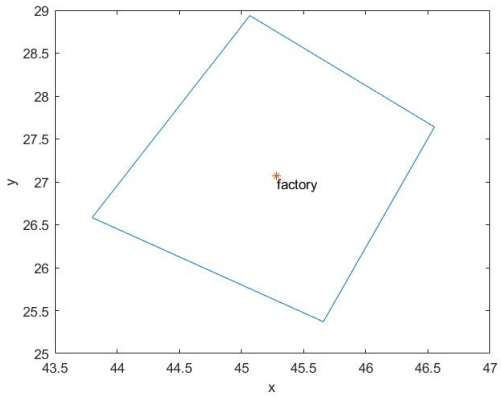


Figure 5: Optimal factory location with four extreme points

scenarios, we can assume that most untested scenarios will fall within this area. We see that the area within the extreme values is slightly biased to the left of the optimal factory location, because the most valuable customers tend to be on the left hand side. We can also plot the location of the optimal factory given that three of the customers' mass increased by 5% and two decreased by 5% (data given at the end);

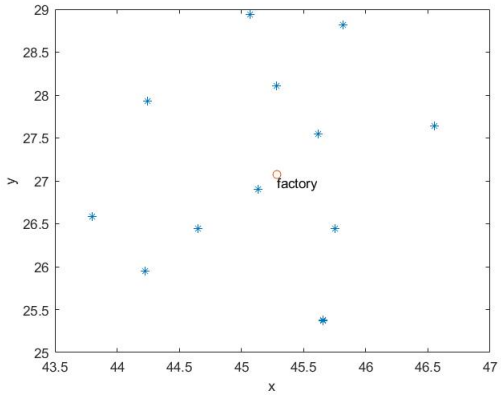


Figure 6: Optimal factory location with 14 extreme points

We can see that most of the points tend to the upper left of the graph, probably because our algorithm favours the most valuable customer, customer 3. If we were to plot all the possible percentage changes of mass for each customer, we can assume that the results would tend to favour the left hand side of the current optimal location.

4 Conclusion

Given that the mass required by customers remains constant, and the objective function is strictly convex, there exists one unique global minimum at which the factory location is optimised. The location is given by $\mathbf{x} = [45.285227.0746]$, with a minimum cost of $\$2839.87 \times c$, where c is not given but refers to the cost to

transport 1 tonne by 1 km. Considering that the mass of sales to each customer could change by up to 5%, we saw that customer 3 was the most valuable customer because changing the mass that they require had the greatest impact on the optimal location of the factory. Considering extreme scenarios of mass change across all customers gave us an approximate area within which the optimal location could change. This area is important because it is more realistic, given that companies would rarely be able to locate their factory precisely at the optimal location and the fact that sales patterns are likely to change over time. The desired area within which we would wish to locate the factory favours the region nearer to customers 3 and 2, as the most valuable customers. In the future it could be useful to drop some of the assumptions that we have made. For example, instead of evenly distributed small customers we could consider high density areas of small customers, which mathematically could translate to changes in the objective function itself. For a more accurate assessment of how much the location of the factory would actually cost, we might also consider adding more customers and accounting for how much it costs to build the factory and which locations are actually viable to build on, avoiding areas such as rivers and protected greenspaces.

References

[Borwein and Lewis] Borwein, J. and Lewis, A. (2006). *Convex Analysis and Nonlinear Optimization Theory and Examples*. Second edition.

Appendix

BFGS code with golden section inexact search

```
function BFES(a,b)
syms x y t
x0=[a b];
n=20;
H0=eye(2);
m=100;
z1=0.1;
z2=0.2;
tol2=0.001;
counter=1;
for i=1:n
    DF0=DF(x0(1),x0(2));
    s0=-H0*DF0;
    tz=x0+t*s0.';
    rhs=C(tz(1),tz(2));

    a1=-5;
    b1=5;
    gam=(-1+sqrt(5))/2;
    tol=5*10^-7;
    t1=b1-gam*(b1-a1);
    t2=a1+gam*(b1-a1);
    ft1=subs(rhs,t1);
```

```
24      ft2=subs(rhs,t2);
25
26      for j=1:m
27          while abs(b1-a1)>tol
28              if(ft1<ft2)
29              b1=t2;
30              t2=t1;
31              t1=b1-gam*(b1-a1);
32              ft1=subs(rhs,t1);
33              ft2=subs(rhs,t2);
34              tz1=x0+t1*s0.';
35              fti=C(tz1(1),tz1(2));
36              fi=C(x0(1),x0(2));
37              DFi=DF(tz1(1),tz1(2));
38              if fi-fti>=-z1*t1*s0.'*DF0 && s0.'*DFi>=z2*s0.'*DF0
39               break
40              end
41
42      else
43          a1=t1;
44          t1=t2;
45          t2=a1+gam*(b1-a1);
46          ft1=subs(rhs,t1);
47          ft2=subs(rhs,t2);
48          tz2=x0+t2*s0.';
49          fti=C(tz2(1),tz2(2));
50          fi=C(x0(1),x0(2));
51          DFi=DF(tz2(1),tz2(2));
52          if fi-fti>=-z1*t2*s0.'*DF0 && s0.'*DFi>=z2*s0.'*DF0
53               break
54          end
55              end
56          end
57      end
58
59  if(ft1<ft2)
60      t0=t1;
61  else
62      t0=t2;
63  end
64
65      x1=x0+t0*s0.';
66      err=norm(x1-x0,inf);
67      if err<tol2
68          break
69      else
70      v=x1.'-x0.';
```

```
71      DF1=DF(x1(1),x1(2));
72      g0=DF1-DF0;
73      w=H0*(g0);
74      H0=H0+(1+(w.'*g0)/(v.'*g0))*(v*v.')/(v.'*g0)-(v*w.'+w*v.')/(v.'*g0);
75      x0=x1;
76      counter=1+counter;
77      end
78 end
79
80      funt=C(x1(1),x1(2));
81      fprintf('Minimizer vector is : [');
82      fprintf('%.4f ', x1);
83      fprintf(']\n');
84      fprintf('Minimum: %.4f \n',funt)
85      fprintf('Iterations: %.4f \n',counter)
86 end
```

Objective function

```
1 function out=C(x,y)
2 x1=sqrt((x-22)^2+(y-9)^2);
3 x2=sqrt((x-12)^2+(y-39)^2);
4 x3=sqrt((x-53)^2+(y-50)^2);
5 x4=sqrt((x-94)^2+(y-21)^2);
6 x5=sqrt((x-50)^2+(y-18)^2);
7 k1=23;
8 k2=15;
9 k3=25;
10 k4=18;
11 k5=14;
12 out=(k1*x1+k2*x2+k3*x3+k4*x4+k5*x5);
13 end
```

Differential of objective function

```
1 function out=DF(x,y)
2 k1=23;
3 k2=15;
4 k3=25;
5 k4=18;
6 k5=14;
7 dfx=(k1*(2*x - 44))/(2*((x - 22)^2 + (y - 9)^2)^(1/2)) + (k2*(2*x - 24))/(2*((x
    - 12)^2+ (y - 39)^2)^(1/2)) + (k5*(2*x - 100))/(2*((x - 50)^2 + (y - 18)^2)
    ^(1/2)) +(k3*(2*x - 106))/(2*((x - 53)^2 + (y - 50)^2)^(1/2)) + (k4*(2*x -
    188))/(2*((x - 94)^2 + (y - 21)^2)^(1/2));
8 dfy=(k1*(2*y - 18))/(2*((x - 22)^2 + (y - 9)^2)^(1/2)) + (k5*(2*y - 36))/(2*((x
    - 50)^2 + (y - 18)^2)^(1/2)) + (k2*(2*y - 78))/(2*((x - 12)^2 + (y - 39)^2)
    ^(1/2)) + (k4*(2*y - 42))/(2*((x - 94)^2 + (y - 21)^2)^(1/2)) + (k3*(2*y -
    100))/(2*((x - 53)^2 + (y - 50)^2)^(1/2));
```

```
9   out=[dfx;dfy];
10 end
```

Data

% change of all masses	Minimiser	Minimum
-5%	[45.2852 27.0746]	2697.8753
-4%	[45.2852 27.0746]	2726.274
-3%	[45.2852 27.0746]	2754.6726
-2%	[45.2852 27.0746]	2783.0713
-1%	[45.2852 27.0746]	2811.47
0%	[45.2852 27.0746]	2839.8687
1%	[45.2852 27.0746]	2868.2674
2%	[45.2852 27.0746]	2896.6661
3%	[45.2852 27.0746]	2925.0648
4%	[45.2852 27.0746]	2953.4635
5%	[45.2852 27.0746]	2981.8621

% change of customer 1	Minimiser	Minimum	total change of minimiser
-5%	[45.6611 27.5165]	2805.6456	0.4419
-4%	[45.5861 27.4251]	2812.5427	0.3505
-3%	[45.5110 27.3353]	2819.4133	0.2607
-2%	[45.4358 27.2469]	2826.2577	0.1723
-1%	[45.3606 27.1600]	2833.0761	0.0854
0%	[45.2853 27.0746]	2839.8687	0
1%	[45.2098 26.9905]	2846.6356	0.0841
2%	[45.1341 26.9078]	2853.377	0.1668
3%	[45.0583 26.8264]	2860.0931	0.2482
4%	[44.9822 26.7463]	2866.784	0.3283
5%	[44.9059 26.6674]	2873.4499	0.4072

% change of customer 2	Minimiser	Minimum	total change of minimiser
-5%	[45.5860 26.9063]	2813.2232	0.3007
-4%	[45.5263 26.9404]	2818.5727	0.241
-3%	[45.4663 26.9743]	2823.912	0.181
-2%	[45.4062 27.0080]	2829.2411	0.1209
-1%	[45.3458 27.0414]	2834.56	0.0605
0%	[45.2853 27.0746]	2839.8687	0
1%	[45.2245 27.1075]	2845.1671	0.0608
2%	[45.1635 27.1403]	2850.4553	0.1218
3%	[45.1022 27.1728]	2855.7332	0.1831
4%	[45.0408 27.2050]	2861.0008	0.2445
5%	[44.9791 27.2371]	2866.2581	0.3062

% change of customer 3	Minimiser	Minimum	total change of minimiser
-5%	[45.1434 26.3678]	2809.1812	0.7068
-4%	[45.1698 26.5060]	2815.3897	0.5686
-3%	[45.1972 26.6458]	2821.5632	0.4288
-2%	[45.2256 26.7871]	2827.7012	0.2875
-1%	[45.2549 26.9301]	2833.8032	0.1445
0%	[45.2853 27.0746]	2839.8687	0
1%	[45.3166 27.2207]	2845.8974	0.1461
2%	[45.3490 27.3684]	2851.8887	0.2938
3%	[45.3824 27.5178]	2857.8421	0.4432
4%	[45.4168 27.6687]	2863.7574	0.5941
5%	[45.4523 27.8214]	2869.6338	0.7468

% change of customer 4	Minimiser	Minimum	total change of minimiser
-5%	[44.8994 27.1538]	2795.5098	0.3859
-4%	[44.9773 27.1381]	2804.41	0.308
-3%	[45.0549 27.1224]	2813.2958	0.2304
-2%	[45.1321 27.1066]	2822.1675	0.1532
-1%	[45.2089 27.0906]	2831.0251	0.0764
0%	[45.2853 27.0746]	2839.8687	0
1%	[45.3613 27.0585]	2848.6983	0.076
2%	[45.4369 27.0423]	2857.514	0.1516
3%	[45.5122 27.0260]	2866.3158	0.2269
4%	[45.5871 27.0097]	2875.1039	0.3018
5%	[45.6617 26.9933]	2883.8782	0.3764

% change of customer 5	Minimiser	Minimum	total change of minimiser
-5%	[45.1438 27.4637]	2832.5665	0.3891
-4%	[45.1715 27.3857]	2834.0499	0.3111
-3%	[45.1995 27.3077]	2835.5219	0.2331
-2%	[45.2278 27.2299]	2836.9823	0.1553
-1%	[45.2564 27.1522]	2838.4313	0.0776
0%	[45.2853 27.0746]	2839.8687	0
1%	[45.3145 26.9971]	2841.2946	0.0775
2%	[45.3440 26.9196]	2842.709	0.155
3%	[45.3737 26.8423]	2844.1119	0.2323
4%	[45.4038 26.7651]	2845.5033	0.3095
5%	[45.4342 26.6879]	2846.8831	0.3867

% change 1	% change 2	% change 3	% change 4	% change 5	Minimiser	Minimum	change of minimiser
-5%	-5%	+5%	+5%	+5%	[46.5527 27.6400]	2858.7188	1.2674
+ 5%	+5%	-5%	-5%	-5%	[43.7995 26.5836]	2816.1214	1.4858
-5%	+5%	+5%	-5%	-5%	[45.0675 28.9379]	2808.6518	1.8633
+5%	-5%	-5%	+5%	+5%	[45.6589 25.3701]	2865.773	1.7045
+5%	+5%	+5%	-5%	-5%	[44.2433 27.9282]	2877.2851	1.042
+5%	+5%	-5%	5%	-5%	[44.6526 26.4415]	2906.2557	0.6331
+5%	-5%	+5%	+5%	-5%	[45.6153 27.5435]	2914.2126	0.4689
-5%	+5%	+5%	+5%	-5%	[45.8191 28.8132]	2897.192	1.7386
+5%	+5%	-5%	-5%	+5%	[44.2230 25.9463]	2830.412	1.1283
+5%	-5%	+5%	-5%	+5%	[45.1358 26.9062]	2840.409	0.1684
-5%	+5%	+5%	-5%	+5%	[45.2851 28.1108]	2824.8601	1.0362
+5%	-5%	-5%	+5%	+5%	[45.6589 25.3701]	2865.773	1.7045
-5%	+5%	-5%	+5%	+5%	[45.7514 26.4480]	2853.0566	0.6266
-5%	-5%	+5%	+5%	+5%	[46.5527 27.6400]	2858.7188	1.2674