Solving the UFLP using Lagrangian Relaxations (Uncapacitated Facility location problem)

Kia Babashahi Ashtiani

Objective

The goal of this project was to:

1- Solve the UFLP (Uncapacitated Facility location problem) by applying the Lagrangian Relaxation method by relaxing different sets of constraints.
2- Analyze the effect of the parameter "Epsilon " on the convergence time, the number of iterations and the optimality bound of the algorithm.
3- Compare the effect of relaxing different sets of constraints with each other in terms of the convergence time and the number of iterations and the optimality bound.

Datasets

The datasets which we conducted our experiments on can be clustered in three different groups based on their size. And by size I mean the number of costumes in the problem. In this report, the datasets cap 72-cap 73-cap74,cap101, cap 102,cap103 and cap104 are considered small sized data sets. The data sets cap131,cap132,cap133, and cap134 are average sized data sets and finally, the data sets capa,,capb and capc are the large data sets.

As we shall see the runtime and the number of iterations can differ significantly based on the cluster we choose to run the relaxation on.

Note : In order to compute the upper bound of the problem, I am using the values obtained by the Heuristic provided by my professor: dr. Contreras. I am also solving the problem by using cplex to check the time gap between the time it takes the Heuristic to calculate the upper bound and the time it takes Cplex to do so.

The problem Description :

$$Min \sum_{i \in I} f_i * z_i + \sum_{i \in I} \sum_{j \in J} (d_j * c_{ij})$$

$$st \ (1) \quad x_{ij} \leq z_i \quad \forall i \in I \quad \forall j \in J$$

$$(2) \quad \sum_{i \in I} x_{ij} = 1 \ \forall j \in J$$

$$x_{ij} \in \{0,1\} \quad \forall i \in I \quad \forall j \in J$$

$$z_i \in \{0,1\} \ \forall i \in I$$

1-Relaxation of constraint one which will lead to the following problem:

$$Min \sum_{i\in I}(f_i - \sum_{j\in J}\lambda_{ij}) * z_i + \sum_{i\in I}\sum_{j\in J}(d_j * c_{ij} + \lambda_{ij})\, x_{ij}$$

$$st \quad \sum_{i\in I} x_{ij} = 1 \ \forall j \in J$$

$$x_{ij} \in \{0,1\} \quad \forall i \in I \quad \forall j \in J$$

$$z_i \in \{0,1\} \ \forall i \in I$$

Which can be decomposed into two different sub-problems one which solves the problem for the first component $\sum_{\in I}(f_i - \sum_{j\in J}\lambda_{ij}) * z_i$ and the other for which it solves this sub problem $\sum_{i\in I}\sum_{j\in J}(d_j * c_{ij} + \lambda_{ij})\, x_{ij}$ Subject to constraint number 2 and the integrality conditions.

2-Relaxation of constraint 2 which will lead to the following problem

$$Min \sum_{i\in I} f_i * z_i + \sum_{i\in I}\sum_{j\in J}(d_j * c_{ij} + \lambda_j)x_{ij} - \sum_{i\in I}\lambda_j$$

$$st \qquad x_{ij} \le z_i \quad \forall i \in I \quad \forall j \in J$$

$$x_{ij} \in \{0,1\} \quad \forall i \in I \quad \forall j \in J$$

$$z_i \in \{0,1\} \ \forall i \in I$$

Which will lead to "i" sub problems that can be solved independently.

As you can check in the code, I solved this problem by first calculating the $\sum_{j\in J}(d_j * c_{ij} + \lambda_j)$ and if the value obtained was negative, I would store only those j`s which caused the value to become negative in a vector called costumer assign which after, when I add the value above to its corresponding $f_i$ ,if the result was negative, I will only assign 1 to those $x_{ij}$`s which their corresponding value in the vector, customer assign was 1.

Tables and comparisons:

1-Cplex vs Heuristic

Table 1, which can be found at page 5, has 6 columns. The first column is associated with the names of the data sets. The second one is the Optimal Value for the problem obtained via Cplex. The third column is the Value obtained by the Heuristic. The fourth column is associated with the time it takes for the heuristic value to be calculated and the last column is the gap between the optimal solution and the heuristic.

Observation1: The heuristic as expected, finds out the solution much faster than Cplex (sometimes about 45times faster) and in 28.5% of the cases, it returns the optimal solution. In the rest of the cases, the gap between the optimal solution and the heuristic is less than 10^{-6} which shows that the Heuristic which is being used is a very accurate one.

2-The effect of epsilon:

2.1-The effect of epsilon on the first Lagrangian relaxation (relaxation of constraint1):

To analyze the effect of epsilon on the convergence time, the gap and the number of iterations, I ran the same relaxation (the first set of constraints) for each of the data sets with two different values for epsilon. (epsilon =40, which is a big number, and epsilon=2 which is in the range that the epsilon is usually selected from.)

As you can see in *table 2 and 3*, for epsilon=2, for the smaller and the average sized clusters, the number of iterations made is smaller than for the case where epsilon is equal to forty. However, working with the larger data sets epsilon=40 works 66.66% of the time better (two of the three cases) than epsilon=2. Meaning that it converges before we reach 1000 iterations. On the other hand, the gap between the Upper bound and the best lower bound obtained is mostly better in the case of epsilon=2.

Note: The tool used to capture the time of the system works with milliseconds, therefore, the values presented as time in the tables are in milliseconds.

2.2-The effect of epsilon on the Second Lagrangian relaxation (relaxation of constraint2):

This part made me struggle for a few hours, since at first, when I implemented the relaxation I sat the epsilon to 40 as the other relaxation and ran the code. However, I was not obtaining the solution of the problem. It seemed like I was doing a division by zero somewhere or I was at least dividing something by a very small value which would cause the value of the Lagrangian duals to become NAN (not a number) or in my case infinity.

The problem was with the epsilon, There is a limitation for the amount that we can increase the value of epsilon for this problem. And if we increase it more than that amount, the subgradiant algorithm will fail to compute the lagrangian values.

When I figured out what was causing the problem, I tried different values for epsilon to check which one is better in terms of the number of iterations. And more importantly, I found the maximum value epsilon can take (within a neighborhood of 0.1) that for numbers greater than that value, I will get the NAN error.

3-The comparison of the two Lagrangian Relaxations:

For this comparison, I sat the value of epsilon to two in both cases (which as we can see in the previous section, will generate feasible solutions) and analyzed the differences between the two relaxations. The tables used for this comparison are *table 3, table 4*.

The results could be generalized for datasets in the same cluster size, meaning that we can claim that for all small data sets we will have a very small number of iterations if we use the first relaxation but we will have much more iterations(sometimes around 60 times more) if we use the second relaxation.

The optimality bound (gap between the lower bound and the upper bound) is very small for both relaxations and for all of the average and small sized data-sets, we will meet the epsilon optimality criteria. Meaning that they all satisfy the condition (upper_bound - best_lower_bound) / upper_bound * 100 < 0.005) which will not happen for all of the large data-sets and the algorithm will stop when it reaches a certain number of itterations.

For the first relaxation, the number of iterations for the small data-sets is very small. Which is not the case for the second relaxation. This result also applies to the averaged size datasets. However, relaxing the second set of constraints, for the larger datasets, will result in significantly fewer number of iterations in 66.66%(two out of three) of the cases than the case for which I relaxed the first constraint.

Using the second relaxation for the large datasets, the gap between the upper bound and the lower bound is also much better (comparing the gaps in table 3 and 4). This also applies to the one data set(capa) that reached its maximum number of iterations. Which all of has to do with the convex hull of the integer solutions. Therefore we might be able to conclude that for these data sets, for the datasets with a large number of customers, the second relaxation is much better in terms of the gap and the number of iterations.

Conclusion :

Both of the relaxations solved the problem much faster than Cplex and then found very close values to the optimal value which was evaluated using Cplex. Which shows the power of the Lagrangian relaxation method.

The heuristic used in this work is very powerful. Meaning that in 13/14 cases it generates very reliable results which are close to optimality or optimal. However, in one case (capb)it generates an infeasible upper bound since the difference between the upper bound and the optimal cplex value (highlighted in table1) is negative.

The value of epsilon for the first relaxation has a direct impact on the number of iterations of the program.

The value of epsilon for the second relaxation cannot exceed a certain value and I also figured out the optimality neighborhood for epsilon to be selected from in some of the data sets in each of the clusters.

Last but not least I compared the difference between the two relaxations and although the first relaxation might seem to work better with a smaller number of customers, the second one is much better In terms of the of number of iterations and the optimality gap for large data sets.

Tables

	z^*	OPT_TIME	HEU_VALUE	HEU_TIME	VALUE of (Heuristic-Optimal solution)
cap72	4.0042145230625E9	93	4.0042145230625E9	2	0.0
cap 73	4.0042145230625E9	95	4.0042895230625E9	3	0.0
cap 74	4.0042145230625E9	91	4.0044020230625E9	3	0.0
Cap 101	2.860332101899999 6E9	107	2.8603321019E9	7	4.76837158203125E -7
Cap102	2.860452101899999 6E9	103	2.8604521019E9	5	4.76837158203125E -7
Cap103	2.860572101899999 6E9	98	2.8605721019E9	6	4.76837158203125E -7
Cap 104	2.860752101899999 6E9	100	2.8607521019E9	6	4.76837158203125E-7
Cap131	2.8503079054E9	121	2.850307905400000 6E9	17	4.76837158203125E-7
Cap132	2.8503079054E9	118	2.850535879762501 E9	17	9.5367431640625E-7
Cap133	2.8507608797625E9	116	2.850760879762501 E9	17	9.5367431640625E-7
Cap134	2.8510923056375E9	120	2.851092305637501 E9	19	9.5367431640625E-7
capa	3.145815023928299 E8	4569	3.145815023928301 E8	817	1.7881393432617188E-7
capb	2.524793786291100 4E8	3731	2.524793786291097 E8	2619	**-3.2782554626464844E-7**
Capc	2.272778159042799 8E8	3840	2.272778159042799 8E8	2286	0.0

Table1- Heuristic vs Cplex

Epsilon=40	LR1 time	OPT_TIME	Iter#=t	LR1 Value	LB	UB	(UB-LB)*100/LB
cap72	1	93	1	4.004027023062 5E9	4.004027023062 5E9	4.004214523062 5E9	0.00468256 630407994
cap 73	1	95	5	4.00411202049 10426E9	4.00411202049 10426E9	4.00428952306 25E9	0.00443281 0625582118
cap 74	1	91	5	4.00421265130 99775E9	4.00421265130 99775E9	4.00440202306 25E9	0.00472908 9422885672
Cap 101	2	107	9	2.86021315956 36315E9	2.86021315956 36315E9	2.8603321019E 9	0.00415834 0085389117
Cap102	2	103	10	2.86033436175 92015E9	2.86033436175 92015E9	2.8604521019E 9	0.00411613 7470729263 5
Cap103	2	98	10	2.86043830542 94505E9	2.86043830542 94505E9	2.8605721019E 9	0.00467726 2651786174
Cap 104	2	100	11	2.86063819056 60486E9	2.86063819056 60486E9	2.8607521019E 9	0.00398186 6652333061
Cap131	6	121	24	2.85017613669 8562E9	2.85017613669 8562E9	2.85030790540 00006E9	0.00462296 3757311514
Cap132	5	118	26	2.85040162251 4749E9	2.85040162251 4749E9	2.85053587976 2501E9	0.00470989 5030786238
Cap133	6	116	27	2.85062730153 31116E9	2.85062730153 31116E9	2.85076087976 2501E9	0.00468570 4449554506
Cap134	7	120	28	2.85096341077 0705E9	2.85096341077 0705E9	2.85109230563 7501E9	0.00452089 4203974891
capa	9298	4569	10000	3.14493525596 99005E8	3.14493997735 62306E8	3.14581502392 8301E8	0.02781621 1869234997
capb	5879	3731	6156	2.52466760067 25585E8	2.52466760067 25585E8	2.52479378629 11004E8	0.00499785 8408357472
Capc	3278	3840	3428	2.27266453712 3748E8	2.27266453712 3748E8	2.27277815904 27998E8	0.00499925 2505117334

Table 2-first relaxation with epsilon=40

Epsilon=2	LR1 time	OPT_TIME	Iter#=t	LR1 Value	LB	UB	(UB-LB)*100/LB
cap72	1	93	1	4.0040270230625E9	4.0040270230625E9	4.0042145230625E9	0.004682566304079943
cap 73	1	95	2	4.0042160630625E9	4.0042160630625E9	4.0042895230625E9	0.001834532682437443 6
cap 74	1	91	2	4.0042865630625E9	4.0042865630625E9	4.0044020230625E9	0.002883326882142021
Cap 101	2	107	2	2.8603143998E9	2.8603143998E9	2.8603321019E9	6.188826810755946E
Cap102	2	103	2	2.8604029998E9	2.8604029998E9	2.8604521019E9	0.001716585289692126 3
Cap103	2	98	2	2.8604875688E9	2.8604875688E9	2.8605721019E9	0.002955111669576778 4
Cap 104	2	100	2	2.8606132688E9	2.8606132688E9	2.8607521019E9	0.004853027981966598 5
Cap131	6	121	2	2.8502775104974496E9	2.8502775104974496E9	2.8503079054000006E9	0.001066372599495654 2
Cap132	5	118	2	2.8504769156250095E9	2.8504769156250095E9	2.8505358797625 01E9	0.002068528163768945
Cap133	6	116	2	2.8506705090225096E9	2.8506705090225096E9	2.8507608797625 01E9	0.003170056830537008
Cap134	7	120	2	2.8509584668124957E9	2.8509584668124957E9	2.8510923056375 01E9	0.004694299961471373
capa	9298	4569	10000	3.1402299999130267E8	3.1402351460499656E8	3.1458150239283 01E8	0.177374633787822 3
capb	5879	3731	10000	2.5215429793877244E8	2.5215439376747504E8	2.5247937862910 97E8	0.128717388088978 15
capc	3278	3840	10000	2.2710115329301032E8	2.2710115329301032E8	2.2727781590427998E8	0.077729808589878 31

Table 3-first relaxation with epsilon=2

Epsilon =2	LR2 time	OPT_ TIME	Iter#=t	LR2 Value	LB	UB	(UB-LB)*100/LB
cap72	10	93	128	4.004131100097736E9	4.004131100097736E9	4.004214523062 5E9	0.002083379006 8848436
cap 73	10	95	117	4.004156490186231E9	4.004156490186231E9	4.004289523062 5E9	0.003322259179 8786685
cap 74	10	91	123	4.004223688838 1405E9	4.004223688838 1405E9	4.004402023062 5E9	0.004453454556571352
Cap 101	11	107	99	2.860254069124699E9	2.860254069124699E9	2.8603321019E9	0.002728101930 8613035
Cap102	11	103	106	2.860396777157 5837E9	2.860396777157 5837E9	2.8604521019E9	0.001934125811078376
Cap103	9	98	85	2.860535998714857E9	2.860535998714857E9	2.8605721019E9	0.001262096666572889
Cap 104	11	100	96	2.860666748083252E9	2.860666748083252E9	2.8607521019E9	0.002983614577839 6376
Cap131	13	121	118	2.850224044778 1825E9	2.850224044778 1825E9	2.850307905400 0006E9	0.002942160096429897
Cap132	20	118	109	2.850480110048 5535E9	2.850480110048 5535E9	2.850535879762501E9	0.001956464198 3016444
Cap133	13	116	112	2.850660010579 9804E9	2.850660010579 9804E9	2.850760879762501E9	0.003538324916567294
Cap134	17	120	128	2.850967224722 3153E9	2.850967224722 3153E9	2.851092305637501E9	0.004387122610450999
capa	4374	4569	10000	3.141639915712068E8	3.141639915712068E8	3.145815023928301E8	0.059391398106298524
capb	136	3731	118	2.524707300402239E8	2.524707300402239E8	2.524793786291 1004E8	0.003425463470 7599134
capc	183	3840	120	2.272668945741176E8	2.272668945741176E8	2.272778159042 7998E8	0.004805277681379945

Table 4-second relaxation with epsilon=2