SOLVING THE P-CENTER PROBLEM WITH SIMULATED ANNEALING

Tayyar Büyükbaşaran
Middle East Technical University, Industrial Eng. Dep., 06531, Ankara

Abstract: The p-center problem is a well-known NP-hard discrete location problem. This paper presents a metaheuristic, simulated annealing, for vertex p-center problem. Firstly, generic and problem specific decisions, and parameter setting of the algorithm are delineated. Secondly, the heuristic is tested on well-known p-center problems taken from literature. Thirdly, the results of the heuristic are compared with those of local search heuristics and other metaheuristic applications from the literature on the same test problem set. Simulated Annealing outperforms almost all local search heuristics in all problem instances. Moreover, it finds objective values as good as other metaheuristics in some problem instances. Finally, extension of the algorithm for the 'capacitated' case of vertex p-center is discussed.

Keywords: *Modern Heuristics, Simulated Annealing, p-Center, Facility Location*

1. Introduction

The *p*-center problem is a well-known NP-hard discrete location problem. It consists of locating *p* facilities and assigning clients to them in order to minimize the maximum distance between a client and the facility to which he or she is assigned (i.e. the closest facility). In this project, a metaheuristic, simulated annealing, is applied to vertex p-center problem (vertex means possible locations of the facilities are the clients' locations). The next section will outline the simulated annealing algorithm; Section 3 will present the computational results of the algorithm; the last section will discuss the extension of heuristic to capacitated case and will conclude.

2. Simulated Annealing Algorithm

2.1 Generic and Problem Specific Decisions

During this project, a solution of the problem is represented as an array. First p elements of this array shows the location (vertex number) of the facilities (also it represents corresponding client locations), other n-p elements of the array shows the non-facility client locations of the solution

Neighborhood structure: Two neighborhood structures are observed in the algorithm. First results from simple move that randomly selects a facility node and a non-facility node, and change them, i.e. change a non-facility demand node as a facility demand node and change a facility node as a non-facility node. Second results from complex move that selects the facility node, which gives objective function value, and changes it by a randomly selected non-facility node if objective value is not changed for (1/4.number of facilities) iterations. Otherwise, complex move is the same as simple move.

Cooling Schedule: By the recommendation of Lundy and Mees (1986) and Reeves (1995) usage of geometric cooling schedule is decided. Where t represents temperature, that schedule is $a(t)=t/(1+bt)$ where b is small, implying very slow cooling. At each temperature, only one iteration is done.

Stopping Condition: When t decreases to final temperature (t_f) or when the best known is found.

Starting Solution: Since performance of any metaheuristic should not depend on initial solution, any specific construction heuristic is not applied. Random construction of initial solution is done.

Steps of the Simulated Annealing algorithm after these decisions can be seen below:

Step 1. Set temperature (t) to starting temperature (t_s). Construct an initial solution randomly and find the objective that corresponds to this solution, assign it to current solution objective value (fcur).

Step 3. Repeat 3.1 Try a move (simple move for small problem, complex move for bigger problem)

3.2 Find objective corresponding to solution after move (fstr), find delta as delta=fstr-fcur

3.3 If (delta<0) then update current solution by making move and equating fcur to fstr

Else generate $u \in U[0,1]$, if $u < \exp(-delta/t)$ update current solution by making move and equating fcur to fstr (acceptance probability of non-improving move is $p_a=\exp(-delta/t)$)

3.4 Decrease Temperature with geometric cooling schedule

Until (Temperature (t)≤ Final Temperature (t_f)) or (best solution of algorithm = best known)

Step 4. Record the best solution and the iteration at which it is found, and finalize the program.

2.2 Parameter Setting of the Algorithm

For the algorithm described above the main parameters to be set is, number of iterations, starting temperature, final temperature and type of move. Firstly, the number of iterations directly affects the

average running time of the algorithm. Moreover, it is expected that quality of the solution be affected by the number of iterations. Therefore the total number of iteration is taken as an important control parameter. Secondly, the quality of the solution depends on t_s and t_f, although the running time of the algorithm does not depend on t_s and t_f since geometric cooling schedule is used and the number of iterations is pre-determined. Thirdly, the type of move is expected to affect the quality of solution. Furthermore, since complex move requires more computation, average running time is influenced by the move type.

In order to understand the solution structure of the problem set (first 20 problems of p median problem set of Beasley (1990) and of web site of Beasley (1990)), random searches (move new solution regardless of the objective value improvement in each iteration) for each problem instance are done. With some selected problem instances for experimentation (problem instances with bold numbers in Table 1), full factorial design of experiment is used in order to set the values for control parameters.

Experimentation values for number of iterations: Required number of iteration is taken as a function of number of clients (tnode) and number of facilities (tfac), i.e. iteration=f(tnode,tfac). Iteration numbers of best objective value found in the random searches of experimentation problems are regressed on tnode and tfac with many functional forms. Eventually, the form, iteration = e^a.$tnode^b$.$tfac^c$ with a=6.75, b=0.5 and c=0.75, produces better results. Experimentation values of the iteration are determined as the values come from the regression results r (i.e. r = $e^{6.75}$.$tnode^{0.5}$.$tfac^{0.75}$), r/2 and 2.r.

Experimentation values for t_s and t_f: From the random searches, two values are calculated for each problem instance. One is $delta_{max}$ which is the maximum delta (fstar-fcur) occurs during the random search of that problem, the other is $delta_{ave}$ which is the average of positive delta values in random search of that instance. Starting temperature should be hot enough to allow free exchange of neighbor solutions and to make the final solution independent of the starting solution. In other words, p_a should be high enough (like 0.99 or 0.95) to accept the move even in the case of worst objective deterioration. Hence $t_s(p_a)= [-delta_{max} / \ln(p_a)]$ for p_a values of 0.99 and 0.95 are taken as experimentation values. Through the end of the algorithm, temperature should be low enough to fully exploit neighborhood of the latter solutions. Therefore, $t_f(p_a)=[-delta_{ave}/\ln(p_a)]$ for p_a values of 0.01 and 0.05 are taken as control parameters.

Experimentation for move types: It can be expected that when the facility number increases the requirement for complex move increases. Therefore, the problem instances are divided into two groups; the small ones (tfac≤10) and bigger ones (tfac > 10). Factorial design experiments are done separately for two groups. Simple move, complex move after (1/3.tfac) and after (1/4.tfac) iterations are control values.

Results of the experimentation: Full factorial design of MINITAB®13 with 4 factors is used for experimentation. For small problems, t_f and iteration are observed as significant factors. From the response graph of the experiment t_f (0.01) and iteration=r are selected. For insignificant factors, t_s=0.95 and simple move are chosen. For bigger problems, move type, t_f and iteration are observed as significant factors. From the response graph of the experiment, complex move after (1/4.tfac) iterations, t_f (0.01) and iteration=r are selected, and t_s is assigned to t_s (0.95). (Details and graphs can be seen in congress CD).

3. Computational Results and Comparisons of the Algorithm

After setting the parameters of the algorithm, b parameter of the cooling schedule is determined as **b** = $[(t_s - t_f) / (\text{iteration}. t_s . t_f)]$ for each problem instance. 50 replications for each of 20 problems of Beasley (1990) are done on Pentium III 500 MHz with 256 MB RAM. The results of these runs are summarized in shaded columns of Table 1. In this table, best known value (for first 12 problems it is optimum, for others it is best value found so far) and objective values in best replications of other heuristics is taken from Mladenovic et al. (2003). Variable neighborhood search (VNS), tabu search 1 (TS1), tabu search 2 (TS2) and Simulated Annealing of this paper (SA) are metaheuristics; others are local search heuristics (detailed description of heuristics can be seen in Mladenovic et al. (2003)). As can be seen in Table 1, SA is as capable of finding good deviations from best known as other metaheuristics in six problem instances (1, 2, 6, 7, 11, 16). Moreover, in two problem instances (18, 19) it does not have the worst deviation. SA dominates all the local search heuristics except multi interchange (MI) and multi interchange plus alternate (MA+I) in terms of objective value in best replication. MI and MA+I produce better results in problem instances 3, 4, 5, 10 and 12. However, on average SA (15.3 % deviation) produces better results than MI (18.2 % deviation). Since the other heuristics are solved on another machine, time comparison cannot be provided here. However, from Mladenovic et al. (2003) it is observed that SA average time (16.19 s.) is not worse than other metaheuristic (TS2 (shortest) has average time 16.78 s.) although they are solved on a more sophisticated machine (Sun Sparc Station 10). (Details can be seen in congress CD).

Table 1. Comparison of heuristics

			Objective value in best replication															% Deviation of best value from best known					Ave. Time (sec.)
Pr	# of node	# of fac.	Best Known	B-S	I	Gr	Gr +I	GrP	GrP +I	A	A+I	MI	MA	MA +I	VNS	TS1	TS2	SA	MI	VNS	TS1	TS2	SA	SA
1	100	5	127	184	133	166	133	148	133	148	148	127	127	127	127	127	127	127	0.0	0.0	0.0	0.0	0.0	0.73
2	100	10	98	142	102	155	109	131	119	112	112	98	98	98	98	98	98	98	0.0	0.0	0.0	0.0	0.0	1.12
3	100	10	93	137	108	191	102	168	102	131	99	93	95	93	93	93	93	95	0.0	0.0	0.0	0.0	2.2	1.11
4	100	20	74	109	135	157	111	127	111	111	111	74	79	74	74	74	74	79	0.0	0.0	0.0	0.0	6.8	3.72
5	100	33	48	76	74	143	90	106	70	87	87	48	59	48	48	48	48	51	0.0	0.0	0.0	0.0	6.3	5.35
6	200	5	84	113	93	115	100	99	94	100	92	84	84	84	84	84	84	84	0.0	0.0	0.0	0.0	0.0	2.38
7	200	10	64	87	77	97	84	85	84	77	77	64	66	64	64	64	64	64	0.0	0.0	0.0	0.0	0.0	3.54
8	200	20	55	76	92	110	84	94	94	86	86	58	67	57	55	55	55	58	5.5	0.0	0.0	0.0	5.5	7.92
9	200	40	37	59	57	98	68	76	73	71	71	46	56	42	37	37	37	42	24.3	0.0	0.0	0.0	13.5	11.65
10	200	67	20	33	53	70	70	70	70	77	42	30	34	29	20	20	20	33	50.0	0.0	0.0	0.0	65.0	24.78
11	300	5	59	71	68	71	66	98	68	64	64	59	59	59	59	59	59	59	0.0	0.0	0.0	0.0	0.0	2.26
12	300	10	51	78	62	83	69	77	77	77	59	51	72	51	51	51	51	52	0.0	0.0	0.0	0.0	2.0	3.59
13	300	30	36	58	54	72	59	58	55	59	54	41	41	39	36	36	36	39	13.9	0.0	0.0	0.0	8.3	10.81
14	300	60	26	43	60	76	76	60	60	76	76	36	40	35	26	26	26	29	38.5	0.0	0.0	0.0	11.5	31.35
15	300	100	18	30	40	59	51	49	49	44	44	29	35	27	18	25	18	27	61.1	0.0	38.9	0.0	50.0	48.6
16	400	5	47	55	59	61	47	54	47	52	48	47	47	47	47	47	47	47	0.0	0.0	0.0	0.0	0.0	2.11
17	400	10	39	55	53	53	50	51	51	49	43	40	40	39	39	39	39	40	2.6	0.0	0.0	0.0	2.6	3.97
18	400	40	28	44	50	62	50	61	50	50	50	38	40	34	28	28	35	35	35.7	0.0	0.0	25.0	25.0	19.95
19	400	80	19	31	36	41	41	38	38	36	36	28	28	26	19	23	19	23	47.4	0.0	21.1	0.0	21.1	49.17
20	400	133	14	22	37	49	49	34	32	44	44	26	29	22	14	22	14	26	85.7	0.0	57.1	0.0	85.7	89.67
																	Average:		18.2	0.0	5.9	1.3	15.3	16.19

Main shortcoming of the SA algorithm is that it cannot produce better results when the facility number is high. Making more iterations for problem instances with facility numbers higher than 10 or applying more complex move like interchange (I) move of Mladenovic et al. (2003) can solve this problem.

4. Extension of the Algorithm for the 'Capacitated' Case and Conclusion

In the capacitated case of vertex p-center problem, main problem is the capacity constraint. When the algorithm is constructed as not to violate this constraint, it is seen that feasible move number from current solution decreases substantially, which causes cycles in the moves (same solutions in many steps). When infeasible moves are allowed, repaired infeasible moves cannot produce better results than even the most basic heuristics. Therefore, more complex move structures, which can foresee the future of the algorithm, will be used for the capacitated case. Namely, customer improvement graphs of Pallottino et al. (2003) will be used for this complex moves. However, the application is beyond the scope of this article and will be subject of another article.

In this paper, p-Center problem was solved with Simulated Annealing algorithm. It produced better objective values than almost all local search heuristics, and produced objective values as good as other metaheuristics in some problem instances. Finally, the extension to capacitated case was discussed.

References

Beasley, J.E. *OR-Library: distributing test problems by electronic mail*, Journal of the Operational Research Society 41 (11), 1069 - 1072, 1990 and http://mscmga.ms.ic.ac.uk/jeb/orlib/pmedinfo.html.

Laundy,M. and Mees, A., *Cooling Schedule for Optimal Annealing*, MOR,13, 311-329, 1986.

Mladenovic N., Labbe M., Hansen P., *Solving the p-Center problem with Tabu Search and Variable Neighborhood Search*, NETWORKS, 42 (1): 48-64, AUG 2003.

Pallottino S, Scapparra M.P and Scutella M.G, *Large Scale Local Heuristics for the Capacitated Vertex p-Center Problem*, TR 02- 16, Dipartimento di Informatica, University of Pisa, 2003.

Reeves,R., *Modern Heuristic Techniques for Combinatorial Problems*, McGraw Hill, Berkshire, 1995.