European Journal of Operational Research 21 (1985) 270–273
North-Holland

Short Communication

A note on solving large p-median problems

J.E. BEASLEY
Department of Management Science, Imperial College, London SW7 2BX, England

Abstract: In a previous paper we presented a tree search algorithm for the p-median problem, the problem of locating p facilities (medians) on a network, which was based upon Lagrangean relaxation and subgradient optimisation. That algorithm solved (optimally) problems with an arbitrary number of medians and having up to 200 vertices.

In this note we show that it is possible to enhance that algorithm to solve (optimally) problems having up to 900 vertices using the Cray-1S computer.

Keywords: Location, networks, integer programming

1. Introduction

The p-median problem is the problem of locating p facilities (medians) on a network so as to minimise the sum of all the distances from each vertex to its nearest facility. In a previous paper [2] we presented two separate algorithms for this problem, both based upon Lagrangean relaxation and subgradient optimisation. The most successful of these two algorithms (denoted LR1 in [2]) solved p-median problems involving up to 200 vertices.

Recently Boffey and Karkazis [1] reported solving a p-median problem involving 206 vertices (with $p = 45$) and, as far as we are aware, this is the largest problem that has been reported as solved in the literature.

In this note we show that it is possible to solve p-median problems involving up to 900 vertices by a combination of:

(a) the vector processing capability of the Cray-1S 'super-computer'; and

(b) algorithmic enhancement to LR1.

We shall deal with each of these in turn and note here that to avoid repetition we shall assume throughout familiarity with the details relating to algorithm LR1 presented in [2].

Received October 1984; revised January 1985

2. Vector processing

The Cray-1S is an example of what is called a 'super-computer' in that, for certain calculations, it is much faster than conventional computers. For example, adding together two one-dimensional vectors, each of length m, would involve $O(m)$ operations on a conventional computer but would involve only (essentially) O(1) operations on the Cray-1S. This ability to substantially speed up certain vector calculations is known as the vector processing capability of the Cray-1S.

We found it comparatively easy to alter our FORTRAN code for algorithm LR1 so that it was able to take advantage of the vector processing capability of the Cray-1S For example, our implementation of the algorithm of Floyd [5] for calculating the shortest distances between all n vertices of a network requires $O(n^3)$ operations on a conventional computer, but only $O(n^2)$ operations on the Cray-1S.

3. Algorithmic enhancements

We implemented three algorithmic enhancements to LR1, relating to:

(1) the subgradient ascent at the initial tree node

0377-2217/85/$3.30

(2) the subgradient ascent at the tree search nodes
(3) restarting the entire algorithm.
We shall deal with each of these enhancements in turn.

3.1. Subgradient ascent—initial tree node

In [2] we followed the approach of Held, Wolfe and Crowder [6] in deciding, for the initial tree node, the value of π (equation (46) in [2]) to use at each subgradient iteration and the total number of subgradient iterations. Our subsequent computational experience has been that, for the p-median problem, this approach generates an excessive number of subgradient iterations and that a similar duality gap can be obtained with fewer subgradient iterations by following the approach given by Fisher [4]. Hence in the computational results reported later we set $\pi = 2$ initially and halved π if z_L^* (the best lower bound found—see equation (43) in [2]) failed to increase in thirty subgradient iterations.

3.2. Subgradient ascent—tree search nodes

In [2] we carried out thirty subgradient iterations at each tree search node with a constant value of π. Subsequent computational experience has indicated that varying π in a systematic fashion leads (in total) to fewer tree search nodes. Accordingly, in the computational results given later we set $\pi = 1$ initially, halved π every five iterations and (as before) carried out thirty subgradient iterations at each tree search node.

3.3. Restarting

Crowder et al. [3] have reported that in solving large-scale zero–one linear programs they found it advantageous to abandon the tree search after a certain stage and to restart the problem. This enabled them to make use of reduction tests that could be implemented at the initial tree node but which were difficult to apply at the tree search nodes.

We used a similar strategy of restarting the problem in enhancing LR1. If, at any stage of the tree search, we find an improved feasible solution z_{UB} (which we cannot immediately show to be optimal because $z_{UB} \neq z_L^*$ (the maximum lower bound found at the initial tree node)) then we restart the problem from scratch with a new initial tree node. Obviously, we carry forward any reductions that have been made at the previous initial tree node and restart the problem using Lagrange multipliers equal to those corresponding to z_L^* at the previous initial tree node.

Note here that we might restart the problem a number of times (generating a sequence of improved feasible solutions).

4. Upper bound generation

In [2] we briefly mentioned that an initial upper bound z_{UB} (corresponding to a feasible solution) can be found by applying any heuristic for the problem (e.g. Teitz and Bart [7]). Whilst, for that paper, the generation of an initial upper bound was not a computationally significant task we found that, for the much larger problems considered in this note, the generation of an initial upper bound became a computationally significant task

To generate an initial upper bound we used the simple interchange heuristic given below.

(1) Let V be the entire set of vertices and randomly generate a set Q of vertices with $|Q| = p$.
(2) Choose vertices $i \in Q$, $j \in V - Q$ and if $Q - [i] + [j]$ is a better p-median set than Q (i.e. of lower cost) set $Q = Q - [i] + [j]$, else do not.
(3) Repeat (2) for all vertices $i \in Q$, $j \in V - Q$ until no further improvements can be made

Our implementation of the above heuristic required, for a network with n vertices, $O(p(2n - p))$ operations on the Cray-1S. As will be seen from the computational results given later the time to generate an initial upper bound becomes significant when p is large.

5. Computational results

The algorithmic enhancements given above were implemented in the existing FORTRAN code for LR1 together with changes to enable advantage to be taken of the vector processing capability of the Cray-1S (using the CFT compiler). We shall refer to this enhanced version of LR1 as LR1E.

Table 1 gives details of the comparison between LR1 and LR1E for the five largest problems solved in [2]. In that table we show, for each problem, the

Table 1
Comparison of the original and enhanced algorithms

Problem number (from [2])	Number of vertices	p	Number of subgradient iterations initial tree node		Duality gap (%) initial tree node		Number of tree nodes		Computation time in seconds CDC 7600 Cray-1S	
			LR1	LR1E	LR1	LR1E	LR1	LR E	LR1	LR1E
15	150	5	581	340	0.594	0.593	25	29	10.2	2 1
16		10	581	294	0.275	0 274	89	27	27.8	2.1
17		15	451	230	–	–	1	1	11.0	1 0
18		50	406	80	–	–	1	1	16.1	0.7
19	200	5	796	365	0.709	0.698	67	45	43.9	3 7

Table 2
Computational results

Problem number	Number of vertices	p	Number of interchange applications	Interchange time Cray-1S seconds	Number of subgradient iterations initial tree node	Duality gap (%) initial tree node	Number of restarts	Total number of tree nodes	Computation time Cray-1S seconds
1	100	5	3	0.1	76	–	–	1	0.2
2		10	3	0.5	316	0.117	–	5	0.8
3		10	3	0.3	341	0.236	–	15	1.3
4		20	3	0.6	228	0.001	–	16	2.1
5		33	3	1.2	189	0.001	–	16	2.2
6	200	5	3	0.5	367	0.521	–	29	3 0
7		10	3	1.1	389	–	–	1	2.0
8		20	3	2.5	69	–	–	1	1.0
9		40	3	6.1	271	0.001	1	55	13.1
10		67	3	7.1	81	–	–	1	1.4
11	300	5	3	1.9	530	0.050	–	7	3 0
12		10	3	7.6	474	0.125	–	43	5.2
13		30	3	6.3	41	–	–	1	1.2
14		60	3	20.2	392	0.028	–	33	17.4
15		100	3	22.4	297	0.001	–	118	77.7
16	400	5	3	5.0	383	0.860	–	47	11.0
17		10	3	16.7	380	0.435	–	71	12 6
18		40	3	16.1	172	–	–	1	4.4
19		80	3	36.9	363	0.001	–	97	103 2
20		133	2	44 4	446	0.002	4	582	558.1
21	500	5	3	4.4	25	–	–	1	3.4
22		10	3	18.1	492	0.419	–	41	17.2
23		50	3	40.9	91	–	–	1	5.0
24		100	2	43.3	210	0.001	1	158	368.9
25		167	1	49.3	454	0.021	–	150	227.3
26	600	5	3	3.2	427	0.648	1	44	20.1
27		10	3	10.2	628	0.077	–	23	16.8
28		60	2	53.5	553	0.001	–	62	116.1
29		120	1	54.8	399	0.132	2	278	(600.0)
30		200	1	75.3	326	0.999	–	204	(600.0)
31	700	5	3	5.8	419	0.597	–	39	14.5
32		10	3	11.9	661	0.071	–	17	19.3
33		70	2	58.9	474	0.011	1	162	450 3
34		140	1	67.6	394	0.605	1	192	(600.0)
35	800	5	3	7.4	388	0.952	–	43	28.2
36		10	3	16.2	558	1.027	–	1077	403.3
37		80	1	54.7	514	0.000	1	82	171 0
38	900	5	3	9.4	493	1.032	–	99	40.6
39		10	3	18.3	524	0.637	–	119	68.1
40		90	1	95.1	503	0.001	–	99	500.2

A time in brackets indicates that the algorithm did not finish in the time shown. In such cases the duality gap is calculated from the best feasible solution found.

number of subgradient iterations at the initial tree node, the duality gap at the initial tree node and the total number of tree nodes. Computation times are also shown (excluding the set-up time needed to generate the allocation cost matrix). Note here that the duality gap at the initial tree node is measured by [(optimal value − maximum lower bound at the initial tree node)/(optimal value)] × 100%. LR1E can be said to be an improvement over LR1 both:

(a) algorithmically—generating a lower duality gap at the initial tree node in substantially fewer subgradient iterations and generating (almost always) fewer nodes in the tree search; and

(b) computationally—in terms of the time required on the Cray-1S, especially since the original FORTRAN program for LR1 (before both algorithmic enhancement and enhancement to take advantage of the vector processing capability of the Cray-1S) required 43.4 Cray-1S seconds to solve problem 19 in Table 1 (almost exactly the same time as the CDC 7600 required).

We also solved a number of randomly generated problems with n, the number of vertices, taking values from 100 to 900 in steps of 100 with values of p equal to 5, 10, $n/10$, $n/5$ and $n/3$. For each distinct pair of values (n, p) we randomly generated a new network with $n^2/50$ edges. Each edge cost (allocation cost) was an integer uniformly generated from [1,100] and, as in [2], the entire cost matrix was subjected to Floyd's algorithm [5] to ensure that the cost matrix was triangular.

Table 2 gives the results for the problems solved. In that table we give, for each problem:

(a) the number of applications of the interchange heuristic to generate an initial upper bound (and the corresponding computation time),

(b) the number of subgradient iterations at the initial tree node,

(c) the duality gap at the initial tree node,

(d) the number of problem restarts,

(e) the total number of tree nodes (summed over all restarts),

(f) the total computation time for LR1E.

Again the computation times quoted exclude the set-up time needed to generate the allocation cost matrix using Floyd's algorithm [5].

The strategy behind deciding the number of applications of the interchange heuristic for any particular problem was to carry out as many applications as possible subject to a maximum of three applications and a time limit of approximately 60 Cray-1S seconds. The best feasible solution found was used as the initial upper bound for LR1E. The idea behind this strategy was to prevent too much time being spent in deciding an initial upper bound, whilst at the same time generating a reasonable upper bound.

Note here that we did not attempt to solve some of the larger problems with $p = n/5$ and $p = n/3$ since the results for smaller problems indicated that we would not be able to solve such problems within our (self-imposed) 600 second time limit.

6. Conclusions

In this paper we considered the p-median problem and enhanced an existing Lagrangean relaxation tree search algorithm for the problem. Computational results indicated that it is possible to solve optimally much larger p-median problems than have previously been reported in the literature by combining this enhanced algorithm with the Cray-1S computer.

References

[1] Boffey, T.B., and Karkazis, J., "p-medians and multimedians", *Journal of the Operational Research Society* 35 (1984) 57–64.

[2] Christofides, N., and Beasley, J.E., "A tree search algorithm for the p-median problem", *European Journal of Operational Research* 10(2) (1982) 196–204.

[3] Crowder, H., Johnson, E.L., and Padberg, M., "Solving large-scale zero-one linear programming problems", *Operations Research* 31 (1983) 803–834.

[4] Fisher, M.L., "The Lagrangean relaxation method for solving integer programming problems", *Management Science* 27 (1981) 1–18.

[5] Floyd, R.W., "Algorithm 97—shortest path", *Communications of the ACM* 5 (1962) 345.

[6] Held, M., Wolfe, P., and Crowder, H.P., "Validation of subgradient optimisation", *Mathematical Programming* 6 (1974) 62–88.

[7] Teitz, M.B., and Bart, P., "Heuristic methods for estimating the generalised vertex median of a weighted graph", *Operations Research* 16 (1968) 955–961.