Computationally Difficult Instances for the Uncapacitated Facility Location Problem

Yuri Kochetov* Dmitry Ivanenko*

*Sobolev Institute of Mathematics
Novosibirsk, 630090, Russia
{jkochet,ivanen}@math.nsc.ru

1 Introduction

Design and analysis of computationally difficult instances is one of the promising areas in combinatorial optimization. It was the subject of final discussion session at the fourth MIC-conference. The difficult benchmarks allow us to understand advantages and disadvantages of mathematical methods and incite new research directions. In this paper we present several new classes of benchmarks for the Uncapacitated Facility Location Problem [3]. The first class is polynomially solvable. Nevertheless, it has many strong local optima and large mutual distance for each pair of them. Two other classes have exponential number of strong local optima. Finally, three last classes have large duality gap and one of them is the most difficult for metaheuristics and the branch and bound method.

2 Problem statement and its properties

In the Uncapacitated Facility Location Problem ($UFLP$) we are given a matrix $(g_{ij}), i \in I, j \in J$ and a vector $f_i \geq 0, i \in I$. We need to find a nonempty subset $S \subseteq I$ such that minimizes the objective function

$$F(S) = \sum_{i \in S} f_i + \sum_{j \in J} \min_{i \in S} g_{ij}.$$

It is a well known NP-hard combinatorial problem.

The neighborhood $Flip$ for S is a collection of subsets $S' \subseteq I$ which can be produced by dropping or adding one element to S. Standard local search algorithm starts from initial solution S_0 and moves to a better neighboring solution until it terminates at a local optimum. The running time of the algorithm is determined by the number of iterations. The local search problem ($UFLP, Flip$) is to find a local optimum with respect to the neighborhood. This problem belongs to the class PLS [6]. Whether the local search problem is polynomially solvable or not still remains an open question. The first step in this direction is the following.

Proposition 1. The local search problem $(UFLP, Flip)$ is PLS-complete.

In other words, the local search problem $(UFLP, Flip)$ is the most hard in PLS. Hence, we may assume that standard local search algorithm takes exponential time for the problem in the worst case. Nevertheless, in average, we observe linear running time of the algorithm. It is true for $Flip + Swap$ neighborhood too. So, in order to create computationally difficult benchmarks we have to generate instances with the following properties:
— large number of local optima;
— large minimal distance for pairs of local optima;
— large area where local optima located and others.
In the following sections we present several classes of benchmarks for the $UFLP$ which are computationally hard for the famous metaheuristics.

3 Polynomially solvable instances

Let us consider a finite projective plane of order k [2]. It is a collection of $n = k^2 + k + 1$ points $x_1, \dots, x_n$ and lines $L_1, \dots, L_n$. An incidence matrix A is an $n \times n$ matrix defining the following: $a_{ij} = 1$ if $x_j \in L_i$ and $a_{ij} = 0$ otherwise. The incidence matrix A satisfying the following properties:

1. A has constant row sum $k + 1$;
2. A has constant column sum $k + 1$;
3. the inner product of any two district rows of A is 1;
4. the inner product of any two district columns of A is 1.

These matrices exist if k is a power of prime. A set of lines $B_j = \{L_i \mid x_j \in L_i\}$ is called a bundle for the point x_j. We define a class of instances for the $UFPL$. Put $I = J = \{1, \dots, n\}$ and

$$g_{ij} = \begin{cases} \xi_{ij}, & \text{if } a_{ij} = 1, \\ +\infty & \text{otherwise,} \end{cases} \qquad f_i = f > \sum_{i \in I} \sum_{j \in J} \xi_{ij}.$$

We denote the class by FPP_k.

Proposition 2. Optimal solution for FPP_k corresponds to a bundle of the plane.

Proposition 3. Optimal solution for FPP_k can be found in polynomial time.

Every bundle corresponds to a strong local optimum of the $UFLP$ with neighborhood $Flip + Swap$. Hamming distance for arbitrary pair of the strong local optima equals $2k$. Hence, the diameter of area where local optima is located is quite big. Moreover, there are no other local optima with distance less or equal k to the bundle. Class FPP_k is hard enough for the local search methods when k is sufficiently large.

4 Instances with exponential number of strong local optima

Let us consider two classes of instances where number of strong local optima grows exponentially as dimension increases. The first class uses the binary perfect codes with code distance 3. The second class is constructed with help a chess board.

4.1 Instances based on perfect codes

Let B^k be a set of words (or vectors) of length k over an alphabet $\{0, 1\}$. A binary code of length k is an arbitrary nonempty subset of B^k. Perfect binary code with distance 3 is a subset $C \subseteq B^k, |C| = 2^k/(k+1)$ such that Hamming distance $d(c_1, c_2) \geq 3$ for all $c_1, c_2 \in C, c_1 \neq c_2$. These codes exist for $k = 2^r - 1, r > 1$, integer.

Put $n = 2^k, I = J = \{1, \ldots, n\}$. Every element $i \in I$ corresponds to a vertex $x(i)$ of the binary hyper cube $\mathbf{Z}_2^k$. Therefore, we may define a distance $d_{ij} = d(x(i), x(j))$ for any two elements $i, j \in I$. Now we define

$$g_{ij} = \begin{cases} \xi_{ij}, & \text{if } d(x(i), x(j)) \leq 1, \\ +\infty & \text{otherwise}, \end{cases} \qquad f_i = f > \sum_{i \in I} \sum_{j \in J} \xi_{ij}.$$

Arbitrary perfect code C produces a partition of $\mathbf{Z}_2^k$ into $2^k/(k+1)$ disjointed spheres of radius 1 and corresponds to a strong local optimum for the $UFLP$. Number of perfect codes grows exponentially as k increases. The best lower bound [4] is

$$2^{2^{\frac{k+1}{2} - \log_2(k+1)}} \cdot 3^{2^{\frac{k-3}{4}}} \cdot 2^{2^{\frac{k+5}{4} - \log_2(k+1)}}.$$

The minimal distance between two perfect codes or strong local minima is at least $2^{(k+1)/2}$. We denote the class of benchmarks by BPC_k.

4.2 Instance based on a chess board

Let us glue boundaries of the $3k \times 3k$ chess board so that we get a torus. Put $r = 3k$. Each cell of the torus has 8 neighboring cells. For example, the cell $(1, 1)$ has the following neighbors: $(1, 2), (1, r), (2, 1), (2, 2), (2, r), (r, 1), (r, 2), (r, r)$. Define $n = 9k^2, I = J = \{1, \ldots, n\}$ and

$$g_{ij} = \begin{cases} \xi_{ij}, & \text{if the cells } i, j \text{ are neighbors}, \\ +\infty & \text{otherwise}, \end{cases} \qquad f_i = f > \sum_{i \in I} \sum_{j \in J} \xi_{ij}.$$

The torus is divided into k^2 squares by 9 cells in each of them. Every cover of the torus by k^2 squares corresponds to a strong local optimum for the $UFPL$. Total number of these strong local optima is $2 \cdot 3^{k+1} - 9$. The minimal distance between them is $2k$. We denote this class of benchmarks by CB_k.

5 Instances with large duality gap

Duality gap for described classes is quite small. Therefore, the branch and bound algorithm finds an optimal solution and proves the optimality quickly. It is interesting to design benchmarks which are computationally difficult for both metaheuristics and enumeration methods.

As in previous cases, let the $n \times n$ matrix (g_{ij}) has the following property: each row and column have the same number of non-infinite elements. We denote this number by l. The value l/n is called the density of the matrix. Now we present an algorithm to generate random matrices (g_{ij}) with the fixed density.

Random matrix generator (l, n)

1. $J \leftarrow \{1, \dots, n\}$
2. Column $[j] \leftarrow 0$ for all $j \in J$
3. $g[i,j] \leftarrow +\infty$ for all $i, j \in J$
4. **for** $i \leftarrow 1$ **to** n
5.   **do** $l_0 \leftarrow 0$
6.   **for** $j \leftarrow 1$ **to** n
7.   **do if** $n - i + 1 = l-$ Column $[j]$
8.   **then** $g[i,j] \leftarrow \xi[i,j]$
9. $l_0 \leftarrow l_0 + 1$
10.   Column $[j] \leftarrow$ Column $[j] + 1$
11. $J \leftarrow J \setminus \{j\}$
12.   select a subset $J' \subset J, |J'| = l - l_0$ at random and put $g[i,j] \leftarrow \xi[i,j]$ for $j \in J'$

The array Column $[j]$ keeps the number of *small* elements in j-th column of the generating matrix. Variable l_0 is used to count the columns where small elements must be located in i-th row. These columns are detected in advance (line 7) and removed from the set J (line 11). Note that we may get random matrices with exactly l small elements for each row only if we remove lines 6–11 from the algorithm. By transposing we get random matrices with this property for columns only. We define three classes of benchmarks:

Gap-A: each column of $g(ij)$ has exactly l small elements;
Gap-B: each row of $g(ij)$ has exactly l small elements;
Gap-C: each column and row of $g(ij)$ have exactly l small elements.

For these classes we save $I = J = \{1, \dots, n\}$ and $f_i = f > \sum_{i \in I} \sum_{j \in J} \xi_{ij}$. The instances have significant duality gap

$$\delta = \frac{F_{opt} - F_{LP}}{F_{opt}} \cdot 100\%,$$

where F_{LP} is an optimal solution for the linear programming relaxation [3]. For $l = 10, n = 100$ we observe that $\delta \in [21\%, 28\%]$. The branch and bound algorithm evaluates at least $0,5 \cdot 10^9$ nodes in branching tree for the most part of instances from the class Gap-C (see Table 1).

Benchmarks classes	Dimension n	Gap δ	Iterations B & B	The best iteration	Running time
BPC_7	128	0.1	374 264	371 646	00:00:15
CB_4	144	0.1	138 674	136 236	00:00:06
FPP_{11}	133	7.5	6 656 713	6 652 295	00:05:20
$Gap-A$	100	25.6	10 105 775	3 280 342	00:04:52
$Gap-B$	100	21.2	30 202 621	14 656 960	00:12:24
$Gap-C$	100	28.4	541 320 830	323 594 521	01:42:51
$Uniform$	100	4.7	9 834	2748	00:00:00
$Euclidean$	100	0.1	1 084	552	00:00:00

Table 1: Performance of the branch and bound algorithm in average

6 Computational experiments

To study the behaviour of metaheuristics we generate 30 random test instances for each class. The values of ξ_{ij} are taken from the set $\{0,1,2,3,4\}$ at random and we set $f = 3000$. Optimal solutions are found for all instances by branch and bound algorithm [1, 3]. These instances are available by address: http://www.math.nsc.ru/AP/benchmarks/english.html. Table 1 shows performance of the algorithm in average. Column *Running time* presents execution time on PC Pentium 1200 MHz, RAM 128 Mb. Column *The best iteration* shows iterations for which optimal solutions were discovered. For comparison we include two well known classes [5]:

Uniform: values g_{ij} are selected in interval $[0, 10^4]$ at random with uniform distribution and independently from each other.

Euclidean: values g_{ij} are Euclidean distances between points i and j. The points are selected in square 7000×7000 at random with uniform distribution and independently from each other.

For these classes $f = 3000$. The interval and size of the square are taken in such way that optimal solutions have the same cardinality as in previous classes. Table 1 shows that class *Gap-C* is the most difficult for the branch and bound algorithm. Classes *Euclidean* and *Uniform* are quite easy.

Table 2 shows the frequency of finding optimal solution by the following metaheuristics: Probabilistic Tabu Search (PTS), Genetic Algorithm (GA) and Greedy Randomizes Adaptive Search Procedure with Local Descent ($GRASP + LD$). Stopping criteria for the algorithms is the maximal number of steps by the neighborhood $Flip + Swap$. We use number 10^4 as the criteria. Genetic algorithm uses local search by the neighborhoods for each offspring during the evolution. Note that class $Gap-C$ is the most difficult again. Classes *Euclidean* and $Uniform$ are easy. Average number of steps to get optimal solution is less than 10^3 for these metaheuristics.

Benchmarks classes	Dimension n	PTS	GA	GRASP+LD
BPC_7	128	0.93	0.90	0.99
CB_4	144	0.99	0.88	0.68
FPP_{11}	133	0.67	0.46	0.99
Gap-A	100	0.85	0.76	0.87
Gap-B	100	0.59	0.44	0.49
Gap-C	100	0.53	0.32	0.42
Uniform	100	1.0	1.0	1.0
Euclidean	100	1.0	1.0	1.0

Table 2: Frequency of obtaining optimal solutions by metaheuristics

7 Conclusions

In this paper we present six new classes of random instances for the Uncapacitated Facility Location Problem. Class FPP_k is polynomially solvable and has many strong local optima with large mutual distance for each pair of them. Classes BPC_k, CB_k have exponential number of strong local optima. Finally, classes *Gap-A, Gap-B, Gap-C* have large duality gap. Class *Gap-C* is the most difficult for both metaheuristics and branch and bound method. For comparison, we consider two famous classes *Euclidean and Uniform.* Density for these classes is 1. They are quite easy. We believe that density is a crucial parameter for the Uncapacitated Facility Location Problem. In order to get difficult instances we need to use small values of the density. Classes BPC_k, CB_k, FPP_k have the small density but regular structure of the matrix (g_{ij}). The most hard instances belong to the class *Gap-C.* They have random structure of the matrix and the same small density for all columns and rows.

References

[1] V. L. Beresnev, An implicit enumeration algorithm for location and standardization type problems, *Upravlyaemye sistemy*, N 12, 1974, 24–34, (in Russian).

[2] M. Hall Jr., *Combinatorial Theory*, Blaisdell, Waltham, MA, 1967.

[3] D. Erlenkotter, A dual-based procedure for uncapacitated facility location, *Operations Research*, **26**, 1978, 992–1009.

[4] D. S. Krotov, Lower bounds for number of m-quasi groups of order 4 and number of perfect binary codes, *Discrete Analysis and Operations Research*, **7**(2), 2000, 47–53, (in Russian).

[5] D. A. Schilling, K. E. Rosing, C. S. ReVelle, Network distance characteristics that affect computational effort in p-median location problems, *European Journal of Operational Research*, **127**, 2000, 525–536.

[6] M. Yannakakis. Computational complexity. E.Aarts and J.K. Lenstra (eds.) *Local Search in Combinatorial Optimization*, Chichester: Wiley, 1997. 19–55.