Chapter 5

COMPUTATIONALLY DIFFICULT INSTANCES FOR THE UNCAPACITATED FACILITY LOCATION PROBLEM

Yuri Kochetov, Dmitry Ivanenko

Sobolev Institute of Mathematics Novosibirsk, 630090, Russia

Abstract: Design and analysis of computationally difficult instances is one of the promising areas in combinatorial optimization. In this paper we present several new classes of benchmarks for the Uncapacitated Facility Location Problem. The first class is polynomially solvable. It has many strong local optima and large mutual pair distances. 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.

Key words: facility location, benchmarks, PLS-complete problem, local search, branch and bound method.

1. PROBLEM STATEMENT AND ITS PROPERTIES

In the Uncapacitated Facility Location Problem (*UFLP*) we are given a finite set of clients J, a finite set of facilities I, fixed costs of opening facilities $f_i \geq 0$, $i \in I$, and matrix of transportation costs (g_{ij}), $i \in I$, $j \in J$.

We need to find a nonempty subset of facilities $S \subseteq I$ 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 combinatorial problem with wide range applications [2, 3, 9]. The p-median problem, set covering problem, minimization problem for

polynomials in Boolean variables are closely related to (*UFLP*) [7]. Notice that it is NP-hard in strong sense.

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 improvement 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.

Definition 1. [12] A local search problem Π is in class PLS (polynomial--time local search problems) if there are three polynomial time algorithms A_Π, B_Π, C_Π with the following properties:

1. Given a string $x \in \{0,1\}^*$, algorithm A_Π determines whether x is an instance of the problem Π, and in the case it produces a feasible solution.

2. Given an instance x and a string s, algorithm B_Π determines whether s is a feasible solution and if so, algorithm B_Π computes the value of objective function for the solution s.

3. Given an instance x and a feasible solution s, algorithm C_Π determines whether s is a local optimum, and if it is not, algorithm C_Π finds a neighboring solution with strictly better value of objective function.

It is easy to verify that the local search problem Π=(*UFLP*, *Flip*) belongs to the class *PLS*.

Definition 2. [12] Let Π_1 and Π_2 be two local search problems. A *PLS*-reduction from Π_1 to Π_2 consists of two polynomial–time computable functions h and g such that:

1. h maps instances x of Π_1 to instances $h(x)$ of Π_2;

2. g maps pairs (x, solution of $h(x)$) to solutions of x;

3. for all instances x of Π_1, if s is a local optimum for instance $h(x)$ of Π_2, then $g(x,s)$ is a local optimum for x.

If Π_1 *PLS*-reduces to Π_2 and if there is a polynomial–time algorithm for finding local optima for Π_2, then there is a polynomial–time algorithm for finding local optima for Π_1. Local search problem $\Pi \in PLS$ is called *PLS*-complete if every problem in *PLS* can be *PLS*-reduced to it [6]. A list of *PLS*-complete problems can be found in [6, 12].

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

Corollary 1. Standard local improvement algorithm takes exponential time in the worst case for the problem Π=(*UFLP*, *Flip*) regardless of the tie-breaking and pivoting rules used.

Proofs are in the Appendix.

These properties of the local search problem Π=(*UFLP*, *Flip*) deal with the worst case behavior only. In average case, our computational results yield linear running time of the local improvement algorithm. We observe the same effect for *Flip+Swap* neighborhood too. So, we have pessimistic predictions for the worst case and optimistic one for the average case. In order to create computationally difficult benchmarks we may try to generate instances with the following properties:

- large number of local optima;
- large area where local optima located;
- large minimal distance for pairs of local optima;
- large basins of attractions;
- long paths from starting points to local optima and others.

Below we present several classes of benchmarks for the *UFLP* which are computationally difficult for the famous metaheuristics.

2. POLYNOMIALLY SOLVABLE INSTANCES

Let us consider a finite projective plane of order k [4]. It is a collection of $n = k^2 + k + 1$ points $x_1,\ldots, x_n$ and lines $L_1,\ldots, L_n$. 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 $I = J = \{1,\ldots, n\}$ 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.

Corollary 2. 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 2*k*. 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.

3. 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.

3.1 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 use 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 $\aleph(k)$ grows exponentially as k increases. The best known lower bound [8] is

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

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.

3.2 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.

4. INSTANCES WITH LARGE DUALITY GAP

As we will see later, the 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. $J \leftarrow \{1,\ldots, n\}$
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. Now we introduce 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 $I = J = \{1,\ldots, n\}$ 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 [5].

For $l = 10$, $n = 100$ we observe that $\delta \in [21\%, 29\%]$. The branch and bound algorithm evaluates at least $0{,}5\cdot 10^9$ nodes in branching tree for the most of instances from the class Gap-C (see Table 1).

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

Benchmarks classes	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	6 635 295	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	2 748	00:00:00
Euclidean	100	0.1	1 084	552	00:00:00

5. COMPUTATIONAL EXPERIMENTS

To study the behavior 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, 5]. All 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 [11]:

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 a way that optimal solutions have the same cardinality as in previous classes. Table 1 comfirms that classes *Gap-A, Gap-B, Gap-C* have large duality gap and they are the most difficult for the branch and bound algorithm. The classes BPC_7, CB_4, *Euclidean* have small duality gap. Nevertheless, the classes BPC_7, CB_4 are more difficult than *Euclidean* class. This effect has simple explanation. Classes BPC_7, CB_4 have many strong local optima with small waste over the global optimum.

In order to understand the difference between classes from the point of view the local optima allocation, we produce the following computational experiment. For 9000 random starting points we apply standard local improvement algorithm with *Flip+Swap* neighborhood and get a set of local optima. Some of them are identical. Impressive difference between benchmark classes deals with the cardinality of the local optima set. Classes *Uniform* and *Euclidean* have pathological small cardinalities of local optima sets and as we will see later these classes are very easy for metaheuristics. In Table 2 the column *N* shows the cardinalities for typical instances in each class. The column *Diameter* yields lower bound for diameter of area where local optima are located. This value equals to maximal mutual Hamming distance over all pairs of local optima.

Figures 1–8 plot the costs of local optima against their distances from global optimum. For every local optimum we draw a sphere. The center of sphere has coordinates (*x*,*y*), where *x* is the distance, and *y* is the value of objective function. The radius of the sphere is a number of local optima which are located near the given local optimum. More exactly, the Hamming distance *d* less or equals to 10. In Table 2 columns *min*, *ave*, and *max* show the minimal, average, and maximal radiuses for the corresponding sets of local optima. The class FPP_{11} has extremely small maximal and average value of the radiuses. Hence, the basins of attraction for the local optima are quite large. So, we may predict that Tabu search and other metaheuristics face some difficultes to solve the instances [10]. Column *R** gives the radius of the sphere for the global optimum. Column R_{100} shows the minimal radius for 100 biggest spheres. This column indicates that the classes *Gap-A*, *Gap-B*, *Gap-C* have many large spheres. The average radiuses are quite large too. It seems that the classes are simple for metaheuristics. Table 3 disproves the prediction. Class *Gap-C* is the most difficult for metaheuristics and branch and bound method.

Table 2: Attributes of the local optima allocation

Benchmarks classes	N	Diameter	Radius			R_{100}	*R**
			min	Ave	max		
BPC_7	8868	55	1	3	357	24	52
CB_4	8009	50	1	13	178	78	53
FPP_{11}	8987	51	1	2	8	3	1
Gap – A	6022	36	1	53	291	199	7
Gap – B	8131	42	1	18	164	98	16
Gap – C	8465	41	1	14	229	134	21
Uniform	1018	33	1	31	101	61	1
Euclidean	40	21	11	13	18		10

Table 3 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 Improvement (*GRASP+LI*). 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 improvements for each offspring during the evolution. Classes *Euclidean* and *Uniform* are easy. Average number of steps to get optimal solution is less than 10^3 for these classes.

Figure 9 shows the average radius R_{ave} as a function of d. For every class the function has three typical intervals: flat, slanting, and flat again. The first interval corresponds to the distances d where the spheres overlap rarely. At the last interval the value of d is closed to diameter of area where local optima are located. The transition point from the second interval to the third one conforms to the diameter of area which covers the most part of local optima. For the class *Gap-C* this point approximately equals to 33. For the classes FPP_{11}, BPS_7, CB_4 this value is near 42. Hence, the area of local optima is bigger for the classes FPP_{11}, BPS_7, CB_4 than for the class *Gap-C*. The column *Diameter* in Table 2 gives us the same conclusion. Nevertheless, the class *Gap-C* is the most difficult. It has the same density but random structure of the matrix (g_{ij}). It looks like that this property is very important for generation difficult instances for (*UFLP*).

Table 3: Frequency of obtaining optimal solutions by metaheuristics

Benchmarks Classes	Dimension	PTS	GA	GRASP+LI
BPC_7	128	0.93	0.90	0.99
CB_4	144	0.99	0.88	9.68
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

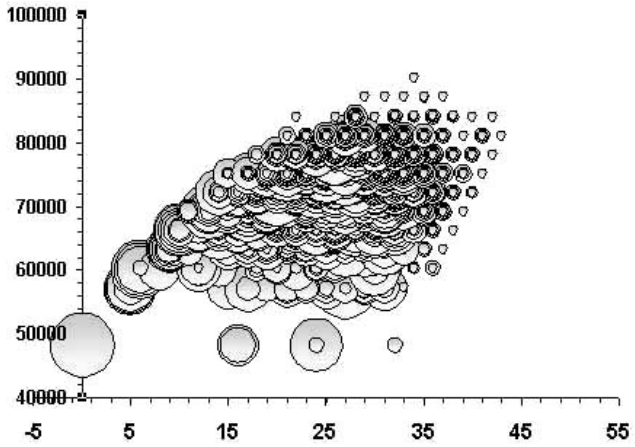

Figure 1. Analysis of local optima for the class BPC_7

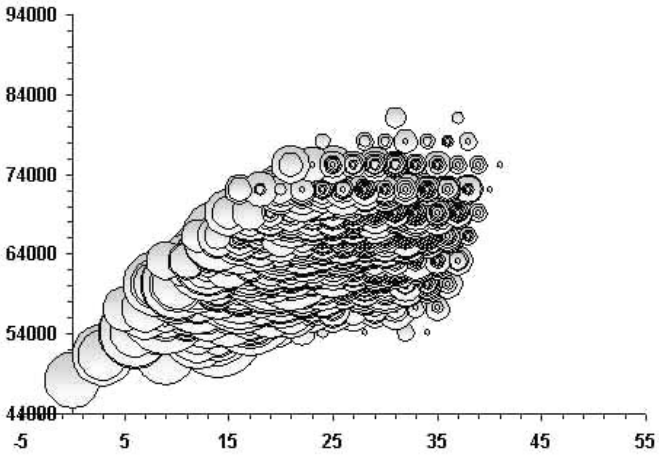

Figure 2. Analysis of local optima for the class CB_4

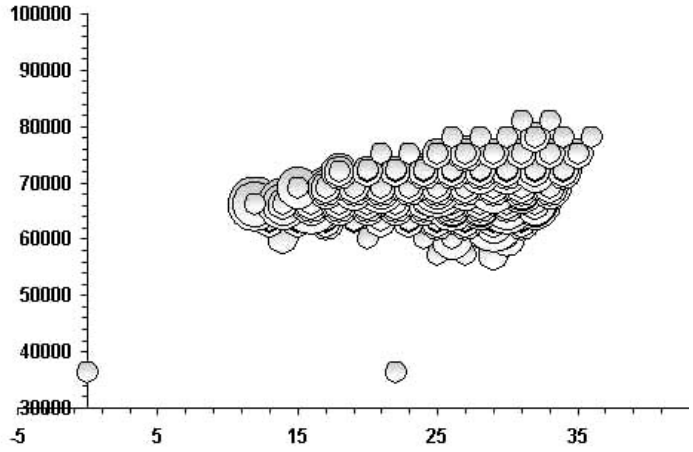

Figure 3. Analysis of local optima for the class FPP_{11}

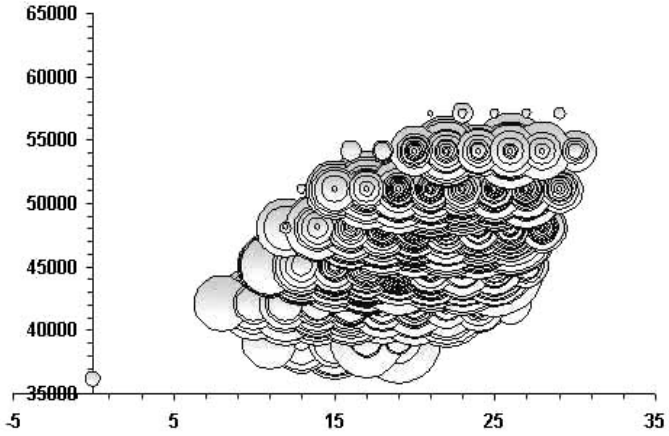

Figure 4. Analysis of local optima for the class *Gap-A*

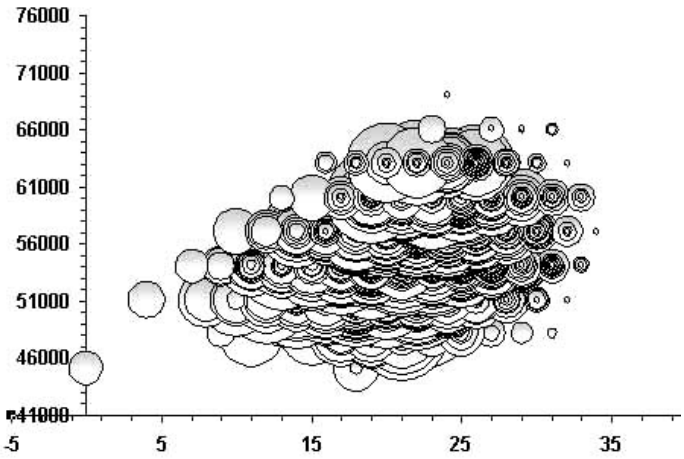

Figure 5. Analysis of local optima for the class *Gap-B*

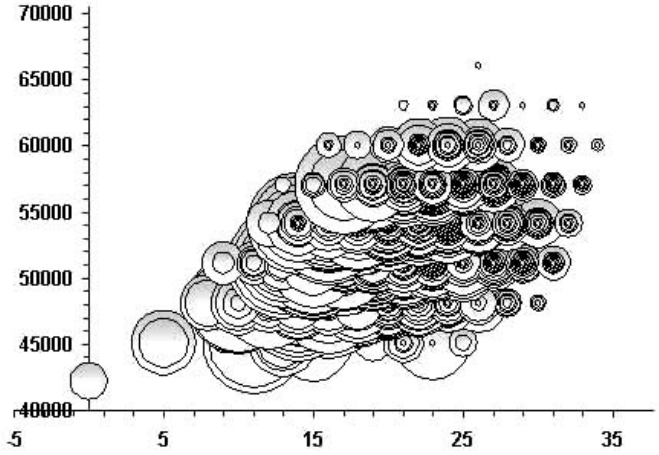

Figure 6. Analysis of local optima for the class *Gap-C*

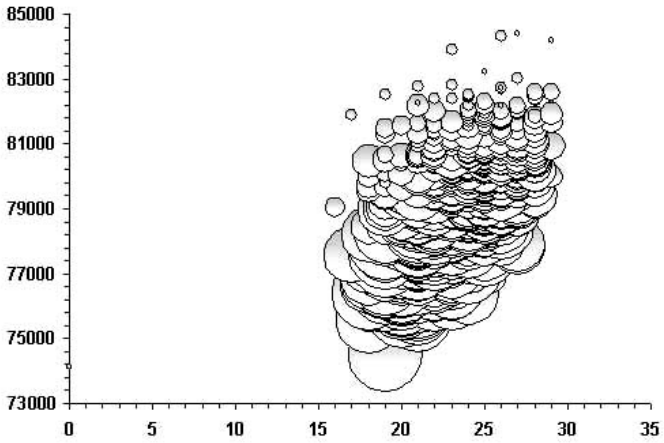

Figure 7. Analysis of local optima for the class *Uniform*

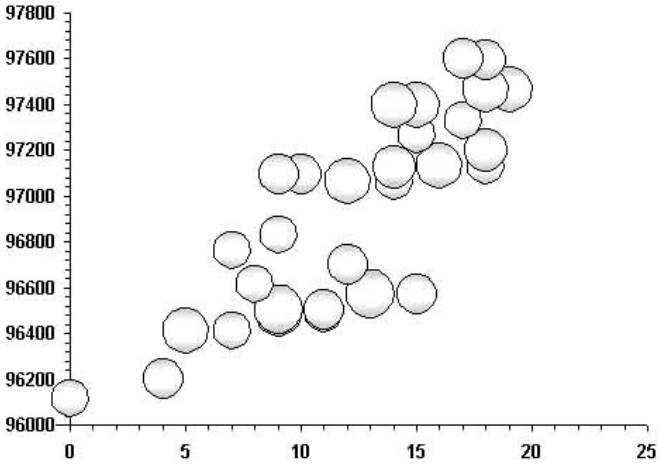

Figure 8. Analysis of local optima for the class *Euclidean*

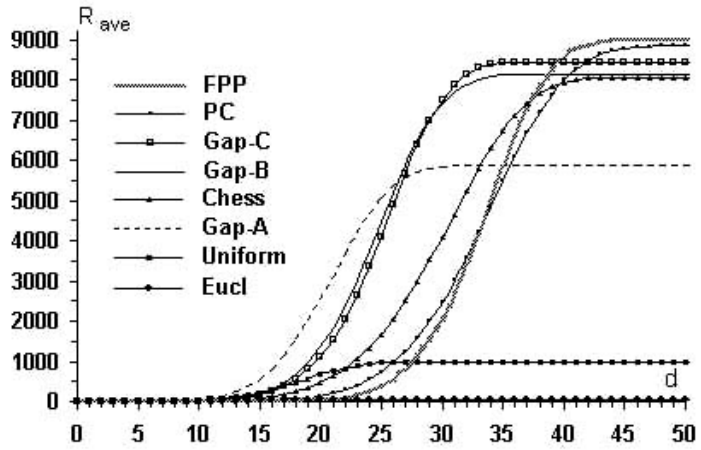

Figure 9. Average radius of spheres as a function of *d*

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 pair distances. 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.

ACKNOWLEDGEMENT

This research was supported by the Russian Foundation for Basic Research, grant No. 03-01-00455.

APPENDIX

Proof of the Proposition 1. Let us consider the *MAX-2SAT* problem. Denote by y_i, i=1,…, n the boolean variables. A literal is either variable or its negation. A clause is a disjunction of literals. In the *MAX-2SAT* problem we are given a set of clauses C_j, j=1,…, m. Each clause has a positive weight w_j and has at most two literals. We should find an assignment of variables in such a way that to maximize the total weight of satisfied clauses. The local search problem (*MAX*-2*SAT*, *Flip*) is *PLS*-complete [12]. Below we reduce this problem to (*UFLP*, *Flip*).

Let us present *MAX-2SAT* problem as a 0-1 linear program:

$$\max_{z_j, y_i \in \{0,1\}} \sum_{j=1}^{m} w_j z_j$$

s.t. $$\sum_{i \in C_j^+} y_i + \sum_{i \in C_j^-} (1 - y_i) \geq z_j \ , \ j = 1,..., m,$$

where the clause $C_j^+(C_j^-)$ contains all variables $y_i(\bar{y}_i)$ of the clause C_j. The variable z_j shows is the clause C_j satisfied or not. We have

$$z_j = 1 - \prod_{i \in C_j^+} (1 - y_i) \prod_{i \in C_j^-} y_i \ , \quad j = 1,..., m,$$

and *MAX-2SAT* problem can be written as unconstrained minimization problem *P*:

$$\min_{y_i \in \{0,1\}} \sum_{j=1}^{m} w_j \prod_{i \in C_j^+} (1 - y_i) \prod_{i \in C_j^-} y_i .$$

It can be rewritten in the following way:

$$\min_{y_i \in \{0,1\}} \left\{ \sum_{j \in J_1} w_j \prod_{i \in C_j'} y_i + \sum_{j \in J_2} w_j (1 - \prod_{i \in C_j'} y_i) \right\} + \text{const.}$$

with appropriate sets J_1, J_2 and clauses C_j'. Let us apply substitution

$$x_j = 1 - \prod_{i \in C'_j} y_i , \ \ j \in J_2$$

and add the following item

$$\sum_{j \in J_2} W \sum_{i \in C'_j} (1 - x_j)(1 - y_i)$$

with large constant $W > \sum_{j=1}^{m} w_j$ to the objective function. We get a problem P':

$$\min \left\{ \sum_{j \in J_1} w_j \prod_{i \in C_j} y_i + \sum_{j \in J_2} w_j x_j \sum_{j \in J_2} W \sum_{i \in C'_j} (1 - x_j)(1 - y_i) \right\}.$$

If (y_i) is an optimal solution of P and

$$x_j = 1 - \prod_{i \in C'_j} y_i , \ \ j \in J_2$$

then (x,y) is an optimal solution of P'. In order to verify this property it is sufficiently to note that

$$\sum_{j \in J_2} \sum_{i \in C'_j} (1 - x_j)(1 - y_i) = 0$$

if and only if

$$x_j = 1 - \prod_{i \in C'_j} y_i , \ \ j \in J_2 .$$

Moreover, if (y_i) is a local optimum for P with respect to *Flip* neighborhood, then (x,y) is a local optimum for P' too. Now we rewrite P' as *UFPL*. For this purpose we present P' as follows

$$\min \left\{ \sum_{i \in I} \overline{w}_i (1 - y_i) + \sum_{j \in J_2} \overline{w}_j (1 - x_j) + \sum_{j \in J_2} w_j \prod_{i \in C_j} y_i + \sum_{j \in J_2} \overline{w}_j x_j \sum_{i \in C'_j} y_i \right\}$$

for appropriate weights $\overline{w}$. It is minimization problem for polynomial in boolean variables with positive weights for nonlinear items. It is known [1] that this problem is equivalent to *UFPL*.

Proof of the Corollary 1. The statement is true for *(MAX-2SAT, Flip)* [12]. The reduction from *MAX-2SAT* to *UFPL* in the proof of Proposition 1 introduces new variables x_j, $j \in J_2$. Nevertheless, the equality

$$x_j = 1 - \prod_{i \in C'_j} y_i , \ \ j \in J_2$$

guarantees that the value of objective function does not change under this reduction. Hence, the standard local improvement algorithm produces the same steps for both local search problems regardless of the tie-breaking and pivoting rules used.

REFERENCES

1. V. Beresnev, E. Gimadi, and V. Dement'ev. *Extremal standardization problems*. Nauka, 1978, (in Russian).
2. M. Daskin. *Network and Discrete Location Problem: Models, Algorithms, and Applications*. John Wiley & Sons, Inc., 1995.
3. Z. Drezner (Ed.) *Facility Location: A survey of Applications and Methods.* Springer Series in Operations Research, Springer, 1995.
4. M. Hall Jr. *Combinatorial Theory*. Blaisdell. Waltham. MA, 1967.
5. D. Erlenkotter. A dual-based procedure for uncapacitated facility location, *Operations Research*, 26: 992–1009, 1978.
6. D. Johnson, C. Papadimitriou, and M. Yannakakis. How easy is local search? *Journal of Computer and System Sciences*, 37: 79–100. 1988.
7. J. Krarup and P. M. Pruzan. The simple plant location problem: survey and synthesis. *European Journal of Operational Research*, 12: 36–81, 1983.
8. D. Krotov. Lower bounds for number of *m*-quasi groups of order 4 and number of perfect binary codes. *Discrete Analysis and Operations Research*, 7: 47–53, 2000. (in Russian).
9. P. Mirchandani and R. Francis (Eds.) *Discrete Location Theory*. John Wilew & Sons, 1990.
10. M. Resende and R. Werneck. A hybrid multistart heuristic for the uncapacitated facility location problem, Manuscript, http://www.research.att.com/mgcr/doc/guflp.pdf.
11. D. Schilling, K. Rosing, and C. ReVelle. Network distance characteristics that affect computational effort in *p*-median location problems. *European Journal of Operational Research*, 127: 525–536, 2000.
12. M. Yannakakis. Computational complexity. E.Aarts and J.K. Lenstra (Eds.) *Local Search in Combinatorial Optimization*, pages 19–55, Chichester: Wiley, 1997.