SOLVING THE P-DISPERSION PROBLEM AS A SEQUENCE OF SET PACKING FEASIBILITY PROBLEMS

Liana Franco
Universidade Federal Fluminense
Rua Passo da Pátria, 156, São Domingos, Niterói - RJ
lianafranco@id.uff.br

Eduardo Uchoa
Universidade Federal Fluminense
Rua Passo da Pátria, 156, São Domingos, Niterói - RJ
uchoa@producao.uff.br

RESUMO

O problema de p-dispersão consiste em selecionar p entre n pontos de modo a maximizar distância mínima entre quaisquer dois pontos escolhidos. O problema tem muitas aplicações em localização de facilidades e em diversificação de amostras. Propomos um método em que uma sequencia de problemas de *set packing* de viabilidade é resolvida, em O(log n) iterações com o procedimento de busca binária. As experiências computacionais mostram que o algoritmo pode resolver muitas instâncias com tamanho de até n = 1.400. Este trabalho mostra que soluções exatas para o problema de p-dispersão podem ser obtidas de forma consistente para problemas moderadamente grandes, especialmente para o caso de distâncias geométricas ou geométricas ponderadas.

PALAVARAS CHAVE. Dispersão, Diversidade, Localização.

Área principal: Programação Matemática

ABSTRACT

The p-dispersion problem consists in selecting p out of n points, such that the minimum distance between any pair of chosen points is maximized. The problem has many applications in in facility location and in diversified sampling. We propose a method where a sequence of set packing feasibility problems are solved, in O(log n) iterations with a binary search procedure. Computational experiments show that the algorithm may solve many instances of size up to n=1,400 in quite reasonable times. Overall, this work shows that exact solutions for the p-dispersion problem may consistently be obtained for moderately large problems, especially for geometric and weighted geometric distances.

KEYWORDS. Dispersion, Diversity, Location.

Main area: Mathematical Programming

1. Introduction

The p-dispersion problem (also known as max-min diversity problem) can be described as follows. Given a set of *n* objects and a matrix of distances between each pair of points, the objective is to choose *p* points in order to maximize the smallest distance between two chosen objects.

This problem has many applications in facility location, when one aims to scatter the selected facilities, putting them as apart from one another as possible. A classic example happens in military defense, when scattering helps to avoid the destruction of multiple facilities in a single enemy attack. In telecommunications, when positioning cell phone antennas or when choosing frequencies, scattering is used to minimize interferences. When selecting the location of franchise shops, scattering minimizes mutual concurrence. Gomes *et al*. (2014) used the scattering principle in cartographic labelling to minimize the overlap of labels.

There are also applications where there are *n* objects described by multiple attributes and one desires to choose a sample of *p* objects as diversified as possible. In those cases, the distances are calculated using some metric over the attributes. For example, Erkut (Erkut, 1990) described a situation where different aspects of a product, such as price, quality, shape, appeal etc, are considered and the objective is having the maximum diversity in the portfolio of products offered to the market.

The p-dispersion problem was first proposed by Moon and Chaudhry (1984). Erkut and Neuman (1988) studied the problem and proposed variants of the p-dispersion problem, based on different dispersion metrics. The four variants found in the literature are: the **maxisum-sum** seeks to maximize the sum of all distances among the selected points; the **maximin-sum**, maximizes the smallest sum of the distances from each select point to all the other selected points; the **maxisum-min** which aims at maximizing the sum of the minimum distances from each selected point to its closest selected point and finally the **maximin-min** the most classical variation of the problem and the object of study of this paper. This last problem is sometimes addressed as max-min p-dispersion problem, as the max-min diversity problem or (as we doing) just as the p-dispersion problem

Erkut (1990) shows that the p-dispersion problem is NP-hard with the following reduction to the max clique problem. In order to verify if a graph $G=(V,E)$ with *n* vertices contains a clique of size *p*, one can construct a matrix of distances where d(i,j)=1 if (i,j) in *E*, d(i,j)=0 otherwise, and solve the resulting p-dispersion instance. Ghosh (1996) proved that the problem is strongly NP-hard, even if the distances obeys the triangle inequality, by reducing the original problem to the vertex cover problem.

The previous work is described in section 2, and then our set packing feasibility formulation is introduced in section 3. Computational experiments are exposed in section 4 and the conclusion remarks are made in section 5.

2. Previous methods

Due to the difficulty of the problem, most of the methods proposed to solve the p-dispersion are heuristics. Recent heuristics include Della Croce, Grosso, and Locatelli (2009), that use the relation (pointed by Erkut (1990)) of the p-dispersion with the max-clique problem; the GRASP with path relinking by Resende, Martí, Gallego, and Duarte (2010); the tabu search by Porumbel, Hao, and Glover (2011) ; and the Variable Neighborhood Search by Saboonchi, Hansen, and Perron (2015). All those articles include tests with a common group of instances.

2.1. Exact methods

This subsection presents an overview of the exact methods proposed so far. Kuby (1987) proposes a mixed integer formulation with a big M parameter. Using a facility location notation, Kuby's formulation is described below.

Maximize r

S.t. $\sum_{i=1}^{n} x_i = p$

$r \leq d_{ij}\left(1 + M(1 - x_i) + M\left(1 - x_j\right)\right)$, $\quad i,j \in N \mid i < j$

$x_i \in \{0,1\}$, $\quad i \in N$

Where:

r = the smallest separation distance between any pair of open facilities

$x_i = \begin{cases} 1 & \text{if a facility locates at node i} \\ 0 & \text{otherwise} \end{cases}$

n = number of potential facility sites

p = number of facilities to be opened

N = {1,…,n}

M = a sufficiently large number. D_{max}, the maximum distance between two facilities can be used.

d_{ij} = distance between node i and node j

Erkut, (1990) proposes two formulations for the p-dispersion problem, the first is similar to the one proposed by Kuby(1987) as shown below.

Maximize r

S.t. $r \leq M\,(2 - x_i - x_j) + d_{ij}$ $\quad i,j \in N \mid i < j$

$\sum_{i \in N} x_i = p$

$x_i \in \{0,1\}$ $\quad i \in N$

Where:

r = the minimum of the distances between the selected points.

Erkut (1990) also proposed another exact formulation, he defined the Boolean variables $X = \{\bar{x}_i : 1 \leq i \leq n\}$ and the Boolean coefficient matrix $A(r) = \{a_{ij}(r) : 1 \leq j < i \leq n\}$ where

$\bar{x}_i = \begin{cases} 1 & \text{if candidate point } i \text{ is not selected} \\ 0 & \text{otherwise} \end{cases}$

$a_{ij}(r) = \begin{cases} 1 & \text{if } d_{ij} < r \\ 0 & \text{otherwise} \end{cases}$

He formulated the problem as follows:

Maximize r

S.t. $\bar{x}_i + \bar{x}_j \geq a_{ij}(r)$, $\quad i,j \in N \mid i < j$

$\sum_{i \in N} \bar{x}_i = p$

$\bar{x}_i \in \{0,1\}$, $\quad i \in N$

Erkut (1990) proposes solving the model above by fixing r to different values. The related problem that arrises when r is fixed was named r-separation and he suggested two different models to solve it. The two integer programming models for the r-separation are presented below.

Minimize $\sum_{i \in N} \bar{x}_i$

S.t. $\bar{x}_i + \bar{x}_j \geq 1 \quad , \qquad i, j \in N \mid \text{such that } i < j \text{ and } d_{ij} < r$

$\bar{x}_i \in \{0,1\} \quad , \qquad i \in N$

Since the problem has $O(n^2)$ constraints, Erkut (1990) defines the index set $S_i = \{j: d_{ij} < r, j \neq i\}$ and adds the constraints $\bar{x}_i$ over S_i to get a single constraint, this model may be described as below.

Minimize $\sum_{i \in N} \bar{x}_i$

S.t. $|S_i|\bar{x}_i + \sum_{j \in S_i} \bar{x}_j \geq |S_i| \quad , \qquad i \in N$

$\bar{x}_i \in \{0,1\} \quad , \qquad i \in N$

Erkut (1990) used both models to solve the r-separation, the first may have fewer constraints on a smaller r, therefore the use of each model may be determined depending on the size of r. Two Branch-and-Bound procedures have also been proposed to accelerate the convergence to an optimal solution. His method solved instances with n up to 40 and p = 16 to optimality.

Ghosh (1996) proposed an exact algorithm by solving the vertex packing problem repeatedly, also in a binary search scheme for finding the optimal value of r. Ghosh's vertex packing 0-1 quadratic formulation is shown below.

Minimize $-\sum_{i \in N} x_i + M \sum_{(i,j) \in F} x_i x_j$

Where,

F= $\{(i,j) \mid d_{ij} < r \,;\, i < j \text{ and } i, j \in N\}$

Ghosh (1996) solved instances with the same size as Erkut (1990), the instances with n=40 and p=16, but in little less time.

Pisinger (2006) proposes the following 0-1 quadratic formulation:

Maximize r

S.t. $r x_i x_j \leq d_{ij} \qquad i \,,\, j \in N$

$\sum_{i \in N} x_i = p$

$x_i \in \{0,1\} \quad , \qquad i \in N$

$r \geq 0$

Afterward the p-dispersion problem is decomposed into a number of clique problems, leading to the dense subgraph problem formulation, shown below. Again, those clique problems

are solved O(log n) times through a binary search scheme to converge to an optimal solution for the p-dispersion problem.

Maximize $\sum_{i\in N}\sum_{j\in N} e_{ij}x_i x_j$

S.t. $\sum_{i\in N} x_i = p$

$x_i \in \{0,1\}$, $\forall\, i \in N$

Where

$$e_{ij} = \begin{cases} 1 & \text{if and only if (i, j)} \in \text{E, hence } d_{ij} \geq r \\ 0 & otherwise \end{cases}$$

The clique problem results in a dense subgraph formulation, to solve it, Pisinger (2006) relies on a search of fast upper bounds, based on Lagrangian relaxation, semidefinite programming and reformulations techniques. Afterwards, a branch-and-Bound is derived to compute at each branching node a reformulation-based upper bound.

3. Proposed method

In this section, we propose a linear integer programming formulation to solve the p-dispersion problem. The formulation starts by solving the set packing feasibility problem to identify whether, given a distance r, it is feasible to select p among the n available locations. The distance between each pair of selected facilities must be no smaller than r, the set packing feasibility problem formulation for the p-dispersion problem is shown below:

Maximize 0

S.t. $x_i + x_j \leq 1$, $i, j \in N$ such that $i < j$ and $d_{ij} < r$

$$\sum_{i\in N} x_i = p$$

$x_i \in \{0,1\}$ $i \in N$

At first, this verification method might seem little helpful, given that when all possible distances are taken into account, there would be up to $n(n-1)/2$ interactions, considering $i < j$. However, the solution of the set packing problem may be used to set upper and lower bounds to the p-dispersion problem. It is trivial to infer that by solving the set packing feasibility problem a lower bound could be set, should the problem be feasible, likewise an upper bound could be set, should the problem be infeasible.

Table 1

n	n (n-1)/2	$\log_2[\frac{n(n-1)}{2}]$
10	45	5.5
100	4,950	12.3
1,000	499,500	18.9
10,000	49,995,000	25.6
100,000	4,999,950,000	32.2
1,000,000	499,999,500,000	38.9

The convergence to an exact solution is accelerated by a binary search to select the distance to be tested from the possible distances after they have been sorted. In this study a quicksort method is used. By resorting to binary search to select the distances to be tested, the number of maximum interactions is expressively reduced to the roundup of $\log_2[\frac{n(n-1)}{2}]$. By testing the actual distances, the solution is not only exact, but as accurate as it may possibly be. Table 1 illustrates how modest is the increase in $\log_2[\frac{n(n-1)}{2}]$ compared to the increase in n and in $n(n-1)/2$.

Since the aim is to find the smallest distance between each pair of p selected locations, the initial upper bound may ignore the $p-1$ biggest distances, furthermore, since the goal is to maximize, the initial lower bound may also disregard the smallest $n-p-1$ distances. This may be applied to reduce the initial set of possible solutions, from $\{1 \leq s \leq n(n-1)/2\}$ to $\{n-p \leq s \leq n(n-1)/2-p\}$ reducing the set of solutions by $n-2$. The removal of occasional equivalent distances may help to reduce the set of solutions expressively depending on the time of instance.

Figure 1

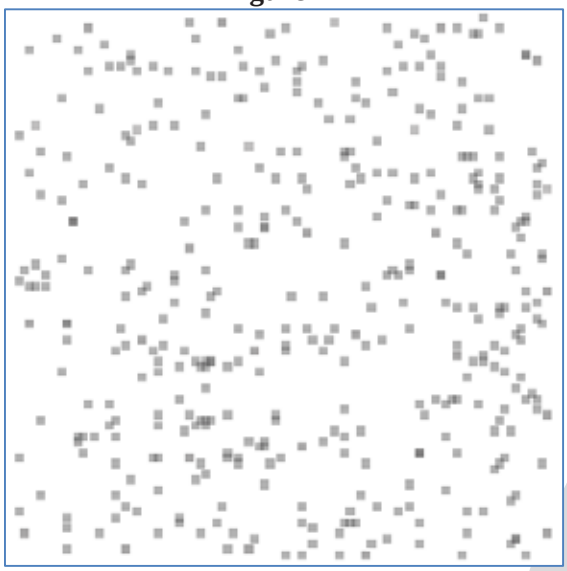


Figure 2

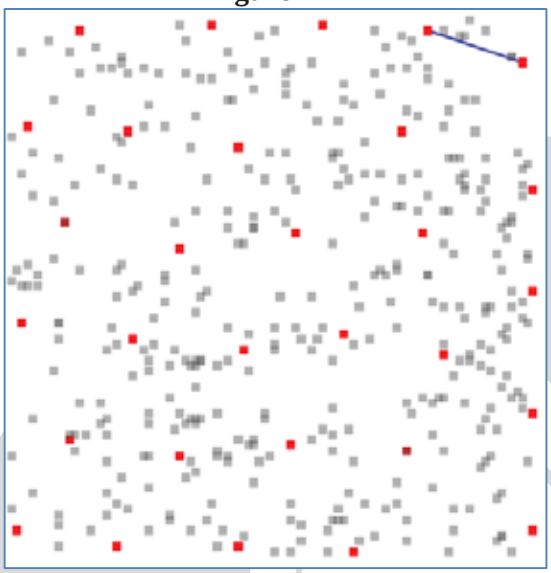
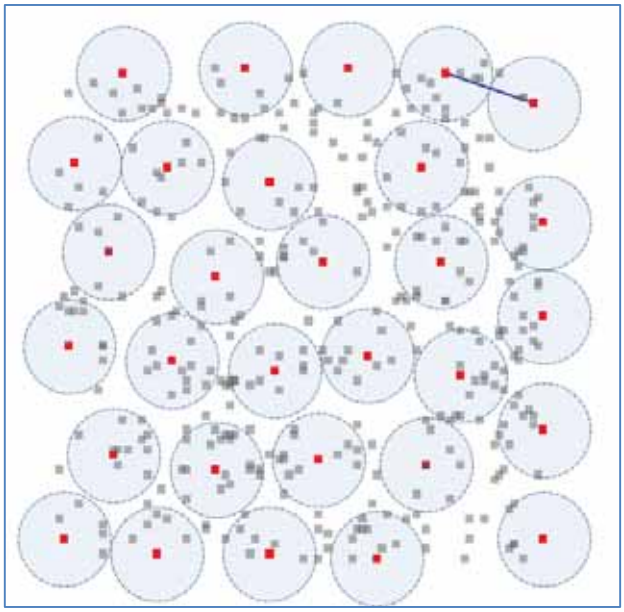


Figure 1 represents graphically a random generated instance of n = 400 , where the coordinates are known integers that vary from 0 to 100 in a bi-dimensional space. Each square represents a candidate location. Figure 2 show the optimal solution for the instance of Figure 1 where p = 30, the red squares represent the solution of the 30 selected locations. The blue line emphasizes the smallest distance, which has been maximized.

On Figure 3 the same solution is presented but each of the selected facilities is involved by a circle with diameter equal to the smallest distance. The circles of the 2 locations with the smallest distance between them, obviously tangency one another. As no circle surpasses another, we may graphically confirm that the given solution is in fact the smallest distance for this chosen subset of facilities.

Figure 3



4. Computational experiments

This section shows the computational experiments performed to test the efficiency of the proposed formulation. The instances were created using the same generator implemented by Pisinger (2006) and are described below.

GEO: The geometrical problems reflects a typical location problem in the Euclidean space, as presented in Erkut [3].The n facilities are randomly located in a 100×100 rectangle, and dij is the Euclidean distance between facilities i and j.

WGEO: The weighted geometrical problems are constructed to illustrate the case where the facilities have different weights (e.g. the radio transmitters located at some positions have different effect). As previously, the n facilities are randomly located in a 100 × 100 rectangle, and each facility is assigned a weight wi in the interval [5 ... 10]. The distance dij is then found as wiwj times the Euclidean distance between facility i and j.

EXP: Instances with exponential distribution of the distances were presented in Kincaid [9] (class 2). These instances should investigate how the algorithms perform when the triangle inequality was not satisfied. Each dij with i<j is randomly drawn from an exponential distribution with mean value 50.We set dij =dji to ensure symmetry.

RAN: Instances with random distances are generated with dij randomly distributed in [1 ... 100].

DSUB: Finally dense subgraph instances reflect the unweighted case. Hence dij is set to 100 or 0 with 50% probability each.

All the experiments were conducted on an Intel® Core™ i7 computer running at 3.6 GHz with 3 GB of RAM. The solver CPLEX 12.5.1 was called by the programming language Visual Basic for Applications (VBA) using the Dynamic Link Library UFFLP. A time limit of 20 minutes was assigned to each instance.

Table 2

n	GEO	WGEO	EXP	RAN	DSUB
10	0.03	0.04	0.02	0.04	0.00
20	0.05	0.05	0.03	0.04	0.00
30	0.03	0.05	0.06	0.05	0.01
40	0.04	0.05	0.05	0.07	0.01
50	0.04	0.06	0.09	0.09	0.01
60	0.05	0.06	0.14	0.13	0.01
70	0.04	0.06	0.11	0.26	0.03
80	0.04	0.06	0.08	0.22	0.02
90	0.06	0.08	0.49	0.43	0.04
100	0.05	0.11	0.39	0.59	0.05
150	0.08	0.13	0.06	7.11	0.52
200	0.10	0.16	49.59	105.91	0.12
250	0.14	0.33	240.10(2)	507.99(3)	0.13
300	0.24	0.23	148.89(1)	297.41(2)	4.91
350	0.15	0.32	483.45(4)	364.71(3)	0.37
400	0.12	0.28	360.26(3)	360.58(3)	0.39
450	0.28	0.62	352.94(2)	480.12(4)	1.00
500	0.81	0.21	360.10(3)	600.02(5)	120.84(1)
600	1.27	2.24	600.13(5)	360.21(3)	1.04
700	1.67	2.51	374.47(3)	241.63(2)	1.94
800	2.33	2.13	960.20(8)	480.34(4)	122.13(1)
900	108.13	2.27	482.83(4)	600.31(5)	123.21(1)
1000	2.13	8.12	602.91(4)	481.67(4)	36.41
1100	7.16	6.15	—	361.89(3)	123.62(1)
1200	6.90	8.17	—	604.10(5)	402.60(3)
1300	6.24	16.21	—	—	132.67(1)
1400	7.86	21.91	—	—	32.18

Solution time in seconds – average of 10 instances

Table 3

n	GEO	WGEO	EXP	RAN	DSUB
10	0.00	0.00	0.00	0.00	0.00
20	0.00	0.01	0.01	0.00	0.00
30	0.02	0.03	0.02	0.02	0.00
40	0.24	0.13	0.06	0.05	0.01
50	2.05	2.15	0.22	0.49	0.01
60	15.41	14.04	1.53	6.1	0.03
70	650.55	398.8	17.83	28.47	0.05
80	—	—	1151.4	314.15	0.08
100	—	—	—	—	0.17
200	—	—	—	—	2.73
300	—	—	—	—	22.34
400	—	—	—	—	50.38

Pisinger (2006) solution time in seconds - average of 10 instances

For each class and each given value of n, 10 instances were generated with a random value for p. A total of 1290 instances were studied, summing up to 270 instances generated per class. Table 2 presents the average time for solving the 10 randomly generated instances for a given *n* and a given class. The number in parenthesis indicates the number of instances that could not be solved within the time limit. When the time limit was reached, the maximum of 1200 s was considered, therefore, the actual average would be greater than the reported if the experiment is run to optimality. The dash indicates that the sizes that were not tested. Table 3 shows the results of the work of Pisinger (2006) for comparison; here the dash indicates that it took more than 15 hours to solve the 10 instances.

The proposed exact method solved to optimality all the geometrical and weighted geometrical problems generated, with n up to 1400, within 20 minute. For comparison, Pisinger (2006), the state of art in exact methods for the p-dispersion problem, needed an expressively larger average time to solve instances with *n* of at most 70, as shown in table 3.

On Table 2 we may note a deviation from the pattern of time for the resolution of the geometrical instances with n=900, one of the problems took about 17 minutes to be solved, while the other 9 had the average below 6 seconds. We observed that the selection of p was not the main reason for the great difference in time, an instance with p=80 took 17 minutes while another instance with little variation of p (p=87) took 5 seconds to be solved.

The instances of exponential distributed and random distances were all solved to optimality for $n \leq 200$, while Pisinger (2006) solved instances with $n \leq 80$. The dense subgraph instance type, showed a less stable solution pattern, as an illustration, the instances of n = 500 were all solved with an average of less than a second, except for one instance that could not be solved within the 20 minutes time limit. The results showed similar behavior for some instances with higher values of n, however, the instances with the highest value of n (n=1400) were all solved to optimality with an average time of 32 seconds.

Although Pisinger (2006) performed his experiments on a AMD 64-bit, 2.4 GHz, his time limit used was of 15 hours for 10 instances, while our time limit was of 20 minutes per instance. The differences in the results are far too large to be explained by the use of a more modern machine. Actually, the exponential nature of both solution methods indicate that a small speedup in the processor makes limited difference on the maximum size of the instances that can be solved. As may be observed in Table 3, the increment of 10 units in n, represents a significant increase in resolution time. Therefore it is clear that the set packing feasibility resolution on a binary search scheme leads to significantly better solution times.

5. Conclusions

An overview of the previously proposed exact methods have been presented and an exact method have been proposed for the p-dispersion problem, relying on the repeated solution of set packing feasibility problems. Although most of the previous authors of exact methods have also proposed binary search schemes, the simpler subproblem formulation led to clearly better results.

This work shows that exact solutions for the p-dispersion problem may consistently be obtained for moderately large problems, especially for geometric and weighted geometic distances.

References

Della Croce, F., Grosso, A., and Locatelli, M. (2009). A heuristic approach for the max–min diversity problem based on max-clique. Computers and Operations Research, 36(8), 2429–2433.

Erkut, E., and Neuman, S. (1988), "Comparison of four models for locating p mutually obnoxious facilities", Research Paper 88-6, Dept. of Finance and Management Sci., Univ. of Alberta, Edmonton, Alberta.

Erkut, E. (1990). The discrete p-dispersion problem. European Journal of Operational Research, 46(1), 48–60.

Gomes, S. P., Ribeiro G. M. and Lorena L. A. N.(2014) Duas Novas Abordagens para o Problema de Rotulação Cartográfica e Pontos: r-separação e p-dispersão. Anais do XLVI SBPO.

Ghosh, J. B. (1996). Computational aspects of the maximum diversity problem. Operations Research Letters, 19(4), 175–181

Kuby, M. J. (1987). Programming Models for Facility Dispersion: The p-Dispersion and Maxisum Dispersion Problems. *Geographical Analysis*, *19*(4), 315-329.

Moon, I. D., and Chaudhry, S. S. (1984). An Analysis of Network Location Problems with Distance Constraints. Management Science.

Pisinger, D. (2006). Upper bounds and exact algorithms for -dispersion problems. Computers and Operations Research, 33(5), 1380–1398.

Porumbel, D. C., Hao, J. K., and Glover, F. (2011). A simple and effective algorithm for the MaxMin diversity problem. Annals of Operations Research, 186(1), 275–293.

Resende, M. G. C., Martí, R., Gallego, M., and Duarte, A. (2010). GRASP and path relinking for the max–min diversity problem. Computers and Operations Research, 37(3), 498–508.

Saboonchi, B., Hansen, P., and Perron, S. (2015). MaxMinMin p-dispersion problem: A variable neighborhood search approach. Computers and Operations Research.