Semidefinite Bounds for the Maximum Diversity Problem

Roberto Aringhieri Maurizio Bruglieri Roberto Cordone

July 27, 2006

Abstract

The Maximum Diversity Problem consists in extracting a subset of given cardinality from a larger set in such a way that the sum of their pairwise distances is maximum. In this paper we propose a set of bounds for this problem based on semidefinite programming techniques. An extensive computational campaign shows the tightness of the bounds computed with respect to the best known results in literature and to bounds obtained by a linearized integer programming formulation.

1 Introduction

Given a set N of n objects, an integer number m and a matrix $\{d_{ij}\}$ providing the *diversity* between each pair of objects i and j belonging to N, the *Maximum Diversity Problem* (**MDP**) consists in finding a collection $M \subset N$ of m objects such that the sum of their pairwise distances is maximum. In the sequel we assume, w.l.o.g., that the diversity matrix is symmetric (i. e. $d_{ij} = d_{ji}$ for all $i, j \in N$) and that $d_{ii} = 0$ for all $i \in N$.

This problem has several practical applications. A common requirement in the identification of work teams, student groups and juries is, for instance, to gather individuals with strongly diversified characteristics: work teams take advantage from including the largest possible range of skills, student groups should encourage the exchange between people with different backgrounds, juries should represent the widest variety of points of view existing in a community. The distance between individuals i and j with respect to the relevant characteristics can be modeled by a suitable function d_{ij}, and most of the time the number of individuals in the team is fixed. Other interesting applications concern the allocation of available resources for preserving biological diversity [10], VLSI design, scheduling final exams, medical treatment, and data mining [12].

In this paper we survey the quadratic and linear integer formulations previously proposed in the literature for the MDP. Then, we introduce some

semidefinite formulations and relaxations, in order to compare the strength of the resulting bounds. We extend the comparison to the bounds provided by the continuous relaxation of a linearized integer programming formulation (both in its original form and strengthened by general-purpose cuts). To estimate the tightness of all these bounds, we compare them to the best known results on the two benchmark sets commonly adopted in the literature. These results mainly derive from a couple of Tabu Search approaches [7, 4], from a Scatter Search and a Variable Neighborhood Search algorithm [3] and from several Greedy Randomized Adaptive Search Procedures (*GRASP*) [8, 1, 16, 2, 15]. The semidefinite bounds outperform both in quality and in computational time the linear programming bounds, and the gaps with respect to the best known solution values are quite small.

The paper is organized as follows. Mathematical formulations and bounds deriving from the literature are presented in Section 2, new semidefinite formulations and bounds in Section 3 whilst the computational results are reported and discussed in Section 4. Concluding remarks are presented in Section 5.

2 Formulations from the literature

The MDP is strongly $\mathcal{NP}$-hard [13] and it was introduced by Glover [9]. In [13] Kuo *et al.* developed the following integer quadratic programming formulation:

$$\max \Big\{ \sum_{i,j \in N} d_{ij} x_i x_j \quad : \quad \sum_{i \in N} x_i = m \ , \quad x_i \in \{0,1\} \Big\} \tag{1}$$

where $x_i = 1$ if object $i \in M$, $x_i = 0$ otherwise. In the same article they also provided two equivalent linear integer models, based on the same variables x_i. In the former

$$z = \max \sum_{i,j \in N} d_{ij} y_{ij} \tag{2a}$$

$$\sum_{i \in N} x_i = m \tag{2b}$$

$$y_{ij} \leq x_i \qquad i,j \in N \tag{2c}$$

$$y_{ij} \leq x_j \qquad i,j \in N \tag{2d}$$

$$y_{ij} \geq x_i + x_j - 1 \qquad i,j \in N \tag{2e}$$

$$x_i \in \{0,1\} \qquad i \in N \tag{2f}$$

$$y_{ij} \in \{0,1\} \qquad i,j \in N \tag{2g}$$

where constraints (2c), (2d), (2e) and (2g) guarantee that the auxiliary variable y_{ij} equals 1 if both objects i and $j \in M$, 0 otherwise.

In the latter linearization, if $i \in M$ variable w_i expresses the total diversity $\sum_{j \in M} d_{ij}$ of object i with respect to the other objects in the solution; otherwise, $w_i = 0$.

$$z = \max \sum_{i \in N} w_i \tag{3a}$$

$$\sum_{i \in N} x_i = m \tag{3b}$$

$$w_i \leq U_i x_i \qquad i \in N \tag{3c}$$

$$w_i \leq \sum_{j \in N} d_{ij} x_j - (1 - x_i) L_i \qquad i \in N \tag{3d}$$

$$x_i \in \{0, 1\} \qquad i \in N \tag{3e}$$

where L_i and U_i are, respectively, a lower and an upper bound on the value of $\sum_{j \in N} d_{ij} x_j$. The quality of these bounds strongly affects the quality of the continuous relaxation of (3).

In [8] Ghosh turned the quadratic formulation (1) into an unconstrained one by penalizing the violation of the cardinality constraint. All these approaches are able to solve exactly instances up to 40 elements, much less than the typical dimension of real world problems, at least in some application fields.

3 Semidefinite formulations and bounds

The MDP is related to problems, such as Max Cut and Max Clique, which, though easy to linearize, are intrinsically quadratic. Semidefinite relaxations have proved to be very effective on these problems [11, 6]. It is therefore promising to test this approach on the MDP.

Analogously to the max-clique formulation presented by [6], the MDP can be formulated in terms of an unknown square matrix X of size n as follows

$$z = \max D \bullet X \tag{4a}$$

$$\operatorname{rank}(X) = 1 \tag{4b}$$

$$I \bullet X = m \tag{4c}$$

$$X \succeq 0 \tag{4d}$$

$$X_{ij} \in \{0, 1\} \tag{4e}$$

Here $\bullet$ denotes Frobenius product between matrices, $\operatorname{rank}(X)$ is the rank of matrix X, $X \succeq 0$ indicates that matrix X should be positive semidefinite and I is the identity matrix of size n. Constraints (4b) and (4d) guarantee that $X = xx^T$ for a suitable vector x, whilst constraints (4e) guarantee integrality and constraint (4c) is the *diagonal representation* of the cardinality requirement.

Other formulations can be obtained (see [14]) by replacing the diagonal representation respectively with

$(4c')$ $J \bullet X = m^2$ (*square representation*)

$(4c'')$ $(J - mI) \bullet X = 0$ (*extended square representation*)

$(4c''')$ $\sum_{j \in N} X_{ij} = mX_{ii}$ and $\sum_{j \in N} (X_{jj} - X_{ij}) = m(1 - X_{ii})$ for all $i \in N$

where J is the all-one matrix of size n. The semidefinite bounds are obtained by relaxing the non-convex rank constraint (4b) and the integrality constraint (4e).

Notice that in our formulations the integrality constraint is explicitly imposed by (4e). On the contrary, the standard way to turn a linear formulation into a semidefinite one [14] refers to a different unknown matrix X of size $n + 1$, suitably constrained to guarantee that

$$X = \begin{bmatrix} 1 & x^T \\ x & xx^T \end{bmatrix}$$

In this framework, the integrality constraint is expressed as

$$X_{0i} = X_{ii} \qquad i = 1, \ldots, n + 1$$

and the four representations of the cardinality constraint listed above must be trivially modified to take into account the larger size of X. The resulting four semidefinite bounds, obtained relaxing only the rank constraint (4b), are ordered by non decreasing tightness [14]. We have experimented with the standard technique, but we have rejected it after finding out that the four resulting bounds are very close to the corresponding bounds obtained with our approach, while requiring a longer computational time. This is due to the larger number of variables involved.

4 Computational Results

We computed all the semidefinite bounds described above with CSDP 5.0, a software package for solving semidefinite programming problems (see [5]). Some of the relaxations exhibited numerical convergence problems. However, these problems were overcome by keeping in all tested formulations the diagonal representation of the cardinality constraint (4c). In other words, instead of replacing constraint (4c) with one of its alternative forms $(4c')$, $(4c'')$ and $(4c''')$, we simply add one of them to the basic formulation.

In our experimental campaign, we consider the following two bounds. The first one, denoted as CSDP1, is given by formulation (4) plus the extended square representation $(4c'')$. The second, denoted as CSDP2, is given

by formulation (4) plus constraints ($4c'''$). We neglect the two possible remaining bounds: the only formulation (4) yields an extremely weak bound; the formulation (4) plus the square representation ($4c'$) is trivially equivalent to the CSDP1 and it takes a slightly longer computational time.

Then we also compute the bounds provided by the continuous relaxation of the linearized integer programming formulation (2). To achieve a remarkable improvement in the bound, we add the constraints

$$\sum_{j \in N} y_{ij} = m x_i \qquad i \in N \tag{5}$$

which are a linearized version of ($4c'''$). We solve the resulting formulation with the state-of-the-art commercial MIP solver CPLEX 8.1 and also try to further strengthen it by applying all general-purpose cuts available. We neglect formulation (3), because it proved weaker than formulation (2) strengthened by constraints (5).

To estimate the tightness of all these bounds, we compare them to the best known results on the two benchmark sets commonly adopted in the literature. Benchmark B_1, proposed in [1], consists of 40 instances with n ranging from 50 to 250 and m from $0.2n$ to $0.4n$; benchmark B_2, proposed in [16], consists of 20 instances with n ranging from 100 to 500 and m from $0.1n$ to $0.4n$. These instances are also available at `http://www.dti.unimi.it/~aringhieri`. The best known results on these instances mainly derive from a couple of Tabu Search approaches [7, 4], from a Scatter Search and a Variable Neighborhood Search algorithm [3] and from several Greedy Randomized Adaptive Search Procedures (*GRASP*) [8, 1, 16, 2, 15].

The experimental campaign has been performed on a Pentium D 3.2Ghz machine with 2GB of main memory running under Linux operating system. The codes have been compiled with GCC 4.0.2 compiler.

Tables 1 and 2 respectively report the results on benchmarks B_1 and B_2. The first three columns of Table 1 and the first two of Table 2 describe the instances, providing the total number of elements n and the number of elements m to be included in the solution. Table 1 also reports in the first column the class of instances (A, B, C or D) with respect to the random generation of the diversity matrix. The following column reports the best known result in the literature. The following two columns provide the results for the semidefinite bounds CSDP1 and CSDP2. Considering the formulation (2) plus constraint (5), the column labeled as CPLEX indicates the solution value obtained by solving its continuous relaxation whilst the column labeled as CPLEX+all indicates the solution values obtained by imposing all available general-purpose cuts to the solver and limiting the computation to the root of the branching tree. For each bound, column *% gap* provides the percent gap with respect to the best known result and column *CPU* the computational time in seconds required to compute it. Note

that the computing time for CPLEX has been limited to 1 day. The gaps for the semidefinite bounds are bolded when they are both better and faster to compute than the corresponding bounds obtained from the formulation (2).

As for Table 1, the average gap obtained by CPLEX exceeds 20%. Adding all available general-purpose cuts seems to have almost no effect on the quality of the bound, apart from few instances in class C. The semidefinite average gaps are 12.89% and 2.45%. The average computational time required by CPLEX is from 3 to 7 times higher than the time required by the stronger semidefinite bound, which is 40 times higher than the time required by the weaker semidefinite bound. The quality of both semidefinite bounds improves clearly when $m \approx n/2$, and this is particularly true for the tighter bound. The computational time, however, does not depend on m. A similar, but weaker, dependence of the gap on m holds for CPLEX, whose computational time increases when the final gap improves.

Table 2 confirms most of the previous remarks. They seem to be harder, as the average gap achieved by CPLEX rises to over 47%, and the average gap achieved by the two semidefinite bounds rises, respectively, to 11.10% and 7.67%. The computational time required by CPLEX with no additional cuts is 10 times larger than the one required by the stronger semidefinite bound. The generation of all general-purpose cuts makes the computation longer than one day for the instances with $n \geq 400$. On the smaller instances, the computation is on average 50 times slower. The computational time for the stronger semidefinite bound is, on its turn, 80 times longer than the one required by the weaker semidefinite bound. Once again, the quality of the semidefinite gap improves sharply when $m \approx n/2$, in particular for the tighter bound, with similar computational times. However, the influence of m on the linearized bound seems opposite to the one reported by Table 1. Here, the computational times seem to get larger when m increases, even though the gap is larger. Both semidefinite bounds outperform CPLEX: in particular, CPLEX could not find a feasible solution in one day of computation for the instance with $n = 500$ and $m = 200$. The only exception is the instance with $n = 100$ and $m = 10$ where CSDP1 and CSDP2 obtain worse gaps than CPLEX (35% and 27% versus 21%).

For the instance with $n = 500$ and $m = 150$, we reported a different best known result with respect to the one published in [16]: this value, equal to 58605, is in fact larger than the bound computed by CSDP2. This remark confirms that the best known value is 56572, as published in the most recent papers [4, 15, 7].

5 Conclusions

In this paper we have introduced a sequence of semidefinite bounds based on quite standard formulation techniques [14] for the Maximum Diversity

Instance			Best	CSDP 1		CSDP 2		CPLEX		CPLEX + all	
Type	n	m	**known**	**% gap**	**CPU**	**% gap**	**CPU**	**% gap**	**CPU**	**% gap**	**CPU**
	50	10	491.9	**8.33%**	0.06	**0.30%**	0.54	9.01%	0.13	8.96%	0.23
	50	20	1931.5	**3.14%**	0.07	**0.03%**	0.51	6.11%	0.33	4.66%	0.68
	100	20	2007.1	**6.93%**	0.38	**0.07%**	5.77	11.50%	3.46	11.50%	8.11
	100	40	7730.0	**3.15%**	0.37	**0.01%**	5.82	7.88%	8.00	7.05%	19.34
A	150	30	4552.1	**6.96%**	1.05	**0.03%**	23.28	11.76%	22.03	11.76%	61.37
	150	60	17482.4	**2.92%**	1.09	**0.00%**	24.44	7.50%	45.93	6.59%	137.31
	200	40	8132.1	**6.77%**	2.19	**0.01%**	96.46	11.14%	79.27	11.13%	261.01
	200	80	31048.6	**2.93%**	2.23	**0.00%**	94.18	7.21%	181.69	6.30%	655.06
	250	50	12654.0	**6.85%**	4.47	**0.01%**	208.52	11.79%	249.49	11.79%	880.03
	250	100	48384.3	**2.89%**	4.15	**0.00%**	196.77	7.44%	553.10	6.59%	1785.10
	50	10	334976	**17.35%**	0.06	**8.16%**	0.50	20.03%	0.20	20.03%	0.33
	50	20	1171416	**5.75%**	0.06	**1.77%**	0.47	27.87%	0.33	27.87%	0.60
	100	20	1267277	**13.33%**	0.33	**7.76%**	4.50	34.34%	5.06	34.34%	12.11
	100	40	4544642	**4.16%**	0.37	**1.99%**	5.00	36.67%	9.09	36.67%	21.50
B	150	30	2758381	**10.85%**	1.01	**6.59%**	18.17	41.17%	32.00	41.17%	73.06
	150	60	9960461	**3.62%**	1.05	**1.83%**	20.76	41.22%	61.51	41.22%	144.62
	200	40	4788086	**9.37%**	2.23	**6.16%**	73.15	45.81%	121.18	45.81%	332.12
	200	80	17544448	**2.75%**	2.32	**1.48%**	78.51	43.04%	324.12	43.04%	795.99
	250	50	7389784	**8.57%**	4.31	**5.94%**	165.85	48.64%	518.44	48.64%	1145.64
	250	100	27168460	**2.74%**	4.18	**1.55%**	175.25	44.95%	1212.75	44.95%	2378.38
	50	10	316409	33.47%	0.06	**2.35%**	0.44	15.91%	0.11	15.87%	0.18
	50	20	1094343	6.07%	0.06	0.34%	0.47	7.84%	0.24	0.00%	0.32
	100	20	1207522	33.48%	0.33	**4.19%**	4.86	28.23%	3.92	28.23%	7.12
	100	40	4219476	**11.24%**	0.35	**0.81%**	5.29	17.62%	9.00	12.94%	22.52
C	150	30	2613286	33.19%	0.99	**4.28%**	20.99	33.15%	25.35	33.15%	55.38
	150	60	9374611	**10.54%**	1.03	**0.39%**	21.13	19.67%	53.79	16.74%	142.00
	200	40	4630545	**34.03%**	2.10	**3.65%**	78.86	36.19%	113.21	36.19%	206.59
	200	80	16759895	**10.56%**	2.11	**0.35%**	83.74	21.23%	218.91	18.37%	626.02
	250	50	7178043	**34.51%**	3.76	**3.73%**	177.04	38.06%	334.49	38.06%	710.53
	250	100	26047022	**11.21%**	3.80	**0.36%**	178.10	23.09%	613.12	20.32%	1708.04
	50	10	381379	18.83%	0.06	**6.67%**	0.44	11.04%	0.20	11.04%	0.35
	50	20	1502908	**4.33%**	0.06	**0.85%**	0.48	10.64%	0.49	10.64%	0.86
	100	20	1570800	26.30%	0.34	**5.69%**	4.85	12.98%	6.89	12.98%	12.01
	100	40	6067776	**8.95%**	0.36	**0.77%**	5.58	11.71%	13.13	11.71%	24.50
	150	30	3502567	27.26%	1.01	**6.22%**	24.02	15.81%	45.49	15.81%	93.82
D	150	60	13611262	**8.98%**	0.97	**0.82%**	22.06	12.43%	103.83	12.43%	184.14
	200	40	6207580	27.43%	2.41	**5.69%**	93.87	16.82%	212.94	16.82%	352.72
	200	80	24133321	**9.02%**	2.24	**0.73%**	87.15	12.70%	456.80	12.70%	770.91
	250	50	9685430	27.74%	3.83	**5.79%**	210.35	17.54%	636.22	17.54%	1249.81
	250	100	37753120	**9.02%**	3.97	**0.65%**	195.33	12.77%	1291.24	12.77%	3237.35

Table 1: Results on benchmark B_1

Instance		Best	CSDP 1		CSDP 2		CPLEX		CPLEX + all	
n	m	known	% gap	CPU	% gap	CPU	% gap	CPU	% gap	CPU
100	10	333	35.47%	0.32	27.08%	4.91	20.94%	2.06	20.65%	75.76
100	20	1195	**15.19%**	0.33	**6.83%**	4.22	34.23%	6.58	34.19%	131.18
100	30	2457	**6.52%**	0.37	**2.53%**	4.54	39.55%	9.60	39.45%	123.56
100	40	4142	**4.99%**	0.35	**2.02%**	4.94	39.67%	11.71	39.62%	164.66
200	20	1247	**30.07%**	2.18	**23.71%**	72.50	36.42%	210.11	36.26%	2817.06
200	40	4450	**10.89%**	2.35	**7.01%**	74.56	48.14%	230.34	48.12%	6849.75
200	60	9437	**5.64%**	2.20	**2.62%**	77.48	48.74%	278.23	48.64%	4364.91
200	80	16225	**3.16%**	2.17	**1.71%**	77.61	45.95%	248.65	45.94%	5233.09
300	30	2694	**25.10%**	6.58	**19.58%**	308.67	44.47%	1742.33	44.39%	21511.81
300	60	9689	**8.66%**	7.11	**5.76%**	303.70	54.64%	3285.25	54.63%	52261.59
300	90	20743	**4.53%**	6.49	**2.68%**	328.11	53.83%	2583.27	53.81%	39832.66
300	120	35881	**3.03%**	6.80	**1.31%**	341.63	49.18%	2611.45	49.16%	50104.42
400	40	4658	**21.92%**	14.54	**17.81%**	992.75	50.09%	6389.14	50.02%	1 day
400	80	16956	**7.84%**	15.43	**5.38%**	977.11	57.92%	10046.77	57.91%	1 day
400	120	36317	**3.81%**	16.48	**2.44%**	1051.53	56.91%	12444.66	56.90%	1 day
400	160	62487	**2.35%**	16.15	**1.46%**	1099.05	52.26%	11763.89	52.26%	1 day
500	50	7141	**19.90%**	27.43	**15.99%**	2585.36	53.69%	16009.04	53.66%	1 day
500	100	26258	**7.02%**	30.43	**4.35%**	2559.37	59.76%	41375.73	n.a.	1 day
500	150	56572	**3.68%**	30.56	**1.93%**	2580.45	52.14%	24925.28	57.60%	1 day
500	200	97344	**2.26%**	30.56	**1.14%**	2718.72	52.92%	36628.22	52.91%	1 day

Table 2: Results on benchmark B_2

Problem. The computational results show the tightness of these bounds with respect to the best known results from literature. The average gap ranges between 2.45% and 12.89% for the easiest instances in B_1 and between 7.67% and 11.10% for the hardest ones in B_2. The average computational time in seconds ranges from 1.45 to 60.34 and from 10.75 to 808.36 for B_1 and B_2, respectively. We have also compared these results with those obtained by solving with CPLEX a linearized integer programming formulation. These gaps, on average, are much larger than the semidefinite bounds. Moreover, CPLEX requires a longer average computational time than CSDP.

Acknowledgment

The authors wish to thank Brian Borchers for his help and useful suggestions in using his code for semidefinite programming CSDP [5].

References

[1] P. M. D. Andrade, A. Plastino, L. S. Ochi, and S. L. Martins. GRASP for the Maximum Diversity Problem. In *Proceedings of the Fifth Metaheuristics International Conference (MIC 2003)*, 2003.

[2] P. M. D. Andrade, L. S. Plastino, and S. L. Martins. GRASP with path-relinking for the maximum diversity problem. In S. Nikoletseas, editor, *Proceedings of the 4th International Workshop on Efficient and Experimental Algorithms (WEA 2005)*, volume 3539 of *Lecture Notes in Computer Science*, pages 558–569. Springer Berlin / Heidelberg, 2005.

[3] R. Aringhieri and R. Cordone. Better and faster solutions for the maximum diversity problem. Note del Polo 93, Università degli Studi di Milano, Crema, April 2006.

[4] R. Aringhieri, R. Cordone, and Y. Melzani. Tabu search vs. GRASP for the Maximum Diversity Problem. Note del Polo 89, Università degli Studi di Milano, Crema, December 2005. [submitted for publication to *4OR*].

[5] B. Borchers. CSDP, a C library for semidefinite programming. *Optimization Methods & Software*, 11(2(1-4)):613–623, 1999.

[6] I. Djukanovic and F. Rendl. Semidefinite programming relaxations for graph coloring and maximal clique problems. Technical report, University of Klagenfurt, 2005. www.math.uni-klu.ac.at/or/Forschung/index.php.

[7] A. Duarte and R. Martí. Tabu search for the maximum diversity problem. *European Journal of Operational Research*. [to appear].

[8] J. B. Ghosh. Computational aspects of maximum diversity problem. *Operation Research Letters*, 19:175–181, 1996.

[9] F. Glover, G. Hersh, and C. McMillian. Selecting subset of maximum diversity. MS/IS 77-9, University of Colorado at Boulder, 1977.

[10] F. Glover, C. C. Kuo, and K. S. Dhir. A discrete optimization model for preserving biological diversity. *Appl. Math. Modelling*, 19(11):696–701, November 1995.

[11] M.X. Goemans and D.P. Williamson. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *Journal of the ACM*, 42:1115–1145, 1995.

[12] G. Kochenberger and F. Glover. Diversity data mining. Working Paper Series HCES-03-99, The University of Mississipi, 1999.

[13] C. C. Kuo, F. Glover, and K.S. Dhir. Analyzing and modeling the maximum diversity problem by zero-one programming. *Decision Science*, 24:1171–1185, 1993.

[14] M. Laurent and F. Rendl. *Semidefinite Programming and Integer Programming*, pages 393–514. Elsevier B.V., 2005.

[15] L.F. Santos, M.H. Ribeiro, A. Plastino, and S.L. Martins. A Hybrid GRASP with Data Mining for the Maximum Diversity Problem. In Andrea Roli Michael Sampels Mara J. Blesa, Christian Blum, editor, *Hybrid Metaheuristics, Second International Workshop*, volume 3636 of *Lecture Notes in Computer Science*, pages 116–127. Springer Berlin / Heidelberg, 2005.

[16] G. C. Silva, L. S. Ochi, and S. L. Martins. Experimental comparison of greedy randomized adaptive search procedures for the maximum diversity problem. In *Proceedings of the 3rd International Workshop on Efficient and Experimental Algorithms (WEA 2004)*, volume 3059 of *Lectures Notes on Computer Science*, pages 498–512. Springer Berlin / Heidelberg, 2004.