Biq Mac Library - A collection of Max-Cut and quadratic 0-1 programming instances of medium size

Angelika Wiegele*

September 2007

Abstract

This is a collection of some Max-Cut and quadratic 0-1 programming instances of medium size ($n = 20$ up to $n = 500$, most of the instances having size $n = 100$).

1 Contents

In the subsequent sections Max-Cut and quadratic 0-1 programming instances, collected while developing the Biq Mac solver [1] (an SDP based Branch & Bound code [9]) are given. For each class of instances a table lists the problem names and the optimal solution values. For instances where the optimum is not known, we give some lower/upper bounds. (Note, that we do not claim, that these are the best known bounds.) Furthermore, the dimension n (which is the number of vertices in the case of Max-Cut problems and the number of variables in the case of quadratic 0-1 problems) and the density d is given for all instances.

Explanations how the data have been generated and details about parameters are given in seperate sections below.

The files containing the datasets in rudy-output format or sparse matrix format, respectively, can be obtained from [11].

2 Quadratic 0-1 Programming problems

The problem to be solved is the following:

$$\min\{y^T Qy : \ y \in \{0,1\}^n\},$$

where Q is a symmetric matrix of order n.

2.1 Beasley instances

These data sets are due to [3] and can be obtained from the OR-Library [2] as well as from [11]. Note that in the OR-Library the problems are given as maximization problems!

*Alpen-Adria-Universität Klagenfurt, Institut für Mathematik, Universitätsstr. 65-67, 9020 Klagenfurt, Austria, angelika.wiegele@uni-klu.ac.at

Problem name	solution
$n = 50, d = 0.1$	
bqp50-1	-2098
bqp50-2	-3702
bqp50-3	-4626
bqp50-4	-3544
bqp50-5	-4012
bqp50-6	-3693
bqp50-7	-4520
bqp50-8	-4216
bqp50-9	-3780
bqp50-10	-3507
$n = 100, d = 0.1$	
bqp100-1	-7970
bqp100-2	-11036
bqp100-3	-12723
bqp100-4	-10368
bqp100-5	-9083
bqp100-6	-10210
bqp100-7	-10125
bqp100-8	-11435
bqp100-9	-11455
bqp100-10	-12565

Problem name	solution
$n = 250, d = 0.1$	
bqp250-1	-45607
bqp250-2	-44810
bqp250-3	-49037
bqp250-4	-41274
bqp250-5	-47961
bqp250-6	-41014
bqp250-7	-46757
bqp250-8	-35726
bqp250-9	-48916
bqp250-10	-40442

Problem name	solution	lower bound
$n = 500, d = 0.1$		
bqp500-1	$\leq$ -116586	-121588.41
bqp500-2	$\leq$ -128223	-132216.45
bqp500-3	$\leq$ -130812	-134214.12
bqp500-4	$\leq$ -130097	-134781.02
bqp500-5	$\leq$ -125487	-129572.87
bqp500-6	$\leq$ -121772	-126429.50
bqp500-7	$\leq$ -122201	-127136.37
bqp500-8	$\leq$ -123559	-128574.61
bqp500-9	$\leq$ -120798	-125821.63
bqp500-10	$\leq$ -130619	-134352.34

Table 1: Beasley data. For details see Section 2.1 on page 1.

All problems have 10% density and all the coefficient elements have an integer uniform value in the [-100,100] interval. The sizes, optimal values or lower/upper bounds are given in Table 1 on page 2.

2.2 Glover, Kochenberger and Alidaee instances

These data sets are due to [5] and can be obtained from the OR-Library [2] as well as from [11]. Note that in the OR-Library the problems are given as maximization problems!

The sizes, densities and optimal solution values are given in Table 2 (page 4) and Table 3 (page 5). The parameters for generating the datasets using the Pardalos-Rodgers generators are given in Section 4.1 below.

Set a	
diagonal coefficients:	integer uniform [-100,100]
off-diagonal coefficients:	integer uniform [-100,100]
Set b	
diagonal coefficients:	integer uniform [-63,0]
off-diagonal coefficients:	integer uniform [0,100]
Set c	
diagonal coefficients:	integer uniform [-100,100]
off-diagonal coefficients:	integer uniform [-50,50]
Set d	
diagonal coefficients:	integer uniform [-75,75]
off-diagonal coefficients:	integer uniform [-50,50]
Set e	
diagonal coefficients:	integer uniform [-100,100]
off-diagonal coefficients:	integer uniform [-50,50]
Set f	
diagonal coefficients:	integer uniform [-75,75]
off-diagonal coefficients:	integer uniform [-50,50]

2.3 Billionnet and Elloumi instances

In [4] instances of size $n = 100, 120, 150, 200$ and different densities using the generator introduced in [8] (see also Section 4.1) are generated. For each class of problems, ten instances have been generated. The parameters are the following:

- diagonal coefficients in the range $[-100, 100]$,
- off-diagonal coefficients in the range $[-50, 50]$,
- seeds 1,2,...,10.

We extended these instances by a set of 10 instances with $n = 250$, $d = 0.1$. The optimal solution values can be found in Tables 4 and 5 on pages 6 and 7, respectively.

3 Max-Cut instances

The problem to be solved is

$$\max\{x^T Lx : \ x \in \{-1, 1\}^n\},$$

where L is the Laplace matrix of the given graph of n vertices.

3.1 Max-Cut instances generated with rudy

Graphs of the following types have been generated using `rudy` [10]. The solution values are listed in Tables 6 to 8 on pages 9 to 11. The data can be downloaded [11] or generated by the rudy calls given in Section 4.2 below.

- $G_{0.5}$ `g05_`n`.`i
  unweighted graphs with edge probability 1/2.
  $n = 60, 80, 100$

Problem name	n	density	solution
gka1a	50	0.1	-3414
gka2a	60	0.1	-6063
gka3a	70	0.1	-6037
gka4a	80	0.1	-8598
gka5a	50	0.2	-5737
gka6a	30	0.4	-3980
gka7a	30	0.5	-4541
gka8a	100	0.0625	-11109
gka1b	20	1	-133
gka2b	30	1	-121
gka3b	40	1	-118
gka4b	50	1	-129
gka5b	60	1	-150
gka6b	70	1	-146
gka7b	80	1	-160
gka8b	90	1	-145
gka9b	100	1	-137
gka10b	125	1	-154
gka1c	40	0.8	-5058
gka2c	50	0.6	-6213
gka3c	60	0.4	-6665
gka4c	70	0.3	-7398
gka5c	80	0.2	-7362
gka6c	90	0.1	-5824
gka7c	100	0.1	-7225

Table 2: [5] data. For details see Section 2.2 on page 2.

Problem name	n	density	solution
gka1d	100	0.1	-6333
gka2d	100	0.2	-6579
gka3d	100	0.3	-9261
gka4d	100	0.4	-10727
gka5d	100	0.5	-11626
gka6d	100	0.6	-14207
gka7d	100	0.7	-14476
gka8d	100	0.8	-16352
gka9d	100	0.9	-15656
gka10d	100	1	-19102
gka1e	200	0.1	-16464
gka2e	200	0.2	-23395
gka3e	200	0.3	-25243
gka4e	200	0.4	-35594
gka5e	200	0.5	-35154

Problem name	n	density	solution	lower bound
gka1f	500	0.1	$\leq$ -61194	-63400.98
gka2f	500	0.25	$\leq$ -100161	-104868.34
gka3f	500	0.5	$\leq$ -138035	-145420.14
gka4f	500	0.75	$\leq$ -172771	-181507.74
gka5f	500	1	$\leq$ -190507	-201130.98

Table 3: [5] data. For details see Section 2.2 on page 2.

Problem name	solution
$n = 100, d = 1.0$	
be100.1	-19412
be100.2	-17290
be100.3	-17565
be100.4	-19125
be100.5	-15868
be100.6	-17368
be100.7	-18629
be100.8	-18649
be100.9	-13294
be100.10	-15352
$n = 120, d = 0.3$	
be120.3.1	-13067
be120.3.2	-13046
be120.3.3	-12418
be120.3.4	-13867
be120.3.5	-11403
be120.3.6	-12915
be120.3.7	-14068
be120.3.8	-14701
be120.3.9	-10458
be120.3.10	-12201
$n = 120, d = 0.8$	
be120.8.1	-18691
be120.8.2	-18827
be120.8.3	-19302
be120.8.4	-20765
be120.8.5	-20417
be120.8.6	-18482
be120.8.7	-22194
be120.8.8	-19534
be120.8.9	-18195
be120.8.10	-19049

Problem name	solution
$n = 150, d = 0.3$	
be150.3.1	-18889
be150.3.2	-17816
be150.3.3	-17314
be150.3.4	-19884
be150.3.5	-16817
be150.3.6	-16780
be150.3.7	-18001
be150.3.8	-18303
be150.3.9	-12838
be150.3.10	-17963
$n = 150, d = 0.8$	
be150.8.1	-27089
be150.8.2	-26779
be150.8.3	-29438
be150.8.4	-26911
be150.8.5	-28017
be150.8.6	-29221
be150.8.7	-31209
be150.8.8	-29730
be150.8.9	-25388
be150.8.10	-28374

Table 4: Instances from [4]. For details see Section 2.3 on page 3.

Problem name	solution
$n = 200, d = 0.3$	
be200.3.1	-25453
be200.3.2	-25027
be200.3.3	-28023
be200.3.4	-27434
be200.3.5	-26355
be200.3.6	-26146
be200.3.7	-30483
be200.3.8	-27355
be200.3.9	-24683
be200.3.10	-23842
$n = 200, d = 0.8$	
be200.8.1	-48534
be200.8.2	-40821
be200.8.3	-43207
be200.8.4	-43757
be200.8.5	-41482
be200.8.6	-49492
be200.8.7	-46828
be200.8.8	-44502
be200.8.9	-43241
be200.8.10	-42832

Problem name	solution
$n = 250, d = 0.1$	
be250.1	-24076
be250.2	-22540
be250.3	-22923
be250.4	-24649
be250.5	-21057
be250.6	-22735
be250.7	-24095
be250.8	-23801
be250.9	-20051
be250.10	-23159

Table 5: Instances from [4]. For details see Section 2.3 on page 3.

- $G_{-1/0/1}$ `pm1s_`$n.i$, `pm1d_`$n.i$
  weighted graph with edge weights chosen uniformly from $\{-1, 0, 1\}$ and density 10% and 99% respectively.

  $n = 80, 100$

- $G_{[-10,10]}$ `w`d`_`$n.i$
  Graph with integer edge weights chosen from [-10,10] and density $d = 0.1, 0.5, 0.9$, $n = 100$

- $G_{[0,10]}$ `pw`d`_`$n.i$
  Graph with integer edge weights chosen from [-10,10] and density $d = 0.1, 0.5, 0.9$, $n = 100$

3.2 Ising instances: Max-Cut instances from applications in statistical physics

Ising instances of two kinds (one-dimensional Ising chains and toroidal grid graphs) are given in this section. The instances can be downloaded from [11], the dimensions and optimal values are given in Table 9 and Table 10 (pages 12, 13). For a detailled description of these instances the reader is referred to the dissertation of Frauke Liers [6] and to [7].

4 Problem generators

4.1 Pardalos-Rodgers

Pardalos and Rodgers [8] have proposed a test problem generator for quadratic 0-1 programming. Their routine generates symmetric integer matrices and has several parameters to control the characteristics of the problem, namely:

n	number of variables
d	density, i.e. the probability that a nonzero will occur for any off-diag coefficient
c^-	lower bound of the diagonal coefficients (q_{ii})
c^+	upper bound of the diagonal coefficients (q_{ii})
q^-	lower bound of the off-diagonal coefficients (q_{ij})
q^+	upper bound of the off-diagonal coefficients (q_{ij})
s	seed to initialize the random number generator

$q_{ii} \sim$ discrete uniform $(c^-, c^+), i = 1, \ldots, n$

$q_{ij} = q_{ji} \sim$ discrete uniform $(q^-, q^+), 1 \leq i < j \leq n.$

The expected degree of each quadratic 0-1 programming instance is the expected number of quadratic nonzeros per variable. Therefore, the set of Pardalos problems have a fixed expected degree, i.e. $(n - 1)d$.

The parameters are given in the following order:

```
n density seed OffDiagonalLower OffDiagonalUpper DiagonalLower DiagonalUpper
```

Problem name	solution
$n = 60$	
g05_60.0	536
g05_60.1	532
g05_60.2	529
g05_60.3	538
g05_60.4	527
g05_60.5	533
g05_60.6	531
g05_60.7	535
g05_60.8	530
g05_60.9	533
$n = 80$	
g05_80.0	929
g05_80.1	941
g05_80.2	934
g05_80.3	923
g05_80.4	932
g05_80.5	926
g05_80.6	929
g05_80.7	929
g05_80.8	925
g05_80.9	923
$n = 100$	
g05_100.0	1430
g05_100.1	1425
g05_100.2	1432
g05_100.3	1424
g05_100.4	1440
g05_100.5	1436
g05_100.6	1434
g05_100.7	1431
g05_100.8	1432
g05_100.9	1430

Table 6: $G_{0.5}$ – unweighted graphs with edge probability 1/2. For details see Section 3.1 on page 3.

Problem name	solution
$n = 80, d = 0.1$	
pm1s_80.0	79
pm1s_80.1	85
pm1s_80.2	82
pm1s_80.3	81
pm1s_80.4	70
pm1s_80.5	87
pm1s_80.6	73
pm1s_80.7	83
pm1s_80.8	81
pm1s_80.9	70
$n = 100, d = 0.1$	
pm1s_100.0	127
pm1s_100.1	126
pm1s_100.2	125
pm1s_100.3	111
pm1s_100.4	128
pm1s_100.5	128
pm1s_100.6	122
pm1s_100.7	112
pm1s_100.8	120
pm1s_100.9	127

Problem name	solution
$n = 80, d = 0.99$	
pm1d_80.0	227
pm1d_80.1	245
pm1d_80.2	284
pm1d_80.3	291
pm1d_80.4	251
pm1d_80.5	242
pm1d_80.6	205
pm1d_80.7	249
pm1d_80.8	293
pm1d_80.9	258
$n = 100, d = 0.99$	
pm1d_100.0	340
pm1d_100.1	324
pm1d_100.2	389
pm1d_100.3	400
pm1d_100.4	363
pm1d_100.5	441
pm1d_100.6	367
pm1d_100.7	361
pm1d_100.8	385
pm1d_100.9	405

Table 7: $G_{-1/0/1}$, density 10% and 99%. For details see Section 3.1 on page 3.

Problem name	solution
$n = 100, d = 0.1$	
w01_100.0	651
w01_100.1	719
w01_100.2	676
w01_100.3	813
w01_100.4	668
w01_100.5	643
w01_100.6	654
w01_100.7	725
w01_100.8	721
w01_100.9	729
$n = 100, d = 0.5$	
w05_100.0	1646
w05_100.1	1606
w05_100.2	1902
w05_100.3	1627
w05_100.4	1546
w05_100.5	1581
w05_100.6	1479
w05_100.7	1987
w05_100.8	1311
w05_100.9	1752
$n = 100, d = 0.9$	
w09_100.0	2121
w09_100.1	2096
w09_100.2	2738
w09_100.3	1990
w09_100.4	2033
w09_100.5	2433
w09_100.6	2220
w09_100.7	2252
w09_100.8	1843
w09_100.9	2043

Problem name	solution
$n = 100, d = 0.1$	
pw01_100.0	2019
pw01_100.1	2060
pw01_100.2	2032
pw01_100.3	2067
pw01_100.4	2039
pw01_100.5	2108
pw01_100.6	2032
pw01_100.7	2074
pw01_100.8	2022
pw01_100.9	2005
$n = 100, d = 0.5$	
pw05_100.0	8190
pw05_100.1	8045
pw05_100.2	8039
pw05_100.3	8139
pw05_100.4	8125
pw05_100.5	8169
pw05_100.6	8217
pw05_100.7	8249
pw05_100.8	8199
pw05_100.9	8099
$n = 100, d = 0.9$	
pw09_100.0	13585
pw09_100.1	13417
pw09_100.2	13461
pw09_100.3	13656
pw09_100.4	13514
pw09_100.5	13574
pw09_100.6	13640
pw09_100.7	13501
pw09_100.8	13593
pw09_100.9	13658

Table 8: $G_{[-10,10]}$ (w) and $G_{[1,10]}$ (pw). For details see Section 3.1 on page 3.

Problem name	solution
$n = 100$	
ising2.5-100_5555	2460049
ising2.5-100_6666	2031217
ising2.5-100_7777	3363230
ising3.0-100_5555	2448189
ising3.0-100_6666	1984099
ising3.0-100_7777	3335814
$n = 150$	
ising2.5-150_5555	4363532
ising2.5-150_6666	4057153
ising2.5-150_7777	4243269
ising3.0-150_5555	4279261
ising3.0-150_6666	3949317
ising3.0-150_7777	4211158
$n = 200$	
ising2.5-200_5555	6294701
ising2.5-200_6666	6795365
ising2.5-200_7777	5568272
ising3.0-200_5555	6215531
ising3.0-200_6666	6756263
ising3.0-200_7777	5560824
$n = 250$	
ising2.5-250_5555	7919449
ising2.5-250_6666	6925717
ising2.5-250_7777	6596797
ising3.0-250_5555	7823791
ising3.0-250_6666	6903351
ising3.0-250_7777	6418276
$n = 300$	
ising2.5-300_5555	8579363
ising2.5-300_6666	9102033
ising2.5-300_7777	8323804
ising3.0-300_5555	8493173
ising3.0-300_6666	8915110
ising3.0-300_7777	8242904

Table 9: Test runs on one-dimensional Ising chain instances from Frauke Liers. For details see Section 3.2 on page 8.

Problem name	solution
2 dimensional	
$n = 10 \times 10$	
t2g10_5555	6049461
t2g10_6666	5757868
t2g10_7777	6509837
$n = 15 \times 15$	
t2g15_5555	15051133
t2g15_6666	15763716
t2g15_7777	15269399
$n = 20 \times 20$	
t2g20_5555	24838942
t2g20_6666	29290570
t2g20_7777	28349398
3 dimensional	
$n = 5 \times 5 \times 5$	
t3g5_5555	10933215
t3g5_6666	11582216
t3g5_7777	11552046
$n = 6 \times 6 \times 6$	
t3g6_5555	17434469
t3g6_6666	20217380
t3g6_7777	19475011
$n = 7 \times 7 \times 7$	
t3g7_5555	28302918
t3g7_6666	33611981
t3g7_7777	29118445

Table 10: Torus graphs with Gaussian distributed weights from Frauke Liers. For details see Section 3.2 on page 8.

Parameters for generating the gka-instances

Set a.

```
50 0.1 10 -100 100 -100 100
60 0.1 10 -100 100 -100 100
70 0.1 10 -100 100 -100 100
80 0.1 10 -100 100 -100 100
50 0.2 10 -100 100 -100 100
30 0.4 10 -100 100 -100 100
30 0.5 10 -100 100 -100 100
100 0.0625 10 -100 100 -100 100
```

Set b.

```
 20 1.0 10 0 63 -100 0
 30 1.0 10 0 63 -100 0
 40 1.0 10 0 63 -100 0
 50 1.0 10 0 63 -100 0
 60 1.0 10 0 63 -100 0
 70 1.0 10 0 63 -100 0
 80 1.0 10 0 63 -100 0
 90 1.0 10 0 63 -100 0
100 1.0 10 0 63 -100 0
125 1.0 10 0 63 -100 0
```

Set c.

```
 40 0.8  10 -100 100 -50 50
 50 0.6  70 -100 100 -50 50
 60 0.4  31 -100 100 -50 50
 70 0.3  34 -100 100 -50 50
 80 0.2   8 -100 100 -50 50
 90 0.1  80 -100 100 -50 50
100 0.1 142 -100 100 -50 50
```

Set d.

```
100 0.1   31 -50 50 -75 75
100 0.2   37 -50 50 -75 75
100 0.3  143 -50 50 -75 75
100 0.4   47 -50 50 -75 75
100 0.5   31 -50 50 -75 75
100 0.6   47 -50 50 -75 75
100 0.7   97 -50 50 -75 75
100 0.8  133 -50 50 -75 75
100 0.9  307 -50 50 -75 75
100 1.0 1311 -50 50 -75 75
```

Set e.

```
200 0.1 51 -50 50 -100 100
200 0.2 43 -50 50 -100 100
200 0.3 34 -50 50 -100 100
200 0.4 73 -50 50 -100 100
200 0.5 89 -50 50 -100 100
```

Set f.

```
500 0.10 137 -50 50 -100 100
500 0.25 137 -50 50 -100 100
500 0.50 137 -50 50 -100 100
500 0.75 137 -50 50 -100 100
500 1.00 137 -50 50 -100 100
```

Parameters for generating the be-instances

```
100 1.0 1. -50 50 -100 100
100 1.0 2. -50 50 -100 100
100 1.0 3. -50 50 -100 100
100 1.0 4. -50 50 -100 100
100 1.0 5. -50 50 -100 100
100 1.0 6. -50 50 -100 100
100 1.0 7. -50 50 -100 100
100 1.0 8. -50 50 -100 100
100 1.0 9. -50 50 -100 100
100 1.0 10. -50 50 -100 100
```

```
120 0.3 1. -50 50 -100 100
...
120 0.3 10. -50 50 -100 100

120 0.8 1. -50 50 -100 100
...
120 0.8 10. -50 50 -100 100

150 0.3 1. -50 50 -100 100
...
150 0.3 10. -50 50 -100 100

150 0.8 1. -50 50 -100 100
...
150 0.8 10. -50 50 -100 100
```

```
200 0.3 1. -50 50 -100 100
...
200 0.3 10. -50 50 -100 100

200 0.8 1. -50 50 -100 100
...
200 0.8 10. -50 50 -100 100

250 0.1 1. -50 50 -100 100
...
250 0.1 10. -50 50 -100 100
```

4.2 Rudy

Most of the Max-Cut instances given in this paper are generated using `rudy` [10]. The commands are given below.

```
rudy -rnd_graph 60 50 6000 > g05_60.0
rudy -rnd_graph 60 50 6001 > g05_60.1
rudy -rnd_graph 60 50 6002 > g05_60.2
rudy -rnd_graph 60 50 6003 > g05_60.3
rudy -rnd_graph 60 50 6004 > g05_60.4
rudy -rnd_graph 60 50 6005 > g05_60.5
rudy -rnd_graph 60 50 6006 > g05_60.6
rudy -rnd_graph 60 50 6007 > g05_60.7
rudy -rnd_graph 60 50 6008 > g05_60.8
rudy -rnd_graph 60 50 6009 > g05_60.9

rudy -rnd_graph 80 50 8000 > g05_80.0
rudy -rnd_graph 80 50 8001 > g05_80.1
rudy -rnd_graph 80 50 8002 > g05_80.2
rudy -rnd_graph 80 50 8003 > g05_80.3
rudy -rnd_graph 80 50 8004 > g05_80.4
rudy -rnd_graph 80 50 8005 > g05_80.5
rudy -rnd_graph 80 50 8006 > g05_80.6
rudy -rnd_graph 80 50 8007 > g05_80.7
rudy -rnd_graph 80 50 8008 > g05_80.8
rudy -rnd_graph 80 50 8009 > g05_80.9

rudy -rnd_graph 100 50 10000 > g05_100.0
rudy -rnd_graph 100 50 10001 > g05_100.1
rudy -rnd_graph 100 50 10002 > g05_100.2
rudy -rnd_graph 100 50 10003 > g05_100.3
rudy -rnd_graph 100 50 10004 > g05_100.4
rudy -rnd_graph 100 50 10005 > g05_100.5
rudy -rnd_graph 100 50 10006 > g05_100.6
rudy -rnd_graph 100 50 10007 > g05_100.7
rudy -rnd_graph 100 50 10008 > g05_100.8
rudy -rnd_graph 100 50 10009 > g05_100.9

rudy -rnd_graph 80 10 800 -random 0 1 800 -times 2 -plus -1 > pm1s_80.0
rudy -rnd_graph 80 10 801 -random 0 1 801 -times 2 -plus -1 > pm1s_80.1
rudy -rnd_graph 80 10 802 -random 0 1 802 -times 2 -plus -1 > pm1s_80.2
```

```
rudy -rnd_graph 80 10 803 -random 0 1 803 -times 2 -plus -1 > pm1s_80.3
rudy -rnd_graph 80 10 804 -random 0 1 804 -times 2 -plus -1 > pm1s_80.4
rudy -rnd_graph 80 10 805 -random 0 1 805 -times 2 -plus -1 > pm1s_80.5
rudy -rnd_graph 80 10 806 -random 0 1 806 -times 2 -plus -1 > pm1s_80.6
rudy -rnd_graph 80 10 807 -random 0 1 807 -times 2 -plus -1 > pm1s_80.7
rudy -rnd_graph 80 10 808 -random 0 1 808 -times 2 -plus -1 > pm1s_80.8
rudy -rnd_graph 80 10 809 -random 0 1 809 -times 2 -plus -1 > pm1s_80.9

rudy -rnd_graph 100 10 1000 -random 0 1 1000 -times 2 -plus -1 > pm1s_100.0
rudy -rnd_graph 100 10 1001 -random 0 1 1001 -times 2 -plus -1 > pm1s_100.1
rudy -rnd_graph 100 10 1002 -random 0 1 1002 -times 2 -plus -1 > pm1s_100.2
rudy -rnd_graph 100 10 1003 -random 0 1 1003 -times 2 -plus -1 > pm1s_100.3
rudy -rnd_graph 100 10 1004 -random 0 1 1004 -times 2 -plus -1 > pm1s_100.4
rudy -rnd_graph 100 10 1005 -random 0 1 1005 -times 2 -plus -1 > pm1s_100.5
rudy -rnd_graph 100 10 1006 -random 0 1 1006 -times 2 -plus -1 > pm1s_100.6
rudy -rnd_graph 100 10 1007 -random 0 1 1007 -times 2 -plus -1 > pm1s_100.7
rudy -rnd_graph 100 10 1008 -random 0 1 1008 -times 2 -plus -1 > pm1s_100.8
rudy -rnd_graph 100 10 1009 -random 0 1 1009 -times 2 -plus -1 > pm1s_100.9

rudy -rnd_graph 80 99 800 -random 0 1 800 -times 2 -plus -1 > pm1d_80.0
rudy -rnd_graph 80 99 801 -random 0 1 801 -times 2 -plus -1 > pm1d_80.1
rudy -rnd_graph 80 99 802 -random 0 1 802 -times 2 -plus -1 > pm1d_80.2
rudy -rnd_graph 80 99 803 -random 0 1 803 -times 2 -plus -1 > pm1d_80.3
rudy -rnd_graph 80 99 804 -random 0 1 804 -times 2 -plus -1 > pm1d_80.4
rudy -rnd_graph 80 99 805 -random 0 1 805 -times 2 -plus -1 > pm1d_80.5
rudy -rnd_graph 80 99 806 -random 0 1 806 -times 2 -plus -1 > pm1d_80.6
rudy -rnd_graph 80 99 807 -random 0 1 807 -times 2 -plus -1 > pm1d_80.7
rudy -rnd_graph 80 99 808 -random 0 1 808 -times 2 -plus -1 > pm1d_80.8
rudy -rnd_graph 80 99 809 -random 0 1 809 -times 2 -plus -1 > pm1d_80.9

rudy -rnd_graph 100 99 1000 -random 0 1 1000 -times 2 -plus -1 > pm1d_100.0
rudy -rnd_graph 100 99 1001 -random 0 1 1001 -times 2 -plus -1 > pm1d_100.1
rudy -rnd_graph 100 99 1002 -random 0 1 1002 -times 2 -plus -1 > pm1d_100.2
rudy -rnd_graph 100 99 1003 -random 0 1 1003 -times 2 -plus -1 > pm1d_100.3
rudy -rnd_graph 100 99 1004 -random 0 1 1004 -times 2 -plus -1 > pm1d_100.4
rudy -rnd_graph 100 99 1005 -random 0 1 1005 -times 2 -plus -1 > pm1d_100.5
rudy -rnd_graph 100 99 1006 -random 0 1 1006 -times 2 -plus -1 > pm1d_100.6
rudy -rnd_graph 100 99 1007 -random 0 1 1007 -times 2 -plus -1 > pm1d_100.7
rudy -rnd_graph 100 99 1008 -random 0 1 1008 -times 2 -plus -1 > pm1d_100.8
rudy -rnd_graph 100 99 1009 -random 0 1 1009 -times 2 -plus -1 > pm1d_100.9

rudy -rnd_graph 100 10 1000 -random -10 10 1000 > w01_100.0
rudy -rnd_graph 100 10 1001 -random -10 10 1001 > w01_100.1
rudy -rnd_graph 100 10 1002 -random -10 10 1002 > w01_100.2
rudy -rnd_graph 100 10 1003 -random -10 10 1003 > w01_100.3
rudy -rnd_graph 100 10 1004 -random -10 10 1004 > w01_100.4
rudy -rnd_graph 100 10 1005 -random -10 10 1005 > w01_100.5
rudy -rnd_graph 100 10 1006 -random -10 10 1006 > w01_100.6
rudy -rnd_graph 100 10 1007 -random -10 10 1007 > w01_100.7
rudy -rnd_graph 100 10 1008 -random -10 10 1008 > w01_100.8
rudy -rnd_graph 100 10 1009 -random -10 10 1009 > w01_100.9

rudy -rnd_graph 100 50 1000 -random -10 10 1000 > w05_100.0
rudy -rnd_graph 100 50 1001 -random -10 10 1001 > w05_100.1
rudy -rnd_graph 100 50 1002 -random -10 10 1002 > w05_100.2
rudy -rnd_graph 100 50 1003 -random -10 10 1003 > w05_100.3
rudy -rnd_graph 100 50 1004 -random -10 10 1004 > w05_100.4
rudy -rnd_graph 100 50 1005 -random -10 10 1005 > w05_100.5
rudy -rnd_graph 100 50 1006 -random -10 10 1006 > w05_100.6
rudy -rnd_graph 100 50 1007 -random -10 10 1007 > w05_100.7
rudy -rnd_graph 100 50 1008 -random -10 10 1008 > w05_100.8
rudy -rnd_graph 100 50 1009 -random -10 10 1009 > w05_100.9

rudy -rnd_graph 100 90 1000 -random -10 10 1000 > w09_100.0
rudy -rnd_graph 100 90 1001 -random -10 10 1001 > w09_100.1
rudy -rnd_graph 100 90 1002 -random -10 10 1002 > w09_100.2
rudy -rnd_graph 100 90 1003 -random -10 10 1003 > w09_100.3
rudy -rnd_graph 100 90 1004 -random -10 10 1004 > w09_100.4
rudy -rnd_graph 100 90 1005 -random -10 10 1005 > w09_100.5
rudy -rnd_graph 100 90 1006 -random -10 10 1006 > w09_100.6
rudy -rnd_graph 100 90 1007 -random -10 10 1007 > w09_100.7
rudy -rnd_graph 100 90 1008 -random -10 10 1008 > w09_100.8
rudy -rnd_graph 100 90 1009 -random -10 10 1009 > w09_100.9

rudy -rnd_graph 100 10 1000 -random 1 10 1000 > pw01_100.0
rudy -rnd_graph 100 10 1001 -random 1 10 1001 > pw01_100.1
rudy -rnd_graph 100 10 1002 -random 1 10 1002 > pw01_100.2
rudy -rnd_graph 100 10 1003 -random 1 10 1003 > pw01_100.3
rudy -rnd_graph 100 10 1004 -random 1 10 1004 > pw01_100.4
rudy -rnd_graph 100 10 1005 -random 1 10 1005 > pw01_100.5
rudy -rnd_graph 100 10 1006 -random 1 10 1006 > pw01_100.6
rudy -rnd_graph 100 10 1007 -random 1 10 1007 > pw01_100.7
rudy -rnd_graph 100 10 1008 -random 1 10 1008 > pw01_100.8
rudy -rnd_graph 100 10 1009 -random 1 10 1009 > pw01_100.9

rudy -rnd_graph 100 50 1000 -random 1 10 1000 > pw05_100.0
rudy -rnd_graph 100 50 1001 -random 1 10 1001 > pw05_100.1
rudy -rnd_graph 100 50 1002 -random 1 10 1002 > pw05_100.2
rudy -rnd_graph 100 50 1003 -random 1 10 1003 > pw05_100.3
rudy -rnd_graph 100 50 1004 -random 1 10 1004 > pw05_100.4
rudy -rnd_graph 100 50 1005 -random 1 10 1005 > pw05_100.5
rudy -rnd_graph 100 50 1006 -random 1 10 1006 > pw05_100.6
rudy -rnd_graph 100 50 1007 -random 1 10 1007 > pw05_100.7
rudy -rnd_graph 100 50 1008 -random 1 10 1008 > pw05_100.8
rudy -rnd_graph 100 50 1009 -random 1 10 1009 > pw05_100.9
```

```
rudy -rnd_graph 100 90 1000 -random 1 10 1000 > pw09_100.0
rudy -rnd_graph 100 90 1001 -random 1 10 1001 > pw09_100.1
rudy -rnd_graph 100 90 1002 -random 1 10 1002 > pw09_100.2
rudy -rnd_graph 100 90 1003 -random 1 10 1003 > pw09_100.3
rudy -rnd_graph 100 90 1004 -random 1 10 1004 > pw09_100.4
rudy -rnd_graph 100 90 1005 -random 1 10 1005 > pw09_100.5
rudy -rnd_graph 100 90 1006 -random 1 10 1006 > pw09_100.6
rudy -rnd_graph 100 90 1007 -random 1 10 1007 > pw09_100.7
rudy -rnd_graph 100 90 1008 -random 1 10 1008 > pw09_100.8
rudy -rnd_graph 100 90 1009 -random 1 10 1009 > pw09_100.9
```

References

[1] Biq Mac solver – a solver for *bi*nary *q*uadratic and *ma*x-*c*ut problems. `http://BiqMac.uni-klu.ac.at/`.

[2] John E. Beasley. Or-library. `http://people.brunel.ac.uk/~mastjjb/jeb/info.html`, 1990.

[3] John E. Beasley. Heuristic algorithms for the unconstrained binary quadratic programming problem. Technical report, The Management School, Imperial College, London SW7 2AZ, England, 1998.

[4] Alain Billionnet and Sourour Elloumi. Using a mixed integer quadratic programming solver for the unconstrained quadratic 0-1 problem. *Math. Programming*, 109(1, Ser. A):55–68, 2007.

[5] Fred Glover, Gary Kochenberger, and Bahram Alidaee. Adaptative memory tabu search for binary quadratic programs. *Management Sci.*, 44(3):336–345, 1998.

[6] Frauke Liers. *Contributions to Determining Exact Ground-States of Ising Spin-Glasses and to their Physics*. PhD thesis, Universität zu Köln, 2004.

[7] Frauke Liers, Michael Jünger, Gerhard Reinelt, and Giovanni Rinaldi. Computing exact ground states of hard ising spin glass problems by branch-and-cut. In Alexander Hartmann and Heiko Rieger, editors, *New Optimization Algorithms in Physics*, pages 47–68. Wiley, 2004.

[8] Panos M. Pardalos and Gregory P. Rodgers. Computational aspects of a branch and bound algorithm for quadratic zero-one programming. *Computing*, 45(2):131–144, 1990.

[9] Franz Rendl, Giovanni Rinaldi, and Angelika Wiegele. A branch and bound algorithm for Max-Cut based on combining semidefinite and polyhedral relaxations. In *Integer programming and combinatorial optimization*, volume 4513 of *Lecture Notes in Comput. Sci.*, pages 295–309. Springer, Berlin, 2007.

[10] Giovanni Rinaldi. Rudy. `http://www-user.tu-chemnitz.de/~helmberg/rudy.tar.gz`, 1998.

[11] Angelika Wiegele. Biq Mac Library. `http://biqmac.uni-klu.ac.at/biqmaclib.html`, 2007.