CDMTCS Research Report Series

Solving the Broadcast Time Problem Using a D-Wave Quantum Computer

Cristian S. Calude
Michael J. Dinneen

Department of Computer Science,
University of Auckland,
Auckland, New Zealand

CDMTCS-473
November 2014

Centre for Discrete Mathematics and
Theoretical Computer Science

Solving the Broadcast Time Problem Using a D-Wave Quantum Computer

Cristian S. Calude and Michael J. Dinneen
Department of Computer Science, University of Auckland, Auckland, New Zealand
{cristian,mjd}@cs.auckland.ac.nz

Abstract

Abstract. We provide a concise integer program formulation for the network broadcast time problem then convert it to QUBO form to be run on the D-Wave quantum computer. We then explore the feasibility of this method on several well-known graphs.

1 Introduction

Broadcasting concerns the dissemination of a message originating at one node of a network to all other nodes [5, 8]. This task is accomplished by placing a series of calls over the communication lines of the network between neighboring nodes. Each call requires a unit of time, a call can involve only two nodes and a node can participate in only one call per time step.

A *broadcast tree/protocol* for a vertex v (called the originator) of an undirected graph $G = (V, E)$ is a sequence $V_0 = \{v\}, E_1, V_1, E_2, \ldots, E_t, V_t = V$ (*of broadcast height* t) such that each $V_i \subseteq V$, each $E_i \subseteq E$, and for every $1 \leq i \leq t$:

(1) each edge in E_i has exactly one endpoint in V_{i-1},

(2) no two edges in E_i share a common endpoint, and

(3) $V_i = V_{i-1} \cup \{w \mid \{u, w\} \in E_i\}$.

The problem of minimizing t for an input graph is a well-known NP-hard problem (see [ND49] of [6]), even for graphs of maximum vertex degree 3 (see [3]).

Broadcast Problem 1:

Instance: Connected graph $G = (V, E)$, originator $v \in V$ and integer t.
Question: Is there a broadcast tree T_v rooted at v with the height of T_v at most t?
Or, as we write, is the (originator) broadcast time $B(G, v) \leq t$?

Broadcast Problem 2:

Instance:	Connected graph $G = (V, E)$ and integer t.
Question:	Is the graph broadcast time $B(G) = \max_{v \in V} B(G, v) \leq t$?

These decision problems have corresponding search/optimization problems where we want to find the smallest value of t. Clearly we can just focus on finding an efficient algorithm for the first broadcast problem since the second can be checked in $O(|V|)$ calls to the first.

In the development of the proposed solution, we will use the following example shown in Figure 1 (hypercube Q_3) with an optimal broadcast tree illustrated for originating vertex 0.

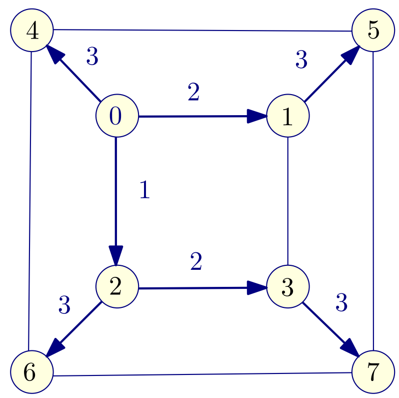

Figure 1: The graph Q_3 with broadcast time $B(Q_3) = 3$.

The quantum solution will be presented in a sequence of four phases.

2 Integer Programming Formulation

In the first phase we present a simple formulation (i.e. polynomial-time reduction) of the Broadcast Problem with the originator fixed[1] at $v = 0$ to an *Integer Programming (IP) Optimization Problem* (see [1, 10]). The input is a connected graph $G = (V = \{0, 1, \ldots, n-1\}, E)$ representing a network with $n = |V|$ vertices and $m = |E|$ edges. For the graph G, we use the following $n + 2m + 1$ variables:

- t is the required time to complete a broadcast,
- v_i is the time in $\{0, \ldots, t\}$ in which the vertex $i \in V$ is informed by message, $0 \leq i < n$,
- $b_{i,j}$ is a binary variable which is 1 if and only if the vertex i broadcasts to the vertex j (for each $\{i, j\} \in E$).

The objective function for our optimization problem is $\min t$ (or equivalently, $\max n - t$).

[1]Solving the problem for other originators can be easily done by relabelling the vertices of the graph or doing obvious modifications in the formulation below.

First, the time t must be at most $n-1$:

$$0 \leq t \leq n-1. \tag{1}$$

Every vertex receives the message at a time step at most t:

$$0 \leq v_i \leq t, \text{ for all } i \in V. \tag{2}$$

The originator vertex has no parent and every other vertex must have exactly one parent in the broadcast tree:

$$\sum_{j \neq 0} b_{j,0} \quad = 0, \tag{3}$$

$$\sum_{j \neq i} b_{j,i} \quad = 1, \quad \text{for all } i \in V \setminus \{0\}. \tag{4}$$

There are no broadcast cycles, that is for a child vertex, the informed time of the parent must be strictly less than its message received time:

$$b_{i,j}(1 + v_i - v_j) \quad \leq 0, \quad \text{for all } \{i,j\} \in E. \tag{5}$$

Finally, every two child vertices (j,k) informed by the same parent i must occur at different times:

$$b_{i,j} + b_{i,k} - (v_j - v_k)^2 \quad \leq 1, \quad \text{for all } \{i,j\} \in E, \{i,k\} \in E \text{ with } j \neq k \tag{6}$$

For our running example for Q_3, these integer programming constraints are listed in Appendix A.

3 Binary Integer Programming Formulation

Next we convert all non-binary variables (in IP formulation) into binary variables. The following simple procedure converts an integer constrained variable $0 \leq x \leq D$ into a set of $O(\lg D)$ binary variables $x_0, x_1, \ldots, x_c$ representing its binary representation:

$$x = x_0 + 2x_1 + 4x_2 + \cdots + 2^c x_c = \sum_{i=0}^{c} 2^i x_i,$$

where $x_i \in \{0,1\}$ and $2^c \leq D < 2^{c+1}$. Each constraint of the form $x \leq D$ is replaced by the following equivalent constraint:

$$\sum_{i=0}^{c} 2^i x_i \leq D \tag{7}$$

For the graph example Q_3 in Figure 1 we give a binary integer programming formulation using the above procedure in Appendix B.

4 Linear Binary Integer Programming Formulation

Using standard techniques (e.g. see [9]) we convert the above quadratic binary IP formulation into a linear formulation. Each occurrence of a product of two binary variables xy is replaced by a new variable z_{xy} and the following two linear constraints:

$$0 \leq x + y - z_{xy} \leq 1, \tag{8}$$

$$-1 \leq 2z_{xy} - (x + y) \leq 0, \tag{9}$$

enforce $z_{xy} = xy$.

We now show that $z_{xy} = xy$ is enforced. If both x and y are 1 then by equation (8), z_{xy} must be 1 and equation (9) is also satisfied. If at least one of x or y is 0 then by equation (9), z_{xy} must be 0 and equation (8) is also satisfied.

We can reduce the number of "product" binary variables by observing that for equation (5) we do not need to consider $j = 0$ and for equation (6) we only consider vertices $j > 0$ and $k > 0$ with a common neighbor.

A Python implementation for generating a linear binary integer program for the Broadcast Problem 1 is given in Appendix C. We have called the Sage Mixed Integer Program Solver (see [11]) to verify correctness of the generated IP formulation of our problem for several small test cases described later in Section 7.

For the graph example Q_3 in Figure 1 we give the final binary integer programming formulation and the solution in Appendix D.

5 QUBO Formulation

Using the idiom given in the D-Wave documentation (Section 4.1.4 of [2]) we show how our binary integer programming formulation is converted to a *Quadratic Unconstrained Binary Optimization* (QUBO) form. QUBO is an NP-hard mathematical optimization problem of minimizing a quadratic objective function $x^* = \mathbf{x}^T Q \mathbf{x}$, where $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ is a n-vector of binary (Boolean) variables and Q is a symmetric $n \times n$ matrix. Formally, QUBO problems are of the form:

$$x^* = \min_{\mathbf{x}} \sum_{i \geq j} x_i Q_{(i,j)} x_j, \text{ where } x_i \in \{0, 1\}.$$

The first step to converting the current binary linear IP formulation to QUBO is to use a "standard form", where all inequalities are replaced with equalities by introducing slack variables [12].

The next step is to build an equivalent QUBO of the IP formulation and add rules to force all linear equation constraints to be satisfied when assigning 0/1 to the binary variables. Consider a linear equality constraint C_k of the form $\sum_{i=1}^{n} c_{(k,i)} x_i = d_k$ for $x_i \in \{0, 1\}$ with fixed integer constants $c_{(k,i)}$ and d_k. This equation is satisfied if and only if $\sum_{i=1}^{n} c_{(k,i)} x_i - d_k = 0$, or equivalently, if $\langle \mathbf{c}_k, \mathbf{x} \rangle - d_k = 0$, where $\mathbf{c}_k = (c_{(k,i)}, c_{(k,2)}, \ldots, c_{(k,n)})$ and $\langle \mathbf{c}_k, \mathbf{x} \rangle$ is the product of the vectors $\mathbf{c}_k$ and $\mathbf{x}$. If $\langle \mathbf{c}_k, \mathbf{x} \rangle - d_k$ is not zero we need to have a penalty greater than the maximum feasible value of t, which is n. Thus, we can create the following QUBO that is equivalent to the IP formulation of the Broadcast Problem:

$$x^* = \min_{\mathbf{x}} \left(t + n \cdot \sum_k \left(\langle \mathbf{c}_k, \mathbf{x} \rangle - d_k \right)^2 \right), \text{ where } x_i \in \{0, 1\}.$$

Note first that the variable t is obtained from the variables used in equation (1) of Section 2 and is added to the other QUBO entries in Q from the set of linear constraints C_k. The QUBO constants for the binary variables representing t will be powers of 2, as given by equation (7). Second, any term d_k^2 in the square terms of $(\langle \mathbf{c}_k, \mathbf{x} \rangle - d_k)^2$ which does not involve a variable x_i can be ignored since those additive terms are independent of any assignment of variables (i.e. we have a fixed additive offset to the objective solution). Third, since variables x_i are binary, we have $x_i^2 = x_i$ and the constants for those terms are included in the main diagonal entries of Q.

Our program to convert an arbitrary broadcast problem instance to QUBO is listed in Appendix E.

5.1 Q_3 Example

In this section we summarize the quantum solution phases for Q_3. In the first phase (IP formulation) we get 33 integer variables (the variable t, eight variables of the type v_i, and 24 variables of the type $b_{i,j}$) and 65 quadratic constraints. The conversion to the binary formulation results in 51 binary variables as we need three binary variables for each of the previous integer variables $t, v_0, \ldots, v_7$; e.g. $51 = 33 + 2 \cdot 9$. The next conversion (Section 4) produces 447 binary variables and 851 linear constraints. Finally, the conversion to QUBO generated 999 slack variables, so in total 1446 binary variables: they represent the number of logical qubits for our QUBO formulation.

To be able to solve this QUBO problem on D-Wave we need one more step (see [2]) which will be illustrated in the next section with a feasible example for D-Wave Two.

6 K_2 Example

As a concrete example, we present both the final IP formulation (see Table 1) and QUBO matrix Q (see Table 2) for the broadcast problem for the graph K_2 of one edge. The total number of binary variables is 22 (which 13 of them are slack variables) and the QUBO offset is 12. When run on the D-Wave simulator (without embedding onto the hardware, which has limited qubit connections) we get the following expected result:

```
answer={
'energies': [-11.0],
'solutions': [[1,0,0,1,1,0,0,1,0,0,1,1,0,0,0,0,1,0,1,0,0,0]]
}
```

When we add the offset 12 to the minimum energy state we get our expected broadcast time of 1. We can also see that variable $t = x_1 = 1$ and $b_{0,1} = x_7 = 1$ and $b_{1,0} = x_8 = 0$, which indicates a valid broadcast tree from the obtained solution x^*.

To actually run this QUBO instance on the D-Wave machine we need to find a minor-containment embedding on the actual physical qubit hardware (a Chimera graph [2]). One valid heuristic is to map each logical qubit to a path of physical qubits. One such example is given below where our 22 logical qubits, labeled 0 to 21, becomes 50 active hardware qubits.

```
'embedding=': [ 0=[224, 226, 228], 1=[230], 2=[276, 283, 284, 288, 292],
3=[290], 4=[227, 291, 348, 355, 356, 357], 5=[229], 6=[336, 338, 341, 347, 349],
7=[293, 297, 301, 361], 8=[294, 296, 302], 9=[289], 10=[300], 11=[298, 362, 365],
12=[359, 367], 13=[364], 14=[345, 351], 15=[344], 16=[346],
17=[275, 277, 281, 285, 339], 18=[272], 19=[274], 20=[303], 21=[343] ]
```

This best energy solution of -11 is also obtained when we run this on an actual D-Wave machine. This optimal answer occurs about 33% of the time for our trials of about 1000 runs.

Table 1: Final Binary Integer Program for broadcasting in K_2.

Integer Program Constraints	Comments
$x_0 + x_1 = 1$	x_0 is objective variable t and x_1 is a slack variable
$-x_0 + x_2 + x_3 = 0$	x_2 is vertex variable v_0; equation (2)
$-x_0 + x_4 + x_5 = 0$	x_4 is vertex variable v_1; equation (2)
$x_6 = 0$	x_6 is broadcast variable $b_{1,0}$; equation (3)
$x_7 = 1$	x_7 is broadcast variable $b_{0,1}$; equation (4)
$x_2 + x_7 - x_8 + x_9 = 1$	x_8 is for product $b_{0,1}v_0$ with equation (8)
$-x_2 - x_7 + 2x_8 + x_{10} = 0$	equation (5) with equation (9)
$x_4 + x_7 - x_{11} + x_{12} = 1$	x_{11} is for product $b_{0,1}v_1$
$-x_4 - x_7 + 2x_{11} + x_{13} = 0$	equation (5)
$x_4 + x_6 - x_{14} + x_{15} = 1$	x_{14} is for product $b_{1,0}v_1$
$-x_4 - x_6 + 2x_{14} + x_{16} = 0$	equation (5)
$x_2 + x_6 - x_{17} + x_{18} = 1$	x_{17} is for product $b_{1,0}v_0$
$-x_2 - x_6 + 2x_{17} + x_{19} = 0$	equation (5)
$x_7 + x_8 - x_{11} + x_{20} = 0$	equation (6)
$x_6 + x_{14} - x_{17} + x_{21} = 0$	equation (6)

7 Experimental Results

We have produced QUBO representations of the Broadcast Problem for small common graphs using our IP formulation procedure. We summarize these in Tables 3 and 4 for known small families and known special graphs. These actual graphs maybe obtained from Sage [11]

Table 2: Final QUBO matrix Q for broadcasting in K_2.

3	4	-4	-4	-4	-4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	**-2**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
-4	0	**2**	4	0	0	8	8	-12	4	-4	0	0	0	0	0	0	-12	4	-4	0	0
-4	0	4	**2**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
-4	0	0	0	**2**	4	8	8	0	0	0	-12	4	-4	-12	4	-4	0	0	0	0	0
-4	0	0	0	4	**2**	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	8	0	8	0	**4**	0	0	0	0	0	0	0	-8	4	-4	-16	4	-4	0	4
0	0	8	0	8	0	0	**0**	-8	4	-4	-16	4	-4	0	0	0	0	0	0	4	0
0	0	-12	0	0	0	0	-8	**16**	-4	8	-4	0	0	0	0	0	0	0	0	4	0
0	0	4	0	0	0	0	4	-4	**-2**	0	0	0	0	0	0	0	0	0	0	0	0
0	0	-4	0	0	0	0	-4	8	0	**2**	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	-12	0	0	-16	-4	0	0	**16**	-4	8	0	0	0	0	0	0	-4	0
0	0	0	0	4	0	0	4	0	0	0	-4	**-2**	0	0	0	0	0	0	0	0	0
0	0	0	0	-4	0	0	-4	0	0	0	8	0	**2**	0	0	0	0	0	0	0	0
0	0	0	0	-12	0	-8	0	0	0	0	0	0	0	**16**	-4	8	-4	0	0	0	4
0	0	0	0	4	0	4	0	0	0	0	0	0	0	-4	**-2**	0	0	0	0	0	0
0	0	0	0	-4	0	-4	0	0	0	0	0	0	0	8	0	**2**	0	0	0	0	0
0	0	-12	0	0	0	-16	0	0	0	0	0	0	0	-4	0	0	**16**	-4	8	0	-4
0	0	4	0	0	0	4	0	0	0	0	0	0	0	0	0	0	-4	**-2**	0	0	0
0	0	-4	0	0	0	-4	0	0	0	0	0	0	0	0	0	0	8	0	**2**	0	0
0	0	0	0	0	0	0	4	4	0	0	-4	0	0	0	0	0	0	0	0	**2**	0
0	0	0	0	0	0	4	0	0	0	0	0	0	0	4	0	0	-4	0	0	0	**2**

using the script listed in Appendix F. Recall that for non-symmetric graphs we initiate the broadcast at vertex labeled 0, using the vertex labels given by Sage's adjacency lists. In these tables, columns 2 and 3 (Integer Variables and Quadratic Constraints) indicate the size of the IP formulation, as given in Section 2. Next, columns 4 and 5 (Binary Variables and Binary Constraints) indicate the size of the IP formulation, as given in Section 4. Finally, columns 6–8 (Slack Variables, Logical Qubits and Chimera/Physical Qubits) indicate the size of the final QUBO representation size, as given in Section 5. Observe that the number of logical qubits equals the number of binary variables plus the number of slack variables.

8 Conclusions

In this paper we have shown the process of converting a well-known combinatorial optimization problem, the Broadcast Network Problem, to a QUBO form that can be solved on a quantum computer like the D-Wave Two. Our procedure of using an integer programming formulation (e.g. standard polynomial-time reduction) can be easily applied to other hard problems. However, this straightforward approach does require a large number of qubits for relatively small input graph instances. Future work is required to reduce this overhead. One area of study is to exploit the problem's characteristics for a possible direct encoding into QUBO form. Also substantial work is needed to reduce the blowup in embedding the logical qubits to physical qubits, which is required for the target machine's architecture.

Table 3: Number of qubits required for some small graphs families.

Graph	Order	Size	Integer Variables	Quadratic Constraints	Binary Variables	Binary Constraints	Slack Variables	Logical Qubits	Chimera Qubits
C3	3	3	10	16	50	86	96	146	394
C4	4	4	13	21	74	131	146	220	662
C5	5	5	16	26	178	324	366	544	3258
C6	6	6	19	31	240	443	495	735	4164
C7	7	7	22	36	311	580	642	953	
C8	8	8	25	41	391	735	807	1198	
C9	9	9	28	46	778	1484	1608	2386	
C10	10	10	31	51	944	1809	1948	2892	
C11	11	11	34	56	1126	2166	2320	3446	
C12	12	12	37	61	1324	2555	2724	4048	
Grid2x3	6	7	21	37	254	472	543	797	4306
Grid3x3	9	12	34	65	832	1597	1816	2648	
Grid3x4	12	17	47	93	1414	2745	3084	4498	
Grid4x4	16	24	65	133	2420	4737	5252	7672	
Grid4x5	20	31	83	173	5537	10909	11815	17352	
K2	2	1	5	7	9	15	13	22	47
K3	3	3	10	16	50	86	96	146	394
K4	4	6	17	33	94	171	202	296	1378
K5	5	10	26	61	248	469	606	854	7973
K6	6	15	37	103	366	713	981	1347	
K7	7	21	50	162	507	1014	1482	1989	
K8	8	28	65	241	671	1375	2127	2798	
K9	9	36	82	343	1264	2591	4200	5464	
K10	10	45	101	471	1574	3279	5588	7162	
K2x1=P2	3	2	8	12	36	59	64	100	170
K1x2=S2	3	2	8	12	40	68	76	116	238
K2x2=C4	4	4	13	21	74	131	146	220	662
K2x3	5	6	18	32	192	353	414	606	4823
K3x3	6	9	25	49	282	529	633	915	
K3x4	7	12	32	69	381	727	894	1275	
K4x4	8	16	41	97	503	973	1227	1730	
K4x5	9	20	50	129	976	1906	2432	3408	
K5x5	10	25	61	171	1214	2391	3124	4338	
K5x6	11	30	72	118	1468	2914	3896	5364	
K6x6	12	36	85	277	1756	3511	4804	6560	
S2=K1x2	3	2	8	12	40	68	76	116	238
S3	4	3	11	18	64	114	130	194	505
S4	5	4	14	25	164	301	354	518	3711
S5	6	5	17	33	226	423	501	727	5120
S6	7	6	20	42	297	564	672	969	
S7	8	7	23	52	377	724	867	1244	
S8	9	8	26	63	760	1471	1736	2496	
S9	10	9	29	75	926	1803	2132	3058	
S10	11	10	32	88	1108	2168	2568	3676	

Table 4: Number of qubits required for hypercubes and some other small known graphs.

Graph	Order	Size	Integer Variables	Quadratic Constraints	Binary Variables	Binary Constraints	Slack Variables	Logical Qubits	Chimera Qubits
Q1=K2	2	1	5	7	9	15	13	22	47
Q2=C4	4	4	13	21	74	131	146	220	662
Q3	8	12	33	65	447	851	999	1446	
Q4	16	32	81	193	2564	5045	5860	8424	
BidiakisCube	12	18	49	97	1432	2779	3124	4556	
Bull	5	5	16	28	178	324	366	544	3523
Butterfly	5	6	18	33	192	353	414	606	5927
Chvatal	12	24	61	145	1540	3013	3604	5144	
Clebsch	16	40	97	273	2708	5373	6628	9336	
Diamond	4	5	15	27	84	151	174	258	742
Dinneen	9	21	52	142	994	1950	2552	3546	
Dodecahedral	20	30	81	161	5515	10855	11645	17160	
Durer	12	18	49	97	1432	2779	3124	4556	
Errera	17	45	108	320	4480	8900	10890	15370	
Frucht	12	18	49	97	1432	2779	3124	4556	
GoldnerHarary	11	27	66	209	1414	2814	3792	5206	
Grotzsch	11	20	52	118	1288	2508	2968	4256	
Heawood	14	21	57	113	1894	3691	4100	5994	
Herschel	11	18	48	101	1252	2429	2800	4052	
Hexahedral	8	12	33	65	447	851	999	1446	
Hoffman	16	32	81	193	2564	5045	5860	8424	
House	5	6	18	32	192	353	414	606	4176
Icosahedral	12	30	73	205	1648	3257	4164	5812	
Krackhardt	10	18	47	114	1088	2116	2548	3636	
Octahedral	6	12	31	73	324	619	795	1119	
Pappus	18	27	73	145	4514	8869	9575	14089	
Petersen	10	15	41	81	1034	1995	2276	3310	
Poussin	15	39	94	276	2446	4863	6152	8598	
Robertson	19	38	96	229	5211	10287	11570	16781	
Shrikhande	16	48	113	369	2852	5715	7508	10360	
Sousselier	16	27	71	154	2474	4849	5452	7926	
Tietze	12	18	49	97	1432	2779	3124	4556	
Wagner	8	12	33	65	447	851	999	1446	

Acknowledgment

This work was supported in part by the Quantum Computing Research Initiatives at Lockheed Martin.

References

[1] D-S. Chen, R. G. Batson and Y. Dang. *Applied Integer Programming: Modeling and Solution*, John Wiley and Sons, 2010.

[2] D-Wave. Programming with QUBOs. Release 1.5.1-beta4, 09-1002A-B. D-Wave Systems Inc., 2013.

[3] M. J. Dinneen. The complexity of broadcasting in bounded-degree networks. Combinatorics report LACES-[05C-94-31], Los Alamos National Laboratory, http://arxiv.org/abs/math/9411222, 1994.

[4] M. J. Dinneen, M. R. Fellows and V. Faber. Algebraic constructions of efficient broadcast networks, *Applied Algebra, Algebraic Algorithms and Error-correcting Codes* (New Orleans, LA, 1991), Lecture Notes in Computer Science, vol. 539, Springer, Berlin, pp. 152–158, 1991.

[5] A. Farley, S. Hedetniemi, S. Mitchell and A. Proskurowski. Minimum broadcast graphs, *Discrete Mathematics* 25, 189–193, 1979.

[6] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness,* W.H. Freeman and Company, 1979.

[7] A. A. Hagberg, D. A. Schult and P. J. Swart. Exploring network structure, dynamics, and function using NetworkX", in *Proceedings of the 7th Python in Science Conference (SciPy2008)*, Gäel Varoquaux, Travis Vaught, and Jarrod Millman (Eds.), (Pasadena, CA USA), pp. 11–15, Aug 2008.

[8] S. M. Hedetniemi, S. T. Hedetniemi and A. L. Liestman. A survey of gossiping and broadcasting in communication networks, *Networks*, 18, 319–349, 1998.

[9] M. Khosravani. *Searching for Optimal Caterpillars in General and Bounded Treewidth Graphs*. PhD dissertation, University of Auckland, Auckland, New Zealand, 2011.

[10] C. H. Papadimitriou and K. Steiglitz. *Combinatorial Optimization: Algorithms and Complexity*, Mineola, NY, Dover, 1998.

[11] W. A. Stein et al. Sage Mathematics Software (Version 6.2), The Sage Development Team, 2014, http://www.sagemath.org.

[12] Wikipedia. Integer Programming (accessed 30 October 2014). URL: http://en.wikipedia.org/wiki/Integer_programming.

A Quadradic Integer Programming Formulation for Broadcasting in Cube Q_3

```
x_0 <= 7                      # time t
x_1 <= x_0                    # vertices v0..v7 informed times <= t
x_2 <= x_0
x_3 <= x_0
x_4 <= x_0
x_5 <= x_0
x_6 <= x_0
x_7 <= x_0
x_8 <= x_0
x_9 + x_10 + x_11 <= 0        # originator has no parent
x_12 + x_13 + x_14 <= 1       # other vertices have one parent
x_15 + x_16 + x_17 <= 1
x_18 + x_19 + x_20 <= 1
x_21 + x_22 + x_23 <= 1
x_24 + x_25 + x_26 <= 1
x_27 + x_28 + x_29 <= 1
x_30 + x_31 + x_32 <= 1
x_12 + x_12 * x_1 − x_12 * x_2 <= 0  # parent time less than child time
x_15 + x_15 * x_1 − x_15 * x_3 <= 0
x_21 + x_21 * x_1 − x_21 * x_5 <= 0
x_9 + x_9 * x_2 − x_9 * x_1 <= 0
x_18 + x_18 * x_2 − x_18 * x_4 <= 0
x_24 + x_24 * x_2 − x_24 * x_6 <= 0
x_10 + x_10 * x_3 − x_10 * x_1 <= 0
x_19 + x_19 * x_3 − x_19 * x_4 <= 0
x_27 + x_27 * x_3 − x_27 * x_7 <= 0
x_13 + x_13 * x_4 − x_13 * x_2 <= 0
x_16 + x_16 * x_4 − x_16 * x_3 <= 0
x_30 + x_30 * x_4 − x_30 * x_8 <= 0
x_11 + x_11 * x_5 − x_11 * x_1 <= 0
x_25 + x_25 * x_5 − x_25 * x_6 <= 0
x_28 + x_28 * x_5 − x_28 * x_7 <= 0
x_14 + x_14 * x_6 − x_14 * x_2 <= 0
x_22 + x_22 * x_6 − x_22 * x_5 <= 0
x_31 + x_31 * x_6 − x_31 * x_8 <= 0
x_17 + x_17 * x_7 − x_17 * x_3 <= 0
x_23 + x_23 * x_7 − x_23 * x_5 <= 0
x_32 + x_32 * x_7 − x_32 * x_8 <= 0
x_20 + x_20 * x_8 − x_20 * x_4 <= 0
x_26 + x_26 * x_8 − x_26 * x_6 <= 0
x_29 + x_29 * x_8 − x_29 * x_7 <= 0
x_12 + x_15 − sqr( x_2 − x_3 ) <= 1  # each child with different times
x_12 + x_21 − sqr( x_2 − x_5 ) <= 1
x_15 + x_21 − sqr( x_3 − x_5 ) <= 1
x_9 + x_18 − sqr( x_1 − x_4 ) <= 1
x_9 + x_24 − sqr( x_1 − x_6 ) <= 1
x_18 + x_24 − sqr( x_4 − x_6 ) <= 1
x_10 + x_19 − sqr( x_1 − x_4 ) <= 1
x_10 + x_27 − sqr( x_1 − x_7 ) <= 1
```

```
x_19 + x_27 - sqr( x_4 - x_7 ) <= 1
x_13 + x_16 - sqr( x_2 - x_3 ) <= 1
x_13 + x_30 - sqr( x_2 - x_8 ) <= 1
x_16 + x_30 - sqr( x_3 - x_8 ) <= 1
x_11 + x_25 - sqr( x_1 - x_6 ) <= 1
x_11 + x_28 - sqr( x_1 - x_7 ) <= 1
x_25 + x_28 - sqr( x_6 - x_7 ) <= 1
x_14 + x_22 - sqr( x_2 - x_5 ) <= 1
x_14 + x_31 - sqr( x_2 - x_8 ) <= 1
x_22 + x_31 - sqr( x_5 - x_8 ) <= 1
x_17 + x_23 - sqr( x_3 - x_5 ) <= 1
x_17 + x_32 - sqr( x_3 - x_8 ) <= 1
x_23 + x_32 - sqr( x_5 - x_8 ) <= 1
x_20 + x_26 - sqr( x_4 - x_6 ) <= 1
x_20 + x_29 - sqr( x_4 - x_7 ) <= 1
x_26 + x_29 - sqr( x_6 - x_7 ) <= 1
```

listings/Q3v1.out

B Binary Quadradic Integer Programming Formulation for Broadcasting in Cube Q_3

```
x_0 + 2*x_1 + 4*x_2 <= 7
-1*x_0 + x_3 + -2*x_1 + 2*x_4 + -4*x_2 + 4*x_5 <= 0
-1*x_0 + x_6 + -2*x_1 + 2*x_7 + -4*x_2 + 4*x_8 <= 0
-1*x_0 + x_9 + -2*x_1 + 2*x_10 + -4*x_2 + 4*x_11 <= 0
-1*x_0 + x_12 + -2*x_1 + 2*x_13 + -4*x_2 + 4*x_14 <= 0
-1*x_0 + x_15 + -2*x_1 + 2*x_16 + -4*x_2 + 4*x_17 <= 0
-1*x_0 + x_18 + -2*x_1 + 2*x_19 + -4*x_2 + 4*x_20 <= 0
-1*x_0 + x_21 + -2*x_1 + 2*x_22 + -4*x_2 + 4*x_23 <= 0
-1*x_0 + x_24 + -2*x_1 + 2*x_25 + -4*x_2 + 4*x_26 <= 0
x_27 + x_28 + x_29 = 0
x_30 + x_31 + x_32 = 1
x_33 + x_34 + x_35 = 1
x_36 + x_37 + x_38 = 1
x_39 + x_40 + x_41 = 1
x_42 + x_43 + x_44 = 1
x_45 + x_46 + x_47 = 1
x_48 + x_49 + x_50 = 1
x_30 + x_30 * ( x_3 + 2*x_4 + 4*x_5 ) - x_30 * ( x_6 + 2*x_7 + 4*x_8 ) <= 0
x_33 + x_33 * ( x_3 + 2*x_4 + 4*x_5 ) - x_33 * ( x_9 + 2*x_10 + 4*x_11 ) <= 0
x_39 + x_39 * ( x_3 + 2*x_4 + 4*x_5 ) - x_39 * ( x_15 + 2*x_16 + 4*x_17 ) <= 0
x_27 + x_27 * ( x_6 + 2*x_7 + 4*x_8 ) - x_27 * ( x_3 + 2*x_4 + 4*x_5 ) <= 0
x_36 + x_36 * ( x_6 + 2*x_7 + 4*x_8 ) - x_36 * ( x_12 + 2*x_13 + 4*x_14 ) <= 0
x_42 + x_42 * ( x_6 + 2*x_7 + 4*x_8 ) - x_42 * ( x_18 + 2*x_19 + 4*x_20 ) <= 0
x_28 + x_28 * ( x_9 + 2*x_10 + 4*x_11 ) - x_28 * ( x_3 + 2*x_4 + 4*x_5 ) <= 0
x_37 + x_37 * ( x_9 + 2*x_10 + 4*x_11 ) - x_37 * ( x_12 + 2*x_13 + 4*x_14 ) <=
    0
x_45 + x_45 * ( x_9 + 2*x_10 + 4*x_11 ) - x_45 * ( x_21 + 2*x_22 + 4*x_23 ) <=
    0
```

```
x_31 + x_31 * ( x_12 + 2*x_13 + 4*x_14 ) - x_31 * ( x_6 + 2*x_7 + 4*x_8 ) <= 0
x_34 + x_34 * ( x_12 + 2*x_13 + 4*x_14 ) - x_34 * ( x_9 + 2*x_10 + 4*x_11 ) <=
    0
x_48 + x_48 * ( x_12 + 2*x_13 + 4*x_14 ) - x_48 * ( x_24 + 2*x_25 + 4*x_26 )
   <= 0
x_29 + x_29 * ( x_15 + 2*x_16 + 4*x_17 ) - x_29 * ( x_3 + 2*x_4 + 4*x_5 ) <= 0
x_43 + x_43 * ( x_15 + 2*x_16 + 4*x_17 ) - x_43 * ( x_18 + 2*x_19 + 4*x_20 )
   <= 0
x_46 + x_46 * ( x_15 + 2*x_16 + 4*x_17 ) - x_46 * ( x_21 + 2*x_22 + 4*x_23 )
   <= 0
x_32 + x_32 * ( x_18 + 2*x_19 + 4*x_20 ) - x_32 * ( x_6 + 2*x_7 + 4*x_8 ) <= 0
x_40 + x_40 * ( x_18 + 2*x_19 + 4*x_20 ) - x_40 * ( x_15 + 2*x_16 + 4*x_17 )
   <= 0
x_49 + x_49 * ( x_18 + 2*x_19 + 4*x_20 ) - x_49 * ( x_24 + 2*x_25 + 4*x_26 )
   <= 0
x_35 + x_35 * ( x_21 + 2*x_22 + 4*x_23 ) - x_35 * ( x_9 + 2*x_10 + 4*x_11 ) <=
    0
x_41 + x_41 * ( x_21 + 2*x_22 + 4*x_23 ) - x_41 * ( x_15 + 2*x_16 + 4*x_17 )
   <= 0
x_50 + x_50 * ( x_21 + 2*x_22 + 4*x_23 ) - x_50 * ( x_24 + 2*x_25 + 4*x_26 )
   <= 0
x_38 + x_38 * ( x_24 + 2*x_25 + 4*x_26 ) - x_38 * ( x_12 + 2*x_13 + 4*x_14 )
   <= 0
x_44 + x_44 * ( x_24 + 2*x_25 + 4*x_26 ) - x_44 * ( x_18 + 2*x_19 + 4*x_20 )
   <= 0
x_47 + x_47 * ( x_24 + 2*x_25 + 4*x_26 ) - x_47 * ( x_21 + 2*x_22 + 4*x_23 )
   <= 0
x_30 + x_33 - sqr( x_6 - x_9 + 2*x_7 - 2*x_10 + 4*x_8 - 4*x_11 ) <= 1
x_30 + x_39 - sqr( x_6 - x_15 + 2*x_7 - 2*x_16 + 4*x_8 - 4*x_17 ) <= 1
x_33 + x_39 - sqr( x_9 - x_15 + 2*x_10 - 2*x_16 + 4*x_11 - 4*x_17 ) <= 1
x_27 + x_36 - sqr( x_3 - x_12 + 2*x_4 - 2*x_13 + 4*x_5 - 4*x_14 ) <= 1
x_27 + x_42 - sqr( x_3 - x_18 + 2*x_4 - 2*x_19 + 4*x_5 - 4*x_20 ) <= 1
x_36 + x_42 - sqr( x_12 - x_18 + 2*x_13 - 2*x_19 + 4*x_14 - 4*x_20 ) <= 1
x_28 + x_37 - sqr( x_3 - x_12 + 2*x_4 - 2*x_13 + 4*x_5 - 4*x_14 ) <= 1
x_28 + x_45 - sqr( x_3 - x_21 + 2*x_4 - 2*x_22 + 4*x_5 - 4*x_23 ) <= 1
x_37 + x_45 - sqr( x_12 - x_21 + 2*x_13 - 2*x_22 + 4*x_14 - 4*x_23 ) <= 1
x_31 + x_34 - sqr( x_6 - x_9 + 2*x_7 - 2*x_10 + 4*x_8 - 4*x_11 ) <= 1
x_31 + x_48 - sqr( x_6 - x_24 + 2*x_7 - 2*x_25 + 4*x_8 - 4*x_26 ) <= 1
x_34 + x_48 - sqr( x_9 - x_24 + 2*x_10 - 2*x_25 + 4*x_11 - 4*x_26 ) <= 1
x_29 + x_43 - sqr( x_3 - x_18 + 2*x_4 - 2*x_19 + 4*x_5 - 4*x_20 ) <= 1
x_29 + x_46 - sqr( x_3 - x_21 + 2*x_4 - 2*x_22 + 4*x_5 - 4*x_23 ) <= 1
x_43 + x_46 - sqr( x_18 - x_21 + 2*x_19 - 2*x_22 + 4*x_20 - 4*x_23 ) <= 1
x_32 + x_40 - sqr( x_6 - x_15 + 2*x_7 - 2*x_16 + 4*x_8 - 4*x_17 ) <= 1
x_32 + x_49 - sqr( x_6 - x_24 + 2*x_7 - 2*x_25 + 4*x_8 - 4*x_26 ) <= 1
x_40 + x_49 - sqr( x_15 - x_24 + 2*x_16 - 2*x_25 + 4*x_17 - 4*x_26 ) <= 1
x_35 + x_41 - sqr( x_9 - x_15 + 2*x_10 - 2*x_16 + 4*x_11 - 4*x_17 ) <= 1
x_35 + x_50 - sqr( x_9 - x_24 + 2*x_10 - 2*x_25 + 4*x_11 - 4*x_26 ) <= 1
x_41 + x_50 - sqr( x_15 - x_24 + 2*x_16 - 2*x_25 + 4*x_17 - 4*x_26 ) <= 1
x_38 + x_44 - sqr( x_12 - x_18 + 2*x_13 - 2*x_19 + 4*x_14 - 4*x_20 ) <= 1
x_38 + x_47 - sqr( x_12 - x_21 + 2*x_13 - 2*x_22 + 4*x_14 - 4*x_23 ) <= 1
x_44 + x_47 - sqr( x_18 - x_21 + 2*x_19 - 2*x_22 + 4*x_20 - 4*x_23 ) <= 1
```

listings/Q3v2.out

C Python Program to Generate the Linear Binary Programming Formulation

```
# broadcast time of graph: IP with binary variables and linear constraints

import networkx as nx
from sage.all import *

def read_graph():
    G=nx.Graph()
    n=int(sys.stdin.readline().strip())
    for u in range(n):
        neighbors=sys.stdin.readline().split()
        for v in neighbors: G.add_edge(u,int(v))
    return G

if len(sys.argv)>1: orig=int(sys.argv[1])
else:               orig=0

G=read_graph()
n=G.order()
L=int(math.log(2*n-1,2))

p=MixedIntegerLinearProgram(solver="Coin",maximization=False)

t=p.new_variable(binary=True)
p.add_constraint(sum([2**j*t[j] for j in range(L)]) <= n-1)

v=p.new_variable(binary=True)
for i in range(n):
    p.add_constraint(sum([2**j*(v[i,j]-t[j]) for j in range(L)]), max=0)

b=p.new_variable(binary=True)
for i in range(n):
    if i==orig:
        p.add_constraint(sum([b[j,i] for j in G[i]]),min=0,max=0)
    else:
        p.add_constraint(sum([b[j,i] for j in G[i]]), min=1, max=1)

z1=p.new_variable(binary=True) # b[i,j] * v[i,c]
z2=p.new_variable(binary=True) # b[i,j] * v[j,c]
for i in range(n):
    for j in G[i]:
        for c in range(L):
            p.add_constraint(b[i,j]+v[i,c]-z1[i,j,c],max=1)
            p.add_constraint(2*z1[i,j,c]-(b[i,j]+v[i,c]),max=0)
            p.add_constraint(b[i,j]+v[j,c]-z2[i,j,c],max=1)
            p.add_constraint(2*z2[i,j,c]-(b[i,j]+v[j,c]),max=0)
for i in range(n):
    for j in G[i]:
        p.add_constraint(b[i,j]+sum([2**k*z1[i,j,k]-2**k*z2[i,j,k] \
            for k in range(L)]),max=0)
```

```
z3=p.new_variable(binary=True)# v[i,c1]*v[j,c2] for i<j (VxV) or i=j (V)
for i in range(n):
    if i==orig: continue
    for j in range(i,n): # diff vertex case # if j<i: continue
        if j==orig: continue
        S=set(G[i]); S.intersection(set(G[j])) # must have common parent
        if len(S)==0: continue                  # or neigbor of degree >= 2
        if i==j and max([G.degree(k) for k in G[i]])<2: continue
        for c1 in range(L):
            for c2 in range(L):
                p.add_constraint(v[i,c1]+v[j,c2]-z3[i,c1,j,c2],max=1)
                p.add_constraint(2*z3[i,c1,j,c2]-(v[i,c1]+v[j,c2]),max=0)

def p3(t1,k1,t2,k2):
    if t1<=t2: return z3[t1,k1,t2,k2]
    else:      return z3[t2,k2,t1,k1]

for i in range(n):
    for jk in Combinations(sorted(G[i]),2).list():
        if orig in jk: continue # we know no b[*,orig] are true
        assert jk[0]<jk[1]
        tmp=[]
        for k in range(L):
          tmp.append((jk[0],k,1)); tmp.append((jk[1],k,-1))
        prod=[(s1*s2)*2**(k1+k2)*p3(t1,k1,t2,k2) \
          for (t1,k1,s1) in tmp for (t2,k2,s2) in tmp]
        p.add_constraint(b[i,jk[0]]+b[i,jk[1]]-sum(prod),max=1)

p.set_objective(sum([2**j*t[j] for j in range(L)]))
p.show()

def getT(bits):
    T=0
    for (key,val) in bits: T+=2**key*val
    return int(T+0.000001)
def getV(bits):
    T=[0.000001]*n
    for ((x,key),val) in bits: T[x]+=2**key*val
    return map(int,T)

try:
    p.solve(log=0)
except sage.numerical.mip.MIPSolverException as e:
    print e
else:
    print "broadcast time:",getT(p.get_values(t).items())
    print "vertices times:",getV(p.get_values(v).items())
    print "broadcast tree:",
    for (key,val) in p.get_values(b).items():
      if val>0: print key,
    print
```

listings/IPbroadcast.py

D Binary Linear Integer Programming Formulation for Broadcasting in Cube Q_3

```
Minimization:
  x_0 + 2.0 x_1 + 4.0 x_2
Constraints:
  x_0 + 2.0 x_1 + 4.0 x_2 <= 7.0
  - x_0 - 2.0 x_1 - 4.0 x_2 + x_3 + 2.0 x_4 + 4.0 x_5 <= 0.0
  - x_0 - 2.0 x_1 - 4.0 x_2 + x_6 + 2.0 x_7 + 4.0 x_8 <= 0.0
  - x_0 - 2.0 x_1 - 4.0 x_2 + x_9 + 2.0 x_10 + 4.0 x_11 <= 0.0
  - x_0 - 2.0 x_1 - 4.0 x_2 + x_12 + 2.0 x_13 + 4.0 x_14 <= 0.0
  - x_0 - 2.0 x_1 - 4.0 x_2 + x_15 + 2.0 x_16 + 4.0 x_17 <= 0.0
  - x_0 - 2.0 x_1 - 4.0 x_2 + x_18 + 2.0 x_19 + 4.0 x_20 <= 0.0
  - x_0 - 2.0 x_1 - 4.0 x_2 + x_21 + 2.0 x_22 + 4.0 x_23 <= 0.0
  - x_0 - 2.0 x_1 - 4.0 x_2 + x_24 + 2.0 x_25 + 4.0 x_26 <= 0.0
  0.0 <= x_27 + x_28 + x_29 <= 0.0
  1.0 <= x_30 + x_31 + x_32 <= 1.0
  1.0 <= x_33 + x_34 + x_35 <= 1.0
  1.0 <= x_36 + x_37 + x_38 <= 1.0
  1.0 <= x_39 + x_40 + x_41 <= 1.0
  1.0 <= x_42 + x_43 + x_44 <= 1.0
  1.0 <= x_45 + x_46 + x_47 <= 1.0
  1.0 <= x_48 + x_49 + x_50 <= 1.0
  x_3 + x_30 - x_51 <= 1.0
  - x_3 - x_30 + 2.0 x_51 <= 0.0
  x_6 + x_30 - x_52 <= 1.0
  - x_6 - x_30 + 2.0 x_52 <= 0.0
  x_4 + x_30 - x_53 <= 1.0
  - x_4 - x_30 + 2.0 x_53 <= 0.0
  x_7 + x_30 - x_54 <= 1.0
  - x_7 - x_30 + 2.0 x_54 <= 0.0
  x_5 + x_30 - x_55 <= 1.0
  - x_5 - x_30 + 2.0 x_55 <= 0.0
  x_8 + x_30 - x_56 <= 1.0
  - x_8 - x_30 + 2.0 x_56 <= 0.0
  x_3 + x_33 - x_57 <= 1.0
  - x_3 - x_33 + 2.0 x_57 <= 0.0
  x_9 + x_33 - x_58 <= 1.0
  - x_9 - x_33 + 2.0 x_58 <= 0.0
  x_4 + x_33 - x_59 <= 1.0
  - x_4 - x_33 + 2.0 x_59 <= 0.0
  x_10 + x_33 - x_60 <= 1.0
  - x_10 - x_33 + 2.0 x_60 <= 0.0
  x_5 + x_33 - x_61 <= 1.0
  - x_5 - x_33 + 2.0 x_61 <= 0.0
  x_11 + x_33 - x_62 <= 1.0
  - x_11 - x_33 + 2.0 x_62 <= 0.0
  x_3 + x_39 - x_63 <= 1.0
  - x_3 - x_39 + 2.0 x_63 <= 0.0
  x_15 + x_39 - x_64 <= 1.0
  - x_15 - x_39 + 2.0 x_64 <= 0.0
  x_4 + x_39 - x_65 <= 1.0
```

```
- x_4 - x_39 + 2.0 x_65 <= 0.0
x_16 + x_39 - x_66 <= 1.0
- x_16 - x_39 + 2.0 x_66 <= 0.0
x_5 + x_39 - x_67 <= 1.0
- x_5 - x_39 + 2.0 x_67 <= 0.0
x_17 + x_39 - x_68 <= 1.0
- x_17 - x_39 + 2.0 x_68 <= 0.0
x_6 + x_27 - x_69 <= 1.0
- x_6 - x_27 + 2.0 x_69 <= 0.0
x_3 + x_27 - x_70 <= 1.0
- x_3 - x_27 + 2.0 x_70 <= 0.0
x_7 + x_27 - x_71 <= 1.0
- x_7 - x_27 + 2.0 x_71 <= 0.0
x_4 + x_27 - x_72 <= 1.0
- x_4 - x_27 + 2.0 x_72 <= 0.0
x_8 + x_27 - x_73 <= 1.0
- x_8 - x_27 + 2.0 x_73 <= 0.0
x_5 + x_27 - x_74 <= 1.0
- x_5 - x_27 + 2.0 x_74 <= 0.0
x_6 + x_36 - x_75 <= 1.0
- x_6 - x_36 + 2.0 x_75 <= 0.0
x_12 + x_36 - x_76 <= 1.0
- x_12 - x_36 + 2.0 x_76 <= 0.0
x_7 + x_36 - x_77 <= 1.0
- x_7 - x_36 + 2.0 x_77 <= 0.0
x_13 + x_36 - x_78 <= 1.0
- x_13 - x_36 + 2.0 x_78 <= 0.0
x_8 + x_36 - x_79 <= 1.0
- x_8 - x_36 + 2.0 x_79 <= 0.0
x_14 + x_36 - x_80 <= 1.0
- x_14 - x_36 + 2.0 x_80 <= 0.0
x_6 + x_42 - x_81 <= 1.0
- x_6 - x_42 + 2.0 x_81 <= 0.0
x_18 + x_42 - x_82 <= 1.0
- x_18 - x_42 + 2.0 x_82 <= 0.0
x_7 + x_42 - x_83 <= 1.0
- x_7 - x_42 + 2.0 x_83 <= 0.0
x_19 + x_42 - x_84 <= 1.0
- x_19 - x_42 + 2.0 x_84 <= 0.0
x_8 + x_42 - x_85 <= 1.0
- x_8 - x_42 + 2.0 x_85 <= 0.0
x_20 + x_42 - x_86 <= 1.0
- x_20 - x_42 + 2.0 x_86 <= 0.0
x_9 + x_28 - x_87 <= 1.0
- x_9 - x_28 + 2.0 x_87 <= 0.0
x_3 + x_28 - x_88 <= 1.0
- x_3 - x_28 + 2.0 x_88 <= 0.0
x_10 + x_28 - x_89 <= 1.0
- x_10 - x_28 + 2.0 x_89 <= 0.0
x_4 + x_28 - x_90 <= 1.0
- x_4 - x_28 + 2.0 x_90 <= 0.0
x_11 + x_28 - x_91 <= 1.0
- x_11 - x_28 + 2.0 x_91 <= 0.0
x_5 + x_28 - x_92 <= 1.0
```

```
- x_5 - x_28 + 2.0 x_92 <= 0.0
x_9 + x_37 - x_93 <= 1.0
- x_9 - x_37 + 2.0 x_93 <= 0.0
x_12 + x_37 - x_94 <= 1.0
- x_12 - x_37 + 2.0 x_94 <= 0.0
x_10 + x_37 - x_95 <= 1.0
- x_10 - x_37 + 2.0 x_95 <= 0.0
x_13 + x_37 - x_96 <= 1.0
- x_13 - x_37 + 2.0 x_96 <= 0.0
x_11 + x_37 - x_97 <= 1.0
- x_11 - x_37 + 2.0 x_97 <= 0.0
x_14 + x_37 - x_98 <= 1.0
- x_14 - x_37 + 2.0 x_98 <= 0.0
x_9 + x_45 - x_99 <= 1.0
- x_9 - x_45 + 2.0 x_99 <= 0.0
x_21 + x_45 - x_100 <= 1.0
- x_21 - x_45 + 2.0 x_100 <= 0.0
x_10 + x_45 - x_101 <= 1.0
- x_10 - x_45 + 2.0 x_101 <= 0.0
x_22 + x_45 - x_102 <= 1.0
- x_22 - x_45 + 2.0 x_102 <= 0.0
x_11 + x_45 - x_103 <= 1.0
- x_11 - x_45 + 2.0 x_103 <= 0.0
x_23 + x_45 - x_104 <= 1.0
- x_23 - x_45 + 2.0 x_104 <= 0.0
x_12 + x_31 - x_105 <= 1.0
- x_12 - x_31 + 2.0 x_105 <= 0.0
x_6 + x_31 - x_106 <= 1.0
- x_6 - x_31 + 2.0 x_106 <= 0.0
x_13 + x_31 - x_107 <= 1.0
- x_13 - x_31 + 2.0 x_107 <= 0.0
x_7 + x_31 - x_108 <= 1.0
- x_7 - x_31 + 2.0 x_108 <= 0.0
x_14 + x_31 - x_109 <= 1.0
- x_14 - x_31 + 2.0 x_109 <= 0.0
x_8 + x_31 - x_110 <= 1.0
- x_8 - x_31 + 2.0 x_110 <= 0.0
x_12 + x_34 - x_111 <= 1.0
- x_12 - x_34 + 2.0 x_111 <= 0.0
x_9 + x_34 - x_112 <= 1.0
- x_9 - x_34 + 2.0 x_112 <= 0.0
x_13 + x_34 - x_113 <= 1.0
- x_13 - x_34 + 2.0 x_113 <= 0.0
x_10 + x_34 - x_114 <= 1.0
- x_10 - x_34 + 2.0 x_114 <= 0.0
x_14 + x_34 - x_115 <= 1.0
- x_14 - x_34 + 2.0 x_115 <= 0.0
x_11 + x_34 - x_116 <= 1.0
- x_11 - x_34 + 2.0 x_116 <= 0.0
x_12 + x_48 - x_117 <= 1.0
- x_12 - x_48 + 2.0 x_117 <= 0.0
x_24 + x_48 - x_118 <= 1.0
- x_24 - x_48 + 2.0 x_118 <= 0.0
x_13 + x_48 - x_119 <= 1.0
```

```
− x_13 − x_48 + 2.0 x_119 <= 0.0
x_25 + x_48 − x_120 <= 1.0
− x_25 − x_48 + 2.0 x_120 <= 0.0
x_14 + x_48 − x_121 <= 1.0
− x_14 − x_48 + 2.0 x_121 <= 0.0
x_26 + x_48 − x_122 <= 1.0
− x_26 − x_48 + 2.0 x_122 <= 0.0
x_15 + x_29 − x_123 <= 1.0
− x_15 − x_29 + 2.0 x_123 <= 0.0
x_3 + x_29 − x_124 <= 1.0
− x_3 − x_29 + 2.0 x_124 <= 0.0
x_16 + x_29 − x_125 <= 1.0
− x_16 − x_29 + 2.0 x_125 <= 0.0
x_4 + x_29 − x_126 <= 1.0
− x_4 − x_29 + 2.0 x_126 <= 0.0
x_17 + x_29 − x_127 <= 1.0
− x_17 − x_29 + 2.0 x_127 <= 0.0
x_5 + x_29 − x_128 <= 1.0
− x_5 − x_29 + 2.0 x_128 <= 0.0
x_15 + x_43 − x_129 <= 1.0
− x_15 − x_43 + 2.0 x_129 <= 0.0
x_18 + x_43 − x_130 <= 1.0
− x_18 − x_43 + 2.0 x_130 <= 0.0
x_16 + x_43 − x_131 <= 1.0
− x_16 − x_43 + 2.0 x_131 <= 0.0
x_19 + x_43 − x_132 <= 1.0
− x_19 − x_43 + 2.0 x_132 <= 0.0
x_17 + x_43 − x_133 <= 1.0
− x_17 − x_43 + 2.0 x_133 <= 0.0
x_20 + x_43 − x_134 <= 1.0
− x_20 − x_43 + 2.0 x_134 <= 0.0
x_15 + x_46 − x_135 <= 1.0
− x_15 − x_46 + 2.0 x_135 <= 0.0
x_21 + x_46 − x_136 <= 1.0
− x_21 − x_46 + 2.0 x_136 <= 0.0
x_16 + x_46 − x_137 <= 1.0
− x_16 − x_46 + 2.0 x_137 <= 0.0
x_22 + x_46 − x_138 <= 1.0
− x_22 − x_46 + 2.0 x_138 <= 0.0
x_17 + x_46 − x_139 <= 1.0
− x_17 − x_46 + 2.0 x_139 <= 0.0
x_23 + x_46 − x_140 <= 1.0
− x_23 − x_46 + 2.0 x_140 <= 0.0
x_18 + x_32 − x_141 <= 1.0
− x_18 − x_32 + 2.0 x_141 <= 0.0
x_6 + x_32 − x_142 <= 1.0
− x_6 − x_32 + 2.0 x_142 <= 0.0
x_19 + x_32 − x_143 <= 1.0
− x_19 − x_32 + 2.0 x_143 <= 0.0
x_7 + x_32 − x_144 <= 1.0
− x_7 − x_32 + 2.0 x_144 <= 0.0
x_20 + x_32 − x_145 <= 1.0
− x_20 − x_32 + 2.0 x_145 <= 0.0
x_8 + x_32 − x_146 <= 1.0
```

```
- x_8 - x_32 + 2.0 x_146 <= 0.0
x_18 + x_40 - x_147 <= 1.0
- x_18 - x_40 + 2.0 x_147 <= 0.0
x_15 + x_40 - x_148 <= 1.0
- x_15 - x_40 + 2.0 x_148 <= 0.0
x_19 + x_40 - x_149 <= 1.0
- x_19 - x_40 + 2.0 x_149 <= 0.0
x_16 + x_40 - x_150 <= 1.0
- x_16 - x_40 + 2.0 x_150 <= 0.0
x_20 + x_40 - x_151 <= 1.0
- x_20 - x_40 + 2.0 x_151 <= 0.0
x_17 + x_40 - x_152 <= 1.0
- x_17 - x_40 + 2.0 x_152 <= 0.0
x_18 + x_49 - x_153 <= 1.0
- x_18 - x_49 + 2.0 x_153 <= 0.0
x_24 + x_49 - x_154 <= 1.0
- x_24 - x_49 + 2.0 x_154 <= 0.0
x_19 + x_49 - x_155 <= 1.0
- x_19 - x_49 + 2.0 x_155 <= 0.0
x_25 + x_49 - x_156 <= 1.0
- x_25 - x_49 + 2.0 x_156 <= 0.0
x_20 + x_49 - x_157 <= 1.0
- x_20 - x_49 + 2.0 x_157 <= 0.0
x_26 + x_49 - x_158 <= 1.0
- x_26 - x_49 + 2.0 x_158 <= 0.0
x_21 + x_35 - x_159 <= 1.0
- x_21 - x_35 + 2.0 x_159 <= 0.0
x_9 + x_35 - x_160 <= 1.0
- x_9 - x_35 + 2.0 x_160 <= 0.0
x_22 + x_35 - x_161 <= 1.0
- x_22 - x_35 + 2.0 x_161 <= 0.0
x_10 + x_35 - x_162 <= 1.0
- x_10 - x_35 + 2.0 x_162 <= 0.0
x_23 + x_35 - x_163 <= 1.0
- x_23 - x_35 + 2.0 x_163 <= 0.0
x_11 + x_35 - x_164 <= 1.0
- x_11 - x_35 + 2.0 x_164 <= 0.0
x_21 + x_41 - x_165 <= 1.0
- x_21 - x_41 + 2.0 x_165 <= 0.0
x_15 + x_41 - x_166 <= 1.0
- x_15 - x_41 + 2.0 x_166 <= 0.0
x_22 + x_41 - x_167 <= 1.0
- x_22 - x_41 + 2.0 x_167 <= 0.0
x_16 + x_41 - x_168 <= 1.0
- x_16 - x_41 + 2.0 x_168 <= 0.0
x_23 + x_41 - x_169 <= 1.0
- x_23 - x_41 + 2.0 x_169 <= 0.0
x_17 + x_41 - x_170 <= 1.0
- x_17 - x_41 + 2.0 x_170 <= 0.0
x_21 + x_50 - x_171 <= 1.0
- x_21 - x_50 + 2.0 x_171 <= 0.0
x_24 + x_50 - x_172 <= 1.0
- x_24 - x_50 + 2.0 x_172 <= 0.0
x_22 + x_50 - x_173 <= 1.0
```

```
- x_22 - x_50 + 2.0 x_173 <= 0.0
x_25 + x_50 - x_174 <= 1.0
- x_25 - x_50 + 2.0 x_174 <= 0.0
x_23 + x_50 - x_175 <= 1.0
- x_23 - x_50 + 2.0 x_175 <= 0.0
x_26 + x_50 - x_176 <= 1.0
- x_26 - x_50 + 2.0 x_176 <= 0.0
x_24 + x_38 - x_177 <= 1.0
- x_24 - x_38 + 2.0 x_177 <= 0.0
x_12 + x_38 - x_178 <= 1.0
- x_12 - x_38 + 2.0 x_178 <= 0.0
x_25 + x_38 - x_179 <= 1.0
- x_25 - x_38 + 2.0 x_179 <= 0.0
x_13 + x_38 - x_180 <= 1.0
- x_13 - x_38 + 2.0 x_180 <= 0.0
x_26 + x_38 - x_181 <= 1.0
- x_26 - x_38 + 2.0 x_181 <= 0.0
x_14 + x_38 - x_182 <= 1.0
- x_14 - x_38 + 2.0 x_182 <= 0.0
x_24 + x_44 - x_183 <= 1.0
- x_24 - x_44 + 2.0 x_183 <= 0.0
x_18 + x_44 - x_184 <= 1.0
- x_18 - x_44 + 2.0 x_184 <= 0.0
x_25 + x_44 - x_185 <= 1.0
- x_25 - x_44 + 2.0 x_185 <= 0.0
x_19 + x_44 - x_186 <= 1.0
- x_19 - x_44 + 2.0 x_186 <= 0.0
x_26 + x_44 - x_187 <= 1.0
- x_26 - x_44 + 2.0 x_187 <= 0.0
x_20 + x_44 - x_188 <= 1.0
- x_20 - x_44 + 2.0 x_188 <= 0.0
x_24 + x_47 - x_189 <= 1.0
- x_24 - x_47 + 2.0 x_189 <= 0.0
x_21 + x_47 - x_190 <= 1.0
- x_21 - x_47 + 2.0 x_190 <= 0.0
x_25 + x_47 - x_191 <= 1.0
- x_25 - x_47 + 2.0 x_191 <= 0.0
x_22 + x_47 - x_192 <= 1.0
- x_22 - x_47 + 2.0 x_192 <= 0.0
x_26 + x_47 - x_193 <= 1.0
- x_26 - x_47 + 2.0 x_193 <= 0.0
x_23 + x_47 - x_194 <= 1.0
- x_23 - x_47 + 2.0 x_194 <= 0.0
x_30 + x_51 - x_52 + 2.0 x_53 - 2.0 x_54 + 4.0 x_55 - 4.0 x_56 <= 0.0
x_33 + x_57 - x_58 + 2.0 x_59 - 2.0 x_60 + 4.0 x_61 - 4.0 x_62 <= 0.0
x_39 + x_63 - x_64 + 2.0 x_65 - 2.0 x_66 + 4.0 x_67 - 4.0 x_68 <= 0.0
x_27 + x_69 - x_70 + 2.0 x_71 - 2.0 x_72 + 4.0 x_73 - 4.0 x_74 <= 0.0
x_36 + x_75 - x_76 + 2.0 x_77 - 2.0 x_78 + 4.0 x_79 - 4.0 x_80 <= 0.0
x_42 + x_81 - x_82 + 2.0 x_83 - 2.0 x_84 + 4.0 x_85 - 4.0 x_86 <= 0.0
x_28 + x_87 - x_88 + 2.0 x_89 - 2.0 x_90 + 4.0 x_91 - 4.0 x_92 <= 0.0
x_37 + x_93 - x_94 + 2.0 x_95 - 2.0 x_96 + 4.0 x_97 - 4.0 x_98 <= 0.0
x_45 + x_99 - x_100 + 2.0 x_101 - 2.0 x_102 + 4.0 x_103 - 4.0 x_104 <= 0.0
x_31 + x_105 - x_106 + 2.0 x_107 - 2.0 x_108 + 4.0 x_109 - 4.0 x_110 <= 0.0
x_34 + x_111 - x_112 + 2.0 x_113 - 2.0 x_114 + 4.0 x_115 - 4.0 x_116 <= 0.0
```

```
x_48 + x_117 − x_118 + 2.0 x_119 − 2.0 x_120 + 4.0 x_121 − 4.0 x_122 <= 0.0
x_29 + x_123 − x_124 + 2.0 x_125 − 2.0 x_126 + 4.0 x_127 − 4.0 x_128 <= 0.0
x_43 + x_129 − x_130 + 2.0 x_131 − 2.0 x_132 + 4.0 x_133 − 4.0 x_134 <= 0.0
x_46 + x_135 − x_136 + 2.0 x_137 − 2.0 x_138 + 4.0 x_139 − 4.0 x_140 <= 0.0
x_32 + x_141 − x_142 + 2.0 x_143 − 2.0 x_144 + 4.0 x_145 − 4.0 x_146 <= 0.0
x_40 + x_147 − x_148 + 2.0 x_149 − 2.0 x_150 + 4.0 x_151 − 4.0 x_152 <= 0.0
x_49 + x_153 − x_154 + 2.0 x_155 − 2.0 x_156 + 4.0 x_157 − 4.0 x_158 <= 0.0
x_35 + x_159 − x_160 + 2.0 x_161 − 2.0 x_162 + 4.0 x_163 − 4.0 x_164 <= 0.0
x_41 + x_165 − x_166 + 2.0 x_167 − 2.0 x_168 + 4.0 x_169 − 4.0 x_170 <= 0.0
x_50 + x_171 − x_172 + 2.0 x_173 − 2.0 x_174 + 4.0 x_175 − 4.0 x_176 <= 0.0
x_38 + x_177 − x_178 + 2.0 x_179 − 2.0 x_180 + 4.0 x_181 − 4.0 x_182 <= 0.0
x_44 + x_183 − x_184 + 2.0 x_185 − 2.0 x_186 + 4.0 x_187 − 4.0 x_188 <= 0.0
x_47 + x_189 − x_190 + 2.0 x_191 − 2.0 x_192 + 4.0 x_193 − 4.0 x_194 <= 0.0
2.0 x_6 − x_195 <= 1.0
−2.0 x_6 + 2.0 x_195 <= 0.0
x_6 + x_7 − x_196 <= 1.0
− x_6 − x_7 + 2.0 x_196 <= 0.0
x_6 + x_8 − x_197 <= 1.0
− x_6 − x_8 + 2.0 x_197 <= 0.0
x_6 + x_7 − x_198 <= 1.0
− x_6 − x_7 + 2.0 x_198 <= 0.0
2.0 x_7 − x_199 <= 1.0
−2.0 x_7 + 2.0 x_199 <= 0.0
x_7 + x_8 − x_200 <= 1.0
− x_7 − x_8 + 2.0 x_200 <= 0.0
x_6 + x_8 − x_201 <= 1.0
− x_6 − x_8 + 2.0 x_201 <= 0.0
x_7 + x_8 − x_202 <= 1.0
− x_7 − x_8 + 2.0 x_202 <= 0.0
2.0 x_8 − x_203 <= 1.0
−2.0 x_8 + 2.0 x_203 <= 0.0
x_6 + x_9 − x_204 <= 1.0
− x_6 − x_9 + 2.0 x_204 <= 0.0
x_6 + x_10 − x_205 <= 1.0
− x_6 − x_10 + 2.0 x_205 <= 0.0
x_6 + x_11 − x_206 <= 1.0
− x_6 − x_11 + 2.0 x_206 <= 0.0
x_7 + x_9 − x_207 <= 1.0
− x_7 − x_9 + 2.0 x_207 <= 0.0
x_7 + x_10 − x_208 <= 1.0
− x_7 − x_10 + 2.0 x_208 <= 0.0
x_7 + x_11 − x_209 <= 1.0
− x_7 − x_11 + 2.0 x_209 <= 0.0
x_8 + x_9 − x_210 <= 1.0
− x_8 − x_9 + 2.0 x_210 <= 0.0
x_8 + x_10 − x_211 <= 1.0
− x_8 − x_10 + 2.0 x_211 <= 0.0
x_8 + x_11 − x_212 <= 1.0
− x_8 − x_11 + 2.0 x_212 <= 0.0
x_6 + x_12 − x_213 <= 1.0
− x_6 − x_12 + 2.0 x_213 <= 0.0
x_6 + x_13 − x_214 <= 1.0
− x_6 − x_13 + 2.0 x_214 <= 0.0
x_6 + x_14 − x_215 <= 1.0
```

```
- x_6 - x_14 + 2.0 x_215 <= 0.0
x_7 + x_12 - x_216 <= 1.0
- x_7 - x_12 + 2.0 x_216 <= 0.0
x_7 + x_13 - x_217 <= 1.0
- x_7 - x_13 + 2.0 x_217 <= 0.0
x_7 + x_14 - x_218 <= 1.0
- x_7 - x_14 + 2.0 x_218 <= 0.0
x_8 + x_12 - x_219 <= 1.0
- x_8 - x_12 + 2.0 x_219 <= 0.0
x_8 + x_13 - x_220 <= 1.0
- x_8 - x_13 + 2.0 x_220 <= 0.0
x_8 + x_14 - x_221 <= 1.0
- x_8 - x_14 + 2.0 x_221 <= 0.0
x_6 + x_15 - x_222 <= 1.0
- x_6 - x_15 + 2.0 x_222 <= 0.0
x_6 + x_16 - x_223 <= 1.0
- x_6 - x_16 + 2.0 x_223 <= 0.0
x_6 + x_17 - x_224 <= 1.0
- x_6 - x_17 + 2.0 x_224 <= 0.0
x_7 + x_15 - x_225 <= 1.0
- x_7 - x_15 + 2.0 x_225 <= 0.0
x_7 + x_16 - x_226 <= 1.0
- x_7 - x_16 + 2.0 x_226 <= 0.0
x_7 + x_17 - x_227 <= 1.0
- x_7 - x_17 + 2.0 x_227 <= 0.0
x_8 + x_15 - x_228 <= 1.0
- x_8 - x_15 + 2.0 x_228 <= 0.0
x_8 + x_16 - x_229 <= 1.0
- x_8 - x_16 + 2.0 x_229 <= 0.0
x_8 + x_17 - x_230 <= 1.0
- x_8 - x_17 + 2.0 x_230 <= 0.0
x_6 + x_18 - x_231 <= 1.0
- x_6 - x_18 + 2.0 x_231 <= 0.0
x_6 + x_19 - x_232 <= 1.0
- x_6 - x_19 + 2.0 x_232 <= 0.0
x_6 + x_20 - x_233 <= 1.0
- x_6 - x_20 + 2.0 x_233 <= 0.0
x_7 + x_18 - x_234 <= 1.0
- x_7 - x_18 + 2.0 x_234 <= 0.0
x_7 + x_19 - x_235 <= 1.0
- x_7 - x_19 + 2.0 x_235 <= 0.0
x_7 + x_20 - x_236 <= 1.0
- x_7 - x_20 + 2.0 x_236 <= 0.0
x_8 + x_18 - x_237 <= 1.0
- x_8 - x_18 + 2.0 x_237 <= 0.0
x_8 + x_19 - x_238 <= 1.0
- x_8 - x_19 + 2.0 x_238 <= 0.0
x_8 + x_20 - x_239 <= 1.0
- x_8 - x_20 + 2.0 x_239 <= 0.0
x_6 + x_21 - x_240 <= 1.0
- x_6 - x_21 + 2.0 x_240 <= 0.0
x_6 + x_22 - x_241 <= 1.0
- x_6 - x_22 + 2.0 x_241 <= 0.0
x_6 + x_23 - x_242 <= 1.0
```

```
− x_6 − x_23 + 2.0 x_242 <= 0.0
 x_7 + x_21 − x_243 <= 1.0
− x_7 − x_21 + 2.0 x_243 <= 0.0
 x_7 + x_22 − x_244 <= 1.0
− x_7 − x_22 + 2.0 x_244 <= 0.0
 x_7 + x_23 − x_245 <= 1.0
− x_7 − x_23 + 2.0 x_245 <= 0.0
 x_8 + x_21 − x_246 <= 1.0
− x_8 − x_21 + 2.0 x_246 <= 0.0
 x_8 + x_22 − x_247 <= 1.0
− x_8 − x_22 + 2.0 x_247 <= 0.0
 x_8 + x_23 − x_248 <= 1.0
− x_8 − x_23 + 2.0 x_248 <= 0.0
 x_6 + x_24 − x_249 <= 1.0
− x_6 − x_24 + 2.0 x_249 <= 0.0
 x_6 + x_25 − x_250 <= 1.0
− x_6 − x_25 + 2.0 x_250 <= 0.0
 x_6 + x_26 − x_251 <= 1.0
− x_6 − x_26 + 2.0 x_251 <= 0.0
 x_7 + x_24 − x_252 <= 1.0
− x_7 − x_24 + 2.0 x_252 <= 0.0
 x_7 + x_25 − x_253 <= 1.0
− x_7 − x_25 + 2.0 x_253 <= 0.0
 x_7 + x_26 − x_254 <= 1.0
− x_7 − x_26 + 2.0 x_254 <= 0.0
 x_8 + x_24 − x_255 <= 1.0
− x_8 − x_24 + 2.0 x_255 <= 0.0
 x_8 + x_25 − x_256 <= 1.0
− x_8 − x_25 + 2.0 x_256 <= 0.0
 x_8 + x_26 − x_257 <= 1.0
− x_8 − x_26 + 2.0 x_257 <= 0.0
 2.0 x_9 − x_258 <= 1.0
 −2.0 x_9 + 2.0 x_258 <= 0.0
 x_9 + x_10 − x_259 <= 1.0
− x_9 − x_10 + 2.0 x_259 <= 0.0
 x_9 + x_11 − x_260 <= 1.0
− x_9 − x_11 + 2.0 x_260 <= 0.0
 x_9 + x_10 − x_261 <= 1.0
− x_9 − x_10 + 2.0 x_261 <= 0.0
 2.0 x_10 − x_262 <= 1.0
 −2.0 x_10 + 2.0 x_262 <= 0.0
 x_10 + x_11 − x_263 <= 1.0
− x_10 − x_11 + 2.0 x_263 <= 0.0
 x_9 + x_11 − x_264 <= 1.0
− x_9 − x_11 + 2.0 x_264 <= 0.0
 x_10 + x_11 − x_265 <= 1.0
− x_10 − x_11 + 2.0 x_265 <= 0.0
 2.0 x_11 − x_266 <= 1.0
 −2.0 x_11 + 2.0 x_266 <= 0.0
 x_9 + x_12 − x_267 <= 1.0
− x_9 − x_12 + 2.0 x_267 <= 0.0
 x_9 + x_13 − x_268 <= 1.0
− x_9 − x_13 + 2.0 x_268 <= 0.0
 x_9 + x_14 − x_269 <= 1.0
```

```
− x_9 − x_14 + 2.0 x_269 <= 0.0
x_10 + x_12 − x_270 <= 1.0
− x_10 − x_12 + 2.0 x_270 <= 0.0
x_10 + x_13 − x_271 <= 1.0
− x_10 − x_13 + 2.0 x_271 <= 0.0
x_10 + x_14 − x_272 <= 1.0
− x_10 − x_14 + 2.0 x_272 <= 0.0
x_11 + x_12 − x_273 <= 1.0
− x_11 − x_12 + 2.0 x_273 <= 0.0
x_11 + x_13 − x_274 <= 1.0
− x_11 − x_13 + 2.0 x_274 <= 0.0
x_11 + x_14 − x_275 <= 1.0
− x_11 − x_14 + 2.0 x_275 <= 0.0
x_9 + x_15 − x_276 <= 1.0
− x_9 − x_15 + 2.0 x_276 <= 0.0
x_9 + x_16 − x_277 <= 1.0
− x_9 − x_16 + 2.0 x_277 <= 0.0
x_9 + x_17 − x_278 <= 1.0
− x_9 − x_17 + 2.0 x_278 <= 0.0
x_10 + x_15 − x_279 <= 1.0
− x_10 − x_15 + 2.0 x_279 <= 0.0
x_10 + x_16 − x_280 <= 1.0
− x_10 − x_16 + 2.0 x_280 <= 0.0
x_10 + x_17 − x_281 <= 1.0
− x_10 − x_17 + 2.0 x_281 <= 0.0
x_11 + x_15 − x_282 <= 1.0
− x_11 − x_15 + 2.0 x_282 <= 0.0
x_11 + x_16 − x_283 <= 1.0
− x_11 − x_16 + 2.0 x_283 <= 0.0
x_11 + x_17 − x_284 <= 1.0
− x_11 − x_17 + 2.0 x_284 <= 0.0
x_9 + x_18 − x_285 <= 1.0
− x_9 − x_18 + 2.0 x_285 <= 0.0
x_9 + x_19 − x_286 <= 1.0
− x_9 − x_19 + 2.0 x_286 <= 0.0
x_9 + x_20 − x_287 <= 1.0
− x_9 − x_20 + 2.0 x_287 <= 0.0
x_10 + x_18 − x_288 <= 1.0
− x_10 − x_18 + 2.0 x_288 <= 0.0
x_10 + x_19 − x_289 <= 1.0
− x_10 − x_19 + 2.0 x_289 <= 0.0
x_10 + x_20 − x_290 <= 1.0
− x_10 − x_20 + 2.0 x_290 <= 0.0
x_11 + x_18 − x_291 <= 1.0
− x_11 − x_18 + 2.0 x_291 <= 0.0
x_11 + x_19 − x_292 <= 1.0
− x_11 − x_19 + 2.0 x_292 <= 0.0
x_11 + x_20 − x_293 <= 1.0
− x_11 − x_20 + 2.0 x_293 <= 0.0
x_9 + x_21 − x_294 <= 1.0
− x_9 − x_21 + 2.0 x_294 <= 0.0
x_9 + x_22 − x_295 <= 1.0
− x_9 − x_22 + 2.0 x_295 <= 0.0
x_9 + x_23 − x_296 <= 1.0
```

```
- x_9 - x_23 + 2.0 x_296 <= 0.0
x_10 + x_21 - x_297 <= 1.0
- x_10 - x_21 + 2.0 x_297 <= 0.0
x_10 + x_22 - x_298 <= 1.0
- x_10 - x_22 + 2.0 x_298 <= 0.0
x_10 + x_23 - x_299 <= 1.0
- x_10 - x_23 + 2.0 x_299 <= 0.0
x_11 + x_21 - x_300 <= 1.0
- x_11 - x_21 + 2.0 x_300 <= 0.0
x_11 + x_22 - x_301 <= 1.0
- x_11 - x_22 + 2.0 x_301 <= 0.0
x_11 + x_23 - x_302 <= 1.0
- x_11 - x_23 + 2.0 x_302 <= 0.0
x_9 + x_24 - x_303 <= 1.0
- x_9 - x_24 + 2.0 x_303 <= 0.0
x_9 + x_25 - x_304 <= 1.0
- x_9 - x_25 + 2.0 x_304 <= 0.0
x_9 + x_26 - x_305 <= 1.0
- x_9 - x_26 + 2.0 x_305 <= 0.0
x_10 + x_24 - x_306 <= 1.0
- x_10 - x_24 + 2.0 x_306 <= 0.0
x_10 + x_25 - x_307 <= 1.0
- x_10 - x_25 + 2.0 x_307 <= 0.0
x_10 + x_26 - x_308 <= 1.0
- x_10 - x_26 + 2.0 x_308 <= 0.0
x_11 + x_24 - x_309 <= 1.0
- x_11 - x_24 + 2.0 x_309 <= 0.0
x_11 + x_25 - x_310 <= 1.0
- x_11 - x_25 + 2.0 x_310 <= 0.0
x_11 + x_26 - x_311 <= 1.0
- x_11 - x_26 + 2.0 x_311 <= 0.0
2.0 x_12 - x_312 <= 1.0
-2.0 x_12 + 2.0 x_312 <= 0.0
x_12 + x_13 - x_313 <= 1.0
- x_12 - x_13 + 2.0 x_313 <= 0.0
x_12 + x_14 - x_314 <= 1.0
- x_12 - x_14 + 2.0 x_314 <= 0.0
x_12 + x_13 - x_315 <= 1.0
- x_12 - x_13 + 2.0 x_315 <= 0.0
2.0 x_13 - x_316 <= 1.0
-2.0 x_13 + 2.0 x_316 <= 0.0
x_13 + x_14 - x_317 <= 1.0
- x_13 - x_14 + 2.0 x_317 <= 0.0
x_12 + x_14 - x_318 <= 1.0
- x_12 - x_14 + 2.0 x_318 <= 0.0
x_13 + x_14 - x_319 <= 1.0
- x_13 - x_14 + 2.0 x_319 <= 0.0
2.0 x_14 - x_320 <= 1.0
-2.0 x_14 + 2.0 x_320 <= 0.0
x_12 + x_15 - x_321 <= 1.0
- x_12 - x_15 + 2.0 x_321 <= 0.0
x_12 + x_16 - x_322 <= 1.0
- x_12 - x_16 + 2.0 x_322 <= 0.0
x_12 + x_17 - x_323 <= 1.0
```

```
- x_12 - x_17 + 2.0 x_323 <= 0.0
x_13 + x_15 - x_324 <= 1.0
- x_13 - x_15 + 2.0 x_324 <= 0.0
x_13 + x_16 - x_325 <= 1.0
- x_13 - x_16 + 2.0 x_325 <= 0.0
x_13 + x_17 - x_326 <= 1.0
- x_13 - x_17 + 2.0 x_326 <= 0.0
x_14 + x_15 - x_327 <= 1.0
- x_14 - x_15 + 2.0 x_327 <= 0.0
x_14 + x_16 - x_328 <= 1.0
- x_14 - x_16 + 2.0 x_328 <= 0.0
x_14 + x_17 - x_329 <= 1.0
- x_14 - x_17 + 2.0 x_329 <= 0.0
x_12 + x_18 - x_330 <= 1.0
- x_12 - x_18 + 2.0 x_330 <= 0.0
x_12 + x_19 - x_331 <= 1.0
- x_12 - x_19 + 2.0 x_331 <= 0.0
x_12 + x_20 - x_332 <= 1.0
- x_12 - x_20 + 2.0 x_332 <= 0.0
x_13 + x_18 - x_333 <= 1.0
- x_13 - x_18 + 2.0 x_333 <= 0.0
x_13 + x_19 - x_334 <= 1.0
- x_13 - x_19 + 2.0 x_334 <= 0.0
x_13 + x_20 - x_335 <= 1.0
- x_13 - x_20 + 2.0 x_335 <= 0.0
x_14 + x_18 - x_336 <= 1.0
- x_14 - x_18 + 2.0 x_336 <= 0.0
x_14 + x_19 - x_337 <= 1.0
- x_14 - x_19 + 2.0 x_337 <= 0.0
x_14 + x_20 - x_338 <= 1.0
- x_14 - x_20 + 2.0 x_338 <= 0.0
x_12 + x_21 - x_339 <= 1.0
- x_12 - x_21 + 2.0 x_339 <= 0.0
x_12 + x_22 - x_340 <= 1.0
- x_12 - x_22 + 2.0 x_340 <= 0.0
x_12 + x_23 - x_341 <= 1.0
- x_12 - x_23 + 2.0 x_341 <= 0.0
x_13 + x_21 - x_342 <= 1.0
- x_13 - x_21 + 2.0 x_342 <= 0.0
x_13 + x_22 - x_343 <= 1.0
- x_13 - x_22 + 2.0 x_343 <= 0.0
x_13 + x_23 - x_344 <= 1.0
- x_13 - x_23 + 2.0 x_344 <= 0.0
x_14 + x_21 - x_345 <= 1.0
- x_14 - x_21 + 2.0 x_345 <= 0.0
x_14 + x_22 - x_346 <= 1.0
- x_14 - x_22 + 2.0 x_346 <= 0.0
x_14 + x_23 - x_347 <= 1.0
- x_14 - x_23 + 2.0 x_347 <= 0.0
x_12 + x_24 - x_348 <= 1.0
- x_12 - x_24 + 2.0 x_348 <= 0.0
x_12 + x_25 - x_349 <= 1.0
- x_12 - x_25 + 2.0 x_349 <= 0.0
x_12 + x_26 - x_350 <= 1.0
```

```
- x_12 - x_26 + 2.0 x_350 <= 0.0
x_13 + x_24 - x_351 <= 1.0
- x_13 - x_24 + 2.0 x_351 <= 0.0
x_13 + x_25 - x_352 <= 1.0
- x_13 - x_25 + 2.0 x_352 <= 0.0
x_13 + x_26 - x_353 <= 1.0
- x_13 - x_26 + 2.0 x_353 <= 0.0
x_14 + x_24 - x_354 <= 1.0
- x_14 - x_24 + 2.0 x_354 <= 0.0
x_14 + x_25 - x_355 <= 1.0
- x_14 - x_25 + 2.0 x_355 <= 0.0
x_14 + x_26 - x_356 <= 1.0
- x_14 - x_26 + 2.0 x_356 <= 0.0
2.0 x_15 - x_357 <= 1.0
-2.0 x_15 + 2.0 x_357 <= 0.0
x_15 + x_16 - x_358 <= 1.0
- x_15 - x_16 + 2.0 x_358 <= 0.0
x_15 + x_17 - x_359 <= 1.0
- x_15 - x_17 + 2.0 x_359 <= 0.0
x_15 + x_16 - x_360 <= 1.0
- x_15 - x_16 + 2.0 x_360 <= 0.0
2.0 x_16 - x_361 <= 1.0
-2.0 x_16 + 2.0 x_361 <= 0.0
x_16 + x_17 - x_362 <= 1.0
- x_16 - x_17 + 2.0 x_362 <= 0.0
x_15 + x_17 - x_363 <= 1.0
- x_15 - x_17 + 2.0 x_363 <= 0.0
x_16 + x_17 - x_364 <= 1.0
- x_16 - x_17 + 2.0 x_364 <= 0.0
2.0 x_17 - x_365 <= 1.0
-2.0 x_17 + 2.0 x_365 <= 0.0
x_15 + x_18 - x_366 <= 1.0
- x_15 - x_18 + 2.0 x_366 <= 0.0
x_15 + x_19 - x_367 <= 1.0
- x_15 - x_19 + 2.0 x_367 <= 0.0
x_15 + x_20 - x_368 <= 1.0
- x_15 - x_20 + 2.0 x_368 <= 0.0
x_16 + x_18 - x_369 <= 1.0
- x_16 - x_18 + 2.0 x_369 <= 0.0
x_16 + x_19 - x_370 <= 1.0
- x_16 - x_19 + 2.0 x_370 <= 0.0
x_16 + x_20 - x_371 <= 1.0
- x_16 - x_20 + 2.0 x_371 <= 0.0
x_17 + x_18 - x_372 <= 1.0
- x_17 - x_18 + 2.0 x_372 <= 0.0
x_17 + x_19 - x_373 <= 1.0
- x_17 - x_19 + 2.0 x_373 <= 0.0
x_17 + x_20 - x_374 <= 1.0
- x_17 - x_20 + 2.0 x_374 <= 0.0
x_15 + x_21 - x_375 <= 1.0
- x_15 - x_21 + 2.0 x_375 <= 0.0
x_15 + x_22 - x_376 <= 1.0
- x_15 - x_22 + 2.0 x_376 <= 0.0
x_15 + x_23 - x_377 <= 1.0
```

```
- x_15 - x_23 + 2.0 x_377 <= 0.0
x_16 + x_21 - x_378 <= 1.0
- x_16 - x_21 + 2.0 x_378 <= 0.0
x_16 + x_22 - x_379 <= 1.0
- x_16 - x_22 + 2.0 x_379 <= 0.0
x_16 + x_23 - x_380 <= 1.0
- x_16 - x_23 + 2.0 x_380 <= 0.0
x_17 + x_21 - x_381 <= 1.0
- x_17 - x_21 + 2.0 x_381 <= 0.0
x_17 + x_22 - x_382 <= 1.0
- x_17 - x_22 + 2.0 x_382 <= 0.0
x_17 + x_23 - x_383 <= 1.0
- x_17 - x_23 + 2.0 x_383 <= 0.0
x_15 + x_24 - x_384 <= 1.0
- x_15 - x_24 + 2.0 x_384 <= 0.0
x_15 + x_25 - x_385 <= 1.0
- x_15 - x_25 + 2.0 x_385 <= 0.0
x_15 + x_26 - x_386 <= 1.0
- x_15 - x_26 + 2.0 x_386 <= 0.0
x_16 + x_24 - x_387 <= 1.0
- x_16 - x_24 + 2.0 x_387 <= 0.0
x_16 + x_25 - x_388 <= 1.0
- x_16 - x_25 + 2.0 x_388 <= 0.0
x_16 + x_26 - x_389 <= 1.0
- x_16 - x_26 + 2.0 x_389 <= 0.0
x_17 + x_24 - x_390 <= 1.0
- x_17 - x_24 + 2.0 x_390 <= 0.0
x_17 + x_25 - x_391 <= 1.0
- x_17 - x_25 + 2.0 x_391 <= 0.0
x_17 + x_26 - x_392 <= 1.0
- x_17 - x_26 + 2.0 x_392 <= 0.0
2.0 x_18 - x_393 <= 1.0
-2.0 x_18 + 2.0 x_393 <= 0.0
x_18 + x_19 - x_394 <= 1.0
- x_18 - x_19 + 2.0 x_394 <= 0.0
x_18 + x_20 - x_395 <= 1.0
- x_18 - x_20 + 2.0 x_395 <= 0.0
x_18 + x_19 - x_396 <= 1.0
- x_18 - x_19 + 2.0 x_396 <= 0.0
2.0 x_19 - x_397 <= 1.0
-2.0 x_19 + 2.0 x_397 <= 0.0
x_19 + x_20 - x_398 <= 1.0
- x_19 - x_20 + 2.0 x_398 <= 0.0
x_18 + x_20 - x_399 <= 1.0
- x_18 - x_20 + 2.0 x_399 <= 0.0
x_19 + x_20 - x_400 <= 1.0
- x_19 - x_20 + 2.0 x_400 <= 0.0
2.0 x_20 - x_401 <= 1.0
-2.0 x_20 + 2.0 x_401 <= 0.0
x_18 + x_21 - x_402 <= 1.0
- x_18 - x_21 + 2.0 x_402 <= 0.0
x_18 + x_22 - x_403 <= 1.0
- x_18 - x_22 + 2.0 x_403 <= 0.0
x_18 + x_23 - x_404 <= 1.0
```

```
- x_18 - x_23 + 2.0 x_404 <= 0.0
x_19 + x_21 - x_405 <= 1.0
- x_19 - x_21 + 2.0 x_405 <= 0.0
x_19 + x_22 - x_406 <= 1.0
- x_19 - x_22 + 2.0 x_406 <= 0.0
x_19 + x_23 - x_407 <= 1.0
- x_19 - x_23 + 2.0 x_407 <= 0.0
x_20 + x_21 - x_408 <= 1.0
- x_20 - x_21 + 2.0 x_408 <= 0.0
x_20 + x_22 - x_409 <= 1.0
- x_20 - x_22 + 2.0 x_409 <= 0.0
x_20 + x_23 - x_410 <= 1.0
- x_20 - x_23 + 2.0 x_410 <= 0.0
x_18 + x_24 - x_411 <= 1.0
- x_18 - x_24 + 2.0 x_411 <= 0.0
x_18 + x_25 - x_412 <= 1.0
- x_18 - x_25 + 2.0 x_412 <= 0.0
x_18 + x_26 - x_413 <= 1.0
- x_18 - x_26 + 2.0 x_413 <= 0.0
x_19 + x_24 - x_414 <= 1.0
- x_19 - x_24 + 2.0 x_414 <= 0.0
x_19 + x_25 - x_415 <= 1.0
- x_19 - x_25 + 2.0 x_415 <= 0.0
x_19 + x_26 - x_416 <= 1.0
- x_19 - x_26 + 2.0 x_416 <= 0.0
x_20 + x_24 - x_417 <= 1.0
- x_20 - x_24 + 2.0 x_417 <= 0.0
x_20 + x_25 - x_418 <= 1.0
- x_20 - x_25 + 2.0 x_418 <= 0.0
x_20 + x_26 - x_419 <= 1.0
- x_20 - x_26 + 2.0 x_419 <= 0.0
2.0 x_21 - x_420 <= 1.0
-2.0 x_21 + 2.0 x_420 <= 0.0
x_21 + x_22 - x_421 <= 1.0
- x_21 - x_22 + 2.0 x_421 <= 0.0
x_21 + x_23 - x_422 <= 1.0
- x_21 - x_23 + 2.0 x_422 <= 0.0
x_21 + x_22 - x_423 <= 1.0
- x_21 - x_22 + 2.0 x_423 <= 0.0
2.0 x_22 - x_424 <= 1.0
-2.0 x_22 + 2.0 x_424 <= 0.0
x_22 + x_23 - x_425 <= 1.0
- x_22 - x_23 + 2.0 x_425 <= 0.0
x_21 + x_23 - x_426 <= 1.0
- x_21 - x_23 + 2.0 x_426 <= 0.0
x_22 + x_23 - x_427 <= 1.0
- x_22 - x_23 + 2.0 x_427 <= 0.0
2.0 x_23 - x_428 <= 1.0
-2.0 x_23 + 2.0 x_428 <= 0.0
x_21 + x_24 - x_429 <= 1.0
- x_21 - x_24 + 2.0 x_429 <= 0.0
x_21 + x_25 - x_430 <= 1.0
- x_21 - x_25 + 2.0 x_430 <= 0.0
x_21 + x_26 - x_431 <= 1.0
```

```
- x_21 - x_26 + 2.0 x_431 <= 0.0
x_22 + x_24 - x_432 <= 1.0
- x_22 - x_24 + 2.0 x_432 <= 0.0
x_22 + x_25 - x_433 <= 1.0
- x_22 - x_25 + 2.0 x_433 <= 0.0
x_22 + x_26 - x_434 <= 1.0
- x_22 - x_26 + 2.0 x_434 <= 0.0
x_23 + x_24 - x_435 <= 1.0
- x_23 - x_24 + 2.0 x_435 <= 0.0
x_23 + x_25 - x_436 <= 1.0
- x_23 - x_25 + 2.0 x_436 <= 0.0
x_23 + x_26 - x_437 <= 1.0
- x_23 - x_26 + 2.0 x_437 <= 0.0
2.0 x_24 - x_438 <= 1.0
-2.0 x_24 + 2.0 x_438 <= 0.0
x_24 + x_25 - x_439 <= 1.0
- x_24 - x_25 + 2.0 x_439 <= 0.0
x_24 + x_26 - x_440 <= 1.0
- x_24 - x_26 + 2.0 x_440 <= 0.0
x_24 + x_25 - x_441 <= 1.0
- x_24 - x_25 + 2.0 x_441 <= 0.0
2.0 x_25 - x_442 <= 1.0
-2.0 x_25 + 2.0 x_442 <= 0.0
x_25 + x_26 - x_443 <= 1.0
- x_25 - x_26 + 2.0 x_443 <= 0.0
x_24 + x_26 - x_444 <= 1.0
- x_24 - x_26 + 2.0 x_444 <= 0.0
x_25 + x_26 - x_445 <= 1.0
- x_25 - x_26 + 2.0 x_445 <= 0.0
2.0 x_26 - x_446 <= 1.0
-2.0 x_26 + 2.0 x_446 <= 0.0
x_30 + x_33 - x_195 - 2.0 x_196 - 4.0 x_197 - 2.0 x_198 - 4.0 x_199 - 8.0
  x_200 - 4.0 x_201 - 8.0 x_202 - 16.0 x_203 + 2.0 x_204 + 4.0 x_205 + 8.0
  x_206 + 4.0 x_207 + 8.0 x_208 + 16.0 x_209 + 8.0 x_210 + 16.0 x_211 + 32.0
   x_212 - x_258 - 2.0 x_259 - 4.0 x_260 - 2.0 x_261 - 4.0 x_262 - 8.0 x_263
   - 4.0 x_264 - 8.0 x_265 - 16.0 x_266 <= 1.0
x_30 + x_39 - x_195 - 2.0 x_196 - 4.0 x_197 - 2.0 x_198 - 4.0 x_199 - 8.0
  x_200 - 4.0 x_201 - 8.0 x_202 - 16.0 x_203 + 2.0 x_222 + 4.0 x_223 + 8.0
  x_224 + 4.0 x_225 + 8.0 x_226 + 16.0 x_227 + 8.0 x_228 + 16.0 x_229 + 32.0
   x_230 - x_357 - 2.0 x_358 - 4.0 x_359 - 2.0 x_360 - 4.0 x_361 - 8.0 x_362
   - 4.0 x_363 - 8.0 x_364 - 16.0 x_365 <= 1.0
x_33 + x_39 - x_258 - 2.0 x_259 - 4.0 x_260 - 2.0 x_261 - 4.0 x_262 - 8.0
  x_263 - 4.0 x_264 - 8.0 x_265 - 16.0 x_266 + 2.0 x_276 + 4.0 x_277 + 8.0
  x_278 + 4.0 x_279 + 8.0 x_280 + 16.0 x_281 + 8.0 x_282 + 16.0 x_283 + 32.0
   x_284 - x_357 - 2.0 x_358 - 4.0 x_359 - 2.0 x_360 - 4.0 x_361 - 8.0 x_362
   - 4.0 x_363 - 8.0 x_364 - 16.0 x_365 <= 1.0
x_36 + x_42 - x_312 - 2.0 x_313 - 4.0 x_314 - 2.0 x_315 - 4.0 x_316 - 8.0
  x_317 - 4.0 x_318 - 8.0 x_319 - 16.0 x_320 + 2.0 x_330 + 4.0 x_331 + 8.0
  x_332 + 4.0 x_333 + 8.0 x_334 + 16.0 x_335 + 8.0 x_336 + 16.0 x_337 + 32.0
   x_338 - x_393 - 2.0 x_394 - 4.0 x_395 - 2.0 x_396 - 4.0 x_397 - 8.0 x_398
   - 4.0 x_399 - 8.0 x_400 - 16.0 x_401 <= 1.0
x_37 + x_45 - x_312 - 2.0 x_313 - 4.0 x_314 - 2.0 x_315 - 4.0 x_316 - 8.0
  x_317 - 4.0 x_318 - 8.0 x_319 - 16.0 x_320 + 2.0 x_339 + 4.0 x_340 + 8.0
  x_341 + 4.0 x_342 + 8.0 x_343 + 16.0 x_344 + 8.0 x_345 + 16.0 x_346 + 32.0
```

```
  x_347 − x_420 − 2.0 x_421 − 4.0 x_422 − 2.0 x_423 − 4.0 x_424 − 8.0 x_425
  − 4.0 x_426 − 8.0 x_427 − 16.0 x_428 <= 1.0
x_31 + x_34 − x_195 − 2.0 x_196 − 4.0 x_197 − 2.0 x_198 − 4.0 x_199 − 8.0
 x_200 − 4.0 x_201 − 8.0 x_202 − 16.0 x_203 + 2.0 x_204 + 4.0 x_205 + 8.0
 x_206 + 4.0 x_207 + 8.0 x_208 + 16.0 x_209 + 8.0 x_210 + 16.0 x_211 + 32.0
  x_212 − x_258 − 2.0 x_259 − 4.0 x_260 − 2.0 x_261 − 4.0 x_262 − 8.0 x_263
  − 4.0 x_264 − 8.0 x_265 − 16.0 x_266 <= 1.0
x_31 + x_48 − x_195 − 2.0 x_196 − 4.0 x_197 − 2.0 x_198 − 4.0 x_199 − 8.0
 x_200 − 4.0 x_201 − 8.0 x_202 − 16.0 x_203 + 2.0 x_249 + 4.0 x_250 + 8.0
 x_251 + 4.0 x_252 + 8.0 x_253 + 16.0 x_254 + 8.0 x_255 + 16.0 x_256 + 32.0
  x_257 − x_438 − 2.0 x_439 − 4.0 x_440 − 2.0 x_441 − 4.0 x_442 − 8.0 x_443
  − 4.0 x_444 − 8.0 x_445 − 16.0 x_446 <= 1.0
x_34 + x_48 − x_258 − 2.0 x_259 − 4.0 x_260 − 2.0 x_261 − 4.0 x_262 − 8.0
 x_263 − 4.0 x_264 − 8.0 x_265 − 16.0 x_266 + 2.0 x_303 + 4.0 x_304 + 8.0
 x_305 + 4.0 x_306 + 8.0 x_307 + 16.0 x_308 + 8.0 x_309 + 16.0 x_310 + 32.0
  x_311 − x_438 − 2.0 x_439 − 4.0 x_440 − 2.0 x_441 − 4.0 x_442 − 8.0 x_443
  − 4.0 x_444 − 8.0 x_445 − 16.0 x_446 <= 1.0
x_43 + x_46 − x_393 − 2.0 x_394 − 4.0 x_395 − 2.0 x_396 − 4.0 x_397 − 8.0
 x_398 − 4.0 x_399 − 8.0 x_400 − 16.0 x_401 + 2.0 x_402 + 4.0 x_403 + 8.0
 x_404 + 4.0 x_405 + 8.0 x_406 + 16.0 x_407 + 8.0 x_408 + 16.0 x_409 + 32.0
  x_410 − x_420 − 2.0 x_421 − 4.0 x_422 − 2.0 x_423 − 4.0 x_424 − 8.0 x_425
  − 4.0 x_426 − 8.0 x_427 − 16.0 x_428 <= 1.0
x_32 + x_40 − x_195 − 2.0 x_196 − 4.0 x_197 − 2.0 x_198 − 4.0 x_199 − 8.0
 x_200 − 4.0 x_201 − 8.0 x_202 − 16.0 x_203 + 2.0 x_222 + 4.0 x_223 + 8.0
 x_224 + 4.0 x_225 + 8.0 x_226 + 16.0 x_227 + 8.0 x_228 + 16.0 x_229 + 32.0
  x_230 − x_357 − 2.0 x_358 − 4.0 x_359 − 2.0 x_360 − 4.0 x_361 − 8.0 x_362
  − 4.0 x_363 − 8.0 x_364 − 16.0 x_365 <= 1.0
x_32 + x_49 − x_195 − 2.0 x_196 − 4.0 x_197 − 2.0 x_198 − 4.0 x_199 − 8.0
 x_200 − 4.0 x_201 − 8.0 x_202 − 16.0 x_203 + 2.0 x_249 + 4.0 x_250 + 8.0
 x_251 + 4.0 x_252 + 8.0 x_253 + 16.0 x_254 + 8.0 x_255 + 16.0 x_256 + 32.0
  x_257 − x_438 − 2.0 x_439 − 4.0 x_440 − 2.0 x_441 − 4.0 x_442 − 8.0 x_443
  − 4.0 x_444 − 8.0 x_445 − 16.0 x_446 <= 1.0
x_40 + x_49 − x_357 − 2.0 x_358 − 4.0 x_359 − 2.0 x_360 − 4.0 x_361 − 8.0
 x_362 − 4.0 x_363 − 8.0 x_364 − 16.0 x_365 + 2.0 x_384 + 4.0 x_385 + 8.0
 x_386 + 4.0 x_387 + 8.0 x_388 + 16.0 x_389 + 8.0 x_390 + 16.0 x_391 + 32.0
  x_392 − x_438 − 2.0 x_439 − 4.0 x_440 − 2.0 x_441 − 4.0 x_442 − 8.0 x_443
  − 4.0 x_444 − 8.0 x_445 − 16.0 x_446 <= 1.0
x_35 + x_41 − x_258 − 2.0 x_259 − 4.0 x_260 − 2.0 x_261 − 4.0 x_262 − 8.0
 x_263 − 4.0 x_264 − 8.0 x_265 − 16.0 x_266 + 2.0 x_276 + 4.0 x_277 + 8.0
 x_278 + 4.0 x_279 + 8.0 x_280 + 16.0 x_281 + 8.0 x_282 + 16.0 x_283 + 32.0
  x_284 − x_357 − 2.0 x_358 − 4.0 x_359 − 2.0 x_360 − 4.0 x_361 − 8.0 x_362
  − 4.0 x_363 − 8.0 x_364 − 16.0 x_365 <= 1.0
x_35 + x_50 − x_258 − 2.0 x_259 − 4.0 x_260 − 2.0 x_261 − 4.0 x_262 − 8.0
 x_263 − 4.0 x_264 − 8.0 x_265 − 16.0 x_266 + 2.0 x_303 + 4.0 x_304 + 8.0
 x_305 + 4.0 x_306 + 8.0 x_307 + 16.0 x_308 + 8.0 x_309 + 16.0 x_310 + 32.0
  x_311 − x_438 − 2.0 x_439 − 4.0 x_440 − 2.0 x_441 − 4.0 x_442 − 8.0 x_443
  − 4.0 x_444 − 8.0 x_445 − 16.0 x_446 <= 1.0
x_41 + x_50 − x_357 − 2.0 x_358 − 4.0 x_359 − 2.0 x_360 − 4.0 x_361 − 8.0
 x_362 − 4.0 x_363 − 8.0 x_364 − 16.0 x_365 + 2.0 x_384 + 4.0 x_385 + 8.0
 x_386 + 4.0 x_387 + 8.0 x_388 + 16.0 x_389 + 8.0 x_390 + 16.0 x_391 + 32.0
  x_392 − x_438 − 2.0 x_439 − 4.0 x_440 − 2.0 x_441 − 4.0 x_442 − 8.0 x_443
  − 4.0 x_444 − 8.0 x_445 − 16.0 x_446 <= 1.0
x_38 + x_44 − x_312 − 2.0 x_313 − 4.0 x_314 − 2.0 x_315 − 4.0 x_316 − 8.0
 x_317 − 4.0 x_318 − 8.0 x_319 − 16.0 x_320 + 2.0 x_330 + 4.0 x_331 + 8.0
```

```
   x_332 + 4.0 x_333 + 8.0 x_334 + 16.0 x_335 + 8.0 x_336 + 16.0 x_337 + 32.0
    x_338 - x_393 - 2.0 x_394 - 4.0 x_395 - 2.0 x_396 - 4.0 x_397 - 8.0 x_398
    - 4.0 x_399 - 8.0 x_400 - 16.0 x_401 <= 1.0
  x_38 + x_47 - x_312 - 2.0 x_313 - 4.0 x_314 - 2.0 x_315 - 4.0 x_316 - 8.0
   x_317 - 4.0 x_318 - 8.0 x_319 - 16.0 x_320 + 2.0 x_339 + 4.0 x_340 + 8.0
   x_341 + 4.0 x_342 + 8.0 x_343 + 16.0 x_344 + 8.0 x_345 + 16.0 x_346 + 32.0
    x_347 - x_420 - 2.0 x_421 - 4.0 x_422 - 2.0 x_423 - 4.0 x_424 - 8.0 x_425
    - 4.0 x_426 - 8.0 x_427 - 16.0 x_428 <= 1.0
  x_44 + x_47 - x_393 - 2.0 x_394 - 4.0 x_395 - 2.0 x_396 - 4.0 x_397 - 8.0
   x_398 - 4.0 x_399 - 8.0 x_400 - 16.0 x_401 + 2.0 x_402 + 4.0 x_403 + 8.0
   x_404 + 4.0 x_405 + 8.0 x_406 + 16.0 x_407 + 8.0 x_408 + 16.0 x_409 + 32.0
    x_410 - x_420 - 2.0 x_421 - 4.0 x_422 - 2.0 x_423 - 4.0 x_424 - 8.0 x_425
    - 4.0 x_426 - 8.0 x_427 - 16.0 x_428 <= 1.0

broadcast time: 3
vertices times: [0, 1, 2, 2, 3, 3, 3, 3]
broadcast tree: (1, 3) (3, 7) (1, 5) (0, 4) (2, 6) (0, 1) (0, 2)
```

listings/Q3v3.out

E Python Program to Generate QUBO instances of the broadcast problem.

```
#!/usr/bin/env sage -python
# broadcast time of graph: IP version -> QUBO Hamiltonian objective

import networkx as nx
from sage.all import *

def read_graph():
    G=nx.Graph()
    n=int(sys.stdin.readline().strip())
    for u in range(n):
        neighbors=sys.stdin.readline().split()
        for v in neighbors: G.add_edge(u,int(v))
    return G

if len(sys.argv)>1: orig=int(sys.argv[1])
else:               orig=0

G=read_graph()
n=G.order()
L=int(math.log(2*n-1,2))
print "n=",n,"; L=",L

p=MixedIntegerLinearProgram(solver="Coin",maximization=False)

s=p.new_variable(binary=True) # slack variables to add to our constraints
scnt=0
def sgen():
```

```
    global scnt
    while True:
        yield s[scnt]
        scnt+=1
svar=sgen()

# (1) total time t constraint
t=p.new_variable(binary=True)
p.add_constraint(sum([2**j*t[j]+2**j*svar.next() for j in range(L)]),  \
    min=n-1, max=n-1)

# (2) times vertices receives messages constraints
v=p.new_variable(binary=True)
for i in range(n):
    p.add_constraint(sum([2**j*(v[i,j]-t[j])+2**j*svar.next() \
        for j in range(L)]), min=0, max=0)

# (3) and (4) broadcast tree constraints
b=p.new_variable(binary=True)
for i in range(n):
    if i==orig:
        p.add_constraint(sum([b[j,i] for j in G[i]]),min=0,max=0)
    else:
        p.add_constraint(sum([b[j,i] for j in G[i]]),min=1,max=1)

# (5) parent's informed time less than child's time
z1=p.new_variable(binary=True) # b[i,j] * v[i,c]
z2=p.new_variable(binary=True) # b[i,j] * v[j,c]
for i in range(n):
    for j in G[i]:
        for c in range(L):
            p.add_constraint(b[i,j]+v[i,c]-z1[i,j,c]+svar.next(), \
                min=1,max=1)
            p.add_constraint(2*z1[i,j,c]-(b[i,j]+v[i,c])+svar.next(), \
                min=0,max=0)
            p.add_constraint(b[i,j]+v[j,c]-z2[i,j,c]+svar.next(), \
                min=1,max=1)
            p.add_constraint(2*z2[i,j,c]-(b[i,j]+v[j,c])+svar.next(), \
                min=0,max=0)
for i in range(n):
    for j in G[i]:
        p.add_constraint(b[i,j]+ \
            sum([2**k*z1[i,j,k]-2**k*z2[i,j,k]+2**k*svar.next() \
            for k in range(L)]),min=0,max=0)

# (6) informed children of a parent must have different times
z3=p.new_variable(binary=True) # v[i,c1]*v[j,c2] for {i<j} in VxV or i=j
    in V
for i in range(n):
    if i==orig: continue
    for j in range(i,n): # diff vertex case # if j<i: continue
        if j==orig: continue
        S=set(G[i]); S.intersection(set(G[j])) # must have common parent
        if len(S)==0: continue                 # or neighbor of deg >= 2
```

```
        if i==j and max([G.degree(k) for k in G[i]])<2: continue
        for c1 in range(L):
          for c2 in range(L):
             p.add_constraint(v[i,c1]+v[j,c2]-z3[i,c1,j,c2]+svar.next(), \
                min=1,max=1)
             p.add_constraint(2*z3[i,c1,j,c2]-(v[i,c1]+v[j,c2])+ \
                svar.next(),min=0,max=0)

def p3(t1,k1,t2,k2):
    if t1<=t2: return z3[t1,k1,t2,k2]
    else:      return z3[t2,k2,t1,k1]

# (6) continued...
for i in range(n):
    for jk in Combinations(sorted(G[i]),2).list():
        if orig in jk: continue # we know no b[*,orig] are true
        assert jk[0]<jk[1]
        tmp=[]
        for k in range(L):
          tmp.append((jk[0],k,1)); tmp.append((jk[1],k,-1))
        prod=[(s1*s2)*2**(k1+k2)*p3(t1,k1,t2,k2) \
          for (t1,k1,s1) in tmp for (t2,k2,s2) in tmp]
        p.add_constraint(b[i,jk[0]]+b[i,jk[1]]-sum(prod)+
    sum(2**k*svar.next() for k in range(2*L)),min=1,max=1)

p.set_objective(sum([2**j*t[j] for j in range(L)]))
p.show()
numqbits=p.number_of_variables()
print("number of logical qbits: ",numqbits)
print("number of slack qbits: ",scnt+1)
print("number of constraints: ",p.number_of_constraints())

#initialize QUBO Hamiltonian
H=[[0]*numqbits for i in range(numqbits)]

m=n
offset=0
# add IP constraint penalties
for (minc, (varidx, varconst), maxc) in p.constraints():
    assert minc==maxc
    offset+=minc*maxc*m
    for i in range(len(varidx)):
        H[varidx[i]][varidx[i]]-=2*varconst[i]*minc*m
        for j in range(len(varidx)):
            if i<=j:
              H[varidx[i]][varidx[j]]+=varconst[i]*varconst[j]*m
            else:
              H[varidx[j]][varidx[i]]+=varconst[i]*varconst[j]*m

# add objective function penalties (first even L variables X_0, X_2, ...,
   correspond to t)
for i in range(L): H[2*i][2*i]+=2**i

Q={} # dictionary version
```

```
print numqbits, int(offset)
if numqbits > 512:
  for i in range(numqbits):
    for j in range(numqbits):
        if i<=j:
            if H[i][j]!=0: Q[(i,j)]=H[i][j]
else:
  for i in range(numqbits):
    for j in range(numqbits):
        if i<=j:
            print int(H[i][j]),
            if H[i][j]!=0: Q[(i,j)]=H[i][j]
        else:
            print int(H[j][i]),
    print

print Q
```

listings/IPv5.py

F Small Test Graphs

```
# Sage script to generate many small ajacency lists of known graphs.

def printg(g):
    n=g.order()
    print n
    for u in range(n):
        for v in g[u]:
            print v,
        print

def printgg(g,d1,d2):
    n=g.order()
    print n
    for u1 in range(d1):
      for u2 in range(d2):
        for (v1,v2) in g[(u1,u2)]:
            print v1*d2+v2,
        print

def printgc(g):
    n=g.order()
    print n
    for u in sorted(g):
        for u2 in g[u]:
            print int(u2,2),
        print

#printg(graphs.CycleGraph(3)) = C3 = K3
print "C4"
```

```
printg(graphs.CycleGraph(4))
print "C5"
printg(graphs.CycleGraph(5))
print "C6"
printg(graphs.CycleGraph(6))
print "C7"
printg(graphs.CycleGraph(7))
print "C8"
printg(graphs.CycleGraph(8))
print "C9"
printg(graphs.CycleGraph(9))
print "C10"
printg(graphs.CycleGraph(10))
print "C11"
printg(graphs.CycleGraph(11))
print "C12"
printg(graphs.CycleGraph(12))

print "K2"
printg(graphs.CompleteGraph(2))
print "K3"
printg(graphs.CompleteGraph(3))
print "K4"
printg(graphs.CompleteGraph(4))
print "K5"
printg(graphs.CompleteGraph(5))
print "K6"
printg(graphs.CompleteGraph(6))
print "K7"
printg(graphs.CompleteGraph(7))
print "K8"
printg(graphs.CompleteGraph(8))
print "K9"
printg(graphs.CompleteGraph(9))
print "K10"
printg(graphs.CompleteGraph(10))

print "S2"
printg(graphs.StarGraph(2))
print "S3"
printg(graphs.StarGraph(3))
print "S4"
printg(graphs.StarGraph(4))
print "S5"
printg(graphs.StarGraph(5))
print "S6"
printg(graphs.StarGraph(6))
print "S7"
printg(graphs.StarGraph(7))
print "S8"
printg(graphs.StarGraph(8))
print "S9"
printg(graphs.StarGraph(9))
print "S10"
```

```
printg(graphs.StarGraph(10))

print "K1x2"
printg(graphs.CompleteBipartiteGraph(1,2))
#printg(graphs.CompleteBipartite(2,2)) = C4
print "K2x3"
printg(graphs.CompleteBipartiteGraph(2,3))
print "K3x3"
printg(graphs.CompleteBipartiteGraph(3,3))
print "K3x4"
printg(graphs.CompleteBipartiteGraph(3,4))
print "K4x4"
printg(graphs.CompleteBipartiteGraph(4,4))
print "K4x5"
printg(graphs.CompleteBipartiteGraph(4,5))
print "K5x5"
printg(graphs.CompleteBipartiteGraph(5,5))
print "K5x6"
printg(graphs.CompleteBipartiteGraph(5,6))
print "K6x6"
printg(graphs.CompleteBipartiteGraph(6,6))

#print "Grid Graphs"
#printg(graphs.GridGraph(2,2)) = C4
print "Grid2x3"
printgg(graphs.GridGraph([2,3]),2,3)
print "Grid3x3"
printgg(graphs.GridGraph([3,3]),3,3)
print "Grid3x4"
printgg(graphs.GridGraph([3,4]),3,4)
print "Grid4x4"
printgg(graphs.GridGraph([4,4]),4,4)
print "Grid4x5"
printgg(graphs.GridGraph([4,5]),4,5)
print "Grid5x5"
printgg(graphs.GridGraph([5,5]),5,5)

#print "Cube Graphs"
#printg(graphs.CubeGraph(2)) = C4
print "Q3"
printgc(graphs.CubeGraph(3))
print "Q4"
printgc(graphs.CubeGraph(4))

print "Hexahedral"
printg(graphs.HexahedralGraph())
print "Octahedral"
printg(graphs.OctahedralGraph())
print "Icosahedral"
printg(graphs.IcosahedralGraph())
print "Dodecahedral"
printg(graphs.DodecahedralGraph())

# small named graphs
```

```
print "BidiakisCube"
printg(graphs.BidiakisCube())
print "Bull"
printg(graphs.BullGraph())
print "Butterfly"
printg(graphs.ButterflyGraph())
print "Chvatal"
printg(graphs.ChvatalGraph())
print "Clebsch"
printg(graphs.ClebschGraph())
print "Diamond"
printg(graphs.DiamondGraph())
print "Dinneen"
print "9\n1 5 6 6\n0 6 8 2\n1 7 8 3\n2 4 6 7\n3 5 6 8"
print "0 4 7 8\n0 1 3 4 7 8\n0 2 3 5 6 8\n1 2 5 4 6 7"
print "Durer"
printg(graphs.DurerGraph())
print "Errera"
printg(graphs.ErreraGraph())
print "Frucht"
printg(graphs.FruchtGraph())
print "GoldnerHarary"
printg(graphs.GoldnerHararyGraph())
print "Grotzsch"
printg(graphs.GrotzschGraph())
print "Heawood"
printg(graphs.HeawoodGraph())
print "Herschel"
printg(graphs.HerschelGraph())
print "Hoffman"
printg(graphs.HoffmanGraph())
print "House"
printg(graphs.HouseGraph())
print "KrackhardtKite"
printg(graphs.KrackhardtKiteGraph())
print "Pappus"
printg(graphs.PappusGraph())
print "Petersen"
printg(graphs.PetersenGraph())
print "Poussin"
printg(graphs.PoussinGraph())
print "Robertson"
printg(graphs.RobertsonGraph())
print "Shrikhande"
printg(graphs.ShrikhandeGraph())
print "Sousselier"
printg(graphs.SousselierGraph())
print "Tietze"
printg(graphs.TietzeGraph())
print "Wagner"
printg(graphs.WagnerGraph())
```

listings/sagegraphs.py