Applications of Reformulations in Mathematical Programming

Alberto Costa

LIX, École Polytechnique, Palaiseau, France

September 18th, 2012

Ph.D. Thesis defense

Outline

Outline

Outline

Outline

Outline

Where are we?

Motivations

- Mathematical Programming: describe (by means of a Mathematical Programming formulation) and solve optimization problems;

Motivations

- Mathematical Programming: describe (by means of a Mathematical Programming formulation) and solve optimization problems;
- given a problem, different formulations can be proposed: reformulations;

Motivations

- Mathematical Programming: describe (by means of a Mathematical Programming formulation) and solve optimization problems;
- given a problem, different formulations can be proposed: reformulations;
- Objective: starting from the original formulation for a problem, propose some reformulations which are somehow "better" (i.e., less time to obtain the optimal solution).

Liberti's Classification of Reformulations

Let P the original problem and Q a reformulation, and f_P and f_Q be respectively their objective functions. Q can be:

Liberti's Classification of Reformulations

Let P the original problem and Q a reformulation, and f_P and f_Q be respectively their objective functions. Q can be:

- exact or opt-reformulation: local (global) optima of P correspond to local (global) optima of Q;

Liberti's Classification of Reformulations

Let P the original problem and Q a reformulation, and f_P and f_Q be respectively their objective functions. Q can be:

- exact or opt-reformulation: local (global) optima of P correspond to local (global) optima of Q;
- narrowing: each global optimum of Q corresponds to a global optimum of P (Q can have fewer global optimum than P);

Liberti's Classification of Reformulations

Let P the original problem and Q a reformulation, and f_P and f_Q be respectively their objective functions. Q can be:

- exact or opt-reformulation: local (global) optima of P correspond to local (global) optima of Q;
- narrowing: each global optimum of Q corresponds to a global optimum of P (Q can have fewer global optimum than P);
- relaxation: the feasible region of P is a subset of the feasible region of Q, and in case of minimization problem $f_Q(x) \leq f_P(x)$ for x in the feasible region of P.

Problems studied

For each kind of reformulation, a problem is studied:

Problems studied

For each kind of reformulation, a problem is studied:

- exact or opt-reformulation: clustering by means of modularity maximization in general and bipartite graphs;

Problems studied

For each kind of reformulation, a problem is studied:

- exact or opt-reformulation: clustering by means of modularity maximization in general and bipartite graphs;
- narrowing: circle packing in a square;

Problems studied

For each kind of reformulation, a problem is studied:

- exact or opt-reformulation: clustering by means of modularity maximization in general and bipartite graphs;
- narrowing: circle packing in a square;
- relaxation: convex relaxations for multilinear terms.

Where are we?

Clustering in graphs

Graph $G = (V, E)$

- V: set of n vertices;
- E: set of m edges connecting pairs of vertices.

Clustering in graphs

Graph $G = (V, E)$

- V: set of n vertices;
- E: set of m edges connecting pairs of vertices.

Goal: one seeks clusters which contains more inner edges (vertices in the same cluster) than cut edges (vertices in different clusters).

Clustering in graphs

Graph $G = (V, E)$

- V: set of n vertices;
- E: set of m edges connecting pairs of vertices.

Goal: one seeks clusters which contains more inner edges (vertices in the same cluster) than cut edges (vertices in different clusters).

Modularity [Newman, Girvan; Phys. Rev. E, 2004]

Find a partition of V into clusters, maximizing the number of inner edges minus the expected number of such edges in a random graph having the same distribution of degrees of G.

Modularity

Modularity [Newman, Girvan; Phys. Rev. E, 2004]

$$Q = \sum_{c=1}^{N_c} \left(\frac{m_c}{m} - \frac{{D_c}^2}{4m^2} \right)$$

Modularity

Modularity [Newman, Girvan; Phys. Rev. E, 2004]

$$Q = \sum_{c=1}^{N_c} \left(\frac{m_c}{m} - \frac{{D_c}^2}{4m^2} \right)$$

- N_c: number of clusters;

Modularity

Modularity [Newman, Girvan; Phys. Rev. E, 2004]

$$Q = \sum_{c=1}^{N_c} \left(\frac{m_c}{m} - \frac{{D_c}^2}{4m^2} \right)$$

- N_c: number of clusters;
- m: number of edges of the graph;

Modularity

Modularity [Newman, Girvan; Phys. Rev. E, 2004]

$$Q = \sum_{c=1}^{N_c} \left(\frac{m_c}{m} - \frac{{D_c}^2}{4m^2} \right)$$

- N_c: number of clusters;
- m: number of edges of the graph;
- m_c: number of edges in cluster c;

Modularity

Modularity [Newman, Girvan; Phys. Rev. E, 2004]

$$Q = \sum_{c=1}^{N_c} \left(\frac{m_c}{m} - \frac{{D_c}^2}{4m^2} \right)$$

- N_c: number of clusters;
- m: number of edges of the graph;
- m_c: number of edges in cluster c;
- D_c: sum of degrees of vertices in cluster c;

Modularity

Modularity [Newman, Girvan; Phys. Rev. E, 2004]

$$Q = \sum_{c=1}^{N_c} \left(\frac{m_c}{m} - \frac{{D_c}^2}{4m^2} \right)$$

- N_c: number of clusters;
- m: number of edges of the graph;
- m_c: number of edges in cluster c;
- D_c: sum of degrees of vertices in cluster c;
- $\frac{m_c}{m}$: fraction of edges in cluster c;

Modularity

Modularity [Newman, Girvan; Phys. Rev. E, 2004]

$$Q = \sum_{c=1}^{N_c} \left(\frac{m_c}{m} - \frac{{D_c}^2}{4m^2} \right)$$

- N_c: number of clusters;
- m: number of edges of the graph;
- m_c: number of edges in cluster c;
- D_c: sum of degrees of vertices in cluster c;
- $\frac{m_c}{m}$: fraction of edges in cluster c;
- $\frac{{D_c}^2}{4m^2}$: expected number of edges in cluster c in a graph where vertices have same degrees but edges are placed randomly.

Locally optimal hierarchical divisive heuristic

In this thesis we focus on a hierarchical divisive heuristic [Cafieri, Hansen, Liberti; Phys. Rev. E, 2011].

Locally optimal hierarchical divisive heuristic

In this thesis we focus on a hierarchical divisive heuristic [Cafieri, Hansen, Liberti; Phys. Rev. E, 2011].

Algorithm Divisive (input of first call is $G = (V, E)$)

- Input: cluster $c = (V_c, E_c)$ of graph G

Locally optimal hierarchical divisive heuristic

In this thesis we focus on a hierarchical divisive heuristic [Cafieri, Hansen, Liberti; Phys. Rev. E, 2011].

Algorithm Divisive (input of first call is $G = (V, E)$)

- Input: cluster $c = (V_c, E_c)$ of graph G
- Output: partition into clusters of c

Locally optimal hierarchical divisive heuristic

In this thesis we focus on a hierarchical divisive heuristic [Cafieri, Hansen, Liberti; Phys. Rev. E, 2011].

Algorithm Divisive (input of first call is $G = (V, E)$)

- Input: cluster $c = (V_c, E_c)$ of graph G
- Output: partition into clusters of c
- if $|V_c| \leq 3$ save c as cluster, and return;

Locally optimal hierarchical divisive heuristic

In this thesis we focus on a hierarchical divisive heuristic [Cafieri, Hansen, Liberti; Phys. Rev. E, 2011].

Algorithm Divisive (input of first call is $G = (V, E)$)

- Input: cluster $c = (V_c, E_c)$ of graph G
- Output: partition into clusters of c
- if $|V_c| \leq 3$ save c as cluster, and return;
- divide c in c_1 and c_2 in an optimal way (maximizing modularity using a $0-1$ MIQP model for bipartition);

Locally optimal hierarchical divisive heuristic

In this thesis we focus on a hierarchical divisive heuristic [Cafieri, Hansen, Liberti; Phys. Rev. E, 2011].

Algorithm Divisive (input of first call is $G = (V, E)$)

- Input: cluster $c = (V_c, E_c)$ of graph G
- Output: partition into clusters of c
- if $|V_c| \leq 3$ save c as cluster, and return;
- divide c in c_1 and c_2 in an optimal way (maximizing modularity using a $0-1$ MIQP model for bipartition);
- if $Q(c) > Q(c_1) + Q(c_2)$ save c as cluster, and return;

Locally optimal hierarchical divisive heuristic

In this thesis we focus on a hierarchical divisive heuristic [Cafieri, Hansen, Liberti; Phys. Rev. E, 2011].

Algorithm Divisive (input of first call is $G = (V, E)$)

- Input: cluster $c = (V_c, E_c)$ of graph G
- Output: partition into clusters of c
- if $|V_c| \leq 3$ save c as cluster, and return;
- divide c in c_1 and c_2 in an optimal way (maximizing modularity using a $0-1$ MIQP model for bipartition);
- if $Q(c) > Q(c_1) + Q(c_2)$ save c as cluster, and return;
- call Divisive(c_1) and Divisive(c_2);

$0-1$ MIQP model used by the hierarchical divisive heuristic - 1

Objective function (split the cluster c into two clusters; $D_c = D_1 + D_2$ is known before solving the problem)

$$Q = \left(\frac{m_1 + m_2}{m} - \frac{{D_1}^2 + {D_2}^2}{4m^2} \right) = \left(\frac{m_1 + m_2}{m} - \frac{2{D_1}^2 + {D_c}^2 - 2D_1 D_c}{4m^2} \right)$$

$0-1$ MIQP model used by the hierarchical divisive heuristic - 1

Objective function (split the cluster c into two clusters; $D_c = D_1 + D_2$ is known before solving the problem)

$$Q = \left(\frac{m_1 + m_2}{m} - \frac{{D_1}^2 + {D_2}^2}{4m^2} \right) = \left(\frac{m_1 + m_2}{m} - \frac{2{D_1}^2 + {D_c}^2 - 2D_1 D_c}{4m^2} \right)$$

Variables

- $X_{i,j,s} = 1$ if the edge (v_i, v_j) is inside the cluster s, 0 otherwise (s is either 1 or 2);

$0-1$ MIQP model used by the hierarchical divisive heuristic - 1

Objective function (split the cluster c into two clusters; $D_c = D_1 + D_2$ is known before solving the problem)

$$Q = \left(\frac{m_1 + m_2}{m} - \frac{{D_1}^2 + {D_2}^2}{4m^2} \right) = \left(\frac{m_1 + m_2}{m} - \frac{2{D_1}^2 + {D_c}^2 - 2D_1 D_c}{4m^2} \right)$$

Variables

- $X_{i,j,s} = 1$ if the edge (v_i, v_j) is inside the cluster s, 0 otherwise (s is either 1 or 2);
- $Y_i = 1$ if the vertex v_i is inside the cluster 1, 0 otherwise;

$0-1$ MIQP model used by the hierarchical divisive heuristic - 1

Objective function (split the cluster c into two clusters; $D_c = D_1 + D_2$ is known before solving the problem)

$$Q = \left(\frac{m_1 + m_2}{m} - \frac{{D_1}^2 + {D_2}^2}{4m^2} \right) = \left(\frac{m_1 + m_2}{m} - \frac{2{D_1}^2 + {D_c}^2 - 2D_1 D_c}{4m^2} \right)$$

Variables

- $X_{i,j,s} = 1$ if the edge (v_i, v_j) is inside the cluster s, 0 otherwise (s is either 1 or 2);
- $Y_i = 1$ if the vertex v_i is inside the cluster 1, 0 otherwise;
- k_i is the degree of the vertex v_i.

$0-1$ MIQP model used by the hierarchical divisive heuristic - 2 (OB model)

$$\max \quad \frac{1}{m}\left(m_1 + m_2 - \frac{1}{2m}\left({D_1}^2 + \frac{{D_c}^2}{2} - D_1 D_c\right)\right)$$

$$\text{s.t.} \quad X_{i,j,1} \leq Y_i \qquad \forall (v_i, v_j) \in E_c$$

$$X_{i,j,1} \leq Y_j \qquad \forall (v_i, v_j) \in E_c$$

$$X_{i,j,2} \leq 1 - Y_i \qquad \forall (v_i, v_j) \in E_c$$

$$X_{i,j,2} \leq 1 - Y_j \qquad \forall (v_i, v_j) \in E_c$$

$$m_s = \sum_{(v_i, v_j) \in E_c} X_{i,j,s} \qquad \forall s \in \{1, 2\}$$

$$D_1 = \sum_{v_i \in V_c} k_i Y_i$$

$$Y_i \in \{0, 1\} \qquad \forall v_i \in V_c$$

$$X_{i,j,s} \geq 0 \qquad \forall (v_i, v_j) \in E_c, \forall s \in \{1, 2\}$$

Improving the $0-1$ MIQP formulation

- reduction of number of variables and constraints;

Improving the $0-1$ MIQP formulation

- reduction of number of variables and constraints;
- symmetry breaking.

Reduction of number of variables

Consider the variables X of the original model:

$$X_{i,j,s} = \begin{cases} 1, & \text{if edge } (v_i, v_j) \text{ belongs to cluster } s, \\ 0, & \text{otherwise.} \end{cases}$$

Reduction of number of variables

Consider the variables X of the original model:

$$X_{i,j,s} = \begin{cases} 1, & \text{if edge } (v_i, v_j) \text{ belongs to cluster } s, \\ 0, & \text{otherwise.} \end{cases}$$

We do not actually need to know if an edge is in the cluster 1 or 2, but only if it is within a cluster or not:

$$X_{i,j} = \begin{cases} 1, & \text{if } Y_i = Y_j, \\ 0, & \text{otherwise.} \end{cases}$$

Reduction of number of variables

Consider the variables X of the original model:

$$X_{i,j,s} = \begin{cases} 1, & \text{if edge } (v_i, v_j) \text{ belongs to cluster } s, \\ 0, & \text{otherwise.} \end{cases}$$

We do not actually need to know if an edge is in the cluster 1 or 2, but only if it is within a cluster or not:

$$X_{i,j} = \begin{cases} 1, & \text{if } Y_i = Y_j, \\ 0, & \text{otherwise.} \end{cases}$$

Half of the variables X needed.

New variables

Variables X can then be expressed as

$$X_{i,j} = 2Y_iY_j - Y_i - Y_j + 1, \qquad \forall (v_i, v_j) \in E_c.$$

New variables

Variables X can then be expressed as

$$X_{i,j} = 2Y_iY_j - Y_i - Y_j + 1, \qquad \forall (v_i, v_j) \in E_c.$$

Variables S linearize the product of the binary variables Y:

$$S_{i,j} = Y_iY_j, \qquad \forall (v_i, v_j) \in E_c.$$

New variables

Variables X can then be expressed as

$$X_{i,j} = 2Y_iY_j - Y_i - Y_j + 1, \qquad \forall (v_i, v_j) \in E_c.$$

Variables S linearize the product of the binary variables Y:

$$S_{i,j} = Y_iY_j, \qquad \forall (v_i, v_j) \in E_c.$$

So we obtain

$$X_{i,j} = 2S_{i,j} - Y_i - Y_j + 1, \qquad \forall (v_i, v_j) \in E_c.$$

Fortet linearization

Relationship $S_{i,j} = Y_i Y_j$ (Fortet inequalities):

$$
\begin{aligned}
& S_{i,j} \geq 0 && \forall (v_i, v_j) \in E_c \\
& S_{i,j} \geq Y_j + Y_i - 1 && \forall (v_i, v_j) \in E_c \\
& S_{i,j} \leq Y_i && \forall (v_i, v_j) \in E_c \\
& S_{i,j} \leq Y_j && \forall (v_i, v_j) \in E_c.
\end{aligned}
$$

Fortet linearization

Relationship $S_{i,j} = Y_i Y_j$ (Fortet inequalities):

$$
\begin{aligned}
& S_{i,j} \geq 0 && \forall (v_i, v_j) \in E_c \\
& S_{i,j} \geq Y_j + Y_i - 1 && \forall (v_i, v_j) \in E_c \\
& S_{i,j} \leq Y_i && \forall (v_i, v_j) \in E_c \\
& S_{i,j} \leq Y_j && \forall (v_i, v_j) \in E_c.
\end{aligned}
$$

Objective function maximizes variables $S \rightarrow$ half of the constraints needed:

$$
\begin{aligned}
& S_{i,j} \leq Y_i && \forall (v_i, v_j) \in E_c \\
& S_{i,j} \leq Y_j && \forall (v_i, v_j) \in E_c.
\end{aligned}
$$

OB_1 formulation

$$\max \quad \frac{1}{m}\left(\sum_{(v_i,v_j)\in E_c}(2S_{i,j}-Y_i-Y_j)+|E_c|-\frac{1}{2m}\left({D_1}^2+\frac{{D_c}^2}{2}-D_1D_c\right)\right)$$

$$\text{s.t.} \quad S_{i,j}\leq Y_i \qquad \forall(v_i,v_j)\in E_c$$

$$S_{i,j}\leq Y_j \qquad \forall(v_i,v_j)\in E_c$$

$$D_1=\sum_{v_i\in V_c}k_iY_i$$

$$Y_i\in\{0,1\} \qquad \forall v_i\in V_c,$$

OB_1 formulation

$$\max \quad \frac{1}{m}\left(\sum_{(v_i,v_j)\in E_c}(2S_{i,j}-Y_i-Y_j)+|E_c|-\frac{1}{2m}\left({D_1}^2+\frac{{D_c}^2}{2}-D_1D_c\right)\right)$$

$$\text{s.t.} \quad S_{i,j}\le Y_i \qquad \forall(v_i,v_j)\in E_c$$

$$S_{i,j}\le Y_j \qquad \forall(v_i,v_j)\in E_c$$

$$D_1=\sum_{v_i\in V_c}k_iY_i$$

$$Y_i\in\{0,1\} \qquad \forall v_i\in V_c,$$

where in the objective function we use the fact that $\sum_{(v_i,v_j)\in E_C} 1=|E_c|$.

Symmetry breaking constraint - Fixing a vertex

If a solution is found, another equivalent solution is obtained by swapping the clusters (i.e., vertices in cluster 1 are placed in cluster 2, and vice-versa). $\rightarrow$ fix a vertex in one of the clusters.

Symmetry breaking constraint - Fixing a vertex

If a solution is found, another equivalent solution is obtained by swapping the clusters (i.e., vertices in cluster 1 are placed in cluster 2, and vice-versa). $\rightarrow$ fix a vertex in one of the clusters.

Good choice: fix the vertex with highest degree in one cluster.

$$Y_g = 0, \quad g = \arg\max\{k_i, \forall v_i \in V_c\}.$$

Numerical results

Tests: 2.8GHz Intel iCore i7 CPU, 8 GB RAM, Linux, CPLEX 12.2 [IBM; 2010]

Numerical results

Tests: 2.8GHz Intel iCore i7 CPU, 8 GB RAM, Linux, CPLEX 12.2 [IBM; 2010]

graph			OB		$OB_1 + SBC$	
	vertices	edges	nodes	CPU time	nodes	CPU time
Karate	34	78	45	0.14	**17**	**0.04**
Dolphins	62	159	207	0.59	**93**	**0.16**
Les Misérables	77	254	205	1.09	**105**	**0.35**
A00 main	83	135	76	0.35	**26**	**0.04**
P53 protein	104	226	275	1.10	**119**	**0.26**
Political books	105	441	313	3.04	**152**	**0.51**
Football	115	613	8853	307.56	**3822**	**44.38**
A01 main	249	635	1119	47.83	**726**	**9.72**
USAir97	332	2126	16682	4585.04	**8665**	**446.06**
Netscience main	379	914	291	3.64	**94**	**0.85**
S838	512	819	392	5.26	**186**	**1.18**
Power	4941	6594	1459	708.51	**891**	**123.85**

Bipartite graphs

For bipartite graphs the definition of modularity is the following

Bipartite Modularity [Barber; Pys. Rev. E, 2007; Leicht, Newman; Phys. Rev. Lett., 2008]

$$Q_b = \sum_{c=1}^{N_c} \left(\frac{m_c}{m} - \frac{R_c B_c}{m^2} \right)$$

Bipartite graphs

For bipartite graphs the definition of modularity is the following

Bipartite Modularity [Barber; Pys. Rev. E, 2007; Leicht, Newman; Phys. Rev. Lett., 2008]

$$Q_b = \sum_{c=1}^{N_c} \left(\frac{m_c}{m} - \frac{R_c B_c}{m^2} \right)$$

- R_c: sum of degrees of red vertices in cluster c;

Bipartite graphs

For bipartite graphs the definition of modularity is the following

Bipartite Modularity [Barber; Pys. Rev. E, 2007; Leicht, Newman; Phys. Rev. Lett., 2008]

$$Q_b = \sum_{c=1}^{N_c} \left(\frac{m_c}{m} - \frac{R_c B_c}{m^2} \right)$$

- R_c: sum of degrees of red vertices in cluster c;
- B_c: sum of degrees of blue vertices in cluster c;

Bipartite graphs

For bipartite graphs the definition of modularity is the following

Bipartite Modularity [Barber; Pys. Rev. E, 2007; Leicht, Newman; Phys. Rev. Lett., 2008]

$$Q_b = \sum_{c=1}^{N_c} \left(\frac{m_c}{m} - \frac{R_c B_c}{m^2} \right)$$

- R_c: sum of degrees of red vertices in cluster c;
- B_c: sum of degrees of blue vertices in cluster c;
- all edges have a red and a blue end vertices.

Bipartite divisive heuristic

We adapt the divisive heuristic to the bipartite case $\rightarrow P$ model:

$$
\begin{aligned}
\max \quad & \frac{1}{m}\left(\sum_{(v_i,v_j)\in E_c}(2S_{i,j}-Y_i-Y_j)+|E_c|-\frac{1}{m}\left(2R_1B_1-B_cR_1-R_cB_1+R_cB_c\right)\right) \\
\text{s.t.} \quad & S_{i,j}\leq Y_i \qquad \forall(v_i,v_j)\in E_c \\
& S_{i,j}\leq Y_j \qquad \forall(v_i,v_j)\in E_c \\
& R_1=\sum_{v_i\in V_{R_c}} k_iY_i \\
& B_1=\sum_{v_j\in V_{B_c}} k_jY_j \\
& Y_g=1, \quad g=\arg\max\{k_i,\ \forall v_i\in V_c\} \\
& Y_i\in\{0,1\} \qquad \forall v_i\in V_c,
\end{aligned}
$$

V_{R_c} and V_{B_c} are respectively the sets of red and blue vertices, and $V_c = V_{R_c} \cup V_{B_c}$.

Fortet linearizations

Nonlinear model: $R_1 B_1$ in the objective function.

Fortet linearizations

Nonlinear model: $R_1 B_1$ in the objective function.
One can apply the Fortet linearization for $R_1 B_1 \rightarrow P_{1a}$ model.

Fortet linearizations

Nonlinear model: $R_1 B_1$ in the objective function.

One can apply the Fortet linearization for $R_1 B_1 \rightarrow P_{1a}$ model.
A more compact formulation is possible $\rightarrow P_{1b}$

$$
\begin{aligned}
\max \quad & \frac{1}{m} \sum_{v_i \in V_{R_c}} \sum_{v_j \in V_{B_c}} H_{i,j} \left(2W_{i,j} - Y_i - Y_j + 1\right) \\
\text{s.t.} \quad & W_{i,j} \geq 0 \quad \forall v_i \in V_{R_c}, \forall v_j \in V_{B_c} \; : \; H_{i,j} < 0 \\
& W_{i,j} \geq Y_i + Y_j - 1 \quad \forall v_i \in V_{R_c}, \forall v_j \in V_{B_c} \; : \; H_{i,j} < 0 \\
& W_{i,j} \leq Y_i \quad \forall v_i \in V_{R_c}, \forall v_j \in V_{B_c} \; : \; H_{i,j} > 0 \\
& W_{i,j} \leq Y_j \quad \forall v_i \in V_{R_c}, \forall v_j \in V_{B_c} \; : \; H_{i,j} > 0 \\
& Y_g = 1, \quad g = \arg\max\{k_i, \forall v_i \in V_{R_c} \cup V_{B_c}\} \\
& Y_i \in \{0,1\} \quad \forall v_i \in V_{R_c} \cup V_{B_c}.
\end{aligned}
$$

$H_{i,j} = T_{i,j} - \frac{k_i k_j}{m}$, and $T_{i,j} = 1$ if there exists the edge (i,j), 0 otherwise.

Binary decomposition

$$R_1 = \sum_{v_i \in V_{R_c}} k_i Y_i = \sum_{h=0}^{t_R} 2^h a_h$$

$$B_1 = \sum_{v_j \in V_{B_c}} k_j Y_j = \sum_{l=0}^{t_B} 2^l b_l$$

$$R_1 B_1 = \sum_{h=0}^{t_R} 2^h a_h \sum_{l=0}^{t_B} 2^l b_l = \sum_{h=0}^{t_R} \sum_{l=0}^{t_B} 2^{l+h} a_h b_l$$

Binary decomposition

$$R_1 = \sum_{v_i \in V_{R_c}} k_i Y_i = \sum_{h=0}^{t_R} 2^h a_h$$

$$B_1 = \sum_{v_j \in V_{B_c}} k_j Y_j = \sum_{l=0}^{t_B} 2^l b_l$$

$$R_1 B_1 = \sum_{h=0}^{t_R} 2^h a_h \sum_{l=0}^{t_B} 2^l b_l = \sum_{h=0}^{t_R} \sum_{l=0}^{t_B} 2^{l+h} a_h b_l$$

each product $a_l b_h$ is then linearized using the Fortet inequalities $\rightarrow P_2$ model

Numerical results

Tests: 2.8GHz Intel iCore i7 CPU, 8 GB RAM, Linux, CPLEX 12.2 [IBM; 2010]

Numerical results

Tests: 2.8GHz Intel iCore i7 CPU, 8 GB RAM, Linux, CPLEX 12.2 [IBM; 2010]

graph				P_{1a}		P_{1b}		P_2	
	red vertices	total vertices	edges	nodes	time	nodes	time	nodes	time
1	18	32	89	437	0.30	**72**	**0.19**	670	0.39
2	26	35	147	154	0.19	**10**	**0.09**	618	0.43
3	26	35	86	45	0.14	**6**	**0.07**	183	0.19
4	18	36	99	2169	1.46	**1360**	1.24	1854	**0.93**
5	26	41	98	1963	1.25	**276**	0.44	647	**0.39**
6	50	59	225	1123	0.77	**27**	**0.16**	2521	2.12
7	62	102	192	1223370	4440.04	407104	3038.06	**38910**	**5.26**
8	108	244	358	-	-	-	-	**3793**	**5.81**
9	314	674	613	-	-	-	-	**71927548**	**15450.40**
10	960	2549	2580	-	-	-	-	**91917**	**38.49**

Clustering based on strong and almost-strong conditions

- Not related with modularity maximization;

Clustering based on strong and almost-strong conditions

- Not related with modularity maximization;
- Community in the strong sense [Radicchi et al.; PNAS, 2004]: a subset S of vertices where the number of neighbors of each vertex within S is larger than the number of neighbors outside S.

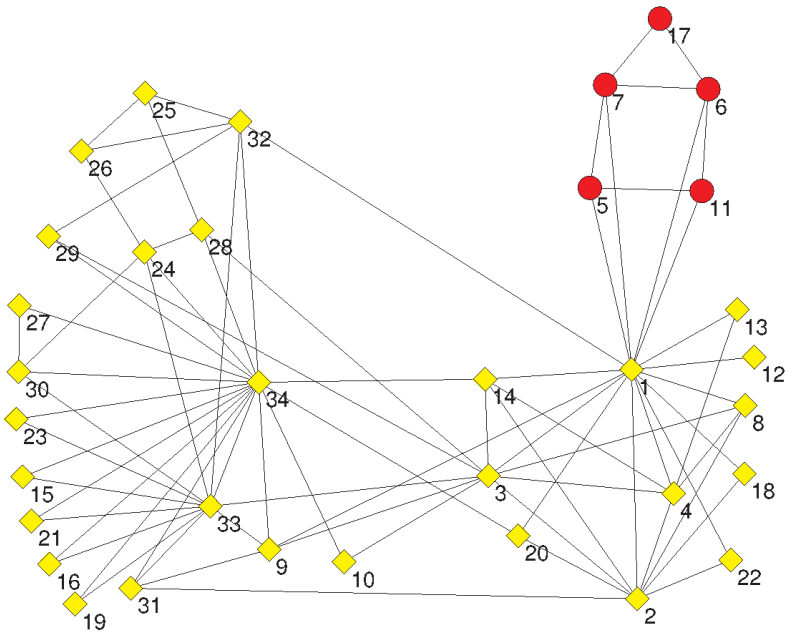
Clustering based on strong and almost-strong conditions

- Not related with modularity maximization;
- Community in the strong sense [Radicchi et al.; PNAS, 2004]: a subset S of vertices where the number of neighbors of each vertex within S is larger than the number of neighbors outside S.
- Strong conditions can be too stringent $\rightarrow$ we propose the almost-strong conditions: same definition as the strong conditions, except for degree 2 vertices, for which the number of neighbors within S is larger or equal to the number of neighbors outside S;

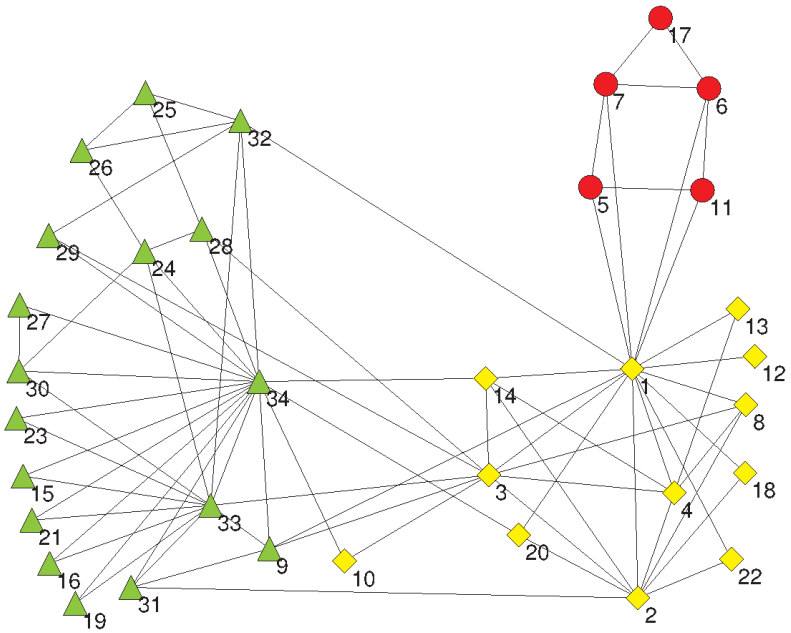
Clustering based on strong and almost-strong conditions

- Not related with modularity maximization;
- Community in the strong sense [Radicchi et al.; PNAS, 2004]: a subset S of vertices where the number of neighbors of each vertex within S is larger than the number of neighbors outside S.
- Strong conditions can be too stringent $\rightarrow$ we propose the almost-strong conditions: same definition as the strong conditions, except for degree 2 vertices, for which the number of neighbors within S is larger or equal to the number of neighbors outside S;
- We designed an algorithm to find strong and almost-strong communities in graphs, and we compare the results.

Test 1 - Zachary karate club - strong vs almost-strong

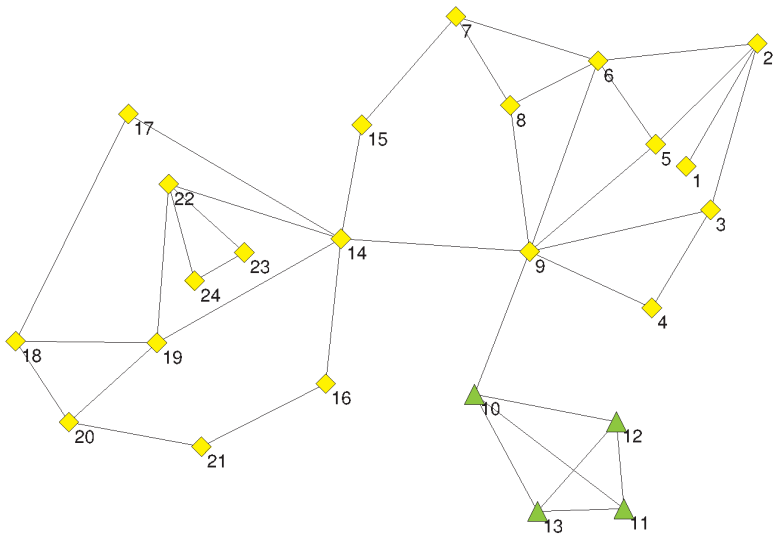


(a)

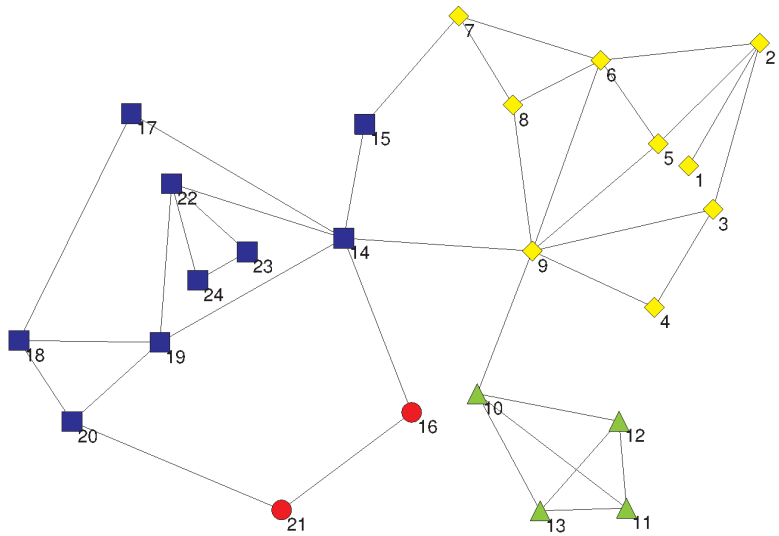


(b)

Test 2- strike - strong vs almost-strong

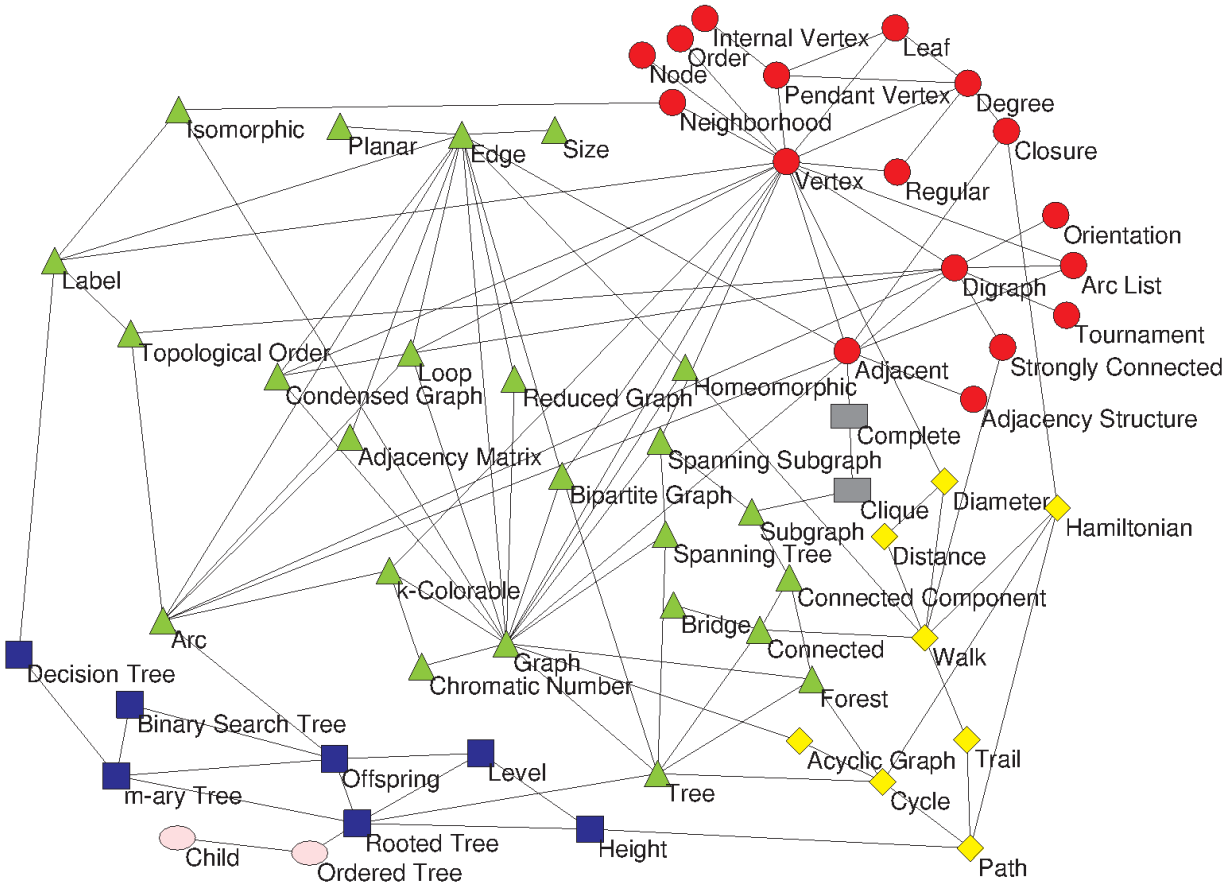


(c)

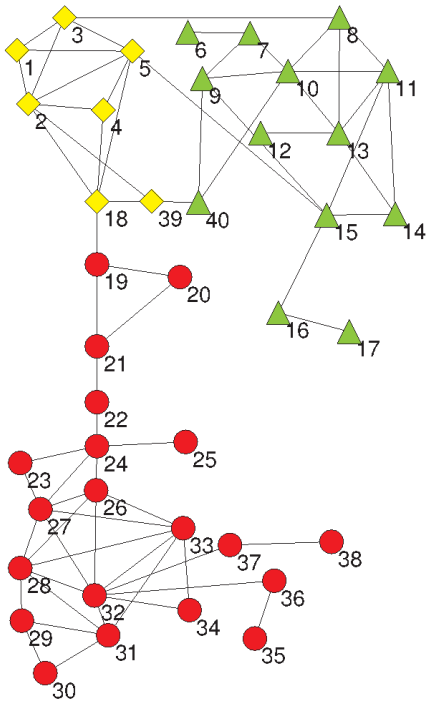


(d)

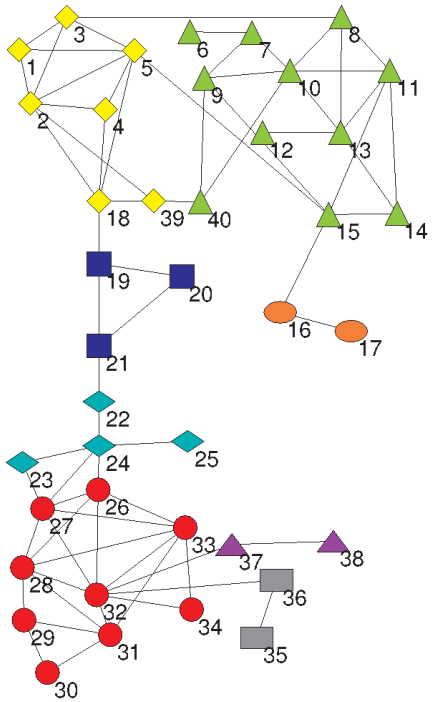
Test 3 - graph - almost strong (strong: trivial partition)



Test 4 - dolphins small - strong vs almost-strong



(e)



(f)

Where are we?

The problem: Packing Equal Circles in a Square (PECS)

Consider the following problem: Place $n \in \mathbb{N}$ non-overlapping circles of radius $r \in \mathbb{R}$ in the unit square such that the radius is maximized.

The problem: Packing Equal Circles in a Square (PECS)

Consider the following problem: Place $n \in \mathbb{N}$ non-overlapping circles of radius $r \in \mathbb{R}$ in the unit square such that the radius is maximized.

Non-linear Non-convex formulation

$$\begin{aligned} \max \quad & r \\ \text{s.t.} \quad & (x_i - x_j)^2 + (y_i - y_j)^2 \geq 4r^2 \quad \forall i < j \leq n \\ & x_i, y_i \in [r, 1-r] \quad \forall i \leq n, \end{aligned}$$

The problem: Packing Equal Circles in a Square (PECS)

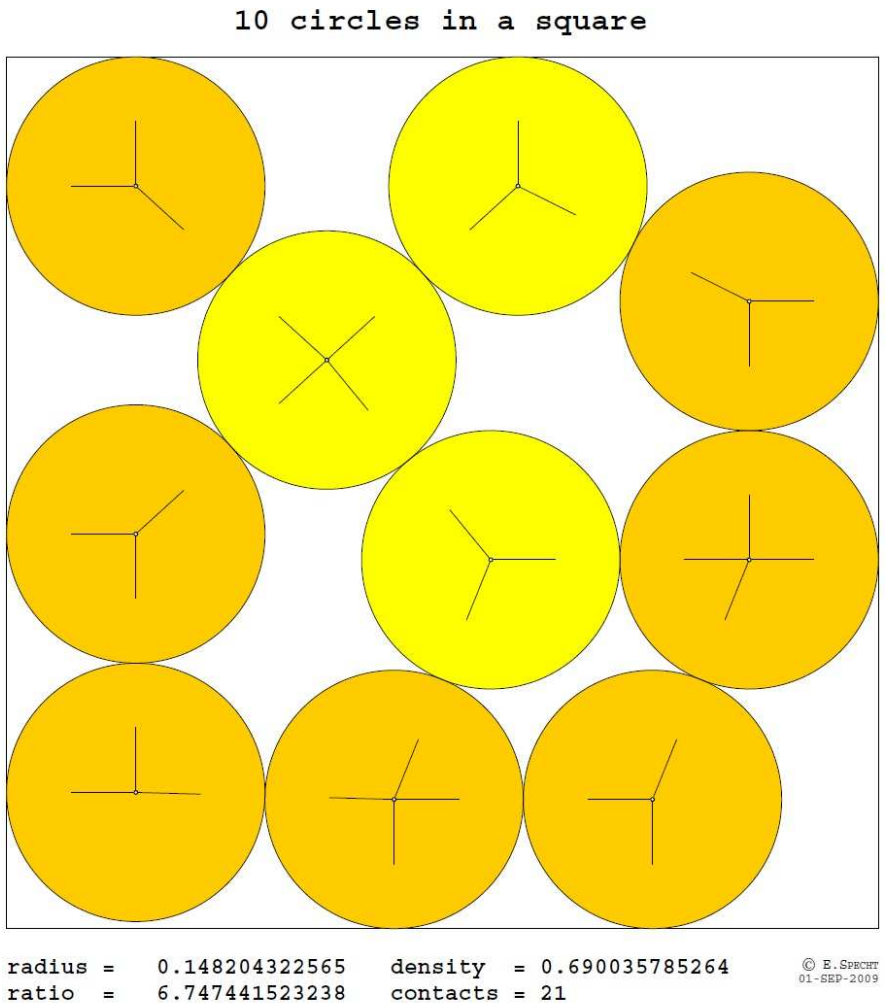
Consider the following problem: Place $n \in \mathbb{N}$ non-overlapping circles of radius $r \in \mathbb{R}$ in the unit square such that the radius is maximized.

Non-linear Non-convex formulation

$$\begin{aligned} \max \quad & r \\ \text{s.t.} \quad & (x_i - x_j)^2 + (y_i - y_j)^2 \geq 4r^2 \quad \forall i < j \leq n \\ & x_i, y_i \in [r, 1-r] \quad \forall i \leq n, \end{aligned}$$

where (x_i, y_i) represents the coordinates of the center of the i-th circle, and $r \geq 0$ is the common radius to maximize.

Example: optimal solution of PECS with 10 circles



Applications

- cutting problems (cut out as many identical disks as possible from a piece of material);

Applications

- cutting problems (cut out as many identical disks as possible from a piece of material);
- container loading (place as many identical objects as possible into a container);

Applications

- cutting problems (cut out as many identical disks as possible from a piece of material);
- container loading (place as many identical objects as possible into a container);
- tree exploitation (plant trees in a given region maximizing both the density and the size of trees);

Applications

- cutting problems (cut out as many identical disks as possible from a piece of material);
- container loading (place as many identical objects as possible into a container);
- tree exploitation (plant trees in a given region maximizing both the density and the size of trees);
- cheese packing!

Point Packing in a Square (PPS)

Consider the following problem: Place $n \in \mathbb{N}$ points in the unit square such that the minimum pairwise distance d^* is maximal.

Point Packing in a Square (PPS)

Consider the following problem: Place $n \in \mathbb{N}$ points in the unit square such that the minimum pairwise distance d^* is maximal.

Non-linear Non-convex formulation

$$
\begin{aligned}
\max \quad & \alpha \\
\text{s.t.} \quad & (x_i - x_j)^2 + (y_i - y_j)^2 \geq \alpha \quad \forall i < j \leq n \\
& x_i \in [0,1] \quad \forall i \leq n \\
& y_i \in [0,1] \quad \forall i \leq n \\
& \alpha \geq 0
\end{aligned}
$$

Point Packing in a Square (PPS)

Consider the following problem: Place $n \in \mathbb{N}$ points in the unit square such that the minimum pairwise distance d^* is maximal.

Non-linear Non-convex formulation

$$
\begin{aligned}
\max \quad & \alpha \\
\text{s.t.} \quad & (x_i - x_j)^2 + (y_i - y_j)^2 \geq \alpha \quad \forall i < j \leq n \\
& x_i \in [0,1] \quad \forall i \leq n \\
& y_i \in [0,1] \quad \forall i \leq n \\
& \alpha \geq 0
\end{aligned}
$$

where (x_i, y_i) represents the coordinates of the i-th point and $d^* = \sqrt{\alpha^*}$.

Relationship between PECS and PPS

A point belongs to an edge in PPS $\Leftrightarrow$ the corresponding center is at distance r from that edge in PECS.

Relationship between PECS and PPS

A point belongs to an edge in PPS $\Leftrightarrow$ the corresponding center is at distance r from that edge in PECS.

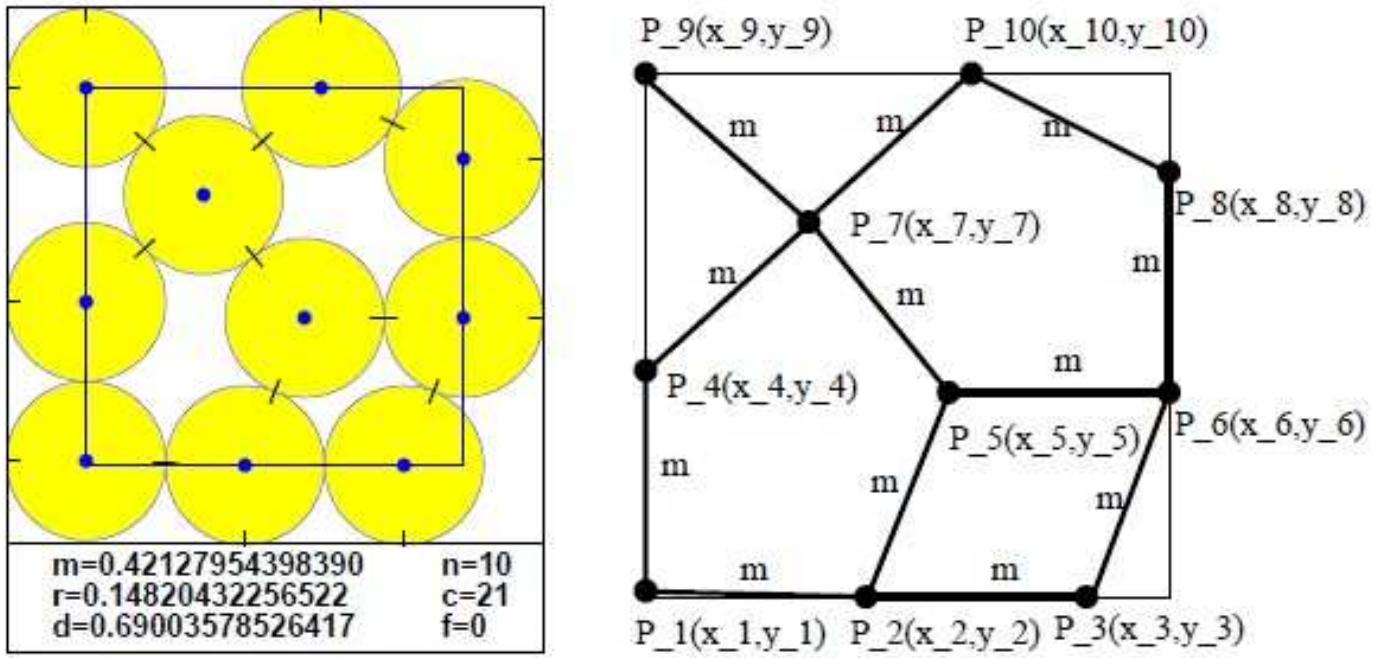


Figure: Relationship between PECS and PPS (figure taken from [Szabó; Contributions to Algebra and Geometry, 2005]).

Narrowing in CPS

Problem

CPS has a lot of symmetric global optima. Branch-and-Bound algorithms do not work very efficiently in this situation, because the BB tree is large.

Narrowing in CPS

Problem

CPS has a lot of symmetric global optima. Branch-and-Bound algorithms do not work very efficiently in this situation, because the BB tree is large.

Possible solution

Removing some of the global optima, by adjoining some Symmetry Breaking Constraints (SBCs) $\rightarrow$ narrowing reformulation.

BB trees

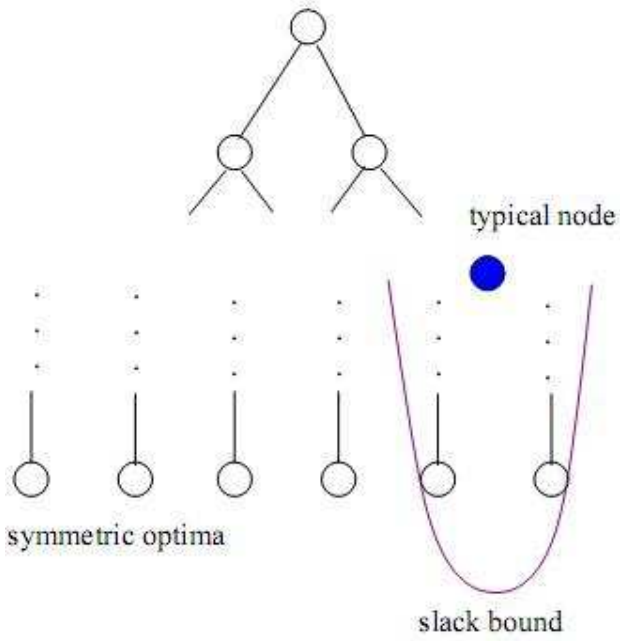


Figure: Original Formulation

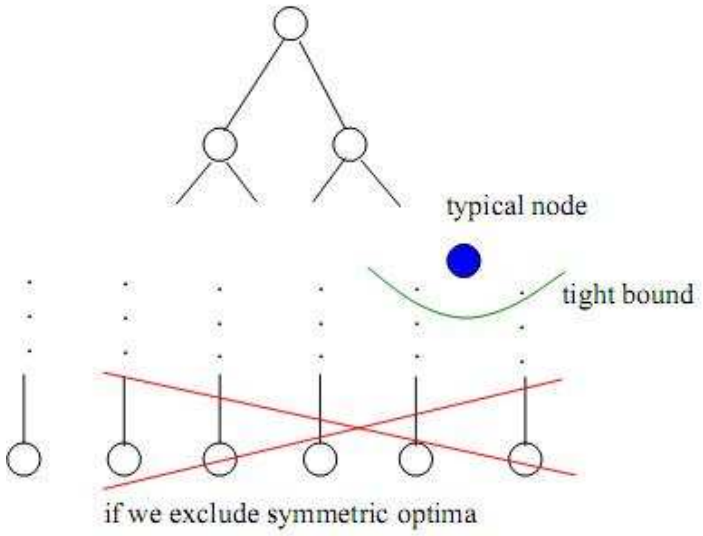


Figure: Narrowing Reformulation

Symmetries in Circle Packing

It is proved that the formulation group (class of symmetries which can be computed from the mathematical model of the problem) of CPS is isomorphic to C_2 x S_n, where:

Symmetries in Circle Packing

It is proved that the formulation group (class of symmetries which can be computed from the mathematical model of the problem) of CPS is isomorphic to C_2 x S_n, where:

- C_2 represents the permutation between x and y axes.

Symmetries in Circle Packing

It is proved that the formulation group (class of symmetries which can be computed from the mathematical model of the problem) of CPS is isomorphic to C_2 x S_n, where:

- C_2 represents the permutation between x and y axes.
- S_n represents the permutation of the circle indicies (we can swap some circles, and the solution does not change).

SBCs (Symmetry Breaking Constraints)

In order to eliminate some global optima, we adjoin these constraints (that give an order on the variables) [Hansen, C., Liberti; ISCO10]:

SBCs (Symmetry Breaking Constraints)

In order to eliminate some global optima, we adjoin these constraints (that give an order on the variables) `[Hansen, C., Liberti; ISCO10]`:

- *weak* constraints: $x_1 \leq x_i, \forall i \in \{2, \ldots, n\}$

SBCs (Symmetry Breaking Constraints)

In order to eliminate some global optima, we adjoin these constraints (that give an order on the variables) [Hansen, C., Liberti; ISCO10]:

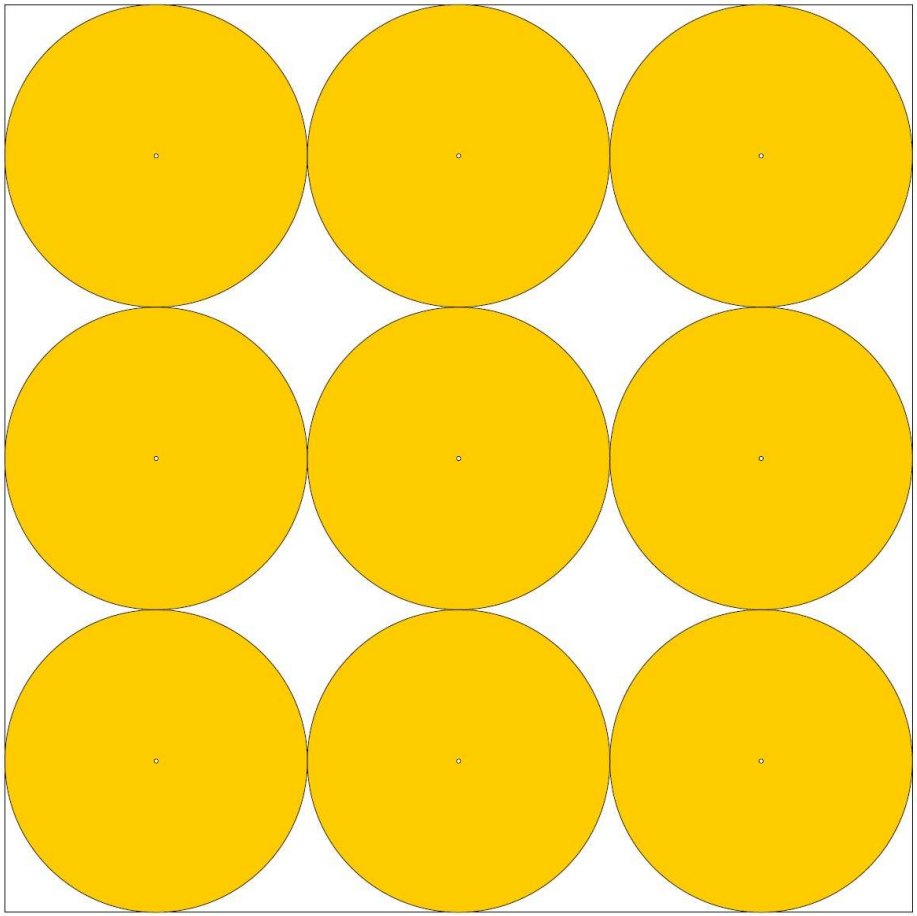
- *weak* constraints: $x_1 \leq x_i, \forall i \in \{2, \ldots, n\}$
- *strong* constraints: $x_i \leq x_{i+1}, \forall i \in \{1, \ldots, n-1\}$

SBCs (Symmetry Breaking Constraints)

In order to eliminate some global optima, we adjoin these constraints (that give an order on the variables) [Hansen, C., Liberti; ISCO10]:

- *weak* constraints: $x_1 \leq x_i, \forall i \in \{2, \ldots, n\}$
- *strong* constraints: $x_i \leq x_{i+1}, \forall i \in \{1, \ldots, n-1\}$
- *mixed* constraints, introduced in [C., Liberti, Hansen; DAM, 2012], that mix contraints on the x and y variables.

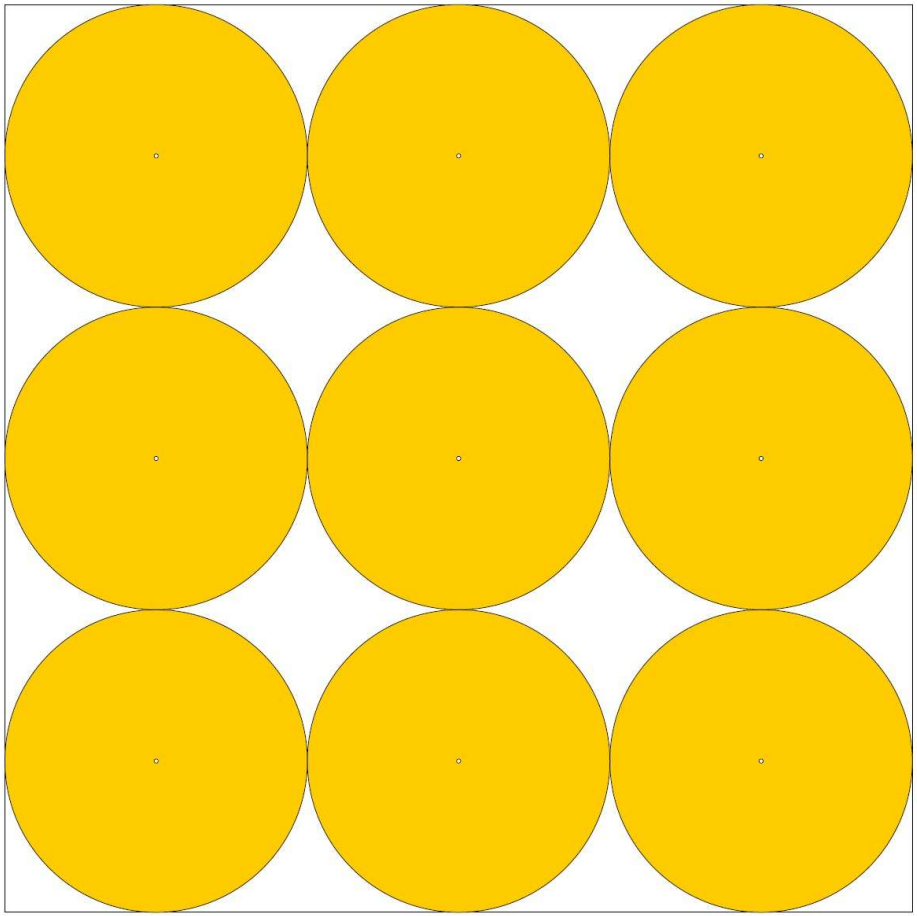
Example - $n = 9$



weak constraints

$x_1 \leq x_2,\ x_1 \leq x_3, \ldots, x_1 \leq x_9$

Example - $n = 9$



weak constraints

$$x_1 \leq x_2,\ x_1 \leq x_3, \ldots, x_1 \leq x_9$$

strong constraints

$$x_1 \leq x_2,\ x_2 \leq x_3, \ldots, x_8 \leq x_9$$

Mixed SBCs

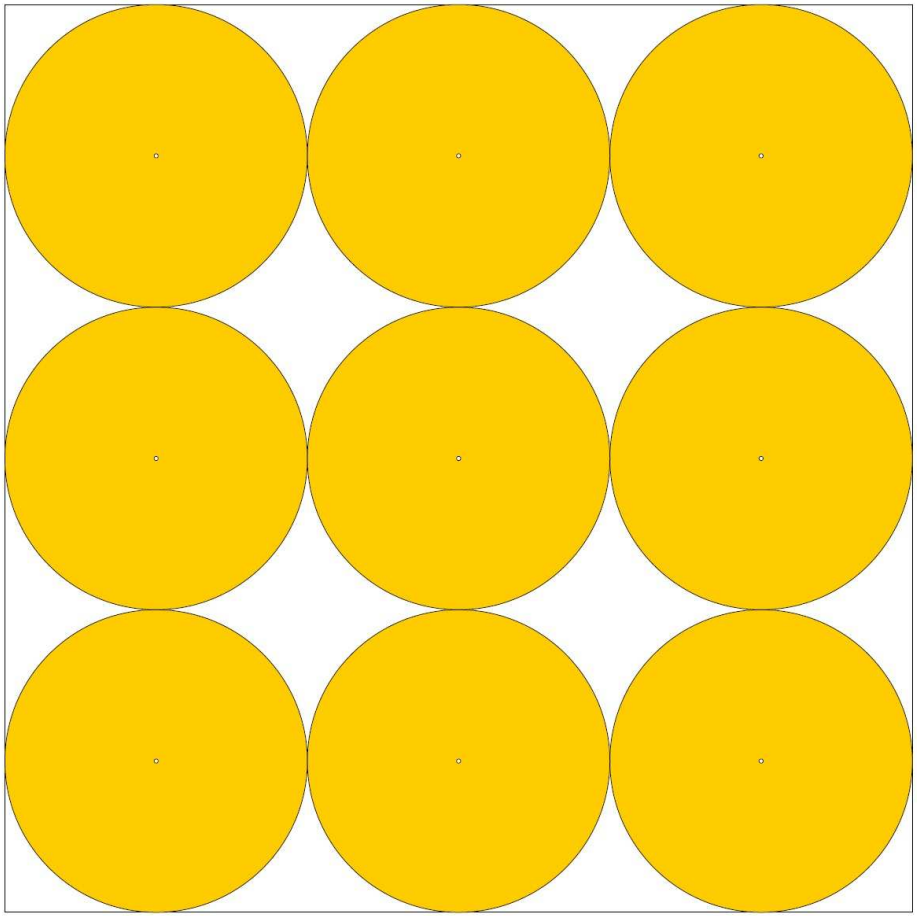
Idea: *strong* constraints give some conditions only for the x coordinates of the centres of the circles; it would be better to have also some conditions for the y coordinates.

Mixed SBCs

Idea: *strong* constraints give some conditions only for the x coordinates of the centres of the circles; it would be better to have also some conditions for the y coordinates.

Starting from the *strong* constraints, we replace $x_{iS} \leq x_{iS+1}$ with $y_{1+(i-1)S} \leq y_{1+iS}$, $\forall i \in \left\{1, 2, \ldots, \left\lceil \frac{N}{S} \right\rceil - 1\right\}$ (best results with $S = 2$).

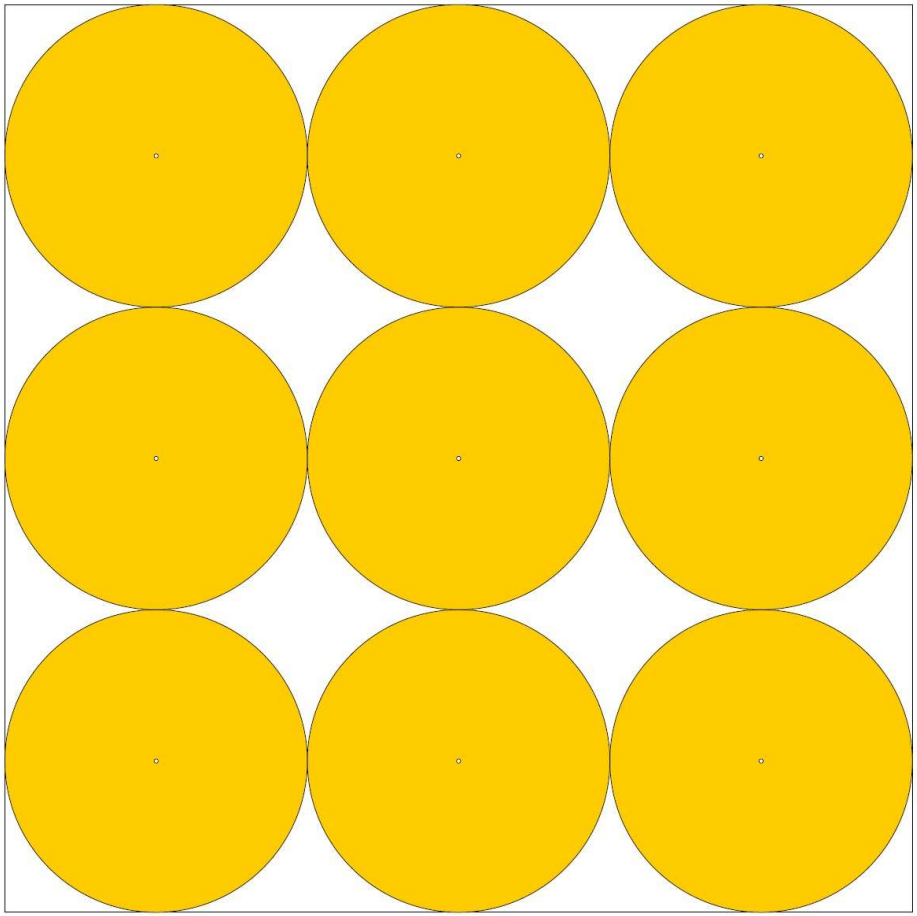
Strong and mixed SBCs, $S = 3$



strong constraints

$$x_1 \leq x_2,\ x_2 \leq x_3,\ \mathbf{x_3} \leq \mathbf{x_4},$$
$$x_4 \leq x_5,\ x_5 \leq x_6,\ \mathbf{x_6} \leq \mathbf{x_7},$$
$$x_7 \leq x_8,\ x_8 \leq x_9$$

Strong and mixed SBCs, $S = 3$



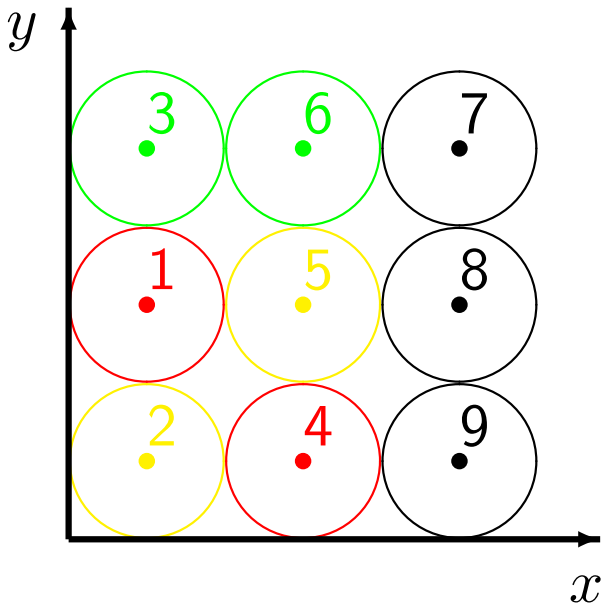
strong constraints

$$x_1 \leq x_2,\ x_2 \leq x_3,\ \mathbf{x_3} \leq \mathbf{x_4},$$
$$x_4 \leq x_5,\ x_5 \leq x_6,\ \mathbf{x_6} \leq \mathbf{x_7},$$
$$x_7 \leq x_8,\ x_8 \leq x_9$$

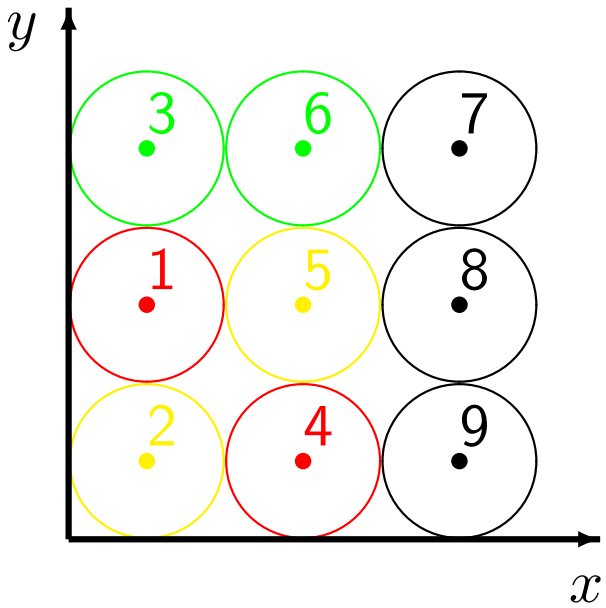
mixed constraints

$$x_1 \leq x_2,\ x_2 \leq x_3,\ \mathbf{y_1} \leq \mathbf{y_4},$$
$$x_4 \leq x_5,\ x_5 \leq x_6,\ \mathbf{y_4} \leq \mathbf{y_7},$$
$$x_7 \leq x_8,\ x_8 \leq x_9$$

Why mixed SBCs are valid? - Example



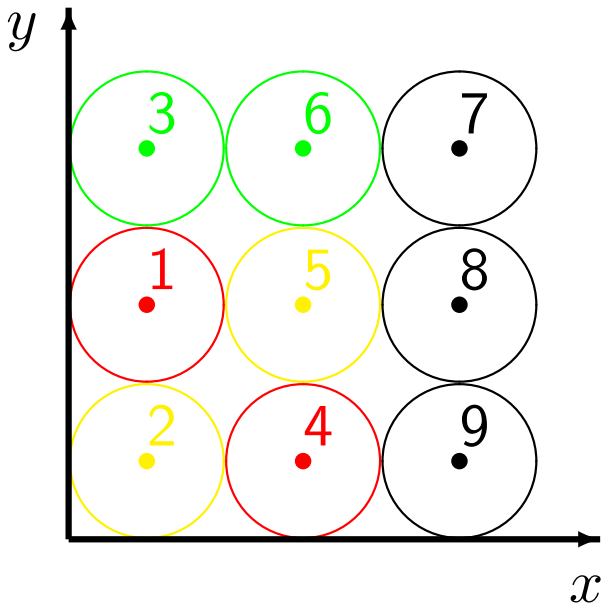
Why mixed SBCs are valid? - Example

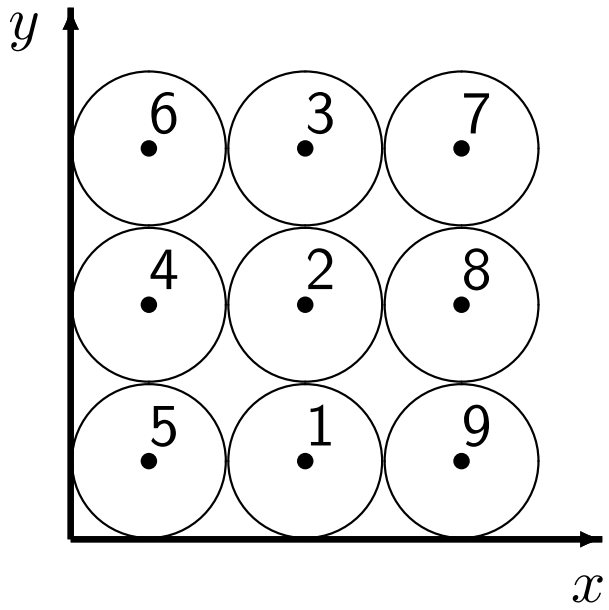


This solution respects the *strong* constraints, but not the *mixed* constraints.

$$x_1 \leq x_2,\ x_2 \leq x_3,\ \mathbf{y_1 \leq y_4},$$
$$x_4 \leq x_5,\ x_5 \leq x_6,\ \mathbf{y_4 \leq y_7},$$
$$x_7 \leq x_8,\ x_8 \leq x_9$$

Why mixed SBCs are valid? - Example

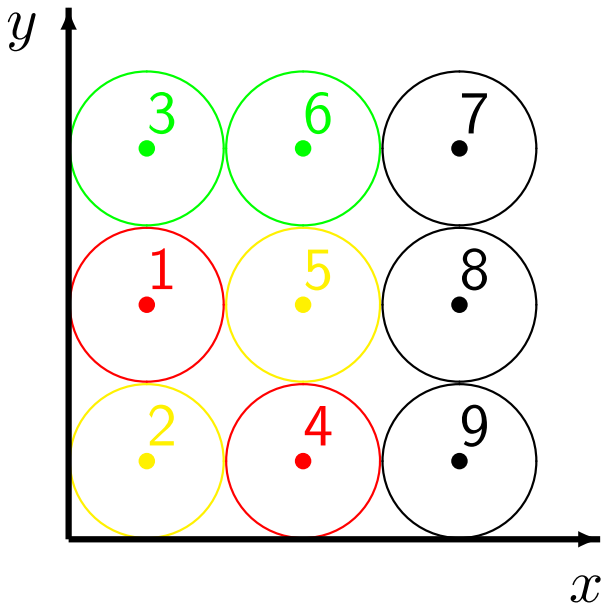




This solution respects the *strong* constraints, but not the *mixed* constraints.

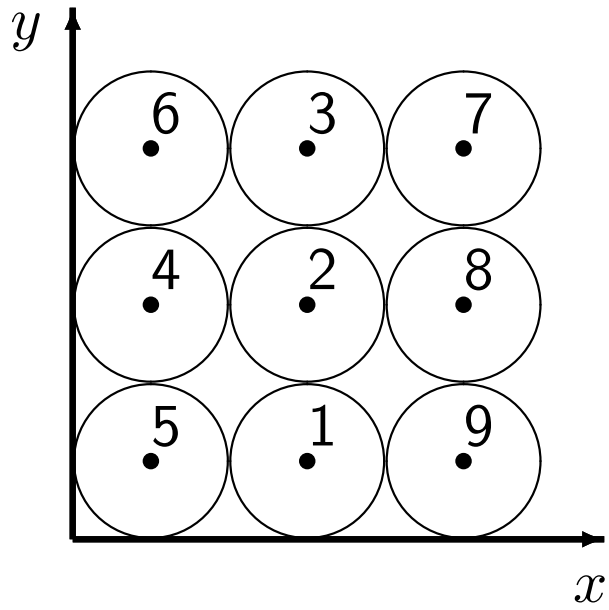
$$x_1 \leq x_2,\ x_2 \leq x_3,\ \mathbf{y_1 \leq y_4},$$
$$x_4 \leq x_5,\ x_5 \leq x_6,\ \mathbf{y_4 \leq y_7},$$
$$x_7 \leq x_8,\ x_8 \leq x_9$$

Why mixed SBCs are valid? - Example



This solution respects the *strong* constraints, but not the *mixed* constraints.

$$x_1 \leq x_2,\ x_2 \leq x_3,\ \mathbf{y_1 \leq y_4},$$
$$x_4 \leq x_5,\ x_5 \leq x_6,\ \mathbf{y_4 \leq y_7},$$
$$x_7 \leq x_8,\ x_8 \leq x_9$$



Now, after the swapping, the solution respects the *mixed* constraints.

$$x_1 \leq x_2,\ x_2 \leq x_3,\ \mathbf{y_1 \leq y_4},$$
$$x_4 \leq x_5,\ x_5 \leq x_6,\ \mathbf{y_4 \leq y_7},$$
$$x_7 \leq x_8,\ x_8 \leq x_9$$

Some results

- strong constraints better than weak ones;

Some results

- strong constraints better than weak ones;
- mixed constraints better than strong ones.

Some results

- strong constraints better than weak ones;
- mixed constraints better than strong ones.

Mixed constraints results: COUENNE solver on a 2.4 GHz Intel Xeon CPU with 24 GB RAM running Linux.

n	r^*	r_r	r'	$\bar{r}$	$t(r')$	sBB nodes
20	0.111382	0.111382	0.111382	0.322063	16.45	441828
25	0.1	0.096852	0.1	0.250133	553.68	125632
30	0.091671	0.091671	0.091671	0.316273	86.24	90230
35	0.084290	0.082786	0.083766	0.351545	1495.31	46162
40	0.079186	0.078913	0.078913	0.2501	19.68	17116
45	0.074727	0.07444	0.07444	0.353325	357.90	12915
50	0.071377	0.070539	0.070539	0.250121	5429.88	2

Statistics: the best known solution r^*, the solution found at the root node r_r, the largest radius r' found by our method within the time limit, the tightest upper bound $\bar{r}$ on r', the time $t(r')$ at which the solution r' was found and the number of nodes explored within the time limit.

Conjecture about the bounds on the variables

Consider PPS: the linear relaxation computed at the root node does not provide good bounds because of the bounds of the variables x and y.

Conjecture about the bounds on the variables

Consider PPS: the linear relaxation computed at the root node does not provide good bounds because of the bounds of the variables x and y.

The real problem is that all the variables have the same lower and upper bounds (i.e., respectively, 0 and 1).

Linear relaxation of PPS - 1

Proposition

The optimal solution of the linear relaxation of PPS is always $\alpha^* = 2$.

Linear relaxation of PPS - 1

Proposition

The optimal solution of the linear relaxation of PPS is always $\alpha^* = 2$.

This means that for all the instances (that is, for all the values of n number of points), the Upper Bound obtained as solution at the root node is always the same, even if the optimal value of α obviously decreases when n increases.

Proof - 1

Let L_{x_i}, U_{x_i}, L_{y_i} and U_{x_i} be respectively the lower and upper bounds for the variables x_i and y_i. The linear relaxation of PPS is ([Locatelli, Raber; Tech. Rep. 09/99]):

Proof - 1

Let L_{x_i}, U_{x_i}, L_{y_i} and U_{x_i} be respectively the lower and upper bounds for the variables x_i and y_i. The linear relaxation of PPS is ([Locatelli, Raber; Tech. Rep. 09/99]):

Linear relaxation of PPS

$$
\begin{aligned}
\max \quad & \alpha \\
\text{s.t.} \quad & -l(i,j) \geq \alpha \quad \forall i < j \leq n \\
& x_i \in [0,1] \quad \forall i \leq n \\
& y_i \in [0,1] \quad \forall i \leq n \\
& \alpha \geq 0
\end{aligned}
$$

Proof - 1

Let L_{x_i}, U_{x_i}, L_{y_i} and U_{x_i} be respectively the lower and upper bounds for the variables x_i and y_i. The linear relaxation of PPS is ([Locatelli, Raber; Tech. Rep. 09/99]):

Linear relaxation of PPS

$$\begin{aligned} \max \quad & \alpha \\ \text{s.t.} \quad & -l(i,j) \geq \alpha \quad \forall i < j \leq n \\ & x_i \in [0,1] \quad \forall i \leq n \\ & y_i \in [0,1] \quad \forall i \leq n \\ & \alpha \geq 0 \end{aligned}$$

and $l(i,j) = -(L_{x_i} - U_{x_j} + U_{x_i} - L_{x_j})(x_i - x_j) - (L_{y_i} - U_{y_j} + U_{y_i} - L_{y_j})(y_i - y_j) + (L_{x_i} - U_{x_j})(U_{x_i} - L_{x_j}) + (L_{y_i} - U_{y_j})(U_{y_i} - L_{y_j})$ is the linearization of the nonlinear distance constraints.

Proof - 2

Since $L_{x_i} = L_{y_i} = 0, \forall i \leq n$ and $U_{x_i} = U_{y_i} = 1, \forall i \leq n$, we obtain $l(i,j) = -2, \forall i < j \leq n$.

Proof - 2

Since $L_{x_i} = L_{y_i} = 0, \forall i \leq n$ and $U_{x_i} = U_{y_i} = 1, \forall i \leq n$, we obtain $l(i,j) = -2, \forall i < j \leq n$. The model can be rewritten as

Linear relaxation of PPS

$$\begin{aligned} \max \quad & \alpha \\ \text{s.t.} \quad & 2 \geq \alpha \\ & x_i \in [0,1] \quad \forall i \leq n \\ & y_i \in [0,1] \quad \forall i \leq n \\ & \alpha \geq 0 \end{aligned}$$

Proof - 2

Since $L_{x_i} = L_{y_i} = 0, \forall i \leq n$ and $U_{x_i} = U_{y_i} = 1, \forall i \leq n$, we obtain $l(i,j) = -2, \forall i < j \leq n$. The model can be rewritten as

Linear relaxation of PPS

$$\begin{aligned} \max \quad & \alpha \\ \text{s.t.} \quad & 2 \geq \alpha \\ & x_i \in [0,1] \quad \forall i \leq n \\ & y_i \in [0,1] \quad \forall i \leq n \\ & \alpha \geq 0 \end{aligned}$$

the optimal solution is obviously $\alpha^* = 2$, and it does not depend on the value of the variables x and y.

Considerations on the bound

Considerations on the bound

- Upper bound $d_{UB} = \sqrt{2}$. Not good: it is the optimal solution when $n = 2$ (2 points placed in the opposite vertices).

Considerations on the bound

- Upper bound $d_{UB} = \sqrt{2}$. Not good: it is the optimal solution when $n = 2$ (2 points placed in the opposite vertices).
- This value does not depend on n, x, y: all the coefficients of x and y are 0 in $l(x, y)$.

Considerations on the bound

- Upper bound $d_{UB} = \sqrt{2}$. Not good: it is the optimal solution when $n = 2$ (2 points placed in the opposite vertices).
- This value does not depend on n, x, y: all the coefficients of x and y are 0 in $l(x, y)$.
- In order to improve this bound, we should change the bounds on some variables.

Conjecture about bounds for the variables

We present the following conjecture (easy to see that is is true, but not easy to prove)

Conjecture about bounds for the variables

We present the following conjecture (easy to see that is is true, but not easy to prove)

Conjecture

Consider an instance of PPS with n points. Divide the unit square in k^2 equal subsquares, with $k = \arg\min_s \left|\frac{n}{2} - s^2\right|$, $s \in \left\{\left\lceil \sqrt{\frac{n}{2}} \right\rceil, \left\lfloor \sqrt{\frac{n}{2}} \right\rfloor\right\}$. There is at least one point of the optimal solution in each subsquare.

Conjecture about bounds for the variables

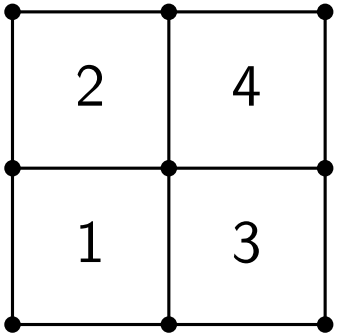
We present the following conjecture (easy to see that is is true, but not easy to prove)

Conjecture

Consider an instance of PPS with n points. Divide the unit square in k^2 equal subsquares, with $k = \arg\min_s \left|\frac{n}{2} - s^2\right|$, $s \in \left\{\left\lceil\sqrt{\frac{n}{2}}\right\rceil, \left\lfloor\sqrt{\frac{n}{2}}\right\rfloor\right\}$. There is at least one point of the optimal solution in each subsquare.

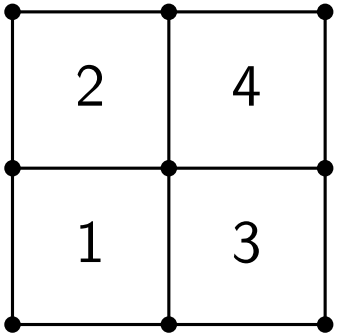
This means that we can modify the bounds for k^2 variables.

Example - $n = 9$



Consider the example with $n = 9$. In this case, $k = 2$. So we can divide the square in 4 subsquares, and in each of them there is a point of the optimal solution.

Example - $n = 9$



Consider the example with $n = 9$. In this case, $k = 2$. So we can divide the square in 4 subsquares, and in each of them there is a point of the optimal solution.

The new bounds becomes:

$$x_1 \in [0, 0.5],\ y_1 \in [0, 0.5]$$
$$x_2 \in [0, 0.5],\ y_2 \in [0.5, 1]$$
$$x_3 \in [0.5, 1],\ y_3 \in [0, 0.5]$$
$$x_4 \in [0.5, 1],\ y_4 \in [0.5, 1]$$

while the bounds for the other variables remain 0 and 1.

Tests

The tests were performed on one 2.4GHz Intel Xeon CPU of a computer with 24 GB RAM running Linux, using the solver COUENNE [Belotti, Lee, Liberti, Margot; 2009].

		Original formulation		Bounds constraints formulation	
n	d^*	LB	UB	LB	UB
9	0.5	0.000098	1.414213	**0.300463**	**0.707107**
10	0.421279	0.000098	1.414213	**0.396156**	**0.707107**
11	0.398207	0.000099	1.414213	0.000099	**0.707107**
12	0.388730	0.000099	1.414213	**0.360065**	**0.707107**
13	0.366096	0.000098	1.414213	**0.339654**	**0.502948**
14	0.348915	0.000098	1.414213	**0.340830**	**0.502874**
15	0.341081	0.000098	1.414213	**0.334524**	**0.502793**
16	0.333333	0	1.414213	**0.290033**	**0.502793**
17	0.306153	0	1.414213	**0.000099**	**0.502793**
18	0.300462	0	1.414213	**0.252819**	**0.502793**
19	0.289541	0.000047	1.414213	**0.252337**	**0.502793**
20	0.286611	0	1.414213	**0.276468**	**0.502793**

Statistics (root node)

- opt. sol. d^*
- best sol. LB
- opt. sol. of linear relaxation UB

Where are we?

Definitions

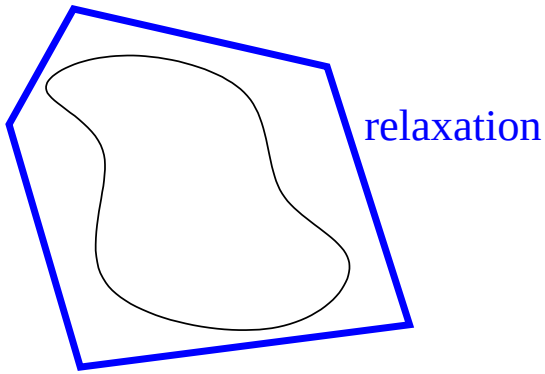
- Let $S \subseteq \mathbb{R}^n$ be non-empty

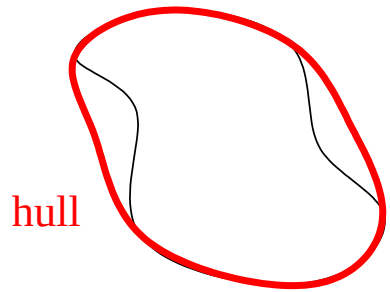
Definitions

- Let $S \subseteq \mathbb{R}^n$ be non-empty
- Any convex set containing S is a **convex relaxation** of S

Definitions

- Let $S \subseteq \mathbb{R}^n$ be non-empty
- Any convex set containing S is a **convex relaxation** of S
- The **convex hull** $\mathrm{conv}(S)$ of S is the intersection of all convex relaxations of S





Relaxing problems having multilinear terms

Conisder a problem involving multilinear terms (i.e., product of variables). In order to obtain its convex relaxation, we compare two methods:

Relaxing problems having multilinear terms

Conisder a problem involving multilinear terms (i.e., product of variables). In order to obtain its convex relaxation, we compare two methods:

- primal relaxation: each multilinear term is replaced by a new variable, and a set of linear constraints (convex envelopes) is adjoined, thus defining the convex hull;

Relaxing problems having multilinear terms

Conisder a problem involving multilinear terms (i.e., product of variables). In order to obtain its convex relaxation, we compare two methods:

- primal relaxation: each multilinear term is replaced by a new variable, and a set of linear constraints (convex envelopes) is adjoined, thus defining the convex hull;
- dual relaxation: the convex hull is represented as the convex combination of its extreme points.

Primal relaxation

- For the general case, convex envelopes for multilinear terms are available explicitly in function of x^L, x^U for $k = 2, 3$ and partly $k = 4$

Primal relaxation

- For the general case, convex envelopes for multilinear terms are available explicitly in function of x^L, x^U for $k = 2, 3$ and partly $k = 4$
- They consist of sets of constraints to be adjoined to the Mathematical Programming formulation

Primal relaxation

- For the general case, convex envelopes for multilinear terms are available explicitly in function of x^L, x^U for $k = 2, 3$ and partly $k = 4$
- They consist of sets of constraints to be adjoined to the Mathematical Programming formulation
- No further variables are needed

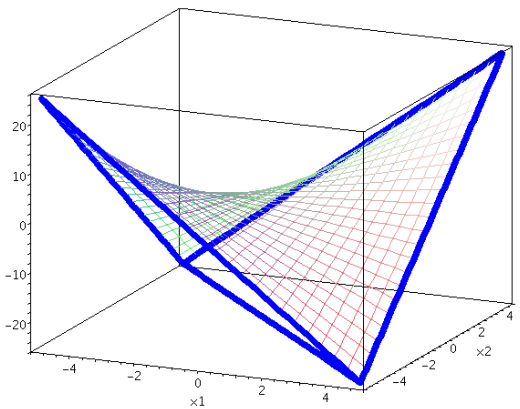
Bilinear terms: McCormick's inequalities

- Let $W = \{(w, x_1, x_2) \mid w = x_1 x_2 \wedge (x_1, x_2) = [x^L, x^U]\}$, then $\mathrm{conv}(W)$ is given by:

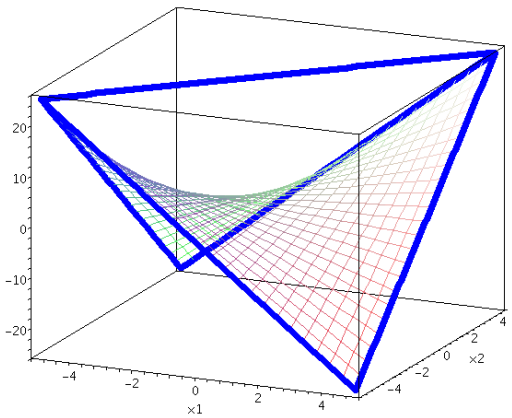
$$\begin{aligned} w &\geq x_1^L x_2 + x_2^L x_1 - x_1^L x_2^L \\ w &\geq x_1^U x_2 + x_2^U x_1 - x_1^U x_2^U \\ w &\leq x_1^L x_2 + x_2^U x_1 - x_1^L x_2^U \\ w &\leq x_1^U x_2 + x_2^L x_1 - x_1^U x_2^L \end{aligned}$$

- Stated [McCormick; MP, 1976], proved [Al-Khayyal, Falk; MOR, 1983]

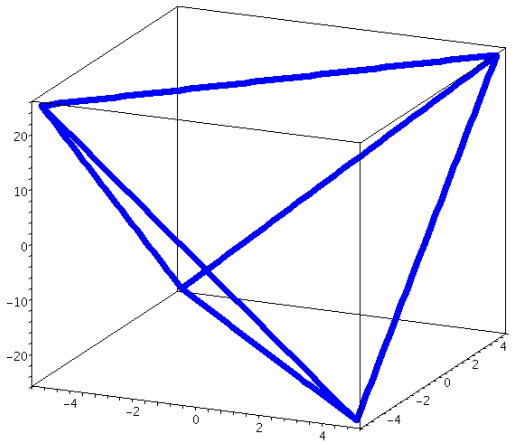
McCormick's envelopes



Lower envelopes



Upper envelopes



Both

Special case: Fortet's linearization

If x_1 and x_2 are binary variables, the McCormick's inequalities lead to the Fortet's inequalities [Fortet; RFRO, 1960]:

$$\begin{aligned} w &\geq 0 \\ w &\geq x_2 + x_1 - 1 \\ w &\leq x_1 \\ w &\leq x_2 \end{aligned}$$

The resulting reformulation is an exact linearization as shown in [Liberti; RAIRO-RO, 2009]

Trilinear case

It is not as easy as bilinear convex relaxation:

Trilinear case

It is not as easy as bilinear convex relaxation:

- the number of constraints is greater than 4

Trilinear case

It is not as easy as bilinear convex relaxation:

- the number of constraints is greater than 4
- there are several cases, depending on sign of bounds of the variables: $x_i^L x_i^U \geq 0$ [Meyer, Floudas; 2003]; mixed case [Meyer, Floudas; JOGO, 2004]

Trilinear case

It is not as easy as bilinear convex relaxation:

- the number of constraints is greater than 4
- there are several cases, depending on sign of bounds of the variables: $x_i^L x_i^U \geq 0$ [Meyer, Floudas; 2003]; mixed case [Meyer, Floudas; JOGO, 2004]
- there are further conditions to check

Example (1): $x_1^U,\ x_2^U,\ x_3^U \leq 0$

Permute variables x_1, x_2 and x_3 such that:

$$\begin{aligned} x_1^U x_2^L x_3^L + x_1^L x_2^U x_3^U &\leq x_1^L x_2^U x_3^L + x_1^U x_2^L x_3^U \\ x_1^U x_2^L x_3^L + x_1^L x_2^U x_3^U &\leq x_1^U x_2^U x_3^L + x_1^L x_2^L x_3^U \end{aligned}$$

Example (1): $x_1^U, x_2^U, x_3^U \leq 0$

Permute variables x_1, x_2 and x_3 such that:

$$\begin{aligned} x_1^U x_2^L x_3^L + x_1^L x_2^U x_3^U &\leq x_1^L x_2^U x_3^L + x_1^U x_2^L x_3^U \\ x_1^U x_2^L x_3^L + x_1^L x_2^U x_3^U &\leq x_1^U x_2^U x_3^L + x_1^L x_2^L x_3^U \end{aligned}$$

Lower envelope:

$$\begin{aligned} w &\geq x_2^L x_3^L x_1 + x_1^L x_3^L x_2 + x_1^L x_2^L x_3 - 2x_1^L x_2^L x_3^L \\ w &\geq x_2^U x_3^U x_1 + x_1^U x_3^U x_2 + x_1^U x_2^U x_3 - 2x_1^U x_2^U x_3^U \\ w &\geq x_2^L x_3^U x_1 + x_1^L x_3^U x_2 + x_1^U x_2^L x_3 - x_1^L x_2^L x_3^U - x_1^U x_2^L x_3^U \\ w &\geq x_2^U x_3^L x_1 + x_1^U x_3^L x_2 + x_1^L x_2^U x_3 - x_1^U x_2^U x_3^L - x_1^L x_2^U x_3^L \\ w &\geq c_1 x_1 + x_1^U x_3^L x_2 + x_1^U x_2^L x_3 + x_1^L x_2^U x_3^U - c_1 x_1^L - x_1^U x_2^U x_3^L - x_1^U x_2^L x_3^U \\ w &\geq c_2 x_1 + x_1^L x_3^U x_2 + x_1^L x_2^U x_3 + x_1^U x_2^L x_3^L - c_2 x_1^U - x_1^L x_2^L x_3^U - x_1^L x_2^U x_3^L, \end{aligned}$$

where $c_1 = \frac{x_1^U x_2^U x_3^L - x_1^L x_2^U x_3^U - x_1^U x_2^L x_3^L + x_1^U x_2^L x_3^U}{x_1^U - x_1^L}$ and

$c_2 = \frac{x_1^L x_2^L x_3^U - x_1^U x_2^L x_3^L - x_1^L x_2^U x_3^U + x_1^L x_2^U x_3^L}{x_1^L - x_1^U}$

Example (2): $x_1^U,\ x_2^U,\ x_3^U \leq 0$

Upper envelope:

$$
\begin{aligned}
w &\leq x_2^L x_3^L x_1 + x_1^U x_3^L x_2 + x_1^U x_2^U x_3 - x_1^U x_2^U x_3^L - x_1^U x_2^L x_3^L \\
w &\leq x_2^U x_3^L x_1 + x_1^L x_3^L x_2 + x_1^U x_2^U x_3 - x_1^U x_2^U x_3^L - x_1^L x_2^U x_3^L \\
w &\leq x_2^L x_3^L x_1 + x_1^U x_3^U x_2 + x_1^U x_2^L x_3 - x_1^U x_2^L x_3^U - x_1^U x_2^L x_3^L \\
w &\leq x_2^U x_3^U x_1 + x_1^L x_3^L x_2 + x_1^L x_2^U x_3 - x_1^L x_2^U x_3^U - x_1^L x_2^U x_3^L \\
w &\leq x_2^L x_3^U x_1 + x_1^U x_3^U x_2 + x_1^L x_2^L x_3 - x_1^U x_2^L x_3^U - x_1^L x_2^L x_3^U \\
w &\leq x_2^U x_3^U x_1 + x_1^L x_3^U x_2 + x_1^L x_2^L x_3 - x_1^L x_2^U x_3^U - x_1^L x_2^L x_3^U .
\end{aligned}
$$

Quadrilinear terms

The convex envelope is not known explicitly for quadrilinear terms

- Combine bilinear and trilinear envelope [Cafieri, Lee, Liberti; JOGO, 2011]
- Convex envelope for some cases presented in [Balram; M.Sc. Thesis, 2019] (e.g., when $x_1^L, x_2^L, x_3^L, x_4^L \geq 0$, then 44 constraints are generated)

Beyond quadrilinear terms

- envelopes for multilinear terms larger than quadrilinear: not known explicitly

Beyond quadrilinear terms

- envelopes for multilinear terms larger than quadrilinear: not known explicitly
- software as PORTA can compute the convex hull of a given set of points in $\mathbb{R}^n$

Beyond quadrilinear terms

- envelopes for multilinear terms larger than quadrilinear: not known explicitly
- software as PORTA can compute the convex hull of a given set of points in $\mathbb{R}^n$
- Balram's thesis reports a similar procedure to compute the convex hull (but less refined)

Dual relaxation: preliminaries

- Consider the 2^k point set P:

$$\{ \begin{array}{l} (x_1^L, \ldots, x_{k-1}^L, x_k^L), \\ (x_1^L, \ldots, x_{k-1}^L, x_k^U), \\ (x_1^L, \ldots, x_{k-1}^U, x_k^L), \\ (x_1^L, \ldots, x_{k-1}^U, x_k^U), \\ \ldots, \\ (x_1^U, \ldots, x_{k-1}^U, x_k^L), \\ (x_1^U, \ldots, x_{k-1}^U, x_k^U) \quad \} \end{array}$$

(i.e., all combinations of lower/upper bounds)

- Let $w(x) = \prod_{i \leq k} x_i$: lift P to (x, w) space, get $P_W \subseteq \mathbb{R}^{k+1}$

$$\forall \bar{x} \in P \quad (\bar{x}, w(\bar{x})) \in P_W$$

Dual representation of a point set

- Convex hull of $P = \{p_1, \ldots, p_m\} \subseteq \mathbb{R}^n$ is given by $x \in \mathbb{R}^n \mid$:

$$\exists \lambda \in \mathbb{R}^m \left(x = \sum_{i \leq m} \lambda_i p_i \ \wedge \ \sum_{i \leq m} \lambda_i = 1 \ \wedge \ \forall i \leq m \ (\lambda_i \geq 0) \right)$$

Dual representation of a point set

- Convex hull of $P = \{p_1, \ldots, p_m\} \subseteq \mathbb{R}^n$ is given by $x \in \mathbb{R}^n \mid$:

$$\exists \lambda \in \mathbb{R}^m \left(x = \sum_{i \leq m} \lambda_i p_i \ \wedge \ \sum_{i \leq m} \lambda_i = 1 \ \wedge \ \forall i \leq m \ (\lambda_i \geq 0) \right)$$

- $\Leftrightarrow x$ is a convex combination of points in P

Dual representation of a point set

- Convex hull of $P = \{p_1, \ldots, p_m\} \subseteq \mathbb{R}^n$ is given by $x \in \mathbb{R}^n \mid$:

$$\exists \lambda \in \mathbb{R}^m \left(x = \sum_{i \leq m} \lambda_i p_i \ \wedge \ \sum_{i \leq m} \lambda_i = 1 \ \wedge \ \forall i \leq m \ (\lambda_i \geq 0) \right)$$

- $\Leftrightarrow x$ is a convex combination of points in P
- Can express points in P_W in function of x, w, x^L, x^U and of added (dual) variables λ for any k

Dual representation of a point set

- Convex hull of $P = \{p_1, \ldots, p_m\} \subseteq \mathbb{R}^n$ is given by $x \in \mathbb{R}^n \mid$:

$$\exists \lambda \in \mathbb{R}^m \left(x = \sum_{i \leq m} \lambda_i p_i \ \wedge \ \sum_{i \leq m} \lambda_i = 1 \ \wedge \ \forall i \leq m \ (\lambda_i \geq 0) \right)$$

- $\Leftrightarrow x$ is a convex combination of points in P
- Can express points in P_W in function of x, w, x^L, x^U and of added (dual) variables λ for any k
- Automatically get explicit convex envelopes for multilinear terms

Example: bilinear term

Using a matrix representation, we have:

$$\begin{bmatrix}\lambda_1 & \lambda_2 & \lambda_3 & \lambda_4\end{bmatrix} \cdot \begin{bmatrix} x_1^L & x_2^L \\ x_1^L & x_2^U \\ x_1^U & x_2^L \\ x_1^U & x_2^U \end{bmatrix} = \begin{bmatrix} x_1 & x_2 \end{bmatrix}$$

Example: bilinear term

Using a matrix representation, we have:

$$\begin{bmatrix}\lambda_1 & \lambda_2 & \lambda_3 & \lambda_4\end{bmatrix} \cdot \begin{bmatrix} x_1^L & x_2^L \\ x_1^L & x_2^U \\ x_1^U & x_2^L \\ x_1^U & x_2^U \end{bmatrix} = \begin{bmatrix} x_1 & x_2 \end{bmatrix}$$

$$\begin{bmatrix}\lambda_1 & \lambda_2 & \lambda_3 & \lambda_4\end{bmatrix} \cdot \begin{bmatrix} x_1^L x_2^L \\ x_1^L x_2^U \\ x_1^U x_2^L \\ x_1^U x_2^U \end{bmatrix} = w$$

Example: bilinear term

Using a matrix representation, we have:

$$\begin{bmatrix}\lambda_1 & \lambda_2 & \lambda_3 & \lambda_4\end{bmatrix} \cdot \begin{bmatrix} x_1^L & x_2^L \\ x_1^L & x_2^U \\ x_1^U & x_2^L \\ x_1^U & x_2^U \end{bmatrix} = \begin{bmatrix} x_1 & x_2 \end{bmatrix}$$

$$\begin{bmatrix}\lambda_1 & \lambda_2 & \lambda_3 & \lambda_4\end{bmatrix} \cdot \begin{bmatrix} x_1^L x_2^L \\ x_1^L x_2^U \\ x_1^U x_2^L \\ x_1^U x_2^U \end{bmatrix} = w$$

$$\begin{aligned} x_1 &= \lambda_1 x_1^L + \lambda_2 x_1^L + \lambda_3 x_1^U + \lambda_4 x_1^U \\ x_2 &= \lambda_1 x_2^L + \lambda_2 x_2^U + \lambda_3 x_2^L + \lambda_4 x_2^U \\ w &= \lambda_1 x_1^L x_2^L + \lambda_2 x_1^L x_2^U + \lambda_3 x_1^U x_2^L + \lambda_4 x_1^U x_2^U \\ \sum_{i \leq 4} \lambda_i &= 1 \end{aligned}$$

Experimental set-up

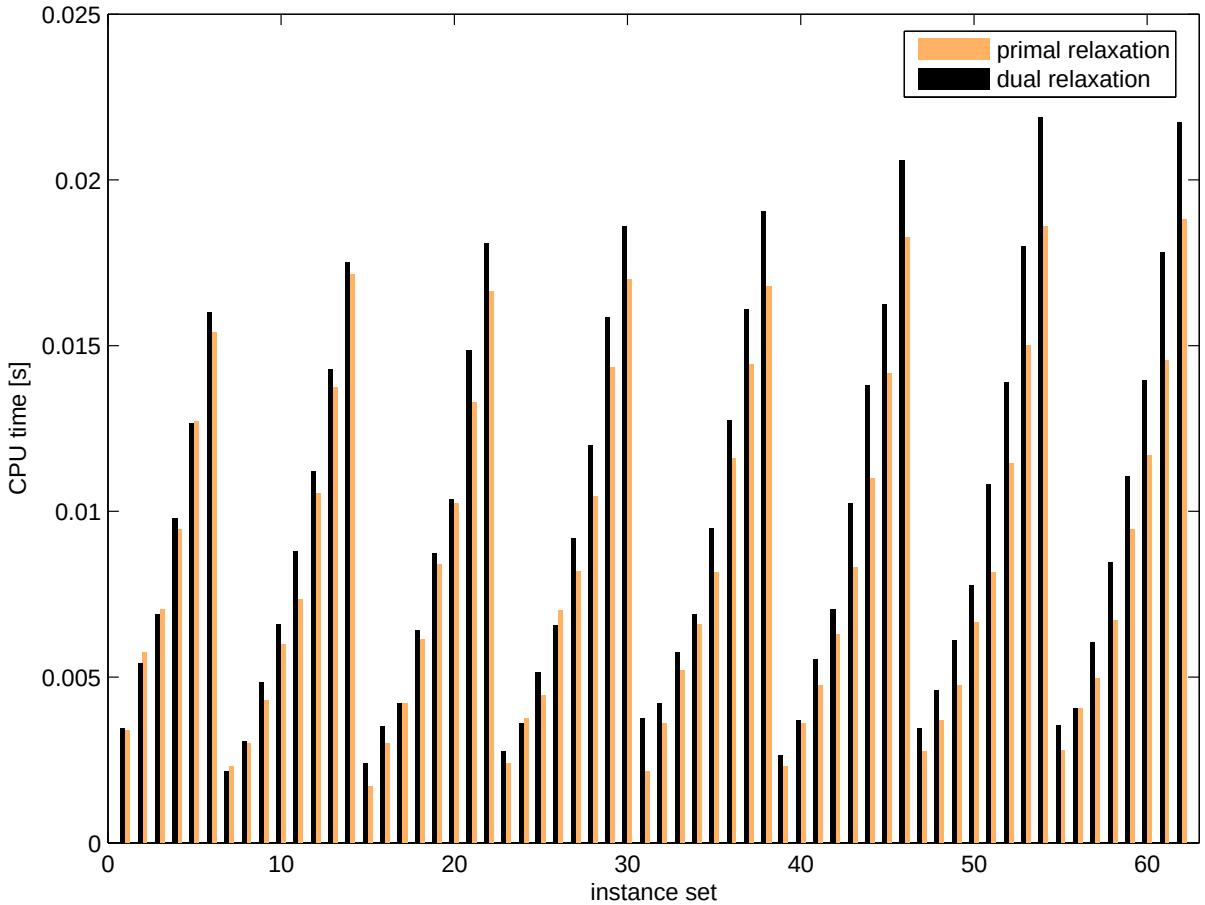
- Generate random multilinear NLPs P
  - linear, bilinear, trilinear terms
- Generate primal convex LP relaxation R_P
- Generate dual convex LP relaxation Λ_P
- Solve R_P, Λ_P using CPLEX, compare CPU times
- *To "get a feel" about how R_P, Λ_P might perform in BB, add integrality constraints on primal variables, get MILP relaxations R'_P, Λ'_P*
- Solve R'_P, Λ'_P using CPLEX, compare CPU times

Instance set

- 2520 random instances
- # variables $n \in \{10, 20\}$
- $n = 10$:
  - # bilinear terms $\beta \in \{0, 10, 13, 17, 21, 25, 29, 33\}$
  - # trilinear terms $\tau \in \{0, 10, 22, 34, 46, 58, 71, 83\}$
- $n = 20$:
  - $\beta \in \{0, 20, 38, 57, 76, 95, 114, 133\}$
  - $\tau \in \{0, 20, 144, 268, 393, 517, 642, 766\}$
- 20 instances for each parameter combination yielding multilinear NLPs (and then MINLPs after imposing integrality on some variables)
- Variable bounds chosen at random, magnitude $\pm 2.0 \times 10^1$

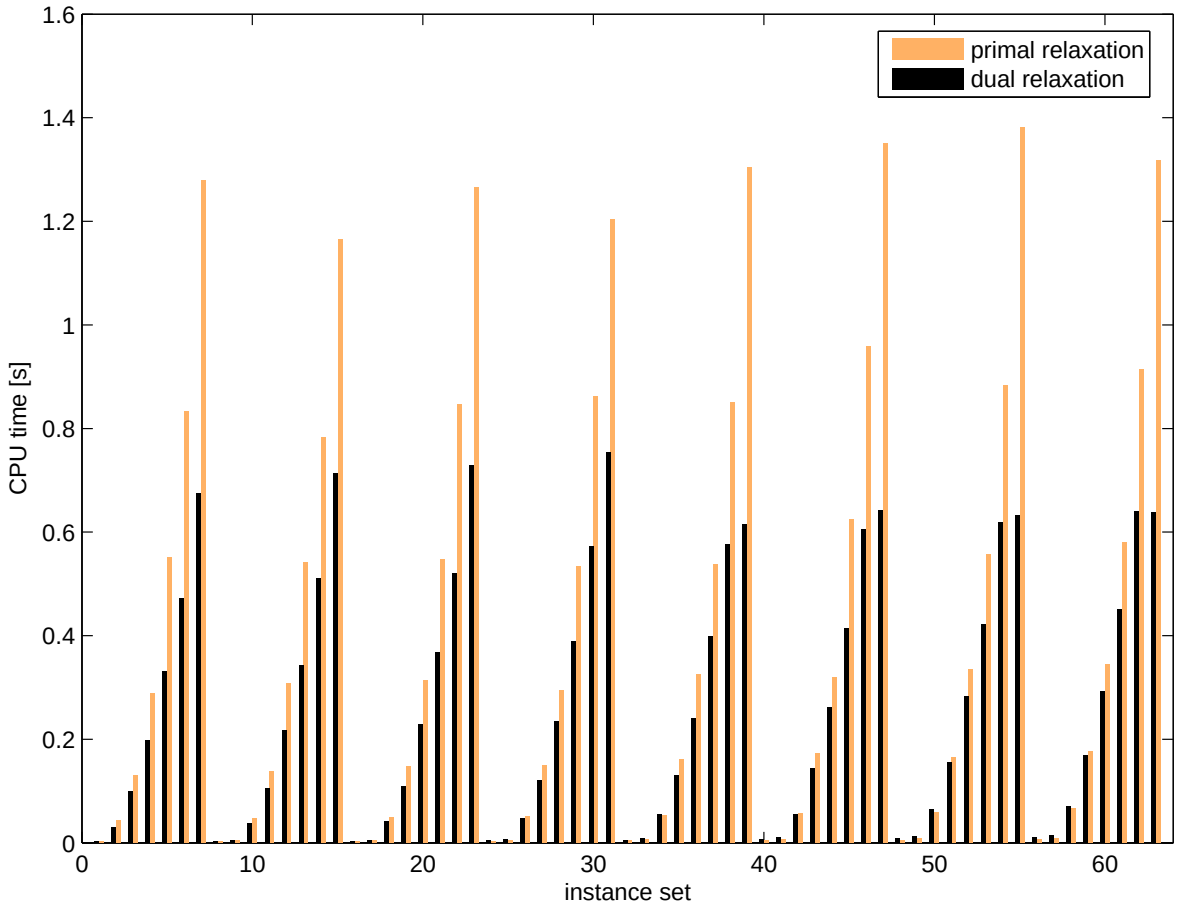
LP relaxation test, $n = 10$

CPU time averages over each 20-instance block with given (n, β, τ)



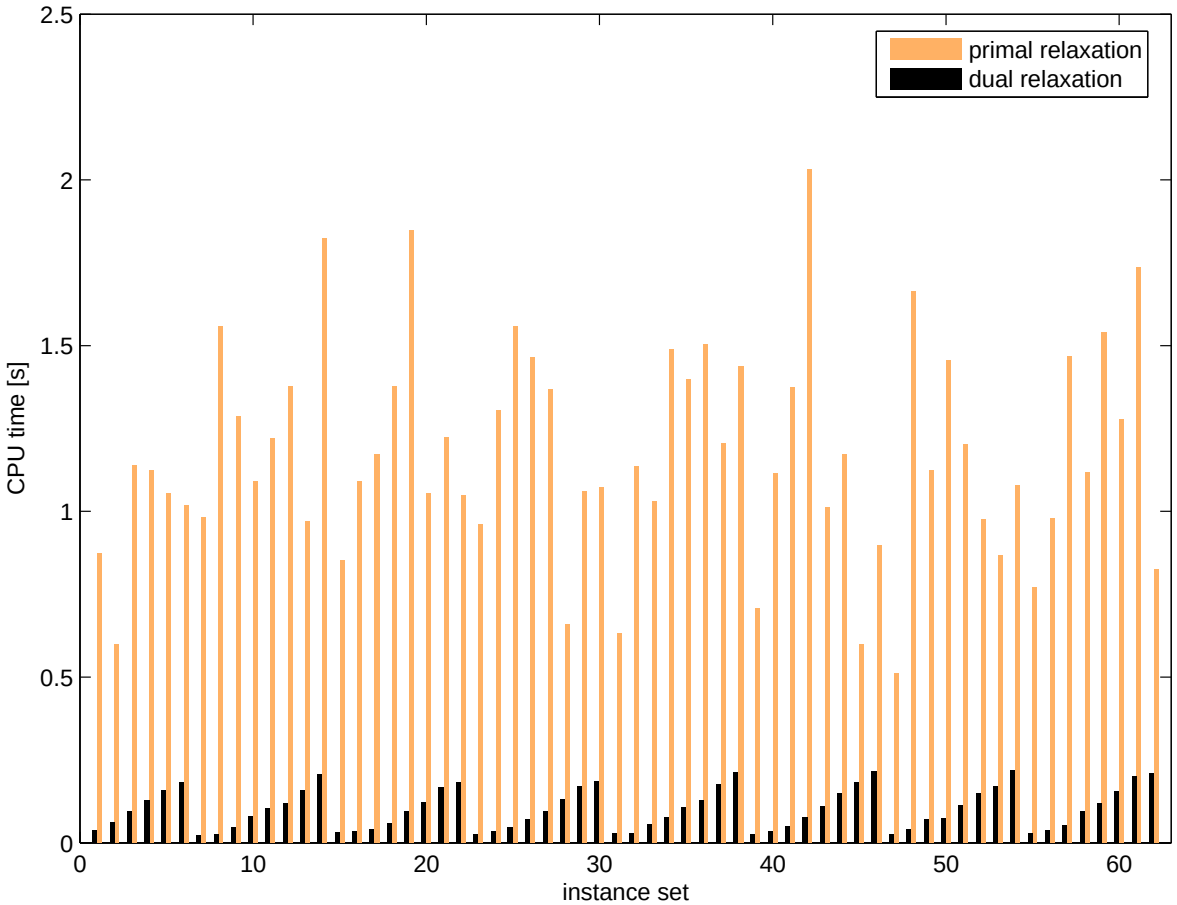
LP relaxation test, $n = 20$

CPU time averages over each 20-instance block with given (n, β, τ)



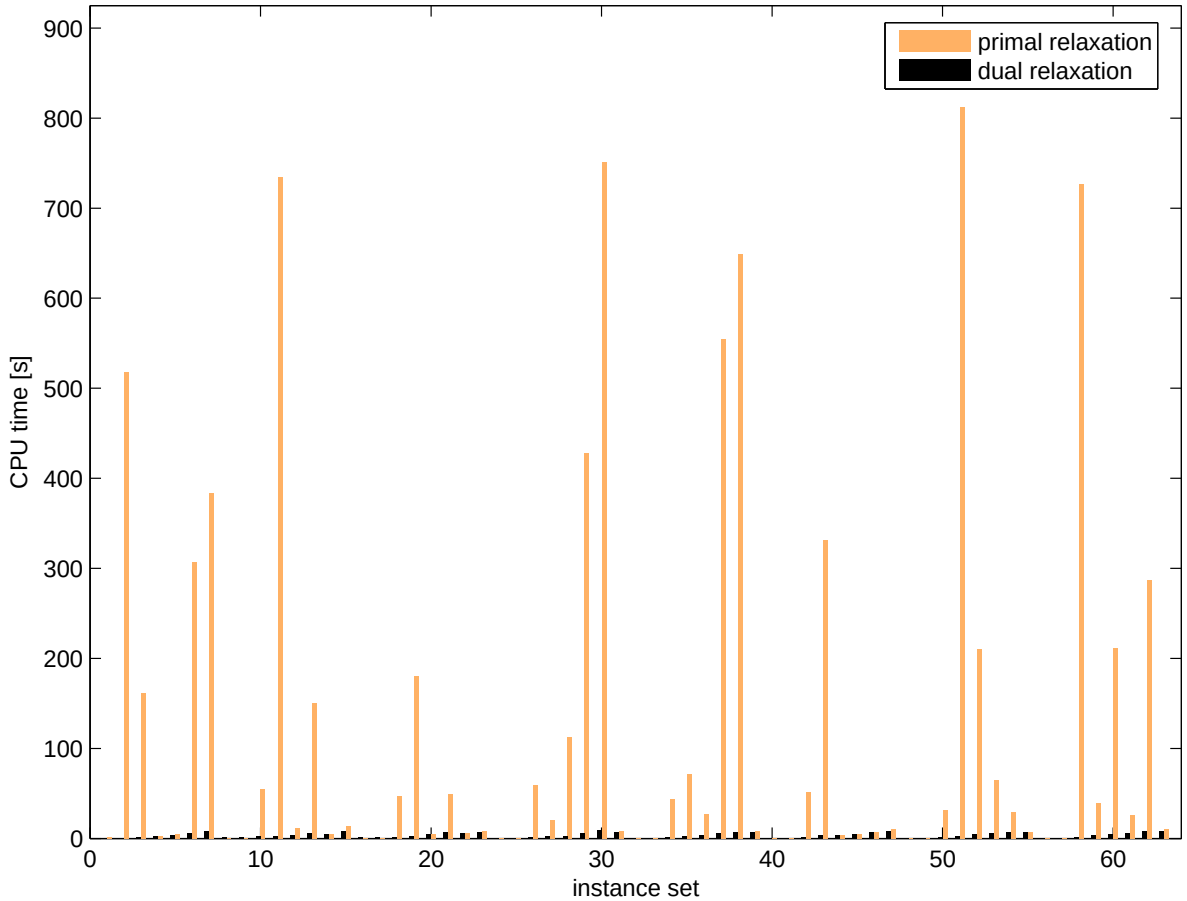
MILP relaxation test, $n = 10$

CPU time averages over each 20-instance block with given (n, β, τ)



MILP relaxation test, $n = 20$

CPU time averages over each 20-instance block with given (n, β, τ)



Where are we?

Conclusions

Final considerations

- Reformulations can have a high impact in terms of computational times

Conclusions

Final considerations

- Reformulations can have a high impact in terms of computational times
- Reformulations can allow to employ different solvers

Conclusions

Final considerations

- Reformulations can have a high impact in terms of computational times
- Reformulations can allow to employ different solvers
- Human contribution is important: automatic reformulations are not easy to derive due to the specific features a problem can present.

Conclusions

Final considerations

- Reformulations can have a high impact in terms of computational times
- Reformulations can allow to employ different solvers
- Human contribution is important: automatic reformulations are not easy to derive due to the specific features a problem can present.

Future work

- Clustering: implement an exact method for bipartite modularity maximization

Conclusions

Final considerations

- Reformulations can have a high impact in terms of computational times
- Reformulations can allow to employ different solvers
- Human contribution is important: automatic reformulations are not easy to derive due to the specific features a problem can present.

Future work

- Clustering: implement an exact method for bipartite modularity maximization
- Circle packing: prove the conjecture about bound constraints

Conclusions

Final considerations

- Reformulations can have a high impact in terms of computational times
- Reformulations can allow to employ different solvers
- Human contribution is important: automatic reformulations are not easy to derive due to the specific features a problem can present.

Future work

- Clustering: implement an exact method for bipartite modularity maximization
- Circle packing: prove the conjecture about bound constraints
- Relaxations for multilinear terms: try to implement the dual approach for some sBB solver.