An improved algorithm for finding minimum cycle bases in undirected graphs

Edoardo Amaldi

Dipartimento di Elettronica e Informazione (DEI), Politecnico di Milano, Italy

Joint work with C. Iuliano (DEI), R. Rizzi (Univ. Udine)
K. Mehlhorn and T. Jurkiewicz (MPI, Saarbrücken)

Outline

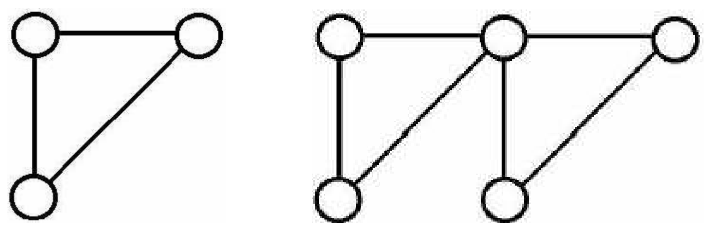
- Cycle bases in undirected graphs
- The minimum cycle basis problem
- Previous and related work
- New hybrid algorithm
- Some computational results
- Concluding remarks

Preliminaries

Simple connected undirected **graph** $G = (V, E)$, $n = |V|$, $m = |E|$, with a **weight** $w_e \geq 0$ for each edge $e \in E$

Elementary cycle = connected subset of edges whose nodes have degree 2

Cycle = subset of edges $C \subseteq E$ such that every node of V is incident with an even number of edges in C



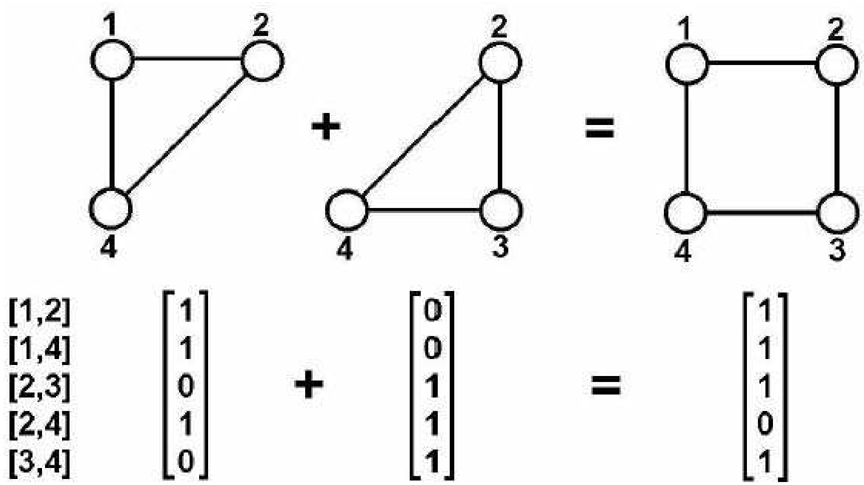
Cycles can be viewed as the (possibly empty) union of edge-disjoint elementary cycles

Cycle composition

Cycles can be represented by edge-incidence vectors in $\{0,1\}^{|E|}$

Composition of two cycles:

- symmetric difference of the edge-sets $(C_1 \cup C_2) \setminus (C_1 \cap C_2)$
- *modulo 2 addition* of the incidence vectors

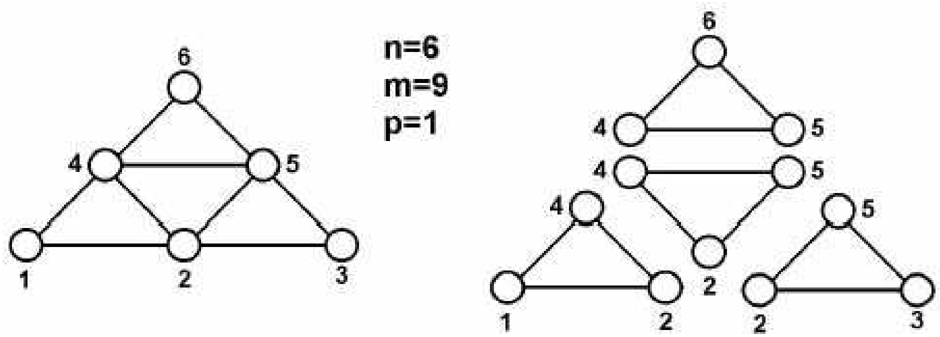


$$\begin{bmatrix}1\\1\\0\\1\\0\end{bmatrix} + \begin{bmatrix}0\\0\\1\\1\\1\end{bmatrix} = \begin{bmatrix}1\\1\\1\\0\\1\end{bmatrix}$$

Cycle bases

The collection of all cycles forms a vector space over $GF(2)$, called the **cycle space** $\mathcal{C}$

A cycle basis $B = \{b_1, \ldots, b_\nu\}$ of $\mathcal{C}$ is of dimension $\nu = m - n + 1$



The problem

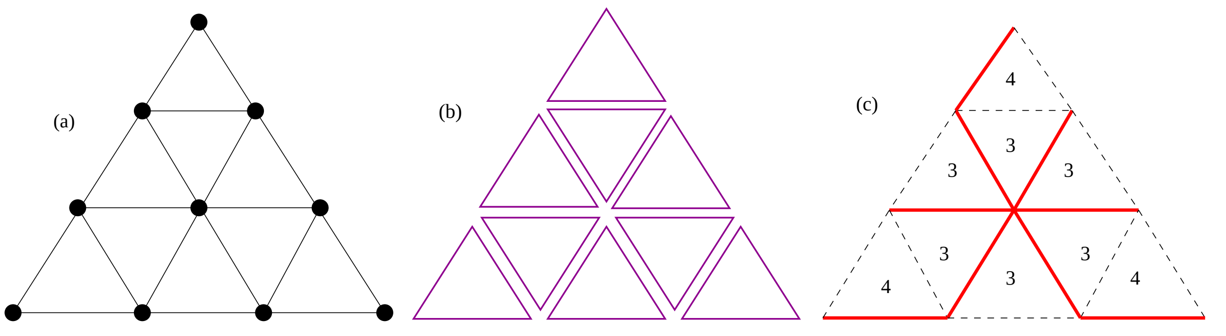
MIN CB:

Given a connected graph $G = (V, E)$ with a **weight** $w_e \geq 0$ for each $e \in E$, find a **Minimum Cycle Basis** $B = \{b_1, \ldots, b_\nu\}$, i.e., B with minimum $w(B) = \sum_{i=1}^{\nu} \sum_{e \in b_i} w_e$.

The problem

MIN CB:

Given a connected graph $G = (V, E)$ with a **weight** $w_e \geq 0$ for each $e \in E$, find a **Minimum Cycle Basis** $B = \{b_1, \ldots, b_\nu\}$, i.e., B with minimum $w(B) = \sum_{i=1}^{\nu} \sum_{e \in b_i} w_e$.



edge weights =1 CB weight = 27 CB weight = 30

The problem

MIN CB:

Given a connected graph $G = (V, E)$ with a **weight** $w_e \geq 0$ for each $e \in E$, find a **Minimum Cycle Basis** $B = \{b_1, \ldots, b_\nu\}$, i.e., B with minimum $w(B) = \sum_{i=1}^{\nu} \sum_{e \in b_i} w_e$.

Applications:

- test of electrical circuits
- structural engineering
- frequency analysis of computer programs
- planning complex syntheses in organic chemistry
- periodic event scheduling,...

Previous work

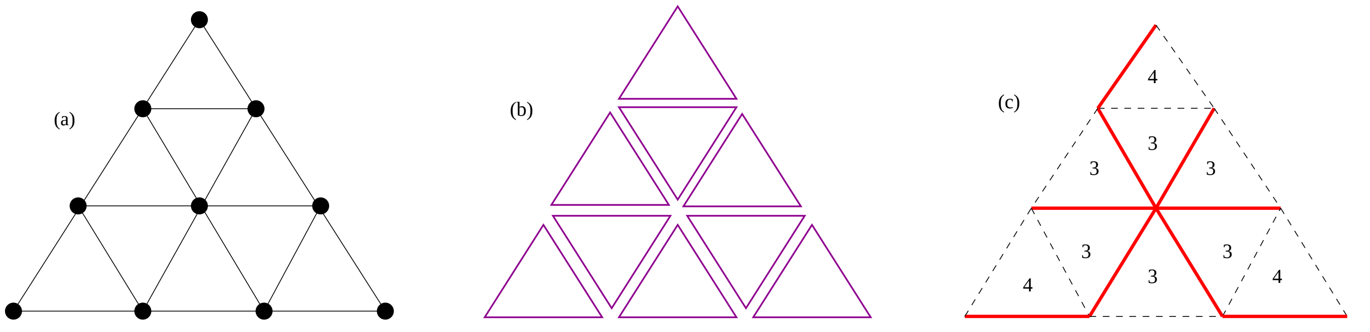
- Early methods (Stepanec 64, Zykov 69, Hubicka and Syslo 76) are not polynomial
- First polynomial algorithm by Horton (87) is $O(m^3 n)$
- Improved $O(m^\omega n)$ version, where $\omega < 2.376$ is the exponent of fast matrix multiplication (Golynski and Horton 02)
- Different $O(m^3 + mn^2 \log n)$ algorithm (de Pina 95)
- Improved $O(m^2 n + mn^2 \log n)$ variant of de Pina's algorithm using fast matrix multiplication (Kavitha, Mehlhorn et al. 04)

Previous work

- $O(m^2n^2)$ hybrid algorithm (Mehlhorn and Michail 06)
- $O(m^2n)$ algorithm based on minimum feedback vertex set, can be improved to $O(m^2n/\log n + mn^2)$ using a bit packing trick (Mehlhorn and Michail 07)

Related problem

Let T be an arbitrary spanning tree of G, the ν cycles obtained by adding $e \in E \setminus T$ form a fundamental cycle basis (FCB) of G



Not all cycle bases are fundamental

MIN FCB is NP-hard (Deo et al. 82), in fact APX-hard but approximable within $O(\log^2 n \log\log n)$ (Galbiati, A. and Rizzi 07)

Edge-swapping algorithm (A., Liberti et al. 04/09)

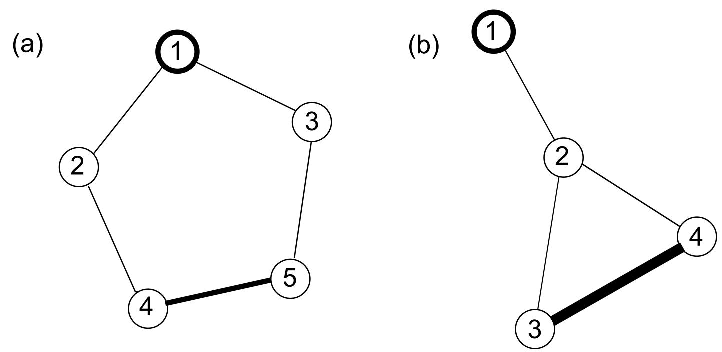
Horton algorithm

Assumption: shortest paths are unique (lexicographic order)

Proposition: the collection of cycles

$$\mathcal{H} = \{P_{u,v_1} \cdot e \cdot P_{v_2,u} \mid u \in V, e = [v_1, v_2] \in E\}$$

contains a minimum cycle basis.

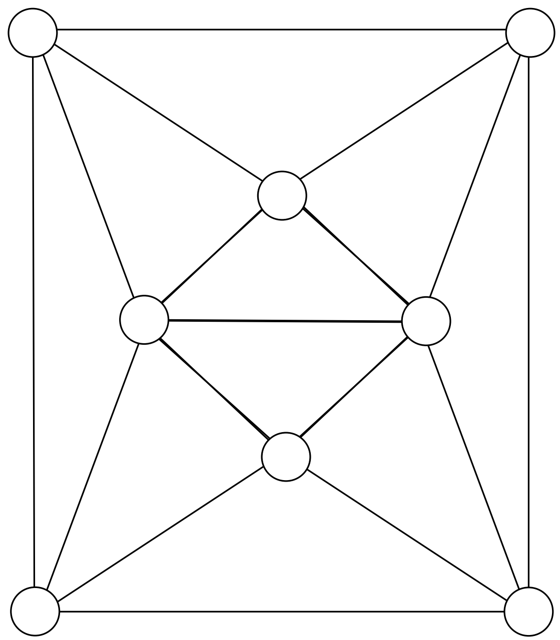


$$|\mathcal{H}| \leq mn$$

Horton algorithm

Since the set of all cycles forms a **matroid**, a greedy procedure yields a minimum CB

Need to test linear independence because not all cycles in $\mathcal{H}$ are in a minimum CB



Horton algorithm

1) For each node u, determine **shortest path tree**

$$O(nm \log n) \text{ Dijkstra with heap}$$

2) Construct the **candidate cycles** in $\mathcal{H}$ and order them by non-decreasing weight ($|\mathcal{H}| \leq \nu n \leq mn$)

$$O(mn^2) \text{ construction and } O(mn \log n) \text{ ordering}$$

3) Find a minimum cycle basis by selecting the ν lightest **linearly independent** candidate cycles

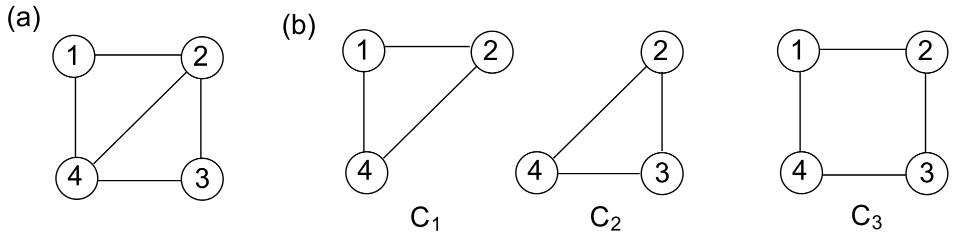
$$O(m^3 n) \text{ see below}$$

Overall complexity: $O(m^3 n)$

Horton algorithm

Binary Gaussian elimination:

(a) (b)



(c)

$$
\begin{array}{c|ccccc}
 & [1,2] & [1,4] & [2,3] & [2,4] & [3,4] \\
\hline
C_1 & 1 & 1 & 0 & 1 & 0 \\
C_2 & 0 & 0 & 1 & 1 & 1 \\
C_3 & 1 & 1 & 1 & 0 & 1
\end{array}
\longrightarrow
\begin{array}{c|ccccc}
 & [1,2] & [1,4] & [2,3] & [2,4] & [3,4] \\
\hline
C_1 & \boxed{1} & 1 & 0 & 1 & 0 \\
C_2 & 0 & 0 & 1 & 1 & 1 \\
C_3 & 0 & 0 & 1 & 1 & 1
\end{array}
$$

(d)

$$
\begin{array}{c|ccccc}
 & [1,2] & [1,4] & [2,3] & [2,4] & [3,4] \\
\hline
C_1 & 1 & 1 & 0 & 1 & 0 \\
C_2 & 0 & 0 & 1 & 1 & 1 \\
C_3 & 0 & 0 & 1 & 1 & 1
\end{array}
\longrightarrow
\begin{array}{c|ccccc}
 & [1,2] & [1,4] & [2,3] & [2,4] & [3,4] \\
\hline
C_1 & \boxed{1} & 1 & 0 & 1 & 0 \\
C_2 & 0 & 0 & \boxed{1} & 1 & 1 \\
C_3 & 0 & 0 & 0 & 0 & 0
\end{array}
$$

(e)

$$
\begin{array}{c|ccccc}
 & [1,2] & [1,4] & [2,3] & [2,4] & [3,4] \\
\hline
C_1 & 1 & 1 & 0 & 1 & 0 \\
C_2 & 0 & 0 & 1 & 1 & 1 \\
C_3 & \cancel{0} & \cancel{0} & \cancel{0} & \cancel{0} & \cancel{0}
\end{array}
\longrightarrow
\begin{array}{c|ccccc}
 & [1,2] & [1,4] & [2,3] & [2,4] & [3,4] \\
\hline
C_1 & 1 & 1 & 0 & 1 & 0 \\
C_2 & 0 & 0 & 1 & 1 & 1
\end{array}
$$

each row can be processed in $O(mr)$, where r is the number of rows above

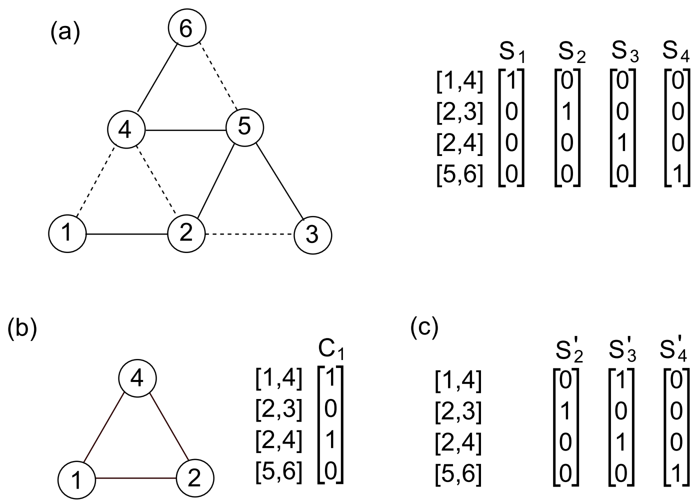
since $r \leq \nu$ and $|\mathcal{H}| \leq n\nu$, we have $O(m\nu^2 n) = O(m^3 n)$

Improved de Pina algorithm

Idea: determine cycles of min CB **sequentially**, considering at each step a basis orthogonal to the lin. subspace generated by cycles computed so far.

Let T be any spanning tree of G, and $e_1, \ldots, e_\nu$ the edges in $E \setminus T$ in some arbitrary order.

Any cycle of G can be viewed as a restricted incidence vector in $\{0,1\}^\nu$ (lin. indep. of the restricted and full vectors is equivalent).



Improved de Pina algorithm

Let $\{S_1, \ldots, S_\nu\}$ be the canonical basis
for i=1 to ν **do**
  Find C_i as the shortest cycle in G s.t. $< C_i, S_i >= 1$
  for j = i+1 to ν **do**
    if $< S_j, C_i >= 1$ **then** $S_j :=< S_j, S_i >$

Since S_i is orthogonal to $C_1, \ldots, C_{i-1}$ and $< C_i, S_i >= 1$, C_i is lin. indep.

A shortest C_i with $< C_i, S_i >= 1$ can be found by shortest path computations in a two level graph

Update S_j's so that $\{S_{i+1}, \ldots, S_\nu\}$ is still a basis of the subspace orthogonal to $\{C_1, \ldots, C_i\}$.

$O(m^3 + mn^2 \log n)$ can be reduced to $O(m^2 n + mn^2 \log n)$ with fast matrix multiplication (Kavitha, Mehlhorn et al. 04)

FVS-based algorithm

Mehlhorn and Michail 07

Consider only Horton candidate cycles whose node u belongs to a close-to-minimum feedback vertex set (FVS) – NP-hard but 2-approximable

$O(m^2n + mn^2)$ algorithm with a "simple" way to extract a minimum CB from the above set of candidate cycles

$O(m^2n/\log(n) + mn^2)$ variant by using a bit-packing trick

New hybrid algorithm

Main ideas:

1) Substantially reduce the number of candidate cycles (trim $\mathcal{H}$)

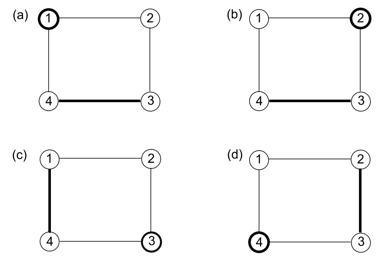
the candidate cycles in $\mathcal{H}' \subseteq H$ are "sparse"

2) Devise an adaptive variant of the linear independence test à la de Pina that iteratively builds the spanning tree T.

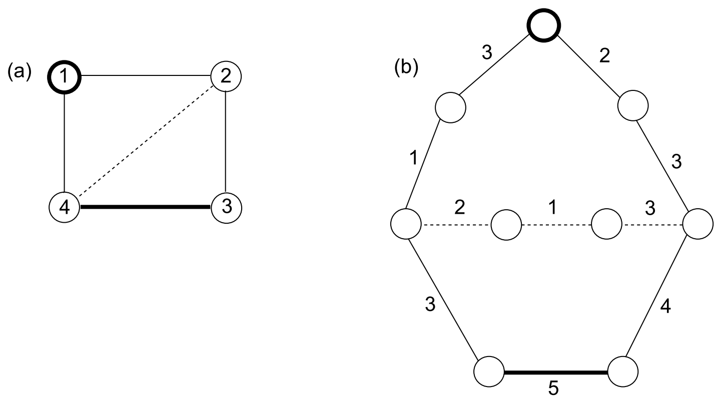
Algorithm: order the candidate cycles in $\mathcal{H}'$ by non-decreasing weight, and select the lightest ν linear independent ones

Reduced candidate cycle set

Besides discarding duplicates



Only keep in $\mathcal{H}'$ the **isometric cycles** $C \in \mathcal{H}$, i.e., which have for **each node** u an **edge** $e = [v_1, v_2]$ in C s.t. $C = P_{u,v_1} \cdot e \cdot P_{v_2,u}$



Reduced candidate cycle set

Isometric cycles can be found in $O(mn \log n)$ by using binary search

Although we still have $|\mathcal{H}'| = O(mn)$, the incidence vectors of these candidate cycles are sparse!

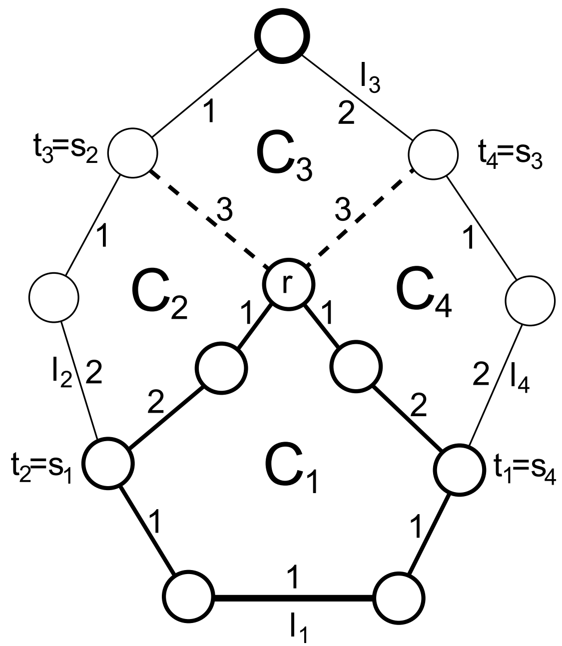
Property (sparsity): $\sum_{C_i \in \mathcal{H}'} |C_i| \leq mn$, where $|C_i|$ denotes the number of edges in C_i.

Obvious because each $C_i \in \mathcal{H}'$ represents $|C_i|$ cycles in $\mathcal{H}$ and $|\mathcal{H}| \leq mn$.

Example: K_n

Reduced candidate cycle set

We can also discard any C that admits a **wheel decomposition**, that is s.t. $C = C_1 + \ldots + C_k$ w.r.t. some root r and with $|C_j| < |C|$ for all $j = 1, \ldots, k$



NB: non-isometric is special case with $k = 2$ and $r \in C$

Complexity: $O(mn^2)$

New independence test à la de Pina

Idea: Build the spanning tree T and order the co-tree edges $e_1, \ldots, e_\nu$ (and hence the witnesses S_i) adaptively so as to reduce the computational load.

We try to avoid updating the other witnesses...

Complexity: $O(m^2 n)$ – the bottleneck

Some computational results

Instances:

- Hypercubes with $n = 2^d$; random graphs with densities $0.3, 0.5, 0.9$ or sparse ($m = 2n$) and random weights (Mehlhorn and Michail 06)
- Euclidean graphs with density $0.1 - 0.9$, weighted hypercubes, toroidal graphs

Intel Xeon(TM) with 2.80 GHz and 2GB RAM

Some computational results

Cpu time for random graphs with density=0.5

			Horton	Hybrid Mehlhorn et al.	New-isometric
n	m	ν	avg - stddev	avg - stddev	avg - stddev
50	612	563	0.01 - 0.00	0.04 - 0.01	0.00 - 0.00
60	885	826	0.02 - 0.01	0.08 - 0.01	0.01 - 0.01
70	1207	1138	0.03 - 0.01	0.19 - 0.03	0.01 - 0.01
80	1580	1501	0.07 - 0.01	0.34 - 0.03	0.02 - 0.01
90	2002	1913	0.10 - 0.01	0.51 - 0.02	0.02 - 0.01
100	2475	2376	0.11 - 0.01	0.72 - 0.03	0.03 - 0.01
125	3875	3751	0.33 - 0.01	5.87 - 0.24	0.05 - 0.01

Efficient implementation of Horton algorithm performs better than the other algorithms in the literature with better worst-case complexity

Some computational results

Number of candidate cycles and cpu time for Euclidean graphs with n=150

density	m	ν	Horton	New-isometric	New-no-wheels
0.1	1228	1079	21311 - 0.03	3289 - 0.02	1163 - 0.09
0.2	2388	2239	54626 - 0.07	9963 - 0.03	2342 - 0.15
0.3	3452	3303	106971 - 0.11	21531 - 0.04	3436 - 0.28
0.4	4613	4464	155120 - 0.17	43860 - 0.07	4577 - 0.34
0.5	5668	5519	200715 - 0.28	76318 - 0.16	5625 - 0.84
0.6	6725	6576	262562 - 0.50	122494 - 0.31	6670 - 1.00
0.7	7866	7717	334915 - 0.59	190806 - 0.36	7791 - 1.70
0.8	8936	8787	398996 - 0.62	276504 - 0.49	8872 - 2.14
0.9	10108	9959	472676 - 0.74	397897 - 0.57	10015 - 3.51

Some computational results

Cpu time for Euclidean graphs with n=1000

density	Horton	New-isometric
0.1	31.59	9.44
0.2	122.16	21.36
0.3	289.26	37.41
0.4	630.49	64.38
0.5	1321.30	105.48
0.6	–	152.73
0.7	–	221.61
0.8	–	331.72

Concluding remarks

- A version of our new hybrid algorithm has a $O(m^2 n / \log n)$ worst-case complexity
- In practice it performs at least as well and in general much better than other algorithms
- Since the adaptive linear independence test à la de Pina is very efficient, the version without wheel decomposition is faster
- Is there still margin for improvement? Can we do without independence test –even though it is unlikely to lead to an overall more efficient algorithm?