Counting Spanning Trees to Guide Search in Constrained Spanning Tree Problems

CP2013

Simon Brockbank Gilles Pesant Louis-Martin Rousseau

École polytechnique de Montréal, Montreal, Canada

CIRRELT, Université de Montréal, Montreal, Canada
{simon.brockbank,gilles.pesant,louis-martin.rousseau}@polymtl.ca

Wednesday, September 18th 2013

Outline

Introduction

The last decade has witnessed renewed interest in the design of robust generic branching heuristics.

- Weighted Degree
- Impact-Based Search
- Activity-Based Search
- Counting-Based Search [Zanarini and Pesant CP'07]

Solution Density

Solution density can be defined as follows :

Definition (solution density CP'07)

Given a constraint $c(x_1, \ldots, x_n)$, its number of solutions $\#c(x_1, \ldots, x_n)$, respective finite domains D_i $1 \leq i \leq n$, a variable x_i in the scope of c, and a value $d \in D_i$, we will call

$$\sigma(x_i, d, c) = \frac{\#c(x_1, \ldots, x_{i-1}, d, x_{i+1}, \ldots, x_n)}{\#c(x_1, \ldots, x_n)}$$

the *solution density* of pair (x_i, d) in c. It measures how often a certain assignment is part of a solution to c.

Solution Density

CBS requires fast algorithms to compute (exact or approximate) solution densities for each family of constraints.

Such an algorithm has been found for several constraints, such as :

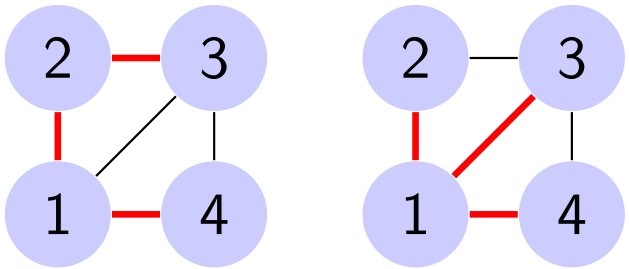
- Alldifferent [CP'07]
- Regular [CP'07]
- Knapsack [CPAIOR'08]
- Element [CPAIOR'11]
- Spead/Deviation [MODREF'11]
- Gcc [JAIR 43]

However, many constraints still require an effective algorithm to allow the use of solution density.

Spanning Tree Constraint

Definition (Spanning Tree Constraint)

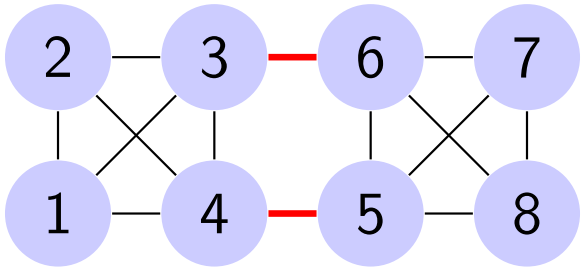
Given an undirected graph $G(V, E)$ and a set variable $T \subseteq E$, constraint $\texttt{spanningTree}(G, T)$ restricts T to be a spanning tree of G.



Our interest : How many spanning trees involve edge (2,3) ?

Spanning Tree Constraint

Finding edges that are present in many spanning trees (high solution density) might be worthwhile :



All spanning trees involve either edge (3,6) or (4,5) (or both).

Related Work

Research in the CP community about imposed tree structures has focused so far on filtering algorithms :

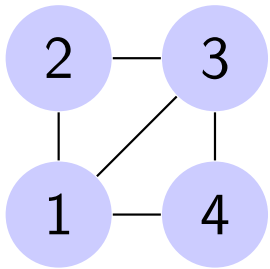
- Beldiceanu et al.[CPAIOR'12]
- Dooms and Katriel [CP'06]
- Dooms and Katriel [CPAIOR'07]
- Régin [CPAIOR'08].
- Régin et al. [CPAIOR'10]

Our paper focuses on branching heuristics.

Outline

Laplacian Matrix

The *Laplacian matrix* $L(G)$ of a graph G is formed by subtracting the adjacency matrix of G from the diagonal matrix whose i^{th} entry is equal to the degree of vertex i in G.



$$L = \begin{pmatrix} 3 & -1 & -1 & -1 \\ -1 & 2 & -1 & 0 \\ -1 & -1 & 3 & -1 \\ -1 & 0 & -1 & 2 \end{pmatrix}$$

FIGURE : The kite graph and its Laplacian matrix.

Counting Spanning Trees

The (i,j)-*minor* of a square matrix M, denoted M_{ij}, is the determinant of the sub-matrix obtained by removing from M its i^{th} row and j^{th} column.

The Laplacian matrix has the interesting property that its (i,j)-minor, for any row i and column j, is equal to the number of spanning trees of the corresponding graph.

Theorem (Kirchhoff's Matrix-Tree Theorem)

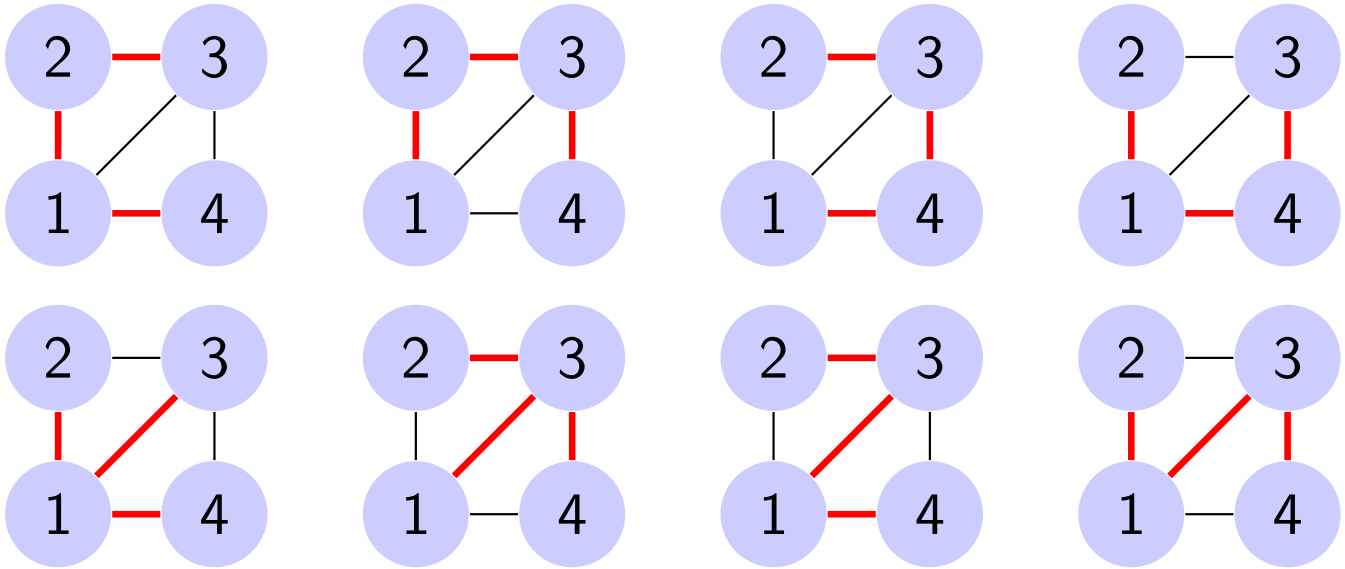
Denote by $\tau(G)$ the number of spanning trees of graph G on n vertices. For any $1 \leq i,j \leq n$,

$$\tau(G) = L_{ij}.$$

Counting Spanning Trees

$$L = \begin{pmatrix} 3 & -1 & -1 & -1 \\ -1 & 2 & -1 & 0 \\ -1 & -1 & 3 & -1 \\ -1 & 0 & -1 & 2 \end{pmatrix} \qquad L_{11} = \begin{pmatrix} 2 & -1 & 0 \\ -1 & 3 & -1 \\ 0 & -1 & 2 \end{pmatrix}$$

$$det(L_{11}) = 8$$



Counting Spanning Trees

Therefore, we can use the following formula to compute the density associated with each edge :

$$\sigma((i,j),0,\texttt{spanningTree}(G,T)) = \frac{\tau(G \setminus \{(i,j)\})}{\tau(G)}.$$

If we take a look at the matrices :

$$L(G) = \begin{pmatrix} 3 & -1 & -1 & -1 \\ -1 & 2 & -1 & 0 \\ -1 & -1 & 3 & -1 \\ -1 & 0 & -1 & 2 \end{pmatrix} L(G-\{i,j\}) = \begin{pmatrix} 2 & 0 & -1 & -1 \\ 0 & 1 & -1 & 0 \\ -1 & -1 & 3 & -1 \\ -1 & 0 & -1 & 2 \end{pmatrix}$$

Counting Spanning Trees

let's take a minor with row/column i removed (here, i = 1) :

$$L_{11}(G - \{i,j\}) = \begin{pmatrix} 1 & -1 & 0 \\ -1 & 3 & -1 \\ 0 & -1 & 2 \end{pmatrix}$$

M' is obtained from matrix M by replacing its j^{th} column, $(M)_j$, by column vector u.

Counting Spanning Trees

Example

Suppose j = 2

$$M = \begin{pmatrix} 2 & -1 & 0 \\ -1 & 3 & -1 \\ 0 & -1 & 2 \end{pmatrix} \qquad u = \begin{pmatrix} 1 \\ -1 \\ 0 \end{pmatrix}$$

$$M' = \begin{pmatrix} 1 & -1 & 0 \\ -1 & 3 & -1 \\ 0 & -1 & 2 \end{pmatrix}$$

Counting Spanning Trees

Then, the *Sherman-Morrison formula* tells us that :

$$\det(M') = (1 + e_j^\top M^{-1}(u - (M)_j))\det(M).$$

In our case $(u - (M)_j) = -e_j$, so the previous equation simplifies to $(1 - e_j^\top M^{-1} e_j)\det(M) = (1 - m_{jj}^{-1})\det(M)$.

Counting Spanning Trees

Finaly, using the *Sherman-Morrison formula*, we can simplify the preceding equation to :

$$\sigma((i,j),0,\texttt{spanningTree}(G,T)) = \frac{L'_{ii}}{L_{ii}} = \frac{(1-m_{jj}^{-1})L_{ii}}{L_{ii}} = 1-m_{jj}^{-1},$$

and of course

$$\sigma((i,j),1,\texttt{spanningTree}(G,T)) = m_{jj}^{-1}.$$

Counting Spanning Trees

- Can be repeated for every vertex.
- Takes $\mathcal{O}(\gamma n^3)$ time, γ being the size of the vertex cover.

Example

Let M be the sub-matrix of L obtained by removing its first row and column as before. Then

$$M^{-1} = \begin{pmatrix} 5/8 & 2/8 & 1/8 \\ 2/8 & 4/8 & 2/8 \\ 1/8 & 2/8 & 5/8 \end{pmatrix}$$

Outline

Updating Vertex Cover

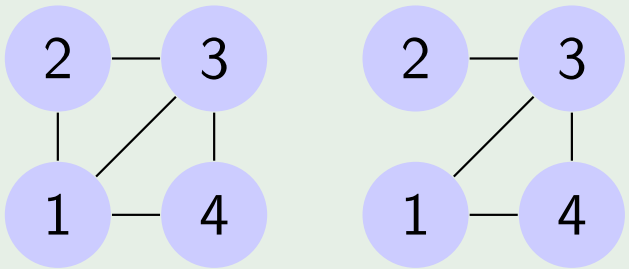
- Computation needed for every edge of the graph.
- NP-hard problem
- Construction with a heuristic greedy algorithm.
- Update through the search.
  - As some vertices are merged following edge contraction, some edges are no longer covered.
  - Add smallest vertex to the current cover.

Forbidden Edge

Example

Suppose edge $(1, 2)$ is now forbidden for the spanning tree :

$$L = \begin{pmatrix} 3 & -1 & -1 & -1 \\ -1 & 2 & -1 & 0 \\ -1 & -1 & 3 & -1 \\ -1 & 0 & -1 & 2 \end{pmatrix} \qquad L = \begin{pmatrix} 2 & 0 & -1 & -1 \\ 0 & 1 & -1 & 0 \\ -1 & -1 & 3 & -1 \\ -1 & 0 & -1 & 2 \end{pmatrix}$$

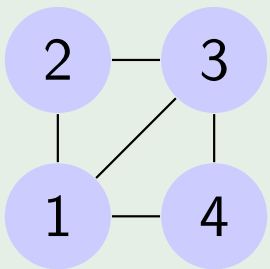


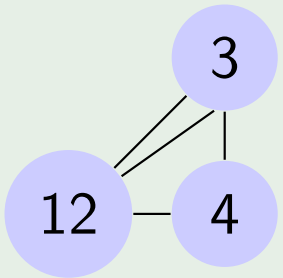
Required Edge

Example

Suppose edge $(1, 2)$ is now required for the spanning tree : we contract it and merge vertex 2 with 1 :

$$L = \begin{pmatrix} 3 & -1 & -1 & -1 \\ -1 & 2 & -1 & 0 \\ -1 & -1 & 3 & -1 \\ -1 & 0 & -1 & 2 \end{pmatrix} \qquad L = \begin{pmatrix} 3 & 0 & -2 & -1 \\ 0 & 1 & 0 & 0 \\ -2 & 0 & 3 & -1 \\ -1 & 0 & -1 & 2 \end{pmatrix}$$





Update Solution Densities

To avoid recomputing the densities from scratch, we can use the *Sherman-Morrison formula* again, this time to incrementaly update the Laplacian matrix. Hence, the previous formula updates to :

$$M'^{-1} = M^{-1} - \frac{(M^{-1}(u - (M)_i))(e_i^\top M^{-1})}{1 + e_i^\top M^{-1}(u - (M)_i)}.$$

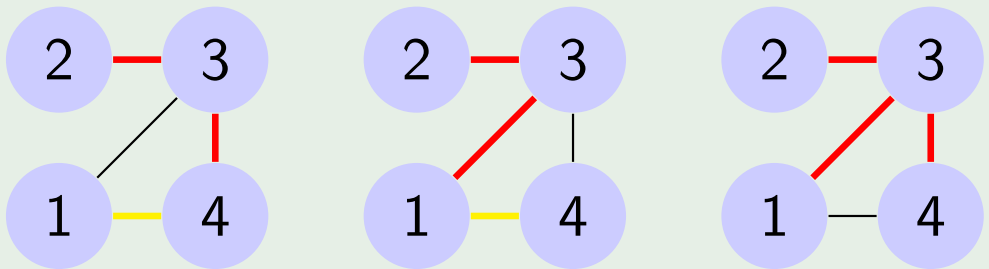
This can be computed in $\mathcal{O}(\gamma n^2)$ time.

Update Solution Densities

Sometimes possible to update in constant time (see paper).

Example

- Solution density of edge $(1, 4)$ is $\frac{5}{8}$.
- Edge $(1, 2)$ is forbidden
- Updated solution density :
  $\frac{5}{8} + (m_{42}^{-1})^2/(1 - m_{22}^{-1}) = \frac{5}{8} + (\frac{1}{8})^2/(1 - \frac{5}{8}) = \frac{2}{3}$.



Outline

Experiments

We consider the degree-constrained spanning tree problem.

Our instances come from 2 different sources :

- Random graph generator.
- Generator designed to produce hard Hamiltonian path instances for backtracking algorithms.

We compare 3 approaches :

- maxSD
- IBS
- randomized minsize

Spanning Trees degree 2

TABLE : Number of backtracks before finding a spanning tree of maximum degree 2. Each line represents an average over 10 instances.

n	maxSD	IBS	minsize
15	0.2	229.8	49.0
20	1.5	533.0	976.6
25	2.1	1772.3	5919.6
30	71.7	12517.1	91454.4
35	112.2	18405.4	139861.3

Spanning Trees degree 2

TABLE : Time in seconds before finding a spanning tree of maximum degree 2. Each line represents an average over 10 instances.

n	maxSD	iMaxSD	IBS	minsize
15	0.029	0.011	0.001	0.001
20	0.080	0.027	0.012	0.020
25	0.187	0.069	0.085	0.173
30	0.815	0.305	0.897	1.873
35	1.769	1.027	4.742	14.646

Spanning Trees degree 3

TABLE : Number of backtracks before finding a spanning tree of maximum degree 3. Each line represents an average over 10 instances.

n	maxSD	IBS	minsize
15	0.0	225.6	1.3
20	0.0	315.2	53.2
25	0.0	446.7	882.0
30	0.0	495.1	18589.8
35	0.0	566.8	20001.4

Spanning Trees degree 3

TABLE : Time in seconds before finding a spanning tree of maximum degree 3. Each line represents an average over 10 instances.

n	maxSD	iMaxSD	IBS	minsize
15	0.039	0.012	0.002	0.001
20	0.100	0.031	0.013	0.001
25	0.222	0.068	0.021	0.311
30	0.441	0.149	0.039	0.093
35	0.852	0.281	0.063	2.333

Hard Hamiltonian Graphs

TABLE : Number of backtracks before finding a Hamiltonian path in hard graphs. Each line represents an average over 10 instances.

cc	n	maxSD	IBS	minsize
3	21	0.2	7721.9	8530.5
4	28	0.1	262011.7	191195.8
5	35	0.4	162353.0	-

Hard Hamiltonian Graphs

TABLE : Time in seconds before finding a Hamiltonian path in hard graphs. Each line represents an average over 10 instances.

cc	n	maxSD	iMaxSD	IBS	minsize
3	21	0.085	0.036	0.255	0.062
4	28	0.280	0.114	26.379	3.674
5	35	0.676	0.290	586.679	-

Outline

Conclusion

We presented a new algorithm that computes exact solution densities for the spanning tree constraint :

- $\mathcal{O}(\gamma n^3)$ time.
- Possibility of solution density update in $\mathcal{O}(\gamma n^2)$ time.
- Elegant method using Laplacian matrices and matrix inversion.
- Effective approach to solve constrained spanning tree problems.

Future Work

There are several application areas that involve finding spanning trees :

- Network Design
- Telecommunication
- Transportation

Examples of these problems are :

- the degree-constrained problem (including Hamiltonian path)
- the diameter-constrained minimum spanning tree.