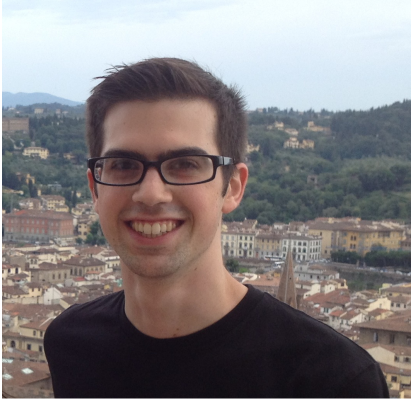
An Experimental Evaluation of the Best-of-Many Christofides' Algorithm for the Traveling Salesman Problem

David P. Williamson
Cornell University

Joint work with Kyle Genova, Cornell University

July 14, 2015
ISMP 2015
Pittsburgh, PA, USA



Kyle will be applying to CS grad schools this coming year.
Look for his application!

The traveling salesman problem

Traveling Salesman Problem (TSP)

Input:

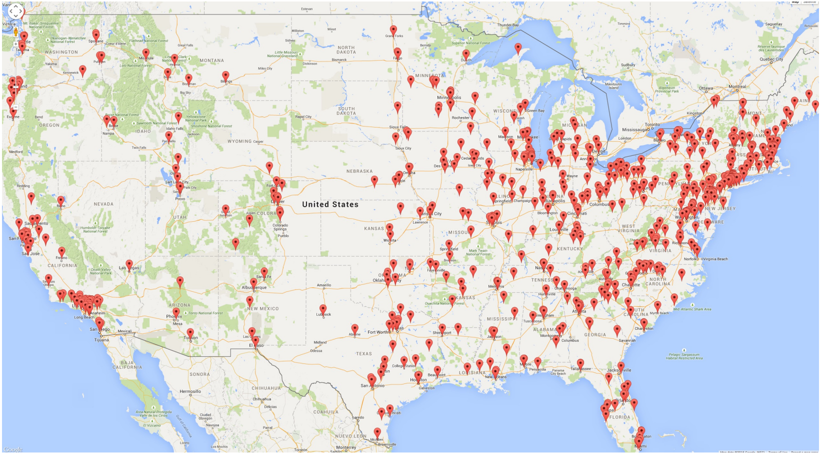
- A complete, undirected graph $G = (V, E)$;
- Edge costs $c(i,j) \geq 0$ for all $e = (i,j) \in E$.

Goal: Find the min-cost tour that visits each city exactly once.

Costs are *symmetric* ($c(i,j) = c(j,i)$) and obey the *triangle inequality* ($c(i,k) \leq c(i,j) + c(j,k)$).

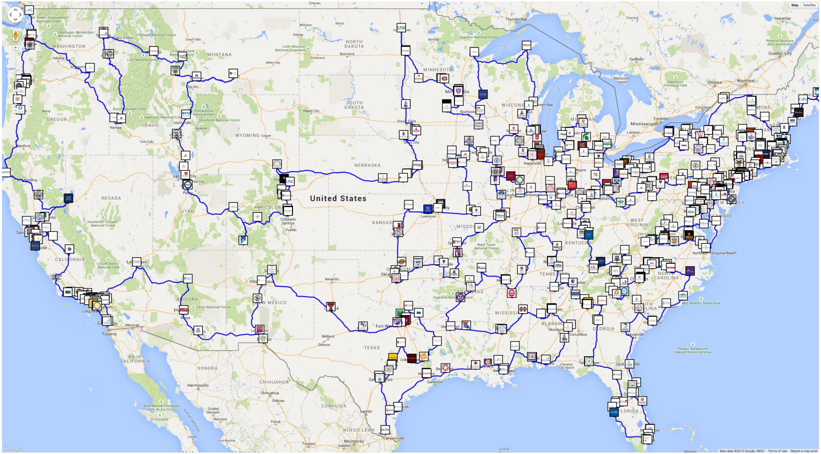
Asymmetric TSP (ATSP) input has complete directed graph, and $c(i,j)$ may not equal $c(j,i)$.

The traveling salesman problem



From Bill Cook, tour of 647 US colleges
(`www.math.uwaterloo.ca/tsp/college`)

The traveling salesman problem



From Bill Cook, tour of 647 US colleges
(`www.math.uwaterloo.ca/tsp/college`)

Approximation Algorithms

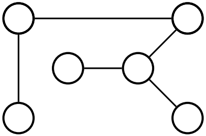
Definition

An α-approximation algorithm is a polynomial-time algorithm that returns a solution of cost at most α times the cost of an optimal solution.

Long known: A $\frac{3}{2}$-approximation algorithm due to Christofides (1976). No better approximation algorithm yet known.

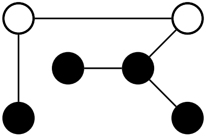
Christofides' algorithm

Compute minimum spanning tree (MST) F on G, then compute a minimum-cost perfect matching M on odd-degree vertices of T. "Shortcut" Eulerian traversal in resulting Eulerian graph of $F \cup M$.



Christofides' algorithm

Compute minimum spanning tree (MST) F on G, then compute a minimum-cost perfect matching M on odd-degree vertices of T.
"Shortcut" Eulerian traversal in resulting Eulerian graph of $F \cup M$.

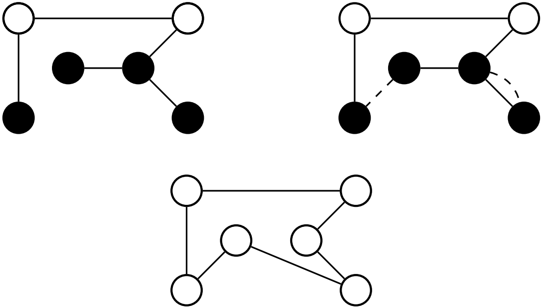


Christofides' algorithm

Compute minimum spanning tree (MST) F on G, then compute a minimum-cost perfect matching M on odd-degree vertices of T.
"Shortcut" Eulerian traversal in resulting Eulerian graph of $F \cup M$.

Christofides' algorithm

Compute minimum spanning tree (MST) F on G, then compute a minimum-cost perfect matching M on odd-degree vertices of T. "Shortcut" Eulerian traversal in resulting Eulerian graph of $F \cup M$.



Special cases

Some progress in the case of *graph* TSP:
input is a graph $G = (V, E)$, cost $c(i, j)$ is number of edges in shortest path from i to j.

Oveis Gharan, Saberi, Singh	(2011)	$\frac{3}{2} - \epsilon$
Mömke, Svensson	(2011)	1.462
Mucha	(2012)	$\frac{13}{9} \approx 1.444$
Sebő, Vygen	(2012)	$\frac{7}{5} = 1.4$

Special cases

Also progress on s-t path TSP:
Usual TSP input plus $s, t \in V$, find a min-cost path from s to t visiting all other nodes in between.

Hoogeveen	(1991)	$\frac{5}{3}$
An, Kleinberg, Shmoys	(2012)	$\frac{1+\sqrt{5}}{2} \approx 1.618$
Sebő	(2013)	$\frac{8}{5} = 1.6$
Vygen	(2015)	1.5999

A central idea

Idea: run Christofides', but start with tree determined by LP relaxation of TSP (or s-t path TSP), the *Subtour LP*.

$$
\begin{array}{lll}
\text{Min} & \displaystyle\sum_{e\in E} c_e x_e & \\
\text{subject to:} & x(\delta(v)) = 2, & \forall v \in V, \\
& x(\delta(S)) \geq 2, & \forall S \subset V, S \neq \emptyset, \\
& 0 \leq x_e \leq 1, & \forall e \in E,
\end{array}
$$

where $\delta(S)$ is the set of all edges with exactly one endpoint in S, and $x(F) = \sum_{e\in F} x_e$.

The Subtour LP

$$\text{Min} \quad \sum_{e \in E} c_e x_e$$

subject to:

$$\begin{aligned} x(\delta(v)) &= 2, && \forall v \in V, \\ x(\delta(S)) &\geq 2, && \forall S \subset V, S \neq \emptyset, \\ 0 \leq x_e &\leq 1, && \forall e \in E. \end{aligned}$$

For x feasible for LP, $\frac{n-1}{n} x$ in spanning tree polytope

$$\{x \in \Re^{|E|} : x(E) = n-1, \;\; x(E(S)) \leq |S| - 1 \;\; \forall S \subseteq V, |S| \geq 2\},$$

where $E(S)$ is the set of edges with both endpoints in S.

Best-of-Many Christofides'

For Subtour LP soln. x^*, compute decomposition of $\frac{n-1}{n}x^*$ into convex combination of spanning trees $F_1, \ldots, F_k$, that

$$\frac{n-1}{n}x^* = \sum_{i=1}^{k} \lambda_i \chi_{F_i},$$

where $\lambda_i \geq 0$, $\sum_{i=1}^{k} \lambda_i = 1$, and $\chi_F \in \{0,1\}^{|E|}$ the characteristic vector of edges in F.

Then run Christofides' algorithm on each F_i: find matching M_i, shortcut $F_i \cup M_i$. Return best tour found.

Originally proposed by Oveis Gharan, Saberi, Singh (2011), used in An, Kleinberg, Shmoys (2012).

An alternate perspective

An alternate perspective on Best-of-Many Christofides: for Subtour LP soln. x^*, have an *implicit* convex combination $F_1, \ldots, F_k$,

$$\frac{n-1}{n} x^* = \sum_{i=1}^{k} \lambda_i \chi_{F_i},$$

and ability to *sample* a tree F_i with probability λ_i. Then run Christofides' algorithm on F_i, so that expected cost of tree is at most LP solution, and

$$\Pr[\text{edge } e \text{ in sampled tree}] \leq x_e^*.$$

Advantage: Don't need to explicitly construct the convex combination.

The question

Best-of-Many Christofides' (BoMC) is *provably* better than Christofides' for s-t path TSP. What about TSP?

Is BoMC *empirically* better than Christofides'?

The algorithms

We implement algorithms to do the following:

- Run the standard Christofides' algorithm (Christofides 1976);

The algorithms

We implement algorithms to do the following:

- Run the standard Christofides' algorithm (Christofides 1976);
- Construct explicit convex combination via column generation (An 2012);

The algorithms

We implement algorithms to do the following:

- Run the standard Christofides' algorithm (Christofides 1976);
- Construct explicit convex combination via column generation (An 2012);
- Construct explicit convex combination via *splitting off* (Frank 2011, Nagamochi, Ibaraki 1997);

The algorithms

We implement algorithms to do the following:

- Run the standard Christofides' algorithm (Christofides 1976);
- Construct explicit convex combination via column generation (An 2012);
- Construct explicit convex combination via *splitting off* (Frank 2011, Nagamochi, Ibaraki 1997);
- Add sampling scheme *SwapRound* to both of above; gives negative correlation properties (Chekuri, Vondrák, Zenklusen 2010);

The algorithms

We implement algorithms to do the following:

- Run the standard Christofides' algorithm (Christofides 1976);
- Construct explicit convex combination via column generation (An 2012);
- Construct explicit convex combination via *splitting off* (Frank 2011, Nagamochi, Ibaraki 1997);
- Add sampling scheme *SwapRound* to both of above; gives negative correlation properties (Chekuri, Vondrák, Zenklusen 2010);
- Compute and sample from *maximum entropy distribution* (Asadpour, Goemans, Madry, Oveis Gharan, Saberi 2010).

Code available on github (pointer on the last slide).

The instances

We run these algorithms on several types of instances:

- 59 Euclidean TSPLIB (Reinelt 1991) instances up to 2103 vertices;
- 5 non-Euclidean TSPLIB instances (gr120, si175, si535, pa561, si1032);
- 39 Euclidean VLSI instances (Rohe) up to 3694 vertices;
- 9 graph TSP instances (Kunegis 2013) up to 1615 vertices.

Executive summary

- Standard Christofides' in general the worst; 9-10% away from optimal (similar to results in Johnson and McGeoch 2002). 12% away on graph TSP instances (see also Walter and Wegmann 2014).
- BoMC about 3-7% away from optimal on Euclidean instances, 2-3% away from optimal for non-Euclidean, $< 1\%$ for graph TSP instances.
- Maximum entropy sampling the best, though splitting-off $+$ SwapRound also very good.

Outline

1. Introduction
2. The algorithms
3. The instances
4. The results
5. Some conclusions

Standard Christofides'

Use Prim's algorithm to find MST; if Euclidean instance, first find Delaunay triangulation using Triangle (Shewchuk 1996)

Compute matching via Blossom V code of Kolmogorov (2009).

Do simple optimization on shortcutting.

Column generation

From An's Ph.D. thesis. We compute the Subtour LP solution x^* using Concorde (Applegate, Bixby, Chvátal, Cook). Then consider:

$$\text{Min} \quad \sum_{e \in E} s_e$$

subject to:

$$\begin{aligned} \sum_{T: e \in T} y_T + s_e = \frac{n-1}{n} x_e^*, & \qquad \forall e \in E, \\ y_T \geq 0, & \qquad \forall T. \\ s_e \geq 0, & \qquad \forall e \in E. \end{aligned}$$

Optimal solution is $s = 0$.

Column generation

Dual is:

$$\text{Max} \quad \frac{n-1}{n} \sum_{e \in E} x_e^* z_e$$

subject to:

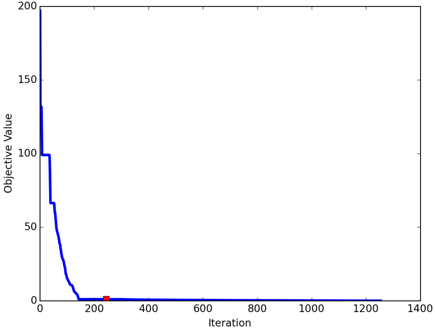
$$\sum_{e \in T} z_e \leq 0, \qquad \forall T,$$

$$z_e \leq 1, \qquad \forall e \in E.$$

Pricing problem is computing max-weight spanning tree on dual solution z_e.

Column generation

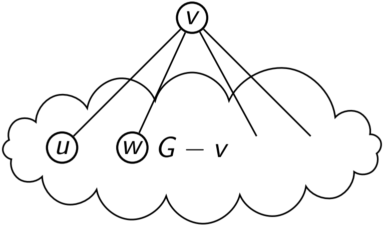
Because convergence to optimal takes a long time, we stop early if there isn't progress in 100 iterations.



Early termination for TSPLIB D198

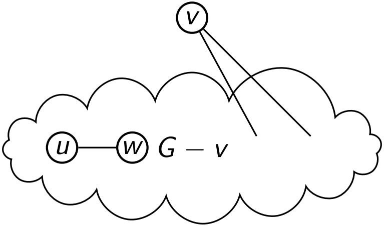
Splitting off

Consider Eulerian multigraph represented by Kx^* for some integer K, graph will be $2K$-edge-connected. Lovász (1976) shows that for Eulerian multigraphs, vertex v, can *split off* edges from v: remove edges (u, v), (v, w), add edge (u, w) such that remaining vertices are still $2K$-edge-connected.



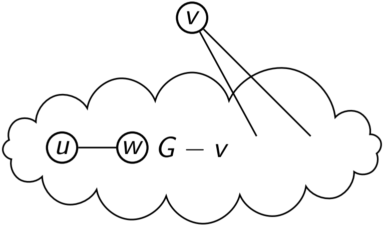
Splitting off

Consider Eulerian multigraph represented by Kx^* for some integer K, graph will be $2K$-edge-connected. Lovász (1976) shows that for Eulerian multigraphs, vertex v, can *split off* edges from v: remove edges (u, v), (v, w), add edge (u, w) such that remaining vertices are still $2K$-edge-connected.



Splitting off

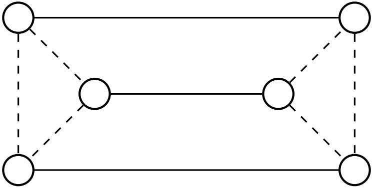
Consider Eulerian multigraph represented by Kx^* for some integer K, graph will be $2K$-edge-connected. Lovász (1976) shows that for Eulerian multigraphs, vertex v, can *split off* edges from v: remove edges (u, v), (v, w), add edge (u, w) such that remaining vertices are still $2K$-edge-connected.



Nagamochi and Ibaraki (1997) show how to compute a complete splitting off from v in $O(nm + n^2 \log n)$ time.

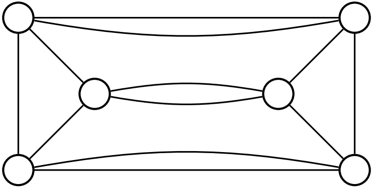
Splitting off

We split off all vertices except two, then inductively construct a collection of trees by *lifting back* the split off edges.



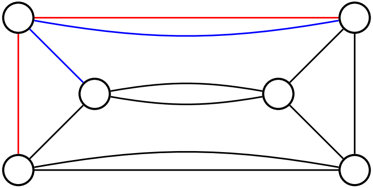
Splitting off

We split off all vertices except two, then inductively construct a collection of trees by *lifting back* the split off edges.



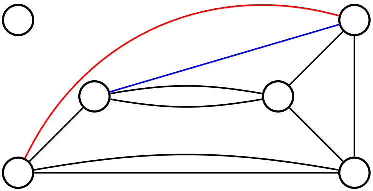
Splitting off

We split off all vertices except two, then inductively construct a collection of trees by *lifting back* the split off edges.



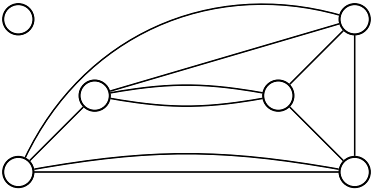
Splitting off

We split off all vertices except two, then inductively construct a collection of trees by *lifting back* the split off edges.



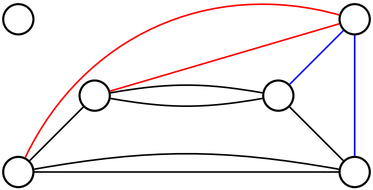
Splitting off

We split off all vertices except two, then inductively construct a collection of trees by *lifting back* the split off edges.



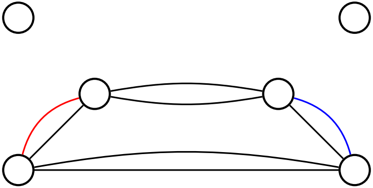
Splitting off

We split off all vertices except two, then inductively construct a collection of trees by *lifting back* the split off edges.



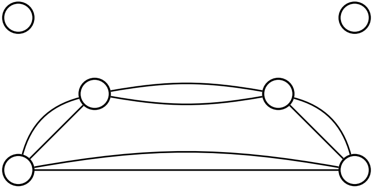
Splitting off

We split off all vertices except two, then inductively construct a collection of trees by *lifting back* the split off edges.



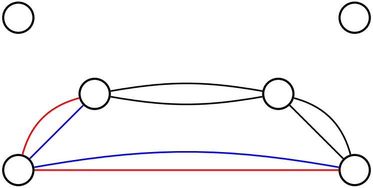
Splitting off

We split off all vertices except two, then inductively construct a collection of trees by *lifting back* the split off edges.



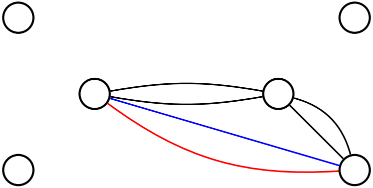
Splitting off

We split off all vertices except two, then inductively construct a collection of trees by *lifting back* the split off edges.



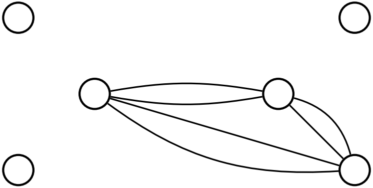
Splitting off

We split off all vertices except two, then inductively construct a collection of trees by *lifting back* the split off edges.



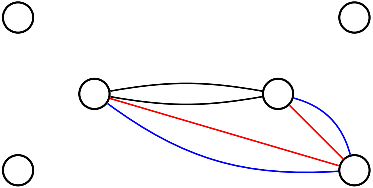
Splitting off

We split off all vertices except two, then inductively construct a collection of trees by *lifting back* the split off edges.



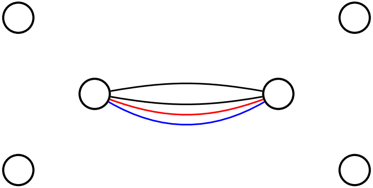
Splitting off

We split off all vertices except two, then inductively construct a collection of trees by *lifting back* the split off edges.



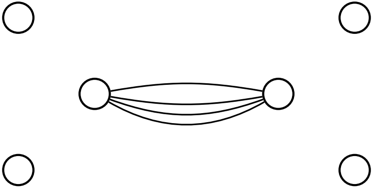
Splitting off

We split off all vertices except two, then inductively construct a collection of trees by *lifting back* the split off edges.



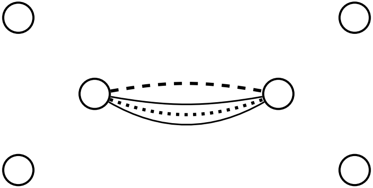
Splitting off

We split off all vertices except two, then inductively construct a collection of trees by *lifting back* the split off edges.



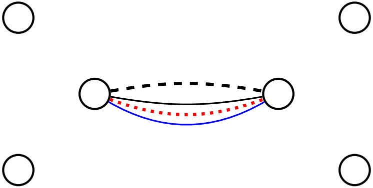
Splitting off

We split off all vertices except two, then inductively construct a collection of trees by *lifting back* the split off edges.



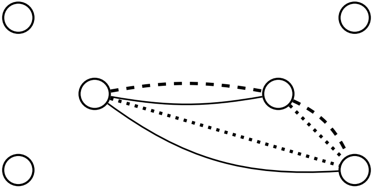
Splitting off

We split off all vertices except two, then inductively construct a collection of trees by *lifting back* the split off edges.



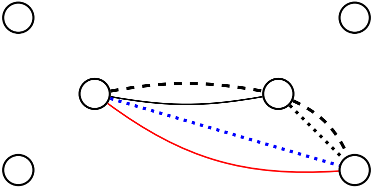
Splitting off

We split off all vertices except two, then inductively construct a collection of trees by *lifting back* the split off edges.



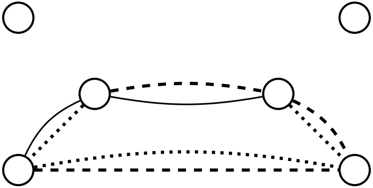
Splitting off

We split off all vertices except two, then inductively construct a collection of trees by *lifting back* the split off edges.



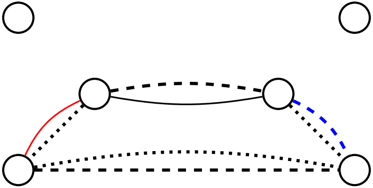
Splitting off

We split off all vertices except two, then inductively construct a collection of trees by *lifting back* the split off edges.



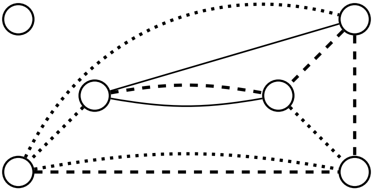
Splitting off

We split off all vertices except two, then inductively construct a collection of trees by *lifting back* the split off edges.



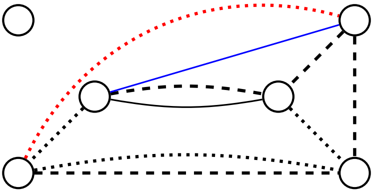
Splitting off

We split off all vertices except two, then inductively construct a collection of trees by *lifting back* the split off edges.



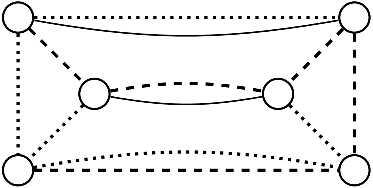
Splitting off

We split off all vertices except two, then inductively construct a collection of trees by *lifting back* the split off edges.



Splitting off

We split off all vertices except two, then inductively construct a collection of trees by *lifting back* the split off edges.



SwapRound

SwapRound (Chekuri, Vondrák, Zenklusen 2010) randomly samples a spanning tree given an explicit convex combination of trees. For any fixed set A of edges, the edges of the sampled tree appearing in A are *negatively correlated*; if X_e is the event edge e appears in the tree, then

$$E\left[\bigwedge_{e\in A} X_e\right] \leq \prod_{e\in A} \Pr[X_e].$$

Negative correlation allows the proof of concentration of measure results (used by Asadpour et al. for ATSP).

SwapRound

SwapRound maintains a single tree (initially the first tree of the combination), and iteratively calls MergeBasis on the current tree and the next tree in the combination.

MergeBasis randomly swaps edges (*base exchanges*) between the two trees until the two are identical.

We run 1000 samples per instance, using four threads.

The maximum entropy distribution

$$\text{Inf} \quad \sum_{T \in \mathcal{T}} p(T) \log p(T)$$

subject to:

$$\begin{aligned} \sum_{T: e \in T} p(T) &= \frac{n-1}{n} x_e^*, && \forall e \in E, \\ \sum_{T \in \mathcal{T}} p(T) &= 1 \\ p(T) &\geq 0, && \forall T. \end{aligned}$$

Asadpour et al. show that there exist γ_e such that the optimal $p(T) \sim \exp\left(\sum_{e \in T} \gamma_e\right)$, and give an algorithm to approximately calculate the γ. They also give a poly-time algorithm to sample a tree T given the γ.

The maximum entropy distribution

We implemented the algorithms of Asadpour et al. but also algorithms in a code of Oveis Gharan. The latter were faster in practice.

As with SwapRound, we compute 1000 samples for each instance in parallel with four threads.

The experiments

The algorithms were implemented in C++, run on a machine with a 4.00Ghz Intel i7-875-K processor with 8GB DDR3 memory.

We run these algorithms on several types of instances:

- 59 Euclidean TSPLIB (Reinelt 1991) instances up to 2103 vertices (avg. 524);
- 5 non-Euclidean TSPLIB instances (gr120, si175, si535, pa561, si1032);
- 39 Euclidean VLSI instances (Rohe) up to 3694 vertices (avg. 1473);
- 9 graph TSP instances (Kunegis 2013) up to 1615 vertices (avg. 363).

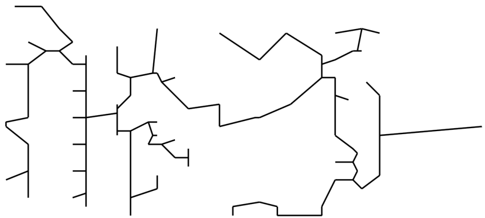
The results

	Std	ColGen		ColGen+SR	
		Best	Ave	Best	Ave
TSPLIB (E)	9.56%	4.03%	6.44%	3.45%	6.24%
VLSI	9.73%	7.00%	8.51%	6.40%	8.33%
TSPLIB (N)	5.40%	2.73%	4.41%	2.22%	4.08%
Graph	12.43%	0.57%	1.37%	0.39%	1.29%

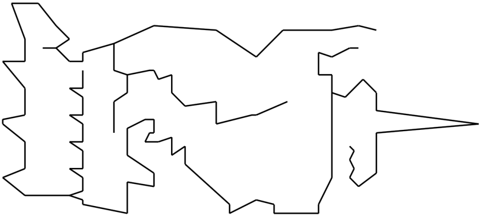
	MaxEnt		Split		Split+SR	
	Best	Ave	Best	Ave	Best	Ave
TSPLIB (E)	3.19%	6.12%	5.23%	6.27%	3.60%	6.02%
VLSI	5.47%	7.61%	6.60%	7.64%	5.48%	7.52%
TSPLIB (N)	2.12%	3.99%	2.92%	3.77%	1.99%	3.82%
Graph	0.31%	1.23%	0.88%	1.77%	0.33%	1.20%

Costs given as percentages in excess of optimal.

The results



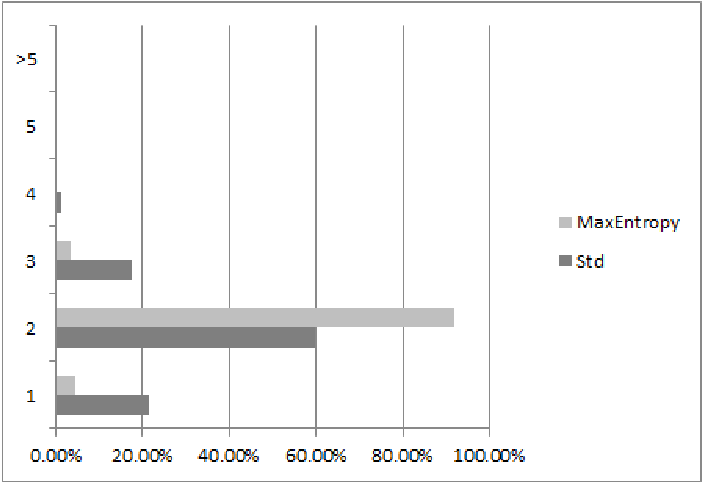
Standard Christofides MST (Rohe VLSI instance XQF131)



Splitting off + SwapRound

The results

BoMC yields more vertices in the tree of degree two.



The results

So while the tree costs more (as percentage of optimal tour)...

	Std	BOM
TSPLIB (E)	87.47%	98.57%
VLSI	89.85%	98.84%
TSPLIB (N)	92.97%	99.36%
Graph	79.10%	98.23%

The results

...the matching costs much less.

	Std	CG	CG+SR	MaxE	Split	Sp+SR
TSPLIB (E)	31.25%	11.43%	11.03%	10.75%	10.65%	10.41%
VLSI	29.98%	14.30%	14.11%	12.76%	12.78%	12.70%
TSPLIB (N)	24.15%	9.67%	9.36%	8.75%	8.77%	8.56%
Graph	39.31%	5.20%	4.84%	4.66%	4.34%	4.49%

Conclusion

Q: Are there empirical reasons to think BoMC might be provably better than Christofides' algorithm?

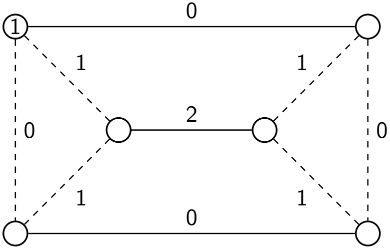
Conclusion

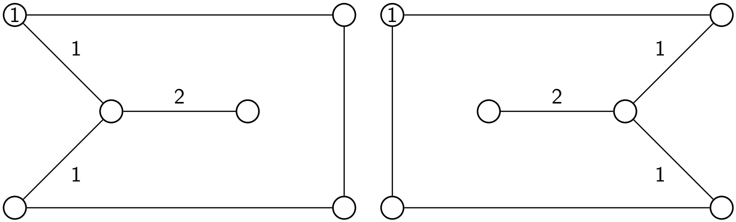
Q: Are there empirical reasons to think BoMC might be provably better than Christofides' algorithm?
A: Yes.

Maximum entropy sampling, or splitting off with SwapRound seem like the best candidates.

Conclusion

However, we have to be careful, as the following, very recent, example of Schalekamp and van Zuylen shows.





Conclusions

So it seems that randomization, or at least, careful construction of the convex combination is needed.

Conclusions

So it seems that randomization, or at least, careful construction of the convex combination is needed.

Vygen (2015) also uses careful construction to improve s-t path TSP from 1.6 to 1.5999.

Conclusions

So it seems that randomization, or at least, careful construction of the convex combination is needed.

Vygen (2015) also uses careful construction to improve s-t path TSP from 1.6 to 1.5999.

If we want to use the best sample from Max Entropy or SwapRound, then might need to prove some tail bounds.

Thanks for your attention.

Paper available at
http://arxiv.org/abs/1506.07776.

Code available at
http://github.com/kylegenova/best-of-many.

Feedback? Contact me at
dpw@cs.cornell.edu.