Improved Inapproximability for TSP

Michael Lampis
KTH Royal Institute of Technology

August 15, 2012
The Traveling Salesman Problem

Input:

- An edge-weighted graph $G(V, E)$

Objective:

- Find an ordering of the vertices $v_1, v_2, \ldots, v_n$ such that $d(v_1, v_2) + d(v_2, v_3) + \ldots + d(v_n, v_1)$ is minimized.

- $d(v_i, v_j)$ is the shortest-path distance of $v_i, v_j$ on $G$
The Traveling Salesman Problem

Improved Inapproximability for TSP – APPROX 2012
The Traveling Salesman Problem
The Traveling Salesman Problem
The Traveling Salesman Problem

Improved Inapproximability for TSP – APPROX 2012
The Traveling Salesman Problem
TSP Approximations – Upper bounds

- $\frac{3}{2}$ approximation (Christofides 1976)

For graphic (un-weighted) case

- $\frac{3}{2} - \epsilon$ approximation (Oveis Gharan et al. FOCS '11)

- 1.461 approximation (Mömke and Svensson FOCS '11)

- $\frac{13}{9}$ approximation (Mucha STACS '12)

- 1.4 approximation (Sebö and Vygen arXiv '12)
TSP Approximations – Lower bounds

- Problem is APX-hard (Papadimitriou and Yannakakis ’93)
- $\frac{5381}{5380}$-inapproximable (Engebretsen STACS ’99)
- $\frac{3813}{3812}$-inapproximable (Böckenhauer et al. STACS ’00)
- $\frac{220}{219}$-inapproximable (Papadimitriou and Vempala STOC ’00, Combinatorica ’06)
- Problem is APX-hard (Papadimitriou and Yannakakis '93)
- $\frac{5381}{5380}$-inapproximable (Engebretsen STACS '99)
- $\frac{3813}{3812}$-inapproximable (Böckenhauer et al. STACS '00)
- $\frac{220}{219}$-inapproximable (Papadimitriou and Vempala STOC '00, Combinatorica '06)

This talk:

**Theorem**
There is no $\frac{185}{184}$-approximation algorithm for TSP, unless $P=NP$. 
We reduce some inapproximable CSP (e.g. MAX-3SAT) to TSP.
First, design some gadgets to represent the clauses
Then, add some choice vertices to represent truth assignments to variables
For each variable, create a path through clauses where it appears positive.
...and another path for its negative appearances
Reduction Technique

Variables

Gadgets for clauses
A truth assignment dictates a general path
Reduction Technique

Variables

Gadgets for clauses

$x_1$ $x_2$ $x_3$

$T$ $T$

$F$ $F$
We must make sure that gadgets are cheaper to traverse if corresponding clause is satisfied.
For the converse direction we must make sure that "cheating" tours are not optimal!
How to ensure consistency

- Papadimitriou and Vempala design a gadget for Parity.
- They eliminate variable vertices altogether.
- Consistency is achieved by hooking up gadgets "randomly"
  - In fact gadgets that share a variable are connected according to the structure dictated by a special graph
  - The graph is called a "pusher". Its existence is proved using the probabilistic method.
How to ensure consistency

- Basic idea here: consistency would be easy if each variable occurred at most $c$ times, $c$ a constant.
  - Cheating would only help a tour "fix" a bounded number of clauses.
How to ensure consistency

- Basic idea here: consistency would be easy if each variable occurred at most $c$ times, $c$ a constant.
  - Cheating would only help a tour "fix" a bounded number of clauses.

- We will rely on techniques and tools used to prove inapproximability for bounded-occurrence CSPs.
  - Main tool: an "amplifier graph" construction due to Berman and Karpinski.
How to ensure consistency

- Basic idea here: consistency would be easy if each variable occurred at most $c$ times, $c$ a constant.
  - Cheating would only help a tour "fix" a bounded number of clauses.

- We will rely on techniques and tools used to prove inapproximability for bounded-occurrence CSPs.
  - Main tool: an "amplifier graph" construction due to Berman and Karpinski.

- Result: an easier hardness proof that can be broken down into independent pieces, and also gives an improved bound.
We start from an instance of MAX-E3-LIN2. Given a set of linear equations (mod 2) each of size three satisfy as many as possible. Known to be 2-inapproximable (Håstad).
We use the Berman-Karpinski amplifier construction to obtain an instance where each variable appears exactly 5 times (and most equations have size 2).
A simple trick reduces this to the 1in3 predicate.
From this instance we construct a graph.
From this instance we construct a graph.

Rest of this talk: some more details about the construction.
1-in-3-SAT

Input:
A set of clauses \((l_1 \lor l_2 \lor l_3)\), \(l_1, l_2, l_3\) literals.

Objective:
A clause is satisfied if exactly one of its literals is true. Satisfy as many clauses as possible.

- Easy to reduce MAX-LIN2 to this problem.
  - Especially for size two equations \((x + y = 1) \iff (x \lor y)\).
- Naturally gives gadget for TSP
  - In TSP we’d like to visit each vertex at least once, but not more than once (to save cost)
TSP and Euler tours

Improved Inapproximability for TSP – APPROX 2012
TSP and Euler tours

Improved Inapproximability for TSP – APPROX 2012
TSP and Euler tours
TSP and Euler tours

- A TSP tour gives an Eulerian multi-graph composed with edges of $G$.

- An Eulerian multi-graph composed with edges of $G$ gives a TSP tour.
  - TSP $\equiv$ Select a multiplicity for each edge so that the resulting multi-graph is Eulerian and total cost is minimized
  - **Note**: no edge is used more than twice
We would like to be able to dictate in our construction that a certain edge has to be used \textit{at least} once.
If we had directed edges, this could be achieved by adding a dummy intermediate vertex.
Here, we add many intermediate vertices and evenly distribute the weight $w$ among them. Think of $B$ as very large.
At most one of the new edges may be unused, and in that case all others are used twice.
In that case, adding two copies of that edge to the solution doesn’t hurt much (for $B$ sufficiently large).
Let's design a gadget for \((x \lor y \lor z)\)
1in3 Gadget

Connect them . . .
1in3 Gadget

... with forced edges
1in3 Gadget

The gadget is a connected component. A good tour visits it once.
1in3 Gadget

... like this
This corresponds to an unsatisfied clause
This corresponds to a dishonest tour
The dishonest tour pays this edge twice. How expensive must it be before cheating becomes suboptimal?

Note that $w = 10$ suffices, since the two cheating variables appear in at most 10 clauses.
Construction

High-level view: construct an origin $s$ and two terminal vertices for each variable.
Connect them with forced edges
Construction

Add the gadgets
An honest traversal for $x_2$ looks like this.
A dishonest traversal looks like this...
There are as many doubly-used forced edges as affected variables

\[ \rightarrow w \leq 5 \]
There are as many doubly-used forced edges as affected variables

\[ \rightarrow w \leq 5 \]

In fact, no need to write off affected clauses. Use random assignment for cheated variables and some of them will be satisfied
Many details missing

- Dishonest variables are set randomly but not independently to ensure that some clauses are satisfied with probability 1.
- The structure of the instance (from BK amplifier) must be taken into account to calculate the final constant.
Many details missing

- Dishonest variables are set randomly but not independently to ensure that some clauses are satisfied with probability 1.
- The structure of the instance (from BK amplifier) must be taken into account to calculate the final constant.

**Theorem:**
There is no $\frac{185}{184}$ approximation algorithm for TSP, unless P=NP.
Conclusions – Open problems

- A simpler reduction for TSP and a better inapproximability threshold
  - But, constant still very low!

Future work

- Better amplifier constructions?
- Get rid of 1in3 SAT?
- ATSP
Questions?