Traveling Salesman Problem

Math 482, Lecture 35

Misha Lavrov

May 1, 2020

The traveling salesman problem

We are given:

1. Cities numbered $1, 2, \ldots, n$ (vertices).

The traveling salesman problem

We are given:

1. Cities numbered $1, 2, \ldots, n$ (vertices).
2. A cost c_{ij} to travel from city i to city j.

The traveling salesman problem

We are given:

1. Cities numbered $1, 2, \dots, n$ (vertices).
2. A cost c_{ij} to travel from city i to city j.

Goal: find a tour of all n cities, starting and ending at city 1, with the cheapest cost.

The traveling salesman problem

We are given:

1. Cities numbered $1, 2, \ldots, n$ (vertices).
2. A cost c_{ij} to travel from city i to city j.

Goal: find a tour of all n cities, starting and ending at city 1, with the cheapest cost.

Common assumptions:

1. $c_{ij} = c_{ji}$: costs are symmetric and direction of the tour doesn't matter.

The traveling salesman problem

We are given:

1. Cities numbered $1, 2, \ldots, n$ (vertices).
2. A cost c_{ij} to travel from city i to city j.

Goal: find a tour of all n cities, starting and ending at city 1, with the cheapest cost.

Common assumptions:

1. $c_{ij} = c_{ji}$: costs are symmetric and direction of the tour doesn't matter.
2. $c_{ij} + c_{jk} \geq c_{ik}$: triangle inequality.

The traveling salesman problem

We are given:

1. Cities numbered $1, 2, \ldots, n$ (vertices).
2. A cost c_{ij} to travel from city i to city j.

Goal: find a tour of all n cities, starting and ending at city 1, with the cheapest cost.

Common assumptions:

1. $c_{ij} = c_{ji}$: costs are symmetric and direction of the tour doesn't matter.
2. $c_{ij} + c_{jk} \geq c_{ik}$: triangle inequality.

Important special case: cities are points in the plane, and c_{ij} is the distance from i to j.

An incomplete ILP formulation: “local constraints”

Describe a tour by variables $x_{ij} \in \{0, 1\}$: $x_{ij} = 1$ if tour goes from i to j, $x_{ij} = 0$ otherwise.

An incomplete ILP formulation: "local constraints"

Describe a tour by variables $x_{ij} \in \{0, 1\}$: $x_{ij} = 1$ if tour goes from i to j, $x_{ij} = 0$ otherwise.

Minimize the total cost of the tour:

$$\text{minimize} \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} x_{ij}.$$

An incomplete ILP formulation: “local constraints”

Describe a tour by variables $x_{ij} \in \{0, 1\}$: $x_{ij} = 1$ if tour goes from i to j, $x_{ij} = 0$ otherwise.

Minimize the total cost of the tour:

$$\text{minimize} \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} x_{ij}.$$

Enter each city exactly once:

$$\sum_{\substack{1 \le i \le n \\ i \ne j}} x_{ij} = 1 \qquad \text{for each } j = 1, 2, \ldots, n.$$

An incomplete ILP formulation: “local constraints”

Describe a tour by variables $x_{ij} \in \{0, 1\}$: $x_{ij} = 1$ if tour goes from i to j, $x_{ij} = 0$ otherwise.

Minimize the total cost of the tour:

$$\text{minimize} \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} x_{ij}.$$

Enter each city exactly once:

$$\sum_{\substack{1 \le i \le n \\ i \ne j}} x_{ij} = 1 \qquad \text{for each } j = 1, 2, \ldots, n.$$

Leave each city exactly once:

$$\sum_{\substack{1 \le k \le n \\ k \ne j}} x_{jk} = 1 \qquad \text{for each } j = 1, 2, \ldots, n.$$

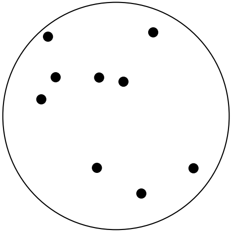
Subtours

The local constraints do not guarantee that we actually find a tour of all n cities!

Subtours

The local constraints do not guarantee that we actually find a tour of all n cities!

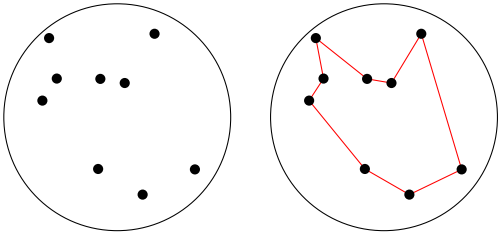
Here is a TSP instance with 9 cities; assume that cost is Euclidean distance.



Subtours

The local constraints do not guarantee that we actually find a tour of all n cities!

Here is a TSP instance with 9 cities; assume that cost is Euclidean distance.

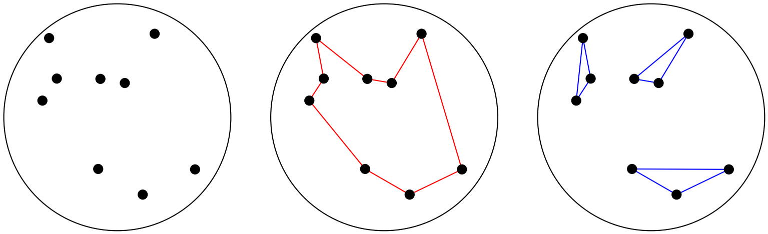


The optimal tour is shown in red.

Subtours

The local constraints do not guarantee that we actually find a tour of all n cities!

Here is a TSP instance with 9 cities; assume that cost is Euclidean distance.



The optimal tour is shown in red.

The optimal solution to the local constraints is in blue. It has three *subtours* that are not connected to each other.

Subtour elimination constraints

Problem: the local constraints allow for subtours that don't visit all n cities.

Subtour elimination constraints

Problem: the local constraints allow for subtours that don't visit all n cities.

Solution #1 (Dantzig, Fulkerson, Johnson, 1954): for every set S of cities, add a constraint saying that the tour leaves S at least once.

Subtour elimination constraints

Problem: the local constraints allow for subtours that don't visit all n cities.

Solution #1 (Dantzig, Fulkerson, Johnson, 1954): for every set S of cities, add a constraint saying that the tour leaves S at least once.

For every $S \subseteq \{1, 2, \ldots, n\}$ with $1 \leq |S| \leq n - 1$:

$$\sum_{i \in S} \sum_{j \notin S} x_{ij} \geq 1.$$

Subtour elimination constraints

Problem: the local constraints allow for subtours that don't visit all n cities.

Solution #1 (Dantzig, Fulkerson, Johnson, 1954): for every set S of cities, add a constraint saying that the tour leaves S at least once.

For every $S \subseteq \{1, 2, \ldots, n\}$ with $1 \leq |S| \leq n - 1$:

$$\sum_{i \in S} \sum_{j \notin S} x_{ij} \geq 1.$$

This will happen for any tour: eventually, we must go from a city in S to a city not in S.

Subtour elimination constraints

Problem: the local constraints allow for subtours that don't visit all n cities.

Solution #1 (Dantzig, Fulkerson, Johnson, 1954): for every set S of cities, add a constraint saying that the tour leaves S at least once.

For every $S \subseteq \{1, 2, \ldots, n\}$ with $1 \leq |S| \leq n - 1$:

$$\sum_{i \in S} \sum_{j \notin S} x_{ij} \geq 1.$$

This will happen for any tour: eventually, we must go from a city in S to a city not in S.

In a solution to the local constraints with subtours, this is violated if we take S to be the set of cities in a subtour.

Huge formulations

Good news: the local constraints, together with the subtour elimination constraints, describe TSP.

Huge formulations

Good news: the local constraints, together with the subtour elimination constraints, describe TSP.

Bad news: for n cities, there are $2^n - 2$ subtour elimination constraints! ($2^{n-1} - 1$ if we assume $1 \in S$.)

Huge formulations

Good news: the local constraints, together with the subtour elimination constraints, describe TSP.

Bad news: for n cities, there are $2^n - 2$ subtour elimination constraints! ($2^{n-1} - 1$ if we assume $1 \in S$.)

Slightly encouraging news: given a solution to the local constraints with subtours, we can quickly find a subtour elimination constraint it violates.

For example, let S be the set of all cities visited by the tour, starting at city 1.

Huge formulations

Good news: the local constraints, together with the subtour elimination constraints, describe TSP.

Bad news: for n cities, there are $2^n - 2$ subtour elimination constraints! ($2^{n-1} - 1$ if we assume $1 \in S$.)

Slightly encouraging news: given a solution to the local constraints with subtours, we can quickly find a subtour elimination constraint it violates.

For example, let S be the set of all cities visited by the tour, starting at city 1.

- If $S = \{1, 2, \ldots, n\}$, we actually do have a tour.
- Otherwise, the constraint saying we must leave S at least once is violated.

Branch-and-cut for TSP

We can use this idea to get a branch-and-cut algorithm for solving TSP problems.

Begin by just solving the LP relaxation of the local constraints.

Branch-and-cut for TSP

We can use this idea to get a branch-and-cut algorithm for solving TSP problems.

Begin by just solving the LP relaxation of the local constraints.

Whenever we get an LP solution that has a lower cost than the best tour found so far:

Branch-and-cut for TSP

We can use this idea to get a branch-and-cut algorithm for solving TSP problems.

Begin by just solving the LP relaxation of the local constraints.

Whenever we get an LP solution that has a lower cost than the best tour found so far:

1. If it's an integer solution representing a subtour, add the subtour elimination constraint it violates.

Branch-and-cut for TSP

We can use this idea to get a branch-and-cut algorithm for solving TSP problems.

Begin by just solving the LP relaxation of the local constraints.

Whenever we get an LP solution that has a lower cost than the best tour found so far:

1. If it's an integer solution representing a subtour, add the subtour elimination constraint it violates.
2. If there is some $x_{ij} \notin \mathbb{Z}$, branch on $x_{ij} = 0$ and $x_{ij} = 1$.

Branch-and-cut for TSP

We can use this idea to get a branch-and-cut algorithm for solving TSP problems.

Begin by just solving the LP relaxation of the local constraints.

Whenever we get an LP solution that has a lower cost than the best tour found so far:

1. If it's an integer solution representing a subtour, add the subtour elimination constraint it violates.
2. If there is some $x_{ij} \notin \mathbb{Z}$, branch on $x_{ij} = 0$ and $x_{ij} = 1$.

   (State-of-the-art algorithms sometimes improve on this option, but it's complicated.)

Branch-and-cut for TSP

We can use this idea to get a branch-and-cut algorithm for solving TSP problems.

Begin by just solving the LP relaxation of the local constraints.

Whenever we get an LP solution that has a lower cost than the best tour found so far:

1. If it's an integer solution representing a subtour, add the subtour elimination constraint it violates.
2. If there is some $x_{ij} \notin \mathbb{Z}$, branch on $x_{ij} = 0$ and $x_{ij} = 1$.

   (State-of-the-art algorithms sometimes improve on this option, but it's complicated.)
3. If it's an integer solution representing a tour, update our best solution found!

Timing constraints

Problem: the local constraints allow for subtours that don't visit all n cities.

Timing constraints

Problem: the local constraints allow for subtours that don't visit all n cities.

Solution #2 (Miller, Tucker, Zemlin, 1960): Add variables representing the times at which a city is visited.

Timing constraints

Problem: the local constraints allow for subtours that don't visit all n cities.

Solution #2 (Miller, Tucker, Zemlin, 1960): Add variables representing the times at which a city is visited.

For $i = 2, \ldots, n$, let t_i denote the time at which we visit city i, with $1 \leq t_i \leq n - 1$. We leave t_1 undefined.

Timing constraints

Problem: the local constraints allow for subtours that don't visit all n cities.

Solution #2 (Miller, Tucker, Zemlin, 1960): Add variables representing the times at which a city is visited.

For $i = 2, \ldots, n$, let t_i denote the time at which we visit city i, with $1 \leq t_i \leq n - 1$. We leave t_1 undefined.

We want an inequality to encode the logical implication

$$\text{if } x_{ij} = 1, \text{ then } t_j \geq t_i + 1$$

for every pair of cities $i, j \neq 1$.

Why do these work?

How do we know that the timing constraints get rid of subtours?

Why do these work?

How do we know that the timing constraints get rid of subtours?

1. **For any tour, we can satisfy the timing constraints.**

   If we visit cities $i_1, i_2, \ldots, i_{n-1}$ in that order from city 1, set $t_{i_1} = 1, t_{i_2} = 2, \ldots, t_{i_{n-1}} = n - 1$.

Why do these work?

How do we know that the timing constraints get rid of subtours?

1. **For any tour, we can satisfy the timing constraints.**

   If we visit cities $i_1, i_2, \ldots, i_{n-1}$ in that order from city 1, set $t_{i_1} = 1, t_{i_2} = 2, \ldots, t_{i_{n-1}} = n - 1$.

2. **If there is a subtour, then we can't satisfy the timing constraints.**

   Suppose $x_{ab} = x_{bc} = x_{ca} = 1$ and none of a, b, c are 1.

Why do these work?

How do we know that the timing constraints get rid of subtours?

1. **For any tour, we can satisfy the timing constraints.**

   If we visit cities $i_1, i_2, \ldots, i_{n-1}$ in that order from city 1, set $t_{i_1} = 1, t_{i_2} = 2, \ldots, t_{i_{n-1}} = n - 1$.

2. **If there is a subtour, then we can't satisfy the timing constraints.**

   Suppose $x_{ab} = x_{bc} = x_{ca} = 1$ and none of a, b, c are 1.

   Then we can't satisfy the three constraints

$$t_b \geq t_a + 1$$
$$t_c \geq t_b + 1$$
$$t_a \geq t_c + 1$$

Going from if-statements to inequalities

The statement we want: **if** $x_{ij} = 1$**, then** $t_j \geq t_i + 1$.

Going from if-statements to inequalities

The statement we want: **if** $x_{ij} = 1$**, then** $t_j \geq t_i + 1$.

With the big number method:

$$t_j \geq t_i + 1 - M(1 - x_{ij})$$

for some large M.

Going from if-statements to inequalities

The statement we want: **if** $x_{ij} = 1$**, then** $t_j \geq t_i + 1$.

With the big number method:

$$t_j \geq t_i + 1 - M(1 - x_{ij})$$

for some large M.

- When $x_{ij} = 1$, this simplifies to $t_j \geq t_i + 1$.

Going from if-statements to inequalities

The statement we want: **if** $x_{ij} = 1$**, then** $t_j \geq t_i + 1$.

With the big number method:

$$t_j \geq t_i + 1 - M(1 - x_{ij})$$

for some large M.

- When $x_{ij} = 1$, this simplifies to $t_j \geq t_i + 1$.
- When $x_{ij} = 0$, we get $t_j \geq t_i + 1 - M$, which doesn't restrict t_i, t_j.

Going from if-statements to inequalities

The statement we want: **if** $x_{ij} = 1$**, then** $t_j \geq t_i + 1$.

With the big number method:

$$t_j \geq t_i + 1 - M(1 - x_{ij})$$

for some large M.

- When $x_{ij} = 1$, this simplifies to $t_j \geq t_i + 1$.
- When $x_{ij} = 0$, we get $t_j \geq t_i + 1 - M$, which doesn't restrict t_i, t_j.

We can check: if we take $M = n$, then any actual tour can satisfy these constraints. The times $t_2, \ldots, t_n$ can be chosen between 1 and $n - 1$, so $t_j \geq t_i + 1 - n$ always holds.

Comparing the methods

At first glance:

- DFJ's formulation has $2^{n-1} - 1$ extra constraints, plus the $2n$ local constraints.
- MTZ's formulation has only n^2 extra constraints. There are $n - 1$ extra variables, which can be integer variables, but don't need to be.

Comparing the methods

At first glance:

- DFJ's formulation has $2^{n-1} - 1$ extra constraints, plus the $2n$ local constraints.
- MTZ's formulation has only n^2 extra constraints. There are $n - 1$ extra variables, which can be integer variables, but don't need to be.

In practice:

- DFJ's formulation has an efficient branch-and-cut approach.
- MTZ's formulation is weaker: the feasible region has the same integer points, but includes more fractional points.