A constant-factor approximation algorithm for the asymmetric travelling salesman problem

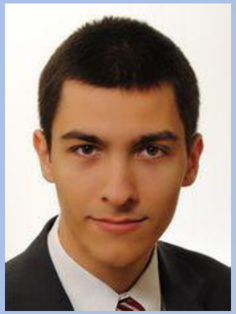
László Végh

London School of Economics

Joint work with

Ola Svensson and Jakub Tarnawski

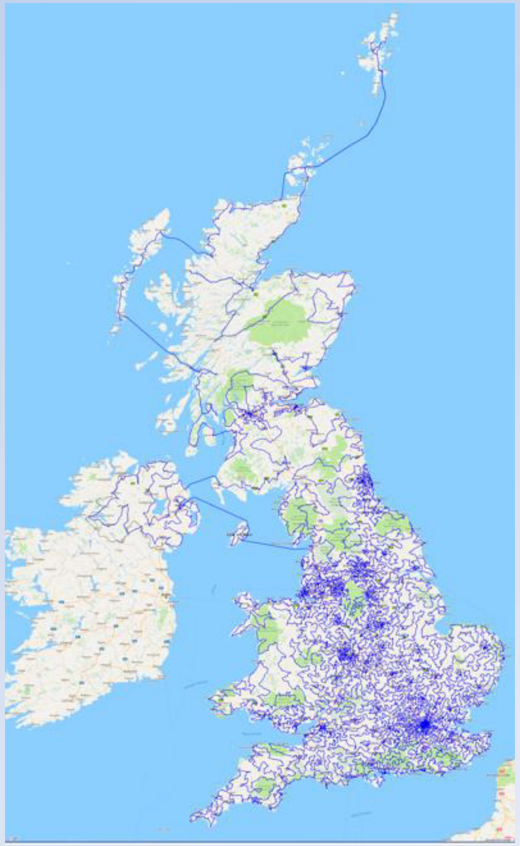
´ cole Polytechnique F́ d́ rale de Lausanne



Travelling salesman problem

Given *n* cities and their pairwise distances,
find a shortest tour visiting all *n* cities.

- One of the best known NP-hard optimization problems
- Studied since the 19th century
- Symmetric TSP: $d(u,v) = d(v,u) \quad \forall u,v$
- Asymmetric TSP: $d(u,v) \neq d(v,u)$ is possible



UK pub tour
[Cook et al., 2015]

Triangle inequality:
$d(u,v) \leq d(u,w) + d(w,v) \quad \forall u,v,w$

Symmetric vs Asymmetric TSP

Symmetric TSP

- 1.5-approximation algorithm [Christofides '76]
- Graphic TSP: unweighted graph shortest path metric
  - Current best 1.4 [Sebő & Vygen '14], following
    [Oveis Gharan, Saberi & Singh '11]
    [Mömke & Svensson '11]
    [Mucha '12]

Symmetric vs Asymmetric TSP

Asymmetric TSP

- $\log_2 n$-approximation algorithm [Frieze, Galbiati & Maffioli ’82]
- $0.99 \log_2 n$ [Bläser ’03]
- $0.84 \log_2 n$ [Kaplan, Lewenstein, Shafrir & Sviridenko ’03]
- $0.67 \log_2 n$ [Feige & Singh ’07]

- $O\left(\frac{\log n}{\log \log n}\right)$ [Asadpour, Goemans, Mądry, Oveis Gharan & Saberi ’10] via thin trees.

Asymmetric TSP – recent developements

- $O(\mathrm{poly}\log\log n)$ bound on integrality gap of LP [Anari & Oveis Gharan '15]

Constant-factor approximations:

- Bounded genus graphs [Oveis Gharan & Saberi '11]
- Node-weighted graphs [Svensson '15]
- Graphs with 2 edge weights [Svensson, Tarnawski & V. '16]

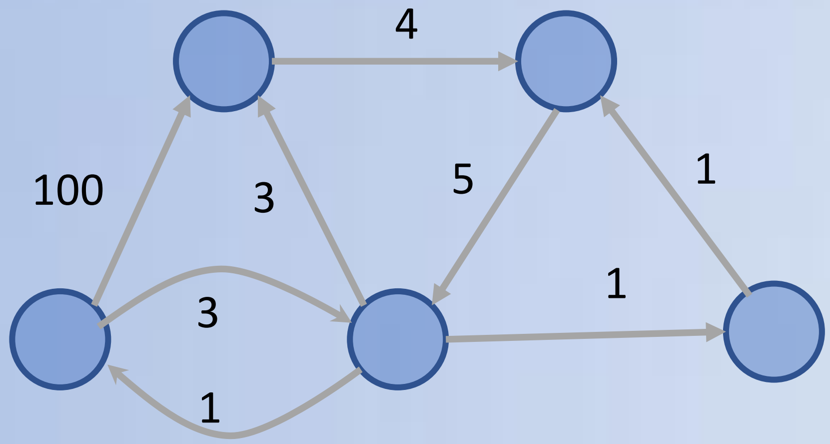
Our result: constant-factor approximation for general ATSP with respect to the Held-Karp relaxation.

ATSP – Graphic formulation

Input: directed graph $G = (V, A)$, edge weights $w : A \to \mathbb{R}_+$
Find a minimum weight **tour** F.

- Tour = closed walk visiting every vertex at least once =
  = Eulerian and connected edge multiset
- Eulerian: $\delta^{in}_F(v) = \delta^{out}_F(v) \quad \forall v \in V$
- Subtour = closed walk (not necessarily connected)

In-degree & out-degree in F

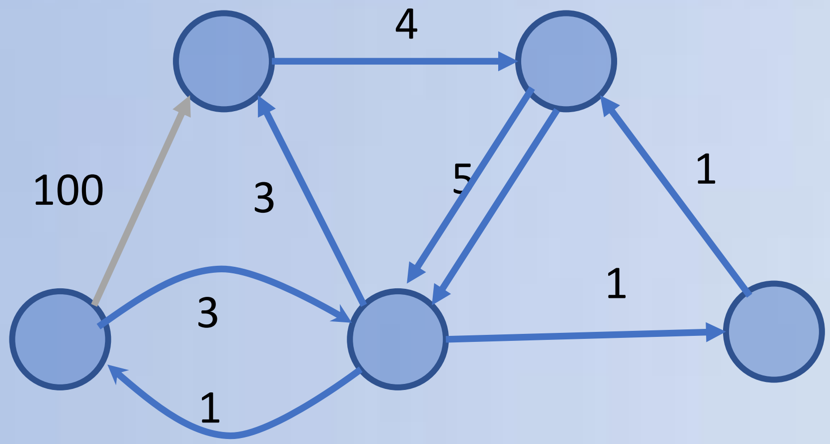


ATSP – Graphic formulation

Input: directed graph $G = (V, A)$, edge weights $w: A \to \mathbb{R}_+$
Find a minimum weight **tour** F.

- Tour = closed walk visiting every vertex at least once =
  = Eulerian and connected edge multiset
- Eulerian: $|\delta^{in}_F(v)| = |\delta^{out}_F(v)| \quad \forall v \in V$
- Subtour = closed walk (not necessarily connected)

In-degree & out-degree in F



Held-Karp relaxation

- Input: $G = (V, A)$, edge weights $c: A \to \mathbb{R}_+$.
- Variables $x: E \to \mathbb{R}_+$: multiplicity of selecting edge a.

minimize $\sum_a c_a x_a$

subject to $x(\delta^{out}(v)) = x(\delta^{in}(v)) \quad \forall v \in N$

$x(\delta(S)) \geq 2 \quad \forall S \subset V, S \neq \emptyset$

$x \geq 0$

Undirected degree:
$\delta(S) = \delta^{out}(S) + \delta^{in}(S)$

Eulerian degree constraints

Subtour elimination constraints

Held-Karp relaxation

- Input: ‶Ο = (,ʽΟ), edge weights ύ:ʽΟ → +.
- Variables ὼ :E → + : multiplicity of selecting edge ʽΏ.

minimize ύ ὼ

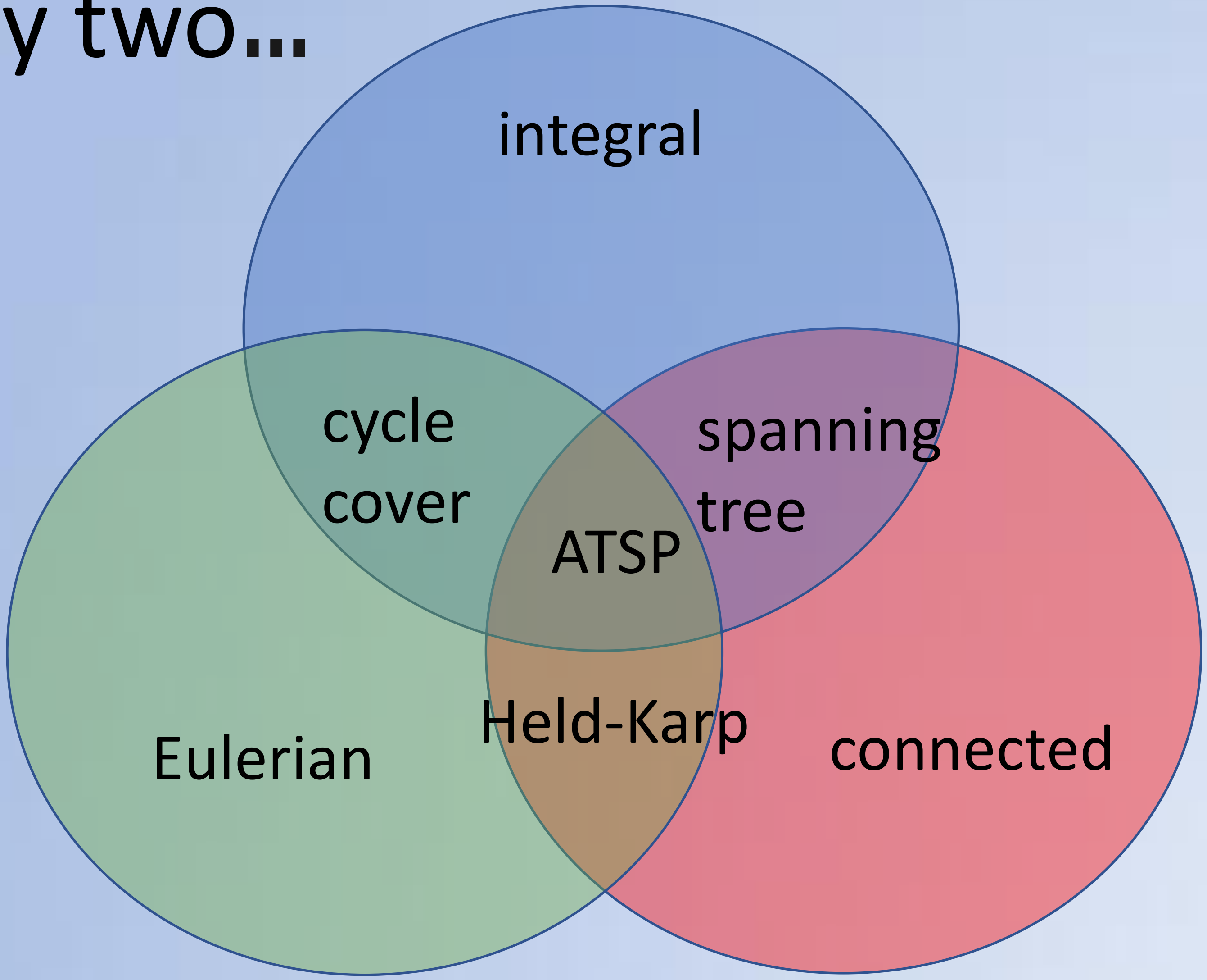
subject to ὼ(ᛚ ࢫ(ὺ)) = ὼ(ᛚ ࢶࢼࢽ(ὺ)) ὺ ᴺ

ὼ(ᛚ (‶Υ)) ς ‶Υ ,‶Υ ≠

ὼ π

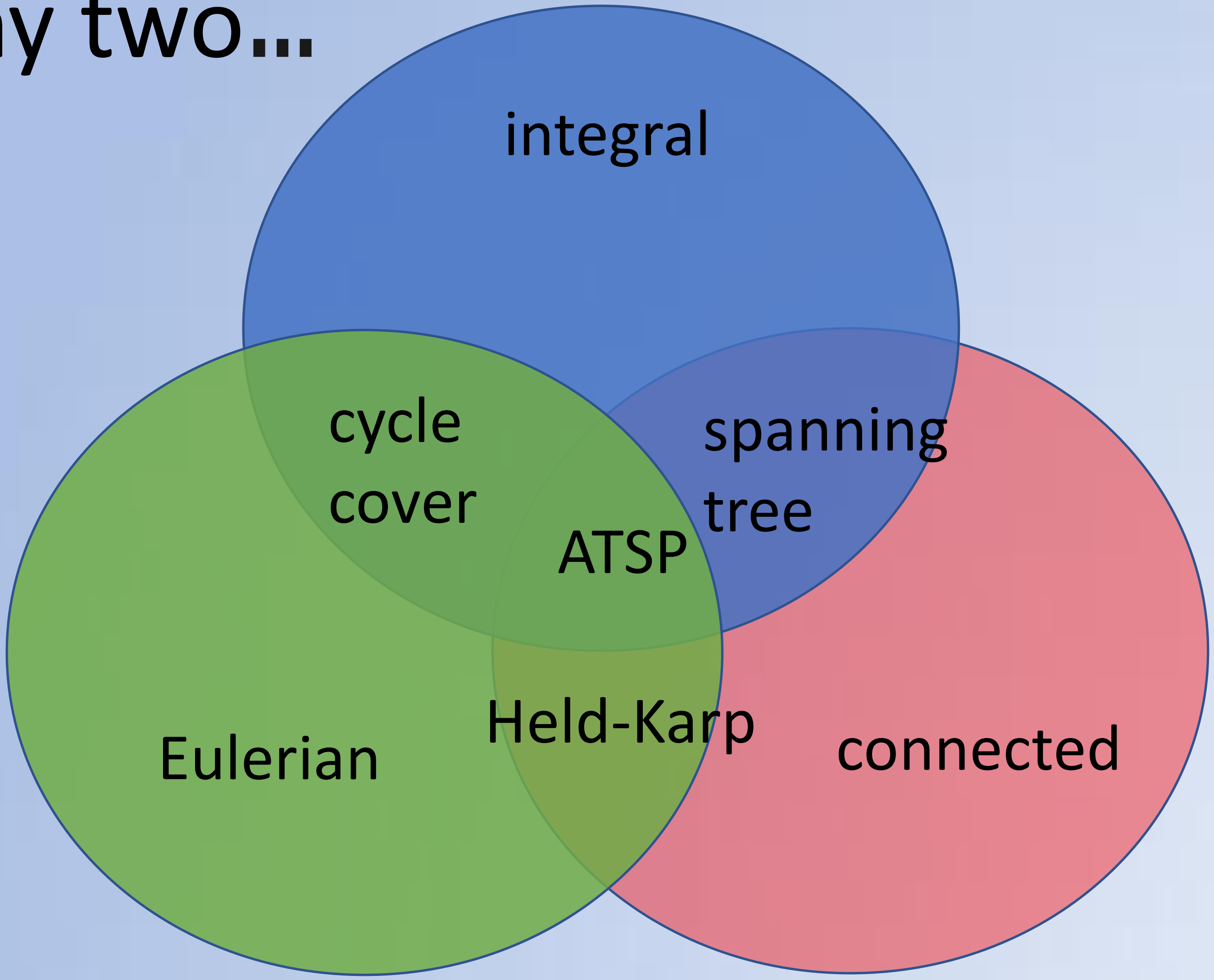
- Can be solved in polynomial time
- Integrality gap ς [Charikar, Goemans & Karloff ’06]

Pick any two…



integral
cycle cover
spanning tree
ATSP
Held-Karp
Eulerian
connected

Pick any two…


integral
cycle cover
spanning tree
ATSP
Held-Karp
Eulerian
connected

Repeated cycle cover algorithm
[Frieze, Galbiati & Maffioli ’82]

Relaxing connectivity:

1. Find minimum weight cycle cover
2. Contract and repeat

- Each cycle cover has cost ŰỦ″Y
- Overall $\log_2 n$ rounds
- $\log_2 n$ approximation

Node-weighted case [Svensson’15]

Directed graph ‶0 = (,Ὸ), node weights : → +
ύ(ό,ὺ) = (ό) + (ὺ) ό,ὺ ᴺ Ὸ

Local-Connectivity ATSP: relaxing connectivity constraints to ἀlocalἐ

-light algorithm for Local-Connectivity ATSP

(ω + -) -approximation for ATSP

Theorem [Svensson’15]
There exists a polytime ςχ + -)-approximation for node-weighted ATSP.

Roadmap

General ATSP

LP duality + uncrossing

Laminarly weighted ATSP

Graph theory: contractions

Irreducible instances

Node weighted algorithm + contractions

Vertebrate pairs

[Svensson ’15]

Local-connectivity ATSP

O(1)-light lcATSP algorithm in vertebrate pairs

Roadmap

General ATSP

LP duality + uncrossing

Laminarly weighted ATSP

Irreducible instances

Graph theory: contractions

Node weighted algorithm + contractions

Vertebrate pairs

[Svensson ’15]

Local-connectivity ATSP

O(1)-light lcATSP algorithm in vertebrate pairs

Dual of the Held-Karp relaxation

minimize Ú ẁ

subject to

ẁ(⁅ ݿ(Ù)) = ẁ(⁅ ݗ(Ù)) Ù ᴺ

ẁ(⁅ (‴Ɣ)) ς ≠ ‴Ɣ

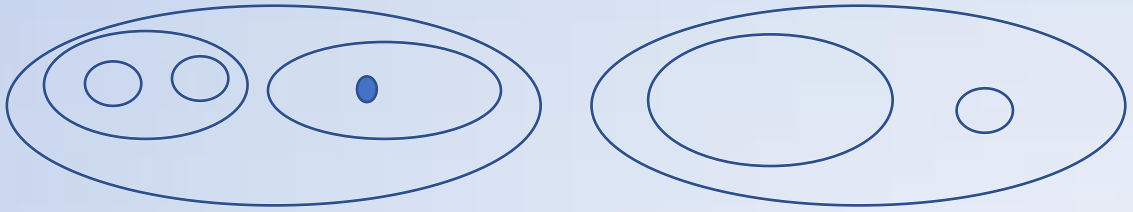
ẁ π

maximize ς ≠ب ẃب

subject to

ب:(ذ,ذ)ᴺ ب ẃب + ذ − ذ Ú(ó, Ù) ó, Ù ᴺ Ὸ

ẃ π



- Dual can be solved in polynomial time.
- One can efficiently find an optimal (, ẃ) such that the support of ẃ is a laminar family of sets.
  Efficient uncrossing [Karzanov’96]

Laminarly weighted ATSP: = ‶Ο, , ὼ, ώ

minimize Ú ὼ

subject to

ὼ(ἰ ᡃ(Ù)) = ὼ(ἰ ݗ(Ù)) Ù ᴺ

ὼ(ἰ (‶Υ)) ς ≠ ‶Υ

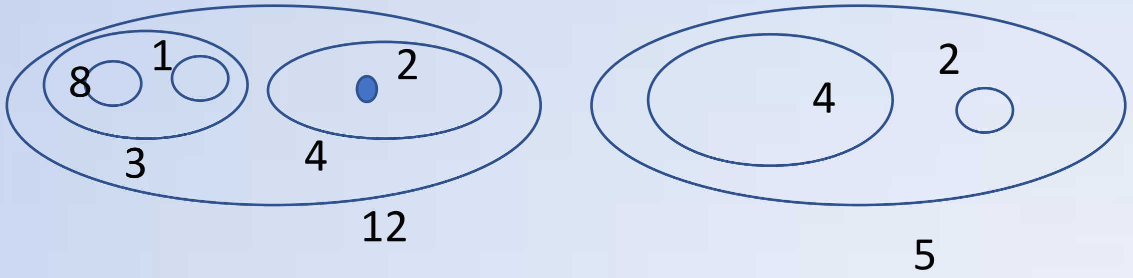
ὼ π

maximize ς ≠ب ώب

subject to

ب:(ذ,ذ)ᴺ ب ώب + ذ − ذ Ú(ό, Ù) ό, Ù ᴺ ‶Ο

ώ π



- ‶Ο: directed graph
- : laminar family of sets
- ὼ: feasible Held-Karp solution tight on every set in : ὼ(ἰ (‶Υ)) = ς ‶Υ ᴺ
- ώ: → +

Laminarly weighted ATSP: = Ὀ, ,ẁ,ẃ

minimize ύ ẁ

subject to

ẁ(ἰ ᵓ(ὺ)) = ẁ(ἰ ἐ(ὺ)) ὺ ᴺ

ẁ(ἰ (Ὑ)) ς ≠ Ὑ

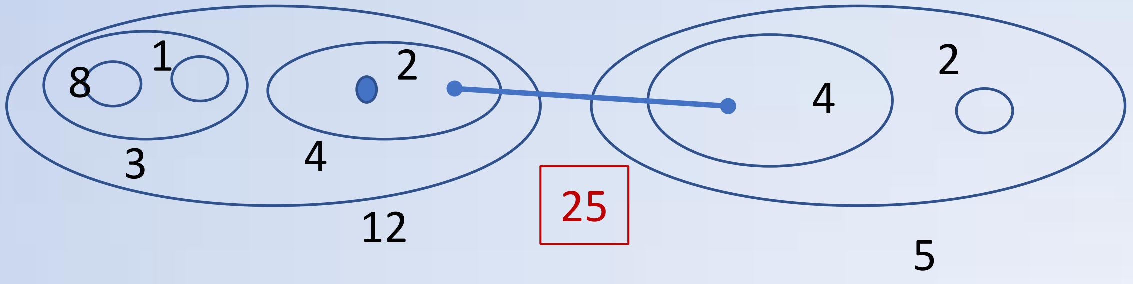
ẁ π

maximize ς ≠ب ẃب

subject to

ب:(ذ,ذ)ᴺ ب ẃب + ذ − ذ ύ(ό,ὺ) ό,ὺ ᴺ Ὀ

ẃ π



- Ὀ: directed graph
- : laminar family of sets
- ẁ: feasible Held-Karp solution tight on every set in : ẁ(ἰ (Ὑ)) = ς Ὑ ᴺ
- ẃ: → +

Induced weight function: ύ(ό,ὺ) = ب:(ذ,ذ)ᴺ ب ẃب

Laminarly weighted ATSP: = ῊΟ, , ẁ, ώ

minimize ύ ẁ

subject to

ẁ(ʅ (ẁ)) = ẁ(ʅ (ẁ)) ẁ ᴺ

ẁ(ʅ (ῩΥ)) ς ≠ ῩΥ

ẁ π

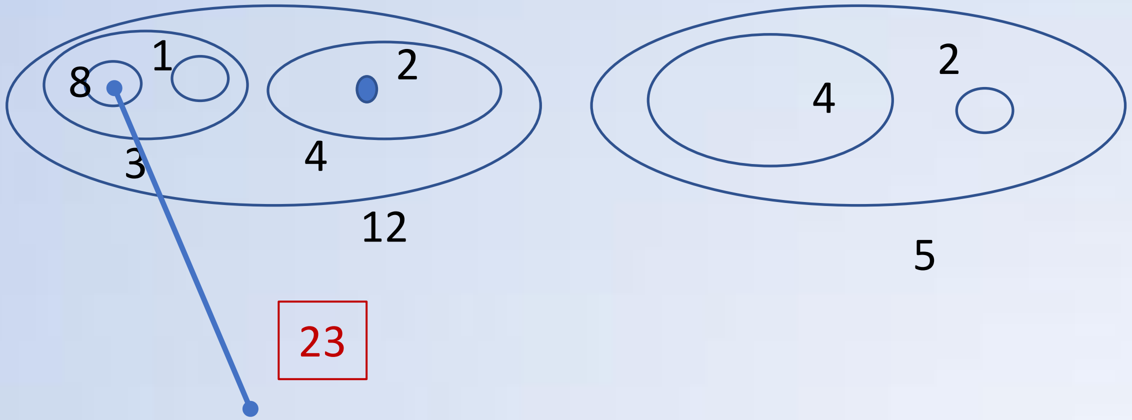
maximize ς ≠ ώ

subject to

ώ + – ύ(ό, ẁ) ό, ẁ ᴺ Ὸ

ώ π

- ῊΟ: directed graph
- : laminar family of sets
- ẁ: feasible Held-Karp solution
  tight on every set in : ẁ(ʅ (ῩΥ)) = ς ῩΥ ᴺ
- ώ: → +



Induced weight function: ύ(ό, ẁ) = ώ

Reduction to laminarly weighted ATSP

- Start with any ʽO and ΰ.
- Compute Held-Karp optimal solution ὼ and dual ώ supported on laminar family
- Delete all edges with ὼ = π.

maximize ς ≠ٻ ώٻ

subject to

ٻ:(ذ,ذ) ᴺ ٻ ώٻ + ذ − ذ ΰ(ό,ὺ) ό,ὺ ᴺ ʽO

ώ π

Observations:

- Optimal solutions and optimum value are the same for ΰ and for

  ΰʹ(ό,ὺ) = ΰ(ό,ὺ) + ذ − ذ

- All remaining edges have

  ΰʹ(ό,ὺ) = ώٻ

  ٻ:(ذ,ذ) ᴺ ٻ

Roadmap

General ATSP

LP duality + uncrossing

Laminarly weighted ATSP

Graph theory: contractions

Irreducible instances

Node weighted algorithm + contractions

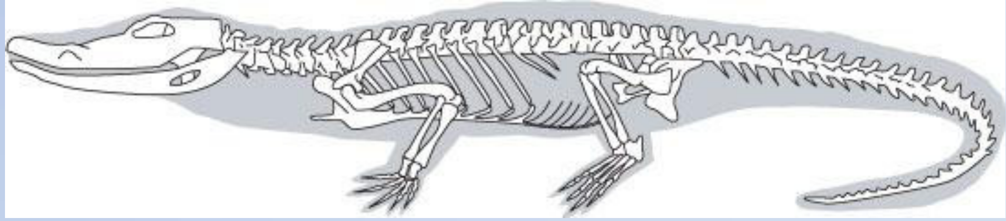
Vertebrate pairs

[Svensson ’15]

Local-connectivity ATSP

O(1)-light lcATSP algorithm in vertebrate pairs

Vertebrate pairs

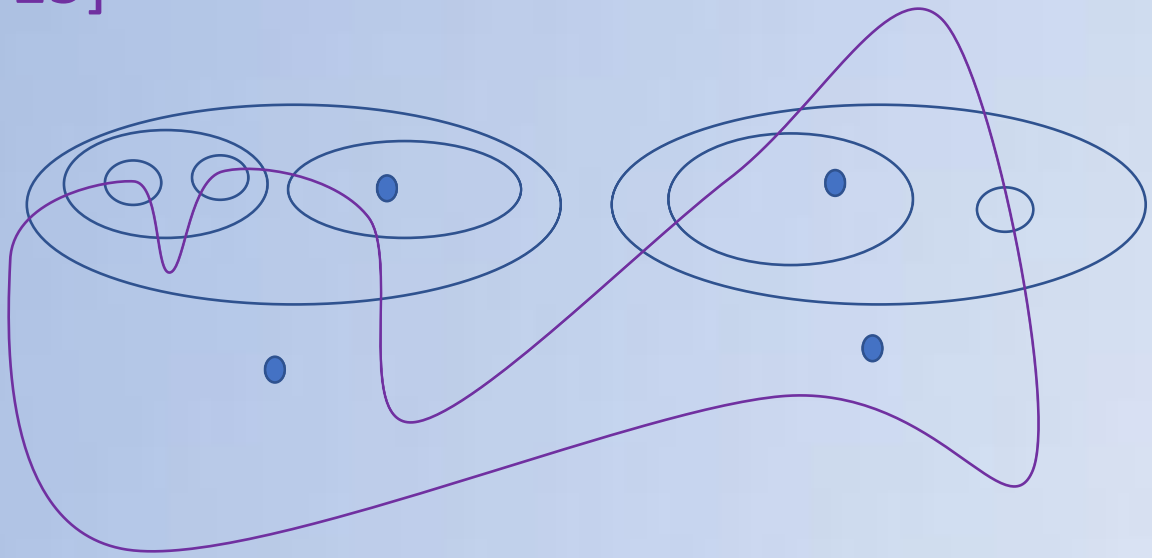


Vertebrate pair (, ő)

- = (‟0, , ὼ, ώ) instance
- ő: backbone = subtour that crosses every nonsingleton set in

Vertebrate pairs

- We will reduce general ATSP to solving ATSP for a vertebrate pair (, ő) with ú(ő) = Θ(ŭŭ̃ʺY) (more or less…)
- Solve Local-Connectivity ATSP on such instances, and apply [Svensson'15]

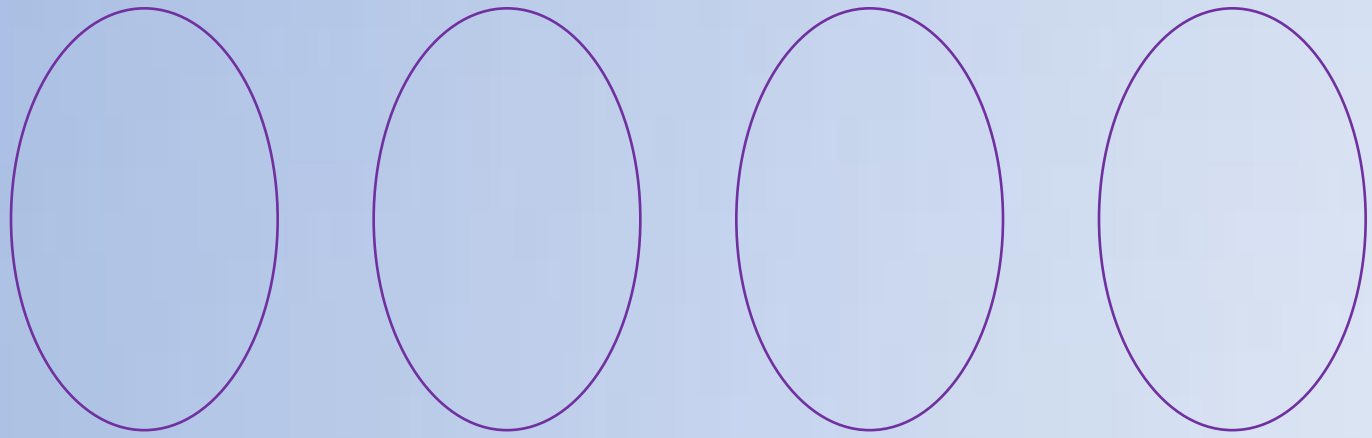


Local-Connectivity ATSP [Svensson'15]

Instance = (‟O, ,ὼ, ώ) with induced weights ύ:Ὸ → +

Lower bound function **lb:** → + with ذN lb ὺ = ΰὓ‶Y

Input: partition of the vertex set = ۳ ۴



۳ ۴ ۵ ۶

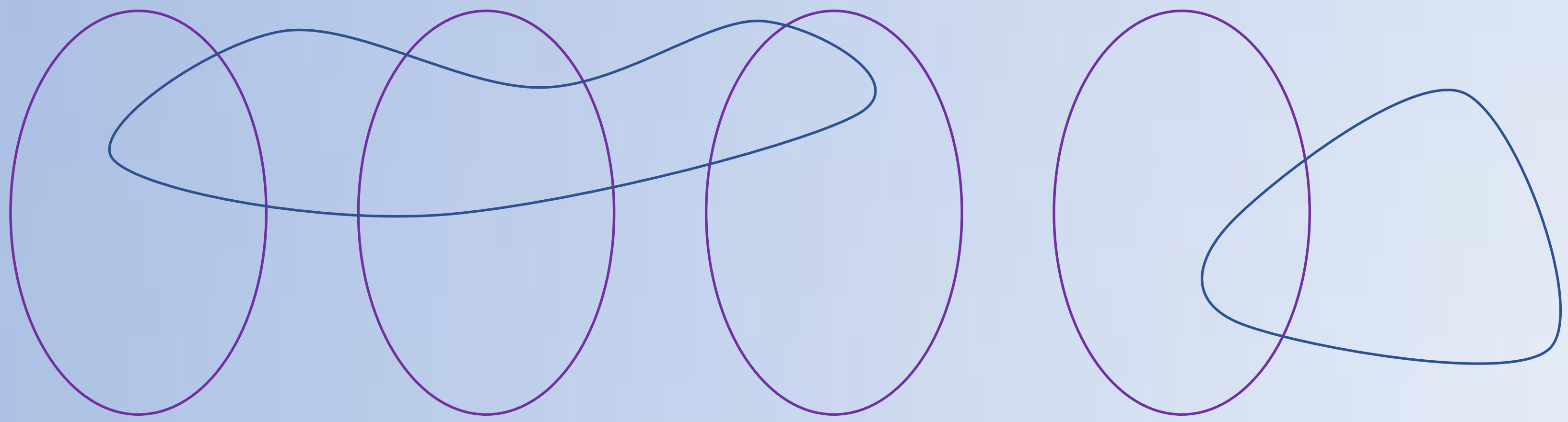
Local-Connectivity ATSP [Svensson'15]

Instance = (ʺ0, , ὼ, ώ) with induced weights **w:** Ὸ → +

Lower bound function **lb:** → + with ذ N **lb** ὺ = ΰὒʺϓ

Input: partition of the vertex set = ٣ ٤

Output: Eulerian edge set ʺ0 with |( ), ʺ0| > π for each



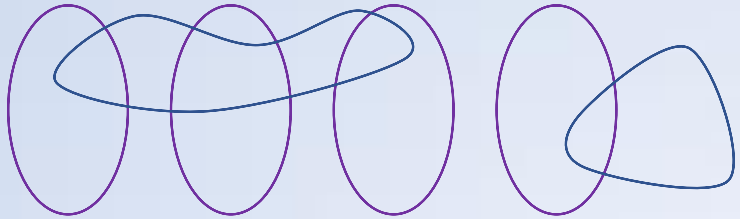
Local-Connectivity ATSP [Svensson’15]

Instance = (῎0, , ὼ, ώ) with induced weights w: Ό → +

Lower bound function lb: → + with ذN lb ὸ = ΰὒ῎Υ

Input: partition of the vertex set = ٣ ˂ ٤ ˂ ˂

Output: Eulerian ῎0 with |‖ ( ) ͵ ῎0| > π for each



-light algorithm: for every component ὃ of ῎0,

$$\frac{ύ(Ό\ ὃ\ )}{\text{lb}\ \ (ὃ)}$$

ά*Every component pays for itself locally*έ

Local-Connectivity ATSP [Svensson'15]

-light algorithm for Local-Connectivity ATSP

(ω + -) -approximation for ATSP

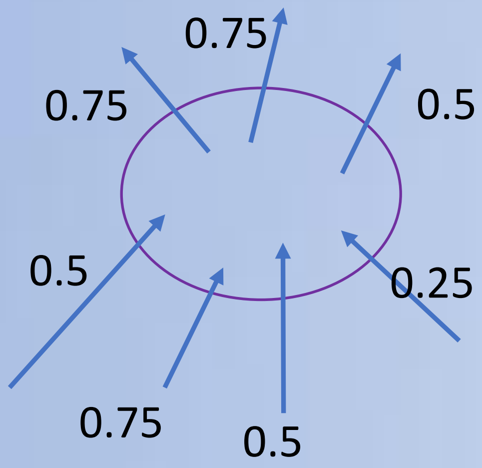
Theorem [Svensson'15]
There exists a polytime ςχ + -)-approximation for node-weighted ATSP.

Local-Connectivity ATSP: node-weighted case

- Instance = (̏0, , ὠ, ώ), with containing only singletons (ignore ő)

  ύ(ό, ὺ) = ώ{ذ} + ώ{ذ}

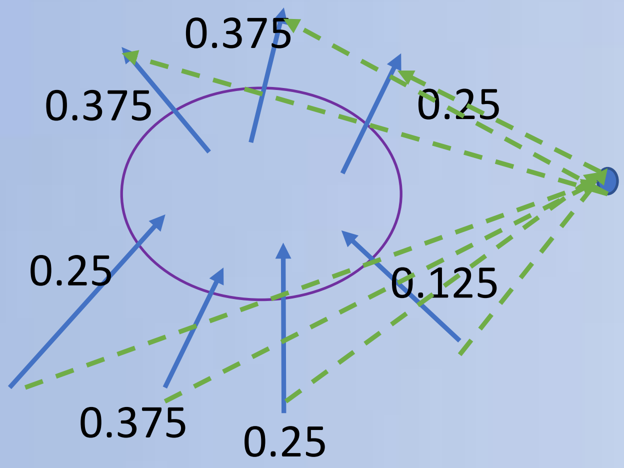
- Define lb(ό) = ςώ{ذ} ό ᴺ
- Partition = ٣ ` ٤ ` ` all strongly connected
- Modify ̏0 and ὠ, and solve an integer circulation problem



Local-Connectivity ATSP: node-weighted case

- Instance = (̎0, ,ὼ,ώ), with containing only singletons (ignore ő)

  ύ(ό,ὺ) = ώ{ذ} + ώ{ذ}
- Define lb(ό) = ςώ{ذ} ό ᴺ
- Partition = ٣ ˂ ٤ ˂ ˂ all strongly connected
- Modify ̎0 and ὼ, and solve an integer circulation problem

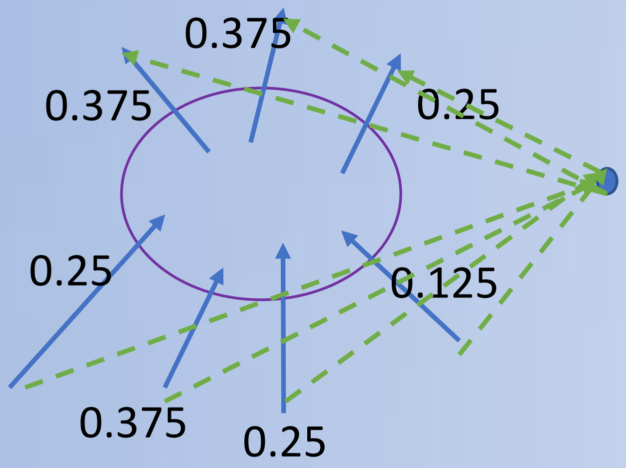


- For each , create auxiliary vertex
- Reroute 1 fractional unit of incoming and outgoing flow ὼ to
- Solve integer circulation problem routing =1 unit through each
- Map back to original ̎0

Local-Connectivity ATSP: node-weighted case

- The rerouted ẁ is feasible to the circulation problem of weight ŬǓ̃″Ƴ
- Flow integrality: there exists integer solution of weight ŬǓ̃″Ƴ
- After mapping back, every vertex with ώẍ > π has in-degree ς
- For a component ŏ, ύ(Ὀ(ŏ)) = ذ,ذ N Ī ώ{ذ} + ώ{ذ} τ ذ N ώ{ذ}
- lb((ŏ)) = ς ذ N ώ{ذ}

2-light algorithm



Local-Connectivity ATSP:
one nonsingular set in

- Vertebrate pair (, ő). Assume has a single non-singleton component ‶Ỵ. Thus,

ύ(ό, ὺ) = ώ{ذ} + ώ{ذ} + ώب ‶Ω(ό, ὺ) ᴺ⸢ ‶Ỵ
ώ{ذ} + ώ{ذ} ‶Ω(ό, ὺ) ᵉ⸢ ‶Ỵ

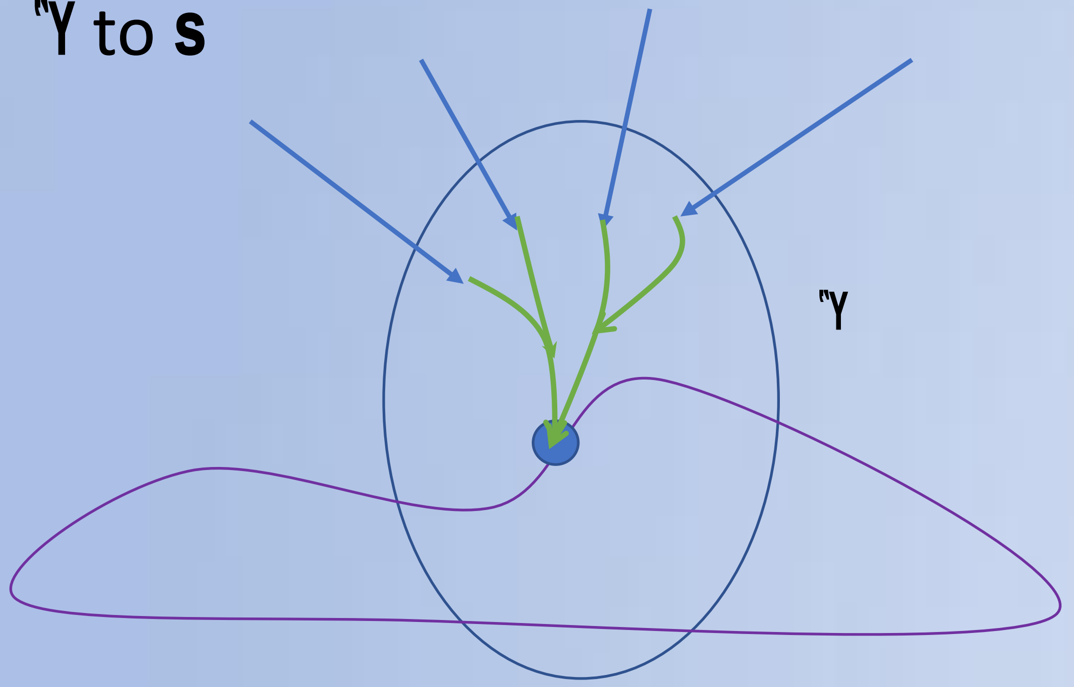
- Define

lb ό = ςώ{ذ} ‶Ωό ᴺ ő
ύ(ő) / |(ő)| ‶Ωό ᴺ ő

- ذᴺ lb ὺ = ΰ(ΰΰ̃‶Ỵ), since ύ(ő) = Θ ΰΰ̃‶Ỵ)

Local-Connectivity ATSP: one nonsingular set in

- By assumption, ẇ꜕ ˀ(‟Y) = ẇ꜕ ẗذد(‟Y) =1
- Backbone property: there is a node ᴺ (ő)ˏ ‟Y
- Simple flow argument: we can route the incoming 1 unit of flow to ‟Y to **s**



Local-Connectivity ATSP: one nonsingular set in

- Partition = ٣˂ ٤˂ ˂
- Add backbone ő into Eulerian set ̎0.
- Via flow splitting, ἀforceἐ all edges entering Ὗ to proceed to ɴ ő
- Create auxiliary vertices as before
- Solve integral circulation problem, and add solution to ̎0.

Local-Connectivity ATSP: one nonsingular set in

Analysis

- For all components ő not crossing *S*, ú(ʽO(ő))/lb((ő)) ς exactly as in the node-weighted case
- Giant component őᵣ containing ő.
    - Contains all edges crossing ʺϒ
    - Has lower bound lb((őᵣ)) lb ő = Θ űǚʺϒ)
    - ú(ʽO(őᵣ)) ú(ʺO) ű űǚʺϒ)
- Therefore solution is ǘ ρ -light.
- Same approach extends to arbitrary : enforce that every subtour crossing a set in must intersect the backbone.

Roadmap

General ATSP

LP duality + uncrossing

Laminarly weighted ATSP

Graph theory: contractions

Irreducible instances

Node weighted algorithm + contractions

Vertebrate pairs

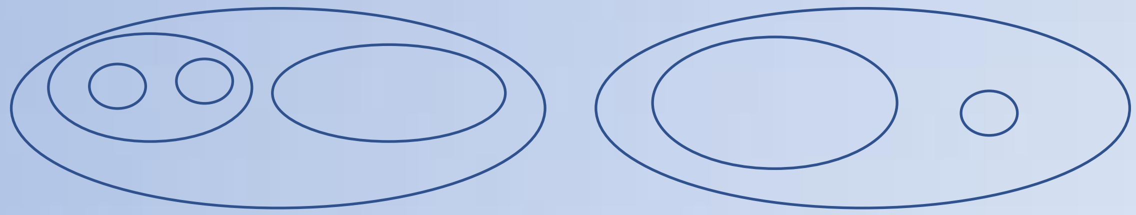
[Svensson’15]

Local-connectivity ATSP

O(1)-light lcATSP algorithm in vertebrate pairs

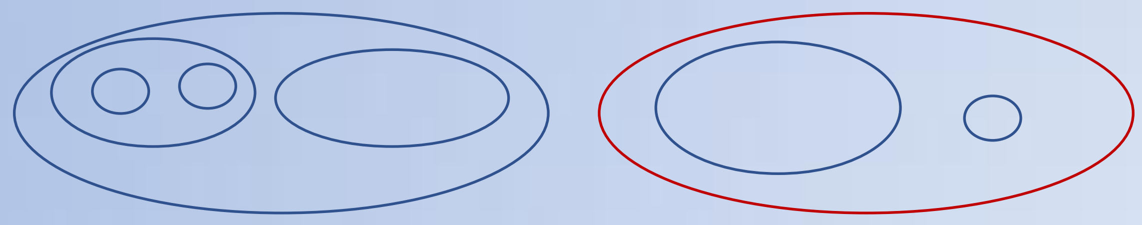
Motivation: reducing by contraction

- All sets in the family are singletons: node-weighted ATSP
- Would like to reduce the problem by contracting nonsingleton sets in



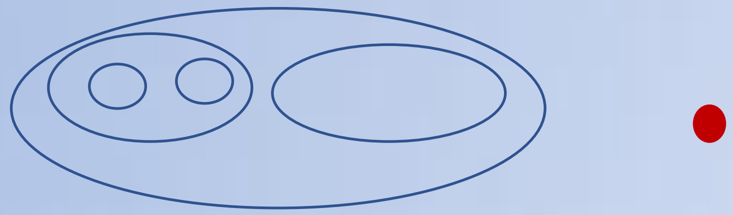
Motivation: reducing by contraction

- All sets in the family are singletons: node-weighted ATSP
- Would like to reduce the problem by contracting nonsingleton sets in



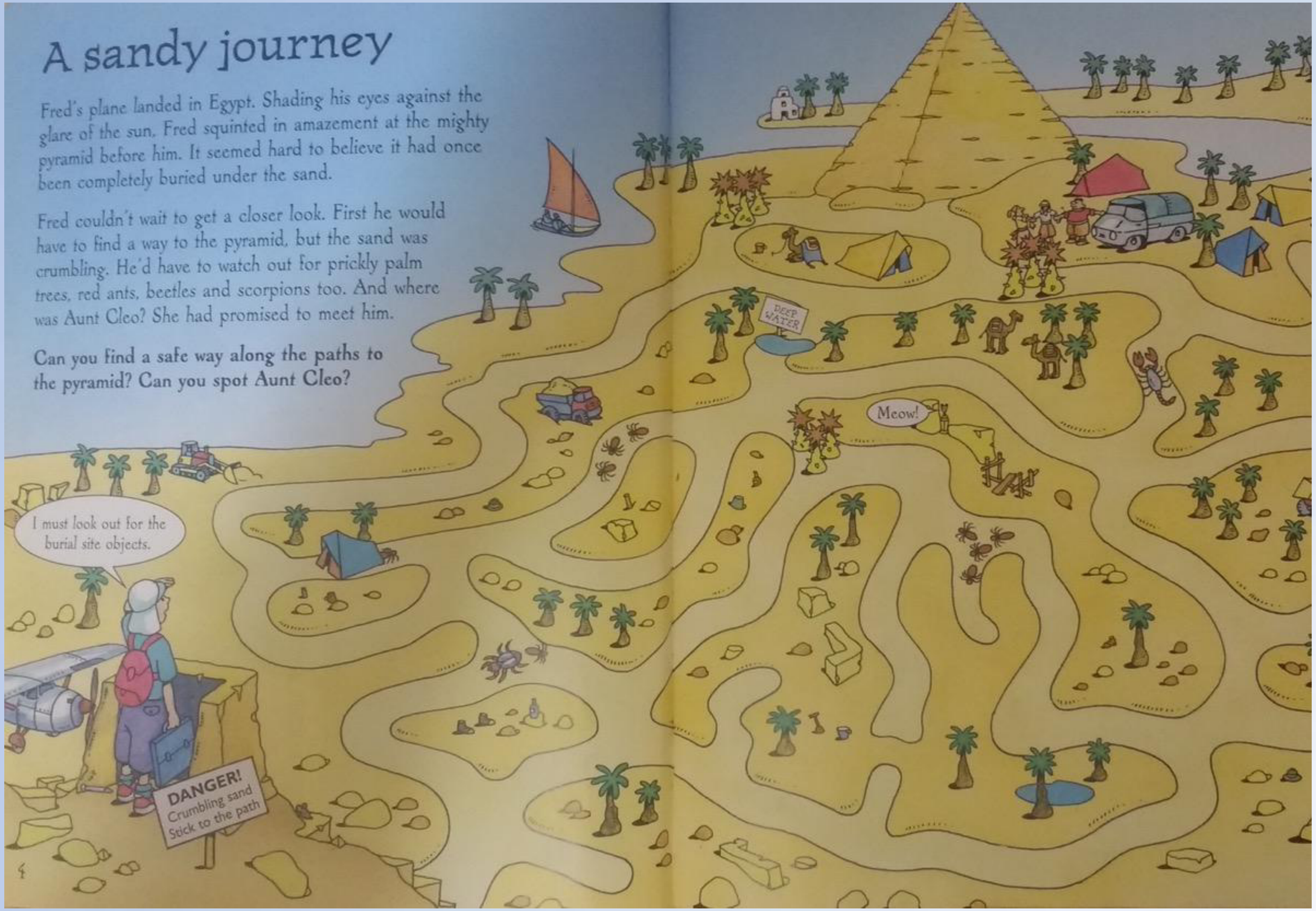
Motivation: reducing by contraction

- All sets in the family are singletons: node-weighted ATSP
- Would like to reduce the problem by contracting nonsingleton sets in



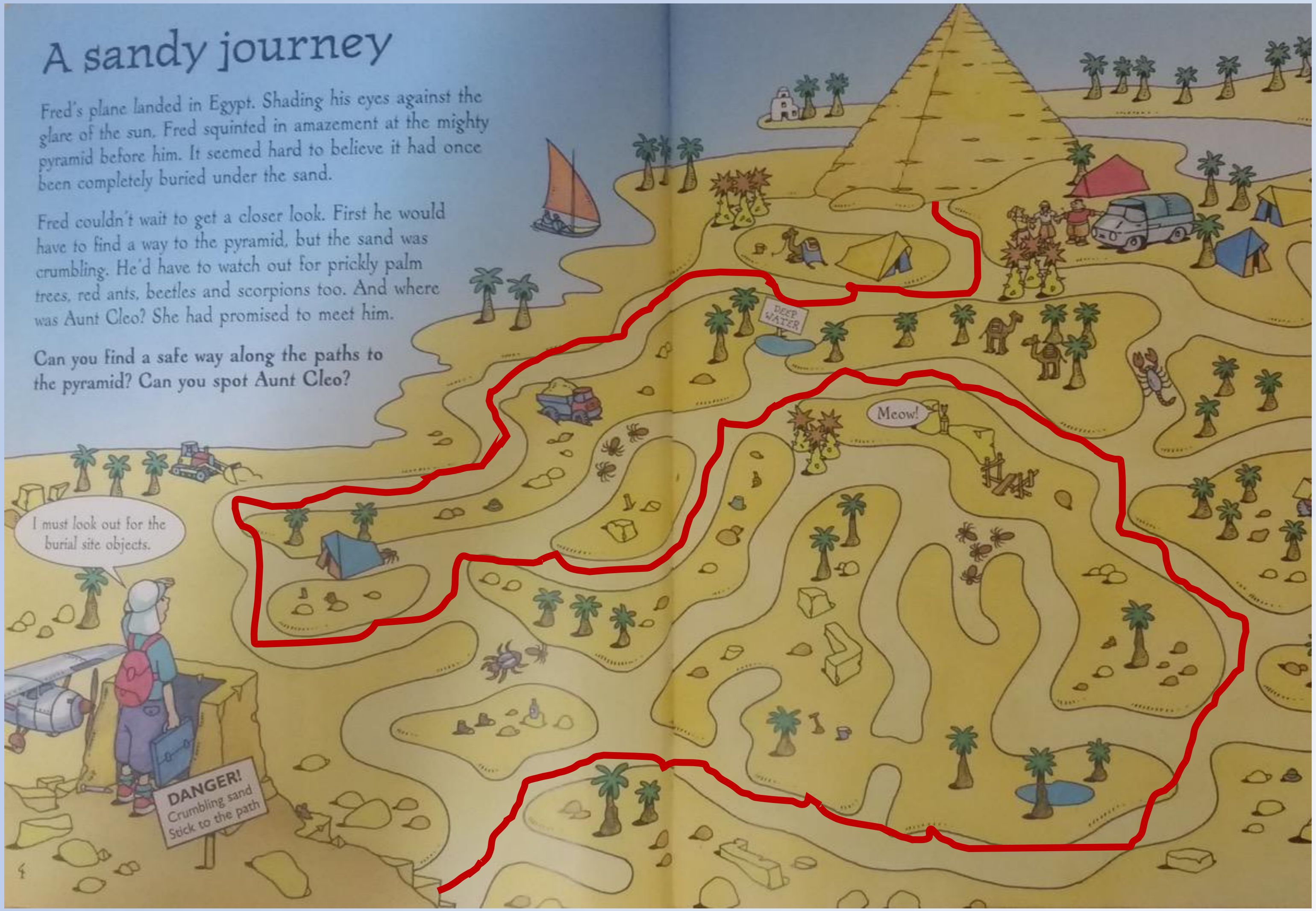
Irreducible instances

- Irreducible set S ᴺ :
  There exists ó, ù ᴺ "Υ
  such that the shortest
  path between **ó** and **ù**
  inside "Υ visits ά*almost all*έ sets "Υ, ᴺ
- Irreducible instance
  = ("0, , ὼ, ώ)*:*
  all sets in are
  irreducible



Irreducible instances

- Irreducible set **S ᴺ** :
  There exists **ó, ù ᴺ "Y**
  such that the shortest
  path between **ó** and **ù**
  inside "Y visits ά*almost all*έ sets "Y, ᴺ
- Irreducible instance
  = ("0, , ὼ, ώ):
  all sets in are
  irreducible



Irreducible instances

- Reducible set $S \subseteq$: For every pair $u, v \in S$, there is a “cheap” path connecting them (if they are connected).
- Reducible sets can be contracted.

Theorem:

polytime -approximation for irreducible instances

polytime ψ -approximation for arbitrary instances

Roadmap

General ATSP

LP duality + uncrossing

Laminarly weighted ATSP

Graph theory: contractions

Irreducible instances

Node weighted algorithm + contractions

Vertebrate pairs

[Svensson ’15]

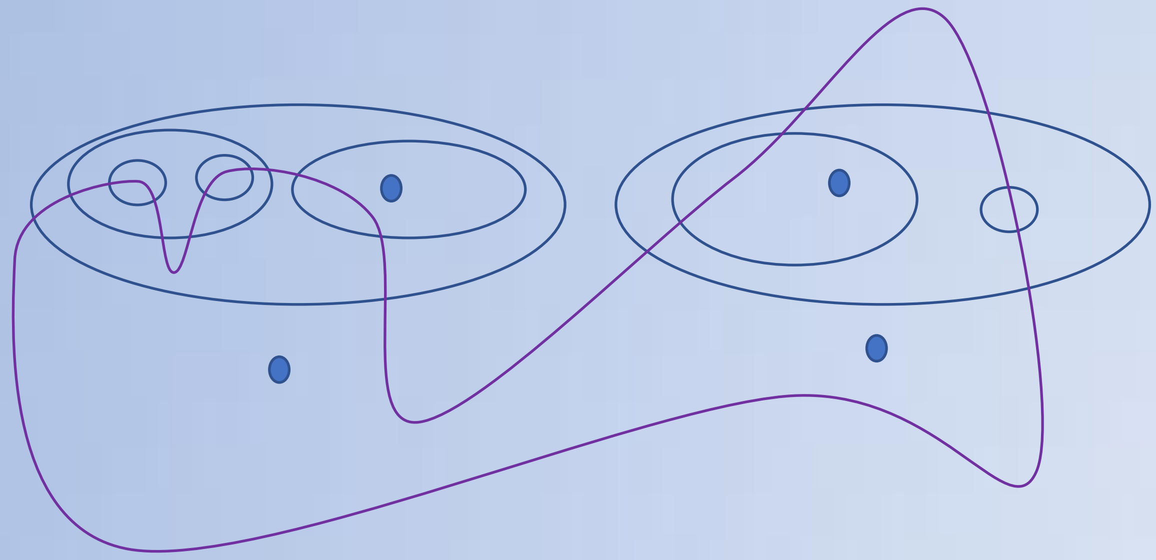
Local-connectivity ATSP

O(1)-light lcATSP algorithm in vertebrate pairs

Vertebrate pairs

Vertebrate pair (, ő)

- = (ῗ0, , ὼ, ώ) instance
- ő: backbone = subtour that crosses every nonsingleton set in



Finding a vertebrate pair in an irreducible instance = (‶0, ,ὼ,ώ)

1. Obtain a node-weighted instance by contracting all maximal sets in
2. Use [Svensson ’15] to find a tour here, and blow it back to a subtour ő in the original instance in a pessimistic way:
   inside each maximal S ᴺ , ő crosses π.χυὺ ἀόὨ ‶Υ
3. If it crosses every set in , then (, ő) is a vertebrate pair
4. Otherwise, recurse by contracting all maximal sets in not crossed by ő.
   This works because their total weight is π.ϛυὺ ἀόὨ

Roadmap

General ATSP

LP duality + uncrossing

Laminarly weighted ATSP

Graph theory: contractions

Irreducible instances

Node weighted algorithm + contractions

Vertebrate pairs

[Svensson '15]

Local-connectivity ATSP

O(1)-light lcATSP algorithm in vertebrate pairs

Summary

- Via all these reductions, we obtain an **5500**-approximation algorithm for ATSP.
- Squeezing the arguments a bit more and opening up black boxes, can be probably decreased to a few hundreds.
- Still very far from lower bound 2 on the integrality gap of Held-Karp

Open questions

- Improve to a constant **< ρππ**
- Thin tree conjecture is still open.
- Bottleneck ATSP.
- Better than 3/2 approximation for symmetric TSP.

ScaleOpt

Scaling Methods for Discrete and Continuous Optimization

- ERC Starting Grant 2018-22
- Openings for post docs and PhD students

  http://personal.lse.ac.uk/veghl/scaleopt.html

European Research Council

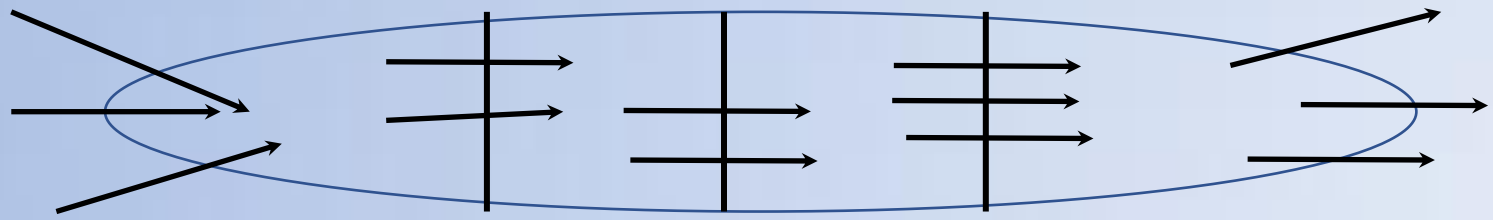
The London School of Economics and Political Science

Thank you!

Simplifying assumption *for the talk*

Assumption: all sets in the family are strongly connected in ‶0.

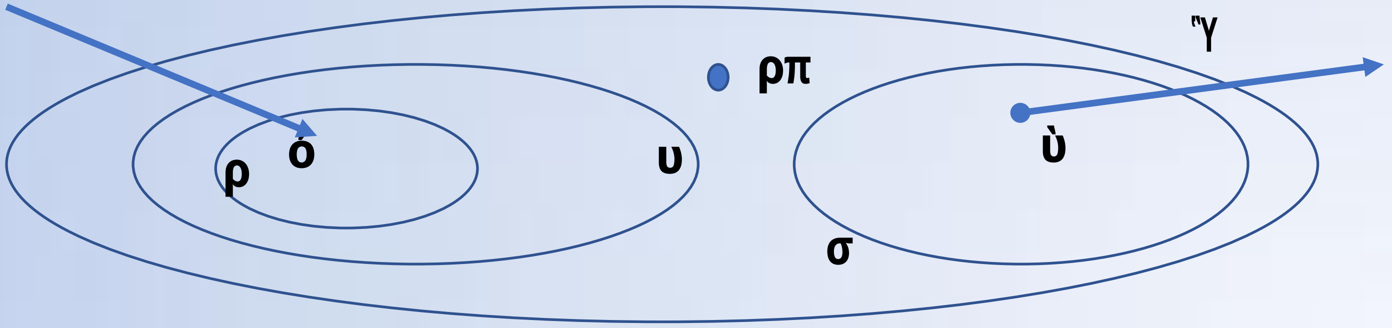
Not true in general, but the connected components have a nice path structure:



Paths traversing a set

- How much is the weight of connecting an incoming and an outgoing edge in a set S ᴺ ?

Ὀᵩ(ó, ὺ) =

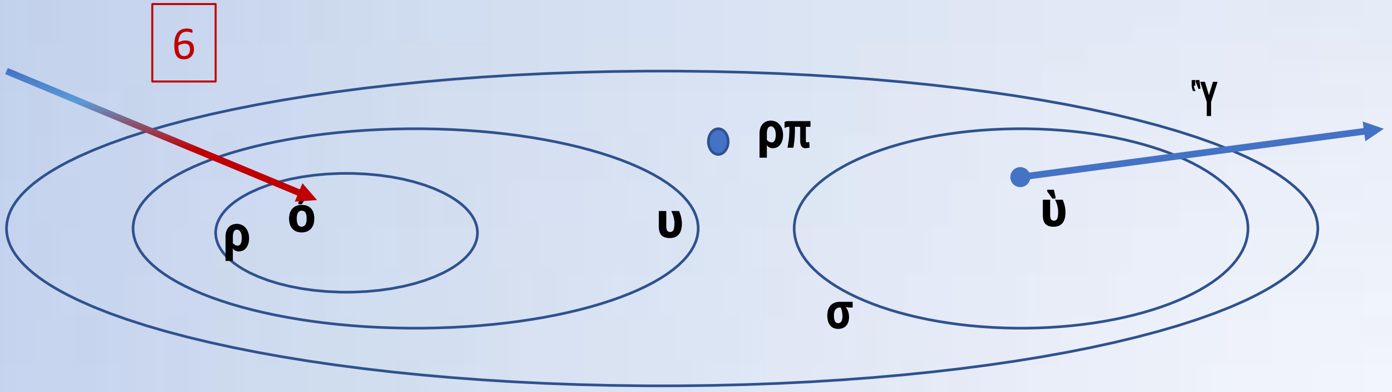


Paths traversing a set

- How much is the weight of connecting an incoming and an outgoing edge in a set **S ᴺ** ?

Ὀṗ(ό, ὺ) = ώɩ

ɩ:ẓ ᴺ ɩ,ɩ ṗ



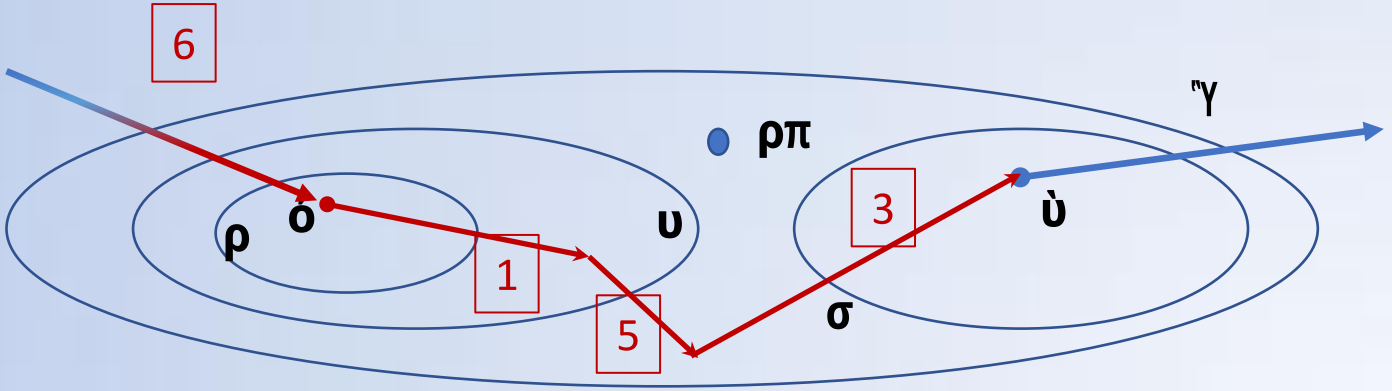
Paths traversing a set

- How much is the weight of connecting an incoming and an outgoing edge in a set S ᴺ ?

Ọ̓ṿ(ó, ὺ) = ώι + Ὼṿ(ó, ὺ)

ι:ϳ ᴺ ι,ι ṿ

Min weight path inside Ὑ.

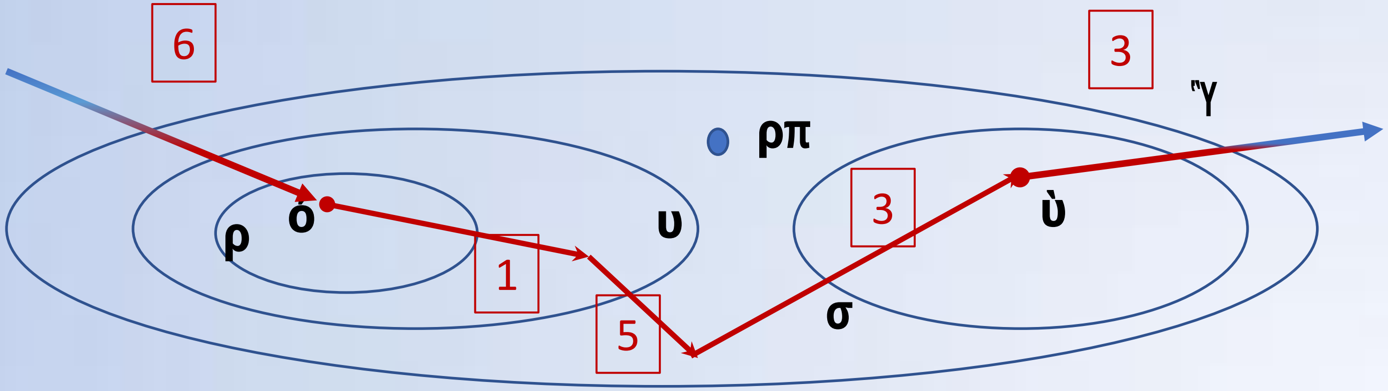


Paths traversing a set

- How much is the weight of connecting an incoming and an outgoing edge in a set S ᴺ ?

Ὀٻ(ό, ὺ) = ώι + Ὼٻ(ό, ὺ) + ώι = ρψ

ι:ذ ᴺ ι,ι ٻ ι:ذᴺ ι,ι ٻ



Paths traversing a set

- How much is the weight of connecting an incoming and an outgoing edge in a set S ɴ ?

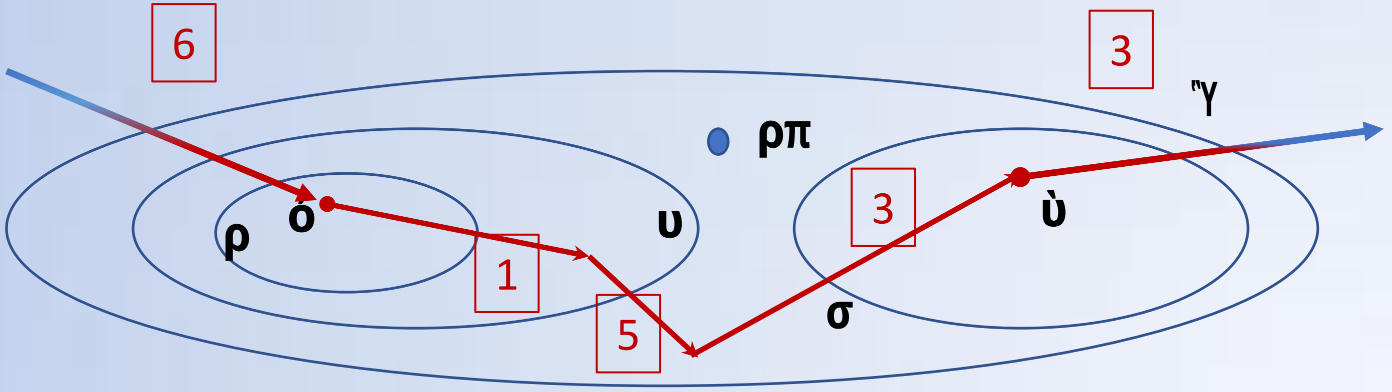
Ὄᶃ(ό, ὸ) = ώɩ + Ὼᶃ(ό, ὸ) + ώɩ = ρψ σψ

ɩ:ذ ɴ ɩ,ɩ ب ɩ:ذ ɴ ɩ,ɩ ب

Lemma:

Ὄᶃ(ό, ὸ) ς ώɩ = ὸ ὰόὨ ἿϒὊ

ɩ ب



Irreducible instances

- Reducible set **S** ᴺ :

  Max ... ʼO(ó, ù) $\frac{\sigma}{\tau}$ ù ὰóὩ(‶Y)

- Irreducible instance = ‶O, , ὼ, ώ :
  no set **S** ᴺ is reducible

Lemma:

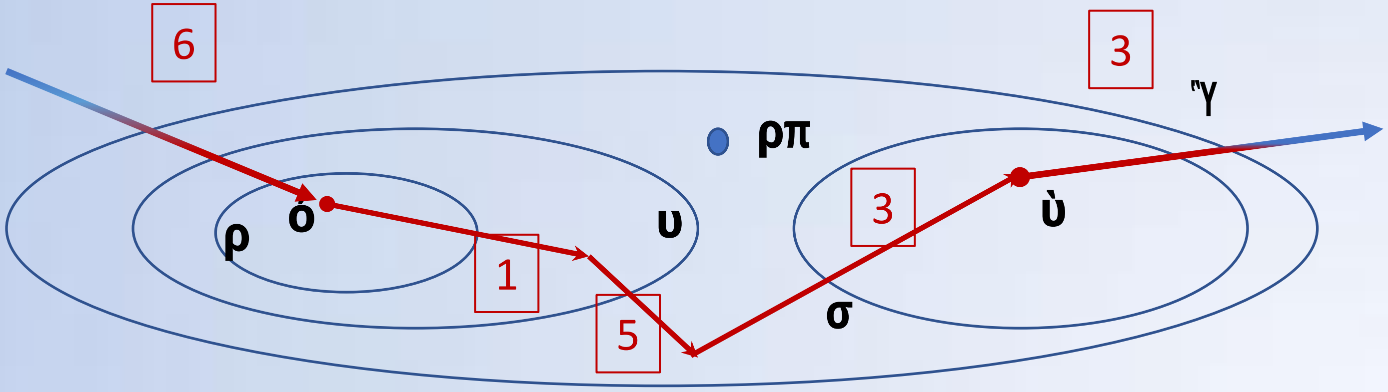
ʼO(ó, ù) ς ώι = ù ὰóὩ ‶Y

ι ب

Theorem:

polytime -approximation for irreducible instances

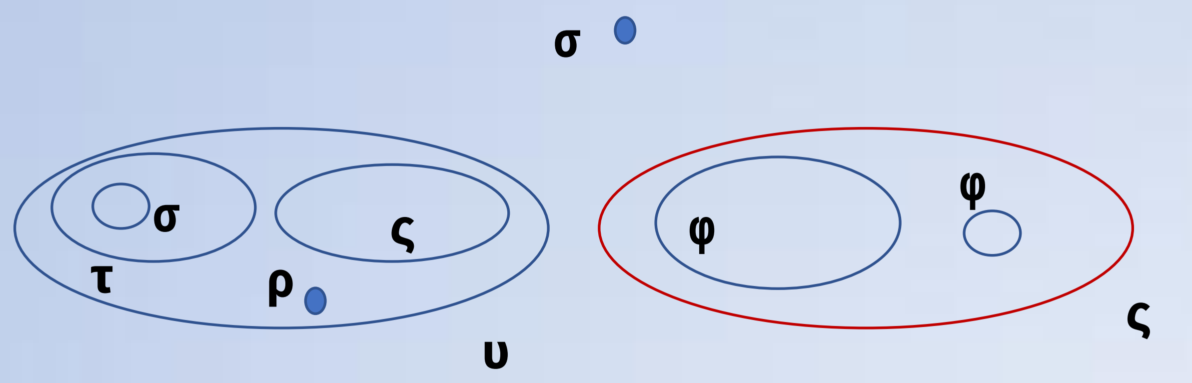
polytime ψ -approximation for arbitrary instances



Recursive algorithm via contractions

- Instance = ὍΟ, ,ὼ,ώ
- ὺ ὰόῺ( ) = ς ι ώι
  =Held-Karp optimum
- ὝΥ: minimal reducible set in .

ὺ ὰόῺ( ) = φτ



ψ -approximation for =

ψ -approximation on instance by contracting ὝΥ

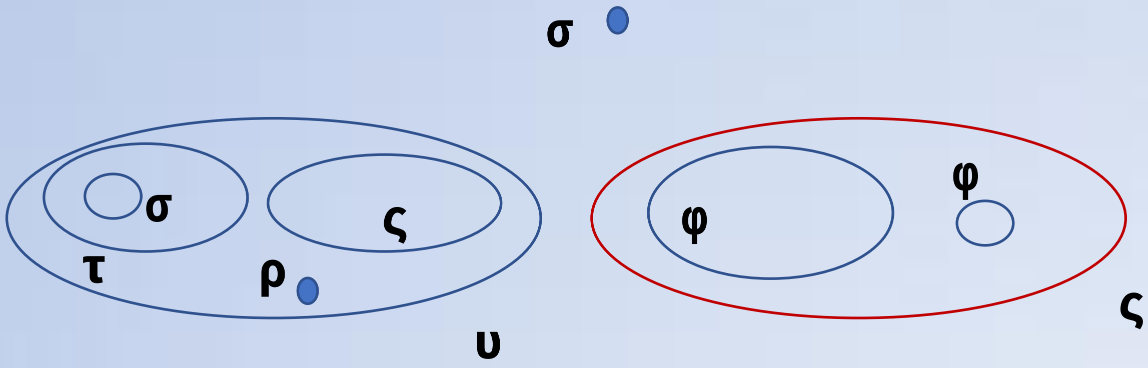
+

-approximation of irreducible instance ὰinsideἐ ὝΥ

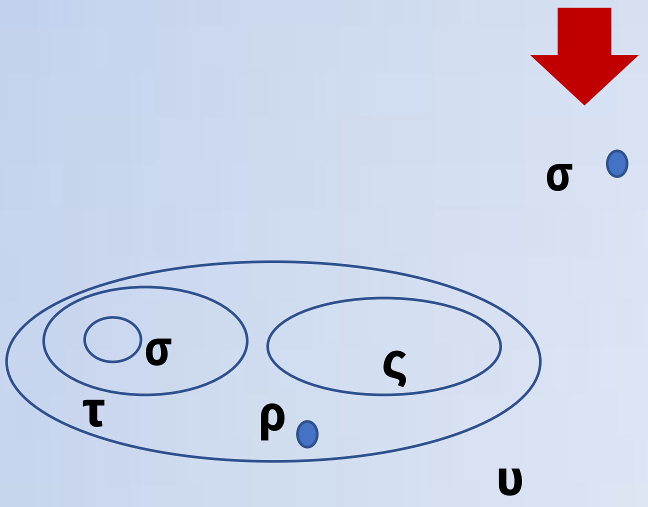
Recursive algorithm via contractions

- Instance = ῙO, , ὼ, ώ
- ὺ ἀόΏ( ) = ς ι ώι
  =Held-Karp optimum
- ῙΥ: minimal reducible set in .
- ′ = /ῙΥ: contract ῙΥ in .
- ῙΥ →
- ώ{ د} = ώ ب + ۵/8 ὺ ἀόΏ ῙΥ
- ὺ ἀόΏ(′) = ὺ ἀόΏ( ) − ۳/۶ ὺ ἀόΏ ῙΥ

ὺ ἀόΏ( ) = φτ



ὺ ἀόΏ(′) = υψ



ρρ = ς + σ/ψ ςτ

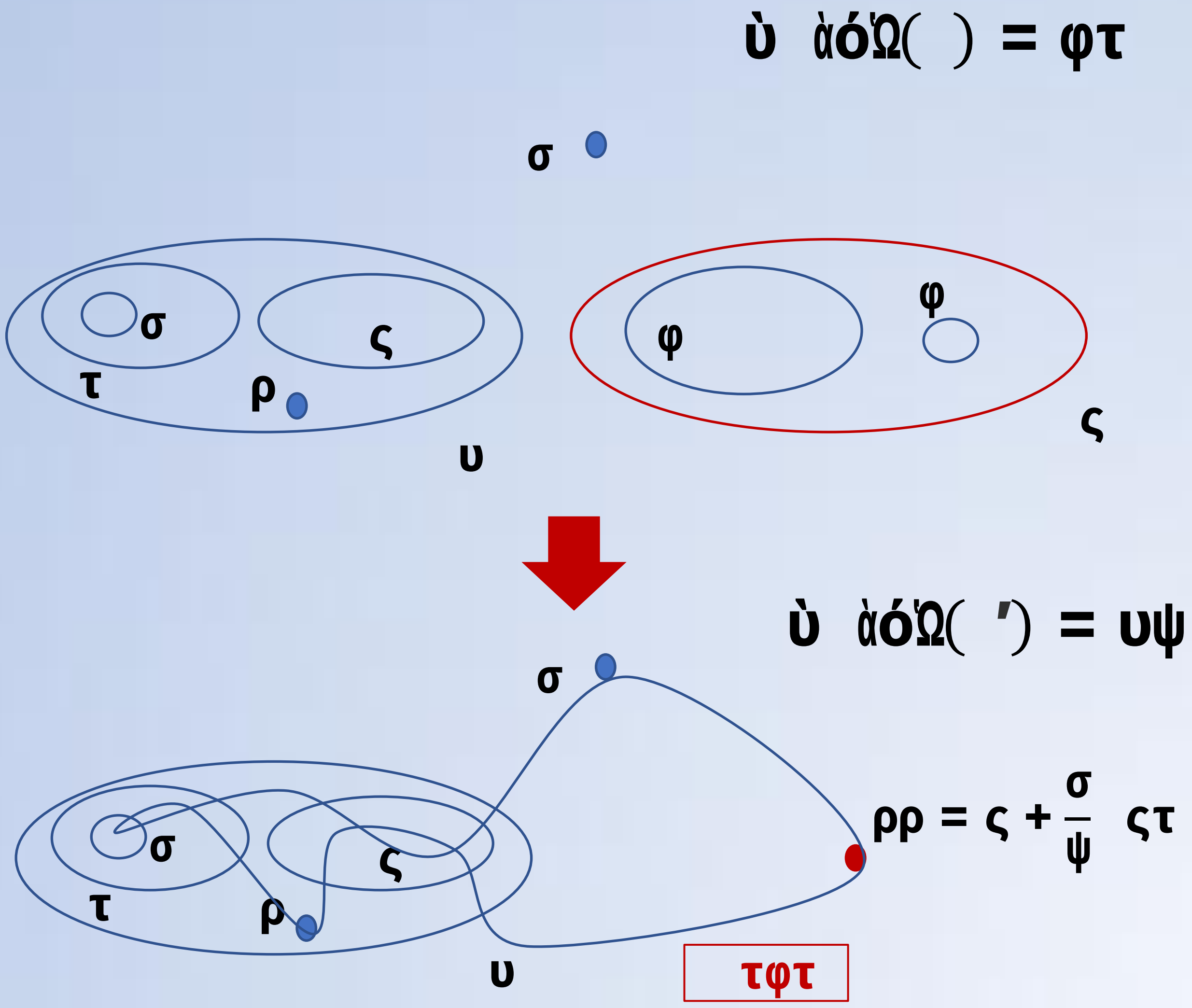
Recursive algorithm via contractions

Inductive assumption: We have a polytime ψ -approximation for smaller instances

- Apply recursively on ′ to obtain tour ‴Ύ′

ΰ(‴Ύ′) ψ ὺ ἀόὩ(′)

= ψ ὺ ἀόὩ( ) − $\frac{ρ}{τ}$ὺ ἀόὩ(‴Ύ)

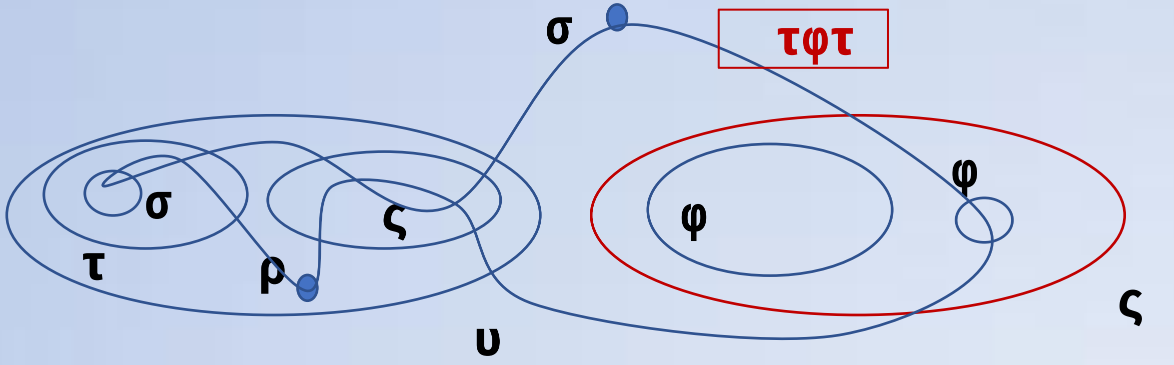


Contracting ʺΥ

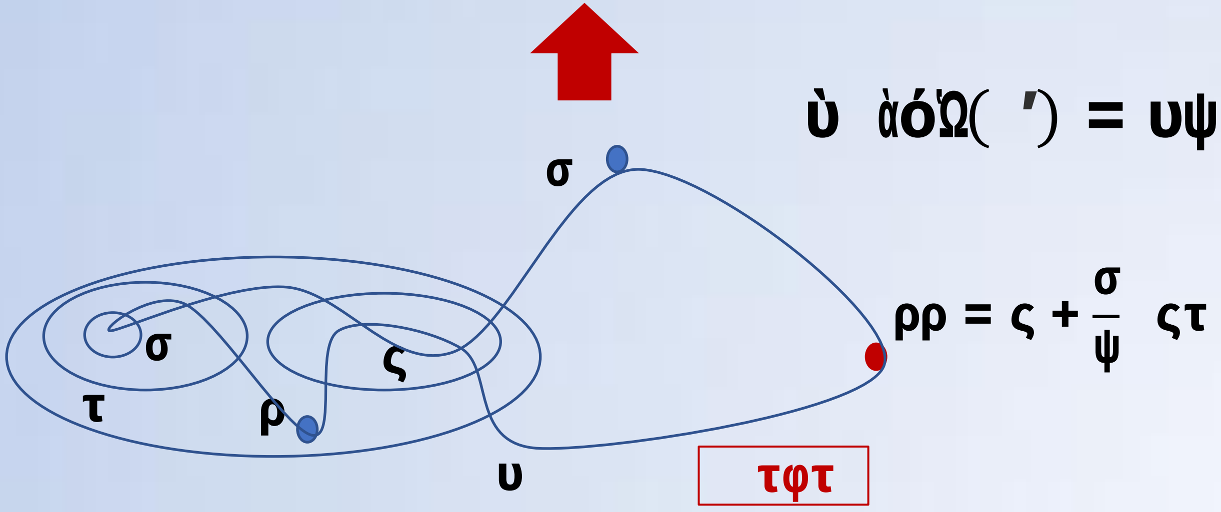
Inductive assumption: We have a polytime ψ -approximation for smaller instances

- Apply recursively on ʹ to obtain tour ʺΥʹ
  ύ (ʺΥʹ) ψὺ ἀόῺ(ʹ)
  = ψ ὺ ἀόῺ( ) − $\frac{\rho}{\tau}$ὺ ἀόῺ(ʺΥ)
- Map back to subtour ʺΥ in with
  ύ (ʺΥ) ύ ʺΥʹ

ὺ ἀόῺ( ) = φτ



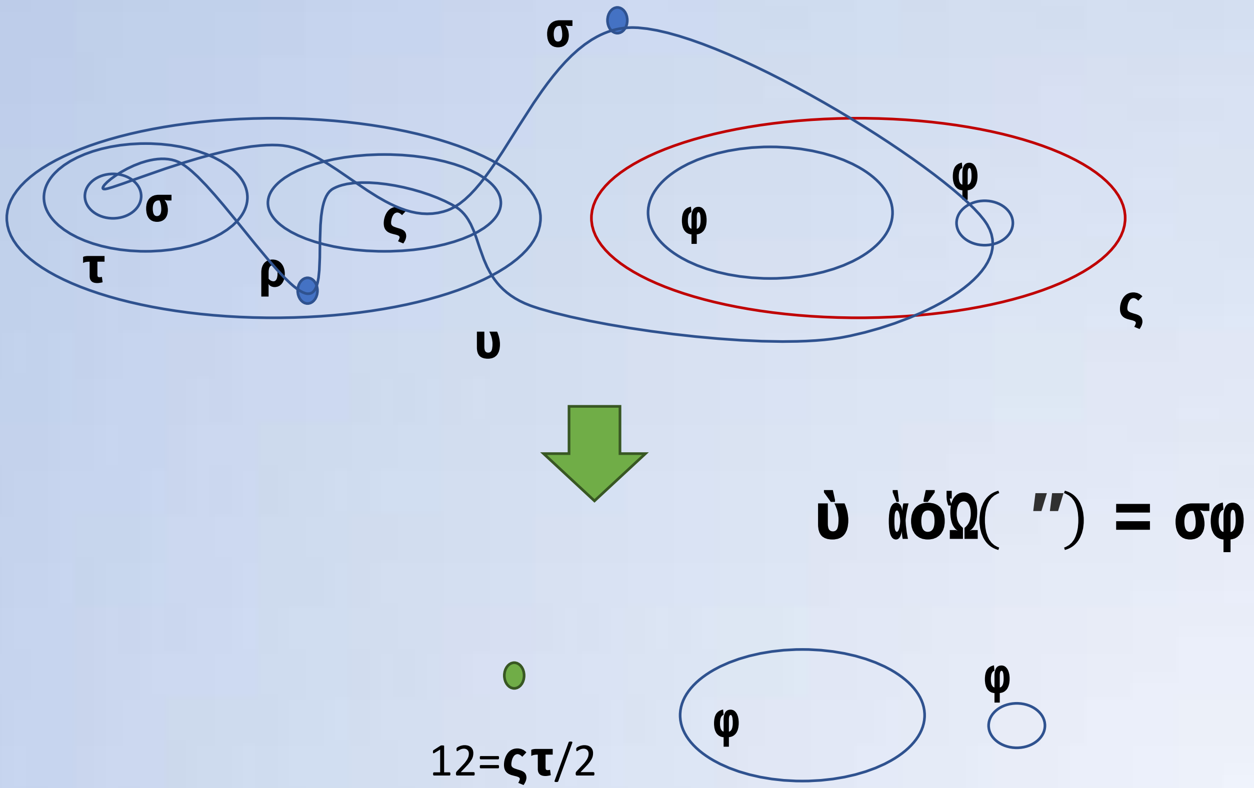
ὺ ἀόῺ(ʹ) = υψ



Inducing on "Ύ

- We add a tour "Ό̦, inside *S*, using the -approximation on irreducible instances.
- ʺ: remove *S*, and contract "Ύ to , with
  ώ{ɔ} = ὺ ἀόΏ "Ύ /ς
- ʺ is irreducible.

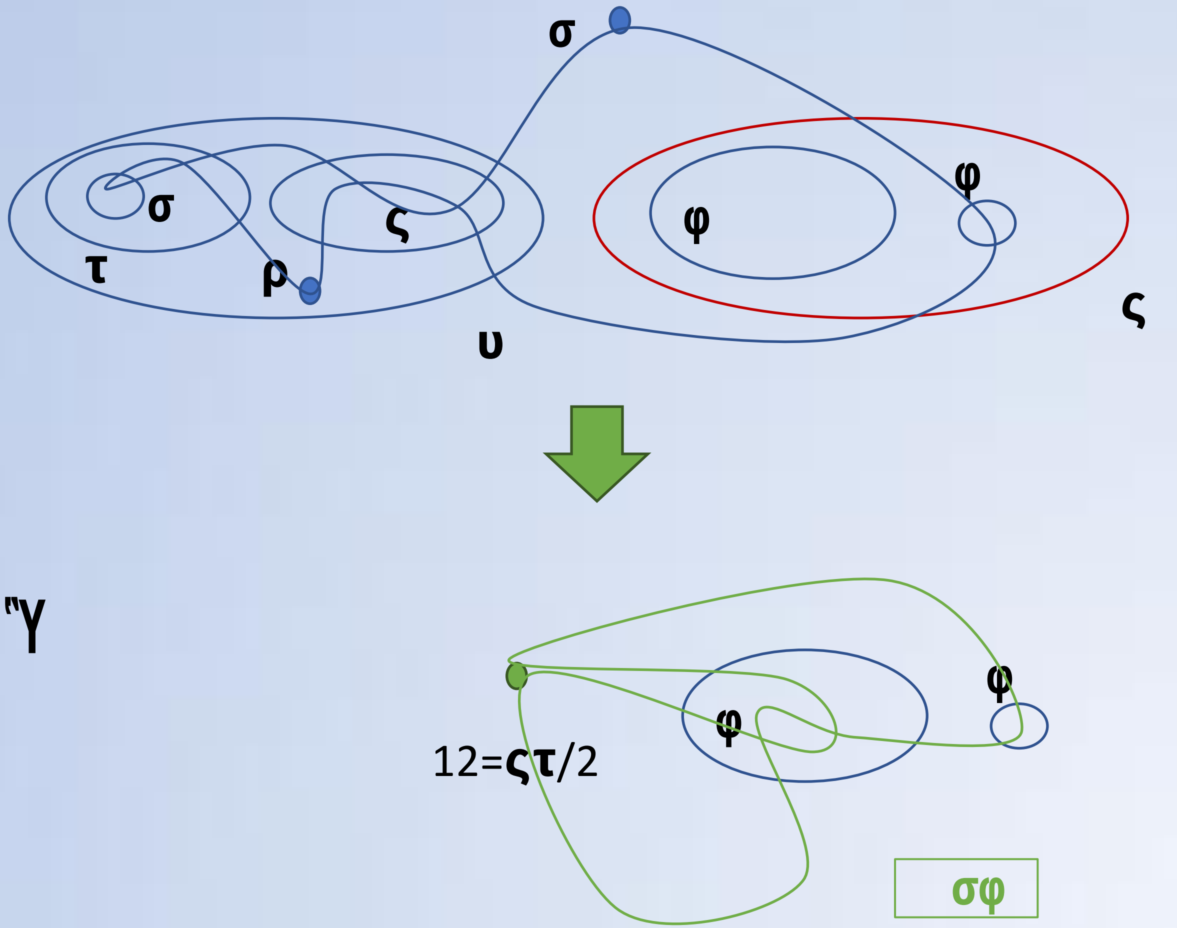
ὺ ἀόΏ( ) = φτ



Inducing on ῪΥ

- We add a tour ῨǪ inside *S*, using the -approximation on irreducible instances.
- ʺ: remove *S*, and contract ῪΥ to , with
  ώ{ɔ} = ὺ ἀόΏ ῪΥ /ς
- ʺ is irreducible.
- Find tour ῨΟʺ in ʺ with weight
  ύ(ῨΟʺ) ὺ ἀόΏ(ʺ) = ς ὺ ἀόΏ ῪΥ

ὺ ἀόΏ( ) = σς



Inducing on ʺϒ

- Find tour ʺ0ʺ in ʺ with weight

ú(ʺ0ʺ) ù àóΩ(ʺ) = ς ù àóΩ ʺϒ

- Map back ʺ0ʺ to ʺ0̦ in with

ú(ʺ0̦) ú(ʺ0) ς ù àóΩ ʺϒ

- ʺϒ˂ ʺ0̦ is a tour in

ú(ʺϒ˂ ʺ0̦) ψ (ù àóΩ( ) − ٣/٤ ù àóΩ(ʺϒ))

+ς ù àóΩ(ʺϒ) = ψ ù àóΩ( )

ù àóΩ( ) = σς