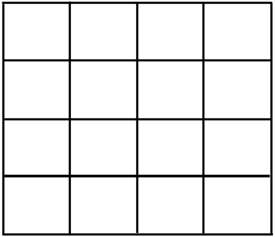
STAR ROUTING: Entre Ruteo de Vehículos y Vertex Cover

Guido Tagliavini Ponce

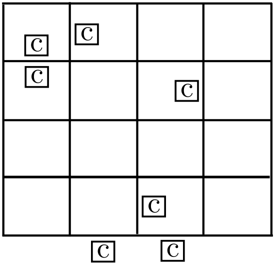
2 de agosto de 2017

El problema

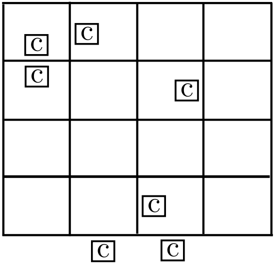
El problema



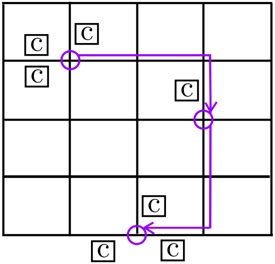
El problema



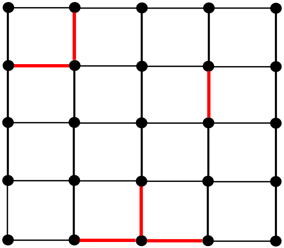
El problema



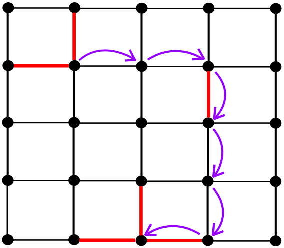
El problema



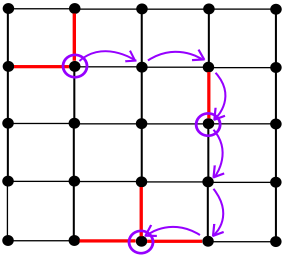
Modelado del problema



Modelado del problema



Modelado del problema



Descripción formal

Descripción formal

STAR ROUTING

INSTANCIA: $\mathsf{G} = (\mathsf{V}, \mathsf{E})$ un grafo simple y no dirigido, y $\mathsf{X} \subseteq \mathsf{E}$.

SALIDA: Un camino de G, que *cubra* X, de longitud mínima.

Convenciones:

Descripción formal

$\mathsf{STAR\ ROUTING}$

INSTANCIA: $\mathsf{G} = (\mathsf{V}, \mathsf{E})$ un grafo simple y no dirigido, y $\mathsf{X} \subseteq \mathsf{E}$.

SALIDA: Un camino de G, que *cubra* X, de longitud mínima.

Convenciones:

- Abreviamos SR a $\mathsf{STAR\ ROUTING}$.

Descripción formal

STAR ROUTING

INSTANCIA: $G = (V, E)$ un grafo simple y no dirigido, y $X \subseteq E$.

SALIDA: Un camino de G, que *cubra* X, de longitud mínima.

Convenciones:

- Abreviamos SR a STAR ROUTING.
- A las aristas de X las llamamos *clientes*.

Descripción formal

STAR ROUTING

INSTANCIA: $G = (V, E)$ un grafo simple y no dirigido, y $X \subseteq E$.

SALIDA: Un camino de G, que *cubra* X, de longitud mínima.

Convenciones:

- Abreviamos SR a STAR ROUTING.
- A las aristas de X las llamamos *clientes*.
- Decimos que un vértice u *cubre* a una arista e, si u es un extremo de e. Este concepto se extiende a conjuntos de vértices, caminos, y conjuntos de aristas.

Descripción formal

STAR ROUTING

INSTANCIA: $\mathsf{G} = (\mathsf{V}, \mathsf{E})$ un grafo simple y no dirigido, y $\mathsf{X} \subseteq \mathsf{E}$.

SALIDA: Un camino de G, que *cubra* X, de longitud mínima.

Observaciones:

Descripción formal

STAR ROUTING

INSTANCIA: $\mathsf{G} = (\mathsf{V}, \mathsf{E})$ un grafo simple y no dirigido, y $\mathsf{X} \subseteq \mathsf{E}$.

SALIDA: Un camino de G, que *cubra* X, de longitud mínima.

Observaciones:

- G es cualquier grafo, no necesariamente una grilla.

Descripción formal

STAR ROUTING

INSTANCIA: $G = (V, E)$ un grafo simple y no dirigido, y $X \subseteq E$.

SALIDA: Un camino de G, que *cubra* X, de longitud mínima.

Observaciones:

- G es cualquier grafo, no necesariamente una grilla.
- G no tiene pesos. Buscamos optimizar la *cantidad* de aristas que usa el camino.

Descripción formal

STAR ROUTING

INSTANCIA: $\mathsf{G} = (\mathsf{V}, \mathsf{E})$ un grafo simple y no dirigido, y $\mathsf{X} \subseteq \mathsf{E}$.

SALIDA: Un camino de G, que *cubra* X, de longitud mínima.

Observaciones:

- G es cualquier grafo, no necesariamente una grilla.
- G no tiene pesos. Buscamos optimizar la *cantidad* de aristas que usa el camino.
- La respuesta del problema es un camino. No nos interesa conocer las esquinas donde debemos detenernos.

Descripción formal

STAR ROUTING

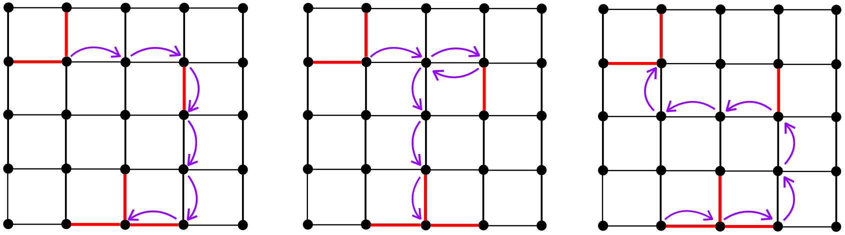
INSTANCIA: $\mathsf{G} = (\mathsf{V}, \mathsf{E})$ un grafo simple y no dirigido, y $\mathsf{X} \subseteq \mathsf{E}$.

SALIDA: Un camino de G, que *cubra* X, de longitud mínima.

Observaciones:

- G es cualquier grafo, no necesariamente una grilla.
- G no tiene pesos. Buscamos optimizar la *cantidad* de aristas que usa el camino.
- La respuesta del problema es un camino. No nos interesa conocer las esquinas donde debemos detenernos.
- La versión de decisión asociada al problema es la obvia.

Ejemplos



(a) (b) (c)

(a) y (b) son soluciones óptimas.
(c) es una solución factible pero no óptima.

Estructura de la tesis y presentación

Estructura de la tesis y presentación

Parte 1: Estudio de la complejidad de SR, restringiendo G a distintas clases de grafos.

Estructura de la tesis y presentación

Parte 1: Estudio de la complejidad de SR, restringiendo G a distintas clases de grafos.

Parte 2: Algoritmos exactos para SR, para G arbitrario.

Estructura de la tesis y presentación

Parte 1: Estudio de la complejidad de SR, restringiendo G a distintas clases de grafos.

Parte 2: Algoritmos exactos para SR, para G arbitrario.

Parte 3: Algoritmos aproximados para SR, para distintas restricciones de G.

Parte 1: Complejidad de SR

Complejidad de SR sobre grafos completos

Complejidad de SR sobre grafos completos

Teorema SR sobre grafos completos es **NP-c**.

Complejidad de SR sobre grafos completos

Teorema SR sobre grafos completos es **NP-c**.

Idea de la demo: reducción desde VERTEX COVER.

Complejidad de SR sobre grafos completos

Teorema SR sobre grafos completos es **NP-c**.

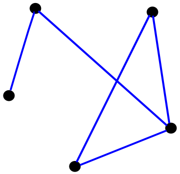
Idea de la demo: reducción desde VERTEX COVER.

VC
INSTANCIA: $G = (V, E)$ un grafo y $k \in \mathbb{Z}_{\geq 0}$.
SALIDA: ¿Existe un subconjunto de V que cubra E, de cardinal menor o igual a k?

Complejidad de SR sobre grafos completos

Teorema SR sobre grafos completos es **NP-c**.

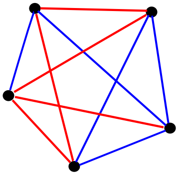
$\mathsf{H} = (\mathsf{V}, \mathsf{E})$



Calcular un
vertex cover mínimo

$\longleftrightarrow$

$\mathsf{G} = \mathsf{K}(\mathsf{V}) \quad \mathsf{X} = \mathsf{E}$

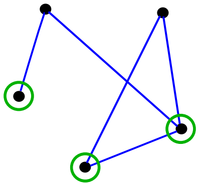


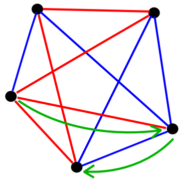
Cubrir aristas azules
con un camino

Complejidad de SR sobre grafos completos

Teorema SR sobre grafos completos es **NP-c**.

$\mathsf{H} = (\mathsf{V}, \mathsf{E})$ $\qquad\longleftrightarrow\qquad$ $\mathsf{G} = \mathsf{K}(\mathsf{V}) \quad \mathsf{X} = \mathsf{E}$





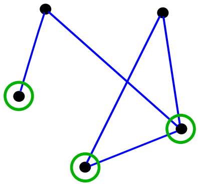
Calcular un vertex cover mínimo

Cubrir aristas azules con un camino

Complejidad de SR sobre grafos completos

Teorema SR sobre grafos completos es **NP-c**.

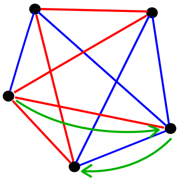
$\mathsf{H} = (\mathsf{V}, \mathsf{E})$



Calcular un
vertex cover mínimo

$\longleftrightarrow$

$\mathsf{G} = \mathsf{K}(\mathsf{V}) \quad \mathsf{X} = \mathsf{E}$



Cubrir aristas azules
con un camino

Corolario SR general es **NP-c**.

Complejidad de SR sobre árboles

Complejidad de SR sobre árboles

Teorema SR sobre árboles está en **P**. Más aún, hay un algoritmo de tiempo lineal que lo resuelve.

Complejidad de SR sobre árboles

Teorema SR sobre árboles está en **P**. Más aún, hay un algoritmo de tiempo lineal que lo resuelve.

Idea de la demo:

Complejidad de SR sobre árboles

Teorema SR sobre árboles está en **P**. Más aún, hay un algoritmo de tiempo lineal que lo resuelve.

Idea de la demo:

- Mirar el árbol como un árbol enraizado.

Complejidad de SR sobre árboles

Teorema SR sobre árboles está en **P**. Más aún, hay un algoritmo de tiempo lineal que lo resuelve.

Idea de la demo:

- Mirar el árbol como un árbol enraizado.
- Algoritmo: PD bottom-up, desde las hojas hacia la raíz, calculando varios valores auxiliares, a partir de los cuales calculamos el resultado final.

Complejidad de SR sobre grafos grilla

Complejidad de SR sobre grafos grilla

Teorema SR sobre grafos grilla, usando una *representación implícita*, es **NP-c**.

Complejidad de SR sobre grafos grilla

Teorema SR sobre grafos grilla, usando una *representación implícita*, es **NP-c**.

¿Qué es una representación implícita?

Complejidad de SR sobre grafos grilla

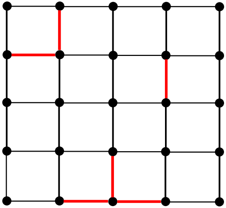
Teorema SR sobre grafos grilla, usando una *representación implícita*, es **NP-c**.

¿Qué es una representación implícita? Es una forma de representación física de la entrada (el grafo G y el subconjunto de aristas X).

Complejidad de SR sobre grafos grilla

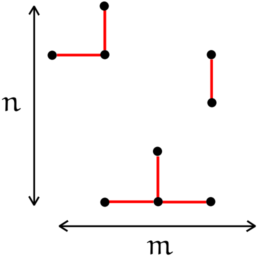
Teorema SR sobre grafos grilla, usando una *representación implícita*, es **NP-c**.

¿Qué es una representación implícita? Es una forma de representación física de la entrada (el grafo G y el subconjunto de aristas X).



Representación tradicional

Espacio $\sim |V| + |E| + |X|$



Representación implícita

Espacio $\sim 1 + |X|$

Complejidad de SR sobre grafos grilla

Teorema SR sobre grafos grilla, usando una *representación implícita*, es **NP-c**.

Observaciones:

Complejidad de SR sobre grafos grilla

Teorema SR sobre grafos grilla, usando una *representación implícita*, es **NP-c**.

Observaciones:

- No es polinomialmente equivalente a una representación tradicional.

Complejidad de SR sobre grafos grilla

Teorema SR sobre grafos grilla, usando una *representación implícita*, es **NP-c**.

Observaciones:

- No es polinomialmente equivalente a una representación tradicional.
- Tiene sentido para la clase de grafos grilla, que tienen una topología regular.

Complejidad de SR sobre grafos grilla

Teorema SR sobre grafos grilla, usando una representación implícita, es **NP-c**.

Idea de la demo: reducción desde PATH TSP rectilíneo.

Complejidad de SR sobre grafos grilla

Teorema SR sobre grafos grilla, usando una representación implícita, es **NP-c**.

Idea de la demo: reducción desde PATH TSP rectilíneo.

PTSP
INSTANCIA: $G = (V, E)$ un grafo completo, $c : E \to \mathbb{Z}_{\geq 0}$ los pesos de G y $k \in \mathbb{Z}_{\geq 0}$.
SALIDA: ¿Existe un camino hamiltoniano de G, de peso menor o igual a k?

Complejidad de SR sobre grafos grilla

Teorema SR sobre grafos grilla, usando una representación implícita, es **NP-c**.

Idea de la demo: reducción desde $\mathsf{PATH\ TSP}$ rectilíneo.

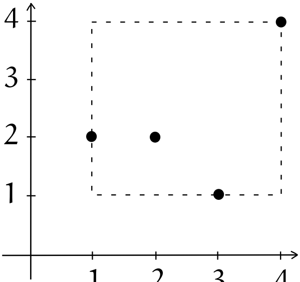
PTSP
INSTANCIA: $G = (V, E)$ un grafo completo, $c : E \to \mathbb{Z}_{\geq 0}$ los pesos de G y $k \in \mathbb{Z}_{\geq 0}$.
SALIDA: ¿Existe un camino hamiltoniano de G, de peso menor o igual a k?

Si además los vértices de V son puntos del plano, de coordenadas enteras, y c es la distancia Manhattan entre ellos, entonces el problema se llama PTSP rectilíneo. Este problema es **NP-c**.

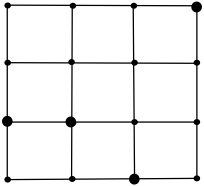
Complejidad de SR sobre grafos grilla

Teorema SR sobre grafos grilla, usando una representación implícita, es **NP-c**.

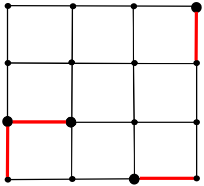
Idea de la demo: reducción desde PATH TSP rectilíneo.



Instancia de PTSP



Construimos un grafo grilla



Agregamos 1 cliente por cada vértice

Complejidad de SR sobre grafos grilla

Teorema SR sobre grafos grilla, usando una representación implícita, es **NP-c**.

Complejidad de SR sobre grafos grilla

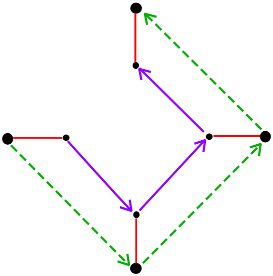
Teorema SR sobre grafos grilla, usando una representación implícita, es **NP-c**.

Problema: un recorrido que cubre los clientes de SR podría ser más corto que uno que visita los vértices de PTSP.

Complejidad de SR sobre grafos grilla

Teorema SR sobre grafos grilla, usando una representación implícita, es **NP-c**.

Problema: un recorrido que cubre los clientes de SR podría ser más corto que uno que visita los vértices de PTSP.



Complejidad de SR sobre grafos grilla

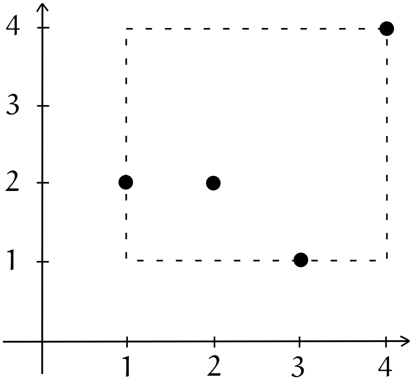
Teorema SR sobre grafos grilla, usando una representación implícita, es **NP-c**.

Solución: refinar la grilla.

Complejidad de SR sobre grafos grilla

Teorema SR sobre grafos grilla, usando una representación implícita, es **NP-c**.

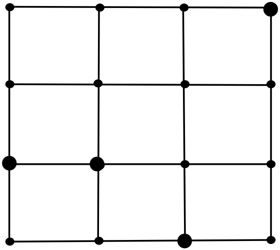
Solución: refinar la grilla.



Complejidad de SR sobre grafos grilla

Teorema SR sobre grafos grilla, usando una representación implícita, es **NP-c**.

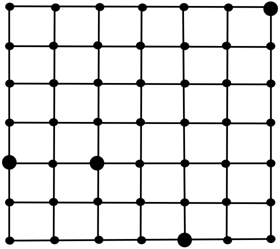
Solución: refinar la grilla.



Complejidad de SR sobre grafos grilla

Teorema SR sobre grafos grilla, usando una representación implícita, es **NP-c**.

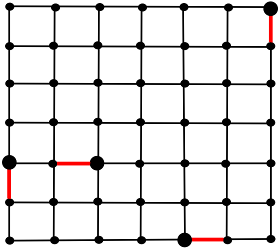
Solución: refinar la grilla.



Complejidad de SR sobre grafos grilla

Teorema SR sobre grafos grilla, usando una representación implícita, es **NP-c**.

Solución: refinar la grilla.



Complejidad de SR

Teorema SR sobre grafos grilla, usando una representación implícita, es **NP-c**.

Complejidad de SR

Teorema SR sobre grafos grilla, usando una representación implícita, es **NP-c**.

Idea clave de la demo:

Complejidad de SR

Teorema SR sobre grafos grilla, usando una representación implícita, es **NP-c**.

Idea clave de la demo:

- Cuanto más refinamos, más se parecen los caminos que cubren los clientes de SR a los caminos que visitan los vértices de PTSP.

Complejidad de SR

Teorema SR sobre grafos grilla, usando una representación implícita, es **NP-c**.

Idea clave de la demo:

- Cuanto más refinamos, más se parecen los caminos que cubren los clientes de SR a los caminos que visitan los vértices de PTSP.
- Para una cantidad suficientemente grande de refinamientos, mirar una solución de SR (en el grafo transformado) es "muy similar" a mirar una solución de PTSP (en el grafo original).

Parte 2: Algoritmos exactos para SR

Algoritmo 1: backtracking

Algoritmos exactos para SR

Algoritmo 1: backtracking

- Idea: cada solución factible de SR para (G, X) se puede pensar como el orden en que cubrimos los clientes de X.

Algoritmos exactos para SR

Algoritmo 1: backtracking

- Idea: cada solución factible de SR para (G, X) se puede pensar como el orden en que cubrimos los clientes de X.
- Algoritmo: backtracking que genera todas las permutaciones de X y para cada una determina el mínimo costo de recorrerlos en ese orden.

Algoritmos exactos para SR

Algoritmo 1: backtracking

- Idea: cada solución factible de SR para (G, X) se puede pensar como el orden en que cubrimos los clientes de X.
- Algoritmo: backtracking que genera todas las permutaciones de X y para cada una determina el mínimo costo de recorrerlos en ese orden.
- Otros backtrackings típicos, por ejemplo para la generación de caminos simples, no funcionan.

Algoritmo 2: programación dinámica

Algoritmos exactos para SR

Algoritmo 2: programación dinámica

- Idea:
  - Supongamos que llevamos construido un camino parcial P_1, que termina en un vértice u, y le resta cubrir un subconjunto de clientes F.

Algoritmos exactos para SR

Algoritmo 2: programación dinámica

- Idea:
    - Supongamos que llevamos construido un camino parcial P_1, que termina en un vértice u, y le resta cubrir un subconjunto de clientes F.
    - Ahora supongamos que tenemos otro camino parcial P_2 en las mismas condiciones.

Algoritmos exactos para SR

Algoritmo 2: programación dinámica

- Idea:
  - Supongamos que llevamos construido un camino parcial P_1, que termina en un vértice u, y le resta cubrir un subconjunto de clientes F.
  - Ahora supongamos que tenemos otro camino parcial P_2 en las mismas condiciones.
  - Los dos caminos pueden completarse a una solución factible, en forma óptima, del mismo modo.

Algoritmos exactos para SR

Algoritmo 2: programación dinámica

- Idea:
  - Supongamos que llevamos construido un camino parcial P_1, que termina en un vértice u, y le resta cubrir un subconjunto de clientes F.
  - Ahora supongamos que tenemos otro camino parcial P_2 en las mismas condiciones.
  - Los dos caminos pueden completarse a una solución factible, en forma óptima, del mismo modo.
- Definimos

$$\begin{aligned} J(u, F) = {} & \text{mínima longitud de un camino} \\ & \text{que empieza en } u \text{ y cubre } F \end{aligned}$$

Algoritmos exactos para SR

Algoritmo 2: programación dinámica

- Idea:
  - Supongamos que llevamos construido un camino parcial P_1, que termina en un vértice u, y le resta cubrir un subconjunto de clientes F.
  - Ahora supongamos que tenemos otro camino parcial P_2 en las mismas condiciones.
  - Los dos caminos pueden completarse a una solución factible, en forma óptima, del mismo modo.
- Definimos

$$J(u, F) = \text{mínima longitud de un camino que empieza en } u \text{ y cubre } F$$

- Algoritmo: PD para calcular J. Buscamos $SR^*(G, X) = \min_{e \in X} (\min\{J(e_1, X - \{e\}), J(e_2, X - \{e\})\})$.

Optimización vía funciones de acotación

Optimización vía funciones de acotación

- Optimizamos los algoritmos mediante *funciones de acotación*.

Optimización vía funciones de acotación

- Optimizamos los algoritmos mediante *funciones de acotación*.
- Idea:
  - Supongamos que llevamos construido un camino parcial P, que termina en u y le falta cubrir F. Llamemos L_{parcial} a su longitud.

Optimización vía funciones de acotación

- Optimizamos los algoritmos mediante *funciones de acotación*.
- Idea:
  - Supongamos que llevamos construido un camino parcial P, que termina en u y le falta cubrir F. Llamemos L_{parcial} a su longitud.
  - Supongamos que para completar P a una solución factible, necesitamos longitud al menos L_{resto}.

Optimización vía funciones de acotación

- Optimizamos los algoritmos mediante *funciones de acotación*.
- Idea:
  - Supongamos que llevamos construido un camino parcial P, que termina en u y le falta cubrir F. Llamemos L_{parcial} a su longitud.
  - Supongamos que para completar P a una solución factible, necesitamos longitud al menos L_{resto}.
  - Sea L_{opt} la longitud de la mejor solución encontrada hasta el momento.

Optimización vía funciones de acotación

- Optimizamos los algoritmos mediante *funciones de acotación*.
- Idea:
  - Supongamos que llevamos construido un camino parcial P, que termina en u y le falta cubrir F. Llamemos L_{parcial} a su longitud.
  - Supongamos que para completar P a una solución factible, necesitamos longitud al menos L_{resto}.
  - Sea L_{opt} la longitud de la mejor solución encontrada hasta el momento.
  - Entonces, si $L_{\text{parcial}} + L_{\text{resto}} > L_{\text{opt}}$, podemos abortar el cómputo de P.

Optimización vía funciones de acotación

- Optimizamos los algoritmos mediante *funciones de acotación*.
- Idea:
  - Supongamos que llevamos construido un camino parcial P, que termina en u y le falta cubrir F. Llamemos L_{parcial} a su longitud.
  - Supongamos que para completar P a una solución factible, necesitamos longitud al menos L_{resto}.
  - Sea L_{opt} la longitud de la mejor solución encontrada hasta el momento.
  - Entonces, si $L_{\text{parcial}} + L_{\text{resto}} > L_{\text{opt}}$, podemos abortar el cómputo de P.

Optimización vía funciones de acotación

- Optimizamos los algoritmos mediante *funciones de acotación*.
- Idea:
  - Supongamos que llevamos construido un camino parcial P, que termina en u y le falta cubrir F. Llamemos L_{parcial} a su longitud.
  - Supongamos que para completar P a una solución factible, necesitamos longitud al menos L_{resto}.
  - Sea L_{opt} la longitud de la mejor solución encontrada hasta el momento.
  - Entonces, si $L_{\text{parcial}} + L_{\text{resto}} > L_{\text{opt}}$, podemos abortar el cómputo de P.
- L_{resto} depende de u y F.

Optimización vía funciones de acotación

- Una *función de acotación* es una función B tal que $B(u, F)$ es una cota inferior de la mínima longitud necesaria para completar un (cualquier) camino parcial que termina en u y que le resta cubrir F.

Optimización vía funciones de acotación

- Una *función de acotación* es una función B tal que $\mathsf{B}(u, \mathsf{F})$ es una cota inferior de la mínima longitud necesaria para completar un (cualquier) camino parcial que termina en u y que le resta cubrir F.
- Uso: si $L_{\text{parcial}} + \mathsf{B}(u, \mathsf{F}) > L_{\text{opt}}$, abortamos la construcción del camino actual.

Optimización vía funciones de acotación

- Una *función de acotación* es una función B tal que $\mathsf{B}(u, \mathsf{F})$ es una cota inferior de la mínima longitud necesaria para completar un (cualquier) camino parcial que termina en u y que le resta cubrir F.
- Uso: si $L_{\text{parcial}} + \mathsf{B}(u, \mathsf{F}) > L_{\text{opt}}$, abortamos la construcción del camino actual.
- Ejemplos:

Optimización vía funciones de acotación

- Una *función de acotación* es una función B tal que $B(u, F)$ es una cota inferior de la mínima longitud necesaria para completar un (cualquier) camino parcial que termina en u y que le resta cubrir F.
- Uso: si $L_{\text{parcial}} + B(u, F) > L_{\text{opt}}$, abortamos la construcción del camino actual.
- Ejemplos:
    - $B(u, F) = 0$. Al menos necesitamos 0 aristas más.

Optimización vía funciones de acotación

- Una *función de acotación* es una función B tal que $B(u, F)$ es una cota inferior de la mínima longitud necesaria para completar un (cualquier) camino parcial que termina en u y que le resta cubrir F.
- Uso: si $L_{\text{parcial}} + B(u, F) > L_{\text{opt}}$, abortamos la construcción del camino actual.
- Ejemplos:
  - $B(u, F) = 0$. Al menos necesitamos 0 aristas más.
  - $B(u, F) = \tau(G[F]) - 1$. Si falta cubrir F, el camino restante debe contener un vertex cover de $G[F]$.

Optimización vía funciones de acotación

- Una *función de acotación* es una función B tal que $B(u, F)$ es una cota inferior de la mínima longitud necesaria para completar un (cualquier) camino parcial que termina en u y que le resta cubrir F.
- Uso: si $L_{\text{parcial}} + B(u, F) > L_{\text{opt}}$, abortamos la construcción del camino actual.
- Ejemplos:
  - $B(u, F) = 0$. Al menos necesitamos 0 aristas más.
  - $B(u, F) = \tau(G[F]) - 1$. Si falta cubrir F, el camino restante debe contener un vertex cover de $G[F]$.
  - $B(u, F) = \lfloor |F|/3 \rfloor - 1$.

Implementación y experimentación

Implementación:

Implementación y experimentación

Implementación:

- Ambos algoritmos son exponenciales.

Implementación y experimentación

Implementación:

- Ambos algoritmos son exponenciales.
- Se implementaron las funciones de acotación para podar ramas (Branch & Bound).

Implementación y experimentación

Implementación:

- Ambos algoritmos son exponenciales.
- Se implementaron las funciones de acotación para podar ramas (Branch & Bound).
- Se implementaron los dos algoritmos, con y sin funciones de acotación.

Implementación y experimentación

Implementación:

- Ambos algoritmos son exponenciales.
- Se implementaron las funciones de acotación para podar ramas (Branch & Bound).
- Se implementaron los dos algoritmos, con y sin funciones de acotación.

Resultados experimentales:

Implementación y experimentación

Implementación:

- Ambos algoritmos son exponenciales.
- Se implementaron las funciones de acotación para podar ramas (Branch & Bound).
- Se implementaron los dos algoritmos, con y sin funciones de acotación.

Resultados experimentales:

- Las funciones de acotación acortan el tiempo de cómputo varios órdenes de magnitud.

Implementación y experimentación

Implementación:

- Ambos algoritmos son exponenciales.
- Se implementaron las funciones de acotación para podar ramas (Branch & Bound).
- Se implementaron los dos algoritmos, con y sin funciones de acotación.

Resultados experimentales:

- Las funciones de acotación acortan el tiempo de cómputo varios órdenes de magnitud.
- Los algoritmos no logran correr instancias de más de 25 clientes en menos de 1h.

Parte 3: Algoritmos aproximados para SR

Algoritmos aproximados

Algoritmos aproximados

- Un algoritmo α-aproximado $\mathcal{A}$ para un problema de optimización Π es un algoritmo *polinomial*, que cumple que para toda instancia I de Π,

Algoritmos aproximados

- Un algoritmo α-aproximado $\mathcal{A}$ para un problema de optimización Π es un algoritmo *polinomial*, que cumple que para toda instancia I de Π,
  1. $\mathcal{A}(\mathrm{I})$ es una solución factible (no necesariamente exacta).

Algoritmos aproximados

- Un algoritmo α-aproximado $\mathcal{A}$ para un problema de optimización Π es un algoritmo *polinomial*, que cumple que para toda instancia I de Π,

  1. $\mathcal{A}(\mathrm{I})$ es una solución factible (no necesariamente exacta).
  2. $\mathrm{val}(\mathcal{A}(\mathrm{I})) \leq \alpha\ \mathsf{OPT}(\mathrm{I})$

Algoritmos aproximados

- Un algoritmo α-aproximado $\mathcal{A}$ para un problema de optimización Π es un algoritmo *polinomial*, que cumple que para toda instancia I de Π,
  1. $\mathcal{A}(\mathrm{I})$ es una solución factible (no necesariamente exacta).
  2. $\mathrm{val}(\mathcal{A}(\mathrm{I})) \leq \alpha \, \mathsf{OPT}(\mathrm{I})$
- Una estrategia frecuente para construir algoritmos aproximados para un problema Π_1 consiste en tomar un algoritmo aproximado $\mathcal{A}$ para otro problema Π_2, y usarlo como caja negra dentro de un algoritmo aproximado para Π_1.

Algoritmo aproximado para SR sobre grillas

Algoritmo aproximado para SR sobre grillas

Teorema Sea $\mathcal{A}$ un algoritmo α-aproximado para PTSP rectilíneo. Existe un algoritmo 3α-aproximado para SR sobre grillas, que usa a $\mathcal{A}$ como caja negra.

Algoritmo aproximado para SR sobre grillas

Teorema Sea $\mathcal{A}$ un algoritmo α-aproximado para PTSP rectilíneo. Existe un algoritmo 3α-aproximado para SR sobre grillas, que usa a $\mathcal{A}$ como caja negra.

Como se conoce un algoritmo (3/2)-aproximado para PTSP *métrico*...

Algoritmo aproximado para SR sobre grillas

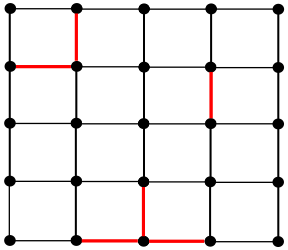
Teorema Sea $\mathcal{A}$ un algoritmo α-aproximado para PTSP rectilíneo. Existe un algoritmo 3α-aproximado para SR sobre grillas, que usa a $\mathcal{A}$ como caja negra.

Como se conoce un algoritmo $(3/2)$-aproximado para PTSP *métrico*...

Corolario Existe un algoritmo $(9/2)$-aproximado para SR sobre grillas.

Algoritmo aproximado para SR sobre grillas

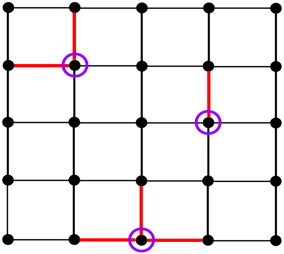
Teorema Sea $\mathcal{A}$ un algoritmo α-aproximado para PTSP rectilíneo. Existe un algoritmo 3α-aproximado para SR sobre grillas, que usa a $\mathcal{A}$ como caja negra.



Instancia de SR

Algoritmo aproximado para SR sobre grillas

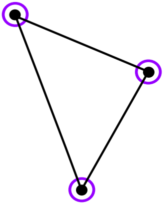
Teorema Sea $\mathcal{A}$ un algoritmo α-aproximado para PTSP rectilíneo. Existe un algoritmo 3α-aproximado para SR sobre grillas, que usa a $\mathcal{A}$ como caja negra.



Vertex cover mínimo de $G[X]$

Algoritmo aproximado para SR sobre grillas

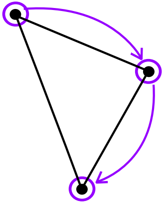
Teorema Sea $\mathcal{A}$ un algoritmo α-aproximado para PTSP rectilíneo. Existe un algoritmo 3α-aproximado para SR sobre grillas, que usa a $\mathcal{A}$ como caja negra.



Instancia de PTSP rectilíneo

Algoritmo aproximado para SR sobre grillas

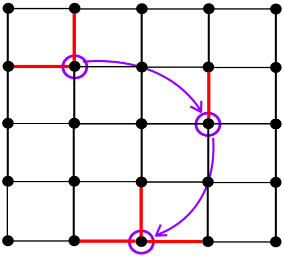
Teorema Sea $\mathcal{A}$ un algoritmo α-aproximado para PTSP rectilíneo. Existe un algoritmo 3α-aproximado para SR sobre grillas, que usa a $\mathcal{A}$ como caja negra.



Solución factible de PTSP rectilíneo

Algoritmo aproximado para SR sobre grillas

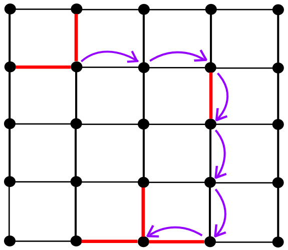
Teorema Sea $\mathcal{A}$ un algoritmo α-aproximado para PTSP rectilíneo. Existe un algoritmo 3α-aproximado para SR sobre grillas, que usa a $\mathcal{A}$ como caja negra.



La miramos sobre
la instancia de SR

Algoritmo aproximado para SR sobre grillas

Teorema Sea $\mathcal{A}$ un algoritmo α-aproximado para PTSP rectilíneo. Existe un algoritmo 3α-aproximado para SR sobre grillas, que usa a $\mathcal{A}$ como caja negra.



Solución factible de SR sobre grillas

Algoritmo aproximado para SR general

Algoritmo aproximado para SR general

- En el algoritmo anterior nos basamos en que G es bipartito para calcular un vertex cover mínimo en tiempo polinomial.

Algoritmo aproximado para SR general

- En el algoritmo anterior nos basamos en que G es bipartito para calcular un vertex cover mínimo en tiempo polinomial.
- No podemos hacer esto si G es cualquier grafo.

Algoritmo aproximado para SR general

- En el algoritmo anterior nos basamos en que G es bipartito para calcular un vertex cover mínimo en tiempo polinomial.
- No podemos hacer esto si G es cualquier grafo.
- Idea: aproximar el vertex cover utilizado.

Algoritmo aproximado para SR general

- En el algoritmo anterior nos basamos en que G es bipartito para calcular un vertex cover mínimo en tiempo polinomial.
- No podemos hacer esto si G es cualquier grafo.
- Idea: aproximar el vertex cover utilizado.

Teorema Sea $\mathcal{A}$ un algoritmo α-aproximado para PTSP métrico. Sea $\mathcal{B}$ un algoritmo β-aproximado para VC. Existe un algoritmo aproximado $\mathcal{C}$ para SR, que usa a $\mathcal{A}$ y $\mathcal{B}$ como cajas negras, tal que

$$\mathcal{C}(G, X) \leq \alpha(1+2\beta)OPT + 2\alpha(\beta-1)$$

Otros algoritmos aproximados para SR sobre grillas

Otros algoritmos aproximados para SR sobre grillas

- Motivación: nuestra empresa de repartos es muy exitosa, y tenemos clientes en casi todos los puntos de la ciudad.

Otros algoritmos aproximados para SR sobre grillas

- Motivación: nuestra empresa de repartos es muy exitosa, y tenemos clientes en casi todos los puntos de la ciudad.
- Formalmente, supongamos que la cantidad de aristas que *no* son clientes es $O(1)$.

Otros algoritmos aproximados para SR sobre grillas

- Motivación: nuestra empresa de repartos es muy exitosa, y tenemos clientes en casi todos los puntos de la ciudad.
- Formalmente, supongamos que la cantidad de aristas que *no* son clientes es $\mathsf{O}(1)$.
- Idea: en este caso, devolver un camino que cubra todas las aristas no debería ser tan malo.

Otros algoritmos aproximados para SR sobre grillas

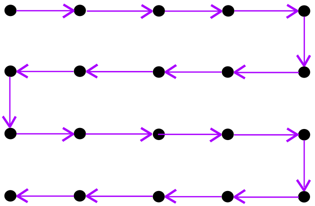
Teorema Existe un algoritmo aproximado $\mathcal{A}$ para SR sobre grillas, tal que

$$\text{length}(\mathcal{A}(\mathsf{G},\mathsf{X})) \leq 2\ \mathsf{OPT} + \mathsf{O}(1)$$

Otros algoritmos aproximados para SR sobre grillas

Teorema Existe un algoritmo aproximado $\mathcal{A}$ para SR sobre grillas, tal que

$$\text{length}(\mathcal{A}(\mathsf{G}, \mathsf{X})) \leq 2\ \mathsf{OPT} + \mathsf{O}(1)$$



Otros algoritmos aproximados para SR sobre grillas

Otros algoritmos aproximados para SR sobre grillas

Teorema Existe un algoritmo aproximado $\mathcal{A}$ para SR sobre grillas, tal que si G es una grilla de n filas y m columnas, con $n, m > 1$, entonces

$$\text{length}(\mathcal{A}(G, X)) \leq (1{,}5 + C(n, m))OPT + O(1)$$

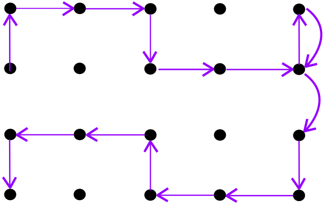
donde $C(n, m) \leq 0{,}5$, y $C(n, m)$ tiende a 0 cuando n y m tienden a infinito simultaneamente.

Otros algoritmos aproximados para SR sobre grillas

Teorema Existe un algoritmo aproximado $\mathcal{A}$ para SR sobre grillas, tal que si G es una grilla de n filas y m columnas, con $n, m > 1$, entonces

$$\text{length}(\mathcal{A}(G, X)) \leq (1{,}5 + C(n, m))OPT + O(1)$$

donde $C(n, m) \leq 0{,}5$, y $C(n, m)$ tiende a 0 cuando n y m tienden a infinito simultaneamente.



Algoritmo aproximado para SR sobre grafos completos

Algoritmo aproximado para SR sobre grafos completos

Teorema Sea $\mathcal{A}$ un algoritmo α-aproximado para VC, con α una constante. Para cada $\varepsilon > 0$ existe un algoritmo $(\alpha + \varepsilon)$-aproximado para SR sobre completos.

Algoritmo aproximado para SR sobre grafos completos

Teorema Sea $\mathcal{A}$ un algoritmo α-aproximado para VC, con α una constante. Para cada $\varepsilon > 0$ existe un algoritmo $(\alpha + \varepsilon)$-aproximado para SR sobre completos.

El algoritmo toma tiempo $\Omega(|X|^{\mathrm{cte}/\varepsilon})$. Cuanto más chico elijamos $\varepsilon > 0$, más preciso es el algoritmo, pero el polinomio tiene un exponente mayor.

Conclusiones

Conclusiones

Complejidad y aproximación

Problema	**Complejidad**	**Factor de aproximación**
SR general	**NP-c**	$\alpha(1+2\gamma)OPT + 2\alpha(\gamma-1)$
SR sobre grillas	**NP-c** (*)	$3\beta OPT$ $(1{,}5 + C(n,m))OPT + O(1)$
SR sobre completos	**NP-c**	$(\gamma+\varepsilon)OPT$
SR sobre árboles	**P**	-

*: asumiendo una representación implícita de la entrada.

Factores de aproximación: α es PTSP métrico, β es PTSP rectilíneo y γ es VC.

Conclusiones

Algoritmos exactos

Los algoritmos combinatorios propuestos no logran resolver instancias de tamaño real.

Trabajo futuro

Trabajo futuro

Trabajo futuro

- Proponer e implementar algoritmos exactos basados en técnicas de programación matemática.

Trabajo futuro

- Proponer e implementar algoritmos exactos basados en técnicas de programación matemática.
- Demostrar, si fuera cierto, que SR general es **NP-c**, asumiendo una representación tradicional del input.

Trabajo futuro

- Proponer e implementar algoritmos exactos basados en técnicas de programación matemática.
- Demostrar, si fuera cierto, que SR general es **NP-c**, asumiendo una representación tradicional del input.
- Conjeturamos que el camino que construye el algoritmo $(1{,}5 + \mathsf{C}(n, m))\mathsf{OPT} + \mathsf{O}(1)$ (o una variante muy similar de ese camino) encuentra la solución óptima de SR para (G, E) (es decir, cuando queremos cubrir todas las aristas). Demostrarlo, si fuera cierto.

Trabajo futuro

- Proponer e implementar algoritmos exactos basados en técnicas de programación matemática.
- Demostrar, si fuera cierto, que SR general es **NP-c**, asumiendo una representación tradicional del input.
- Conjeturamos que el camino que construye el algoritmo $(1{,}5 + C(n, m))OPT + O(1)$ (o una variante muy similar de ese camino) encuentra la solución óptima de SR para (G, E) (es decir, cuando queremos cubrir todas las aristas). Demostrarlo, si fuera cierto.
- Una de las cotas en la demo del algoritmo 3α-aproximado usa una cota que es del peor caso. Es posible que sea mejorable usando argumentos probabilísticos. Investigar este camino.

Fin

Agradecimientos

Gracias a todos!