Linearno programiranje i algoritam simpleksa

Petar Veličković

Matematička gimnazija
Nedelja informatike v2.0

14. decembar 2015.

Linearno programiranje

- **Linearno programiranje** (*Linear programming*) je tehnika za rešavanje *optimizacijskih* problema u kojima su svi odnosi između promenljivih *linearni* ($\sim$ jedine dozvoljene operacije su sabiranje, oduzimanje i množenje skalarom!)

- Ovo je vrlo gruba definicija…krenimo sa jednim primerom.

Politička kampanja

- Zamislite da vodite političku kampanju, i da treba da uložite određena sredstva u četiri moguće polise: izgradnju puteva, kontrolu oružja, subvencije na farme, i porez na gorivo.
- Glasači mogu živeti u gradu, predgrađu ili selu. U zavisnosti od mesta gde žive, predviđa se da će različito reagovati na svaku od ovih polisa.
- Vaša želja je da pridobijete za sebe bar 50000 glasača u gradu, 100000 glasača u predgrađu, i 25000 glasača na selu.
- Takođe želite da ovo uradite sa minimalno uloženih sredstava.

Politička kampanja, *cont'd*

- Vaš kabinet je utvrdio da, ukoliko uložite hiljadu dinara u neku od ovih polisa, dobijate (ili gubite) sledeći broj hiljada glasača:

polisa	grad	predgrađe	selo
putevi	-2	5	3
oružje	8	2	-5
farme	0	0	10
gorivo	10	0	-2

- Označimo sa x_p, x_o, x_f i x_g broj hiljada dinara uloženih u svaku od ovih polisa.

Politička kampanja: linearni program

- Sada je jednostavno formulisati naš problem:

$$
\begin{array}{lrcrcrcrcr}
\text{minimizirati} & x_p & + & x_o & + & x_f & + & x_g & & \\
\text{pod uslovima} & & & & & & & & & \\
& -2x_p & + & 8x_o & & & + & 10x_g & \geq & 50 \\
& 5x_p & + & 2x_o & & & & & \geq & 100 \\
& 3x_p & - & 5x_o & + & 10x_f & - & 2x_g & \geq & 25 \\
& & & x_p, x_o, x_f, x_g & & & & & \geq & 0
\end{array}
$$

- Svi odnosi između promenljivih su linearni: dakle ovo je korektan linearni program.

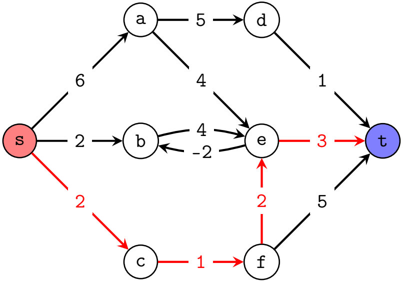
Algoritmi

- Algoritam za rešavanje sistema linearnih nejednakosti je bio poznat još Furijeu u 1827. godini (*Fourier-Motzkin Elimination*).
  - Užasna složenost: u najgorem slučaju $O(m^{2^n})$, gde je n broj promenljivih, a m broj uslova!!
- Dancig postavlja *algoritam simpleksa* 1947. godine. Ovo je prvi algoritam ove vrste koji je našao široku industrijsku primenu (i koristi se i dan-danas).
  - U najgorem slučaju, za sve varijante simpleksa koje su izmišljene do sada, moguće je naći primer koji ga prisiljava da napravi $O(m^{n/2})$ koraka. I dalje je *otvoren problem* da li je moguće napraviti varijantu koja nema ovakve probleme.
  - Međutim, prosečna složenost je reda veličine $O(n+m)$!
- Mnogo kasnije (1979.) pokazano da je linearno programiranje rešivo u polinomnom vremenu.

Primene linearnog programiranja

- *Pop-quiz:* Možete li navesti neke poznate probleme u računarstvu koji su linearni?

Najkraći putevi

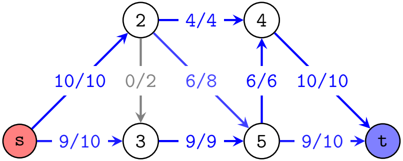


Svaki čvor ima svoju promenljivu d_v, koja označava minimalno rastojanje od izvornog čvora.

$$
\begin{array}{lllll}
\text{maksimizirati} & d_t & & & \\
\text{pod uslovima} & & & & \\
\forall (u,v) \in E & d_v & \leq & d_u & + \quad w_{u,v} \\
 & d_s & = & 0 &
\end{array}
$$

- Uporediti sa *Bellman-Ford* algoritmom.

Maksimalni protok



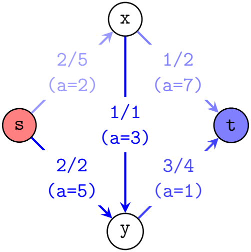
Svaka ivica ima svoju promenljivu $f_{u,v}$, koja označava protok između u i v.

$$
\begin{array}{llcr}
\text{maksimizirati} & \sum_{v\in V} f_{s,v} - \sum_{v\in V} f_{v,s} & & \\
\text{pod uslovima} & & & \\
\forall u,v \in V & f_{u,v} & \leq & c_{u,v} \\
\forall u \in V \setminus \{s,t\} & \sum_{v\in V} f_{v,u} & = & \sum_{v\in V} f_{u,v} \\
\forall u,v \in V & f_{u,v} & \geq & 0
\end{array}
$$

- Lepo, ali za oba ova problema već imamo efikasne algoritme?! (*Dijkstra*, *Johnson*, *Ford-Fulkerson*, *Dinic*. . .)
- Prava snaga linearnog programiranja se pokazuje na problemima gde (za sada) ne postoji propisano rešenje!

Protok sa minimalnom cenom

- Jedan takav problem je problem **protoka sa minimalnom cenom** (*min-cost flow*):
  - Sada, osim kapaciteta, svaka ivica ima i neku cenu a, tako da za svaku jedinicu protoka koju pustimo niz tu ivicu moramo platiti cenu a.
  - Cilj više nije pustiti što veći protok između s i t, već sa što manjom plaćenom cenom pustiti tačno d jedinica.
- Rešen primer ($d = 4$):



Protok sa minimalnom cenom

$$
\begin{array}{llcr}
\text{minimizirati} & \displaystyle\sum_{(u,v)\in E} a_{u,v} f_{u,v} & & \\
\text{pod uslovima} & & & \\
\forall u, v \in V & f_{u,v} & \leq & c_{u,v} \\
\forall u \in V \setminus \{s, t\} & \displaystyle\sum_{v\in V} f_{v,u} - \sum_{v\in V} f_{u,v} & = & 0 \\
 & \boxed{\displaystyle\sum_{v\in V} f_{s,v} - \sum_{v\in V} f_{v,s} \quad = \quad d} & & \\
\forall u, v \in V & f_{u,v} & \geq & 0
\end{array}
$$

Standardna forma

- Da bismo mogli efektivno da primenimo algoritam simpleksa na ovakve i slične probleme, neophodno je prvo da definišemo dve specijalne forme linearnih programa.
- Najpre, linearni program je potrebno izraziti u *standardnoj formi*; formi u kojoj važi sledeće:
  - Potrebno je raditi *maksimizaciju*;
  - Traži se da su sve promenljive *nenegativne*;
  - Svi uslovi (osim uslova nenegativnosti) *koriste relaciju* $\leq$.
- Linearni program se tada može zapisati u sledećoj formi:

$$\begin{array}{ll} \text{maksimizirati} & \vec{c} \cdot \vec{x} \\ \text{pod uslovima} & \\ & \begin{array}{rcl} \mathbf{A}\vec{x} & \leq & \vec{b} \\ \vec{x} & \geq & \vec{0} \end{array} \end{array}$$

Transformacija u standardnu formu

- Počnimo od linearnog programa koji krši sva tri prethodno navedena pravila:

$$
\begin{array}{lrcrcl}
\text{minimizirati} & -2x_1 & + & 3x_2 & & \\
\text{pod uslovima} & & & & & \\
 & x_1 & + & x_2 & = & 7 \\
 & x_1 & - & 2x_2 & \leq & 4 \\
 & x_1 & & & \geq & 0
\end{array}
$$

Transformacija u standardnu formu

- Iz minimizacije u maksimizaciju:

$$
\begin{array}{lrcrcr}
\text{maksimizirati} & 2x_1 & - & 3x_2 & & \\
\text{pod uslovima} & & & & & \\
& x_1 & + & x_2 & = & 7 \\
& x_1 & - & 2x_2 & \leq & 4 \\
& x_1 & & & \geq & 0
\end{array}
$$

Transformacija u standardnu formu

- Ne postoji uslov da je x_2 nenegativan; izraziti ga kao *razliku dva nenegativna broja*: $x_2 = x_3 - x_4$

$$
\begin{array}{lrcrcrcl}
\text{maksimizirati} & 2x_1 & - & 3x_3 & + & 3x_4 & & \\
\text{pod uslovima} & & & & & & & \\
& x_1 & + & x_3 & - & x_4 & = & 7 \\
& x_1 & - & 2x_3 & + & 2x_4 & \leq & 4 \\
& & & x_1, x_3, x_4 & & & \geq & 0
\end{array}
$$

Transformacija u standardnu formu

- Pretvoriti jednakost u dve nejednakosti:

$$
\begin{array}{lrcrcrcr}
\text{maksimizirati} & 2x_1 & - & 3x_3 & + & 3x_4 & & \\
\text{pod uslovima} & & & & & & & \\
 & x_1 & + & x_3 & - & x_4 & \geq & 7 \\
 & x_1 & + & x_3 & - & x_4 & \leq & 7 \\
 & x_1 & - & 2x_3 & + & 2x_4 & \leq & 4 \\
 & & & x_1, x_3, x_4 & & & \geq & 0
\end{array}
$$

Transformacija u standardnu formu

- Pretvoriti nejednakost sa $\geq$ u nejednakost sa $\leq$:

$$
\begin{array}{lrcrcrcr}
\text{maksimizirati} & 2x_1 & - & 3x_3 & + & 3x_4 & & \\
\text{pod uslovima} & & & & & & & \\
& -x_1 & - & x_3 & + & x_4 & \leq & -7 \\
& x_1 & + & x_3 & - & x_4 & \leq & 7 \\
& x_1 & - & 2x_3 & + & 2x_4 & \leq & 4 \\
& & & x_1, x_3, x_4 & & & \geq & 0
\end{array}
$$

Transformacija u standardnu formu

- (Radi konvencije) preimenujemo promenljive:

$$
\begin{array}{lrcrcrcr}
\text{maksimizirati} & 2x_1 & - & 3x_2 & + & 3x_3 & & \\
\text{pod uslovima} & & & & & & & \\
& -x_1 & - & x_2 & + & x_3 & \leq & -7 \\
& x_1 & + & x_2 & - & x_3 & \leq & 7 \\
& x_1 & - & 2x_2 & + & 2x_3 & \leq & 4 \\
& & & x_1, x_2, x_3 & & & \geq & 0
\end{array}
$$

- Linearni program je sada u standardnoj formi.

Slack forma

- Algoritam simpleksa zahteva drugačiju formu linearnog programa—*slack* formu. U ovoj formi:
  - I dalje je potrebno *maksimizovati*;
  - I dalje se zahteva da su sve promenljive *nenegativne*;
  - Jedini dozvoljeni dodatni uslovi su *jednakosti*.

- Lako je transformisati linearni program iz standardne u *slack* formu; za svaki uslov uvodimo *slack promenljivu*, koja predstavlja razliku broja sa desne strane i izraza sa leve strane.

- Pošto su svi uslovi oblika $\leq$, *slack* promenljive moraju biti nenegativne!

Transformacija u *slack* formu

- Vratimo se na prethodni primer:

$$
\begin{array}{lrcrcrcr}
\text{maksimizirati} & 2x_1 & - & 3x_2 & + & 3x_3 & & \\
\text{pod uslovima} & & & & & & & \\
 & -x_1 & - & x_2 & + & x_3 & \leq & -7 \\
 & x_1 & + & x_2 & - & x_3 & \leq & 7 \\
 & x_1 & - & 2x_2 & + & 2x_3 & \leq & 4 \\
 & & & x_1, x_2, x_3 & & & \geq & 0
\end{array}
$$

Transformacija u *slack* formu

- Uvedimo *slack* promenljivu x_4 za prvi uslov:

$$
\begin{array}{llrrrrrrrrr}
\text{maksimizirati} & & & & & 2x_1 & - & 3x_2 & + & 3x_3 \\
\text{pod uslovima} & & & & & & & & & \\
 & x_4 & = & -7 & + & x_1 & + & x_2 & - & x_3 \\
 & x_1 & + & x_2 & - & x_3 & \leq & 7 & & \\
 & x_1 & - & 2x_2 & + & 2x_3 & \leq & 4 & & \\
 & & & \multicolumn{3}{c}{x_1, x_2, x_3, x_4} & & & \geq & 0
\end{array}
$$

Transformacija u *slack* formu

- Uvedimo *slack* promenljivu x_5 za drugi uslov:

$$
\begin{array}{lrcrcrcrcr}
\text{maksimizirati} & & & & & 2x_1 & - & 3x_2 & + & 3x_3 \\
\text{pod uslovima} & & & & & & & & & \\
& x_4 & = & -7 & + & x_1 & + & x_2 & - & x_3 \\
& x_5 & = & 7 & - & x_1 & - & x_2 & + & x_3 \\
& x_1 & - & 2x_2 & + & 2x_3 & \leq & 4 & & \\
& & & \multicolumn{4}{c}{x_1, x_2, x_3, x_4, x_5} & & \geq & 0
\end{array}
$$

Transformacija u *slack* formu

- Uvedimo *slack* promenljivu x_6 za treći uslov:

$$
\begin{array}{lrcrcrcrcr}
\text{maksimizirati} & & & & & 2x_1 & - & 3x_2 & + & 3x_3 \\
\text{pod uslovima} & & & & & & & & & \\
& x_4 & = & -7 & + & x_1 & + & x_2 & - & x_3 \\
& x_5 & = & 7 & - & x_1 & - & x_2 & + & x_3 \\
& x_6 & = & 4 & + & x_1 & + & 2x_2 & - & 2x_3 \\
& & & \multicolumn{5}{c}{x_1, x_2, x_3, x_4, x_5, x_6} & \geq & 0
\end{array}
$$

- Linearni program je sada u *slack* formi.
- Promenljive sa leve strane jednakosti (x_4, x_5, x_6) nazivamo "bazičnim" (*basic*), a promenljive sa desne strane (x_1, x_2, x_3) "nebazičnim" (*nonbasic*).

Parametri za algoritam simpleksa

- Programe u *slack* formi možemo izraziti u sledećem obliku:

$$\begin{array}{ll} \text{maksimizirati} & \vec{c} \cdot \vec{x} \\ \text{pod uslovima} & \\ & \begin{array}{rcccl} \mathbf{A}\vec{x} & + & \vec{b} & = & \vec{x}_s \\ \vec{x}, \vec{x}_s & & & \geq & \vec{0} \end{array} \end{array}$$

- Algoritmu simpleksa treba proslediti parametre $\mathbf{A}$, $\vec{b}$ i $\vec{c}$. Za prethodno određen linearni program, parametri su:

$$\mathbf{A} = \begin{bmatrix} 1 & 1 & -1 \\ -1 & -1 & 1 \\ 1 & 2 & -2 \end{bmatrix}, \ \vec{b} = \begin{bmatrix} -7 \\ 7 \\ 4 \end{bmatrix}, \ \vec{c} = \begin{bmatrix} 2 \\ -3 \\ 3 \end{bmatrix}$$

Osnovne ideje

- Pretpostavimo, za sada, da linearni program koji rešavamo obavezno ima rešenje koje zadovoljava sve nejednakosti.
- U svakoj iteraciji algoritma simpleksa, trenutno rešenje *postavlja sve nebazične promenljive na* 0!
- Svaka iteracija vrši jedno *pivotiranje*—zamenu mesta jedne bazične i nebazične promenljive.
- Ovo se ponavlja sve dok je moguće uvećati funkciju koju želimo maksimizirati.

Primer

- Ilustrovaćemo algoritam simpleksa na sledećem linearnom programu (koji je već u standardnoj formi):

$$
\begin{array}{lrcrcrcr}
\text{maksimizirati} & 3x_1 & + & x_2 & + & 2x_3 & & \\
\text{pod uslovima} & & & & & & & \\
 & x_1 & + & x_2 & + & 3x_3 & \leq & 30 \\
 & 2x_1 & + & 2x_2 & + & 5x_3 & \leq & 24 \\
 & 4x_1 & + & x_2 & + & 2x_3 & \leq & 36 \\
 & & & x_1, x_2, x_3 & & & \geq & 0
\end{array}
$$

Primer: pretvaranje u *slack* formu

- Pretvaranjem linearnog programa u *slack* formu dobijamo:

maksimizirati $3x_1 + x_2 + 2x_3$

pod uslovima

$$\begin{array}{rcrcrcrcr} x_4 & = & 30 & - & x_1 & - & x_2 & - & 3x_3 \\ x_5 & = & 24 & - & 2x_1 & - & 2x_2 & - & 5x_3 \\ x_6 & = & 36 & - & 4x_1 & - & x_2 & - & 2x_3 \\ & & \multicolumn{5}{c}{x_1, x_2, x_3, x_4, x_5, x_6} & \geq & 0 \end{array}$$

Primer: iteracija 1

- *Podsetnik*: Algoritam simpleksa svaki put uzima rešenje koje postavlja sve nebazične promenljive na 0.

$$
\begin{array}{lrcrcrcrcr}
\text{maksimizirati} & & & & & 3x_1 & + & x_2 & + & 2x_3 \\
\text{pod uslovima} & & & & & & & & & \\
& x_4 & = & 30 & - & x_1 & - & x_2 & - & 3x_3 \\
& x_5 & = & 24 & - & 2x_1 & - & 2x_2 & - & 5x_3 \\
& x_6 & = & 36 & - & 4x_1 & - & x_2 & - & 2x_3 \\
& & & \multicolumn{5}{c}{x_1, x_2, x_3, x_4, x_5, x_6} & \geq & 0
\end{array}
$$

$$(x_1, x_2, x_3, x_4, x_5, x_6) = (0, 0, 0, 30, 24, 36), \ val = 0$$

Primer: iteracija 1

- U svakoj iteraciji, pretvaramo jednu nebazičnu promenljivu u bazičnu—ovo treba da bude promenljiva čije će povećanje uvećati funkciju koju maksimiziramo.

$$
\begin{array}{lrcrcrcrcr}
\text{maksimizirati} & & & & & 3x_1 & + & x_2 & + & 2x_3 \\
\text{pod uslovima} & & & & & & & & & \\
& x_4 & = & 30 & - & x_1 & - & x_2 & - & 3x_3 \\
& x_5 & = & 24 & - & 2x_1 & - & 2x_2 & - & 5x_3 \\
& x_6 & = & 36 & - & 4x_1 & - & x_2 & - & 2x_3 \\
& & & \multicolumn{5}{r}{x_1, x_2, x_3, x_4, x_5, x_6} & \geq & 0
\end{array}
$$

- Sve tri nebazične promenljive bi uvećale funkciju (jer imaju pozitivan koeficijent u njoj). Odabraćemo x_1.

Primer: iteracija 1

- Zamenu vršimo sa onom bazičnom promenljivom čiji uslov najviše *ograničava* vrednost x_1.

$$
\begin{array}{lrcrcrcrcr}
\text{maksimizirati} & & & & & 3x_1 & + & x_2 & + & 2x_3 \\
\text{pod uslovima} & & & & & & & & & \\
 & x_4 & = & 30 & - & x_1 & - & x_2 & - & 3x_3 \\
 & x_5 & = & 24 & - & 2x_1 & - & 2x_2 & - & 5x_3 \\
 & x_6 & = & 36 & - & 4x_1 & - & x_2 & - & 2x_3 \\
 & & & \multicolumn{5}{c}{x_1, x_2, x_3, x_4, x_5, x_6} & \geq & 0
\end{array}
$$

$$x_4 : x_1 \leq 30$$

$$x_5 : x_1 \leq 12$$

$$\boxed{x_6 : x_1 \leq 9}$$

Primer: iteracija 1

- Odabrali smo, dakle, zamenu x_1 sa x_6.

$$
\begin{array}{lrcrcrcrcr}
\text{maksimizirati} & & & & & 3x_1 & + & x_2 & + & 2x_3 \\
\text{pod uslovima} & & & & & & & & & \\
& x_4 & = & 30 & - & x_1 & - & x_2 & - & 3x_3 \\
& x_5 & = & 24 & - & 2x_1 & - & 2x_2 & - & 5x_3 \\
& x_6 & = & 36 & - & 4x_1 & - & x_2 & - & 2x_3 \\
& & & \multicolumn{3}{c}{x_1, x_2, x_3, x_4, x_5, x_6} & & & \geq & 0
\end{array}
$$

$$x_1 = 9 - \frac{1}{4}x_2 - \frac{1}{2}x_3 - \frac{1}{4}x_6$$

Primer: iteracija 1

- Posle smene, linearni program izgleda ovako:

$$
\begin{array}{lrcrcrcrcr}
\text{maksimizirati} & & & 27 & + & \tfrac{1}{4}x_2 & + & \tfrac{1}{2}x_3 & - & \tfrac{3}{4}x_6 \\
\text{pod uslovima} & & & & & & & & & \\
 & x_1 & = & 9 & - & \tfrac{1}{4}x_2 & - & \tfrac{1}{2}x_3 & - & \tfrac{1}{4}x_6 \\
 & x_4 & = & 21 & - & \tfrac{3}{4}x_2 & - & \tfrac{5}{2}x_3 & + & \tfrac{1}{4}x_6 \\
 & x_5 & = & 6 & - & \tfrac{3}{2}x_2 & - & 4x_3 & + & \tfrac{1}{2}x_6 \\
 & & & \multicolumn{5}{c}{x_1, x_2, x_3, x_4, x_5, x_6} & \geq & 0
\end{array}
$$

$$(x_1, x_2, x_3, x_4, x_5, x_6) = (9, 0, 0, 21, 6, 0), \ val = 27$$

Primer: iteracija 2

- Sada možemo pretvoriti x_2 ili x_3 u baznu promenljivu. Odabraćemo x_2.

$$\begin{array}{lrcrcrcrcr}
\text{maksimizirati} & & & 27 & + & \frac{1}{4}x_2 & + & \frac{1}{2}x_3 & - & \frac{3}{4}x_6 \\
\text{pod uslovima} & & & & & & & & & \\
 & x_1 & = & 9 & - & \frac{1}{4}x_2 & - & \frac{1}{2}x_3 & - & \frac{1}{4}x_6 \\
 & x_4 & = & 21 & - & \frac{3}{4}x_2 & - & \frac{5}{2}x_3 & + & \frac{1}{4}x_6 \\
 & x_5 & = & 6 & - & \frac{3}{2}x_2 & - & 4x_3 & + & \frac{1}{2}x_6 \\
 & & & \multicolumn{5}{c}{x_1, x_2, x_3, x_4, x_5, x_6} & \geq & 0
\end{array}$$

$$x_1 : x_2 \leq 36$$

$$x_4 : x_2 \leq 28$$

$$\boxed{x_5 : x_2 \leq 4}$$

Primer: iteracija 2

- Odabrali smo, dakle, zamenu x_2 sa x_5.

$$
\begin{array}{lrrrrrrrr}
\text{maksimizirati} & & 27 & + & \tfrac{1}{4}x_2 & + & \tfrac{1}{2}x_3 & - & \tfrac{3}{4}x_6 \\
\text{pod uslovima} & & & & & & & & \\
x_1 & = & 9 & - & \tfrac{1}{4}x_2 & - & \tfrac{1}{2}x_3 & - & \tfrac{1}{4}x_6 \\
x_4 & = & 21 & - & \tfrac{3}{4}x_2 & - & \tfrac{5}{2}x_3 & + & \tfrac{1}{4}x_6 \\
x_5 & = & 6 & - & \tfrac{3}{2}x_2 & - & 4x_3 & + & \tfrac{1}{2}x_6 \\
 & & \multicolumn{5}{c}{x_1, x_2, x_3, x_4, x_5, x_6} & \geq & 0
\end{array}
$$

$$x_2 = 4 - \frac{8}{3}x_3 - \frac{2}{3}x_5 + \frac{1}{3}x_6$$

Primer: iteracija 2

- Posle smene, linearni program izgleda ovako:

$$
\begin{array}{lrcrcrcrcr}
\text{maksimizirati} & & & 28 & - & \frac{1}{6}x_3 & - & \frac{1}{6}x_5 & - & \frac{2}{3}x_6 \\
\text{pod uslovima} & & & & & & & & & \\
 & x_1 & = & 8 & + & \frac{1}{6}x_3 & + & \frac{1}{6}x_5 & - & \frac{1}{3}x_6 \\
 & x_2 & = & 4 & - & \frac{8}{3}x_3 & - & \frac{2}{3}x_5 & + & \frac{1}{3}x_6 \\
 & x_4 & = & 18 & - & \frac{1}{2}x_3 & + & \frac{1}{2}x_5 & & \\
 & & & \multicolumn{5}{c}{x_1, x_2, x_3, x_4, x_5, x_6} & \geq & 0
\end{array}
$$

$$\boxed{(x_1, x_2, x_3, x_4, x_5, x_6) = (8, 4, 0, 18, 0, 0),\ val = 28}$$

Implementacija algoritma simpleksa

- Algoritam simpleksa možda deluje jako intuitivno, ali njegova implementacija krije mnoge komplikovane detalje.

- Najpre, definisaćemo podrutinu za pivotiranje (smenu jedne bazične i nebazične promenljive).

Pivotiranje

Pivot$(N, B, \mathbf{A}, \vec{b}, \vec{c}, l, e)$ // pivotiraj (nebazičnu) x_e oko (bazične) x_l

$\widehat{\mathbf{A}} \leftarrow \mathbf{0}_{|B|\times|N|}$ // inicijalizujemo nove parametre
$\widehat{b} \leftarrow \vec{0}_{|B|}$
$\widehat{c} \leftarrow \vec{0}_{|N|}$
$\widehat{b}_e \leftarrow b_l / \mathbf{A}_{le}$ // određujemo koeficijente za smenu x_e sa x_l
for all $j \in N \setminus \{e\}$
  $\widehat{\mathbf{A}}_{ej} \leftarrow \mathbf{A}_{lj} / \mathbf{A}_{le}$
$\widehat{\mathbf{A}}_{el} \leftarrow 1 / \mathbf{A}_{le}$
for all $i \in B \setminus \{l\}$ // određujemo koeficijente za ostale uslove
  $\widehat{b}_i \leftarrow b_i - \mathbf{A}_{ie}\widehat{b}_e$
  for all $j \in N \setminus \{e\}$
    $\widehat{\mathbf{A}}_{ij} \leftarrow \mathbf{A}_{ij} - \mathbf{A}_{ie}\widehat{\mathbf{A}}_{ej}$
  $\widehat{\mathbf{A}}_{il} \leftarrow -\mathbf{A}_{ie}\widehat{\mathbf{A}}_{el}$

Pivotiranje, *cont'd*

```
for all j ∈ N \ {e}            // određujemo koeficijente u funkciji
    ĉ_j ← c_j − c_e Â_ej
ĉ_l ← −c_e Â_el
N̂ ← N \ {e} ∪ {l}              // određujemo nove skupove promenljivih
B̂ ← B \ {l} ∪ {e}
return (N̂, B̂, Â, b̂, ĉ)
```

for all $j \in N \setminus \{e\}$ // određujemo koeficijente u funkciji

$\widehat{c}_j \leftarrow c_j - c_e \widehat{\mathbf{A}}_{ej}$

$\widehat{c}_l \leftarrow -c_e \widehat{\mathbf{A}}_{el}$

$\widehat{N} \leftarrow N \setminus \{e\} \cup \{l\}$ // određujemo nove skupove promenljivih

$\widehat{B} \leftarrow B \setminus \{l\} \cup \{e\}$

return $(\widehat{N}, \widehat{B}, \widehat{\mathbf{A}}, \widehat{b}, \widehat{c})$

Iteracija algoritma simpleksa

- Sa gotovom metodom za pivotiranje, možemo napraviti i implementaciju osnovne petlje unutar algoritma simpleksa.
- Dokle god možemo, odabiramo jednu nebazičnu promenljivu koja bi poboljšala funkciju koju optimizujemo.
- Zatim je pivotiramo oko one bazične promenljive koja joj zadaje najstriktniji uslov.

Petlja algoritma simpleksa

$\textsc{Simplex-Loop}(N, B, \mathbf{A}, \vec{b}, \vec{c})$

while $\exists j \in N . c_j > 0$
  choose $e \in N$ such that $c_e > 0$
  $\vec{\Delta} \leftarrow \vec{0}_{|B|}$
  for all $i \in B$
    if $\mathbf{A}_{ie} > 0$
      $\Delta_i \leftarrow b_i / \mathbf{A}_{ie}$
    else $\Delta_i \leftarrow +\infty$
  $l \leftarrow \operatorname*{argmin}_{i \in B} \Delta_i$
  if $\Delta_l = +\infty$ // promenljiva x_e nije ograničena
    error "unbounded"
  else $(N, B, \mathbf{A}, \vec{b}, \vec{c}) \leftarrow \textsc{Pivot}(N, B, \mathbf{A}, \vec{b}, \vec{c}, l, e)$
return $(N, B, \mathbf{A}, \vec{b}, \vec{c})$

Začkoljice

- Kako odabrati promenljivu x_e?
  - Ukoliko važi $\Delta_l = 0$, onda pivotiranjem nećemo postići povećanje funkcije koju optimiziramo; u najgorem slučaju može doći do beskonačne petlje!
  - Neke strategije za izbegavanje ciklusa:

Blandovo pravilo. Odabrati onu promenljivu sa najmanjim indeksom e.
Nasumično pravilo. Odabrati nasumično (sa uniformnom verovatnoćom).
Perturbacija. Neznatno promeniti parametre ulaza; $b_i \leftarrow b_i + \varepsilon_i, \varepsilon_i \gg \varepsilon_{i+1}$.

- Kako odrediti da li je moguće zadovoljiti sve nejednakosti linearnog programa?

- Šta ako je moguće zadovoljiti ove uslove, ali nije moguće postaviti sve nebazične promenljive na 0 na početku?

Algoritam simpleksa

- Pretpostavićemo da posedujemo funkciju Initialise-Simplex, koja nas ili obaveštava da linearan program nema rešenje, ili nam daje *slack* formu ekvivalentnu početnoj, u kojoj možemo postaviti sve nebazične promenljive na 0.

Simplex$(\mathbf{A}, \vec{b}, \vec{c})$

$(N, B, \mathbf{A}, \vec{b}, \vec{c}) \leftarrow$ Initialise-Simplex$(\mathbf{A}, \vec{b}, \vec{c})$
$(N, B, \mathbf{A}, \vec{b}, \vec{c}) \leftarrow$ Simplex-Loop$(N, B, \mathbf{A}, \vec{b}, \vec{c})$
for $i \leftarrow 1$ **to** n
    if $i \in B$
        $x_i \leftarrow b_i$
    else $x_i \leftarrow 0$
return $\vec{x}$

Inicijalizacija

- Ostalo je "još samo" da implementiramo metodu INITIALISE-SIMPLEX.
- Ako je bilo koji element vektora $\vec{b}$ negativan, onda ne možemo odmah postaviti sve nebazične promenljive na 0! Ovo bi učinilo da neka bazična promenljiva postane negativna, što nije dozvoljeno.
- Da bismo mogli da odredimo da li uopšte postoji rešenje koje zadovoljava uslove linearnog programa, rešićemo malo izmenjen linearni program, koristeći... *algoritam simpleksa.*
- *So meta!*

Pomoćni linearni program (*auxiliary LP*)

- Pretpostavimo da imamo linearni program:

$$\begin{array}{lrcl} \text{maksimizirati} & \sum_{j=1}^{n} c_j x_j & & \\ \text{pod uslovima} & & & \\ \forall i \in \{1, 2, \dots, m\} & \sum_{j=1}^{n} \mathbf{A}_{ij} x_j & \leq & b_i \\ \forall j \in \{1, 2, \dots, n\} & x_j & \geq & 0 \end{array}$$

- *Pomoćni linearni program* traži da odredimo minimalno "rastojanje", x_0, do toga da originalni problem postane rešiv:

$$\begin{array}{lrcl} \text{maksimizirati} & -x_0 & & \\ \text{pod uslovima} & & & \\ \forall i \in \{1, 2, \dots, m\} & \sum_{j=1}^{n} \mathbf{A}_{ij} x_j - x_0 & \leq & b_i \\ \forall j \in \{0, 1, \dots, n\} & x_j & \geq & 0 \end{array}$$

Prednosti pomoćnog programa

- Ukoliko na početku pivotiramo x_0 oko one bazične promenljive x_k koja ima najmanju vrednost b_k, dobili smo linearni program u kome možemo postaviti sve nebazične promenljive na 0! (svi elementi $\vec{b}$ se ovim pivotiranjem uvećavaju za b_k).
- Ovaj linearni program onda možemo rešiti metodom Simplex-Loop!
- Linearni program nije rešiv *ako i samo ako* dobijemo $x_0 > 0$.
- U suprotnom, ako izbacimo x_0 iz uslova i vratimo početnu objektivnu funkciju, dobili smo ekvivalentnu formu u kojoj možemo postaviti sve nebazične promenljive na 0!

Implementacija inicijalizacije

$\textsc{Initialise-Simplex}(\mathbf{A}, \vec{b}, \vec{c})$

$k \leftarrow \operatorname*{argmin}_i b_i$

if $b_k \geq 0$ // odmah možemo postaviti nebazične promenljive na 0

    return $(\{1, \dots n\}, \{n+1, \dots, n+m\}, \mathbf{A}, \vec{b}, \vec{c})$

form $L_{aux} = (N_{aux}, B_{aux}, \mathbf{A}_{aux}, \vec{b}_{aux}, \vec{c}_{aux})$

// radimo početni pivot (smena x_0 i x_k)

$(N, B, \mathbf{A}, \vec{b}, \vec{c}) \leftarrow \textsc{Pivot}(N_{aux}, B_{aux}, \mathbf{A}_{aux}, \vec{b}_{aux}, \vec{c}_{aux}, k, 0)$

// osnovni deo algoritma simpleksa

$(N, B, \mathbf{A}, \vec{b}, \vec{c}) \leftarrow \textsc{Simplex-Loop}(N, B, \mathbf{A}, \vec{b}, \vec{c})$

Implementacija inicijalizacije, *cont'd*

if $0 \notin B \vee b_0 = 0$

  if $0 \in B$ // ako je x_0 bazična, uraditi pivot sa bilo kojim x_y

    $(N, B, \mathbf{A}, \vec{b}, \vec{c}) \leftarrow \textsc{Pivot}(N, B, \mathbf{A}, \vec{b}, \vec{c}, 0, y)$

  $N \leftarrow N \setminus \{0\}$

  restore the original objective function

  replace each basic variable in the function with its constraint

  return the resulting $(N, B, \mathbf{A}, \vec{b}, \vec{c})$

else error "infeasible"

Kraj implementacije

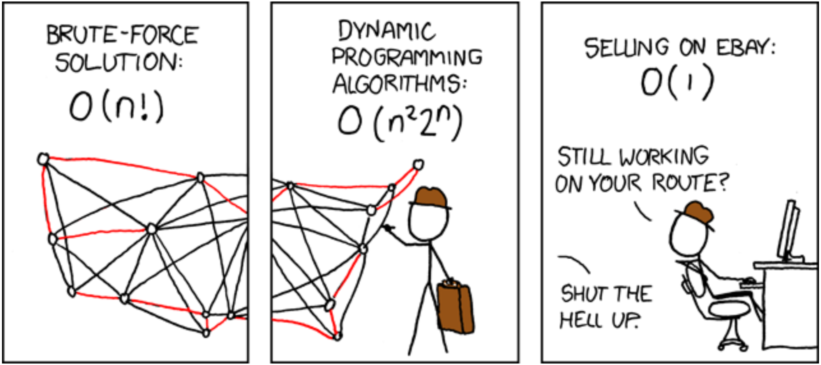
- Sa gotovom rutinom za inicijalizaciju, uspešno smo implementirali algoritam simpleksa. *Phew!*

- Kompletnu implementaciju algoritma simpleksa (u C++) možete naći na sledećem linku: `https://github.com/PetarV-/Algorithms/blob/master/Mathematical%20Algorithms/Simplex%20Algorithm.cpp`

Problem putujućeg trgovca

- Na kraju, osvrnućemo se na jednu, neočekivanu, primenu algoritma simpleksa.
- Verovatno ste do sada već čuli za **problem putujućeg trgovca** (*Traveling Salesman Problem (TSP)*).
- Ovaj problem je verovatno najpopularniji NP-kompletan problem, što znači da je verovatno da ne postoji algoritam koji ga "efikasno" (u polinomnom vremenu) rešava.
  - Čak je i snimljen film o potencijalnim posledicama uspešnog rešavanja ovog problema (*Travelling Salesman, 2012.*)
- Najbolji poznati algoritam zahteva vremensku složenost $O(n^2 2^n)$.

Travelling Salesman Problem (http://xkcd.com/399/)



TSP

- *Problem*: Za zadati graf $G = (V, E)$ sa nenegativnim cenama c_{uv} za svaku ivicu $u \to v \in E$, odrediti *ciklus koji posećuje sve čvorove i ima minimalnu ukupnu cenu.*

- Ovaj problem je očigledno *linearan*, tako da možemo probati da ga rešimo algoritmom simpleksa!

- Ovo su prvi put, još 1954. godine, pokušali Dancig, Fulkerson i Džonson.

LP formulacija

- Koristićemo *indikatorske promenljive* x_{ij}, koje su jednake 1 ukoliko je ivica $i \to j$ uključena u optimalan ciklus, a 0 inače.
- Adekvatan linearan program onda postaje:

$$
\begin{array}{lrcl}
\text{minimizirati} & \sum_{i=1}^{n} \sum_{j=1}^{i-1} c_{ij} x_{ij} & & \\
\text{pod uslovima} & & & \\
\forall i.\ 1 \leq i \leq n & \sum_{j<i} x_{ij} + \sum_{j>i} x_{ji} & = & 2 \\
\forall i, j.\ 1 \leq j < i \leq n & x_{ij} & \leq & 1 \\
\forall i, j.\ 1 \leq j < i \leq n & x_{ij} & \geq & 0
\end{array}
$$

- Ovo *namerno* nije kompletna specifikacija problema:
  - Dozvoljeno je parčanje puta na *podcikluse*.
  - Dozvoljene su “parcijalno upotrebljene ivice” ($0 < x_{ij} < 1$).

Rešenje linearnog programa

Ukoliko sa ovim, manjim, skupom uslova, algoritam pronađe korektan ciklus (bez podciklusa i bez parcijalnih ivica), onda smo uspešno rešili problem!

Dodatni uslovi: podciklusi

- Ukoliko u nađenom rešenju postoji podciklus, možemo ga eliminisati tako što dodamo novi uslov, pa ponovo pokušamo rešiti problem.
- Za podciklus koji koristi čvorove iz skupa $S \subset V$, možemo zahtevati da postoje bar dve ivice između S i $V \setminus S$:

$$\sum_{i \in S, j \in V \setminus S} x_{max(i,j),min(i,j)} \geq 2$$

- Ovakvih uslova u kompletnom problemu ima *eksponencijalno mnogo*! Međutim, često nije neophodno dodati sve uslove da bi se došlo do optimalnog rešenja.

Dodatni uslovi: parcijalne ivice

- Ukoliko u nađenom rešenju postoji parcijalno iskorišćena ivica, možemo da probamo *branch&bound* strategiju.
- Za parcijalno iskorišćenu ivicu $a \to b$, najpre dodamo uslov $x_{ab} = 1$, zatim nastavljamo dalje sa rešavanjem.
- Nakon pronađenog rešenja, brišemo sve uslove koji su naknadno dodati, dodajemo uslov $x_{ab} = 0$, i ponovo rešavamo.
- Konačno rešenje je bolje od dva nađena rešenja! Ukoliko se dovoljno pametno biraju ivice, i ovo može često da se uradi u boljoj složenosti od najgoreg (eksponencijalnog) slučaja.

Demo: čvorovi

Sada ćemo, koristeći ove tehnike, rešiti problem putujućeg trgovca za 42 najveća grada u SAD—koristeći najbolju poznatu metodu za rešavanje, ovo bi trebalo da traje ~ 4 sata!

1. Manchester, N. H.
2. Montpelier, Vt.
3. Detroit, Mich.
4. Cleveland, Ohio
5. Charleston, W. Va.
6. Louisville, Ky.
7. Indianapolis, Ind.
8. Chicago, Ill.
9. Milwaukee, Wis.
10. Minneapolis, Minn.
11. Pierre, S. D.
12. Bismarck, N. D.
13. Helena, Mont.
14. Seattle, Wash.
15. Portland, Ore.
16. Boise, Idaho
17. Salt Lake City, Utah
18. Carson City, Nev.
19. Los Angeles, Calif.
20. Phoenix, Ariz.
21. Santa Fe, N. M.
22. Denver, Colo.
23. Cheyenne, Wyo.
24. Omaha, Neb.
25. Des Moines, Iowa
26. Kansas City, Mo.
27. Topeka, Kans.
28. Oklahoma City, Okla.
29. Dallas, Tex.
30. Little Rock, Ark.
31. Memphis, Tenn.
32. Jackson, Miss.
33. New Orleans, La.
34. Birmingham, Ala.
35. Atlanta, Ga.
36. Jacksonville, Fla.
37. Columbia, S. C.
38. Raleigh, N. C.
39. Richmond, Va.
40. Washington, D. C.
41. Boston, Mass.
42. Portland, Me.

A. Baltimore, Md.
B. Wilmington, Del.
C. Philadelphia, Penn.
D. Newark, N. J.
E. New York, N. Y.
F. Hartford, Conn.
G. Providence, R. I.

Demo: matrica povezanosti

TABLE I

ROAD DISTANCES BETWEEN CITIES IN ADJUSTED UNITS

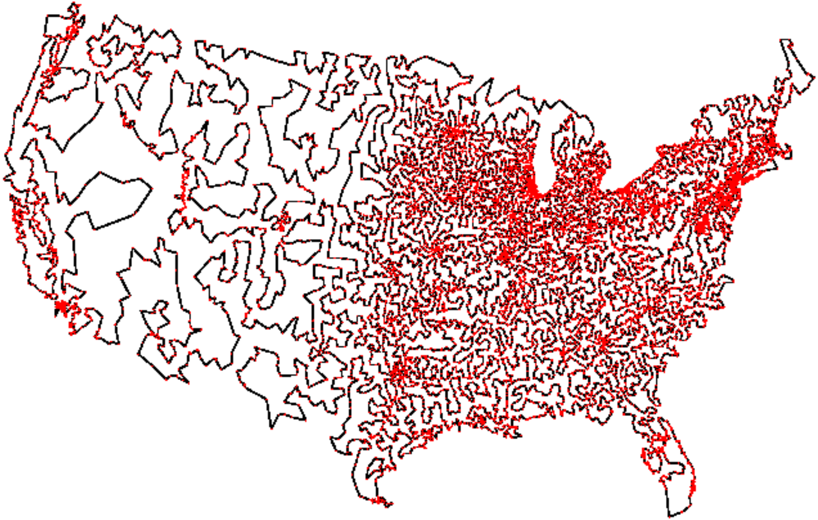
The figures in the table are mileages between the two specified numbered cities, less 11, divided by 17, and rounded to the nearest integer.

2	8																																								
3	39	45																																							
4	37	47	9																																						
5	50	49	21	15																																					
6	61	62	21	20	17																																				
7	58	60	16	17	18	6																																			
8	59	60	15	20	26	17	10																																		
9	62	66	20	25	31	22	15	5																																	
10	81	81	40	44	50	41	35	24	20																																
11	103	107	62	67	72	63	57	46	41	23																															
12	108	117	66	71	77	68	61	51	46	26	11																														
13	145	149	104	108	114	106	99	88	84	63	49	40																													
14	181	185	140	144	150	142	135	124	120	99	85	76	35																												
15	187	191	146	150	156	142	137	130	125	105	90	81	41	10																											
16	161	170	120	124	130	115	110	104	105	90	72	64	34	31	27																										
17	142	146	101	104	111	97	91	85	86	75	51	59	29	53	48	21																									
18	174	178	133	138	143	129	123	117	118	107	83	84	54	46	35	26	31																								
19	185	186	142	143	140	130	126	124	128	118	93	101	72	69	58	58	43	26																							
20	164	165	120	123	124	106	106	105	110	104	86	97	71	93	82	62	42	45	22																						
21	137	139	94	96	94	80	78	77	84	77	56	64	65	90	87	58	36	68	50	30																					
22	117	122	77	80	83	68	62	60	61	50	34	42	49	82	77	60	30	62	70	49	21																				
23	114	118	73	78	84	69	63	57	59	48	28	36	43	77	72	45	27	59	69	55	27	5																			
24	85	89	44	48	53	41	34	28	29	22	23	35	69	105	102	74	56	88	99	81	54	32	29																		
25	77	80	36	40	46	34	27	19	21	14	29	40	77	114	111	84	64	96	107	87	60	40	37	8																	
26	87	89	44	46	46	30	28	29	32	27	36	47	78	116	112	84	66	98	95	75	47	36	39	12	11																
27	91	93	48	50	48	34	32	33	36	30	34	45	77	115	110	83	63	97	91	72	44	32	36	9	15	3															
28	105	106	62	63	64	47	46	49	54	48	46	59	85	119	115	88	66	98	79	59	31	36	42	28	33	21	20														
29	111	113	69	71	66	51	53	56	61	57	59	71	96	130	126	98	75	98	85	62	38	47	53	39	42	29	30	12													
30	91	92	50	51	46	30	34	38	43	49	60	71	103	141	136	109	90	115	99	81	53	61	62	36	34	24	28	20	20												
31	83	85	42	43	38	22	26	32	36	51	63	75	106	142	140	112	93	126	108	88	60	64	66	39	36	27	31	28	28	8											
32	89	91	55	55	50	34	39	44	49	63	76	87	120	155	150	123	100	123	109	86	62	71	78	52	49	39	44	35	24	15	12										
33	95	97	64	63	56	42	49	56	60	75	86	97	126	160	155	128	104	128	113	90	67	76	82	62	59	49	53	40	29	25	23	11									
34	74	81	44	43	35	23	30	39	44	62	78	89	121	159	155	127	108	136	124	101	75	79	81	54	50	42	46	43	39	23	14	14	21								
35	67	69	42	41	31	25	32	41	46	64	83	90	130	164	160	133	114	146	134	111	85	84	86	59	52	47	51	53	49	32	24	24	30	9							
36	74	76	61	60	42	44	51	60	66	83	102	110	147	185	179	155	133	159	146	122	98	105	107	79	71	66	70	70	60	48	40	36	33	25	18						
37	57	59	46	41	25	30	36	47	52	71	93	98	136	172	172	148	126	158	147	124	121	97	99	71	65	59	63	67	62	46	38	37	43	23	13	17					
38	45	46	41	34	20	34	38	48	53	73	96	99	137	176	178	151	131	163	159	135	108	102	103	73	67	64	69	75	72	54	46	49	54	34	24	29	12				
39	35	37	35	26	18	34	36	46	51	70	93	97	134	171	176	151	129	161	163	139	118	102	101	71	65	65	70	84	78	58	50	56	62	41	32	38	21	9			
40	29	33	30	21	18	35	33	40	45	65	87	91	117	166	171	144	125	157	156	139	113	95	97	67	60	62	67	79	82	62	53	59	66	45	38	45	27	15	6		
41	3	11	41	37	47	57	55	58	63	83	105	109	147	186	188	164	144	176	182	161	134	119	116	86	78	84	88	101	108	88	80	86	92	71	64	71	54	41	32	25	
42	5	12	55	41	53	64	61	61	66	84	111	113	150	186	192	166	147	180	188	167	140	124	119	90	87	90	94	107	114	77	86	92	98	80	74	77	60	48	38	32	6
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41

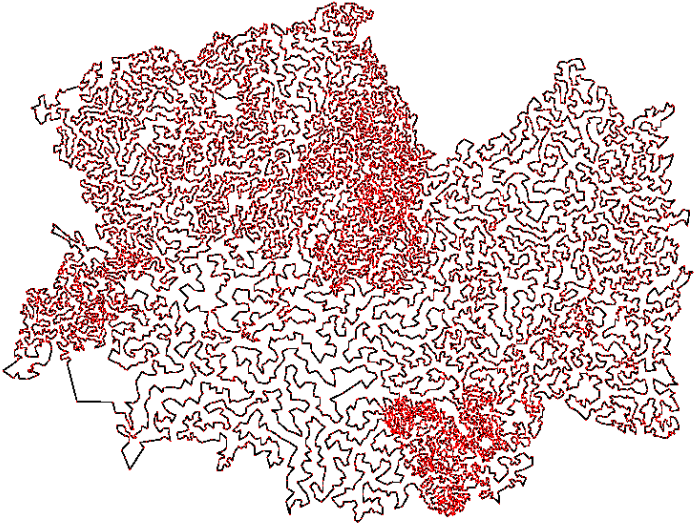
Demo: materijali

- Kompletnu implementaciju ovog TSP rešavaoca u C++ (zajedno sa fajlovima potrebnim za ovu demonstraciju) možete naći na: `https://github.com/mgcsweek/Simplex-TSP-Solver`
- Metode slične ovima su se uspešno primenjivale na rešavanje daleko većih *TSP* instanci…

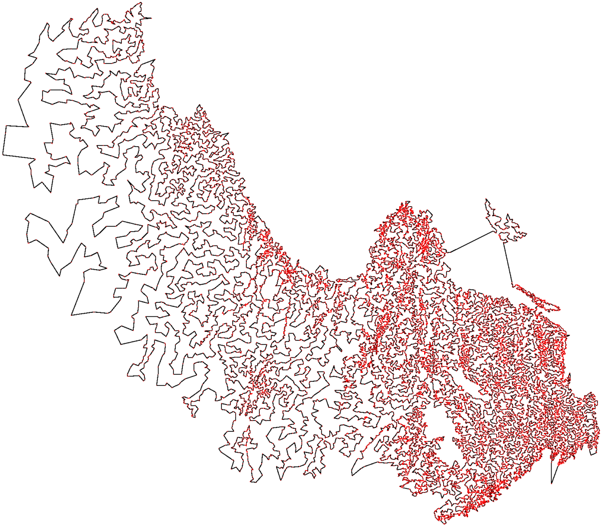
13,509 najvećih naselja u SAD



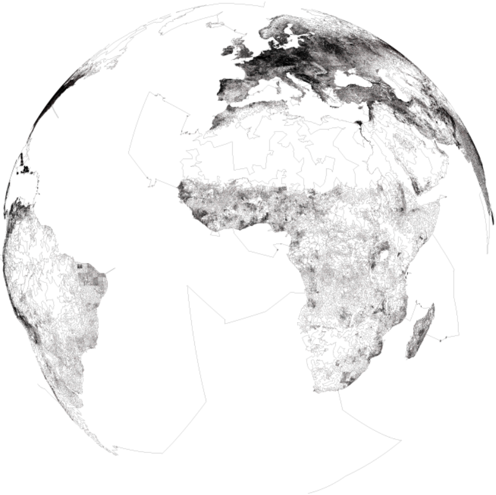
15,112 najvećih naselja u Nemačkoj



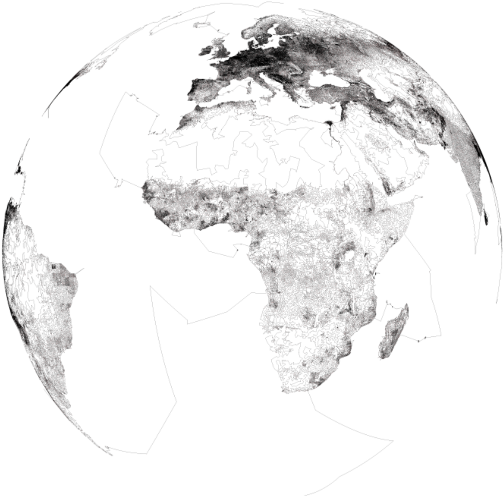
Svih 24,978 naselja u Švedskoj



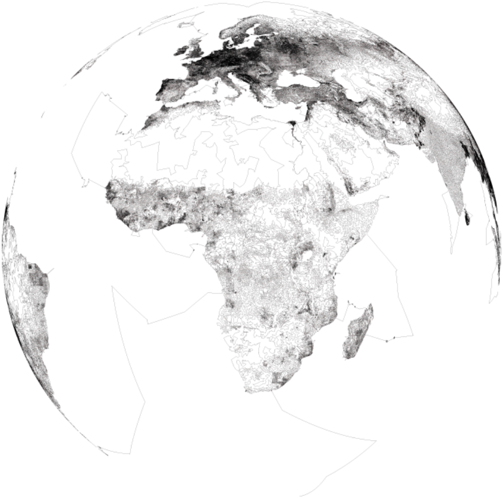
World TSP – 1,904,711 naselja, greška $\leq 0.05\%$



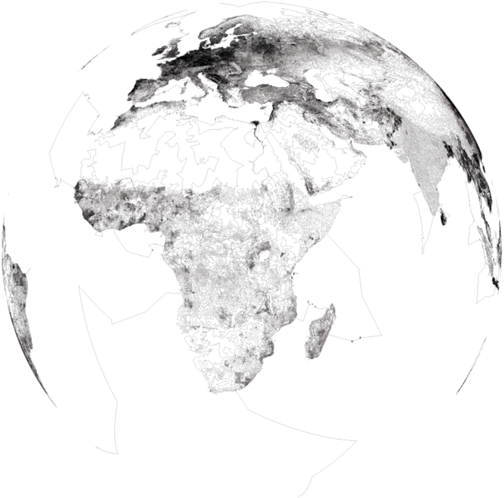
World TSP – 1,904,711 naselja, greška $\leq 0.05\%$



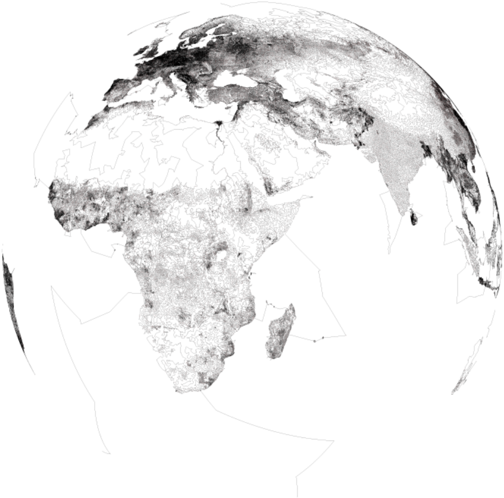
World TSP – 1,904,711 naselja, greška $\leq 0.05\%$



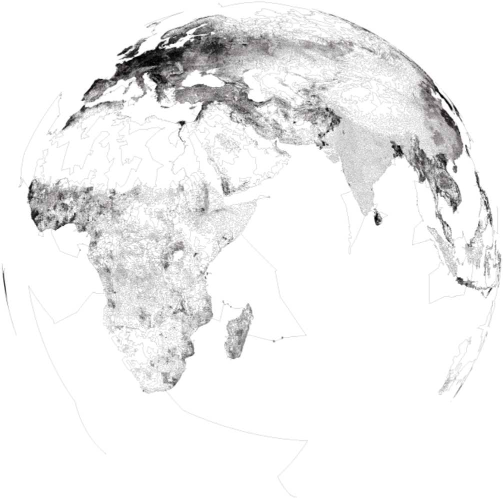
World TSP – 1,904,711 naselja, greška $\leq 0.05\%$



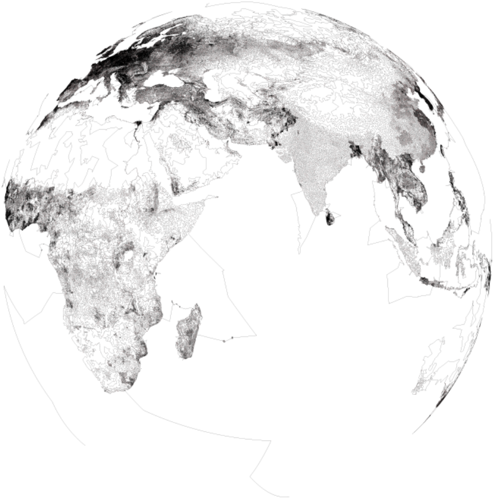
World TSP – 1,904,711 naselja, greška $\leq 0.05\%$



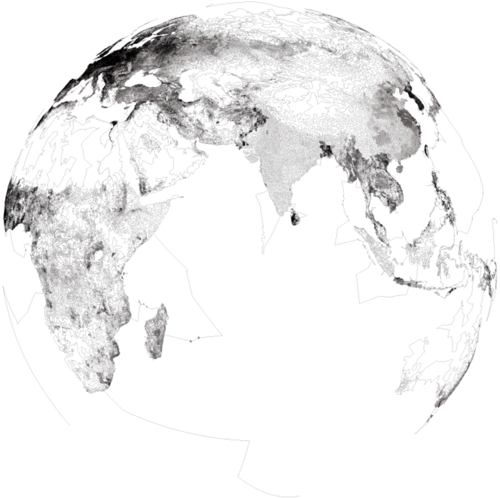
World TSP – 1,904,711 naselja, greška $\leq 0.05\%$



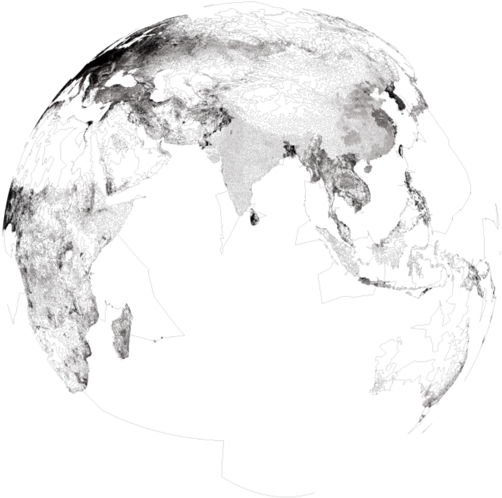
World TSP – 1,904,711 naselja, greška $\leq 0.05\%$



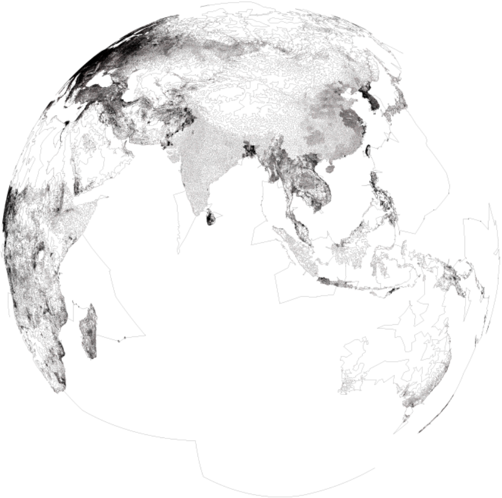
World TSP – 1,904,711 naselja, greška $\leq 0.05\%$



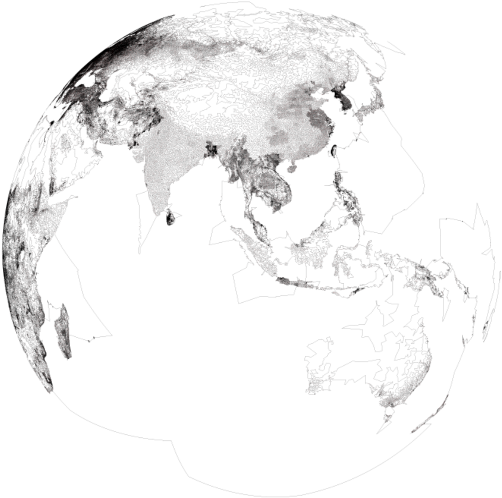
World TSP – 1,904,711 naselja, greška $\leq 0.05\%$



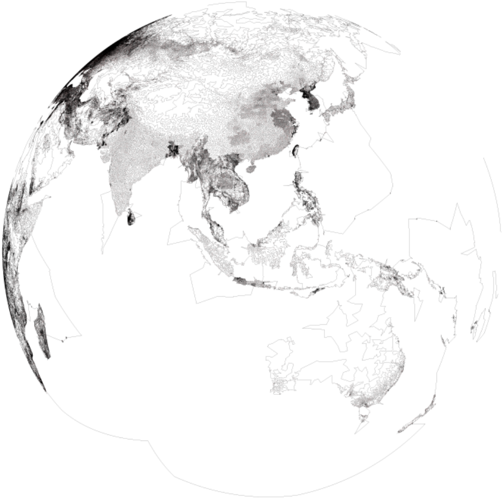
World TSP – 1,904,711 naselja, greška $\leq 0.05\%$



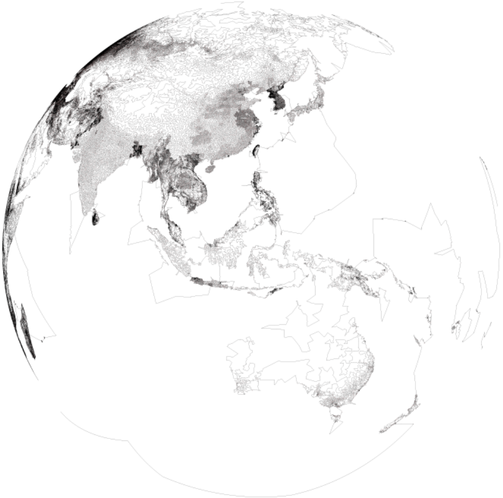
World TSP – 1,904,711 naselja, greška $\leq 0.05\%$



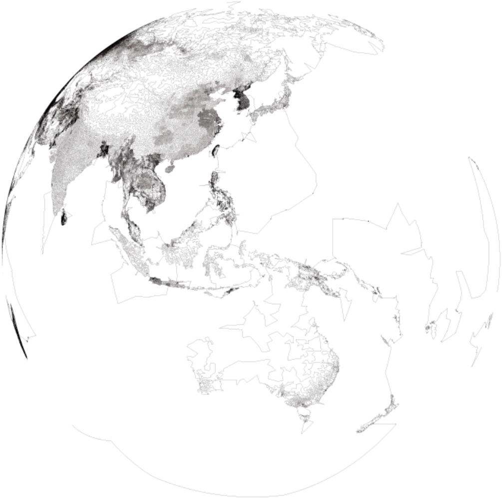
World TSP – 1,904,711 naselja, greška $\leq 0.05\%$



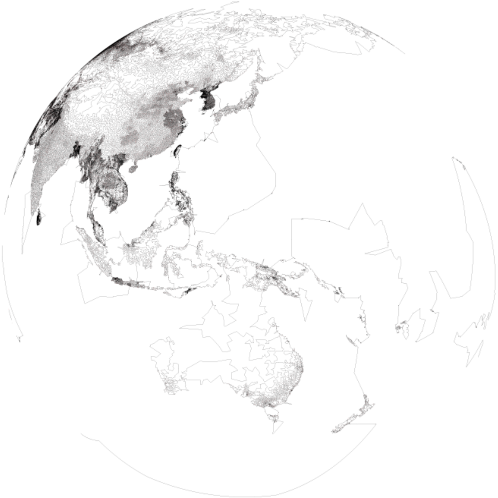
World TSP – 1,904,711 naselja, greška $\leq 0.05\%$



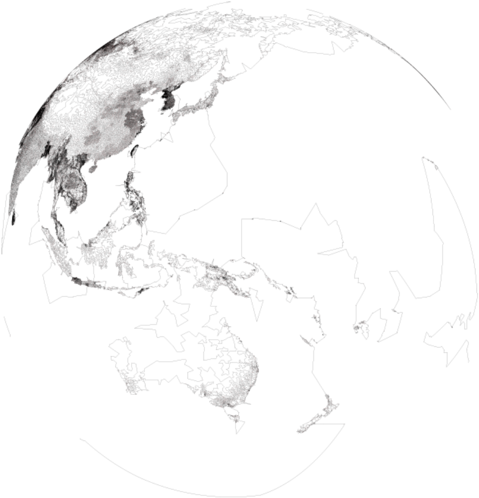
World TSP – 1,904,711 naselja, greška $\leq 0.05\%$



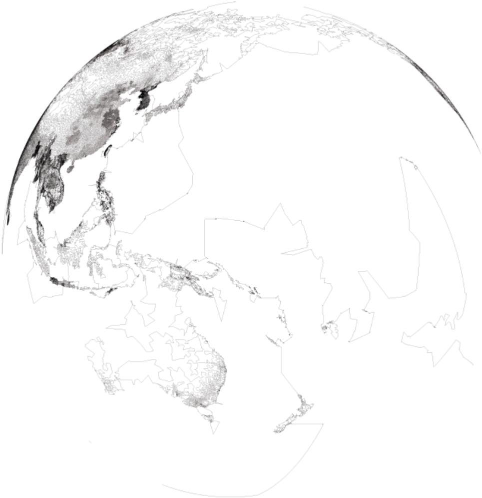
World TSP – 1,904,711 naselja, greška $\leq 0.05\%$



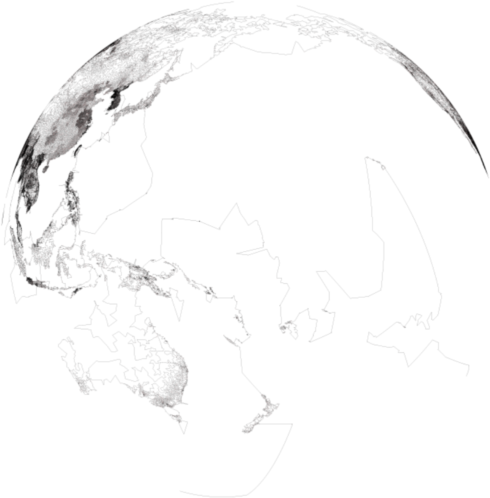
World TSP – 1,904,711 naselja, greška $\leq 0.05\%$



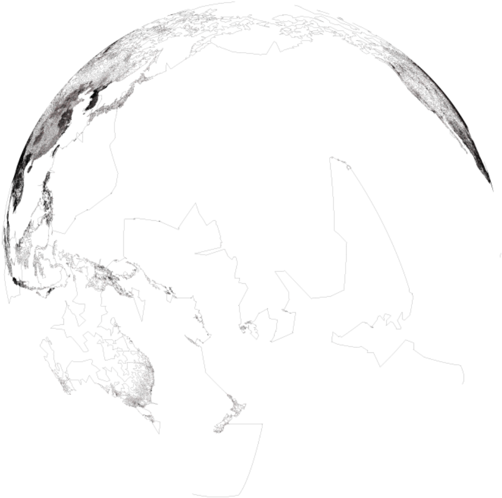
World TSP – 1,904,711 naselja, greška $\leq 0.05\%$



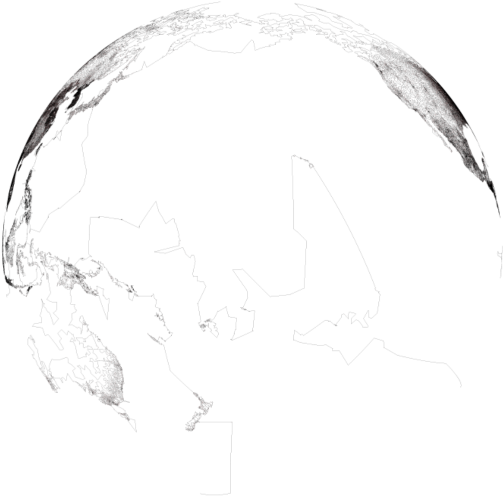
World TSP – 1,904,711 naselja, greška $\leq 0.05\%$



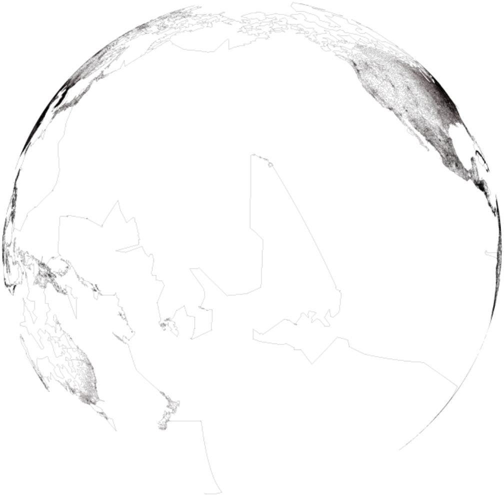
World TSP – 1,904,711 naselja, greška $\leq 0.05\%$



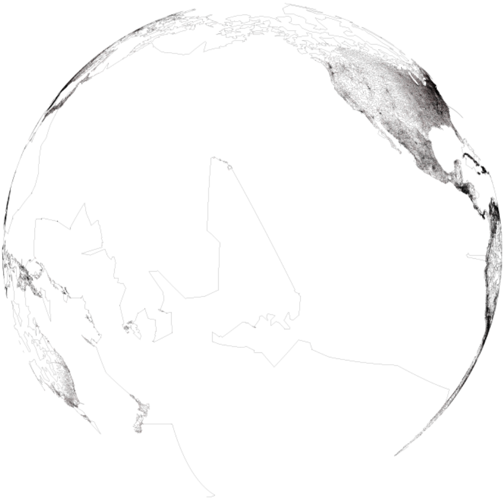
World TSP – 1,904,711 naselja, greška $\leq 0.05\%$



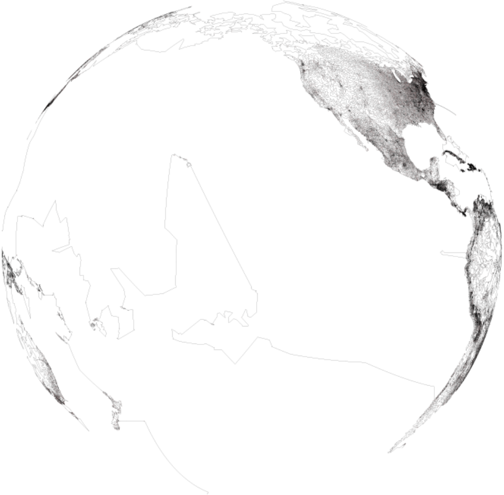
World TSP – 1,904,711 naselja, greška $\leq 0.05\%$



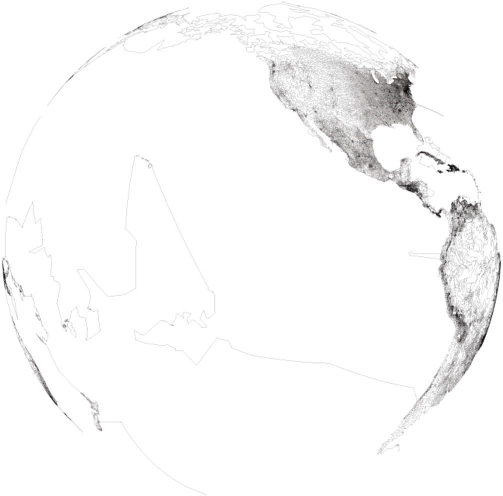
World TSP – 1,904,711 naselja, greška $\leq 0.05\%$



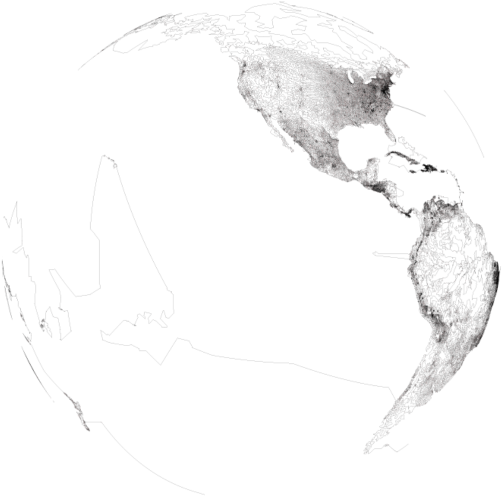
World TSP – 1,904,711 naselja, greška $\leq 0.05\%$



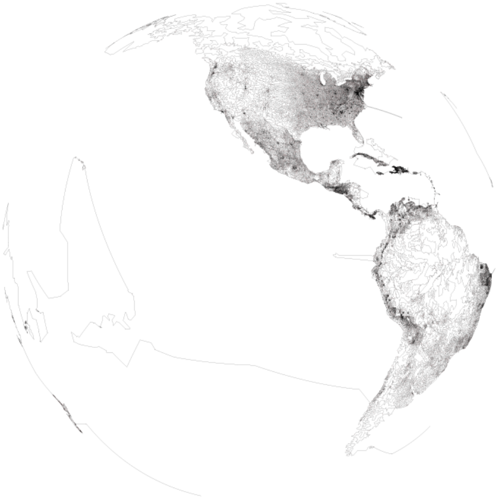
World TSP – 1,904,711 naselja, greška $\leq 0.05\%$



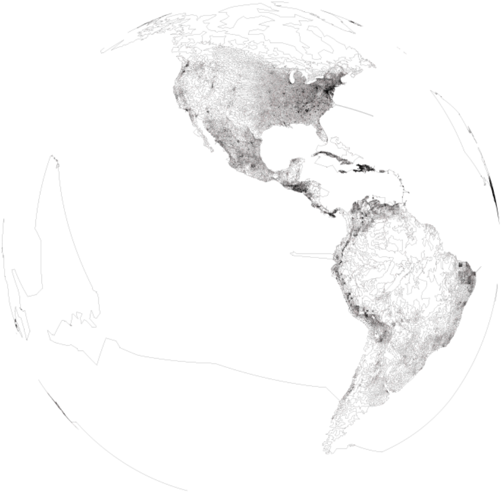
World TSP – 1,904,711 naselja, greška $\leq 0.05\%$



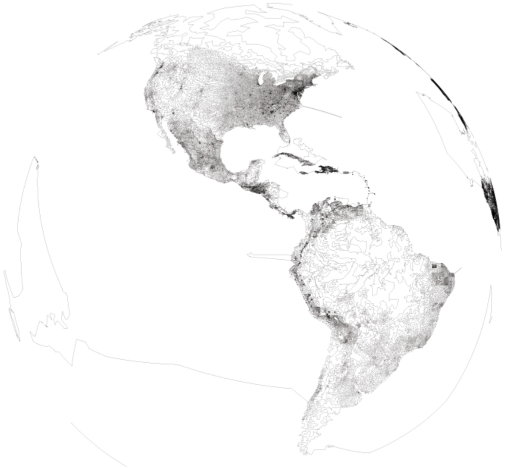
World TSP – 1,904,711 naselja, greška $\leq 0.05\%$



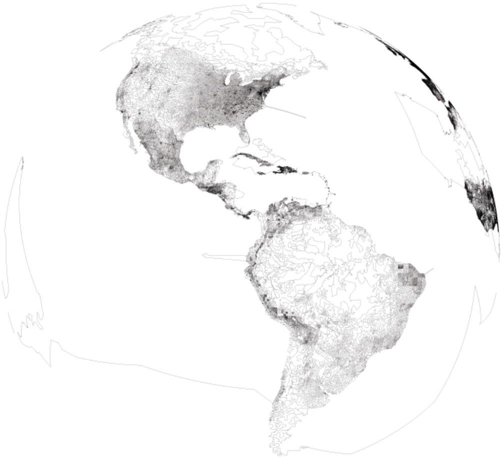
World TSP – 1,904,711 naselja, greška $\leq 0.05\%$



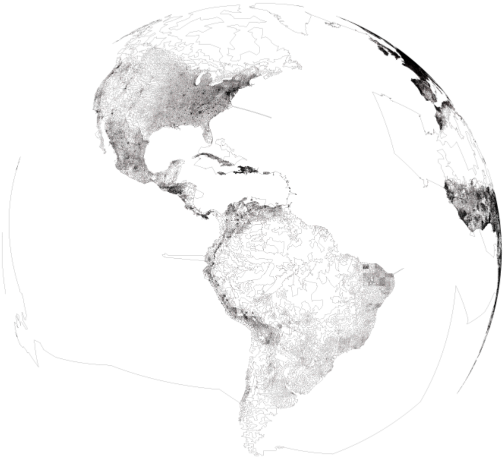
World TSP – 1,904,711 naselja, greška $\leq 0.05\%$



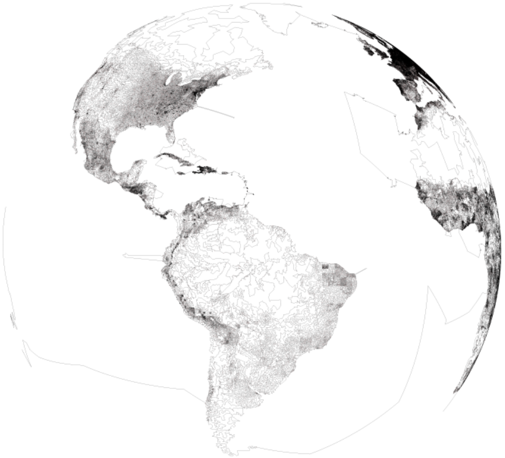
World TSP – 1,904,711 naselja, greška $\leq 0.05\%$



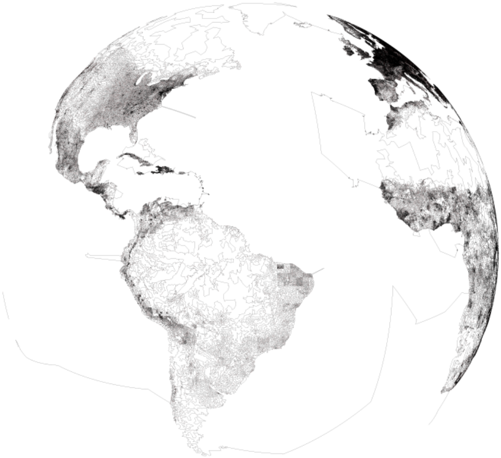
World TSP – 1,904,711 naselja, greška $\leq 0.05\%$



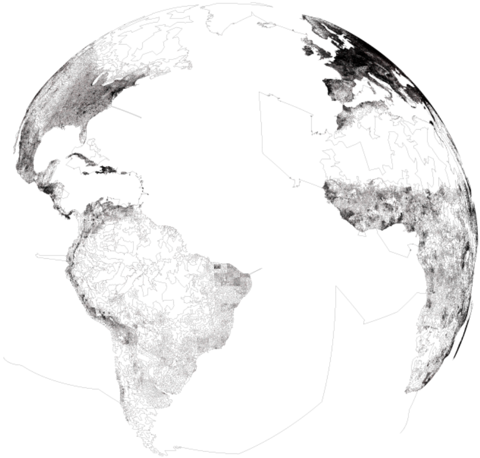
World TSP – 1,904,711 naselja, greška $\leq 0.05\%$



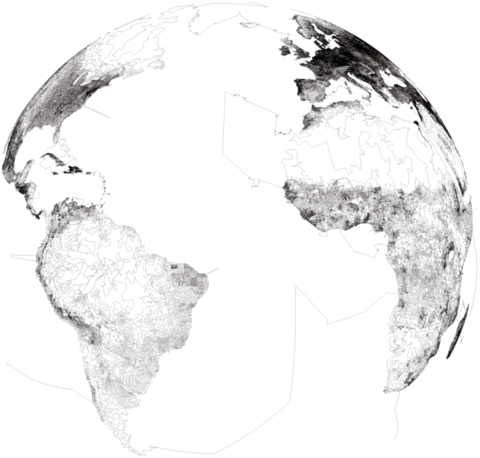
World TSP – 1,904,711 naselja, greška $\leq 0.05\%$



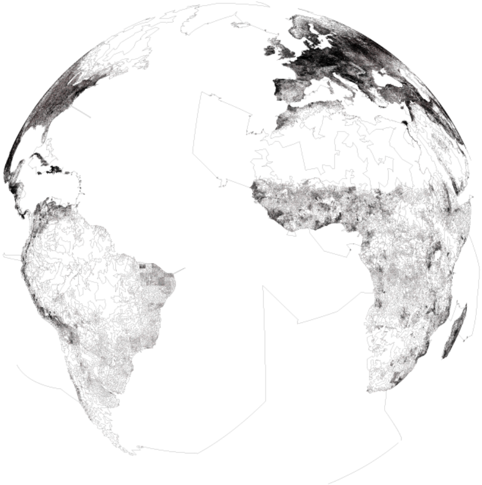
World TSP – 1,904,711 naselja, greška $\leq 0.05\%$



World TSP – 1,904,711 naselja, greška $\leq 0.05\%$



World TSP – 1,904,711 naselja, greška $\leq 0.05\%$



World TSP – 1,904,711 naselja, greška $\leq 0.05\%$