Программная реализация метода условного градиента для поиска равновесия в модели Бэкмана

А.С. Аникин
anton.anikin@htower.ru,

А.Ю. Горнов
gornov@icc.ru

Институт динамики систем и теории управления СО РАН, Иркутск

Модель транспортной сети

$\Gamma = (V, E)$ — граф сети

V — узлы сети (вершины)

$E \subset V \times V$ — дуги сети (рёбра графа)

$O \subseteq V$ — источники корреспонденций

$D \subseteq V$ — стоки

$W \subseteq \{w = (i, j) : i \in O, j \in D\}$ — множество корреспонденций

d_w — поток для корреспонденции $w \in W$

Модель транспортной сети

$p = \{v_i, v_2, ..., v_m\}$ — путь из v_1 в v_m, если $(v_k, v_{k+1}) \in E$

P_w — множество путей, отвечающих корреспонденции $w \in W$

$P = \bigcup_{w \in W} P_w$ совокупность всех путей в сети Γ

$$n = |V| \sim 10^3 - 10^4$$

$$|E| \sim 3n - 4n$$

$|P|$ — число “разумных” маршрутов,

обычно растет не быстрее, чем $O(n^3)$

Модель транспортной сети

x_p — величина потока [автомобилей/час] по пути p

$$x = \{x_p : p \in P\}$$

$f_e(x)$ — величина потока по дуге e :

$$f_e(x) = \sum_{p \in P} \delta_{ep} x_p \ , \ \delta_{ep} = \begin{cases} 1, & e \in p \\ 0, & e \notin p \end{cases}$$

Модель транспортной сети

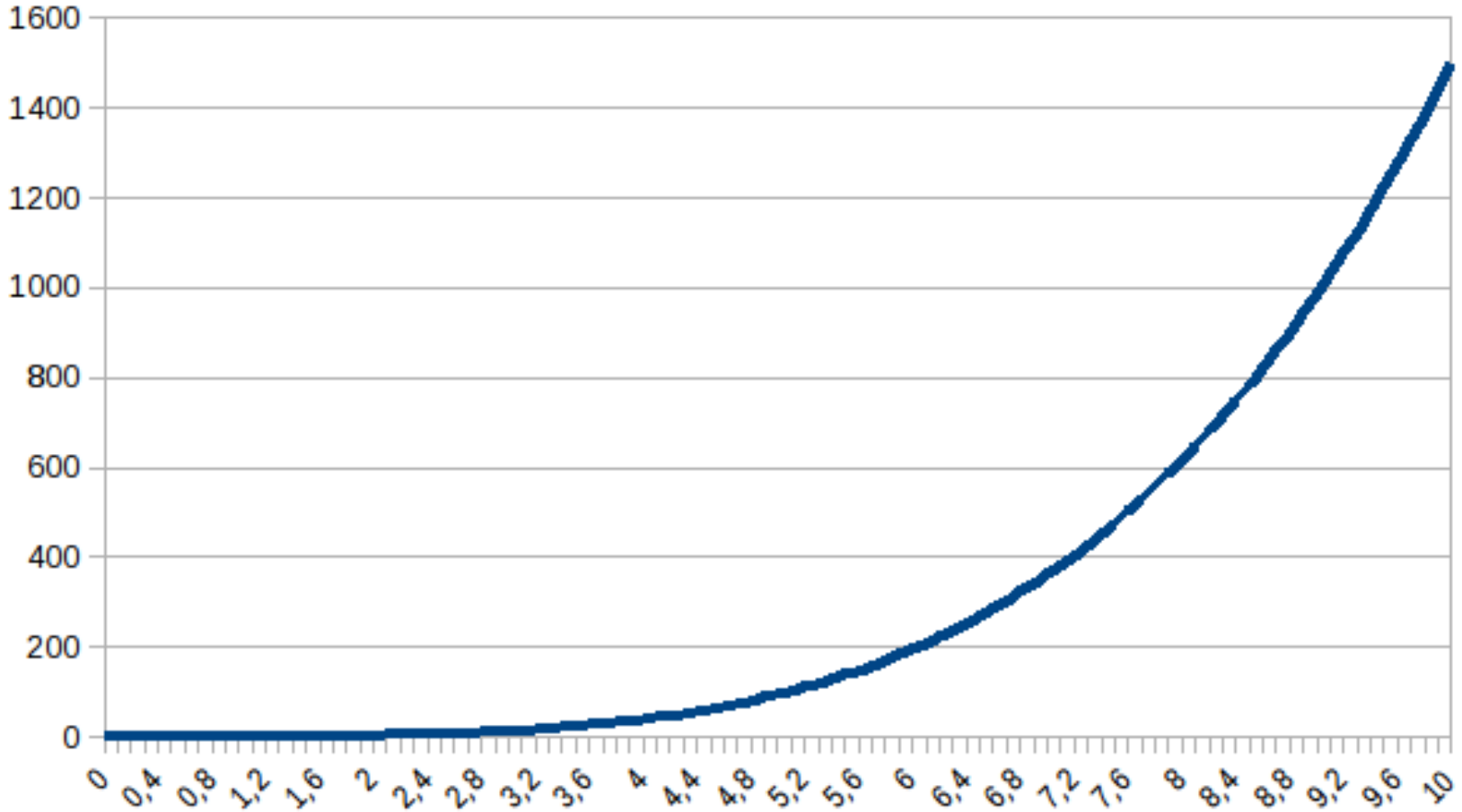
$\tau_e(f_e)$ — удельные затраты на проезд по дуге e

BPR-функция (Bureau of Public Roads) :

$$\tau_e^{\mu} = \bar{t}_e \left(1 + \gamma \left(f_e / \bar{f}_e\right)^{\frac{1}{\mu}}\right)$$

Модель транспортной сети

BPR : $\bar{t}_e = 1.0,\ \gamma = 0.15,\ \mu = 0.25,\ \bar{f}_e = 1.0$



$$\Psi(f) = \sum_{e \in E} \sigma_e(f_e) \to \min_{\substack{f = \Theta x \\ x \in X}}$$

$$\partial \Psi(f) / \partial f_e = \tau_e(f)$$

$$\sigma_e(f_e) = \int_0^{f_e(x)} \tau_e(z) dz$$

$$X = \left\{ x \geq 0 : \sum_{p \in P_w} x_p = d_w, w \in W \right\}$$

Метод условного градиента

M. Frank and P. Wolfe. An algorithm for quadratic programming. *Naval research logistics quarterly*, 3(1-2):95–110, 1956.

$$f^{k+1} = (1 - \gamma^k) f^k + \gamma^k y^k$$

$$\langle \nabla \Psi(f^k), y^k \rangle \to \min_{\substack{y^k = \Theta x \\ x \in X}}$$

$$\gamma^k = \frac{2}{k+1}, \quad k = 1, 2, ...$$

Метод условного градиента

$$\tilde{t}_e = \partial\Psi(f)/\partial f_e$$

$$\langle \nabla\Psi(f^k), y^k \rangle = \sum_{e\in E} \tilde{t}_e^k y_e^k \to \min$$

Эту задачу можно переписать, как

$$\min_{x\in X} \sum_{e\in E} \tilde{t}_e^k \sum_{p\in P} \delta_{ep} x_p = \sum_{w\in W} d_w \min_{p\in P_w} \left\{ \sum_{e\in E} \delta_{ep} \tilde{t}_e^k \right\} = \sum_{w\in W} d_w T_w(\tilde{t}^k)$$

$T_w(\tilde{t}^k)$ — длина кратчайшего пути на графе, ребра которого взвешены вектором $\tilde{t}_e^k$

Метод условного градиента

$$y^k = \sum_{w \in W} d_w s_w$$

$$s_w^i = \begin{cases} 1, & i \in p_w^s \\ 0, & i \notin p_w^s \end{cases}, i = 1, ..., |E|$$

p_w^s — кратчайший путь для корреспонденции $w \in W$

Метод условного градиента

$$f^1 = 0$$

$$\downarrow$$

$\tilde{t}^1$ (взвешиваем граф сети)

$$\downarrow$$

y^1 (ищем перераспределение потоков)

$$\downarrow$$

$$f^2 = (1 - \gamma^1) f^1 + \gamma^1 y^1$$

$$\downarrow$$

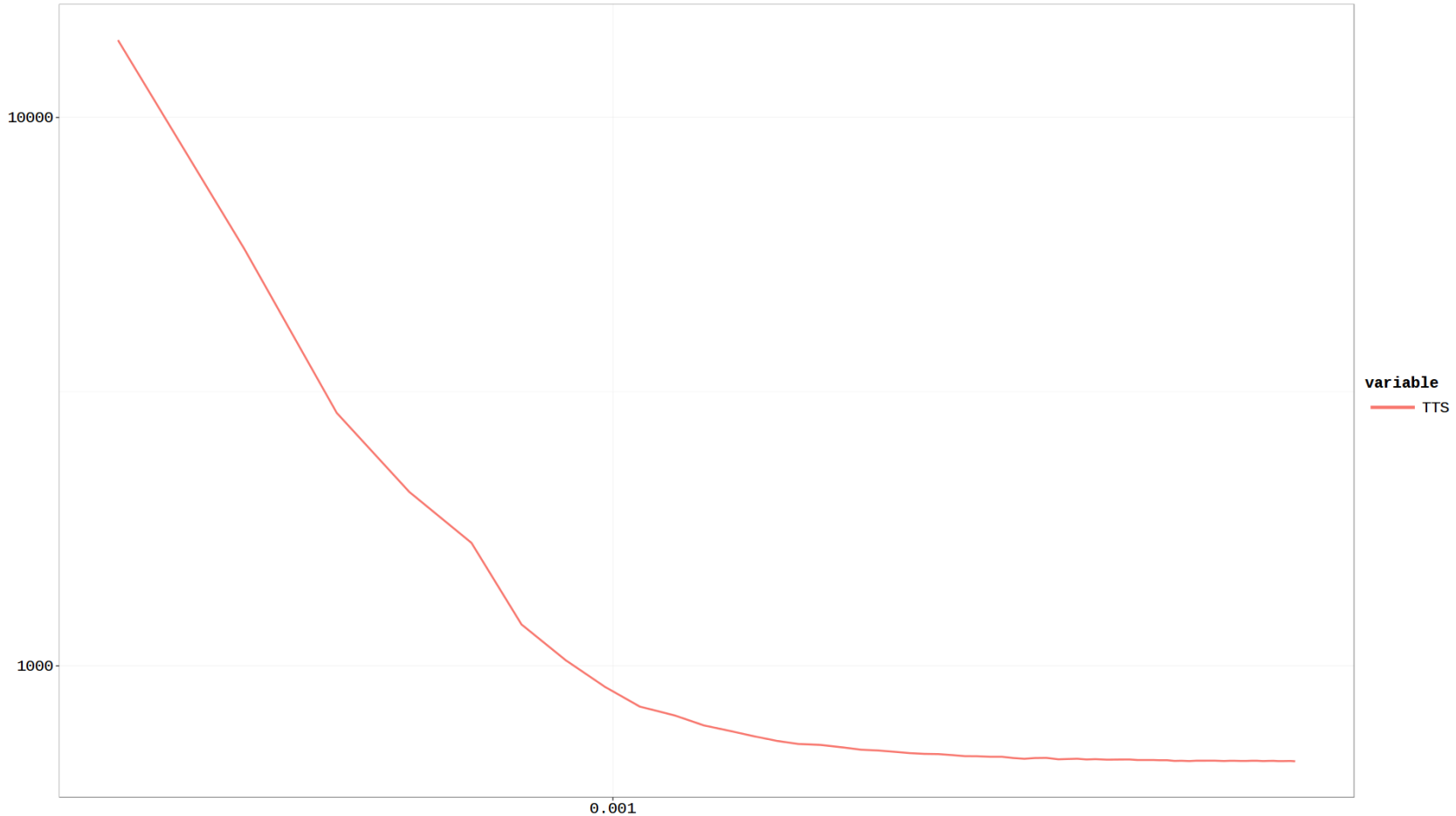
$$\dots$$

Программная реализация

- C++11
- Boost Graph Library (Dijkstra shortest paths)
- Linux / Mac OS X / Windows

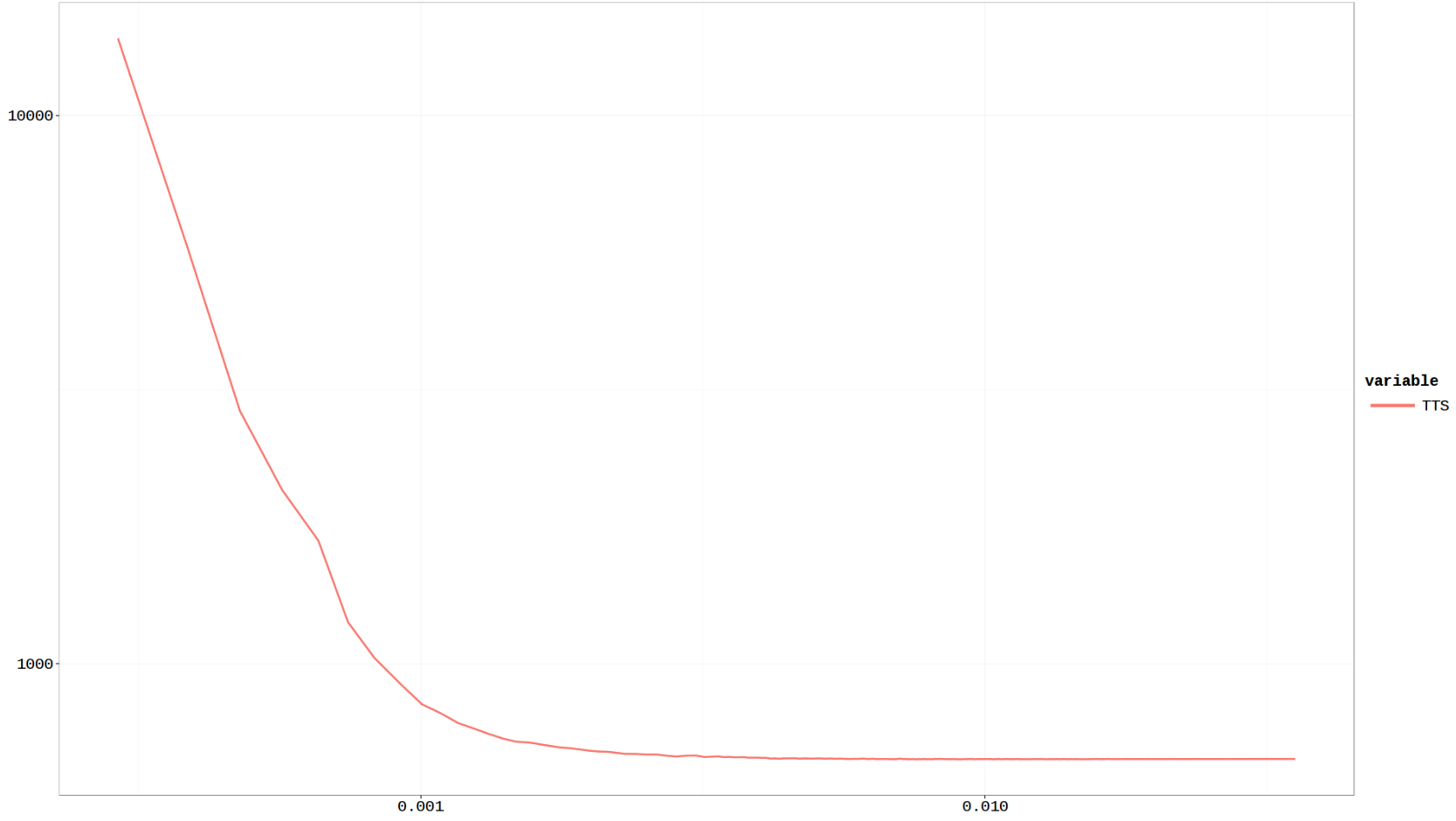
Задача : SiouxFalls

24 узла, 76 дуг, 576 корреспонденций; 66 итераций; 0.006 сек.



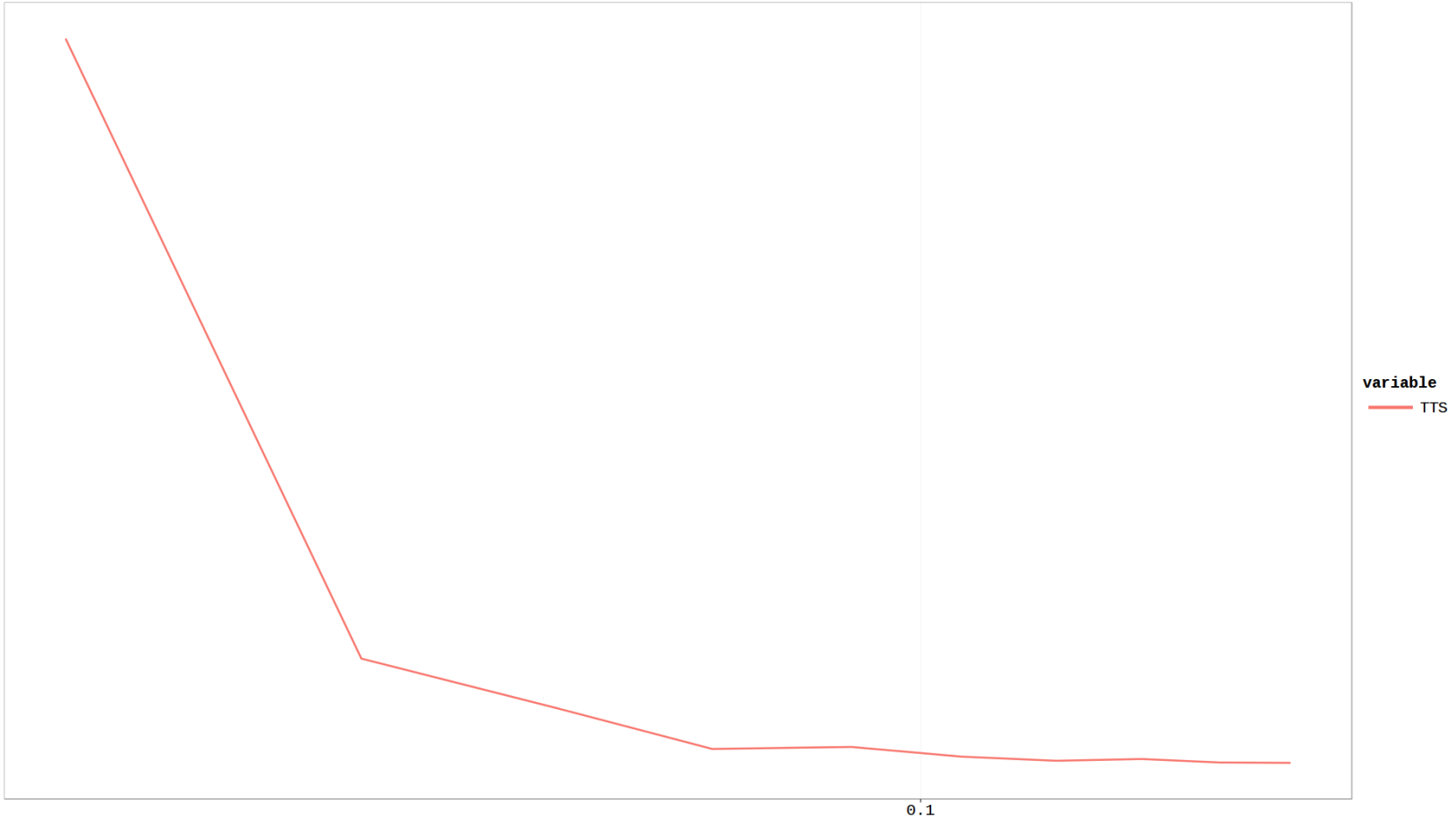
Задача : SiouxFalls

24 узла, 76 дуг, 576 корреспонденций; 500 итераций; 0.084 сек.



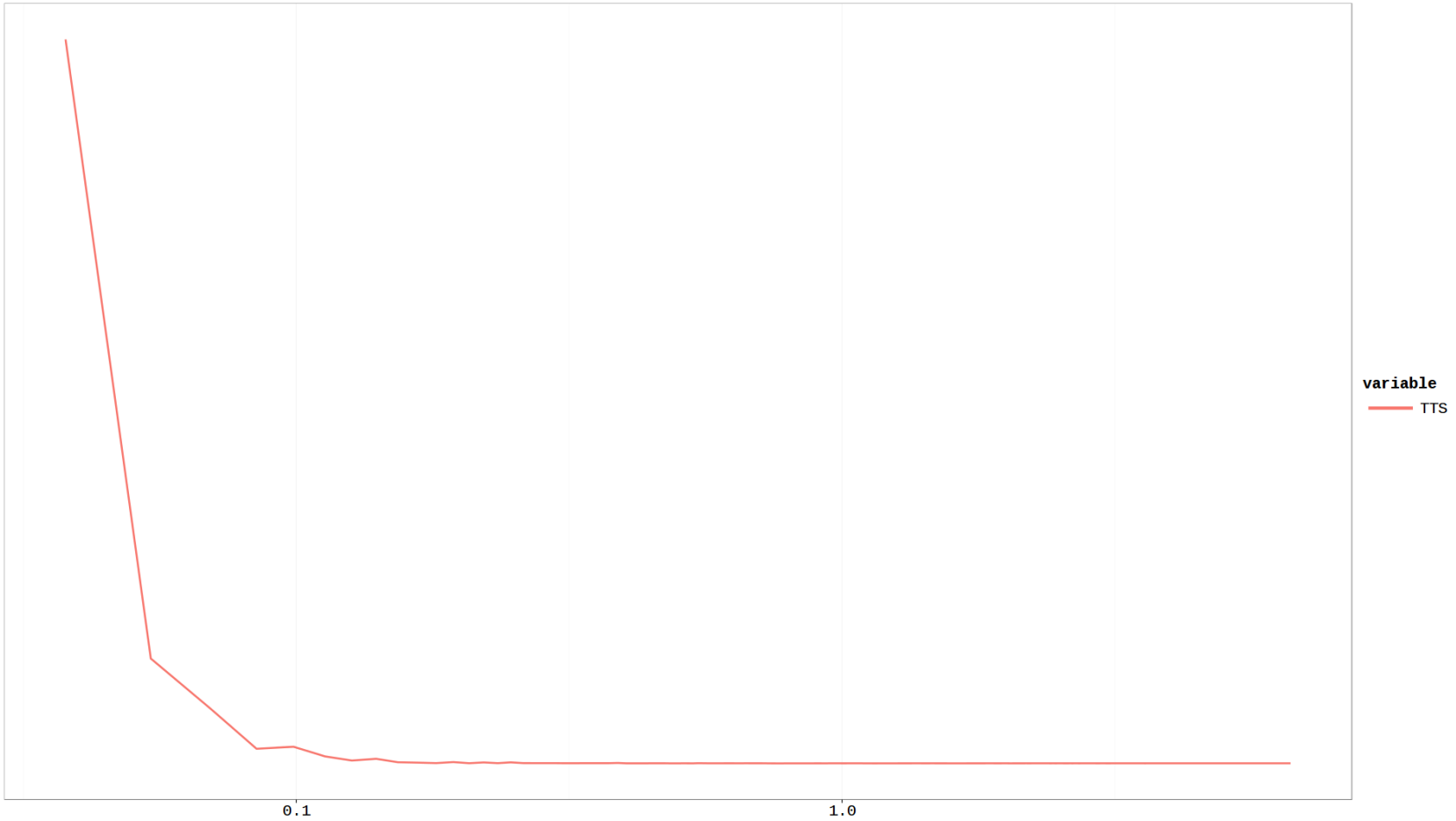
Задача : Winnipeg

1052 узла, 2836 дуг, 4345 корреспонденций; 10 итераций; 0.159 сек.



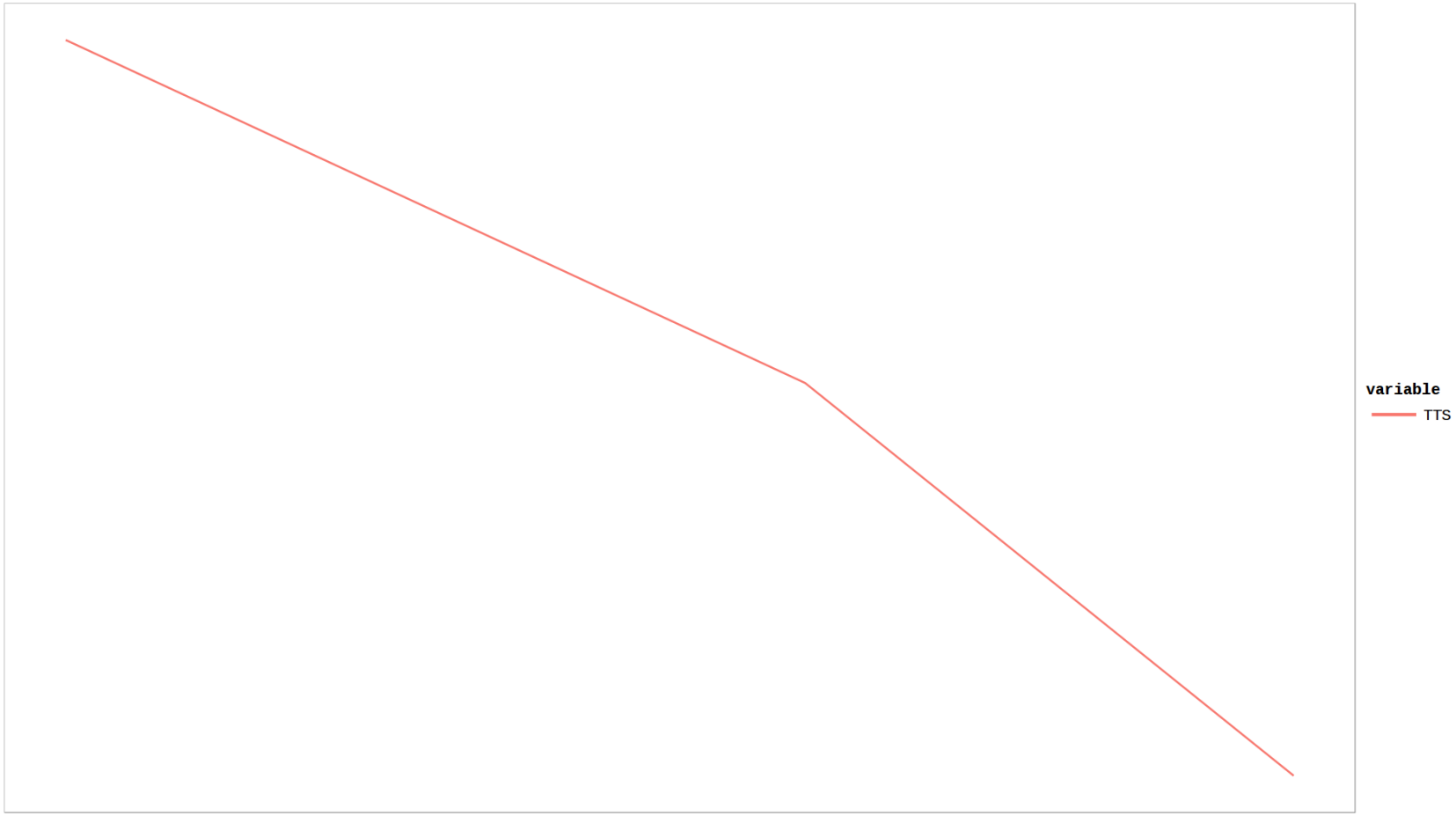
Задача : Winnipeg

1052 узла, 2836 дуг, 4345 корреспонденций; 500 итераций; 6.634 сек.



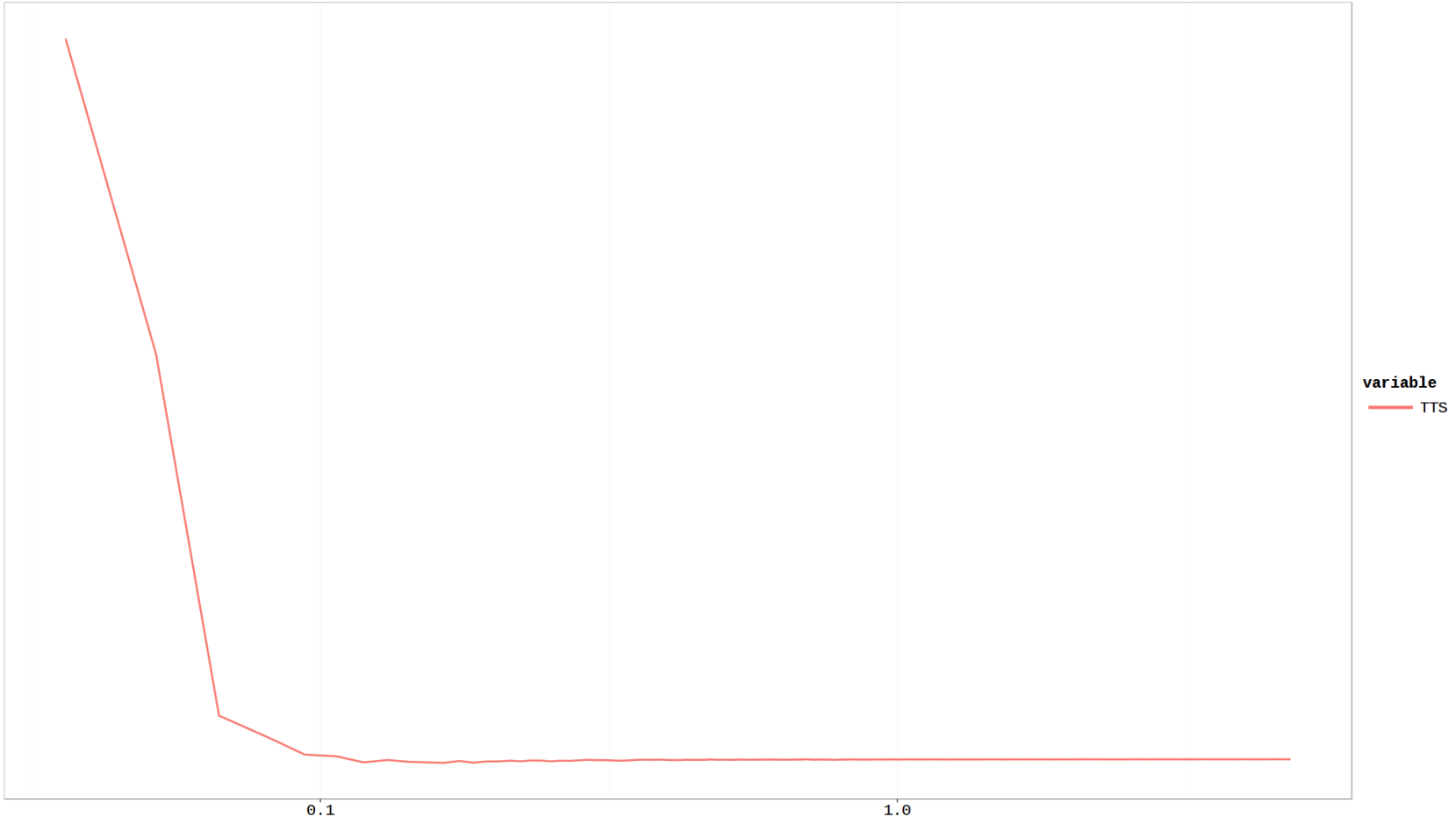
Задача : Barcelona

1020 узла, 2522 дуг, 7922 корреспонденций; 3 итераций; 0.044 сек.



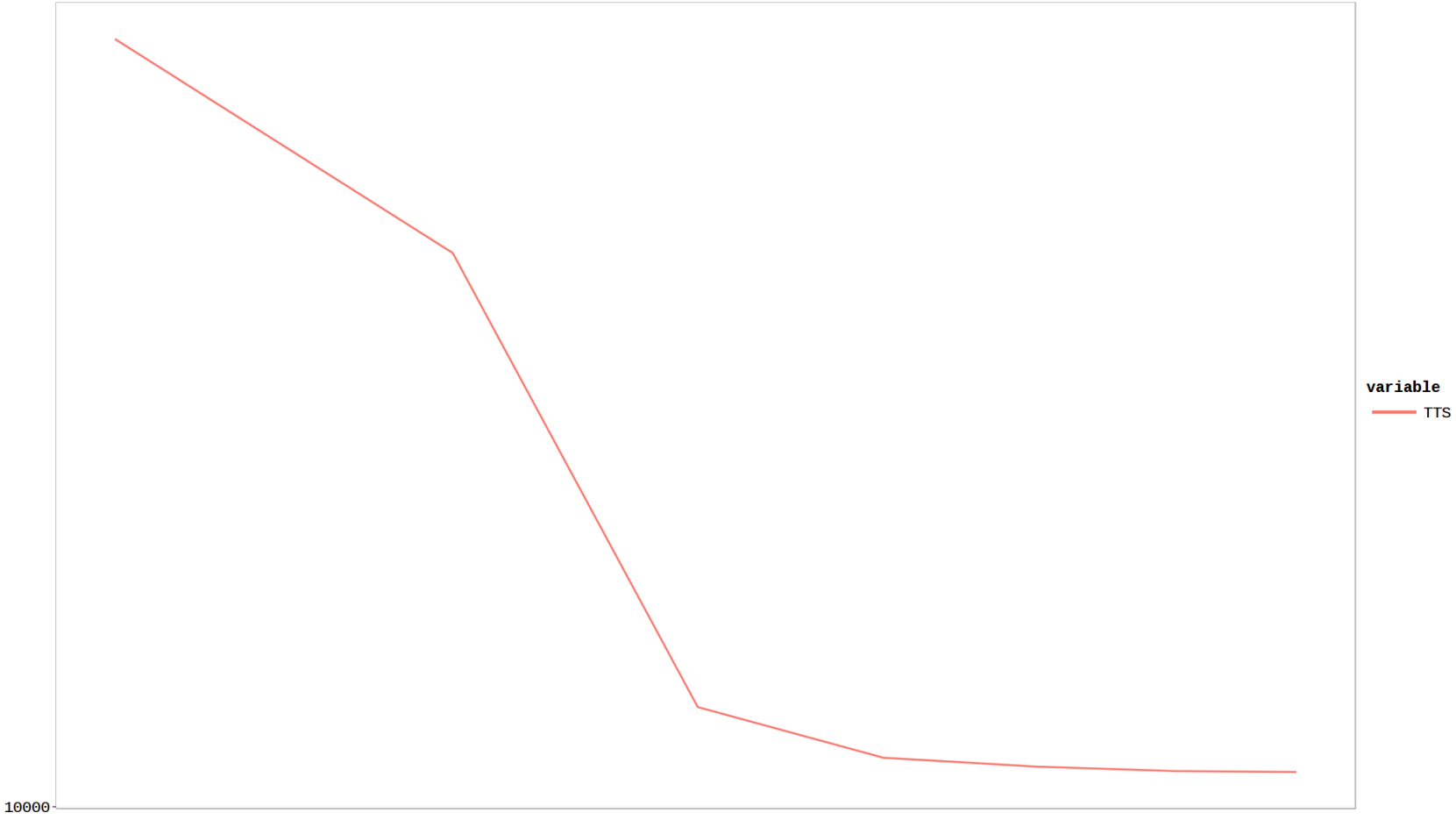
Задача : Barcelona

1020 узла, 2522 дуг, 7922 корреспонденций; 500 итераций; 4.803 сек.



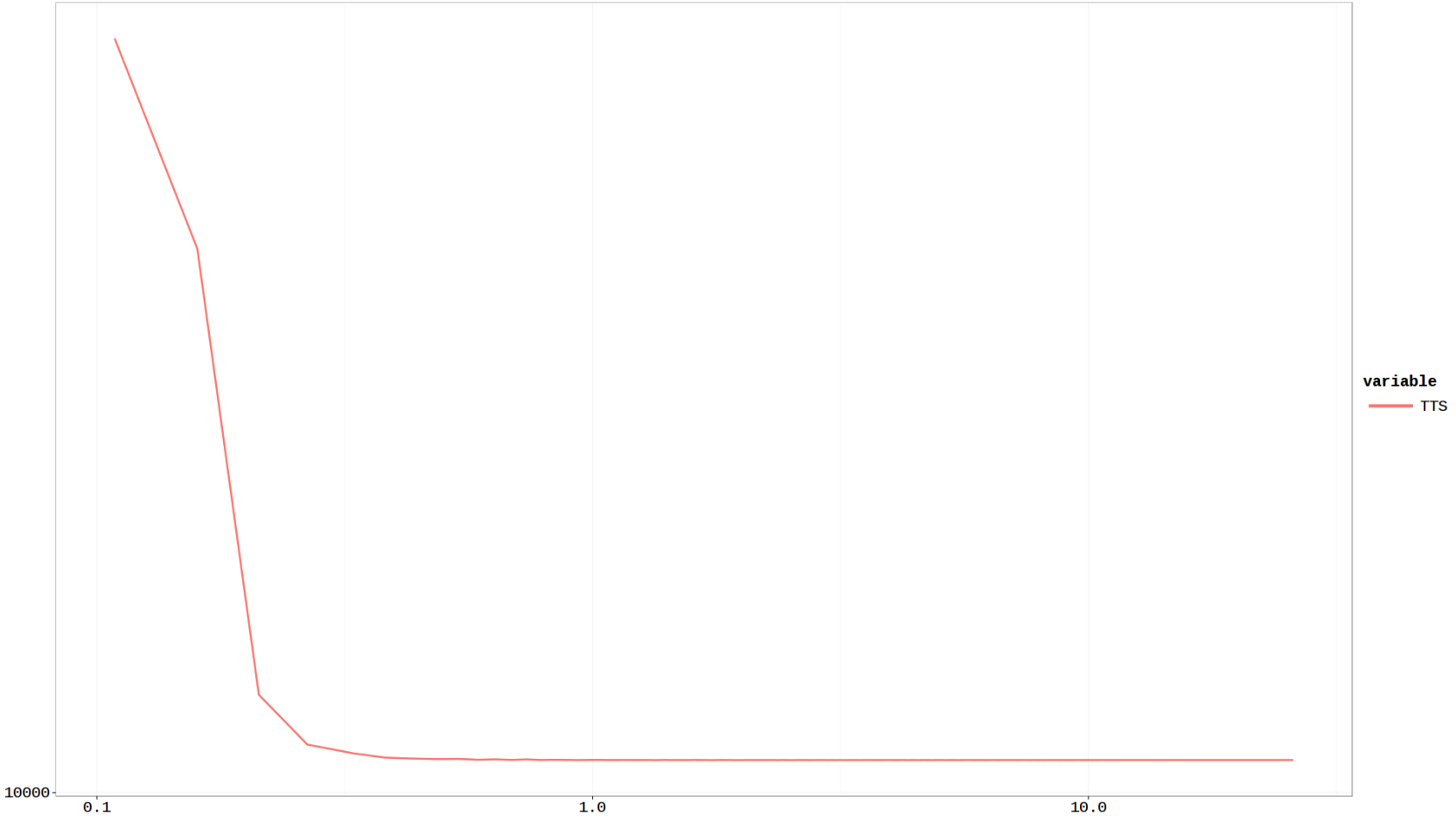
Задача : ChicagoSketch

933 узла, 2950 дуг, 142724 корреспонденций; 7 итераций; 0.044 сек.



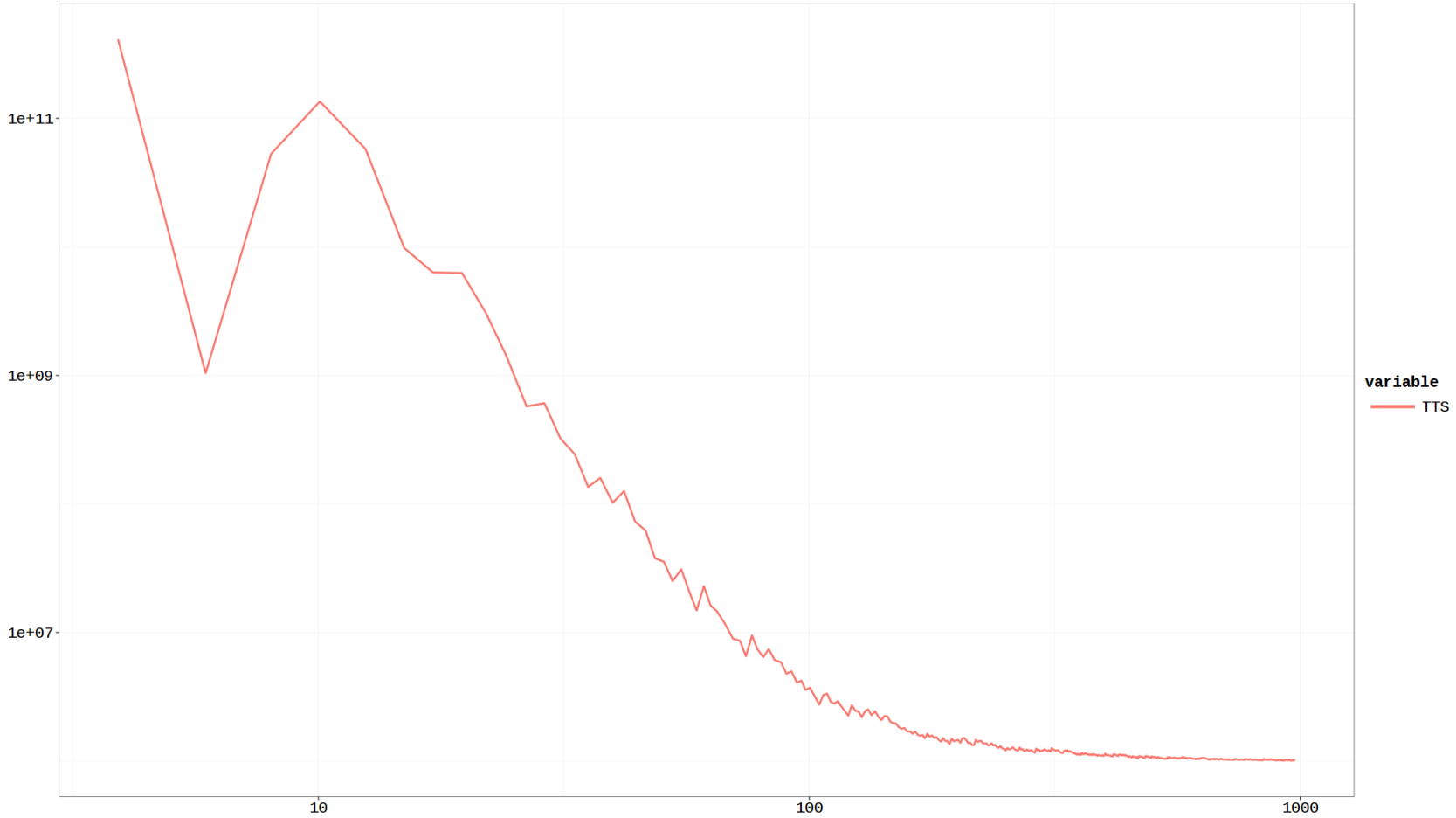
Задача : ChicagoSketch

933 узла, 2950 дуг, 142724 корреспонденций; 500 итераций; 25.895 сек.



Задача : Austin

7388 узла, 18961 дуг, 1246572 корреспонденций; 500 итераций; 977 сек.



Улучшение быстродействия

- Оптимизация кода (профилирование)
- Оптимизация алгоритма поиска кратчайших путей
- Выбор другого алгоритма поиска кратчайших путей
- Распараллеливание поиска кратчайших путей для одного источника — “мелкозернистый параллелизм” (Nvidia CUDA?)
- Распараллеливание поиска кратчайших путей для различных источников — “крупнозернистый параллелизм” (OpenMP, MPI)
- ...

Спасибо за внимание

Программная реализация метода условного градиента для поиска равновесия в модели Бэкмана

А.С. Аникин
anton.anikin@htower.ru,

А.Ю. Горнов
gornov@icc.ru

Институт динамики систем и теории управления СО РАН, Иркутск