ПАРАЛЛЕЛЬНЫЕ АЛГОРИТМЫ ПОИСКА КРАТЧАЙШИХ ПУТЕЙ НА ГРАФАХ

Выполнил: Ткаченко Г.С.
Руководитель: Корнеев Г.А.

13 мая 2015 г.

Университет ИТМО

ПРОБЛЕМА И ЗАДАЧА

РЕШАЕМАЯ ПРОБЛЕМА

- Недостаточное разнообразие параллельных алгоритмов для поиска кратчайших путей
- Низкая производительность отдельных алгоритмов на специфичных графах

ПОСТАНОВКА ЗАДАЧИ

- Эффективное применение алгоритмов поиска кратчайшего пути на **многопроцессорных** архитектурах
- Разработка алгоритмов для поиска пути от одной вершины до всех (**one-to-many**)
- Разработка алгоритмов для поиска пути кратчайшего расстояния между каждой парой вершин (**many-to-many**)

ЗАДАЧА ONE-TO-MANY

ОБЗОР РЕШЕНИЙ

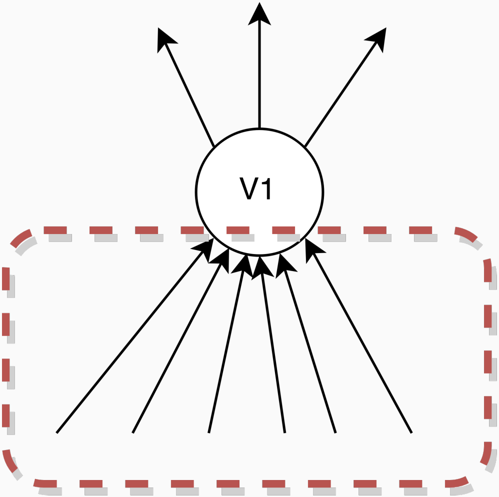
- Алгоритм Беллмана-Форда
    - Классический
    - На основе обхода в ширину
- Алгоритм Дейкстры
- Алгоритм Джонсона (Дейкстра с потенциалами)
- Алгоритмы A* и D*

ПАРАЛЛЕЛЬНЫЙ БЕЛЛМАН-ФОРД

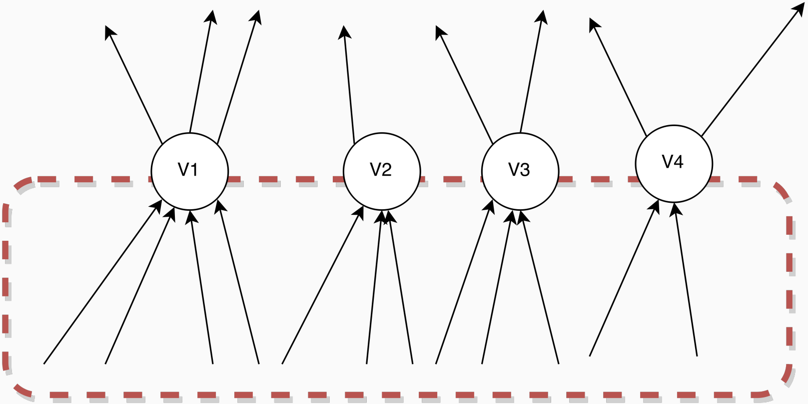
Три подхода

- Параллелизация по ребрам вершины
- Параллелизация по всем ребрам
- Использование параллельного обхода в ширину

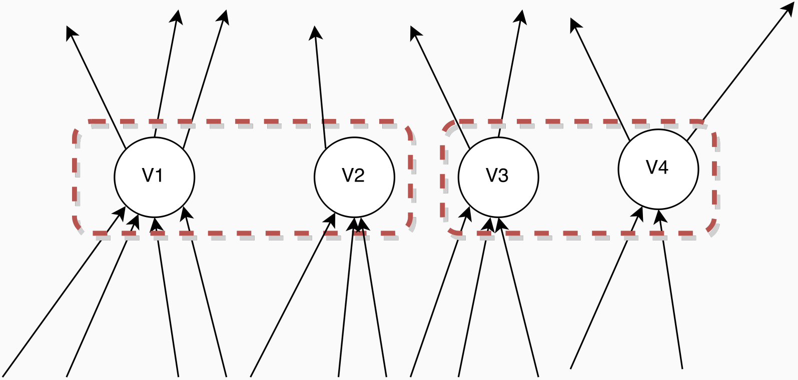
ПАРАЛЛЕЛИЗАЦИЯ ПО РЕБРАМ ВЕРШИНЫ



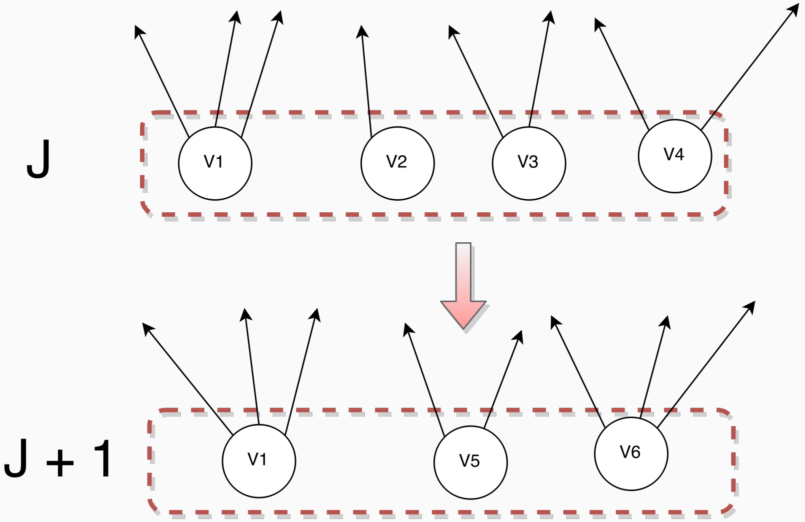
ПАРАЛЛЕЛИЗАЦИЯ ПО ВСЕМ РЕБРАМ



ПАРАЛЛЕЛИЗАЦИЯ ПО ВСЕМ РЕБРАМ



ИСПОЛЬЗОВАНИЕ ПАРАЛЛЕЛЬНОГО ОБХОДА В ШИРИНУ



ЗАДАЧА MANY-TO-MANY

АЛГОРИТМ ФЛОЙДА

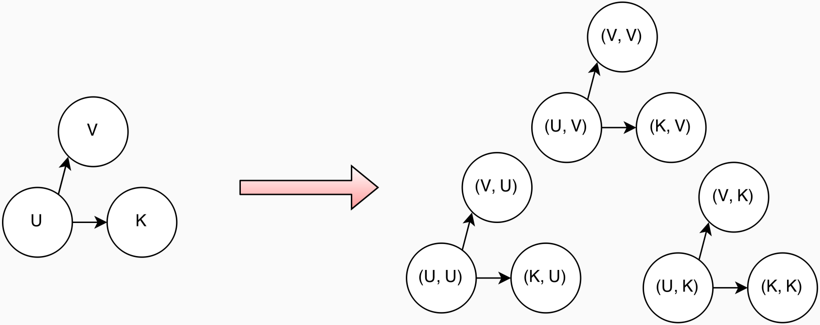
- В некоторых случаях классический алгоритм оказывается медленнее наивных алгоритмов
- Для каждой вершины можно использовать любой алгоритм поиска кратчайшего пути

НАИВНАЯ ПАРАЛЛЕЛЬНАЯ ВЕРСИЯ

```
procedure AllPairsPar1(G)
    return HandleVertices(G, 0, |G.vertices|)

procedure HandleVertices(G, startVertex, endVertex)
    if endVertex − startVertex < threshold then
        distances ← run Bellman-Ford for [startVertex, endVertex)
        return distances
    else
        midV ← (startVertex + endVertex)/2
        fork2(
            HandleVertices(G, startV, midV),
            HandleVertices(G, midV, endV));
```

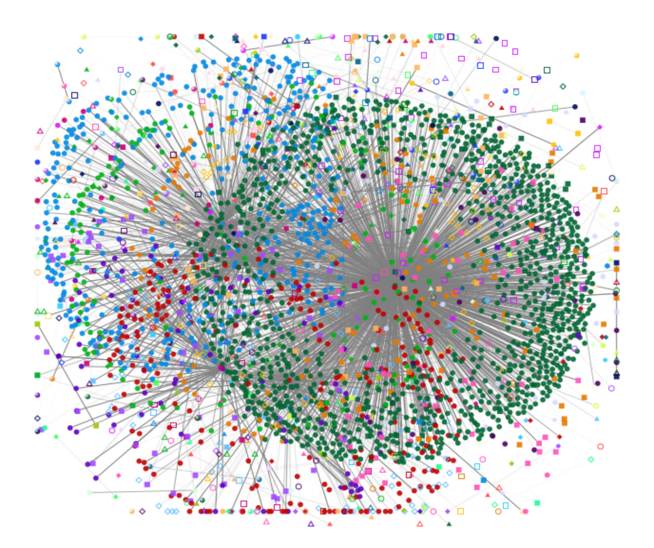
АЛГОРИТМ ДЛЯ ОБЪЕДИНЕННОГО ГРАФА



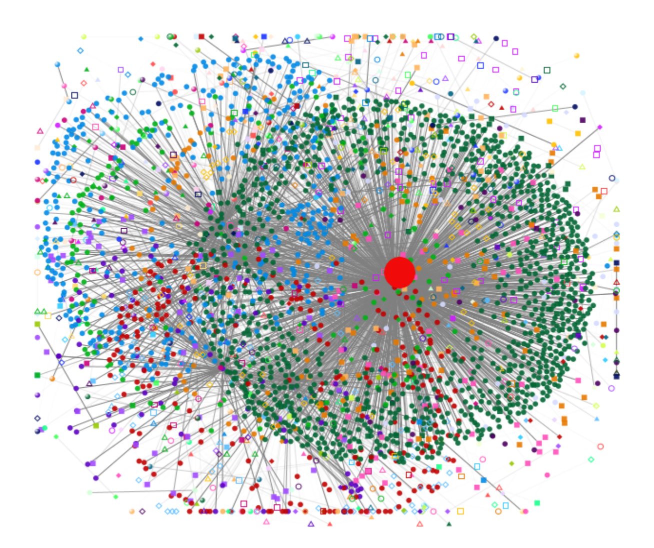
АЛГОРИТМ ДЛЯ СОЦИАЛЬНЫХ ГРАФОВ

- Основан на теории ”Шести рукопожатий”
- Работает не неориентированных невзвешенных социальных графах
- Использует идею динамического программирования

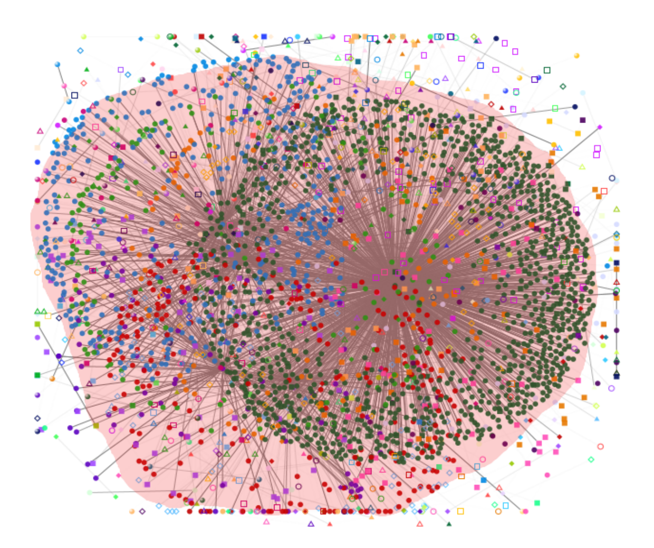
ИДЕЯ



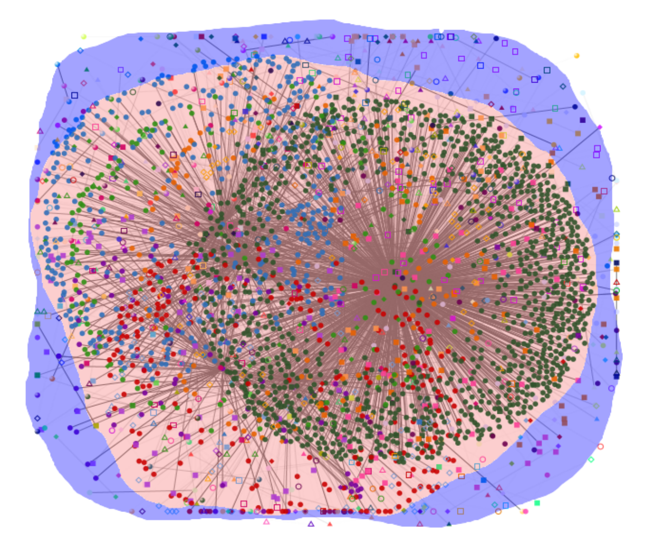
ИДЕЯ



ИДЕЯ



ИДЕЯ



- Посчитаем расстояние от базовой вершины до всех других
- Будем поддерживать две динамики mask и calc. Причем mask[u][i] - сабсет вершин, расстояние от которых до u равно i. В свою очередь calc[u][i] - не более i

$$\mathrm{mask}[v][i] = \neg \mathrm{calc}[v][i-1] \wedge \bigvee_{\exists (u,v) \in E} \mathrm{mask}[u][i-1] \tag{1}$$

$$\mathrm{calc}[v][i] = \mathrm{calc}[v][i-1] \vee \mathrm{mask}[v][i] \tag{2}$$

РЕЗУЛЬТАТЫ

СРАВНЕНИЕ ПАРАЛЛЕЛЬНЫХ ВЕРСИЙ БЕЛЛМАНА-ФОРДА

Algo №	Complete			BalancedTree		SquareGrid	
	TS	+	-	0.5	1	+	+-
1	2.43	4.65	nc	116.31	9.04	5.49	13.40
2	5.17	0.18	10.84	3.59	3.08	5.92	7.10
3	44.63	0.37	23.55	0.44	0.31	4.42	0.58

Таблица: Типичные графы

Algo №	RandomSparse			RandomDense		
	0.5+	0.5-	0.96+	0.5+	0.5-	0.96+
1	nc	nc	24.35	nc	nc	5.01
2	2.77	14.68	2.42	0.48	6.38	0.46
3	0.98	22.59	0.76	0.60	10.25	0.71

Таблица: Случайные графы

РАССТОЯНИЕ МЕЖДУ КАЖДОЙ ПАРОЙ ВЕРШИН СОЦИАЛЬНОГО ГРАФА

Алгоритм	Twitter graph
Наивная параллельная версия	427.217
Алгоритм для социальных графов	210.322

Таблица: Сравнение алгоритмов

ВЫВОДЫ

Вопросы?