Интеллектуальные информационные системы

Работа с графами. PageRank

Кафедра управления и интеллектуальных технологий НИУ «МЭИ»

2023 г.

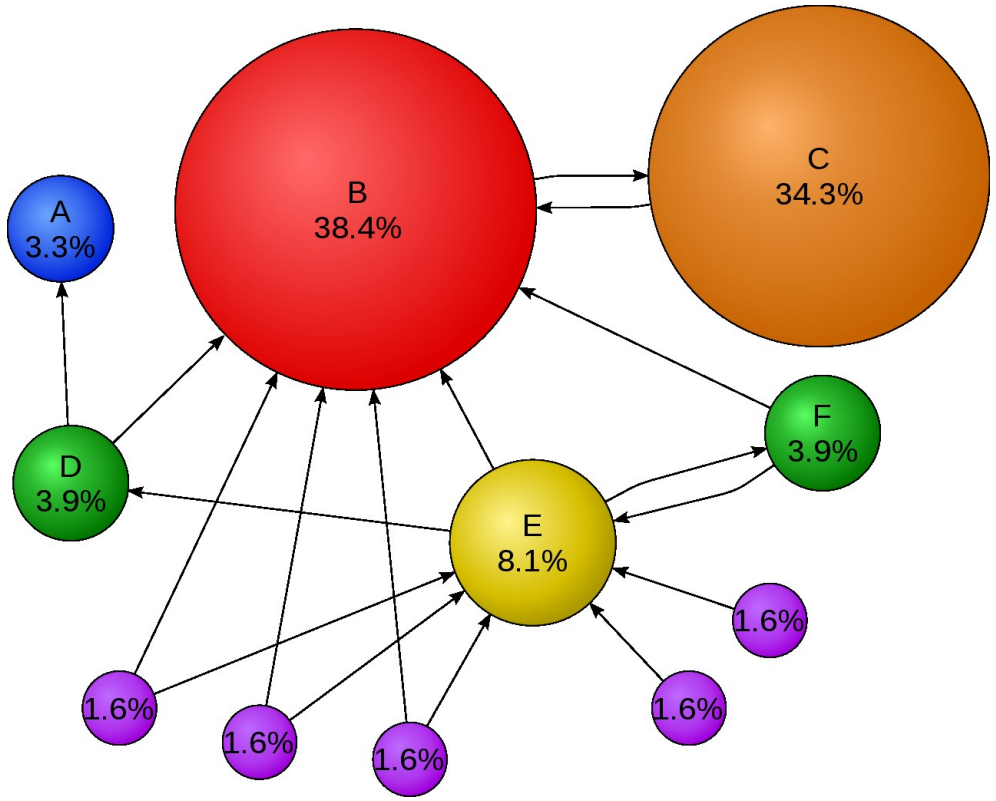
PageRank

Алгоритм ссылочного ранжирования веб-страниц, разработан в 1998 году Лари Пейджем.

Вычисления на узлах графа.

Определяет «авторитетность» страницы используя модель «блуждающего серфера»:

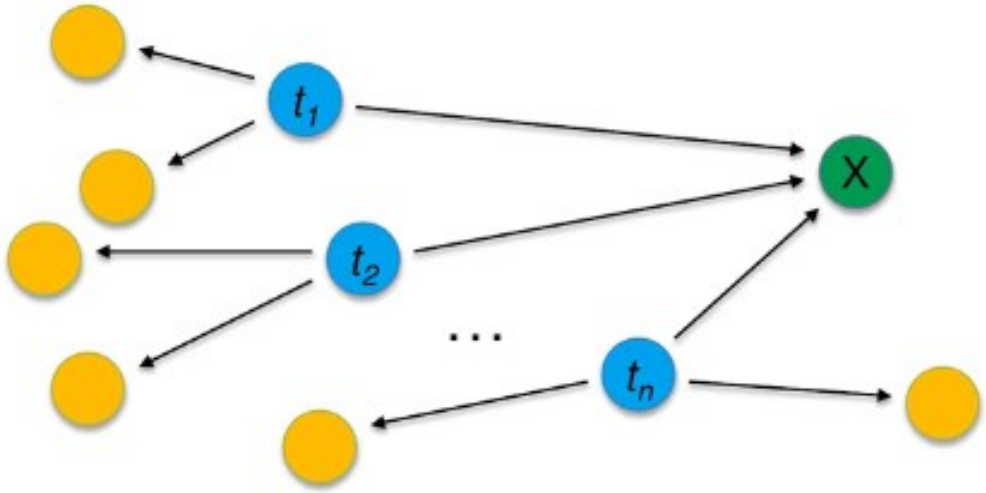
- Пользователь начинает серфинг со случайной страницы
- Произвольно кликает на ссылки и переходит на другие страницы (считаем вероятность перехода одинаковой)
- Нет «подвисших» вершин (без исходящих ссылок)



Вычисление PageRank

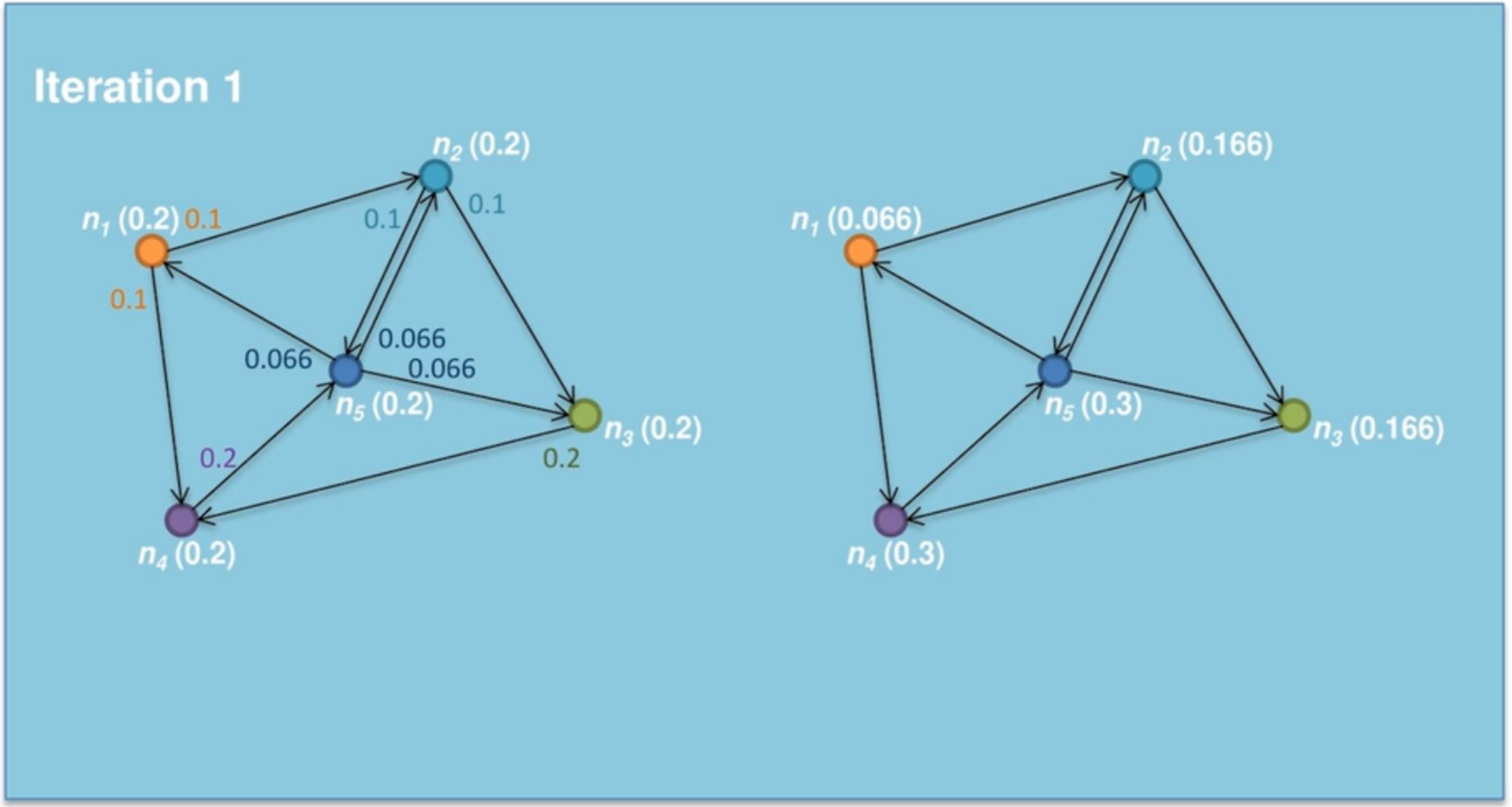
Узел – сайт в Интернете.
Ребра – исходящие ссылки.

N – общее количество страниц
PR(t_i) – PageRank страницы t_i
C(t_i) – количество ссылок со страницы

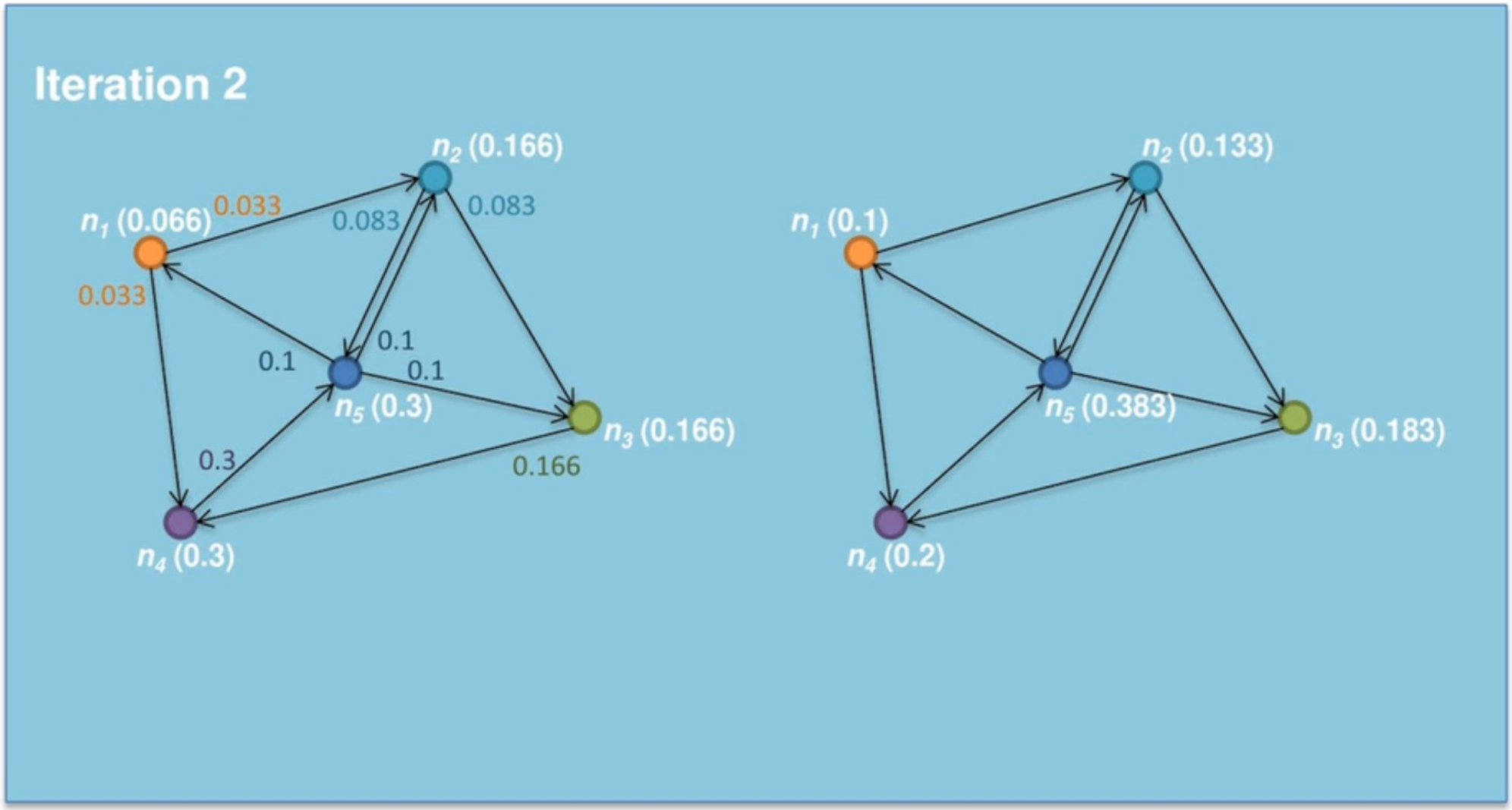


$$PR(x) = \alpha\left(\frac{1}{N}\right) + (1-\alpha)\sum_{i=1}^{n}\frac{PR(t_i)}{C(t_i)}$$

Вычисление PageRank



Вычисление PageRank



PageRank Mapper

```
Method Map(nid n, (pagerank pr, nodes N)):
      Emit (nid n, (pagerank pr, nodes N))
      p = pr / N.CountNodes
      for each node nodeid in nodes N:
            Emit (nodeid , (p, nodes {}))
```

PageRank Reducer

```
Method Reducer(nid n, [(pagerank pr, nodes N)]):
      M = p = ∅
      for each node in [(pagerank pr, nodes N)]:
            if node.N:
                  M.struct = node.N
            else
                  p += node.pr
      M.pr = p
      Emit (nid n, node M)
```

PageRank

Обработка «подвешенных» вершин:
Сумма PR_i в графе всегда – const -> накопление PR в «подвешенной» вершине.

Суммируем PR во всех подвешенных вершинах и равномерно распределяем по всем остальным.

Критерии останова:
PageRank – сходится к конкретным значениям => продолжаем итерации пока значения не перестанут меняться (или меняются не более чем на *delta*)

Фиксированное число итераций

Проблемы графов на MapReduce

Алгоритмы на графах итерационные – следующая итерация не будет запущена, пока не выполнится предыдущая => Любое накопление времени или неравномерность выполнения существенно скажется на итоговой производительности (например, запуск MR задачи).

Передача между mapper и reducer структуры графа.

Проблемы графов на MapReduce. Решение

Использование Combiner, в том числе In-mapper Combiner.

Улучшение партиционирования: зачастую графы имеют локальную структуру:

* Обрабатывать близкие узлы на одной ноде.
* Сложно
* А если близких узлов много?

Schimmy Design Pattern (SDP):
Передаем два набора данных: структуру графа и актуальные вычисления «messages».
Передаем только messages, структуру считываем с диска.

Проблемы графов на MapReduce. Решение

Обе части (S и T) консистентно партиционированы и сортированы по join key