Построение, запуск и анализ потоков данных в распределенных средах

Борисяк Максим

МФТИ

2013г.

Цели работы

- Создание формальной модели потока данных
- Анализ неопределенностей в схемах потоков данных
- Оценка необходимых ресурсов для запуска потока данных

Определение потока данных

Поток данных — модель представления вычеслительных процессов, включающая:

- набор атомарных функций-исполнителей,
- составные исполнители или подпотоки данных,
- описание зависимостей между исполнителями,
- стратегия выполнения.

Задачи потоков данных

- Научные-инженерные задачи.
  - Большие объемы данных.
  - Выполнение в распределенных средах.
- Изолиция от технических аспектов вычислений.
- Отказоустойчивость.

Приемущества потоков данных

- Возможность интуитивно понятного визуального представления.
- Изолированность от технических аспектов вычислительной среды.
- Простота создания.

Способы представления

- Декларативные либо скриптовые ЯП.
- Графы.
- Сети Петри.
- Сети процессов Кана (для потоковой обработки).

Используемая модель потоков данных

Используется представление в виде ориентированного графа. Поток данных — кортеж $c = (B, E, I, O)$:

- набор блоков B — каждый блок осуществляет преобразование данных;
- связи E — зависимости по данным между входными портами одних и выходными портами других блоков;
- I, O — наборы входных и выходных портов (начальных данных и результатов).

Блок

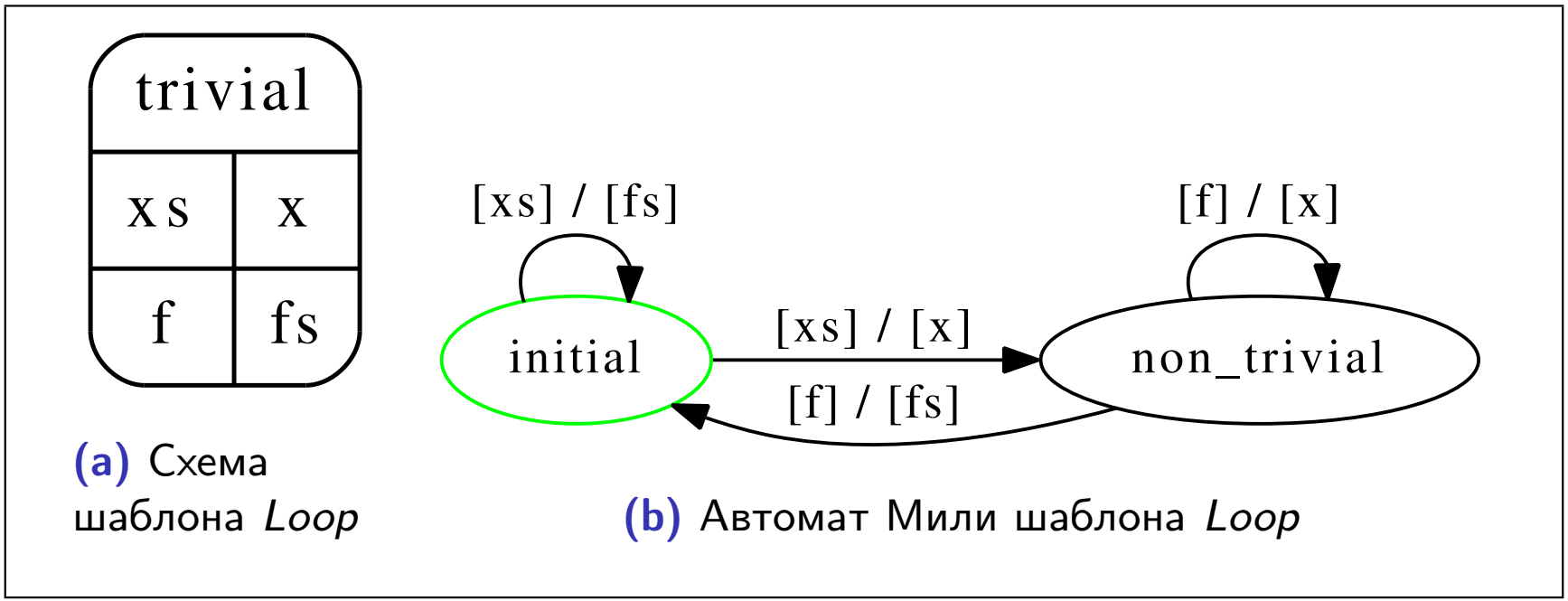
Блок — единица преобразования данных, кортеж $s = (FA, I, O)$:

- FA — автомат Мили, описывающий качественное поведение алгоритма блока:
  - входной алфавит: 2^I;
  - выходной алфавит: 2^O;
  - состояния определяют классы внутренних состояний переменных алгоритма блока;
- I, O — наборы входных и выходных портов.

Различается два типа блоков:

- атомарные — предопределенные блоки элементарных операций;
- составные — потоки данных, представленные в виде блоков.

Наглядное представление блоков и автомата Мили



(a) Схема шаблона *Loop*

(b) Автомат Мили шаблона *Loop*

Рис.: Схематичное изображение шаблона блока *Loop* (a) и соответствующего ему конечного автомата Мили (b).

Наглядное представление иерархичного потока данных

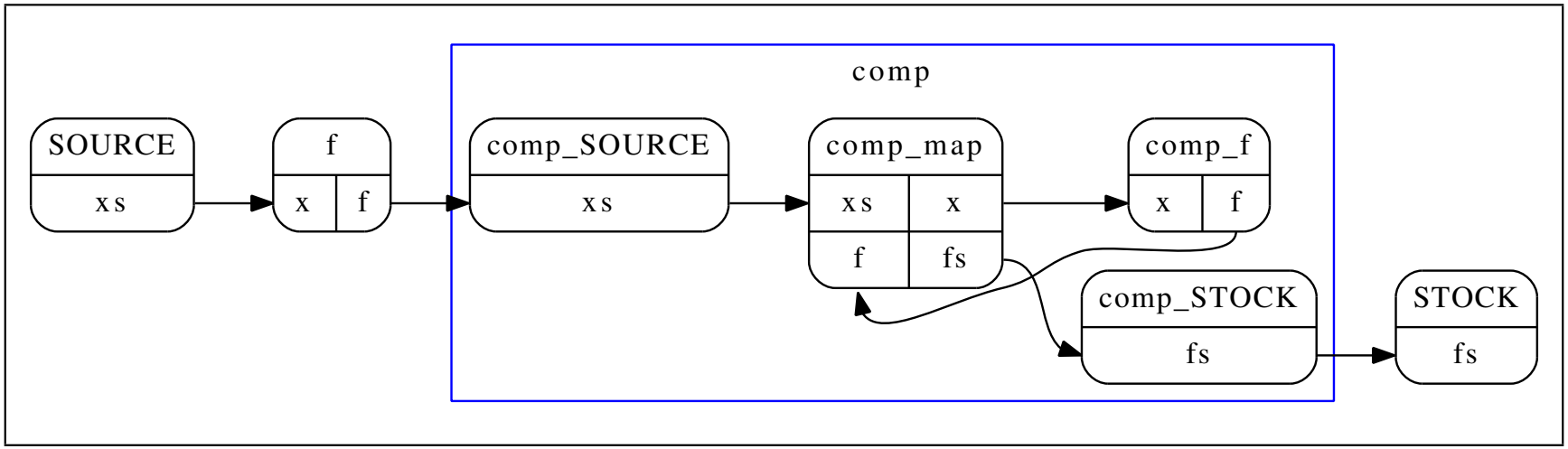
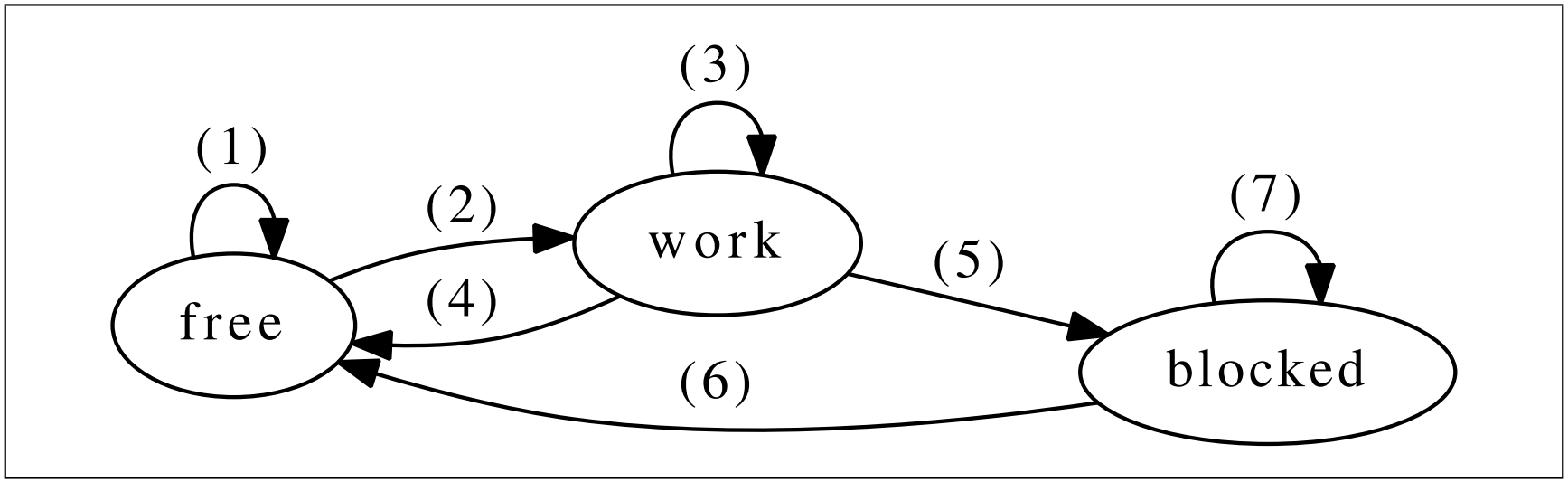


Рис.: Шаблон составного блока. Внутри рамки показана структура шаблона внутреннего блока.

Состояния блока во время запуска потока данных

Три состояния каждого блока:

- работает (work) — выполняет преобразование данных;
- заблокирован (blocked) — завершил работу, но не может испустить данные;
- свободен (free) — не хватает данных для запуска.



Алгоритм запуска потока данных

Алгоритм запуска потока данных:

1. определение блоков для запуска в соответствии с запрашиваемым набором данных ;
2. передача данных, запуск блоков ;
3. проверка возможности вывода блока из состояния 'заблокирован':
   - возможно — переход к шагу 2;
   - иначе — к шагу 4;
4. ожидание окончания работы блоков, проверка завершения работы потока данных :
   - работа завершена — окончание, выдача результата;
   - иначе — к шагу 1.

Неопределенности в работе потоков данных

- Варианты переходов автомата Мили блока.
- Варианты поглощения данных на входных портах блока.
- Состояния гонки: зависимость результата от времен работы блоков.
- Остаточные данные.

Первый тип неопределенности — результат упрощения (детерминированного) алгоритма работы блоков, неопределенности первого рода. Остальные — ошибки построения схемы потока данных, неопределенности второго рода.
Корректный поток данных — поток данных, не допускающий неопределенностей второго рода.

Анализ корректных потоков данных

Траектория запуска — наборы состояний потока данных во все моменты времени.
Граф причинности — качественное описание варианта поведения потока данных:

- вершины — кортежи $r = (v, \pi_v, n)$, $v \in B_c$ — блок, π_v — переход в автомате Мили блока v, n - номер.
- ребра — 'причины' запуска: испущенные данные одного блока приняты другим.

Графы причинности разбивают все возможные траектории запуска на классы эквивалентности. Для корректного набора данных каждый класс содержит траекторий соотвующие любым возможным наборам времен работы блоков.

Простой волновой алгоритм

Для ацикличных корректных потоков данных множество графов причинности конечно.

1. проверка на завершенность потока данных:
   - поток данных завершен — возврат графа причинности;
   - иначе — далее;
2. формирование множеств возможных переходов для каждого блока $\{\Pi(v)\}_{v \in B_c}$;
3. для каждого набора возможных переходов $\{\pi(v) \in \Pi(v)\}_{v \in B_c}$:
   1. осуществление переходов $\pi(v)$ с формированием нового состояния потока данных;
   2. дополнение графа причинности;
   3. получение результата рекурсивного запуска из нового состояния;
4. возврат объединения результатов.

Волновой алгоритм

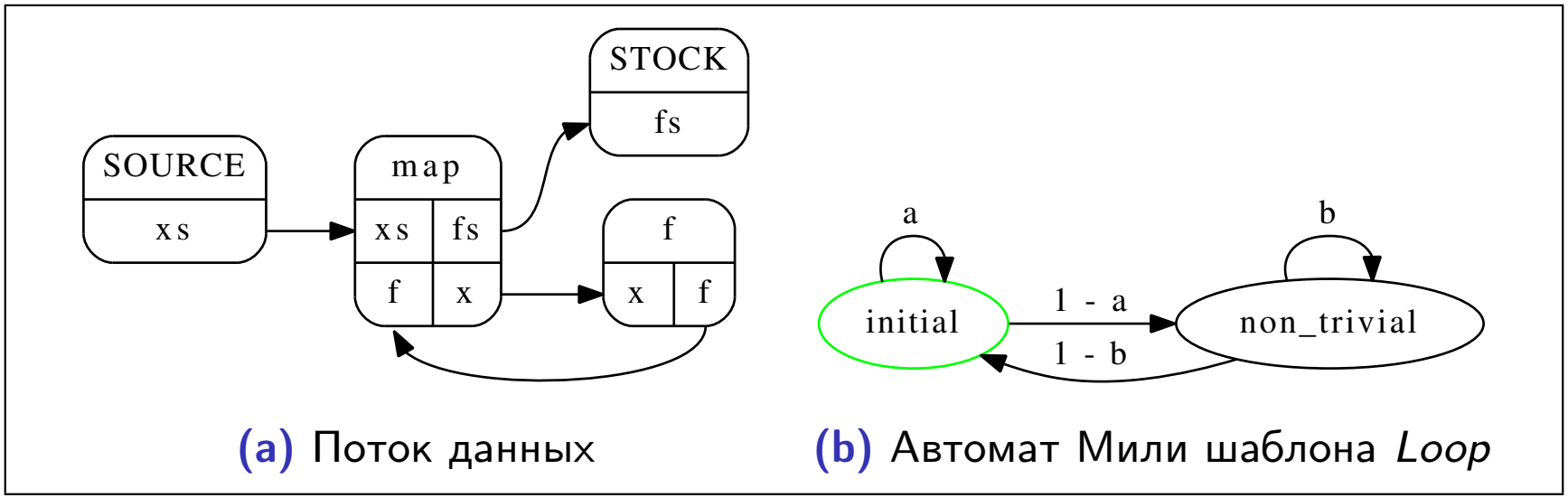
При наличии циклов в потоке данных множество графов причинности бесконечно, но регулярно. Волновой алгоритм строит 'внутренний' конечный автомат потока данных — аналог конечного автомата для регулярных языков:

- вершины — состояния потока данных;
- ребра соответствуют работе блоков в потоке данных.

Циклы соответствуют циклам во 'внутреннем' конечном автомате. Любой путь во 'внутреннем' конечном автомате потока данных соответствует графу причинности и наоборот.

Вероятности на графах причинности

Часто используемый пример.



(a) Поток данных **(b)** Автомат Мили шаблона *Loop*

Рис.: Часто используемая схема потока данных

$$\mathbb{E}(L) = (1-a)(1-b)\sum_{n=1}^{\infty} nb^{n-1} = \frac{1-a}{1-b}$$

Анализ графа причинности

Выполнение потока данных в однородной распределенной среде. Один переход задействует один вычислитель на неопределенное время.

Срез графа причинности S — множество независимых переходов (задач) графа причинности.

Эволюция графа причинности $\Delta = \{S^j\}_{j=0}^{J}$ — набор связанных срезов графа причинности.

$$N_{\min} = \min_{\Delta} \max_{S^j \in \Delta} |S^j|$$

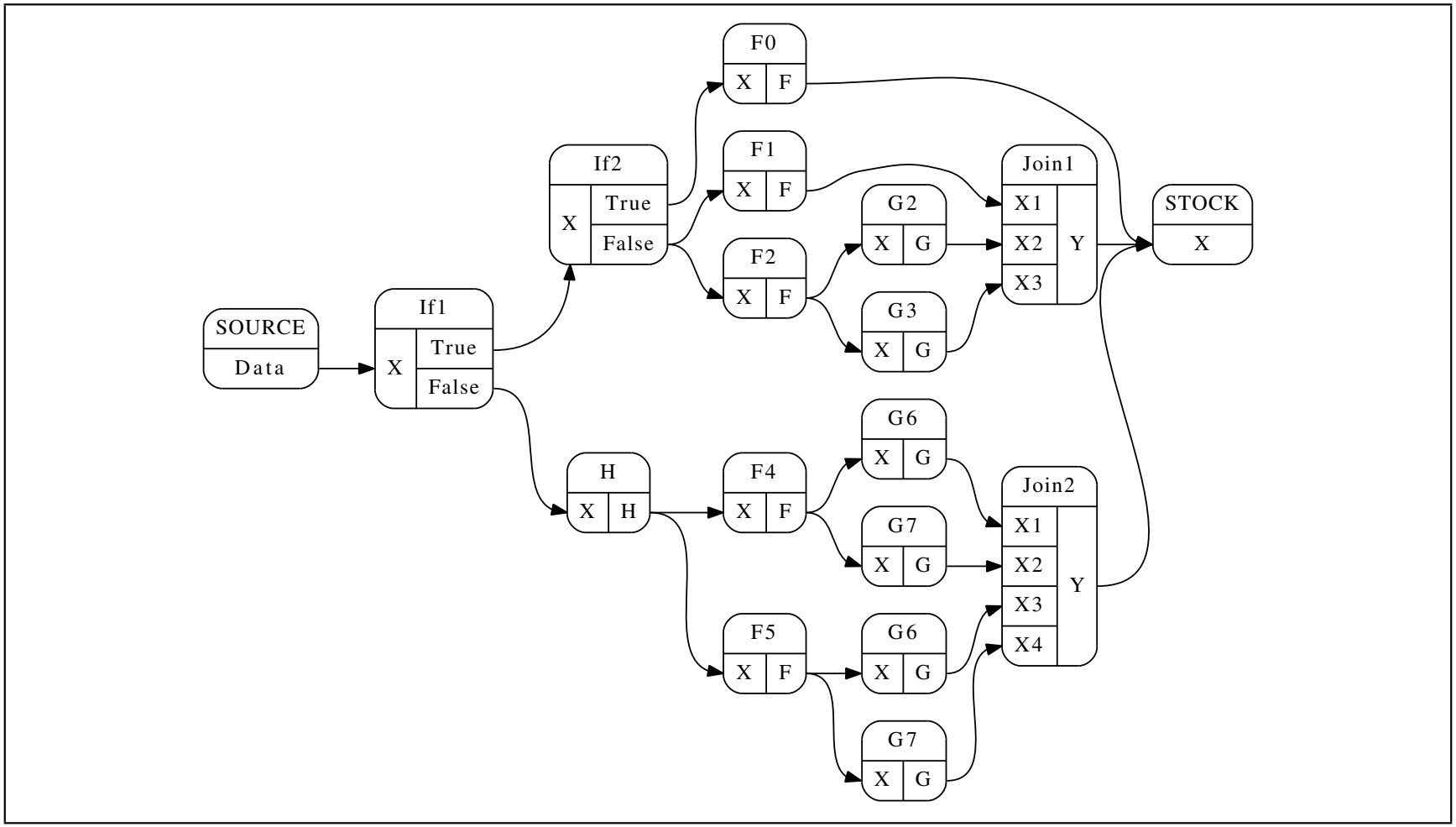
$$N_{\max} = \max_{\Delta} \max_{S^j \in \Delta} |S^j|$$

Выполнять поток данных на $N > N_{\max}$ вычислителях бессмысленно. Для $N < N_{\min}$ — заведомо высокая конкуренция процессов. Для определения $N_{\min}$ и $N_{\max}$ используется **SHELF-алгоритм**.

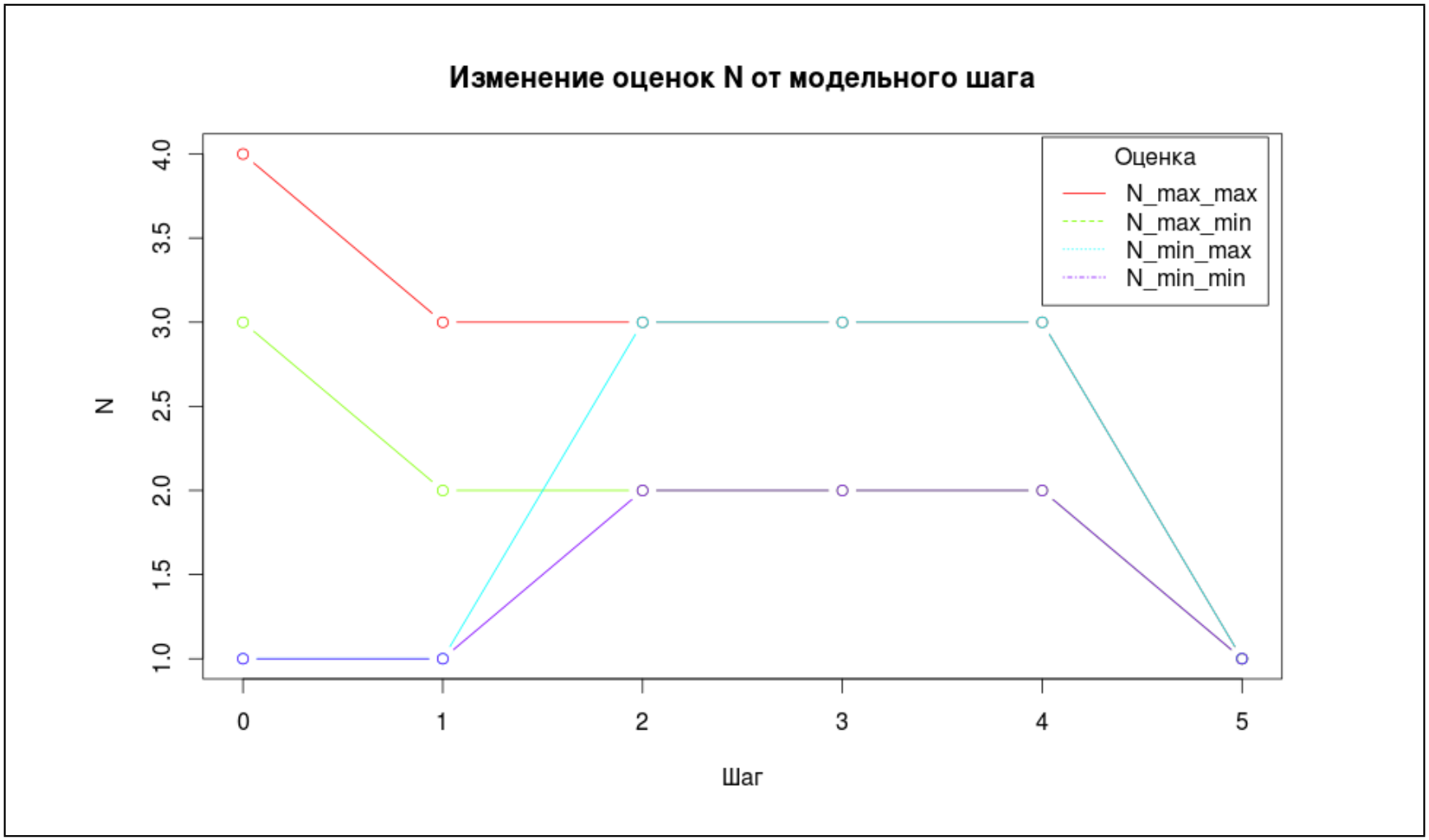
Динамическое изменение оценок

Все графы причинности образуют префиксное дерево. По мере работы потока данных выполняемый граф причинности уточняется — возможно уточнение оценок. Для каждой вершины префиксного дерева можно вычислить $\hat{N}_{\{\max,\min\},\{\max,\min\}}$ для остатков возможных графов причинности. Это позволяет осуществить динамическое уточнение оценок необходимых ресурсов.

Пример потока данных



Результаты динамического уточнения оценок



Обнаружение неопределенностей второго рода

- Остаточные данные — во время работы волнового алгоритма по наличию состояний с остаточным данными.
- Варианты поглощения данных — во время работы волнового алгоритма.
- Состояния гонки — ведение истории на портах, ведение истории данных.

Обнаружение состояний гонки

Результаты

- Построена модель потока данных.
- Определены возможные неопределенности и ошибки построения потока данных.
- Построен алгоритм нахождения ошибок построения потока данных.
- Найдены алгоритмы нахождения стратегий запуска для корректных алгоритмов.

Дальнейшая работа

- Введение модели оценки времени работы потока данных:
  - модель оценок времен работы блоков;
  - стохастическая модель переходов;
  - обработка историй реальных запусков методами анализа данных.
- Алгоритмы автоматической кластеризации блоков.
- Модификация модели потоков данных.
- Анализ работы потоков данных в гетерогенных и/или динамических распределенных средах.