Санкт-Петербургский государственный университет

Кафедра системного программирования

Группа 24.М71-мм

Разложение решений задачи достижимости с контекстно-свободными ограничениями пути

Муравьев Илья Владимирович

Отчёт по учебной практике
в форме «Решение»

Научный руководитель:
доцент кафедры системного программирования, к. ф.-м. н., С. В. Григорьев

Санкт-Петербург
2024

Оглавление

Введение

Задача достижимости с контекстно-свободными ограничениями пути (КС-достижимости) — это задача поиска в рёберно-меченом графе G пар вершин $\langle u, v \rangle$ таких, что из u в v существует путь, метки рёбер вдоль которого формируют слова из данного контекстно-свободного языка $\mathcal{L}$. К задаче КС-достижимости сводятся задачи из различных прикладных областей, например, задачи анализа RDF-графов [8], задачи биоинформатики [32], задача исправления ошибок компиляции минимальными изменениями кода [26], а также такие фундаментальные[1] задачи статического анализа кода, как анализ указателей (англ. *points-to analysis*) [9] и анализ псевдонимов (англ. *alias analysis*) [41]. Тем не менее, несмотря на столь широкий спектр применений КС-достижимости, многие работы, посвящённые разработке решателей КС-достижимости (КС-решателей), неуклонно фокусируются на частных случаях [39, 9]. Это объясняется тем, что для решения задачи КС-достижимости за разумное время зачастую оказывается необходимым учитывать паттерны в решениях, специфичные для определённых КС-грамматик, например, специальным образом обрабатывать полностью транзитивные и частично транзитивные отношения.

Перспективным универсальным алгоритмом решения задачи КС-достижимости является матричный алгоритм Рустама Азимова [4]. Данный алгоритм решает задачу КС-достижимости в терминах хорошо распараллеливаемых операций линейной алгебры с разреженными матрицами, что при использовании современного аппаратного обеспечения часто позволяет ему обходить аналоги по скорости [3]. Тем не менее даже после многочисленных оптимизаций [24, 19] матричный алгоритм всё ещё в ряде случаев уступает специализированным инструментам, учитываемым паттерны решений, специфичные для конкретных КС-грамматик.

Следовательно, перспективным направлением развития универсаль-

[1]«Фундаментальность» задач анализа псевдонимов и указателей заключается в том, что они используется для реализаций множества инспекций кода [15, 12, 13, 34, 25, 10, 21, 16].

ных КС-решателей является динамическое выявление паттернов в решениях задачи КС-достижимости. В частности, в [24] предполагается, что если при итеративном решении задачи КС-достижимости после одной из итераций удастся разложить текущее приближение решения на произведение «сильно разреженных» матриц, «то дальнейшие итерации матричного алгоритма можно будет производить намного быстрее, оперируя более разреженными матрицами», то есть учитывая паттерны в решениях задачи КС-достижимости. Хотя в [24] и отмечается полезность разложения решения задачи КС-достижимости, конкретные подходы для выполнения этого разложения пока не были разработаны.

В связи с этим данная работа посвящена созданию алгоритма разложения решений задачи КС-достижимости на основе различных подходов, используемых при решении смежных задач [40, 23, 18, 20, 35], реализации этого алгоритма с использованием высокопроизводительных библиотек, оценке качества получаемых разложений, а также скорости получения этих разложений.

1 Постановка задачи

Целью работы является разработка разлагателя решений задачи достижимости с контекстно-свободными ограничениями пути (КС-достижимости). Для её выполнения были поставлены следующие задачи.

1. Провести обзор подходов, применяемых для решения смежных задач.
2. Предложить алгоритм разложения решений задачи КС-достижимости и оценить его временную сложность.
3. Спроектировать архитектуру разлагателя решений задачи КС-достижимости.
4. Реализовать и протестировать разлагатель решений задачи КС-достижимости на основе предложенных алгоритма и архитектуры.
5. Экспериментально сравнить на реальных данных производительность разработанного разлагателя и решателей задачи КС-достижимости и измерить достигаемую разлагателем степень сжатия решений задачи КС-достижимости.

2 Обзор

Данный раздел включает:

- формулировку задачи КС-достижимости;
- описание матричного алгоритма решения задачи КС-достижимости и краткое сравнение матричного алгоритма с другими алгоритмами КС-достижимости;
- описание версии матричного алгоритма КС-достижимости, учитывающей паттерны в решениях с помощью разложения матриц;
- обзор возможных подходов к разложению решений задачи КС-достижимости.

2.1 Формулировка задачи КС-достижимости

Для того, чтобы сформулировать задачу КС-достижимости, необходимо определить некоторые вспомогательные понятия.

Определение 1. Кортеж из нуля элементов называется *пустой строкой* и обозначается ε.

Определение 2. Пусть Σ и N — это непересекающиеся множества, $S \in N$, $P \subseteq \{A \to \alpha \mid A \in N, \alpha \in (\Sigma \cup N)^*\}$. Тогда:

- $Gr = \langle N, \Sigma, P, S \rangle$ называется *КС-грамматикой*,
- Σ называется *алфавитом терминалов*,
- N называется *алфавитом нетерминалов*,
- P называется *множеством продукций*,
- S называется *стартовым нетерминалом*.

Определение 3. Пусть $Gr = \langle N, \Sigma, P, S \rangle$ — это КС-грамматика, $\alpha, \beta, \gamma \in (\Sigma \cup N)^*$, $A \in N$, $(A \to \alpha) \in P$. Тогда говорят, что строка

$\beta\alpha\gamma$ *непосредственно выводится* из строки $\beta A\gamma$ в грамматике Gr и пишут $\beta A\gamma \underset{Gr}{\Rightarrow} \beta\alpha\gamma$.

Определение 4. Пусть Gr — это КС-грамматика, тогда рефлексивно-транзитивное замыкание отношения $\underset{Gr}{\Rightarrow}$ называется *отношением выводимости* для Gr и обозначается $\overset{*}{\underset{Gr}{\Rightarrow}}$, то есть

$$\alpha \overset{*}{\underset{Gr}{\Rightarrow}} \beta \overset{\text{def}}{\iff} \exists k \geq 0 : \exists \gamma_1, \ldots, \gamma_k : \alpha \underset{Gr}{\Rightarrow} \gamma_1 \underset{Gr}{\Rightarrow} \gamma_2 \underset{Gr}{\Rightarrow} \ldots \underset{Gr}{\Rightarrow} \gamma_k \underset{Gr}{\Rightarrow} \beta.$$

Определение 5. Пусть $Gr = \langle N, \Sigma, P, S\rangle$ — это КС-грамматика, тогда *языком, задаваемым КС-грамматикой* Gr, называется множество

$$\mathcal{L}(Gr) \overset{\text{def}}{=} \{\omega \in \Sigma^* \mid S \overset{*}{\underset{Gr}{\Rightarrow}} \omega\}.$$

Определение 6. Пусть V и Σ — это непересекающиеся множества, $E \subseteq V \times \Sigma \times V$. Тогда:

- $G = \langle V, E, \Sigma\rangle$ называется *рёберно-меченым графом*,
- V называется *множеством вершин*,
- E называется *множеством рёбер*,
- Σ называется *множеством меток*.

Определение 7. Пусть $G = \langle V, E, \Sigma\rangle$ является рёберно-меченым графом, $n \geq 0$ и $\langle v_1, l_1, v_2\rangle, \langle v_2, l_2, v_3\rangle, \ldots, \langle v_{n-1}, l_{n-1}, v_n\rangle \in E$. Тогда $\pi = \langle\langle v_1, l_1, v_2\rangle, \langle v_2, l_2, v_3\rangle, \ldots, \langle v_{n-1}, l_{n-1}, v_n\rangle\rangle$ называется *путём* в графе G из вершины v_1 в вершину v_n.

Определение 8. Пусть $\pi = \langle\langle v_1, l_1, v_2\rangle, \langle v_2, l_2, v_3\rangle, \ldots, \langle v_{n-1}, l_{n-1}, v_n\rangle\rangle$ — это путь. Тогда $\omega(\pi) \overset{\text{def}}{=} l_1 l_2 \ldots l_{n-1}$ называется *словом вдоль пути* π.

Определение 9. *Булевым полукольцом* называется полукольцо, обозначаемое $\mathbb{B}$, с доменом $\{0, 1\}$, операцией логического «ИЛИ» ($\vee$) в роли сложения и операцией логического «И» ($\wedge$) в роли умножения.

Теперь, когда определены необходимые вспомогательные понятия, можно формально сформулировать задачу достижимости с контекстно-свободными ограничениями.

Определение 10. Пусть $G = \langle V, E, \Sigma \rangle$ — это рёберно-меченый граф, $V = \{v_1, \ldots, v_n\}$, $Gr = \langle N, \Sigma, P, S \rangle$ — КС-грамматика. Тогда *решением задачи КС-достижимости* для G и Gr называется множество

$$\{\langle v_i, v_j \rangle \in V^2 \mid \exists \text{ путь } \pi \text{ в графе } G \text{ из } v_i \text{ в } v_j : \omega(\pi) \in \mathcal{L}(Gr)\},$$

отождествляемое с булевой матрицей $M \in \mathbb{B}^{n \times n}$ такой, что

$$M_{i,j} = \begin{cases} 1, \text{ если } \exists \text{ путь } \pi \text{ в графе } G \text{ из } v_i \text{ в } v_j : \omega(\pi) \in \mathcal{L}(Gr); \\ 0, \text{ иначе.} \end{cases}$$

Пример 1. Пусть $G = \langle V, E, L \rangle$ — это рёберно-меченый граф, показанный на рисунке 1, $Gr = \langle N, \Sigma, P, S \rangle$ — это КС-грамматика со следующими продукциями:

$$\forall i \in \{1, 2\} : S \to call_i \; ret_i;$$
$$\forall i \in \{1, 2\} : S \to call_i \; S \; ret_i \; S.$$

Тогда решением задачи КС-достижимости является множество

$$\{\langle x_1, y_1 \rangle, \langle x_2, y_2 \rangle\},$$

так как:

$$\exists \text{ путь } \pi_1 = \langle \langle x_1, call_1, x \rangle, \langle x, ret_1, y_1 \rangle \rangle \text{ и } \omega(\pi_1) = call_1 \; ret_1 \in \mathcal{L}(Gr);$$
$$\exists \text{ путь } \pi_2 = \langle \langle x_2, call_2, x \rangle, \langle x, ret_2, y_2 \rangle \rangle \text{ и } \omega(\pi_2) = call_2 \; ret_2 \in \mathcal{L}(Gr).$$

Замечание 1. Приведённое выше определение решения задачи КС-достижимости относится к семантике запросов отношения (англ. *relational query semantics*) для всех пар вершин (англ. *all pairs*). Для задачи КС-достижимости также определены семантики запросов одного пути (англ. *single path*), кратчайшего пути (англ. *shortest path*) и всех

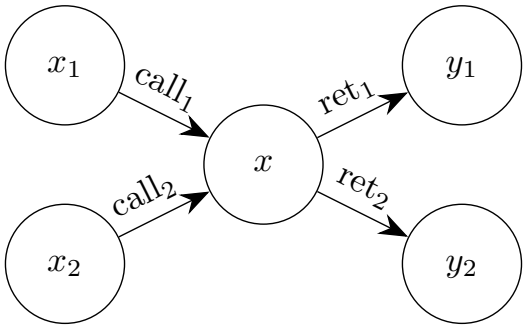

Рис. 1: Граф для анализа потока значений в программе, показанной на листинге 1 (источник: [24]).

```
def identity(x):
    return x

def main():
    y1 = identity(x1)
    y2 = identity(x2)
```

Листинг 1: Пример анализируемой Python-программы (источник: [24]).

путей (англ. *all paths*), а также запросы для одной (англ. *single source*) и нескольких (англ. *multiple source*) стартовых вершин.

2.2 Матричный алгоритм КС-достижимости

Согласно [24], лидером среди универсальных алгоритмом решения задачи КС-достижимости [4, 1, 14, 38] является оптимизированная версия предложенного Рустамом Азимовым [4] матричного алгоритма КС-достижимости (см. алгоритм 1). Данный алгоритм стремится достигнуть высокой производительности с помощью решения задачи в терминах операций линейный алгебры с разреженными булевыми матрицами, эффективно выполняемыми на современном оборудовании [36].

Матричный алгоритм КС-достижимости работает с КС-грамматиками в ослабленной нормальной форме Хомского определяемой следующим образом.

Определение 11. Пусть $Gr = \langle N, \Sigma, P, S \rangle$ — это КС-грамматика и

$$P \subseteq \{A \to \alpha \mid A \in N, \alpha \in \Sigma \cup \{\varepsilon\} \cup N^2\}.$$

Тогда говорят, что Gr находится в *ослабленной нормальной форме Хомского (ОНФХ).*

Замечание 2. По любой КС-грамматике Gr можно построить эквивалентную КС-грамматику $\widetilde{Gr}$ в ОНФХ, то есть такую КС-грамматику в ОНФХ, что $\mathcal{L}(Gr) = \mathcal{L}(\widetilde{Gr})$.

Алгоритм 1: Оригинальная версия матричного алгоритма КС-достижимости, предложенная Рустамом Азимовым [4].

Данные: КС-грамматика Gr, рёберно-меченый граф $G = \langle V, E, \Sigma \rangle$, где $V = \{v_1, \ldots, v_n\}$.
Результат: Матрица $M[S] \in \mathbb{B}^{n \times n}$ — решение задачи КС-достижимости для G и Gr.

$Gr \leftarrow$ КС-грамматика в ОНФХ, эквивалентная Gr
$\langle N, \Sigma, P, S \rangle \leftarrow Gr$
$M \leftarrow emptyDictionary()$ `// отображение из` N `в` $\mathbb{B}^{n \times n}$

для каждого $X \in N$ **выполнять**
  $M[X] \leftarrow 0^{n \times n}$
конец
для каждого $\langle v_i, x, v_j \rangle \in E, (X \to x) \in P$ **выполнять**
  $M[X]_{i,j} \leftarrow 1$
конец
для каждого $v_i \in V, (X \to \varepsilon) \in P$ **выполнять**
  $M[X]_{i,i} \leftarrow 1$
конец

пока M *меняется* **выполнять**
  для каждого $(Z \to X\ Y) \in P$ **выполнять**
    `// матричные операции над полукольцом` $\mathbb{B}$
    $M[Z] \leftarrow M[Z] \vee (M[X] \cdot M[Y])$
  конец
конец

2.3 Учёт паттернов в решениях

Для матричного алгоритма КС-достижимости было предложено множество оптимизаций [24, 19], сделавших его лидером по производительности среди универсальных алгоритмов КС-достижимости [24]. Однако в [24] также было отмечено, что даже при использовании всех

оптимизаций матричный алгоритм всё ещё может уступать более специализированным (лат. *ad hoc*) инструментам PEARL [39] и Gigascale [9].

В [24] объясняется, что это происходит, потому что разработчикам данных аналогов удалось распознать и использовать паттерны в ответах на те конкретные задачи, на которых их инструменты специализируются. Также делается вывод о том, что перспективным направлением дальнейшего развития универсальных КС-решателей является автоматическое распознавание подобных паттернов с помощью разложения матрицы M на сильно разреженные матрицы. Алгоритм 2 показывает, как такое разложение может использоваться в матричном алгоритме КС-достижимости.

При этом для использования алгоритма 2 необходимо разработать разлагатель решений задачи КС-достижимости, работающий существенно быстрее матричного решателя КС-достижимости и реализующий упомянутую на строке 14 функцию *decompose* такую, что если:

- $Gr = \langle N, \Sigma, P, S \rangle$ — КС-грамматика,
- $\forall X \in N : M[X] \in \mathbb{B}^{n \times m}$,
- $L', M', R' = decompose(M)$,

то:

- $L' \in \mathbb{B}^{n \times k}$,
- $R' \in \mathbb{B}^{l \times m}$,
- $\forall X \in N : M'[X] \in \mathbb{B}^{k \times l}$,
- $\forall X \in N : L' \cdot M'[X] \cdot R' = M[X]$,
- $nnz(L') + nnz(R') + \sum\limits_{X \in N} nnz(M'[X]) \ll \sum\limits_{X \in N} nnz(M[X])$, где $nnz(A)$ обозначает число ненулевых элементов в матрице A.

Алгоритм 2: Матричный алгоритм КС-достижимости с учётом паттернов в решениях.

Данные: КС-грамматика Gr, рёберно-меченый граф $G = \langle V, E, \Sigma \rangle$, где $V = \{v_1, \ldots, v_n\}$.

Результат: Матрицы $L \in \mathbb{B}^{n \times k}$, $M[S] \in \{0,1\}^{k \times l}$, $R \in \mathbb{B}^{l \times n}$ такие, что $L \cdot M \cdot R$ — решение задачи КС-достижимости для G и Gr.

$Gr \leftarrow$ КС-грамматика в ОНФХ, эквивалентная Gr
$\langle N, \Sigma, P, S \rangle \leftarrow Gr$
$M \leftarrow initializeM()$ `// Аналогично алгоритму 1`

`// Единичные матрицы` $n \times n$
$L \leftarrow \mathbb{I}_n$
$R \leftarrow \mathbb{I}_n$

пока *L, M или R меняется* **выполнять**
    для каждого $(Z \to X\ Y) \in P$ **выполнять**
        $M[Z] \leftarrow M[Z] \vee (M[X] \cdot R \cdot L \cdot M[Y])$
    конец
    если $should\_decompose(M)$ **тогда**
        $L', M', R' \leftarrow decompose(M)$
        $L \leftarrow L \cdot L'$
        $R \leftarrow R' \cdot R$
    конец
конец

2.4 Подходы к реализации разложения

В данном подразделе проводится обзор различных смежных задач и подходов к их решению и выявление среди них таких, которые могут быть полезны при реализации функции разложения *decompose*.

2.4.1 SD- и SVD-разложение

Задача, которую должна решать используемая в алгоритме 2 функция *decompose* смежна с задачами поиска скелетного разложения (англ. *skeletal decomposition, SD*) [18] и сингулярного разложения (англ. *singular value decomposition, SVD*) [20]. В контексте данной работы логичнее всего было бы использовать алгоритмы разложения для Булевых матриц, однако из-за их плохой масштабируемости [22] придётся остановится на разложение для вещественнозначных матриц.

Определение 12. Пусть $A \in \mathbb{R}^{n \times m}$. Тогда:

- SVD-разложением называется факторизация вида $A = U\Sigma V^T$, где $U \in \mathbb{R}^{n \times k}$ и $V \in \mathbb{R}^{k \times m}$ — это ортогональные матрицы;
- SD-разложением (также известным, как CUR-факторизация) называется факторизация вида $A = CUR$, где $C \in \mathbb{R}^{n \times k}$ и $R \in \mathbb{R}^{k \times m}$ — это матрицы, состоящие соответственно из строк и столбцов матрицы A.

К сожалению, точное построение подобных разложений для содержащих порядка миллиарда ненулевых элементов матриц, с которыми успешно работают существующие КС-решатели [24], непрактично даже для вещественных матриц.

Тем не менее существуют алгоритмы, позволяющие достаточно быстро построить приближённое разложение, с высокой вероятностью обеспечивающую небольшую ошибку $||A - CUR||_2$ [7]. Однако оценки ошибки при использовании подобных подходов даются для значения l^2-нормы, а не для значения $nnz(M - CUR)$, определяющего, насколько вычислительно затратно будет учитывать эту ошибку при дальнейших итерациях алгоритма 2.

2.4.2 Факторизация булевых матриц

Ещё одна весьма известная задача, смежная с задачей, решаемой функцией *decompose* — это задача факторизации булевых матриц (англ. *boolean matrix factorization, BMF*). BMF является NP-трудной задачей, что следует из [27]. Существующие алгоритмы решения BMF [23] масштабируются хуже, чем алгоритмы решения задачи КС-достижимости, из-за чего использование универсальных подходов к решению BMF в данном контексте нецелесообразно и необходима разработка специализированных подходов, учитывающих особенности матриц, возникающих в процессе решения задачи КС-достижимости.

2.4.3 Индексы для задачи достижимости

Одной из задач, связанных с задачей КС-достижимости, является крайне хорошо изученная обычная задача достижимости в ориентированном графе без ограничений на пути. В контексте выявления паттернов в решениях задачи КС-достижимости, наибольший интерес представляют подходы к построению индексов достижимости. Можно выделить три крупных класса подходов к индексированию решений задачи достижимости без ограничений на пути [40]:

- tree cover [2] — основывается на преобразовании графа в ориентированный ациклический граф, построению остовных деревьев и сопоставлении вершинам интервалов на основе обратного (англ. *post-order*) обхода этих деревьев;
- approximate TC [31, 28] — основывается на сопоставлении каждой вершине результата применения к множеству достижимых из неё вершин функции AP, сохраняющей отношение $\not\subseteq$;
- 2-hop [29] — основывается на построении такого графа-индекса $\widetilde{G} = \langle V, \widetilde{E} \rangle$, что пути в исходном графе $G = \langle V, E \rangle$ представимы в виде конкатенации не более чем двух рёбер графа-индекса $\widetilde{G}$.

Среди этих подходов на язык матриц хорошо переводятся лишь 2-hop, так как он эквивалентен факторизации матрицы достижимости.

К сожалению реализации 2-hop [11, 30, 37] полагаются на транзитивность отношения, соответствующего разлагаемой матрице, что в случае КС-достижимости не гарантируется.

2.4.4 Локально-чувствительное хеширование (LSH)

Ещё одним из подходов, используемым в некоторых работах для выявления паттернов является локально-чувствительное хеширование (англ. *locality-sensitive hashing, LSH*) [35], которое заключается в отображении элементов высокоразмерного пространства в компактные хеши таким образом, чтобы элементы, близкие в исходном пространстве, с высокой вероятностью имели одинаковые хеши.

Формально LSH-семейство хеш-функций определяется следующим образом.

Определение 13. Пусть:

- $\mathcal{M} = (M, d)$ — метрическое пространство;
- $\mathcal{F}$ — конечное семейство функций $h : M \to S$;
- $r > 0$ — порог;
- $c > 1$ — фактор аппроксимации;
- $p_1 > p_2$ — вероятности;
- h случайно и равномерно выбрано из $\mathcal{F}$, то есть $h \sim U(\mathcal{F})$, где U — это дискретное равномерное распределение;
- $\forall a, b \in M$:
  - $d(a, b) \leq r \implies Pr[h(a) = h(b)] \geq p_1$;
  - $d(a, b) \geq cr \implies Pr[h(a) = h(b)] \leq p_2$;

Тогда $\mathcal{F}$ называют (r, cr, p_1, p_2)-чувствительным LSH-семейством.

В контексте функции *decompose* в качестве элементов высокоразмерного пространства могут выступать строки (столбцы) разлагаемых матриц. Эти строки (столбцы) можно сгруппировать по их хешам в ячейки хеш-таблицы (англ. *bucket*) и затем применить поэлементное логического «И» к строкам (столбцам) в одной ячейке, чтобы получить для каждой ячейки хеш-таблицы одну строку-агрегатор (столбец-агрегатор). Полученные таким образом строки-агрегаторы (столбцы-агрегаторы) могут выступать в качестве первого приближения строк (столбцов) матрицы R' (L') в алгоритме 2.

2.4.5 Хеш-функции для булевых векторов

Для применения описанного в конце предыдущего параграфа подхода нужно выбрать семейство хеш-функций, работающих с разреженными булевыми векторами, коими являются строки (столбцы) разлагаемой матрицы.

Существует множество подходов к хешированию булевых векторов, среди наиболее популярных можно выделить:

- битовые выборки [17];
- MinHash [5];
- SimHash [6].

Битовые выборки малоприменимы в контексте разреженных векторов. Что же касается MinHash и SimHash, то неплохое их сравнение даётся в [33], где отмечается, что SimHash больше подходит для обнаружения не просто похожих, а практически идентичных элементов, поэтому маловероятно, что в контексте разложения решений задачи КС-достижимости данный подход к хешированию будет перспективным, хотя, возможно, попробовать его всё таки стоит. Подробнее остановимся на определении и свойствах оставшемся подходе, а именно на MinHash.

Определение 14 (MinHash)**.** Пусть:

- Ω — некоторое множество, называемое универсальным множеством;
- $h : \Omega \to \mathbb{N}$ — инъективное отображение;
- $\sigma : \Omega \to \Omega$ — случайно и равномерно выбранная перестановка Ω, то есть $\sigma \sim U(Aut(\Omega))$.

Тогда:

- $h_{min} : (2^{\Omega} \setminus \{\varnothing\}) \to \mathbb{N}$;
- $h_{min}(A) \stackrel{\text{def}}{=} \min\{h(\sigma(a)) \mid a \in A\}$.

В контексте булевых векторов длины n в качестве универсального множества можно взять множество $\Omega = \{1, \ldots, n\}$ и использовать следующее отображение из булевых векторов в универсальное множество:

$$v \mapsto \{i \mid v_i = 1\}.$$

Из свойств MinHash стоит отметить, что MinHash позволяет быстро оценить коэффициент Жаккара — часто используемый индикатор схожести двух множеств.

Определение 15 (коэффициент Жаккара)**.** Пусть A и B — непустые множества, тогда:

$$J(A, B) \stackrel{\text{def}}{=} \frac{|A \cap B|}{|A \cup B|}.$$

Замечание 3 (связь MinHash и коэффициента Жаккара)**.**

$$Pr[h_{min}(A) = h_{min}(B)] = J(A, B).$$

2.5 Выводы

Для ускорения матричного алгоритма КС-достижимости с помощью учёта паттернов в решениях необходимо разработать разлагатель решений задачи КС-достижимости, работающую существенно быстрее матричного решателя задачи КС-достижимости и позволяющую найти кор-

ректное разреженное разложение решения задачи КС-достижимости вида $L \cdot M'[\bullet] \cdot R$ (см. раздел 2.3).

Задача, решаемая разлагателем, связана со многими хорошо исследованными задачами. Тем не менее в контексте данной работы в силу разных причин переиспользовать представляется возможным лишь весьма ограниченную часть подходов, применяемых для решения этих связанных задач. Среди рассмотренных подходов (см. раздел 2.4) наиболее подходящим для реализации разлагателя решений задачи КС-достижимости является использование LSH на основе MinHash (или на основе SimHash) для обнаружения групп схожих столбцов и строк в матрицах, описывающих решение задачи КС-достижимости.

3 Архитектура

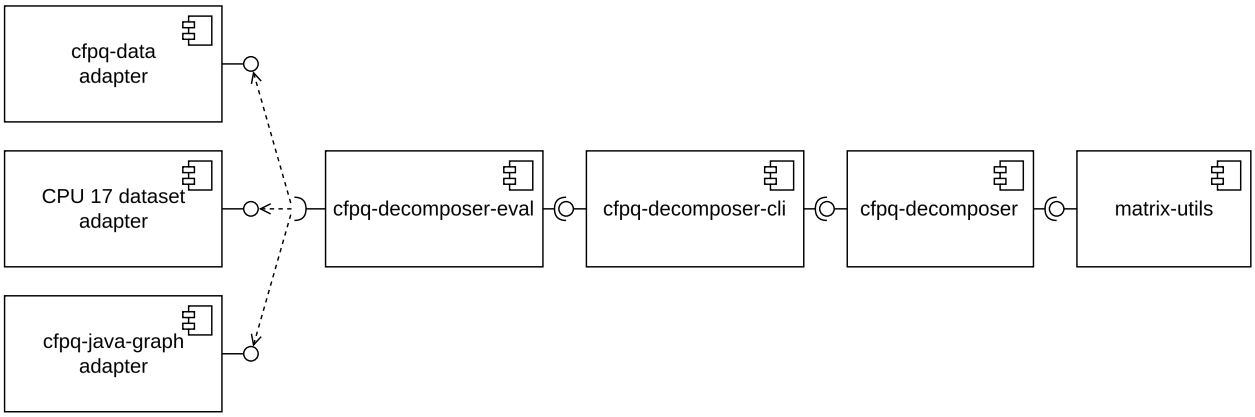

Рис. 2: Диаграмма компонентов разлагателя решений задачи КС-достижимости и системы оценки качества.

Перед реализацией разлагателя решений задачи КС-достижимости необходимо было спроектировать его архитектуру, а также архитектуру системы оценки качества работы разрабатываемого инструмента. При проектировании архитектуры были учтены следующие соображения.

1. Для упрощения разработки, поддержки и понимания решения желательно выделить несколько уровней абстракции.

2. Для запуска разработанного разлагателя желательно реализовать интерфейс командной строки (наиболее типичный способ запуска среди инструментов рассмотренных в разделе 2).

3. Для уменьшения связанности систему оценки качество желательно оформить в виде отдельного инструмента.

4. Для упрощения поддержки различных наборов данных, желательно, чтобы система оценки качества работала с некоторым абстрактным набором данных, допускающим несколько реализаций.

Исходя из вышеперечисленных соображений была спроектирована архитектура, представленная на рисунке 2, в частности были выделены следующие компоненты.

1. `matrix-utils` — компонент, содержащий вспомогательные функции для работы с матрицами, не являющиеся специфичными для предметной области данной работы, потенциально пригодные для переиспользования в работах, несвязанных с задачей КС-достижимости.

2. `cfpq-decomposer` — компонент, содержащий реализацию алгоритма разложения решений задачи КС-достижимости.

3. `cfpq-decomposer-cli` — компонент, отвечающий за интерфейс командной строки.

4. `cfpq-eval` — компонент, отвечающий за оценку качества разлагателя решений задачи КС-достижимости, способный интегрироваться с различными наборами данных.

5. Несколько компонентов, отвечающих за адаптирование различных наборов данных к интерфейсу, совместимому с компонентом `cfpq-eval`. Показанные на диаграмме наборы данных приведены в качестве примеров.

Стоит отметить, что представленная на рисунке 2 диаграмма компонентов отражает желаемую архитектуру ещё нереализованной системы и в ходе реализации могут вскрыться детали, существенно затрудняющие изоляцию друг от друга уровней абстракции, не являющихся соседними. Другими словами, может обнаружится, что, например, прямое использование `matrix-utils` из `cfpq-decomposer-eval` значительно упрощает реализацию.

Заключение

В ходе работы в первом семестре был достигнут следующий результат.

1. По итогам обзора подходов, применяемых для решения задач, смежных с задачей разложения решений задачи КС-достижимости, решено, что для реализации разложения больше всего подходит использование локально-чувствительного хеширования (LSH) на основе MinHash (или, возможно, на основе SimHash).

2. Спроектирована архитектура разлагателя решений задачи КС-достижимости и системы оценки качества.

Также были уточнены задачи на следующие семестры.

1. Предложить алгоритм разложения решений задачи КС-достижимости на основе LSH и оценить его временную сложность.

2. Реализовать и протестировать разлогатель решений задачи КС-достижимости на основе предложенного алгоритма.

3. Экспериментально сравнить на реальных данных производительность разработанного разлогателя и решателей задачи КС-достижимости и измерить достигаемую разлогателем степень сжатия решений задачи КС-достижимости.

Список литературы

[1] Abzalov Vadim. Implementation and experimental investigation of the GLL parser based on a recursive automaton. — 2023. — URL: http://hdl.handle.net/11701/42730 (дата обращения: 13 декабря 2024 г.).

[2] Agrawal R., Borgida A., Jagadish H. V. Efficient management of transitive relationships in large data and knowledge bases // Proceedings of the 1989 ACM SIGMOD International Conference on Management of Data. — SIGMOD '89. — New York, NY, USA : Association for Computing Machinery, 1989. — P. 253–262. — URL: https://doi.org/10.1145/67544.66950 (дата обращения: 13 декабря 2024 г.).

[3] Azimov Rustam. Context-Free Path Querying Using Linear Algebra : Ph. D. thesis / Rustam Azimov ; St. Petersburg State University. — 2022. — URL: https://disser.spbu.ru/files/2022/disser_azimov.pdf (дата обращения: 13 декабря 2024 г.).

[4] Azimov Rustam, Grigorev Semyon. Context-Free Path Querying by Matrix Multiplication // Proceedings of the 1st ACM SIGMOD Joint International Workshop on Graph Data Management Experiences & Systems (GRADES) and Network Data Analytics (NDA). — GRADES-NDA '18. — New York, NY, USA : Association for Computing Machinery, 2018. — 10 p. — URL: https://doi.org/10.1145/3210259.3210264 (дата обращения: 13 декабря 2024 г.).

[5] Broder A.Z. On the resemblance and containment of documents // Proceedings. Compression and Complexity of SEQUENCES 1997 (Cat. No.97TB100171). — 1997. — P. 21–29. — URL: http://dx.doi.org/10.1109/SEQUEN.1997.666900 (дата обращения: 13 декабря 2024 г.).

[6] Charikar Moses S. Similarity estimation techniques from rounding algorithms // Proceedings of the Thiry-Fourth Annual ACM Symposium on Theory of Computing. — STOC '02. — New York, NY,

USA : Association for Computing Machinery, 2002. — P. 380–388. — URL: https://doi.org/10.1145/509907.509965 (дата обращения: 13 декабря 2024 г.).

[7] Chiu Jiawei, Demanet Laurent. Sublinear Randomized Algorithms for Skeleton Decompositions // SIAM Journal on Matrix Analysis and Applications. — 2013. — Vol. 34, no. 3. — P. 1361–1383. — URL: https://doi.org/10.1137/110852310 (дата обращения: 13 декабря 2024 г.).

[8] Context-Free Path Queries on RDF Graphs / Xiaowang Zhang, Zhiyong Feng, Xin Wang et al. // The Semantic Web – ISWC 2016 / Ed. by Paul Groth, Elena Simperl, Alasdair Gray et al. — Cham : Springer International Publishing, 2016. — P. 632–648. — URL: https://link.springer.com/chapter/10.1007/978-3-319-46523-4_38 (дата обращения: 13 декабря 2024 г.).

[9] Dietrich Jens, Hollingum Nicholas, Scholz Bernhard. Giga-Scale Exhaustive Points-to Analysis for Java in under a Minute // Proceedings of the 2015 ACM SIGPLAN International Conference on Object-Oriented Programming, Systems, Languages, and Applications. — OOPSLA 2015. — New York, NY, USA : Association for Computing Machinery, 2015. — P. 535–551. — URL: https://doi.org/10.1145/2814270.2814307 (дата обращения: 13 декабря 2024 г.).

[10] Effective Typestate Verification in the Presence of Aliasing / Stephen J. Fink, Eran Yahav, Nurit Dor et al. // ACM Trans. Softw. Eng. Methodol. — 2008. — may. — Vol. 17, no. 2. — 34 p. — URL: https://doi.org/10.1145/1348250.1348255 (дата обращения: 13 декабря 2024 г.).

[11] Fast and scalable reachability queries on graphs by pruned labeling with landmarks and paths / Yosuke Yano, Takuya Akiba, Yoichi Iwata, Yuichi Yoshida // Proceedings of the 22nd ACM International Conference on Information & Knowledge Management. — CIKM '13. — New York, NY, USA : Association for Computing Machinery, 2013. —

P. 1601–1606. — URL: https://doi.org/10.1145/2505515.2505724 (дата обращения: 13 декабря 2024 г.).

[12] FlowDroid: Precise Context, Flow, Field, Object-Sensitive and Lifecycle-Aware Taint Analysis for Android Apps / Steven Arzt, Siegfried Rasthofer, Christian Fritz et al. // SIGPLAN Not. — 2014. — jun. — Vol. 49, no. 6. — P. 259–269. — URL: https://doi.org/10.1145/2666356.2594299 (дата обращения: 13 декабря 2024 г.).

[13] FlowTwist: Efficient Context-Sensitive inside-out Taint Analysis for Large Codebases / Johannes Lerch, Ben Hermann, Eric Bodden, Mira Mezini // Proceedings of the 22nd ACM SIGSOFT International Symposium on Foundations of Software Engineering. — FSE 2014. — New York, NY, USA : Association for Computing Machinery, 2014. — P. 98–108. — URL: https://doi.org/10.1145/2635868.2635878 (дата обращения: 13 декабря 2024 г.).

[14] Graspan: A Single-Machine Disk-Based Graph System for Interprocedural Static Analyses of Large-Scale Systems Code / Kai Wang, Aftab Hussain, Zhiqiang Zuo et al. // SIGPLAN Not. — 2017. — apr. — Vol. 52, no. 4. — P. 389–404. — URL: https://doi.org/10.1145/3093336.3037744 (дата обращения: 13 декабря 2024 г.).

[15] Grech Neville, Smaragdakis Yannis. P/Taint: Unified Points-to and Taint Analysis // Proc. ACM Program. Lang. — 2017. — oct. — Vol. 1, no. OOPSLA. — 28 p. — URL: https://doi.org/10.1145/3133926 (дата обращения: 13 декабря 2024 г.).

[16] Hovemeyer David, Pugh William. Finding Bugs is Easy // Companion to the 19th Annual ACM SIGPLAN Conference on Object-Oriented Programming Systems, Languages, and Applications. — OOPSLA '04. — New York, NY, USA : Association for Computing Machinery, 2004. — P. 132–136. — URL: https://doi.org/10.1145/1028664.1028717 (дата обращения: 13 декабря 2024 г.).

[17] Indyk Piotr, Motwani Rajeev. Approximate nearest neighbors: towards removing the curse of dimensionality // Proceedings of the Thirtieth Annual ACM Symposium on Theory of Computing. — STOC '98. — New York, NY, USA : Association for Computing Machinery, 1998. — P. 604–613. — URL: https://doi.org/10.1145/276698.276876 (дата обращения: 13 декабря 2024 г.).

[18] Klema V., Laub A. The singular value decomposition: Its computation and some applications // IEEE Transactions on Automatic Control. — 1980. — Vol. 25, no. 2. — P. 164–176. — URL: http://dx.doi.org/10.1109/TAC.1980.1102314 (дата обращения: 13 декабря 2024 г.).

[19] Kutuev Vladimir. Experimental investigation of context-free-language reachability algorithms as applied to static code analysis. — 2023. — URL: http://hdl.handle.net/11701/42628 (дата обращения: 13 декабря 2024 г.).

[20] Mahoney Michael W., Drineas Petros. CUR matrix decompositions for improved data analysis // Proceedings of the National Academy of Sciences. — 2009. — Vol. 106, no. 3. — P. 697–702. — URL: https://www.pnas.org/doi/abs/10.1073/pnas.0803205106 (дата обращения: 13 декабря 2024 г.).

[21] Martin Michael, Livshits Benjamin, Lam Monica S. Finding Application Errors and Security Flaws Using PQL: A Program Query Language // Proceedings of the 20th Annual ACM SIGPLAN Conference on Object-Oriented Programming, Systems, Languages, and Applications. — OOPSLA '05. — New York, NY, USA : Association for Computing Machinery, 2005. — P. 365–383. — URL: https://doi.org/10.1145/1094811.1094840 (дата обращения: 13 декабря 2024 г.).

[22] Miettinen Pauli. The Boolean column and column-row matrix decompositions // Data Mining and Knowledge Discovery. — 2008. — Vol. 17, no. 1. — P. 39–56. — URL: https://doi.org/10.1007/s10618-008-0107-0 (дата обращения: 13 декабря 2024 г.).

[23] Miettinen Pauli, Neumann Stefan. Recent developments in Boolean matrix factorization // Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence. — IJCAI'20. — 2021. — 7 p. — URL: https://dl.acm.org/doi/10.5555/3491440.3492125 (дата обращения: 13 декабря 2024 г.).

[24] Muravev Ilia. Optimisation of the context-free language reachability matrix-based algorithm. — 2024. — URL: https://dspace.spbu.ru/handle/11701/46282 (дата обращения: 13 декабря 2024 г.).

[25] Nanda Mangala Gowri, Sinha Saurabh. Accurate Interprocedural Null-Dereference Analysis for Java // 2009 IEEE 31st International Conference on Software Engineering. — 2009. — P. 133–143. — URL: https://doi.org/10.1109/ICSE.2009.5070515 (дата обращения: 13 декабря 2024 г.).

[26] Zhang Wenjie, Wang Guancheng, Chen Junjie et al. OrdinalFix: Fixing Compilation Errors via Shortest-Path CFL Reachability. — 2023. — URL: https://arxiv.org/abs/2309.06771 (дата обращения: 13 декабря 2024 г.).

[27] Orlin James. Contentment in graph theory: Covering graphs with cliques // Indagationes Mathematicae (Proceedings). — 1977. — Vol. 80, no. 5. — P. 406–424. — URL: https://www.sciencedirect.com/science/article/pii/1385725877900555 (дата обращения: 13 декабря 2024 г.).

[28] Reachability Querying: Can It Be Even Faster? / Jiao Su, Qing Zhu, Hao Wei, Jeffrey Xu Yu // IEEE Transactions on Knowledge and Data Engineering. — 2017. — Vol. 29, no. 3. — P. 683–697. — URL: http://dx.doi.org/10.1109/TKDE.2016.2631160 (дата обращения: 13 декабря 2024 г.).

[29] Reachability and Distance Queries via 2-Hop Labels / Edith Cohen, Eran Halperin, Haim Kaplan, Uri Zwick // SIAM Journal on Computing. — 2003. — Vol. 32, no. 5. — P. 1338–1355. — URL: https://doi.

org/10.1137/S0097539702403098 (дата обращения: 13 декабря 2024 г.).

[30] Reachability queries on large dynamic graphs: a total order approach / Andy Diwen Zhu, Wenqing Lin, Sibo Wang, Xiaokui Xiao // Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data. — SIGMOD '14. — New York, NY, USA : Association for Computing Machinery, 2014. — P. 1323–1334. — URL: https://doi.org/10.1145/2588555.2612181 (дата обращения: 13 декабря 2024 г.).

[31] Reachability querying: an independent permutation labeling approach / Hao Wei, Jeffrey Xu Yu, Can Lu, Ruoming Jin // The VLDB Journal. — 2018. — Vol. 27, no. 1. — P. 1–26. — URL: https://doi.org/10.1007/s00778-017-0468-3 (дата обращения: 13 декабря 2024 г.).

[32] Sevon Petteri, Eronen Lauri. Subgraph Queries by Context-free Grammars // Journal of Integrative Bioinformatics. — 2008. — Vol. 5, no. 2. — P. 157 – 172. — URL: https://www.degruyter.com/view/journals/jib/5/2/article-p157.xml (дата обращения: 13 декабря 2024 г.).

[33] Shrivastava Anshumali, Li Ping. In Defense of Minhash over Simhash // Proceedings of the Seventeenth International Conference on Artificial Intelligence and Statistics / Ed. by Samuel Kaski, Jukka Corander. — Vol. 33 of Proceedings of Machine Learning Research. — Reykjavik, Iceland : PMLR, 2014. — 22–25 Apr. — P. 886–894. — URL: https://proceedings.mlr.press/v33/shrivastava14.html (дата обращения: 13 декабря 2024 г.).

[34] Static Data Race Detection for Concurrent Programs with Asynchronous Calls / Vineet Kahlon, Nishant Sinha, Erik Kruus, Yun Zhang // Proceedings of the 7th Joint Meeting of the European Software Engineering Conference and the ACM SIGSOFT Sym-

posium on The Foundations of Software Engineering. — ESEC/FSE '09. — New York, NY, USA : Association for Computing Machinery, 2009. — P. 13–22. — URL: https://doi.org/10.1145/1595696.1595701 (дата обращения: 13 декабря 2024 г.).

[35] A Survey on Locality Sensitive Hashing Algorithms and their Applications / Omid Jafari, Preeti Maurya, Parth Nagarkar et al. // ArXiv. — 2021. — Vol. abs/2102.08942. — URL: https://api.semanticscholar.org/CorpusID:231942424 (дата обращения: 13 декабря 2024 г.).

[36] A Systematic Survey of General Sparse Matrix-Matrix Multiplication / Jianhua Gao, Weixing Ji, Fangli Chang et al. // ACM Comput. Surv. — 2023. — mar. — Vol. 55, no. 12. — 36 p. — URL: https://doi.org/10.1145/3571157 (дата обращения: 13 декабря 2024 г.).

[37] TF-Label: a topological-folding labeling scheme for reachability querying in a large graph / James Cheng, Silu Huang, Huanhuan Wu, Ada Wai-Chee Fu // Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data. — SIGMOD '13. — New York, NY, USA : Association for Computing Machinery, 2013. — P. 193–204. — URL: https://doi.org/10.1145/2463676.2465286 (дата обращения: 13 декабря 2024 г.).

[38] Taming Transitive Redundancy for Context-Free Language Reachability / Yuxiang Lei, Yulei Sui, Shuo Ding, Qirun Zhang // Proc. ACM Program. Lang. — 2022. — oct. — Vol. 6, no. OOPSLA2. — 27 p. — URL: https://doi.org/10.1145/3563343 (дата обращения: 13 декабря 2024 г.).

[39] Two Birds with One Stone: Multi-Derivation for Fast Context-Free Language Reachability Analysis / C. Shi, H. Li, Y. Sui et al. // 2023 38th IEEE/ACM International Conference on Automated Software Engineering (ASE). — Los Alamitos, CA, USA : IEEE Computer Society, 2023. — sep. — P. 624–636. —

URL: https://doi.ieeecomputersociety.org/10.1109/ASE56229.2023.00118 (дата обращения: 13 декабря 2024 г.).

[40] Zhang Chao, Bonifati Angela, Özsu M. Tamer. An Overview of Reachability Indexes on Graphs // Companion of the 2023 International Conference on Management of Data. — SIGMOD '23. — New York, NY, USA : Association for Computing Machinery, 2023. — P. 61–68. — URL: https://doi.org/10.1145/3555041.3589408 (дата обращения: 13 декабря 2024 г.).

[41] Zheng Xin, Rugina Radu. Demand-Driven Alias Analysis for C // SIGPLAN Not. — 2008. — jan. — Vol. 43, no. 1. — P. 197–208. — URL: https://doi.org/10.1145/1328897.1328464 (дата обращения: 13 декабря 2024 г.).