Классификация алгоритмов построения прямолинейных программ

Евгений Курпилянский

11 июня 2014 года

Способы сжатия

Существует различные способы сжатия данных, например:

- прямолинейные программы;
- антисловари;
- коллаж-системы и др.

Если сжатое представление хорошо **структурировано**, то существуют алгоритмы, способные решать классические задачи **без распаковки** данных.

Примеры таких задач

- поиск подстроки в строке;
- наибольшая общая подстрока;
- и др.

Способы сжатия

Существует различные способы сжатия данных, например:

- прямолинейные программы;
- антисловари;
- коллаж-системы и др.

Если сжатое представление хорошо **структурировано**, то существуют алгоритмы, способные решать классические задачи **без распаковки** данных.

Примеры таких задач

- поиск подстроки в строке;
- наибольшая общая подстрока;
- и др.

Способы сжатия

Существует различные способы сжатия данных, например:

- прямолинейные программы;
- антисловари;
- коллаж-системы и др.

Если сжатое представление хорошо **структурировано**, то существуют алгоритмы, способные решать классические задачи **без распаковки** данных.

Примеры таких задач

- поиск подстроки в строке;
- наибольшая общая подстрока;
- и др.

Вопросы

1. Умеем ли мы эффективно строить маленькие ПП?
2. Насколько эффективно решение классических задач в терминах ПП?

Данная работа посвящена первому вопросу.

Вопросы

1. Умеем ли мы эффективно строить маленькие ПП?
2. Насколько эффективно решение классических задач в терминах ПП?

Данная работа посвящена первому вопросу.

Вопросы

1. Умеем ли мы эффективно строить маленькие ПП?
2. Насколько эффективно решение классических задач в терминах ПП?

Данная работа посвящена первому вопросу.

Определение прямолинейной программы

Определение

Прямолинейная программа (ПП) строки S – это контекстно-свободная грамматика в нормальной форме Хомского, выводящая в точности одно слово S.

Пример

Рассмотрим ПП X, выводящую строку «*abaababaabaab*».

$$
\begin{aligned}
X_1 &\to a \\
X_2 &\to b \\
X_3 &\to X_1 \cdot X_2 \\
X_4 &\to X_3 \cdot X_1 \\
X_5 &\to X_4 \cdot X_3 \\
X_6 &\to X_5 \cdot X_4 \\
X_7 &\to X_6 \cdot X_5
\end{aligned}
$$

Определение прямолинейной программы

Определение

Прямолинейная программа (ПП) строки S – это контекстно-свободная грамматика в нормальной форме Хомского, выводящая в точности одно слово S.

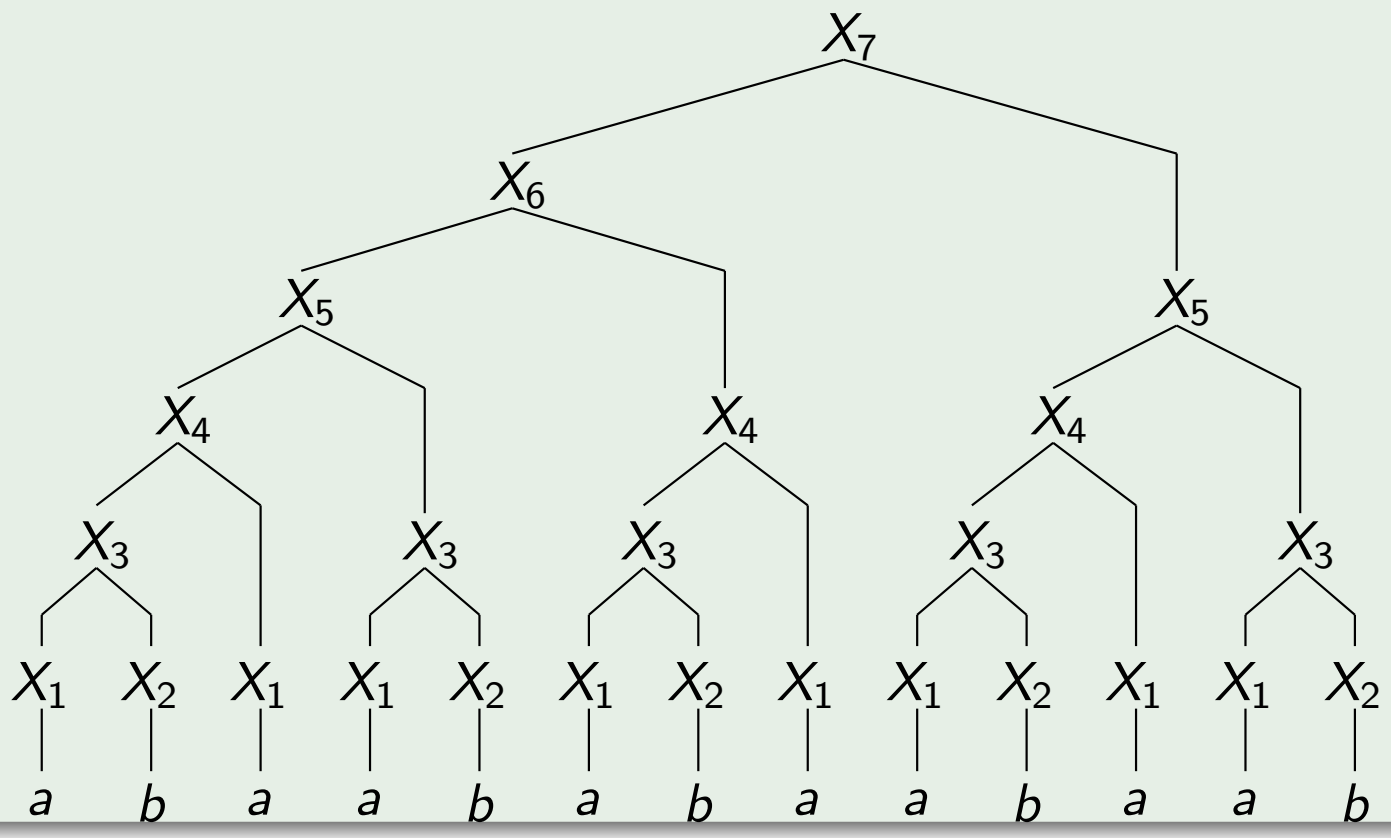
Пример

Рассмотрим ПП X, выводящую строку «*abaababaabaab*».

$$
\begin{aligned}
X_1 &\to a \\
X_2 &\to b \\
X_3 &\to X_1 \cdot X_2 \\
X_4 &\to X_3 \cdot X_1 \\
X_5 &\to X_4 \cdot X_3 \\
X_6 &\to X_5 \cdot X_4 \\
X_7 &\to X_6 \cdot X_5
\end{aligned}
$$

Пример

Графическое изображение ПП:



Как строить ПП?

Утверждение.

Задача построения минимальной ПП, выводящей заданную строку S – NP-трудная.

Для построения ПП требуется использовать приближенные алгоритмы.

Как строить ПП?

Утверждение.

Задача построения минимальной ПП, выводящей заданную строку S – NP-трудная.

$$\Downarrow$$

Для построения ПП требуется использовать приближенные алгоритмы.

Определение

Факторизация строки S – это набор строк w_1, w_2, ..., w_k такой, что $S = w_1 \cdot w_2 \cdot \ldots \cdot w_k$.

Определение

LZ-**факторизация** строки S — это факторизация $S = w_1 \cdot w_2 \cdots w_k$ такая, что для любого $j \in 1..k$

- w_j состоит из одной буквы, не встречающейся в $w_1 \cdot w_2 \cdots w_{j-1}$; или
- w_j — наибольший префикс $w_j \cdot w_{j+1} \cdots w_k$, встречающийся в $w_1 \cdot w_2 \cdots w_{j-1}$.

Факторизации строки «*abaababaabaab*»

- $a \cdot b \cdot a \cdot a \cdot b \cdot a \cdot b \cdot a \cdot a \cdot b \cdot a \cdot a \cdot b$;
- $a \cdot b \cdot a \cdot aba \cdot baaba \cdot ab$ (LZ-факторизация);

Определение

Факторизация строки S – это набор строк w_1, w_2, ..., w_k такой, что $S = w_1 \cdot w_2 \cdot \ldots \cdot w_k$.

Определение

LZ-**факторизация** строки S — это факторизация $S = w_1 \cdot w_2 \cdots w_k$ такая, что для любого $j \in 1..k$

- w_j состоит из одной буквы, не встречающейся в $w_1 \cdot w_2 \cdots w_{j-1}$; или
- w_j — наибольший префикс $w_j \cdot w_{j+1} \cdots w_k$, встречающийся в $w_1 \cdot w_2 \cdots w_{j-1}$.

Факторизации строки «*abaababaabaab*»

- $a \cdot b \cdot a \cdot a \cdot b \cdot a \cdot b \cdot a \cdot a \cdot b \cdot a \cdot a \cdot b$;
- $a \cdot b \cdot a \cdot aba \cdot baaba \cdot ab$ (LZ-факторизация);

Нижняя оценка (Риттер, 2001)

Размер минимальной ПП, выводящей данный текст, не меньше размера LZ-факторизации этого текста.

Постановка задачи

Вход: Строка T.
Выход: ПП, выводящая строку T.

Постановка задачи 2

Вход: Строка T **и ее LZ-факторизация** $F_1, F_2, \ldots, F_k$.
Выход: ПП, выводящая строку T.

Нижняя оценка (Риттер, 2001)

Размер минимальной ПП, выводящей данный текст, не меньше размера LZ-факторизации этого текста.

Постановка задачи

Вход: Строка T.
Выход: ПП, выводящая строку T.

Постановка задачи 2

Вход: Строка T **и ее LZ-факторизация** $F_1, F_2, \ldots, F_k$.
Выход: ПП, выводящая строку T.

Нижняя оценка (Риттер, 2001)

Размер минимальной ПП, выводящей данный текст, не меньше размера LZ-факторизации этого текста.

Постановка задачи

Вход: Строка T.
Выход: ПП, выводящая строку T.

Постановка задачи 2

Вход: Строка T **и ее LZ-факторизация** $F_1, F_2, \ldots, F_k$.
Выход: ПП, выводящая строку T.

Постановка задачи

Вход: Строка T и ее LZ-факторизация $F_1, F_2, \ldots, F_k$.
Выход: ПП, выводящая строку T.

Алгоритмы построения, основанные на сбалансированных бинарных деревьях:

- Алгоритм Риттера **(W. Rytter, 2001)**
- Эвристическая оптимизация **(И. Бурмистров, А. Хворост, 2011)**
- Небольшое обобщение эвристики New!
- Алгоритм построения рандомизированных ПП **(Е. Курпилянский, 2012)**
- Многопоточный алгоритм New!

Постановка задачи

Вход: Строка T и ее LZ-факторизация $F_1, F_2, \ldots, F_k$.
Выход: ПП, выводящая строку T.

Алгоритмы построения, основанные на сбалансированных бинарных деревьях:

- Алгоритм Риттера **(W. Rytter, 2001)**
- Эвристическая оптимизация **(И. Бурмистров, А. Хворост, 2011)**
- Небольшое обобщение эвристики New!
- Алгоритм построения рандомизированных ПП **(Е. Курпилянский, 2012)**
- Многопоточный алгоритм New!

Постановка задачи

Вход: Строка T и ее LZ-факторизация $F_1, F_2, \ldots, F_k$.
Выход: ПП, выводящая строку T.

Алгоритмы построения, основанные на сбалансированных бинарных деревьях:

- Алгоритм Риттера **(W. Rytter, 2001)**
- Эвристическая оптимизация **(И. Бурмистров, А. Хворост, 2011)**
- Небольшое обобщение эвристики New!
- Алгоритм построения рандомизированных ПП **(Е. Курпилянский, 2012)**
- Многопоточный алгоритм New!

Постановка задачи

Вход: Строка T и ее LZ-факторизация $F_1, F_2, \ldots, F_k$.
Выход: ПП, выводящая строку T.

Алгоритмы построения, основанные на сбалансированных бинарных деревьях:

- Алгоритм Риттера **(W. Rytter, 2001)**
- Эвристическая оптимизация **(И. Бурмистров, А. Хворост, 2011)**
- Небольшое обобщение эвристики New!
- Алгоритм построения рандомизированных ПП **(Е. Курпилянский, 2012)**
- Многопоточный алгоритм New!

Постановка задачи

Вход: Строка T и ее LZ-факторизация $F_1, F_2, \ldots, F_k$.
Выход: ПП, выводящая строку T.

Алгоритмы построения, основанные на сбалансированных бинарных деревьях:

- Алгоритм Риттера **(W. Rytter, 2001)**
- Эвристическая оптимизация **(И. Бурмистров, А. Хворост, 2011)**
- Небольшое обобщение эвристики New!
- Алгоритм построения рандомизированных ПП **(Е. Курпилянский, 2012)**
- Многопоточный алгоритм New!

Постановка задачи

Вход: Строка T и ее LZ-факторизация $F_1, F_2, \ldots, F_k$.
Выход: ПП, выводящая строку T.

Алгоритмы построения, основанные на сбалансированных бинарных деревьях:

- Алгоритм Риттера **(W. Rytter, 2001)**
- Эвристическая оптимизация **(И. Бурмистров, А. Хворост, 2011)**
- Небольшое обобщение эвристики New!
- Алгоритм построения рандомизированных ПП **(Е. Курпилянский, 2012)**
- Многопоточный алгоритм New!

Алгоритмы построения, основанные на сбалансированных бинарных деревьях:

- Алгоритм Риттера **(W. Rytter, 2001)**
- Эвристическая оптимизация **(И. Бурмистров, А. Хворост, 2011)**
- Небольшое обобщение эвристики New!
- Алгоритм построения рандомизированных ПП **(Е. Курпилянский, 2012)**
- Многопоточный алгоритм New!

Характеристики

Все эти алгоритмы:

- строят $O(\log n)$-приближение к минимальной;
- требуют на вход LZ-факторизацию текста.

Алгоритмы построения, основанные на сбалансированных бинарных деревьях:

- Алгоритм Риттера **(W. Rytter, 2001)**
- Эвристическая оптимизация **(И. Бурмистров, А. Хворост, 2011)**
- Небольшое обобщение эвристики New!
- Алгоритм построения рандомизированных ПП **(Е. Курпилянский, 2012)**
- Многопоточный алгоритм New!

Характеристики

Все эти алгоритмы:

- строят $O(\log n)$-приближение к минимальной;
- требуют на вход LZ-факторизацию текста.

Постановка задачи

Вход: Строка T.
Выход: ПП, выводящая строку T.

Алгоритмы построения ПП, не требующие построения LZ-факторизации.

Самый лучший алгоритм по качеству сжатия имеет оценку $O(\log^2 n)$.
LCA-online алгоритм **(S. Maruyama, H. Sakamoto, M. Takeda, 2012)**

Постановка задачи

Вход: Строка T.
Выход: ПП, выводящая строку T.

Алгоритмы построения ПП, не требующие построения LZ-факторизации.

Самый лучший алгоритм по качеству сжатия имеет оценку $O(\log^2 n)$.
LCA-online алгоритм **(S. Maruyama, H. Sakamoto, M. Takeda, 2012)**

Постановка задачи

Вход: Строка T.
Выход: ПП, выводящая строку T.

Алгоритмы построения ПП, не требующие построения LZ-факторизации.

Самый лучший алгоритм по качеству сжатия имеет оценку $O(\log^2 n)$.
LCA-online алгоритм **(S. Maruyama, H. Sakamoto, M. Takeda, 2012)**

Алгоритмы построения ПП, не требующие построения LZ-факторизации.

LCA-online алгоритм **(S. Maruyama, H. Sakamoto, M. Takeda, 2012)**

Характеристики

Данный алгоритм:

- строит $O(\log^2 n)$-приближение к минимальной;
- требует на вход только сам текст;
- работает online.

Алгоритмы построения ПП, не требующие построения LZ-факторизации.

LCA-online алгоритм **(S. Maruyama, H. Sakamoto, M. Takeda, 2012)**

Характеристики

Данный алгоритм:

- строит $O(\log^2 n)$-приближение к минимальной;
- требует на вход только сам текст;
- работает online.

Алгоритмы построения ПП, не требующие построения LZ-факторизации.

LCA-online алгоритм **(S. Maruyama, H. Sakamoto, M. Takeda, 2012)**

Характеристики

Данный алгоритм:

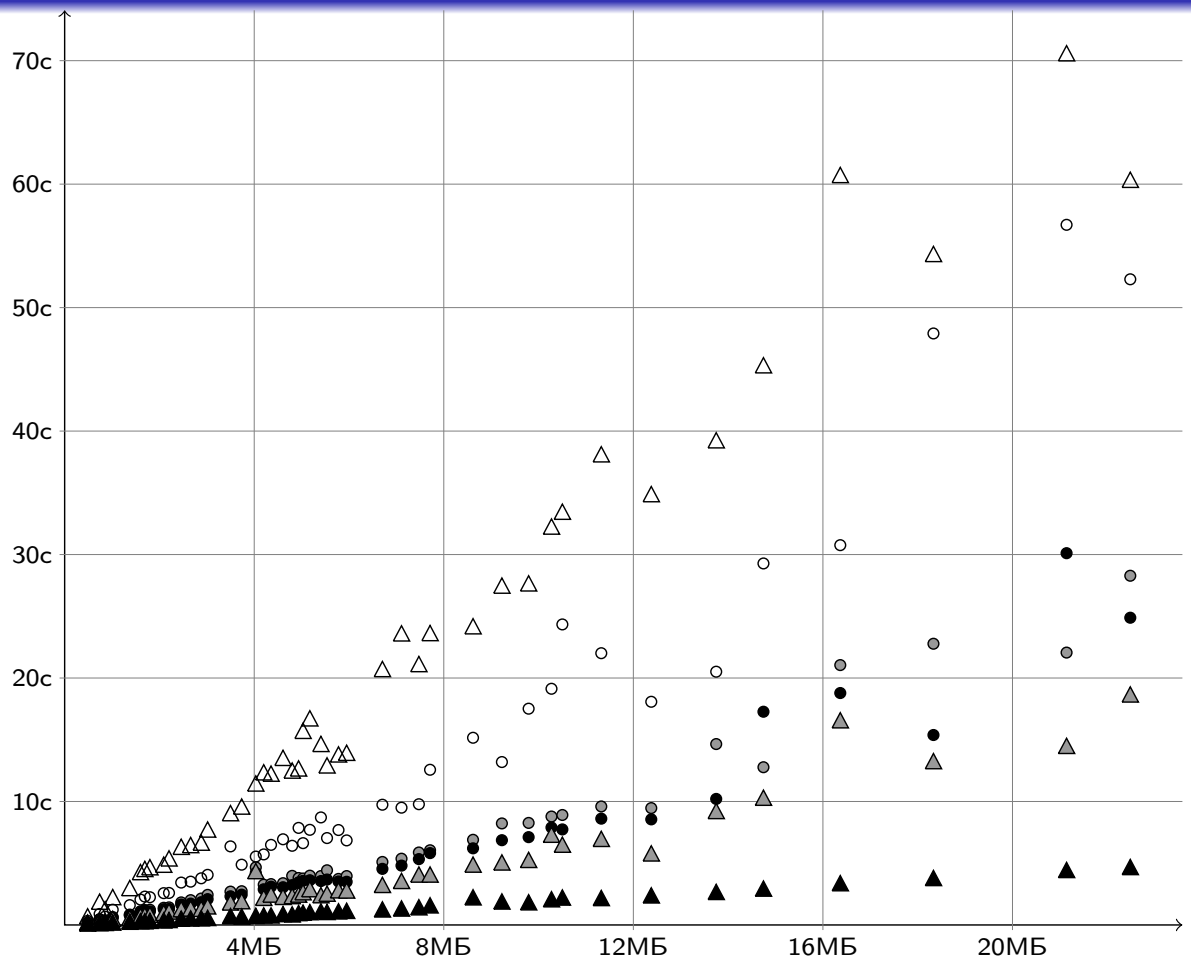
- строит $O(\log^2 n)$-приближение к минимальной;
- требует на вход только сам текст;
- работает online.

Практические результаты

Алгоритмы были протестированы на:

- последовательности строк с большой LZ-факторизацией ($\Omega(\frac{n}{\log n})$);
- случайных строках над четырехбуквенным алфавитом;
- ДНК, взятых с сайта `http://www.ddbj.nig.ac.jp/`.

Скорость работы на строках ДНК



○ – Риттер;

◍ – Риттер с эвристикой;

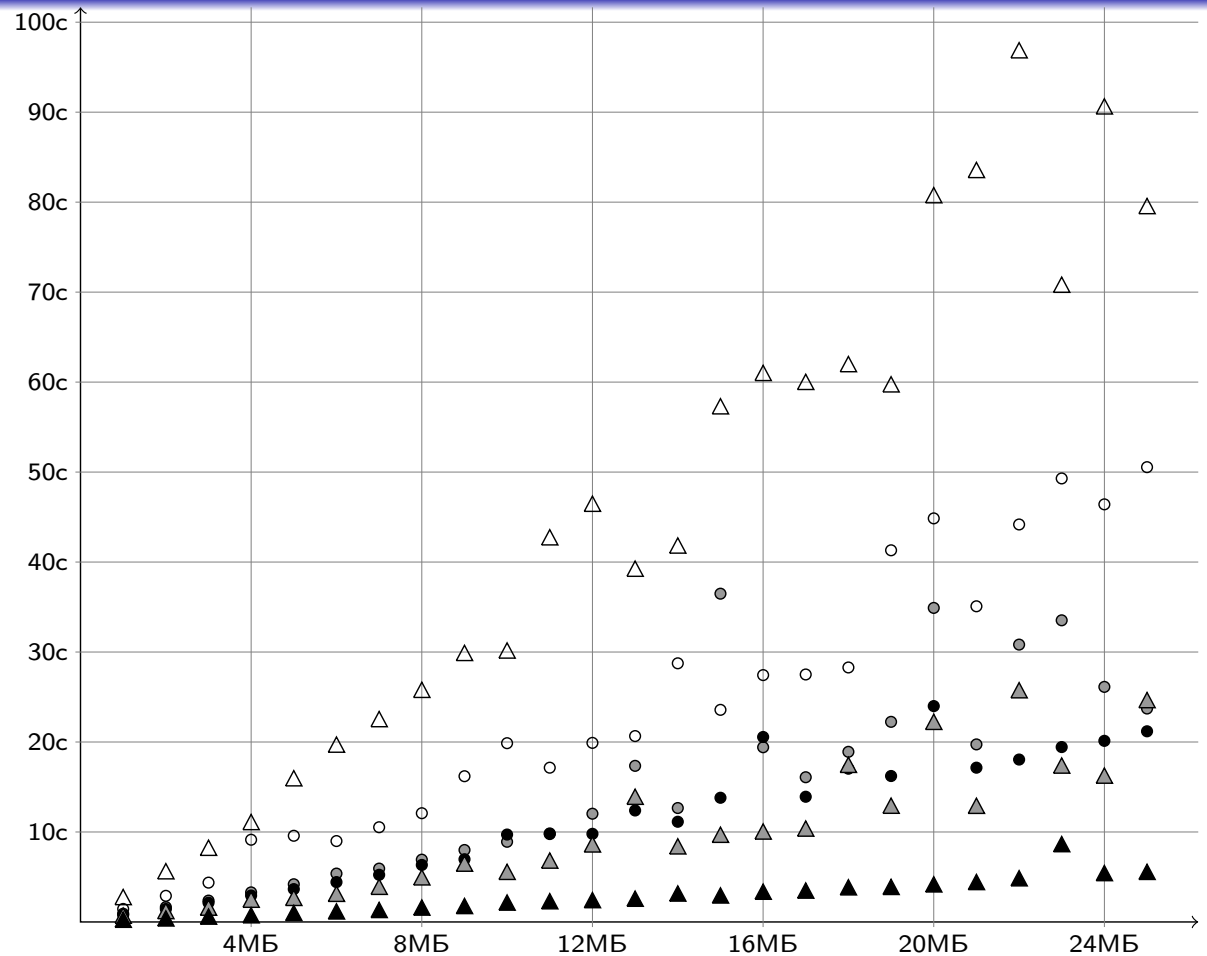
● – Риттер с эвристикой New!;

△ – рандомизированные ПП;

◭ – многопоточный алгоритм New!;

▲ – LCA-online.

Скорость работы на случайных строках



○ – Риттер;

◍ – Риттер с эвристикой;

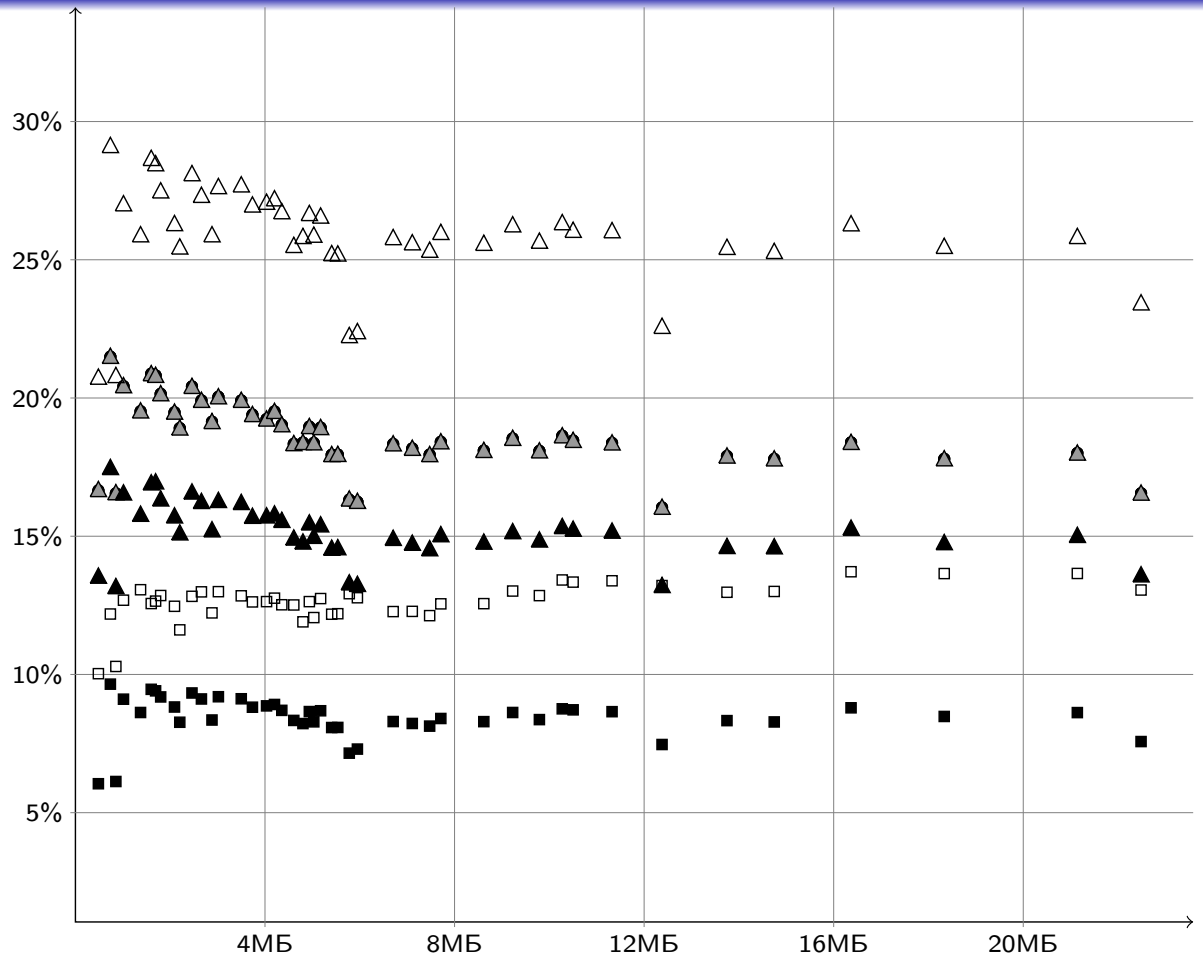
● – Риттер с эвристикой New!;

△ – рандомизированные ПП;

◭ – многопоточный алгоритм New!;

▲ – LCA-online.

Отношение размеров представлений на строках ДНК



□ – LZ с окном сжатия 32КБ;

■ – LZ;

○ – Риттер;

△ – рандомизированные ПП;

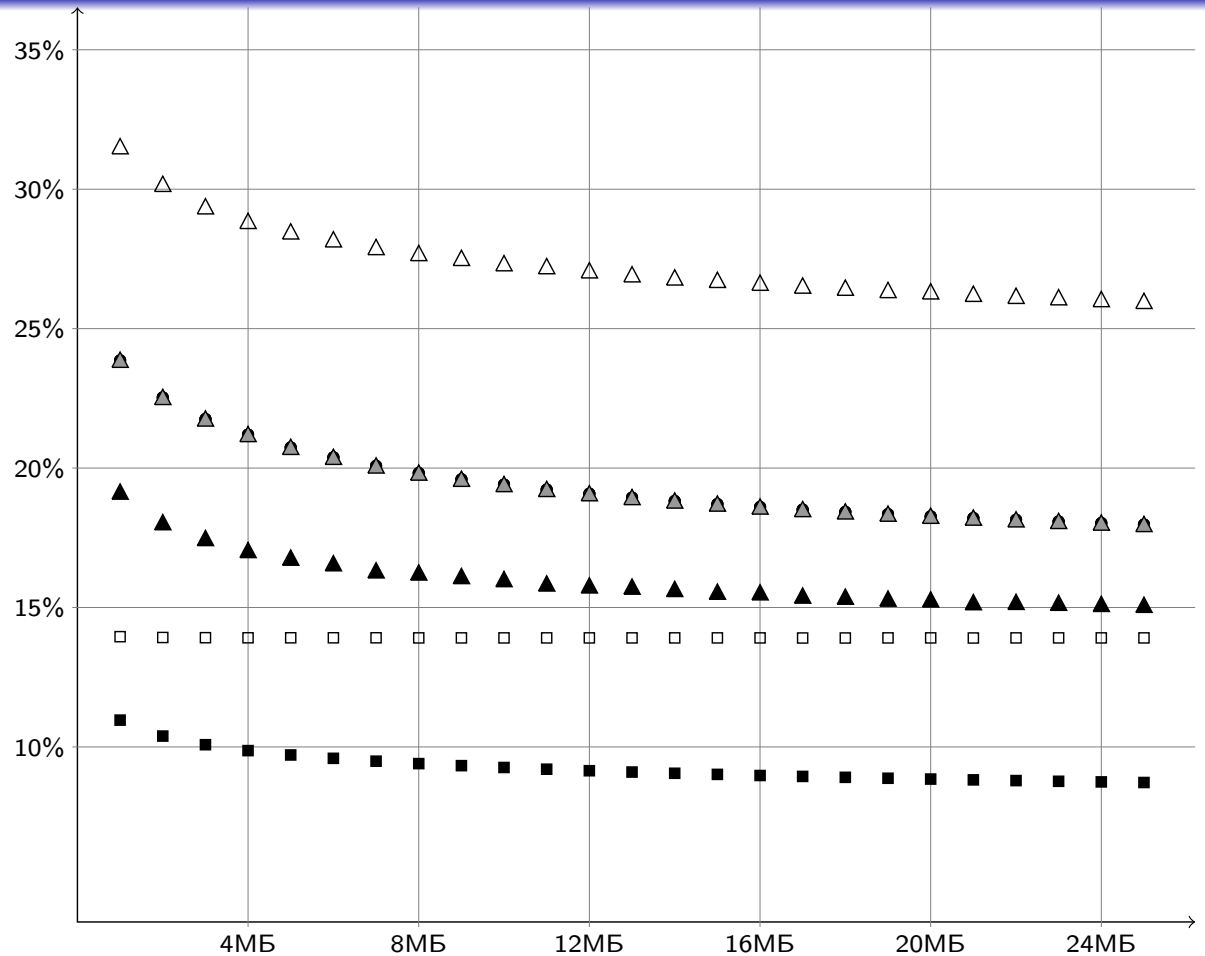
◍ – Риттер с эвристикой;

◭ – многопоточный алгоритм New!;

● – Риттер с эвристикой New!;

▲ – LCA-online.

Отношение размеров представлений на случайных стр.



□ – LZ с окном сжатия 32КБ;

■ – LZ;

○ – Риттер;

△ – рандомизированные ПП;

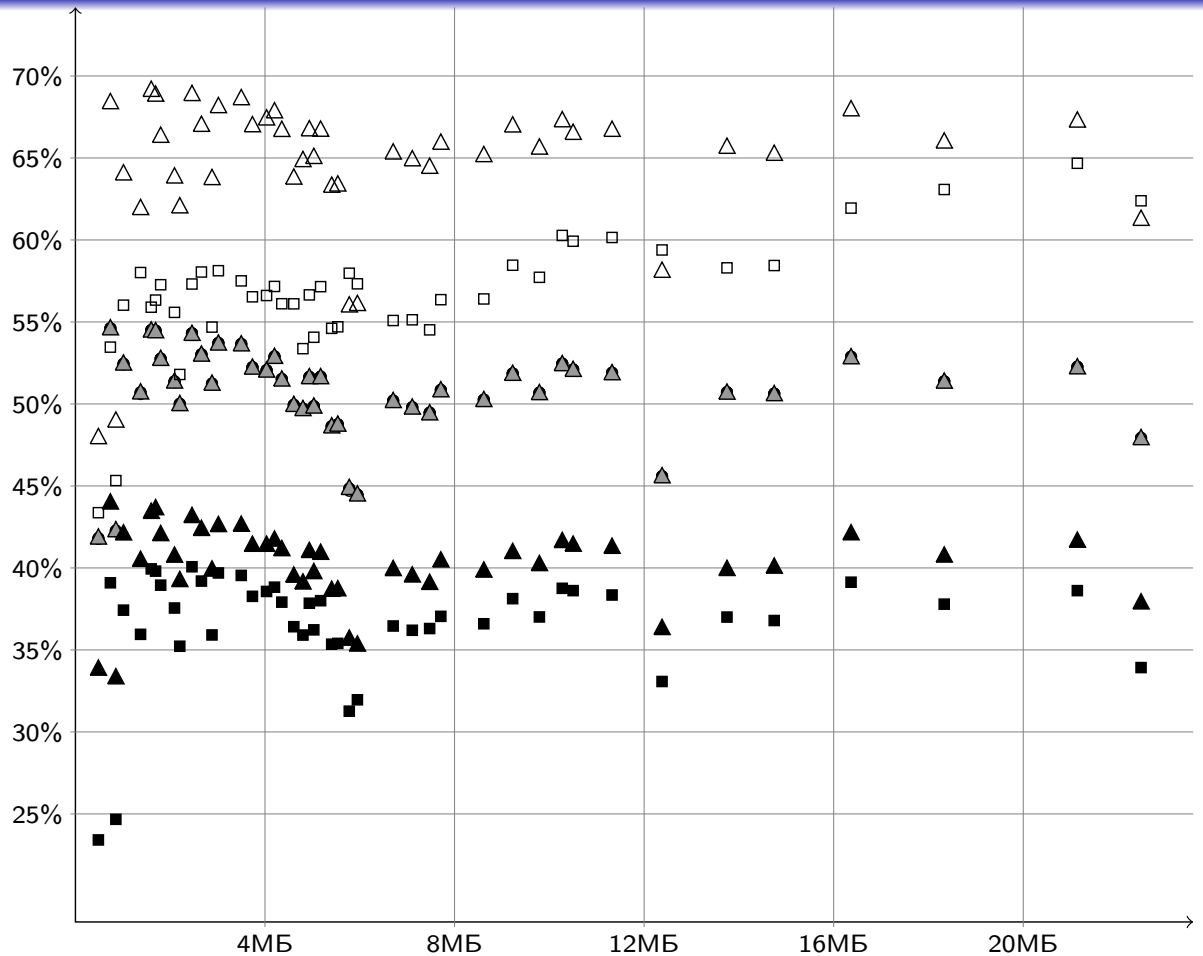
◍ – Риттер с эвристикой;

◭ – многопоточный алгоритм New!;

● – Риттер с эвристикой New!;

▲ – LCA-online.

Коэффициент сжатия на строках ДНК



□ – LZ с окном сжатия 32КБ;

■ – LZ;

○ – Риттер;

△ – рандомизированные ПП;

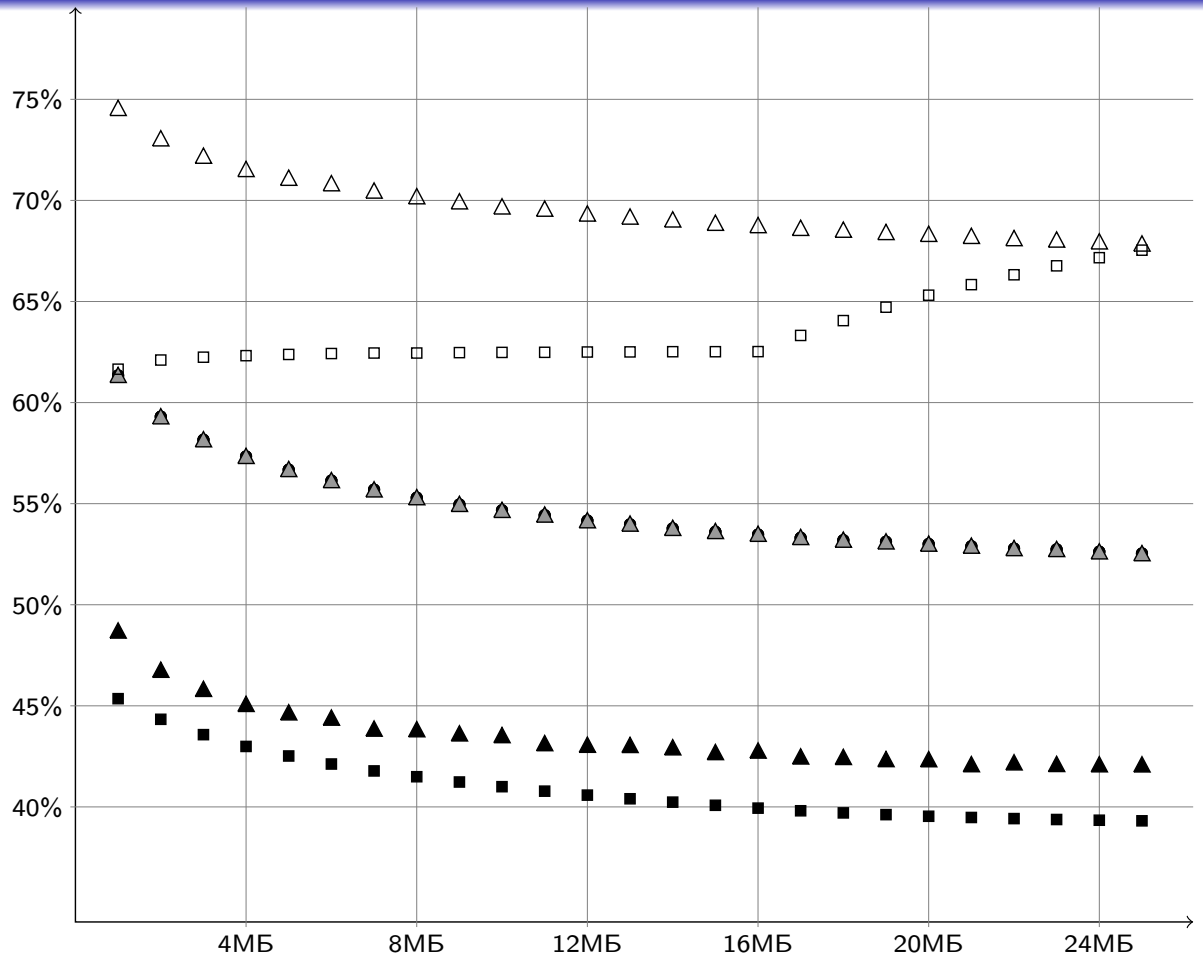
◍ – Риттер с эвристикой;

◭ – многопоточный алгоритм New!;

● – Риттер с эвристикой New!;

▲ – LCA-online.

Коэффициент сжатия на случайных строках



□ – LZ с окном сжатия 32КБ;

■ – LZ;

○ – Риттер;

△ – рандомизированные ПП;

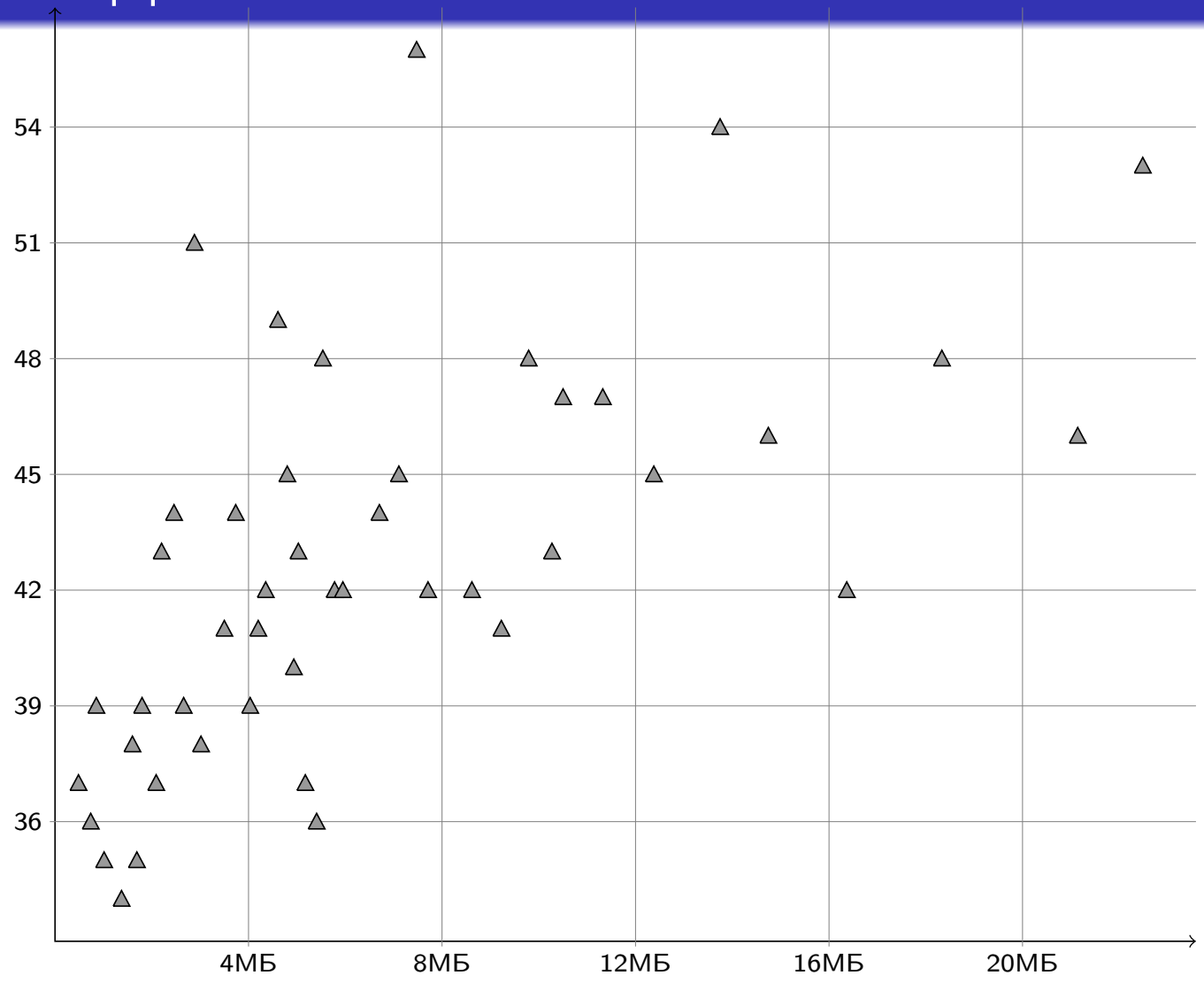
◍ – Риттер с эвристикой;

◭ – многопоточный алгоритм New!;

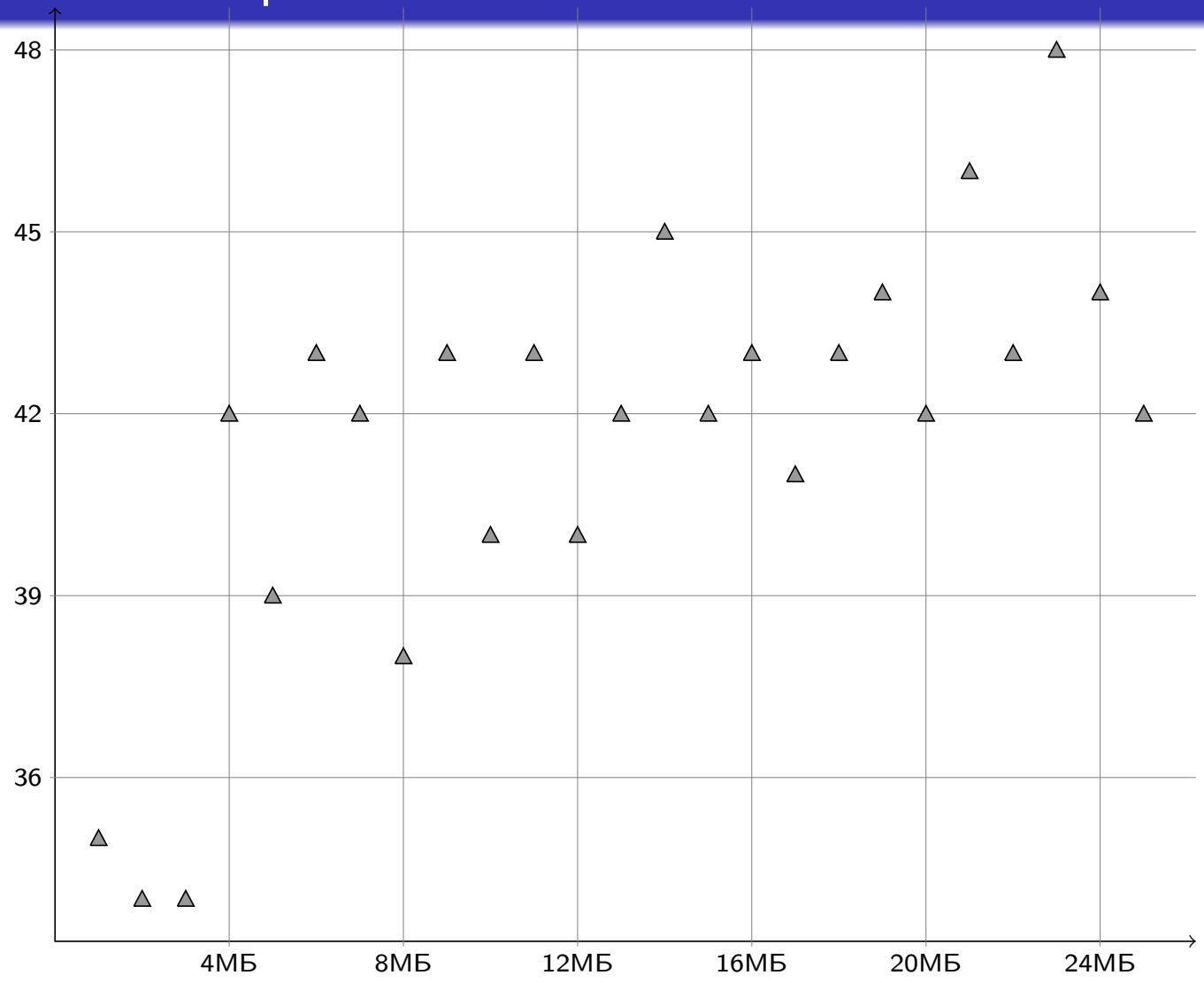
● – Риттер с эвристикой New!;

▲ – LCA-online.

Количеств групп факторов в многопоточном алгоритме на строках ДНК



Количеств групп факторов в многопоточном алгоритме на случайных строках



Результаты и дальнейшие планы

Результаты

- Обобщение эвристики оптимизации порядка конкатенаций.
- Многопоточный алгоритм построения ПП.
- Практические результаты сравнения алгоритмов по двум параметрам: скорость работы и качество сжатия.
- LCA-online лучше всех?

Исходные коды алгоритмов можно посмотреть здесь:
`http://code.google.com/p/overclocking`

Планы

- Исследовать оценку качества сжатия LCA-online алгоритма.
- Доделать реализацию многопоточного алгоритма.

Результаты и дальнейшие планы

Результаты

- Обобщение эвристики оптимизации порядка конкатенаций.
- Многопоточный алгоритм построения ПП.
- Практические результаты сравнения алгоритмов по двум параметрам: скорость работы и качество сжатия.
- LCA-online лучше всех?

Исходные коды алгоритмов можно посмотреть здесь:
`http://code.google.com/p/overclocking`

Планы

- Исследовать оценку качества сжатия LCA-online алгоритма.
- Доделать реализацию многопоточного алгоритма.

Результаты и дальнейшие планы

Результаты

- Обобщение эвристики оптимизации порядка конкатенаций.
- Многопоточный алгоритм построения ПП.
- Практические результаты сравнения алгоритмов по двум параметрам: скорость работы и качество сжатия.
- LCA-online лучше всех?

Исходные коды алгоритмов можно посмотреть здесь:
`http://code.google.com/p/overclocking`

Планы

- Исследовать оценку качества сжатия LCA-online алгоритма.
- Доделать реализацию многопоточного алгоритма.

Результаты и дальнейшие планы

Результаты

- Обобщение эвристики оптимизации порядка конкатенаций.
- Многопоточный алгоритм построения ПП.
- Практические результаты сравнения алгоритмов по двум параметрам: скорость работы и качество сжатия.
- LCA-online лучше всех?

Исходные коды алгоритмов можно посмотреть здесь:
`http://code.google.com/p/overclocking`

Планы

- Исследовать оценку качества сжатия LCA-online алгоритма.
- Доделать реализацию многопоточного алгоритма.

Результаты и дальнейшие планы

Результаты

- Обобщение эвристики оптимизации порядка конкатенаций.
- Многопоточный алгоритм построения ПП.
- Практические результаты сравнения алгоритмов по двум параметрам: скорость работы и качество сжатия.
- LCA-online лучше всех?

Исходные коды алгоритмов можно посмотреть здесь:
`http://code.google.com/p/overclocking`

Планы

- Исследовать оценку качества сжатия LCA-online алгоритма.
- Доделать реализацию многопоточного алгоритма.

Результаты и дальнейшие планы

Результаты

- Обобщение эвристики оптимизации порядка конкатенаций.
- Многопоточный алгоритм построения ПП.
- Практические результаты сравнения алгоритмов по двум параметрам: скорость работы и качество сжатия.
- LCA-online лучше всех?

Исходные коды алгоритмов можно посмотреть здесь:
`http://code.google.com/p/overclocking`

Планы

- Исследовать оценку качества сжатия LCA-online алгоритма.
- Доделать реализацию многопоточного алгоритма.