Декодирование **полярных** кодов: сравнение списочного и стекового алгоритма декодирования.

Анастасия Смешко, Анвар Курмуков

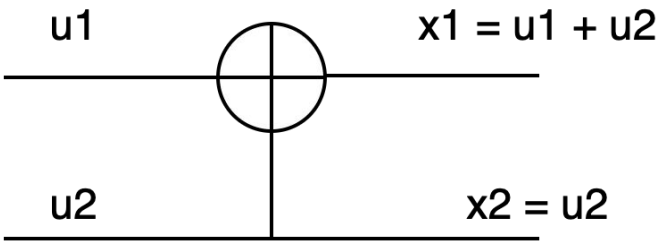
Август, 2020

План презентации

1. Полярные коды
2. Алгоритмы декодирования: SC, SCL, SCS
3. Сравнение алгоритмов: скорость, память.
4. Результаты экспериментов
5. Выводы

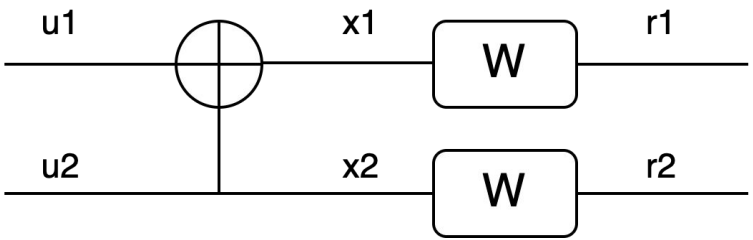
Полярные коды

- Полярное преобразование
- Длина кода 2^n
- Поляризация канала



- При декодировании:
  U1 → R1, R2
  U2 → R1, R2, U1

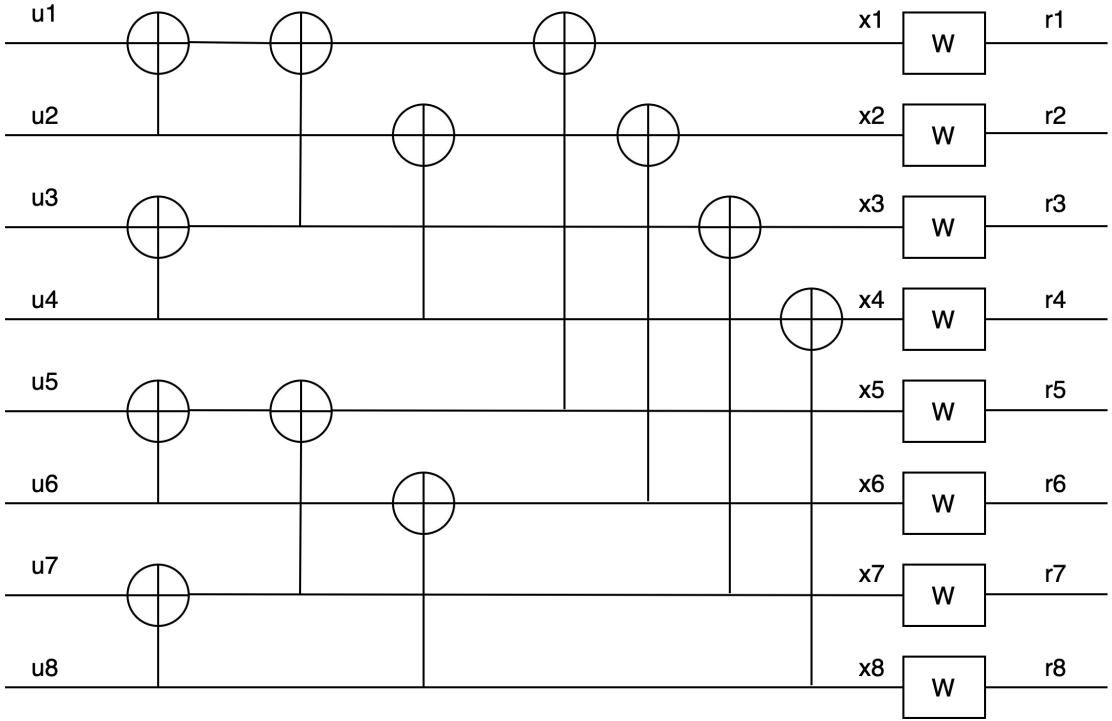
$$G_2 = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}$$



$$G_2^{\otimes 2} = \begin{bmatrix} G_2 & 0 \\ G_2 & G_2 \end{bmatrix}$$

Полярные коды

- Полярный кодер для длины $N = 2^3 = 8$
- Заморозка битов
- После кодирования передаем по каналу
- По полученным из канала значениям оцениваем u
- Сложность **кодирования**:
  - по времени O(N log(N))
  - по памяти O(N)



Алгоритмы декодирования

Последовательные алгоритмы декодирования:

a. Successive cancellation (**SC**) - базовый алгоритм.
b. SC List (**SCL**) - множественное принятие решения, выбор из набора вариантов.
c. SC Stack (**SCS**) - тоже что и SCL только по-другому :)
d. Модификации SC, SCL, SCS.

Successive Cancellation (SC)

Левый потомок:

f(a, b) = sign(a) · sign(b) · min(|a|, |b|)

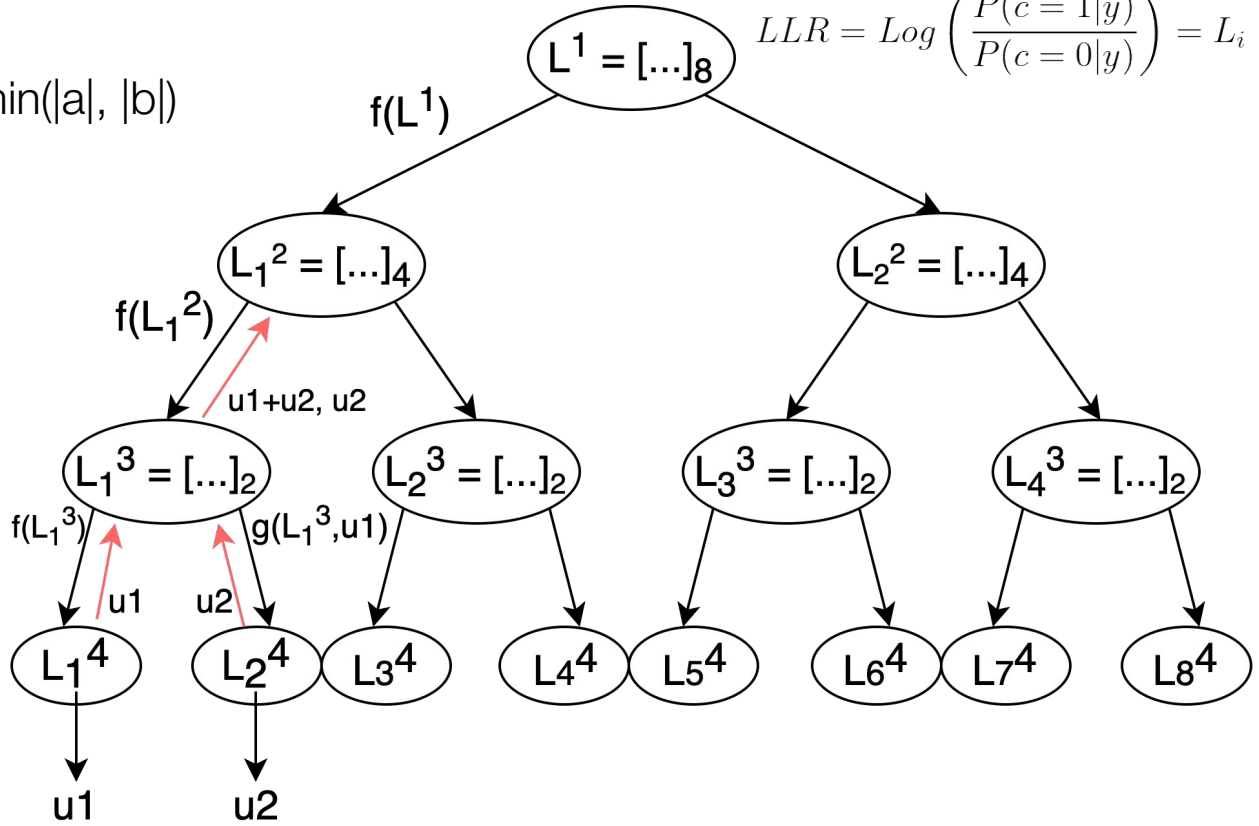
Правый потомок:

g(a, b, c) = b + (1 - 2c)a

Принятие решения ui по Li осуществляется по следующим правилам:

- если ui заморожен → ui = 0
- если ui не заморожен и Li >= 0 → ui = 0
- если ui не заморожен и Li < 0 → ui = 1

$$LLR = Log\left(\frac{P(c=1|y)}{P(c=0|y)}\right) = L_i$$



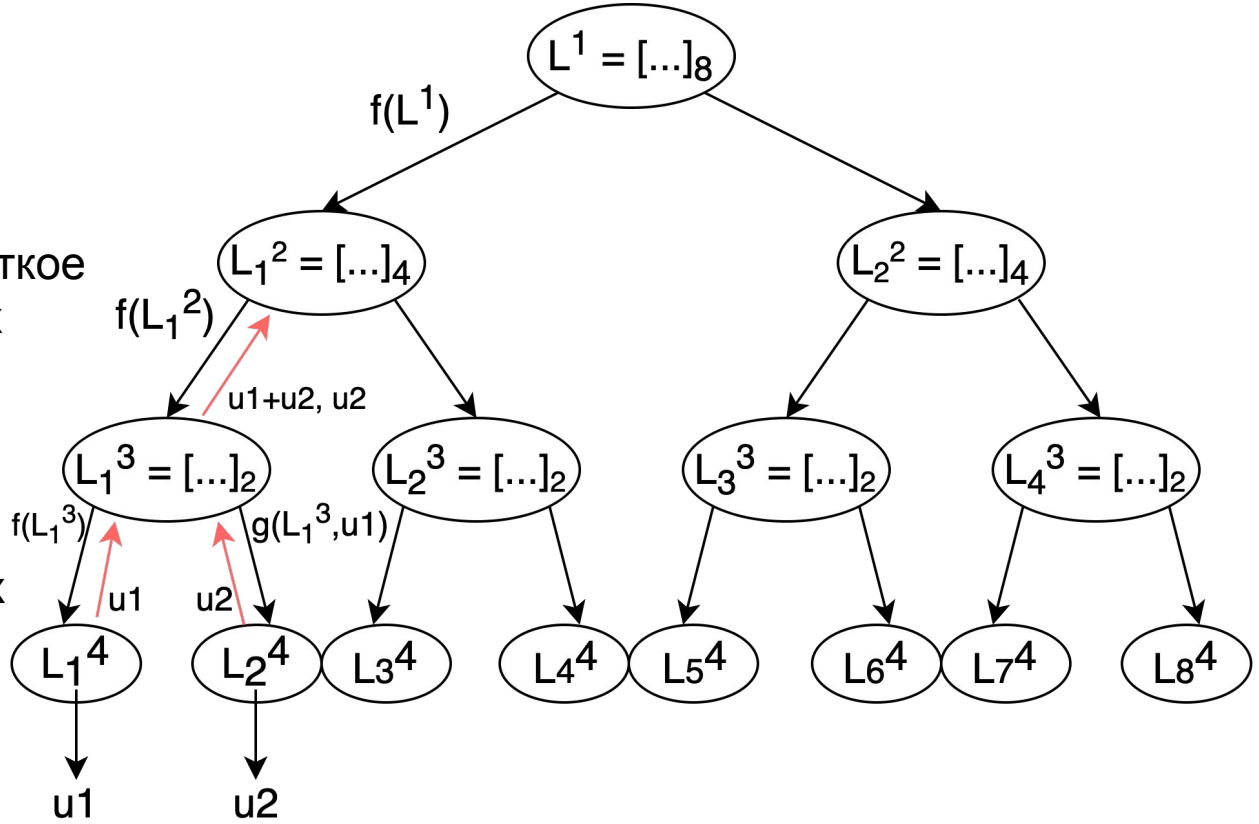
Successive Cancellation (**SC**)

- Сложность алгоритма **декодирования** O(N log(N)).
- Быстро работает, прост в реализации.
- Если ошибаемся при принятии решения в одном бите, дальше все слово декодировано неверно.
- Часто ошибаемся.

Successive Cancellation List (SCL)

Идея:

- Принимаем не одно жесткое решение, а два для всех путей
- Считаем их метрики
- Храним только **L** лучших



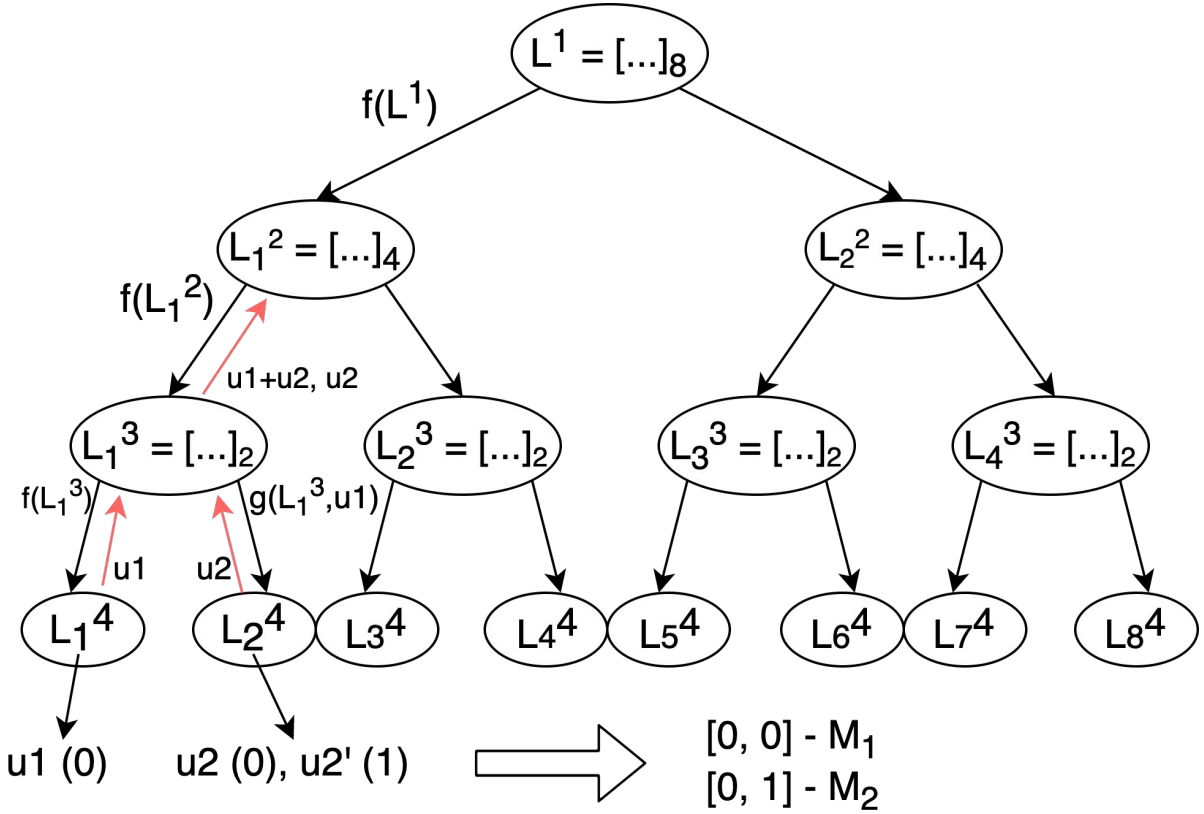
Successive Cancellation List (SCL)

Принимаем решение и считаем метрику:

Если ui заморожен: ui = 0,
- ΔM = 0, если Li >= 0
- ΔM = |Li|, если Li < 0

Если ui не заморожен и Li >= 0:
- ui = 0, ΔM = 0
- ui = 1, ΔM = |Li|

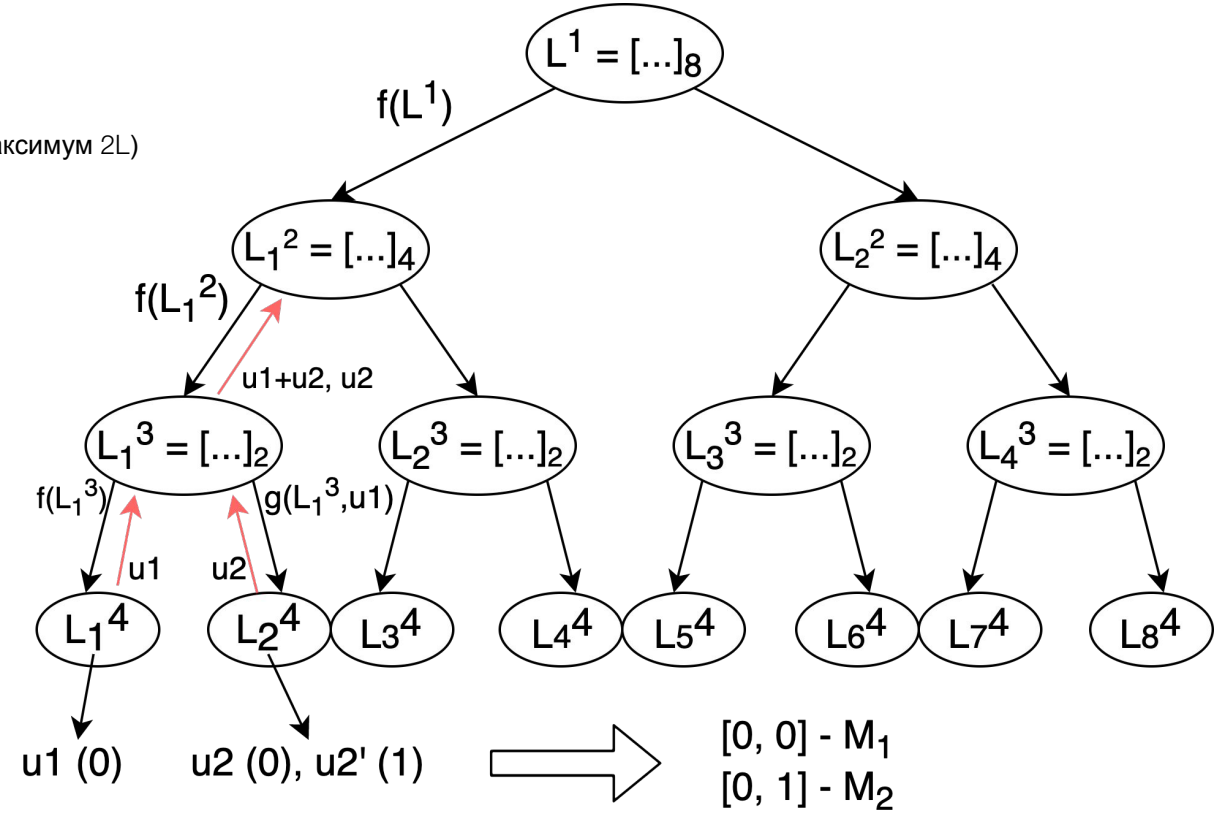
Если ui не заморожен и Li < 0:
- ui = 0, ΔM = |Li|
- ui = 1, ΔM = 0



Successive Cancellation List (SCL)

- Храним максимум L путей
- Каждый продолжаем (получаем максимум 2L)
- Оставляем только L лучших в соответствии с метрикой
- Сложность:
  - по времени O(LN log(N))
  - по памяти O(LN)

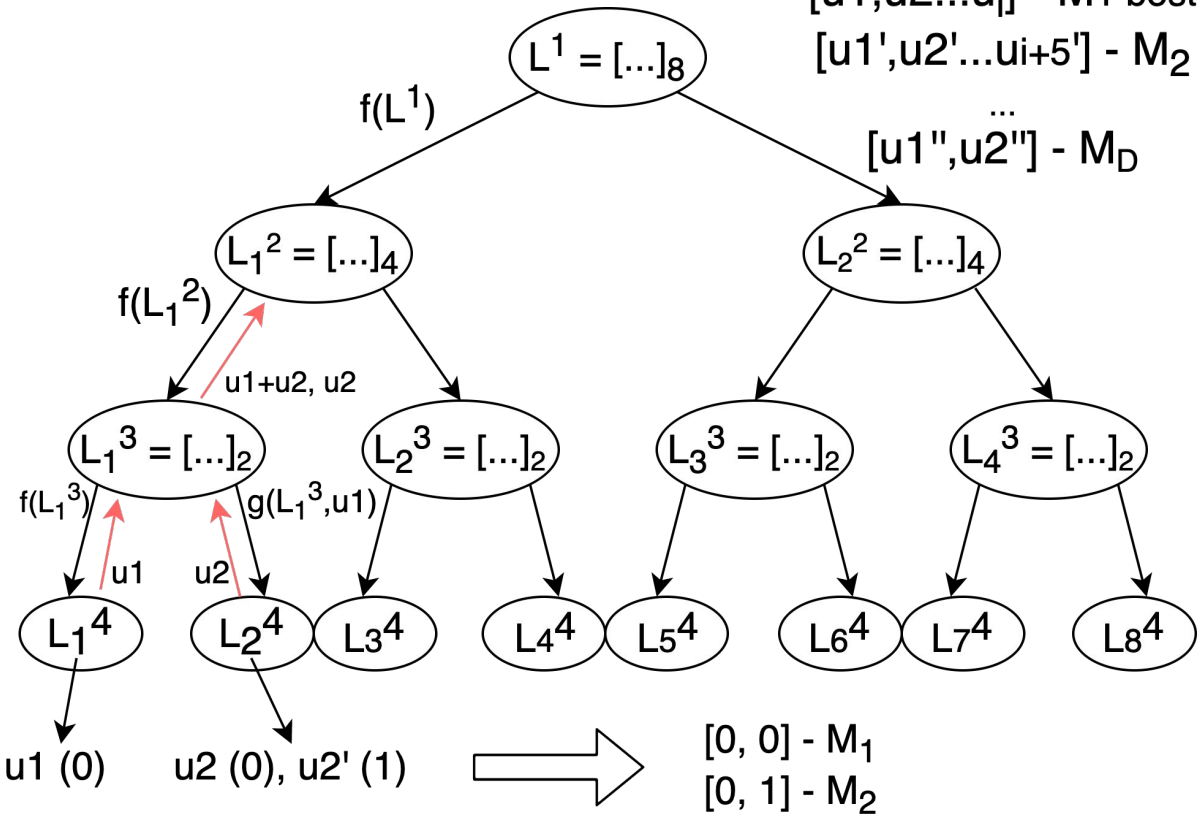
List:
$[u1,u2...u_i]$ - M1
$[u1',u2'...u_i']$ - M2
...
$[u1'',u2''...u_i'']$ - ML



Successive Cancellation Stack (**SCS**)

Идея:

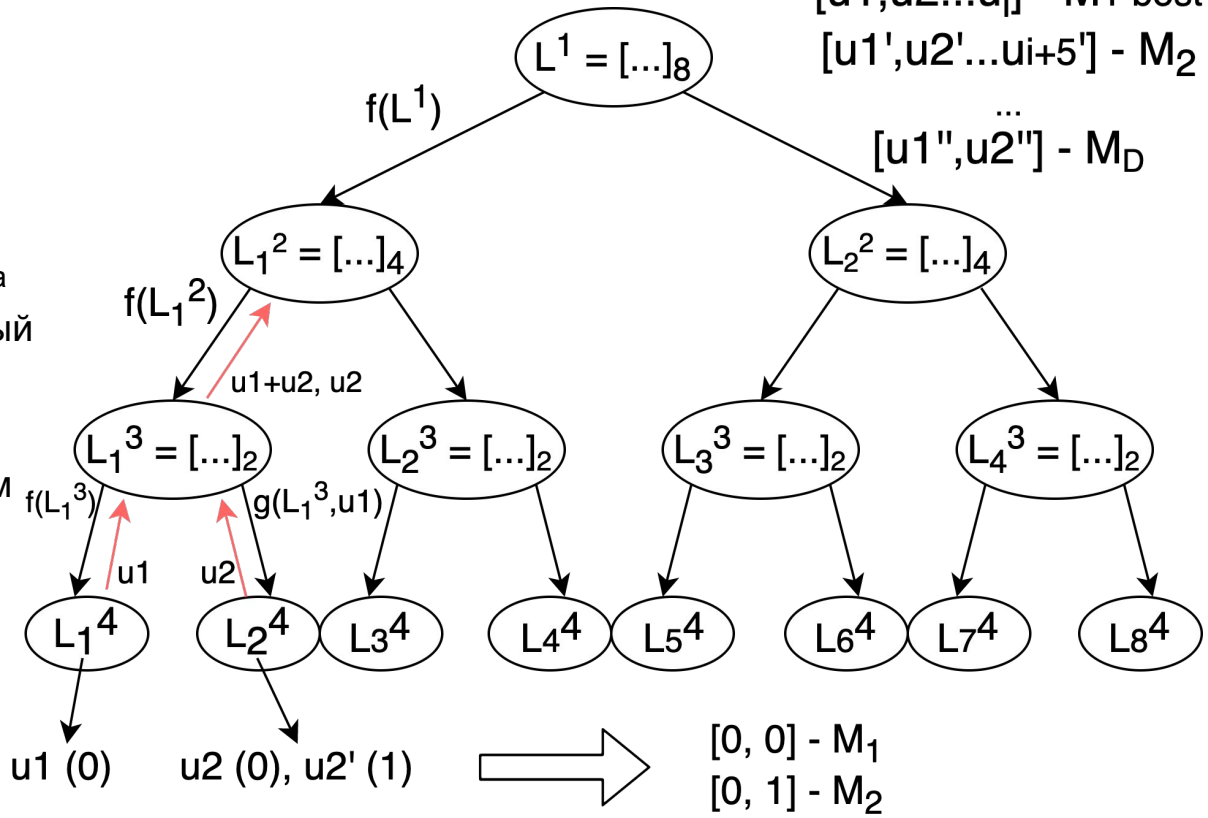
- Каждый раз продолжаем только один лучший путь
- После продолжения пути получаем 1 или 2 новых путей
- Сортируем стек, снова берем лучший путь
- Храним максимум D путей
- Метрики путей считаем так же, как и для SCL



Successive Cancellation Stack (**SCS**)

Добавляем ограничение на продление путей параметром L:

1. Храним массив PL (длины N = длина кодового слова), инициализированный нулями.
2. Каждый раз, когда из стека берем лучший путь длины a:
   PL[a] = PL[a] + 1
3. Если PL[a] >= L, удаляем из стека все пути длины <= a.



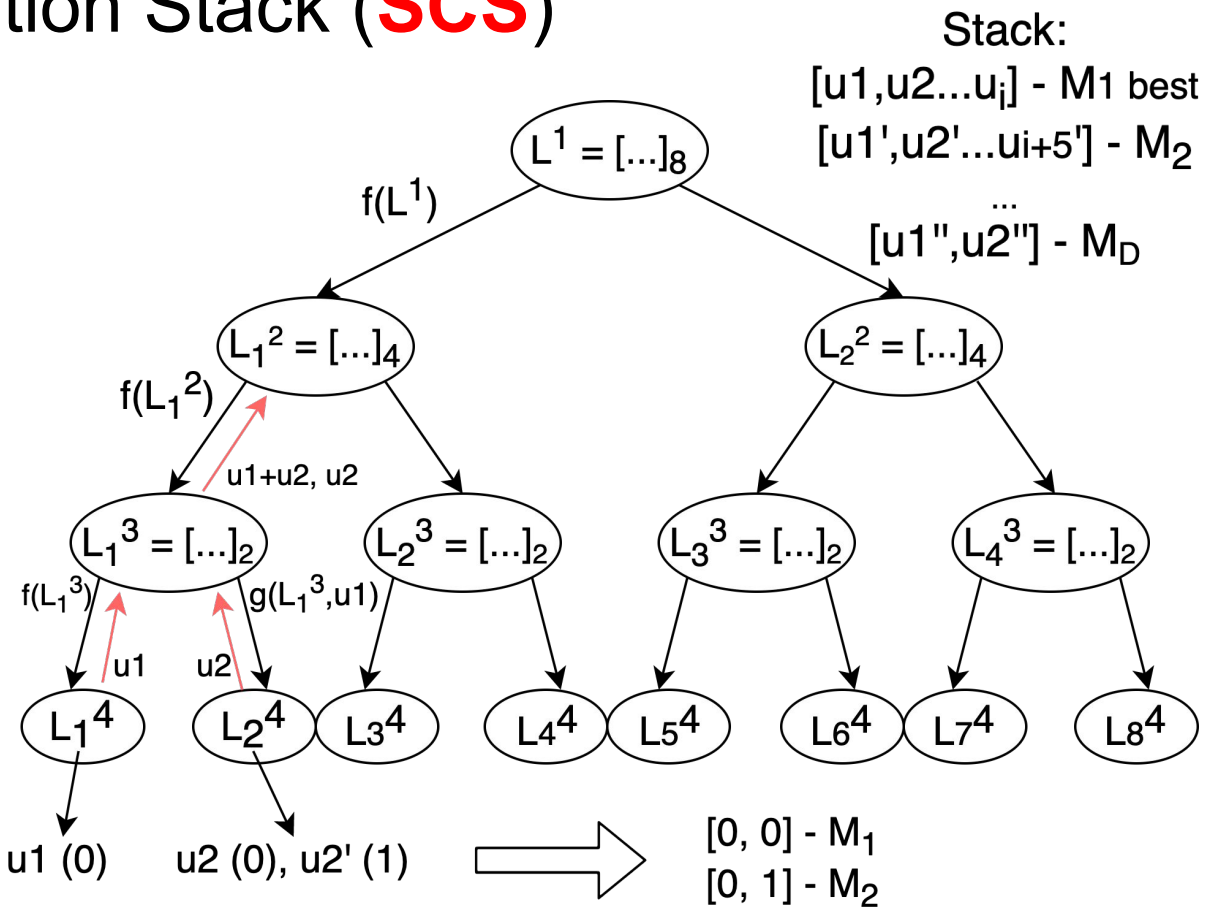
Successive Cancellation Stack (**SCS**)

У алгоритма два параметра:

- Размер стека, D
- Ограничение на продление путей, L

Сложность:

- по времени O(LN log(N))
- по памяти O(DN)



Алгоритмы декодирования: SC, SCL, SCS и их модификации

Алгоритм декодирования	Сложность по времени	Сложность по памяти
SC	O(N log(N))	O(N)
SCL(L)*	O(LN log(N))	O(LN)
SCS(L, D)**	O(LN log(N))	O(DN)

*параметр L берут небольшим (обычно 8)
**параметр D должен быть порядка длины кода (N)

- Улучшение по времени (Fast algorithms)
- Улучшение по памяти

Описание канала

Мы приводим результаты для канала BPSK + AWGN:

$$0 \rightarrow -1$$
$$1 \rightarrow 1$$

С добавлением нормально распределенного шума: $\varepsilon \sim N(0, \boldsymbol{\sigma}^2)$

Cyclic redundancy code (CRC)

1. Добавляем проверку корректности кодового слова, рассчитывая остаток от деления на многочлен
2. Порождающий многочлен подбирается на теории кодирования и исследований
3. Простая реализация, высокая надежность, возможность выбора из множества полученных кодовых слов после декодирования (SCL, SCS)

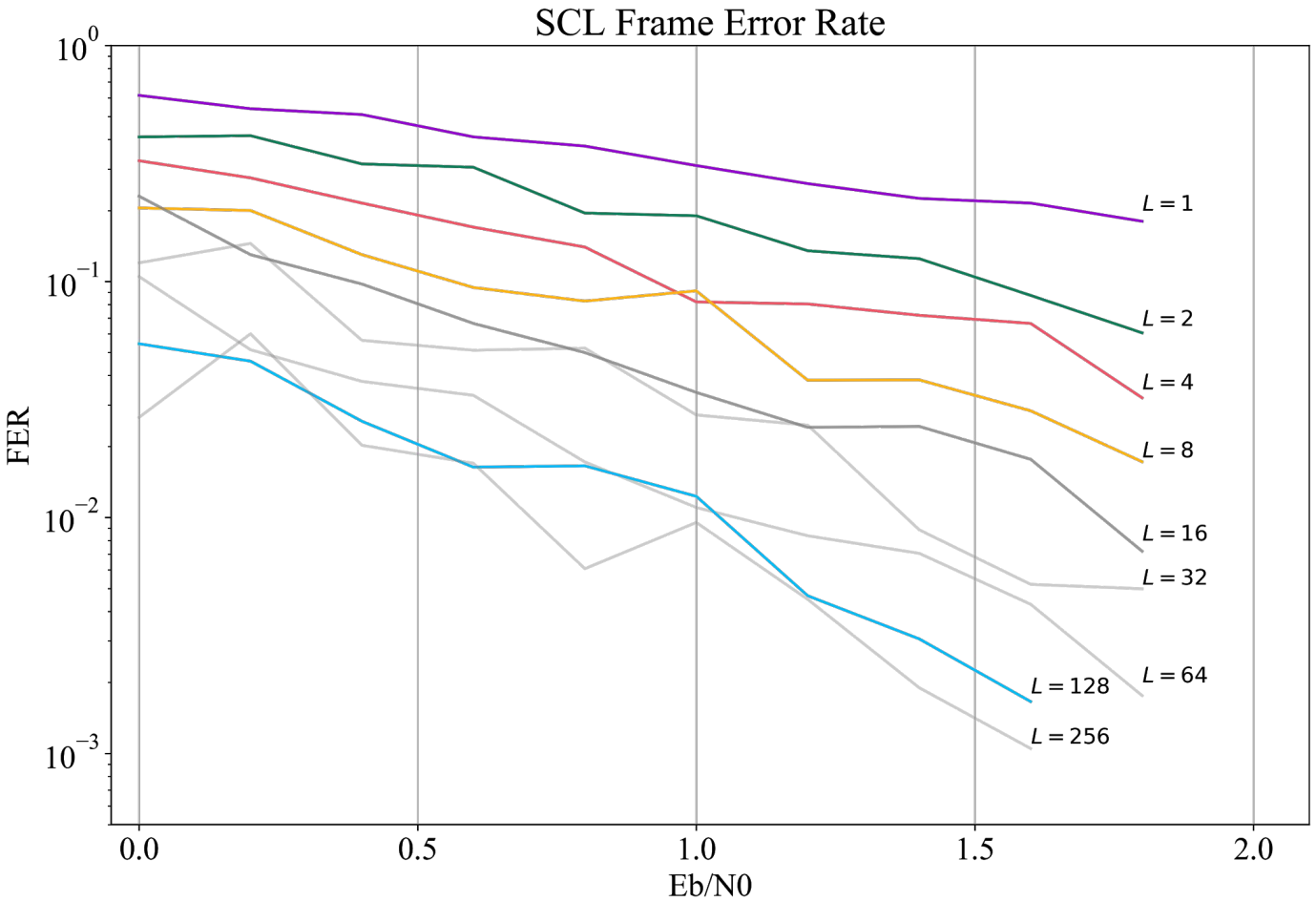
Результаты экспериментов

1. Полярный код K = 21, N = 64, CRC

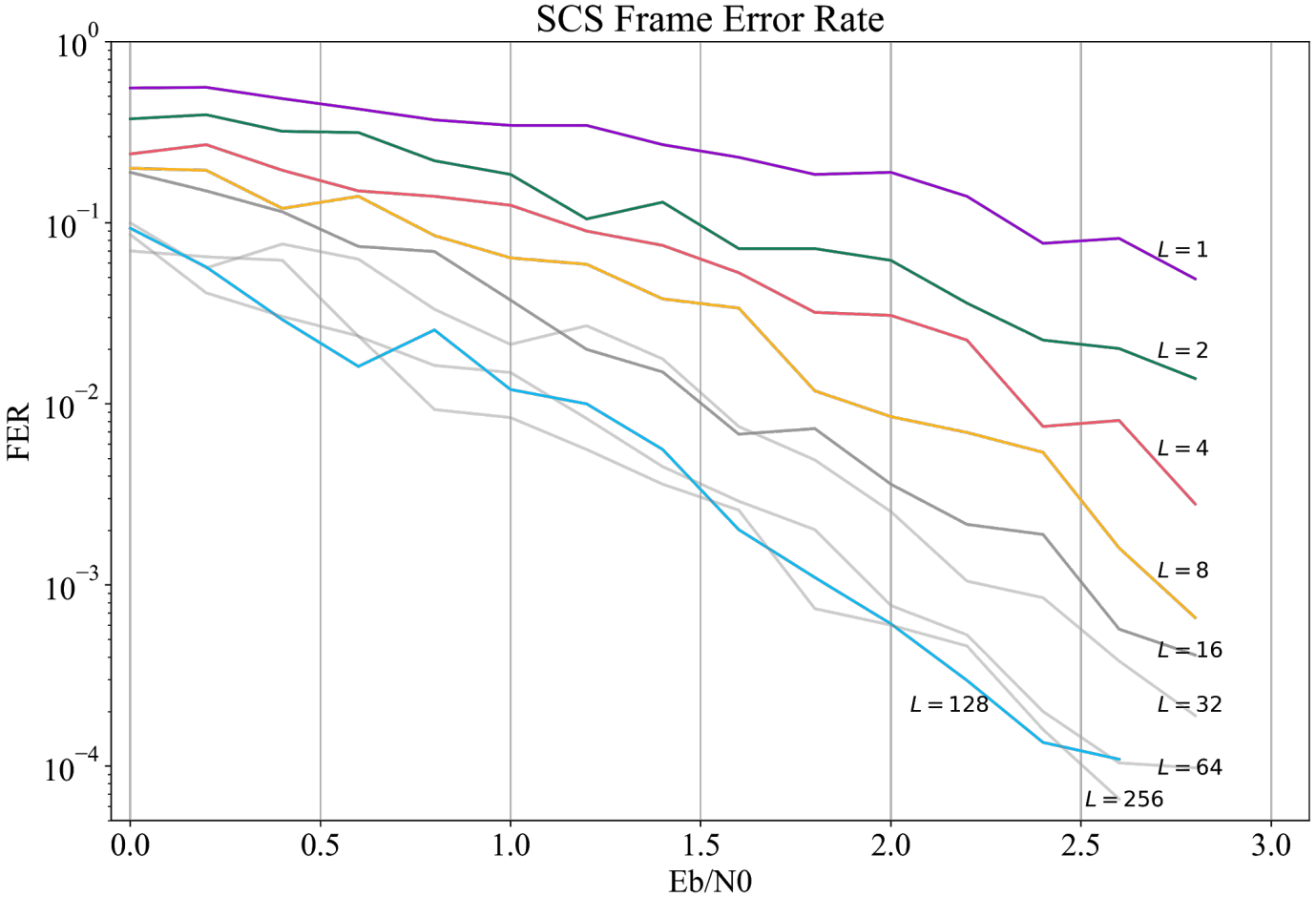
   $z^{11} + z^{10} + z^9 + z^5 + 1$

2. Приведены результаты для SC, SCL, SCS с D = 64
3. Варьировалось значение Eb/N0, которым определяется шум в канале (BPSK)
4. Значение параметра L для SCL и SCS варьировалось от 2^0 до 2^{10}

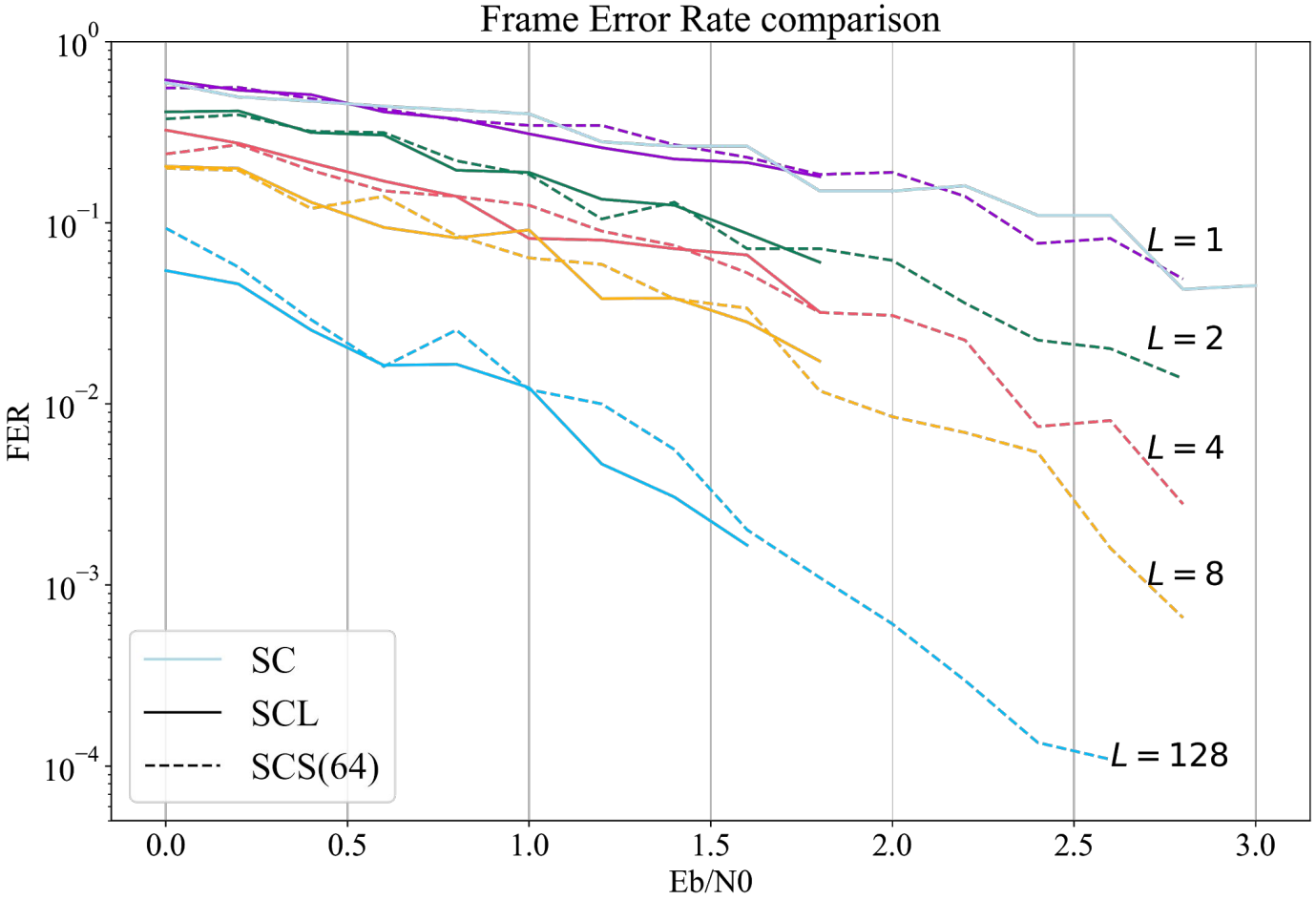
Результаты SCL FER



Результаты SCS FER



Сравнение SC, SCL, SCS FER



Сравнение среднего времени работы

```
N = 128
SC                    | 2.22 ms ± 85.4 µs per loop
SCS L = 8, D = 512    | 5.22 ms ± 176 µs per loop
SCL L = 8             | 63.2 ms ± 14.2 ms per loop

N = 512
SC                    | 8.67 ms ± 312 µs per loop
SCS L = 8, D = 512    | 25.7 ms ± 1.76 ms per loop
SCL L = 8             | 646 ms ± 231 ms per loop
```

Результаты экспериментов

- Результаты SCL лучше, чем у SC (при больших L)
- Для одинаковых L результаты для FER у SCS и SCL совпадают
- SCL имеет значительно меньшую сложность по памяти
- SCS считается значительно быстрее по времени (в среднем), при одинаковой максимальной сложности
- Необходима большая статистика

Результаты проекта

- Реализовано построение полярного кода без/с CRC на Python.
- Реализованы алгоритмы SC, SCL, SCS на Python.
- Собрана статистика для различных значений шума в канале и различных параметров алгоритмов.
- Проанализированы результаты.

Вклад участников:
Смешко Анастасия - поиск статей, написание кода (SCL, SCS), подготовка слайдов
Курмуков Анвар - написание кода (SC), обработка результатов, подготовка слайдов

Список литературы

1. Arikan, Erdal. "Channel polarization: A method for constructing capacity-achieving codes for symmetric binary-input memoryless channels." *IEEE Transactions on information Theory* 55, no. 7 (2009): 3051-3073. Polar codes proposed
2. Tal, Ido, and Alexander Vardy. "List decoding of polar codes." *IEEE Transactions on Information Theory* 61, no. 5 (2015): 2213-2226. SCL proposed
3. Niu, K. and Chen, K., 2012. Stack decoding of polar codes. *Electronics letters*, *48*(12), pp.695-697.
4. Aurora, Harsh, and Warren J. Gross. "Towards Practical Software Stack Decoding of Polar Codes." *arXiv preprint arXiv:1809.03606* (2018). SCS proposed
5. Aurora, Harsh, Carlo Condo, and Warren J. Gross. "Low-complexity software stack decoding of polar codes." In *2018 IEEE International Symposium on Circuits and Systems (ISCAS)*, pp. 1-5. IEEE, 2018.
6. Chen, Kai, Kai Niu, and Jiaru Lin. "Improved successive cancellation decoding of polar codes." *IEEE Transactions on Communications* 61, no. 8 (2013): 3100-3107.

Спасибо за внимание!