Оценки на сложность вычисления некоторых булевых функций и формул де Моргана

Арсентьев Арсений

Март 2022

Содержание

1 Введение

В данной работе представлены сильные оценки на сложность вычисления и коммуникационную сложность игры для некоторых булевых функций. Все блоки, необходимые для создания итоговых схем, найдены SAT-солвером и протестированы с помощью программы на языке Python на всех возможных входных комбинациях. В последнем разделе приведены некоторые из моих идей по улучшению методов поиска схем SAT-солверами.

Определение 1. *Совершенная дизъюнктивная нормальная форма (СДНФ) - это представление булевой функции, в котором:*

- *Нет одинаковых клозов.*
- *В каждом клозе нет повторяющихся переменных.*
- *Каждый клоз содержит все переменные, от которых зависит булева функция (каждая переменная может входить в клоз либо в прямой, либо в инверсной форме).*
- *Переменные соединены конъюнкцией, клозы - дизъюнкцией.*

Определение 2. $size_{B_2}(f)$ *- это размер (сложность вычисления) схемы булевой функции* f *в полном бинарном базисе.*

Определение 3. $Fsize(f)$ *- это размер (сложность вычисления) формулы де Моргана для булевой функции* f*.*

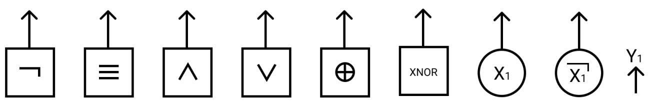


Рис. 1: Обозначения гейтов, входов и выходов.

2 Нижняя оценка для симметричных функций

Теорема 1. *Для любой симметричной булевой функции f справедлива оценка:*

$$size(f) \geq n - 1$$

Доказательство. В симметричных булевых функциях нет фиктивных параметров, следовательно при представлении в виде схемы, получается связный граф (так как каждый вход соединяется хотя бы с 1 гейтом). Связный подграф, подвешенный за одну из вершин (в данном случае за выход схемы), и имеющий минимально возможное количество рёбер, является деревом. Так как входная степень каждого гейта равна 2, а исходящая равна 1, то дерево является бинарным и полным. В полном бинарном дереве количество внутренних вершин на единицу меньше количества листьев, следовательно для реализации любой симметричной булевой функции понадобится минимум $n - 1$ гейт.

Примечание. Данная оценка справедлива и для формул де Моргана, так как формула де Моргана является бинарным деревом.

3 Оценки для схем в полном бинарном базисе

3.1 Точная оценка для MOD_2^n

Теорема 2. $size_{B_2}(MOD_2^n) = n - 1$

Доказательство. Заметим, что бинарная операция $\oplus$ является функцией $\neg(MOD_2^2)$. Из ассоциативности $\oplus$ следует, что функция $\neg(MOD_2^n)$ может быть реализована последовательным использованием оператора $\oplus$ над входами схемы, тогда:

$$MOD_2^n = \neg\left(\bigoplus_{i=1}^{n} x_i\right)$$

Заменим один из операторов XOR на $XNOR$, для осуществления отрицания, тогда сложность вычисления MOD_2^n будет меньше либо равна $n - 1$.
$size_{B_2}(MOD_m^n) \geq n - 1$ (см. раздел 2), следовательно $(n - 1)$ - точная оценка сложности вычисления функции MOD_2^n.

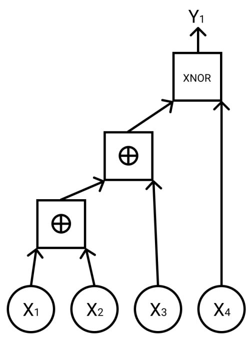
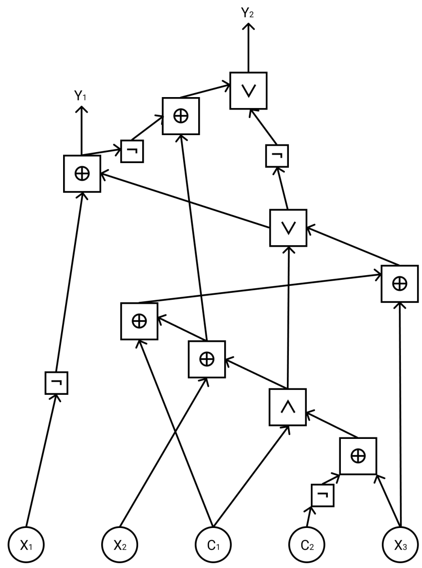


Рис. 2: Схема для MOD_2^4.

3.2 Верхняя оценка для MOD_3^n

Теорема 3. $size_{B_2}(MOD_3^n) \leq 3n + \Theta(1)$

Доказательство. Рассмотрим следующую схему, вычисляющую MOD_3^3:



В ней:

- x_1, x_2, x_3 - входы.
- c_1, c_2 - входы, представляющие собой $(\sum_{i=1}^{n} x_i)\ mod\, 3$.
- y_1, y_2 - выходы, кодирующие $\left(\sum_{i=1}^{n+3} x_i\right)\ mod\, 3$.

Соединением схем MOD_3^3 можно получить схему, состоящую из $n/3$ блоков размера 9, вычисляющую MOD_3^n, следовательно размер схемы MOD_3^n меньше либо равен $3n + \Theta(1)$.

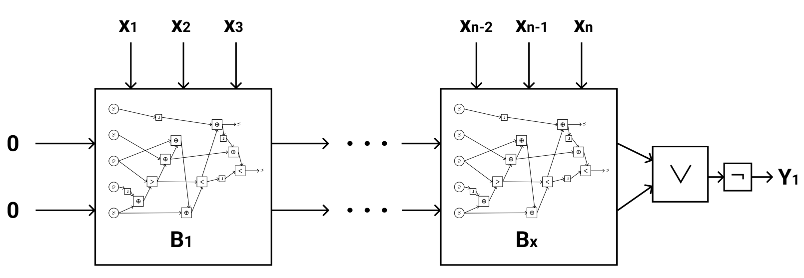
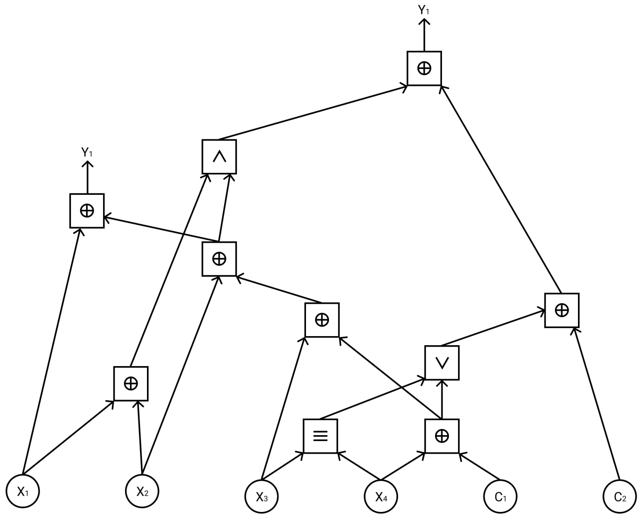


Рис. 3: Пример схемы для MOD_3^n

3.3 Верхняя оценка для MOD_4^n

Теорема 4. $size_{B_2}(MOD_4^n) \leq 2.5n + \Theta(1)$

Доказательство. Рассмотрим следующую схему, вычисляющую MOD_4^4:



В ней:

- x_1, x_2, x_3, x_4 - входы.
- c_1, c_2 - входы, представляющие собой $(\sum_{i=1}^{n} x_i)\ mod\, 4$.
- y_1, y_2 - выходы, кодирующие $\left(\sum_{i=1}^{n+4} x_i\right)\ mod\, 4$.

Соединением схем MOD_4^4 можно получить схему, состоящую из $n/4$ блоков размера 10, вычисляющую MOD_4^n, следовательно размер схемы MOD_4^n меньше либо равен $2.5n + \Theta(1)$.

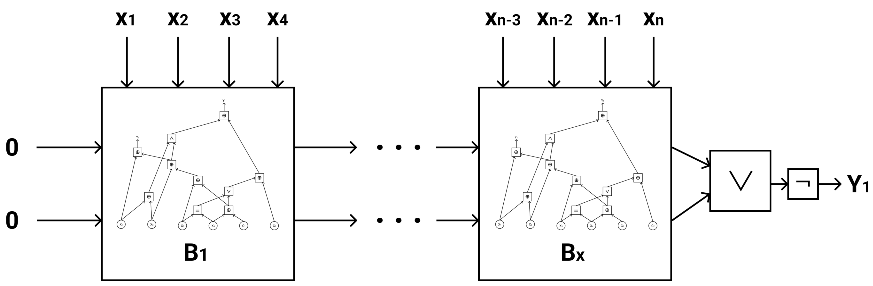
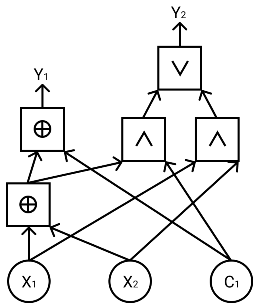


Рис. 4: Пример схемы для MOD_4^n

3.4 Верхняя оценка для MAJ^n

Теорема 5. $size_{B_2}(MAJ^n) \le O(n)$

Доказательство. Функцию MAJ^n можно вычислить посчитав и сравнив количество единичных и нулевых битов. Пусть X - входной набор, $x_1, x_2, \dots, x_n$ - биты X, X' - копия X с инвертированными битами. Посчитаем две суммы: $\sum_{i=1}^{n} x_i$ и $\sum_{i=1}^{n} x'_i$, это можно сделать последовательно соединив полные сумматоры (см. схему ниже). Данные суммы удастся посчитать с помощью n сумматоров. На выходе окажутся двоичные числа, состоящие из $n+1$ бит. Остаётся сравнить эти числа с использованием не более $O(n)$ гейтов, как показано в книге Н. К. Верещагина и А. Шень "Языки и Исчисления". Суммарная сложность схемы будет не более $O(n)$.



4 Оценки для формул де Моргана

В данном разделе верхние оценки для формул де Моргана строятся на основе совершенной дизъюнктивной нормальной формы (СДНФ).

Теорема 6. *Схема на основе СДНФ является формулой де Моргана.*

Доказательство. Рассмотрим некоторые характеристики СДНФ:

- Состоит из бинарных операций $\vee$ и $\wedge$, переменных и их отрицаний.
- Может быть представлена в виде бинарного дерева.

Такие же характеристики определяют формулу де Моргана, следовательно любая схема в СДНФ представима формулой де Моргана.

4.1 Верхняя оценка для MOD_2^n

Теорема 7. $Fsize(MOD_2^n) \leq (n+1)2^{n-1} - 1$

Доказательство. Всего существует 2^n входных наборов для булевой функции с n входами, из них ровно на $\frac{2^n}{2}$ наборах результатом MOD_2^n будет 1. Следовательно всего будет 2^{n-1} клоз СДНФ, каждый из которых будет состоять из n конъюнкций. Клозы связаны $2^{n-1} - 1$ дизъюнкциями. Итоговое количество гейтов формулы, построенной на основе такой СДНФ, будет составлять $(n+1)2^{n-1} - 1$.

4.2 Верхняя оценка для MOD_3^n

Теорема 8. $Fsize(MOD_3^n) \leq (n+1)\lfloor\frac{2^n}{3}\rfloor - 1$

Доказательство. Всего существует 2^n входных наборов для булевой функции с n входами, из них ровно на $\lfloor\frac{2^n}{3}\rfloor$ наборах результатом MOD_3^n будет 1. Следовательно всего будет $\lfloor\frac{2^n}{3}\rfloor$ клоз СДНФ, каждый из которых будет состоять из n конъюнкций. Клозы связаны $\lfloor\frac{2^n}{3}\rfloor - 1$ дизъюнкциями. Итоговое количество гейтов формулы, построенной на основе такой СДНФ, будет составлять $(n+1)\lfloor\frac{2^n}{3}\rfloor - 1$.

4.3 Верхняя оценка для MOD_4^n

Теорема 9. $Fsize(MOD_4^n) \leq (n+1)2^{n-2} - 1$

Доказательство. Всего существует 2^n входных наборов для булевой функции с n входами, из них ровно на $\frac{2^n}{4}$ наборах результатом MOD_4^n будет 1. Следовательно всего будет 2^{n-2} клоз СДНФ, каждый из которых будет состоять из n конъюнкций. Клозы связаны $2^{n-2} - 1$ дизъюнкциями. Итоговое количество гейтов формулы, построенной на основе такой СДНФ, будет составлять $(n+1)2^{n-2} - 1$.

5 Оценка на сложность коммуникационной игры

Теорема 10. *Для любой симметричной булевой функции сложность коммуникационной игры не больше* $log_2^2(n) - log_2(n)$.

Доказательство. Пусть f - симметричная булева функция, A - строка Алисы, B - строка Боба, $A \in f^{-1}(1), B \in f^{-1}(0)$. $a_1, a_2, \ldots, a_n$ и $b_1, b_2, \ldots, b_n$ - биты соответствующих строк. $\sum_{i=1}^{n} a_i \neq \sum_{i=1}^{n} b_i$, так как симметричные булевы функции не могут выдавать различные значения при одинаковой сумме входного набора.

Реализуем протокол на основе бинарного поиска: Алиса отправляет Бобу сумму битов одной половины диапазона поиска (изначально диапазон поиска равен A), Боб сравнивает эту суммы с суммой соответствующего диапазона своей строки и отправляет 0, если если она различается, 1 - если нет. Затем Алиса, исходя из ответа Боба, сужает диапазон поиска вдвое. Этот алгоритм повторится $log_2(n)$ раз, так как каждый раз диапазон поиска сужается вдвое. У Алисы уйдёт не более $log_2(\frac{n}{2})$ бит на кодирование и отправку каждой суммы, потому что половина диапазона поиска не может быть меньше $\frac{n}{2}$. Следовательно итоговая сложность протокола будет не больше $log_2(n) \cdot log_2(\frac{n}{2}) = log_2^2(n) - log_2(n)$.

```
def Alice():
    l = 0
    r = n + 1
    while r - l > 1:
        s = 0
        m = (l + r) // 2
        for i in range(l + 1, m + 1):
            s += A[i]
        if Bob(l + 1, m + 1, s):
            l = m
        else:
            r = m
```

Листинг 1: Псевдокод действий Алисы

```python
def Bob(l, r, s):
    sm = 0
    for i in range(l, r):
        sm += B[i]
    return sm == s
```

Листинг 2: Псевдокод действий Боба

6 Идеи по улучшению методов поиска оценок

В данном проекте особый интерес для меня представляют программные методы поиска верхних оценок функций или блоков, которые помогут эти оценки улучшить. Из-за огромного поля поиска, уже при 25 входах, ни один современный суперкомпьютер не может перебрать все возможные схемы (за разумное время), поэтому я считаю, что необходимо найти новые пути оптимизации процесса поиска схем SAT-солверами. Для достижения приемлемой вычислительной скорости можно воспользоваться следующей эвристикой: использовать модель машинного обучения, для предсказания наиболее рациональной структуры схем из определённого класса булевых функций. Модели машинного обучения используются в задачах, где важно быстро распознавать паттерны, симметричные булевы функции как раз имеют множество закономерностей, поэтому интеграция ML-моделей может улучшить производительность SAT-солверов в задаче поиска оптимальных схем из функциональных элементов.