

МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
имени М.В. ЛОМОНОСОВА ФИЛИАЛ В ГОРОДЕ ТАШКЕНТЕ

ФАКУЛЬТЕТ ПРИКЛАДНОЙ МАТЕМАТИКИ И
ИНФОРМАТИКИ КАФЕДРА ПРИКЛАДНОЙ МАТЕМАТИКИ
И ИНФОРМАТИКИ

Ванесян Роман Грачинович

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

«Оптическое распознавание схем из
функциональных элементов»

Научный руководитель,
к.ф.-м.н. _____ Шуткин Ю.С.

«_____» _____ 2020 г.

ТАШКЕНТ - 2020

1 Аннотация

В данной работе рассматривается проблема распознавания схемы из функциональных элементов. Схема из функциональных элементов — это ориентированный граф с вершинами специального вида и ребрами, — жордановыми дугами. Распознавание схемы происходит в 4 этапа: предварительная обработка данных, сегментация, распознавание и постобработка данных. Результатом данной работы является прототип программы, которая показывает эффективность подхода описанного в данной работе. Результатом данной программы может быть как текстовая запись формулы представленной данной схемой из функциональных элементов, так и ее оцифрованная версия.

Содержание

1	Аннотация	1
2	Введение	3
3	Постановка задачи	5
4	Оптическое распознавание схем из функциональных элементов	6
4.1	Предварительная обработка данных	6
4.2	Сегментация	7
4.3	Распознавание	12
4.3.1	Распознавание меток	12
4.3.2	Топологическое распознавание СФЭ	13
4.4	Постобработка данных	18
5	Организация работы программы	19
6	Результаты тестирования работы программы	20

2 Введение

Пусть задано некоторое множество булевых функций

$$B = f_1(x_1, \dots, x_{n_1}), \dots, f_s(x_1, \dots, x_{n_s}),$$

где $n_1, \dots, n_s \geq 0$.

Будем называть данное множество B *базисом*.

Определение 1. *Схемой из функциональных элементов (СФЭ) над стандартным базисом $B = \{x_1 \wedge x_2, x_1 \vee x_2, \neg x_1\}$ будем называть ориентированный граф без циклов $G = (V, E)$, для которого выполняются следующие условия:*

- *Каждая вершина $v \in V$ имеет полустепень захода $d(v)$, не превосходящую двух, то есть $d(v) \leq 2$;*
- *Каждая вершина $v \in V$ с полустепенью захода, равной 0, называется входной (или входом схемы) и ей приписывается некоторая булева переменная x_i ;*
- *Существует ровно одна вершина с полустепенью захода, равной 1 и приписанной меткой F , называемая выходом СФЭ;*
- *Все другие вершины называются внутренними вершинами схемы;*
- *Каждой внутренней вершине $v \in V$ с полустепенью захода, равной 1 приписывается (функциональный) элемент отрицания (\neg);*
- *Каждой внутренней вершине $v \in V$ с полустепенью захода, равной 2 приписывается либо (функциональный) элемент конъюнкции (\vee), либо (функциональный) элемент дизъюнкции (\wedge).*

Стоит отметить, что существуют СФЭ с базисами отличными от приведенного. Однако, для упрощения изложения в данной работе мы будем рассматривать СФЭ только со стандартным базисом.

При изображении схемы из функциональных элементов входы будем обозначать окружностями, внутри которых записаны

входные переменные x_i . Вершины являющиеся операциями, — треугольниками, внутри которых записаны обозначения соответствующих функций. А выход СФЭ будем помечать прямоугольником, внутри которого записана метка F . Выходы функций будем отмечать выходными ребрами — жордановыми дугами.

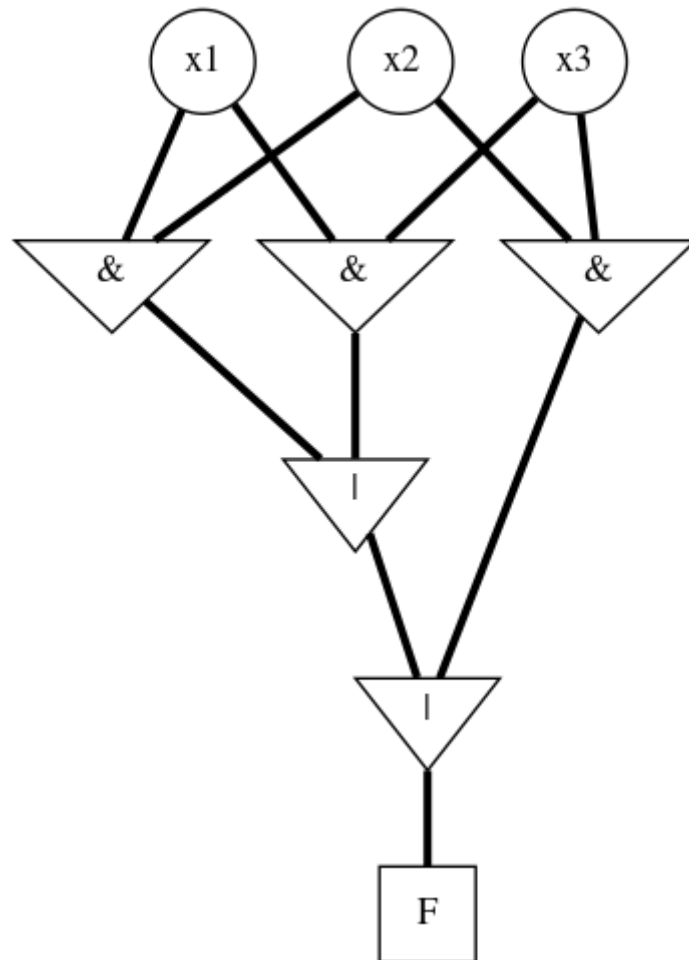


Рис. 1: Пример изображения схемы из функциональных элементов.

3 Постановка задачи

Определение 2. *Цифровым изображением будем называть конечное множество $I = \{p_i \mid p_i \in C\}$ на плоскости, где C — либо конечное множество кортежей арности 3, где каждый элемент имеет значение от 0 до 255, то есть:*

$$C = \{p_i \mid p_i = (x_{i_1}, x_{i_2}, x_{i_3}), 0 \leq x_{i_j} \leq 255, j = \overline{1, 3}\},$$

в таком случае будем называть цифровое изображение трехканальным цифровым изображением; либо конечное множество элементов, значение которых варьируется от 0 до 255, то есть:

$$C = \{p_i \mid p_i \in \mathbb{Z}, 0 \leq p_i \leq 255\},$$

в таком случае будем называть цифровое изображение одноканальным цифровым изображением (полутонное изображение). Элементы данного множества p_i будем называть пикселями.

Определение 3. *Одноканальное цифровое изображение, для которого множество C определено как $\{0, 1\}$ будем называть бинарным цифровым изображением (бинарным изображением).*

На вход программе подается цифровое изображение с рисунок схемы из функциональных элементов (СФЭ) на ней. Выходом программы является либо текстовая запись формулы представленной СФЭ, либо ее оцифрованное представление.

4 Оптическое распознавание схем из функциональных элементов

Оптическое распознавание СФЭ будем проводить в 4 этапа: предварительная обработка цифрового изображения, сегментация, распознавание, постобработка данных.

4.1 Предварительная обработка данных

Как и в любой задаче распознавания визуальных образов этап предварительной обработки направлен на разделение пикселей на два класса: “фоновые” пиксели и “полезные” пиксели (пиксели образующие исследуемый объект).

Пусть на вход программе было подано трехканальное цифровое изображение. Применяя, к примеру, следующую функцию [1]:

$$f(x_1, x_2, x_3) = \lfloor 0.299 * x_1 + 0.587 * x_2 + 0.114 * x_3 \rfloor$$

к каждому пикселю $p_i = (x_{i_1}, x_{i_2}, x_{i_3})$ преобразуем исходное трехканальное цифровое изображение в одноканальное цифровое изображение (полутоновое изображение). Далее, к полученному полутоновому изображению, применяя алгоритм бинаризации с использованием метода Оцу [2] для нахождения оптимального порога бинаризации за счет минимизации внутриклассовой дисперсии, получаем бинарное изображение.

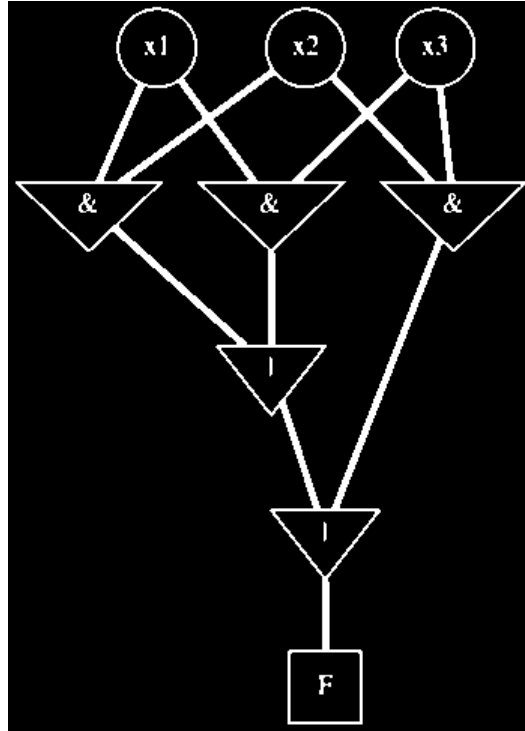


Рис. 2: Пример применения алгоритма бинаризации.

4.2 Сегментация

Входными данными для процесса сегментации является бинаризованное изображение.

Положим D — множество всех пикселей бинаризованного изображения.

Определение 4. Множество пикселей $V = \{p_i \mid p_i \in D\}$ будем считать вершиной, если для него выполняются следующие 3 условия:

1. Пусть $F \subset V$. Множество $H = \{p_i \mid p_i \in F, p_i = 1\}$ образует одну из следующих фигур: треугольник, окружность, либо прямоугольник.
2. Пусть $G = F \setminus H$. Существует такое подмножество $M = \{p_i \mid p_i \in G, p_i = 1\}$ — метка вершины.
3. Никакие две рядом лежащие вершины не расположены так, что пересечение минимальных описывающих прямоугольников, содержащих соответствующие вершины есть множество не пустое, то есть:
 $\forall V_i, V_j, \quad V_i \neq V_j, i \neq j : P(V_i) \cap P(V_j) = \emptyset.$

Проверим условие 3. Положим

$$s(p_{11}, p_{12}, p_{21}, p_{22}) = \begin{cases} 1, & (p_{11} \leq p_{21} \leq p_{22} \leq p_{12}) \wedge \\ & \wedge (p_{11} \leq p_{21} \leq p_{22} \leq p_{12}) \\ 0, & otherwise \end{cases},$$

где (p_{11}, p_{12}) и (p_{21}, p_{22}) пары левой-верхней и правой-нижней точек, задающих соответствующие прямоугольники. Правило порядка для точек задано по координатам.

Применим попарно правило $s(p_{11}, p_{12}, p_{21}, p_{22})$ к элементам множества описывающих минимальных прямоугольников соответствующих вершин. Если для какой-то пары прямоугольников правило $s(p_{11}, p_{12}, p_{21}, p_{22})$ дало значение 1, то исключим фигуру описанную прямоугольником с большей площадью из множества вершин.

Проверим условие 2. Положим

$$c(p_{11}, p_{12}, p_{21}, p_{22}) = \begin{cases} 1, & (p_{11} \leq p_{21} \leq p_{12}) \wedge (p_{11} \leq p_{22} \leq p_{12}) \\ 0, & otherwise \end{cases},$$

где аналогично (p_{11}, p_{12}) и (p_{21}, p_{22}) пары левой-верхней и правой-нижней точек, задающих соответствующие прямоугольники.

Применяя попарно указанное правило к множеству минимальным описывающих прямоугольником однозначно проверим условие 2. То есть, если найдется такой минимальный описывающий прямоугольник не содержащий никаких других прямоугольников, то исключим фигуру вписанную в данный прямоугольник из множества вершин.

Проверим условие 1. Для проверки условия 1, для каждого описывающего минимального прямоугольника пометим все пиксели входящие во внутренность описываемой фигуры цветом 1 (то есть присвоим им значение равное 1), все остальные пиксели прямоугольника - цветом 0. Размер каждого такого изображения есть $w \times h$, где w — ширина, а h — высота. Построим 3 бинарных изображения размера $w \times h$ с рисунками: окружности с радиусом $r = \frac{\min(w, h)}{2}$ и центром в точке $(\frac{w}{2}, \frac{h}{2})$, прямоугольника с шириной w и высотой h , треугольника заданного координатами $\{(0, h), (\frac{w}{2}, 0), (w, h)\}$. Аналогично фоновые пиксели пометим цветом 0, а внутренность с контуром каждой из фигур в цвет 1.

Очевидно, что полученные изображения могут быть заданы матрицами размерности $w \times h$.

Положим

$$\delta(A, B) = \sum_{i=1}^h \sum_{j=1}^w a_{ij} \oplus_2 b_{ij}, \quad a_{ij} \in A, b_{ij} \in B$$

Пусть матрица A задает изображение вершины. Тогда будем считать, что на матрице A изображена одна из трех фигур (окружность, прямоугольник, треугольник), если сумма $\delta(A, B)$ - минимальна и не превышает некоторого заданного числа $k \in \mathbb{N}$.

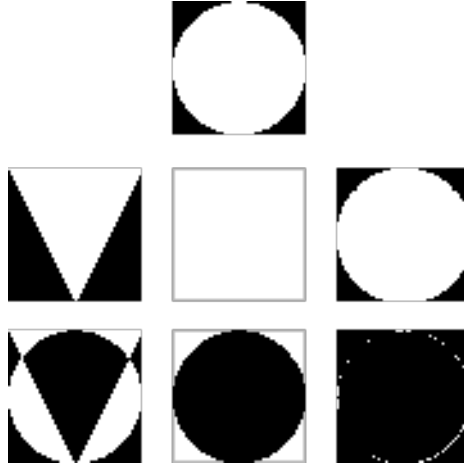


Рис. 4: Пример результата работы алгоритма проверки условия 1.

Таким образом, последовательно применяя алгоритмы проверки условия 1-3 к каждому элементу множества фигур, полученного в ходе работы алгоритма выделения замкнутых контуров, получим множество искомых вершин СФЭ.

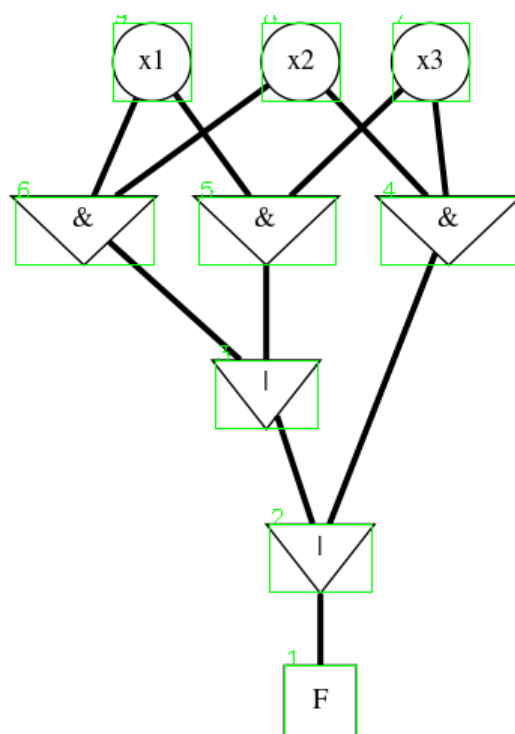


Рис. 5: Пример результата работы алгоритма выделения вершин СФЭ.

Выходом алгоритма сегментации является бинарное изображение с фигурами вершин.

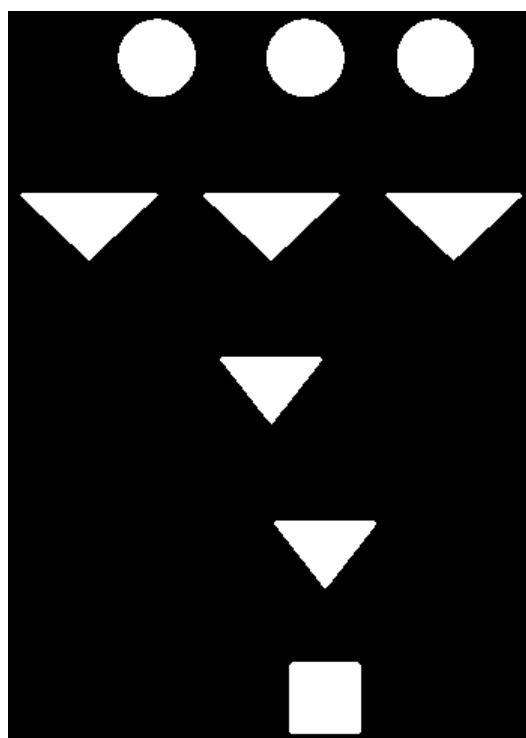


Рис. 6: Пример выхода алгоритма сегментации СФЭ.

4.3 Распознавание

Этап распознавания разобьём на два подэтапа: распознавание меток в вершинах СФЭ и топологическое распознавание СФЭ.

4.3.1 Распознавание меток

Выходными данными алгоритма примененного на этапе сегментации является бинарное изображение с фоновыми пикселями, имеющими значение 0 и внутренностями вершин с контуром, имеющими значения 1. Таким образом, на самом деле, данное изображение является еще и маской для исходного изображения. А стало быть и для его бинаризованного вида. Положим M — множество пикселей маски, а B — множество пикселей бинаризованного изображения. Положим множество L — результат применения операции конъюнкции к каждому пикселям $p_{1i} \in D$ и $p_{2i} \in M$, то есть

$$L = \{p_i \mid p_i = p_{1i} \wedge p_{2i}, p_{1i} \in D, p_{2i} \in M\}.$$

L — бинарное изображение содержащее вершины и их метки заданной СФЭ.

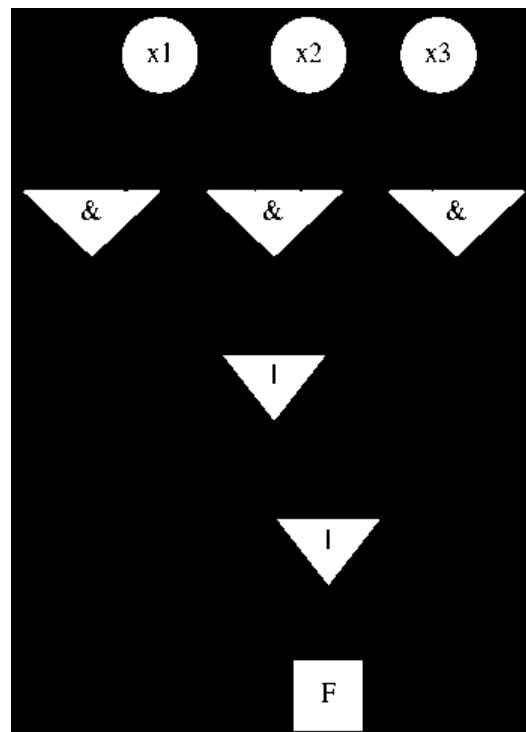


Рис. 7: Пример изображения полученного в результате наложения маски на бинаризованное изображения содержащее СФЭ.

Далее, будем рассматривать каждую вершину по отдельности. Возьмем минимальный прямоугольник содержащий данную фиксированную вершину с меткой, обозначим его через r_1 . Такой прямоугольник содержит два класса пикселей: фоновые и полезные пиксели. Для удаления фоновых пикселей, инвертируем пиксели в прямоугольнике содержащем множества пикселей вершин без метки обозначим, такое множество за r_2 . Применим попиксельно операцию дизъюнкции к r_1 и r_2 . Получили множество пикселей, где фоновые пиксели имеют значение 1, а пиксели образующие метку — значение 0.



Рис. 8: Пример результата работы алгоритма выделения метки.

Далее, к полученной изображению, применяя, к примеру, алгоритм распознавания текста основанный на нейронной сети с долгой краткосрочной памятью (в английской литературе LSTM based OCR) [4, 5] получим текстовую запись метки данной вершины.

Таким образом, применяя приведенный алгоритм к каждой вершине СФЭ, получим все текстовые записи меток.

4.3.2 Топологическое распознавание СФЭ

Определение 5. *Ребром будем называть жорданову дугу образованную последовательностью точек $p_i = 1$ и соединяющее вершины $v_i, v_j, i \neq j$.*

Входными данными для топологического распознавания СФЭ является изображение полученное в результате дизъюнкции бинаризованное изображения СФЭ с выходом алгоритма примененного на этапе сегментации — бинарным изображением, содержащим исключительно вершины СФЭ.

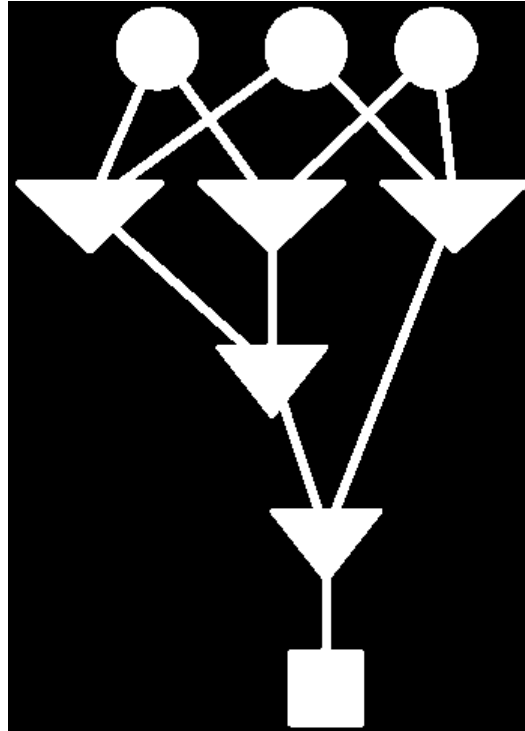


Рис. 9: Пример входного изображения для алгоритма топологического распознавания СФЭ.

Алгоритм топологического распознавания СФЭ приведенный в данной работе основан на алгоритме топологического распознавания графа приведенного в работе [6]. Алгоритм топологического распознавания графа состоит из 3 частей: *построение скелета графа*, *классификация пикселей (точек) графа*, *обход графа*.

Рассмотрим процесс построение скелета графа. Идея построения скелета объекта заключается в удалении пикселей из множества пикселей изучаемого объекта таким образом, чтобы не нарушать топологию самого объекта. Результатом данной операции будет являться *скелет* объекта. Важным свойством процесса построения скелета объекта является сохранение свойства связности пикселей объекта. Это свойство как раз и позволяет нам использовать данный процесс при топологическом распознавании графа.

Построение скелета графа основано на применении алгоритма предложенного в статье [7]. Идея данного алгоритма заключается в изменении значений пикселей с 1 на 0 контура объекта до тех пор, пока связность объекта не будет нарушена.

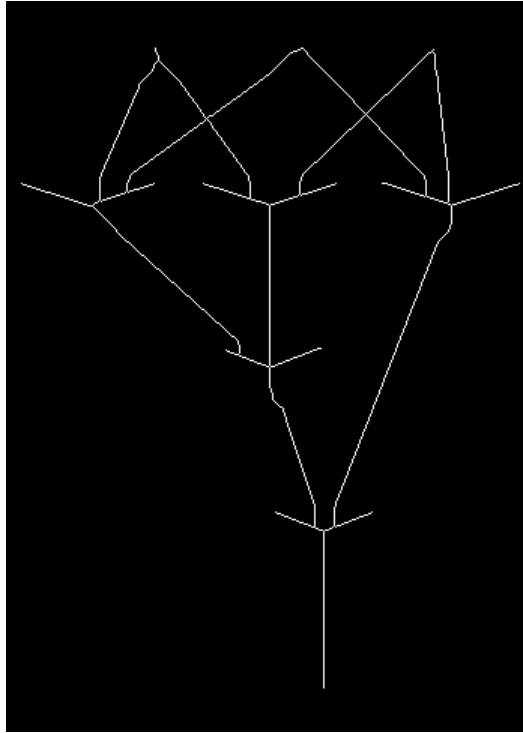


Рис. 10: Пример результата работы построения скелета графа изображенного на рис. 9.

Рассмотрим процесс классификации пикселей графа. Будем классифицировать каждый пиксель изображения как пиксель относящийся к одному из следующих классов: P_v — класс пикселей входящих в одну из вершин графа (будем называть такой класс пикселей *классом «порт» пикселей*), P_e — класс пикселей представляющих собой пиксели одного из ребер графа, P_c класс пикселей включающих в себя пиксели на пересечении двух ребер (такой класс пикселей будем называть *классом «кросс» пикселей*) и P_b — класс *класс «фоновых» пикселей*, — пиксели не входящие ни в один из приведенных ранее классов.

Определение 6. Будем называть *4-окрестностью* пикселя p_i с координатами (x, y) множество состоящее из пикселей с координатами $(x - 1, y)$, $(x + 1, y)$, $(x, y - 1)$, $(x, y + 1)$.

Напомним, что результатом построения скелета объекта изображенного на бинарном изображении является бинарным изображением.

Так как ранее, на этапе сегментации, мы однозначно выделили множество пикселей образующих вершины (с их внутренностью), и скелет объекта включает так же данные пиксели (некоторое их

подмножество), то мы можем однозначно отнести данные пиксели к классу P_v .

Пусть P_r множество пикселей 4-окрестности пикселя p_i .

Положим

$$n_0(p_i) = \sum_{p_j \in P_r} p_j.$$

Будем считать, что пиксель p_i относится к классу P_b , если $n_0(p_i) \leq 1$, иначе если $n_0(p_i) = 2$, то p_i лежит в классе P_e , иначе при $n_0(p_i) \geq 2$ пиксель лежит в классе P_c .

Таким образом, применяя изложенный алгоритм классификации к каждому пикселю получим 4 класса. При том, ни один из классов не содержит пиксели другого класса.

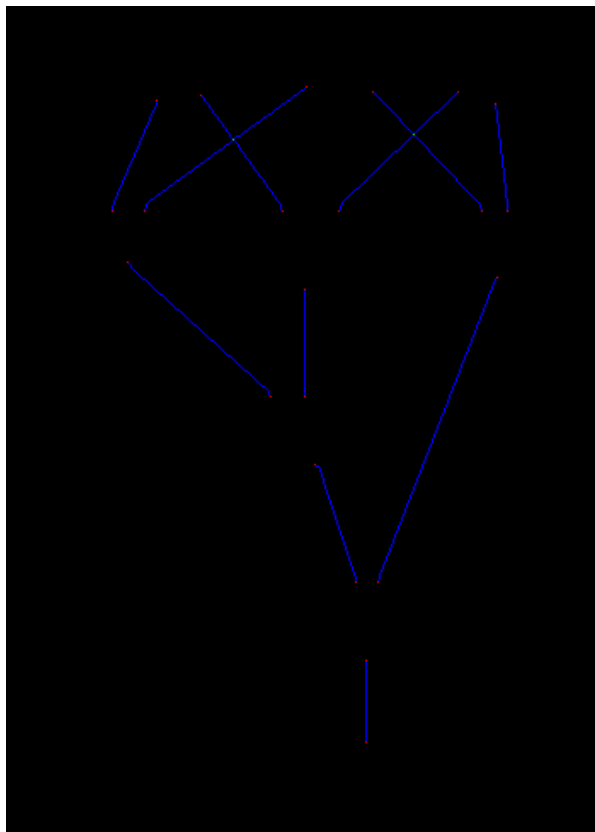


Рис. 11: Пример работы алгоритма классификации пикселей графа. Синим цветом выделены пиксели принадлежащие классу P_e , зеленым — к классу P_c , красным — к классу P_v , а черным — к классу P_b .

Очевидно, что ребро всегда состоит из множества пикселей класса P_e , “соединенное” на концах с ровно одним пикселем из мно-

жества P_v .

Рассмотрим процесс обхода графа.

Определение 7. Будем называть 8-окрестностью пикселя p с координатами (x, y) множество $N_8(p)$ состоящее из пикселей с координатами $\{(x-1, y-1), (x-1, y), (x-1, y+1), (x, y-1), (x, y+1), (x+1, y-1), (x+1, y), (x+1, y+1)\}$.

Обход графа будем всегда начитать с пикселя принадлежащего классу P_v . Обход может быть осуществлен по всем пикселям за исключением пикселей принадлежащих классу P_b . Каждый уже пройденный пиксель будем помечать как пройденный. Каждый следующий пиксель $p_{i+1} \in N_8(p_i)$ будем выбирать таким образом, что выбранный пиксель p_{i+1} не являлся бы “пройденным”.

Тривиальным случаем является случай, когда ребро соединяющее две вершины v_i и v_j не пересекается ни с какими другими ребрами. Для каждого такого тривиального случая будем считать, что между вершина v_i и v_j , $i \neq j$ проведено ребро.

Направлением будем называть вектор $\vec{v} = \overrightarrow{p_i, p_{i+1}} = (x_{i+1} - x_i, y_{i+1} - y_i)$. *Частотой* вектора \vec{v} будем называть количество повторений вектора \vec{v} в n последних направлениях. Обозначим последовательность n последних направлений за $\{\vec{v}_i\}_{i=1}^n$. Отметим, что последовательность $\{\vec{v}_i\}_{i=1}^n$, есть упорядоченная последовательность по частоте, таким образом, что направления с большими частотами идут вначале.

Рассмотрим случай, когда ребро пересекает другое ребро в каком-то фиксированном пикселе $p_k \in P_c$. Аналогично, обход графа начинаем с пикселя из класса P_v и продолжаем “идти” по ребру, — по пикселям класса P_e до тех пор, пока не “встретим” пиксель p_k . Если существует такой еще не “пройденный” пиксель $p_{i+1} \in N_8(p_k)$ и $p_{i+1} \in P_e \cup P_c$, по одному из направлений $\vec{v}_h \in \{\vec{v}_i\}_{i=1}^n$, то выберем данный пиксель p_{i+1} как следующий пиксель за пикселем p_i ; иначе, если такого пикселя не существует, выберем пиксель $p_{i+1} \in N_8(p_k)$, такой что расстояние

$$\rho(p_1, p_2) = \sqrt{\alpha(x_1 - x_2)^2 + \beta(y_1 - y_2)^2}$$

от p_i минимально. Здесь параметры α и β определяются как сред-

ние количества изменений направления по осям X и Y соответственно.

Таким образом, применяя данный алгоритм для каждой вершины, то есть еще не пройденным пикселям $p_j \in P_v$ обойдем весь граф.

4.4 Постобработка данных

На данном этапе мы интерпретируем накопленные данные либо в текстовую форму записи формулы представленной СФЭ, либо ее оцифрованное изображение. Для этого запустим алгоритм поиска в глубину [11] от выходной вершины СФЭ и будем интерпретировать вершины в зависимости от их значений. Отметим, что для бинарных операций (вершин с полустепенью захода $d(v) = 2$) последовательность входов будем считать по координате x , пикселей $p_i \in P_v$ соединенных относительно начала координат.

5 Организация работы программы

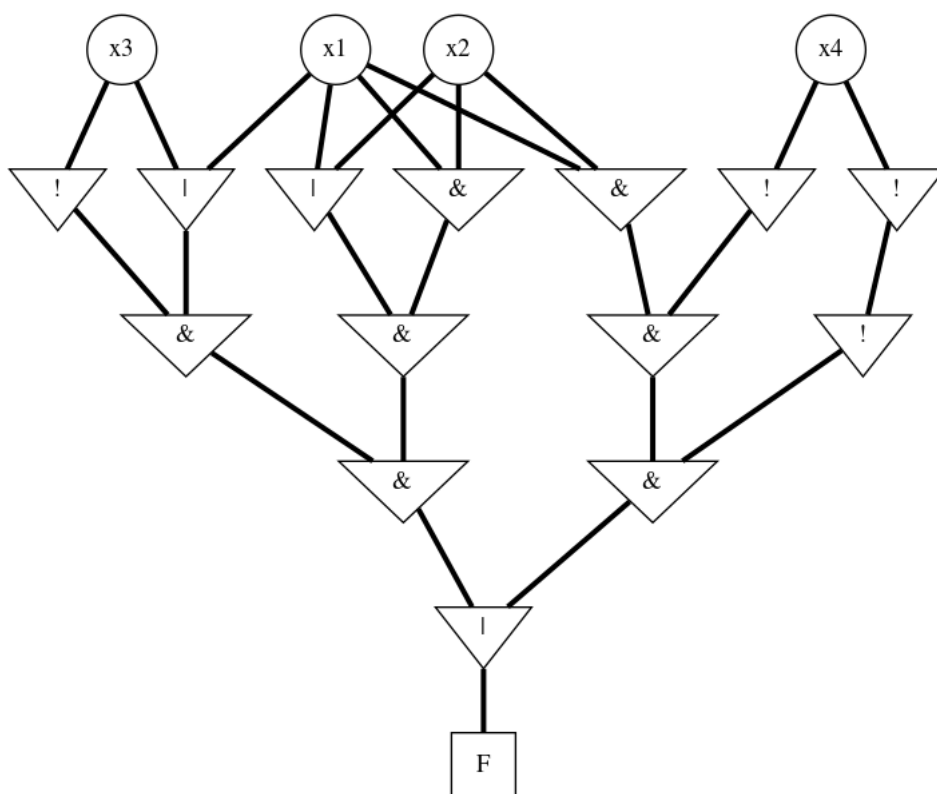
Программа для реализации алгоритма распознавания СФЭ приведенного в данной работе была написана на языке Python 3. Для реализации промежуточных алгоритмов на стадиях бинаризации и сегментации была использована библиотека OpenCV 4 [8]. Для реализации алгоритма распознавания символов была использована библиотека Tesseract 4 [5]. Для представления оцифрованной версии СФЭ используется программа Graphviz [9].

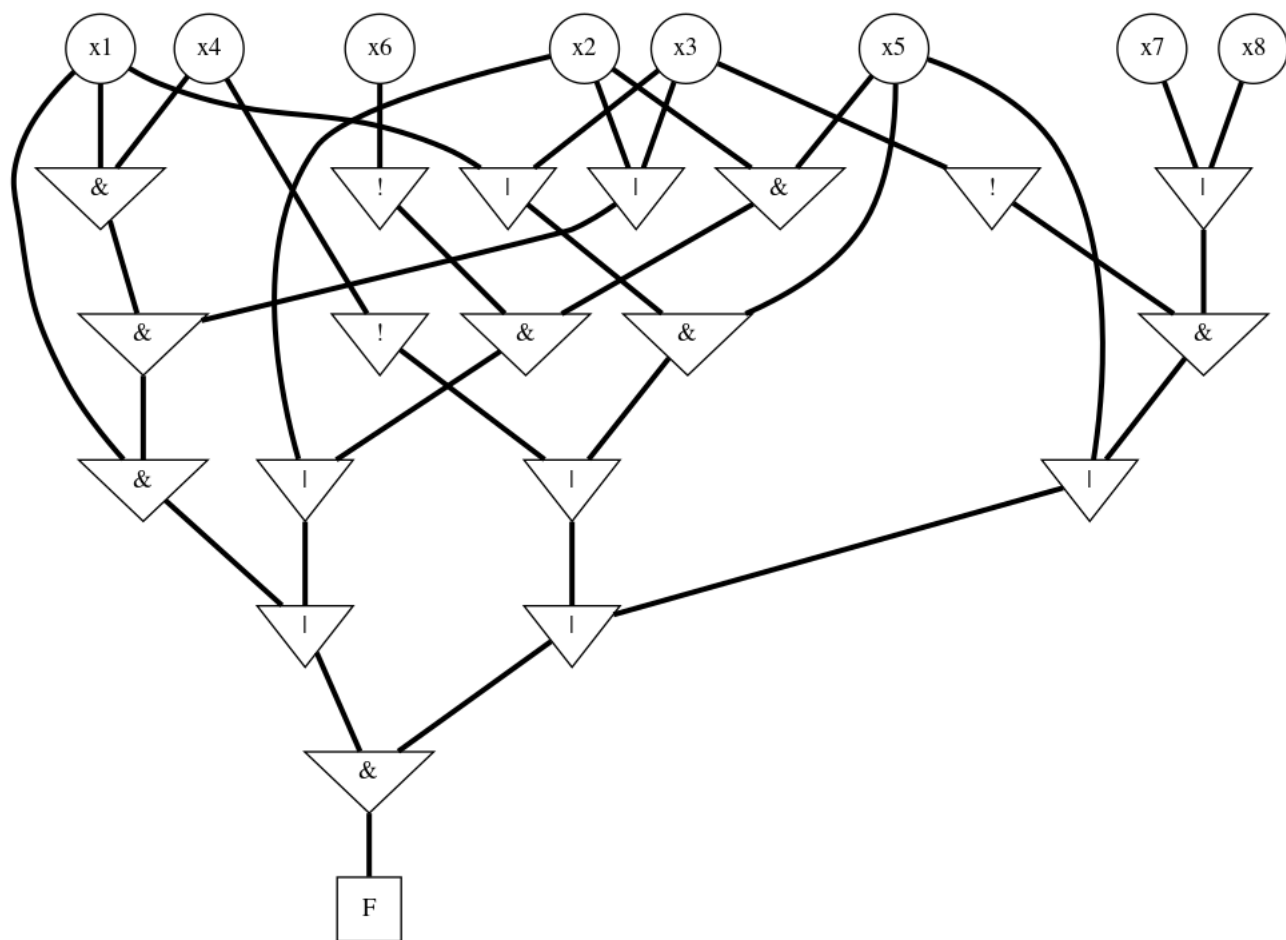
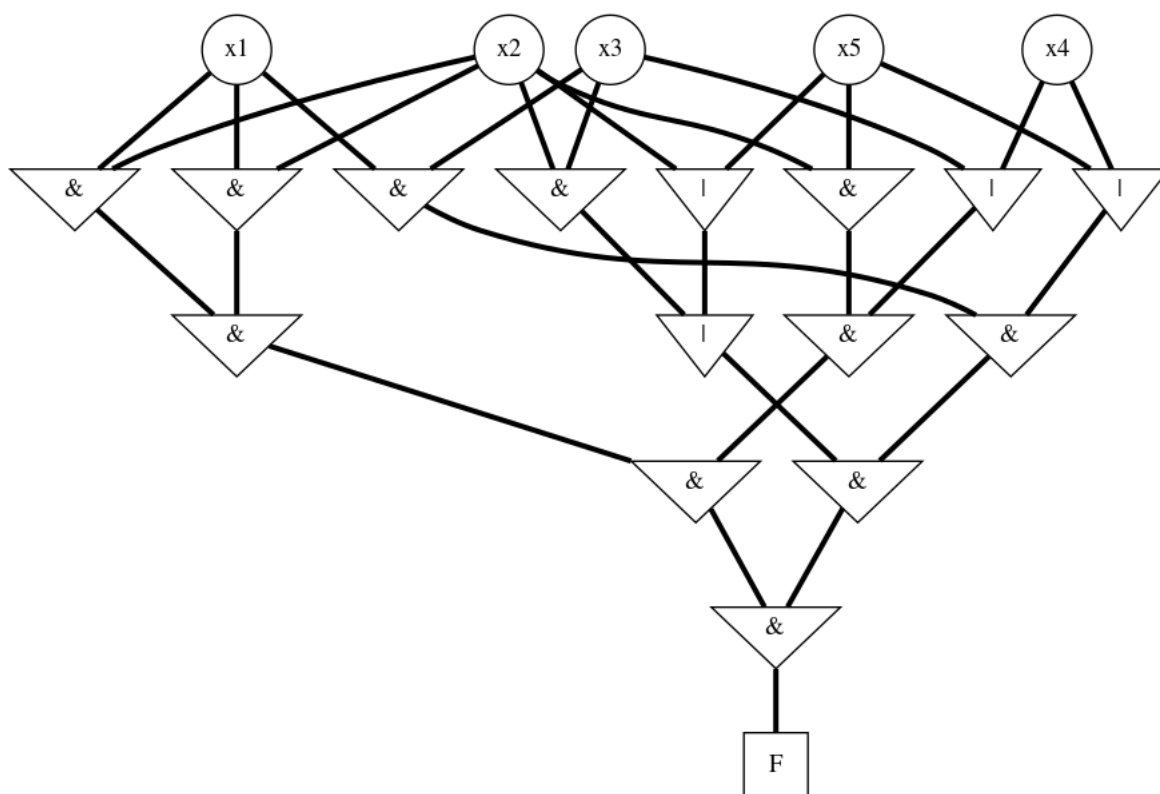
Для интерпретации полученных данных была написана вспомогательная библиотека с реализацией структуры для построения Binary Expression Tree [10], и последующего его обхода, используя алгоритм поиска в глубину [11] с применением шаблона проектирования «посетитель» (в английской литературе Visitor pattern) [12]. Таким образом, данная библиотека позволяет интерпретировать Binary Expression Tree построенное из полученных данных как в графическом представлении (то есть интерпретирует в цифровое изображение), так и в текстовой записи.

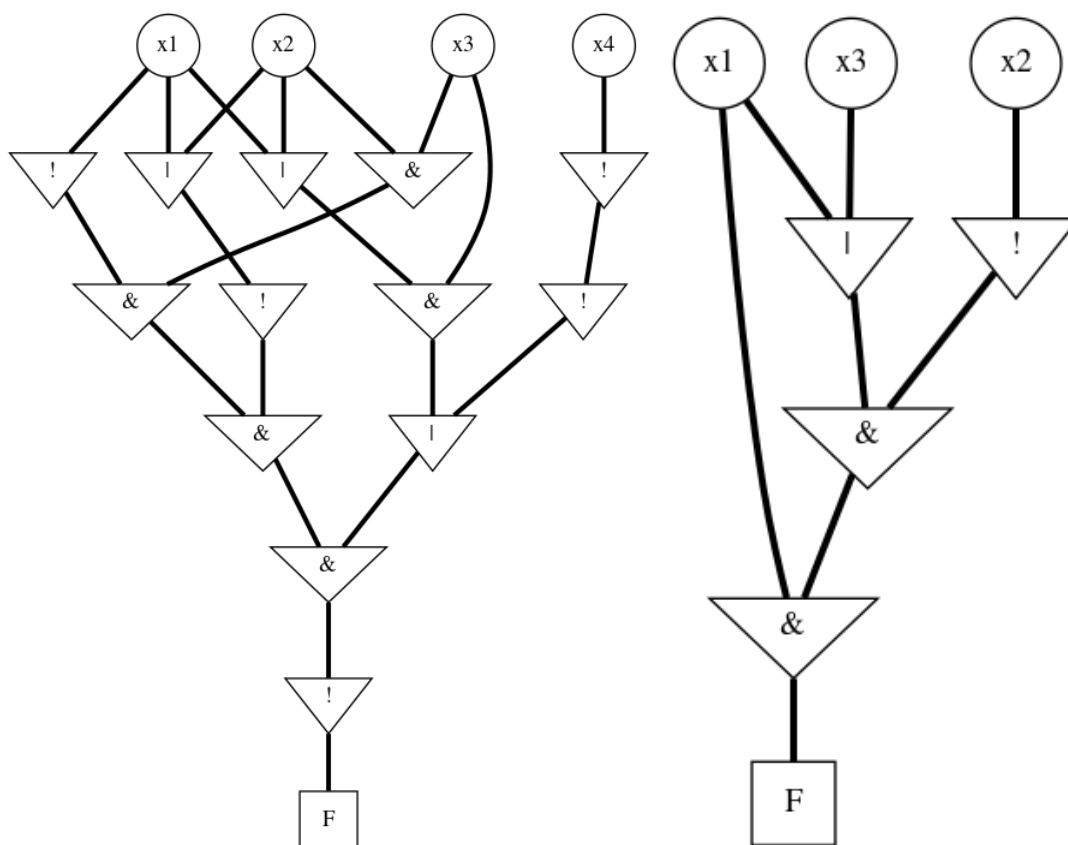
6 Результаты тестирования работы программы

Для подтверждения эффективности алгоритма распознавания СФЭ предложенного в данной работе была составлена тестирующая программа. Данная программа случайным образом генерировала СФЭ с различной сложностью (количеством вершин, пересечений ребер и т.д.). Тестовые испытания проводились на данных размеров 20, 30 и 50. При том, каждый тест запускался 3 раза. Так для тестового массива данных с размерностью 20, вероятность корректно распознавания СФЭ составила $\frac{14}{20}$, с размерностью 30 — $\frac{64}{90}$, с размерностью 50 — $\frac{115}{150}$. Так было полученное, что топология СФЭ без самопересечений распознается с вероятностью 1, для СФЭ с самопересечениями вероятность составила 0.7.

Приведем изображения некоторых успешно распознанных СФЭ.







Проблему для топологического распознавания СФЭ составляют случаи с пересечениями ребер как, например, на рисунке 12.



Рис. 12: Пример пересечения ребер ведущих к некорректному распознаванию СФЭ.

В процессе построения скелета СФЭ, два участка “1” и “2” различных ребер “сливаются” в один контур на участке “3” как изображено на рисунке 13.

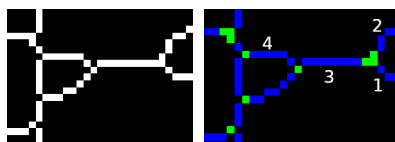


Рис. 13: Пример части скелета СФЭ ведущих к некорректному распознаванию СФЭ.

Это ведет к тому, что при прохождении участка “3” и при достаточно малом n (напомним, что это параметр, который указывает

на максимальную длину последовательности последних n векторов направления, использующийся в алгоритме топологического распознавания графа.), частота вектора направления $\vec{v} = (-1, 0)$, будет максимальной, что приведет к тому, что для обоих участков “1” и “2” двух различных ребер продолжением будет считаться один и тот же участок “4”.

Одним из способов решения данной проблемы является выбор n , таким, что частота вектора направления $\vec{v} = (-1, 0)$ не будет максимальной. Однако, экспериментальным путем было выявлено, что $n = 4$, является оптимальным для большего количества задач.

Рассмотрим следующую проблему, — проблему корректной сегментации СФЭ.

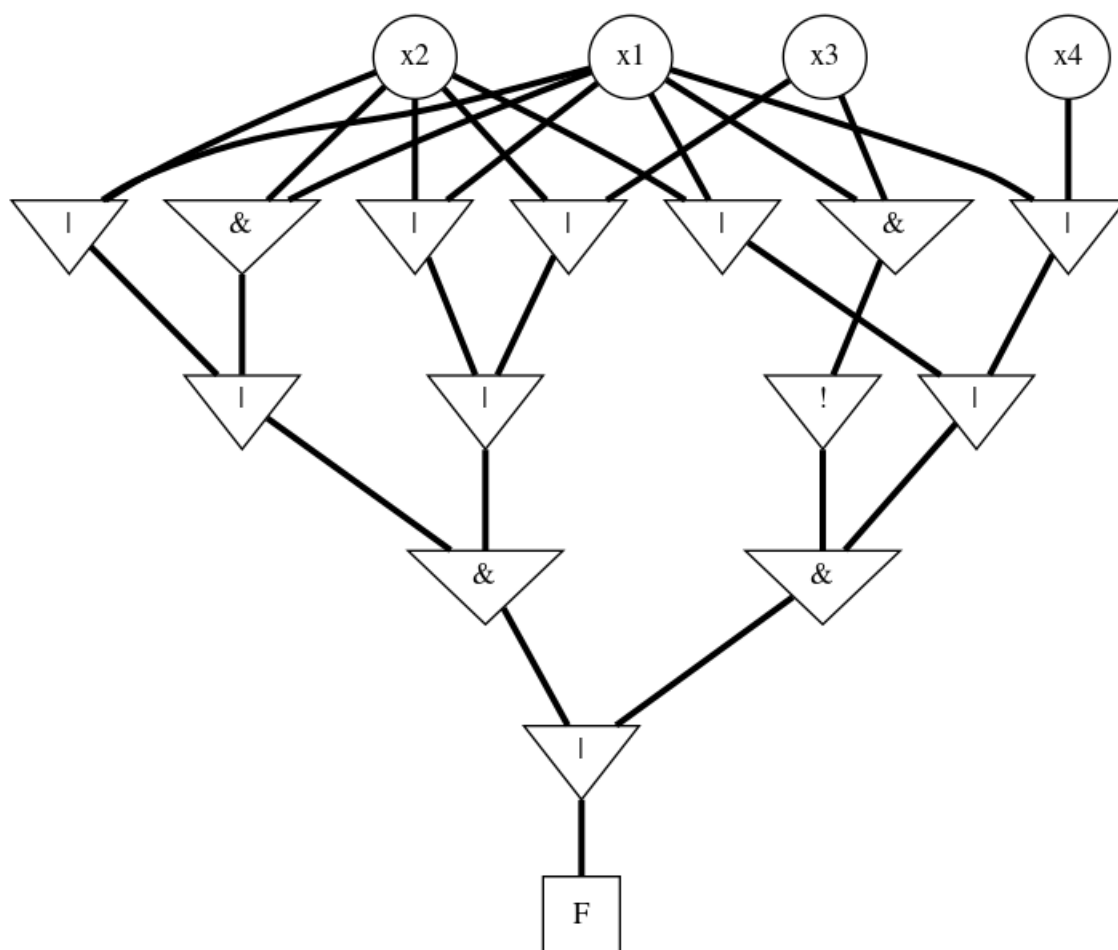


Рис. 14: Пример СФЭ порождающей проблему сегментации (корректного выделения вершин) СФЭ.

На рисунке 14 приведено изображение СФЭ, которая влечет

проблему сегментации. На рисунке 15 приведен результат алгоритма детекции вершин примененного к изображению СФЭ на рисунке 14.

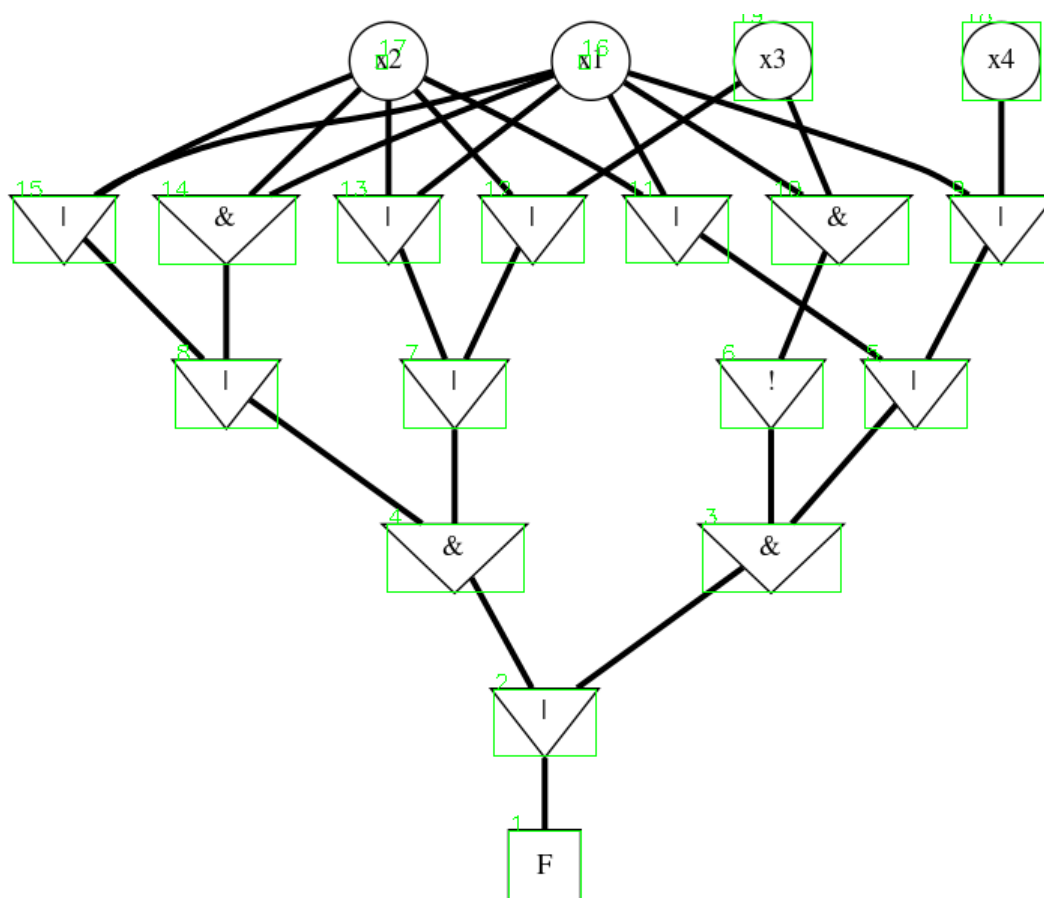


Рис. 15: Пример СФЭ порождающей проблему сегментации (корректного выделения вершин) СФЭ.

Как можно видеть, вершины с метками “x2” и “x1” не были выделены. Это произошло по причине того, что вокруг них имеется превалирующее количество фигур образованных пересечением ребер. Так как в алгоритме выделения вершины для каждой фигуры проверяются все 3 условия, то при проверке условия 3 (об отсутствии пересечений минимальных описывающих прямоугольников с другими минимальным описывающими прямоугольниками) для вершин с метками “x2” и “x1” их минимальные описывающие прямоугольники будут пересекаться с описывающими прямоугольниками фигур, образованных пересечением ребер и при том площади данных вершин будут большими. Таким образом вершины с метками “x2” и “x1” будут исключены из множества вершин.

Одним из способов коррекции данного поведения, является про-

пуск проверки вершины на условия 3. Однако, такое решение может повлечь проблемы при детекции вершин в других задача распознавания СФЭ.

Список литературы

- [1] [https://en.wikipedia.org/wiki/Luma_\(video\)#Rec._601_luma_versus_Rec._709_luma_coefficients](https://en.wikipedia.org/wiki/Luma_(video)#Rec._601_luma_versus_Rec._709_luma_coefficients)
- [2] N. Otsu. *A threshold selection method from gray-level histograms. IEEE Trans. Sys., Man., Cyber. : journal. — 1979. — Vol. 9. — P. 62–66.*
- [3] Suzuki, S. and Abe, K., *Topological Structural Analysis of Digitized Binary Images by Border Following. CVGIP 30 1, pp 32-46 (1985)*
- [4] Adnan Ul-Hasan *Generic Text Recognition using Long Short-Term Memory Networks* https://kluedo.ub.uni-kl.de/frontdoor/deliver/index/docId/4353/file/PhD_Thesis_Ul-Hasan.pdf
- [5] <https://tesseract-ocr.github.io/>
- [6] Ch. Auer, Ch. Bachmaier, F. Gleißner, and J. Reislhuber *Optical Graph Recognition, section 3.3, p. 6-9.* https://link.springer.com/content/pdf/10.1007%2F978-3-642-36763-2_47.pdf
- [7] T. Y. Zhang, C.Y. Suen *A Fast parallel algorithm for thinning digital petterns.* http://agcggs680.pbworks.com/f/Zhan-Suen_algorithm.pdf
- [8] <https://opencv.org/opencv-4-0/>
- [9] <https://www.graphviz.org/>
- [10] https://en.wikipedia.org/wiki/Binary_expression_tree
- [11] https://en.wikipedia.org/wiki/Tree_traversal#Depth-first_search
- [12] https://en.wikipedia.org/wiki/Visitor_pattern