Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное
учреждение высшего образования
«Южно-Уральский государственный университет
(национальный исследовательский университет)»
Институт естественных и точных наук
Факультет математики, механики и компьютерных технологий
Кафедра прикладной математики и программирования
Направление подготовки: 01.03.02 Прикладная математика и информатика

РАБОТА ПРОВЕРЕНА
Рецензент, зам. директора
ВШЭКН по научной работе, д., к.т.н.
_____/Н.В. Плотникова
«____»_____20___г.

ДОПУСТИТЬ К ЗАЩИТЕ
Заведующий кафедрой, д.ф.-м.н.,
профессор
_____/А.А.Замышляева
«____»_____20___ г.

Разработка модуля оптимизации потока пациентов в информационной системе МАУЗ ГКБ №2

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
К ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЕ
ЮУрГУ–01.03.02.2020.094.ПЗ ВКР

Руководитель работы, ст.преп.
кафедры ПМиП
_____/М.Ю. Сартасова
«____»_____2020 г.

Автор работы
Студент группы ЕТ-413
_____/М.С. Самбурский
«____»_____2020 г.
Нормоконтролер,
ст. преподаватель
_____/Н.С. Мидоночева
«____»_____2020 г.

Челябинск
2020

АННОТАЦИЯ

Самбурский М.С. Разработка модуля оптимизации потока пациентов в информационной системе МАУЗ ГКБ №2. – Челябинск: ЮУрГУ, ЕТ-413, 43 с., 22 ил., библиогр. список – 30 наим.

Целью данной работы является разработка модуля оптимизации потока пациентов в информационной системе МАУЗ ГКБ № 2.

В первом разделе описывается процесс проведения медицинских осмотров, рассмотрены существующие медицинские информационные системы и другие средства оптимизации потока пациентов, проанализированы методы, применяемые для оптимизации расписаний. Приводится обоснование выбранного метода и средств реализации.

Во втором разделе были рассмотрены основные определения и термины генетических алгоритмов. Составлена математическая модель потока пациентов поликлиники.

В третьем разделе проведён сравнительный анализ различных конфигураций генетического алгоритма. Выполнено тестирование работы алгоритма на экспериментальных данных.

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Одним из направлений работы медицинских учреждений является проведение периодических медицинских осмотров для сотрудников различных предприятий. Для каждого такого сотрудника, в зависимости от его рода деятельности, поликлиника составляет списки врачей и анализов, необходимых для прохождения медицинского осмотра. В настоящее время такие списки не предлагают пациентам оптимального порядка прохождения специалистов и сдачи анализов. Отсутствие чётких маршрутов движения пациентов приводит к образованию длинных очередей в одни кабинеты, в то же время в другие кабинеты очереди нет. В результате одни врачи загружены больше, чем другие, время нахождения пациентов в поликлинике увеличивается, а её пропускная способность падает.

В целях экономии времени пациентов и увеличения пропускной способности поликлиники, целесообразно автоматизировать процесс генерации маршрутов для сотрудников организаций, проходящих медицинский осмотр.

Таким образом, цель данной работы заключается в разработке модуля оптимизации потока пациентов для информационной системы МАУЗ ГКБ №2.

Для достижения поставленной цели необходимо решить следующие задачи:

1) провести анализ существующих средств оптимизации потоков;

2) составить математическую модель потока пациентов;

3) определить метод, подходящий для решения поставленной задачи;

4) выполнить программную реализацию алгоритма;

5) провести тестирование на экспериментальных данных.

1 ЗАДАЧА ОПТИМИЗАЦИИ ПОТОКА ПАЦИЕНТОВ

1.1 Понятие медицинской информационной системы

Информационные процессы присутствуют во всех областях медицины и здравоохранения. Современные медицинские организации производят и накапливают огромные объёмы данных. Для повышения эффективной работы медицинского учреждения и качества медицинской помощи необходимо, чтобы эти данные были правильно структурированы. Кроме того, многие процессы работы с пациентами и другие задачи могут быть автоматизированы.

Медицинские информационные системы (МИС) – это комплексные системы электронного документооборота для медицинских учреждений. Опишем основные преимущества их использования:

1) ускорение обслуживания пациентов и, следовательно, увеличение пропускной способности медицинского учреждения за счёт компьютерной обработки данных, работа с которыми ранее велась с использованием бумажных носителей;

2) сокращение ошибок в процессах ввода и обработки информации за счёт форматно-логического контроля вводимых данных и автоматической генерации документации и отчётов;

3) надёжное хранение большого массива данных с возможностью оперативного доступа, организованным быстрым поиском и возможностью выборки необходимых данных.

1.2 Структура процесса проведения медосмотра

На основании Приказа Министерства здравоохранения и социального развития РФ от 12 апреля 2011 года №302Н «Об утверждении перечней вредных и (или) опасных производственных факторов и работ, при выполнении которых проводятся обязательные предварительные и периодические медицинские осмотры (обследования), и порядка проведения обязательных предварительных и периодических медицинских осмотров

(обследований) работников, занятых на тяжелых работах и на работах с вредными и (или) опасными условиями труда» медицинские учреждения проводят медицинские осмотры для сотрудников различных организаций. В соответствии с этим приказом каждому работнику, в зависимости от его должности и рода деятельности, назначается определённый шифр вредности. В приказе для каждого шифра описан перечень врачей и обследований, а также периодичность их прохождения.

Организация-работодатель направляет сотрудников на профосмотр. Для этого она предоставляет медицинскому учреждению список сотрудников с шифрами вредности. После, медицинское учреждение на основе этих шифров формирует для каждого пациента индивидуальный перечень врачей и анализов с учётом его пола и возраста.

Проходя осмотры и обследования на каждого пациента составляются соответствующие документы, записи о посещении врачей и результаты анализов попадают в медицинскую карту пациента. В конце медосмотра эти данные используются врачом-профпатологом для вынесения решения о пригодности человека к работе.

1.3 Обзор существующих средств оптимизации потока пациентов

Несмотря на то, что пациенты разделяются на основе шифров вредности, этого недостаточно для эффективного прохождения медосмотра. Необходимо каким-то образом направить поток пациентов так, чтобы минимизировать очереди в каждый конкретный кабинет при этом сократив время простоя других кабинетов [3]. Сделать это можно, например, составив расписание посещения каждого кабинета каждым пациентом. Рассмотрим некоторые существующие средства, решающие эту проблему.

1.3.1 Microsoft Excel

Наиболее простым решением является использование офисного приложения Microsoft Excel. Данная программа обладает функционалом для

оптимизации маршрута одного пациента, но не имеет встроенных средств для оптимизации расписаний в целом. Организовав таблицу определённым образом, можно составлять расписания вручную, но этот подход имеет целый ряд недостатков:

1) невозможность работы с большим количеством пациентов;

2) подверженность человеческим ошибкам;

3) низкая скорость работы;

4) невозможность одновременной работы на разных компьютерах.

1.3.2 МИС Medesk

Одной из возможностей данной информационной системы является управление расписанием приёмов. Расписание представлено в виде таблицы, столбцы которой – сотрудники поликлиники, а по вертикальной оси отложена шкала времени (рисунок 1.1). В таблице в виде прямоугольников отображены существующие записи пациентов.

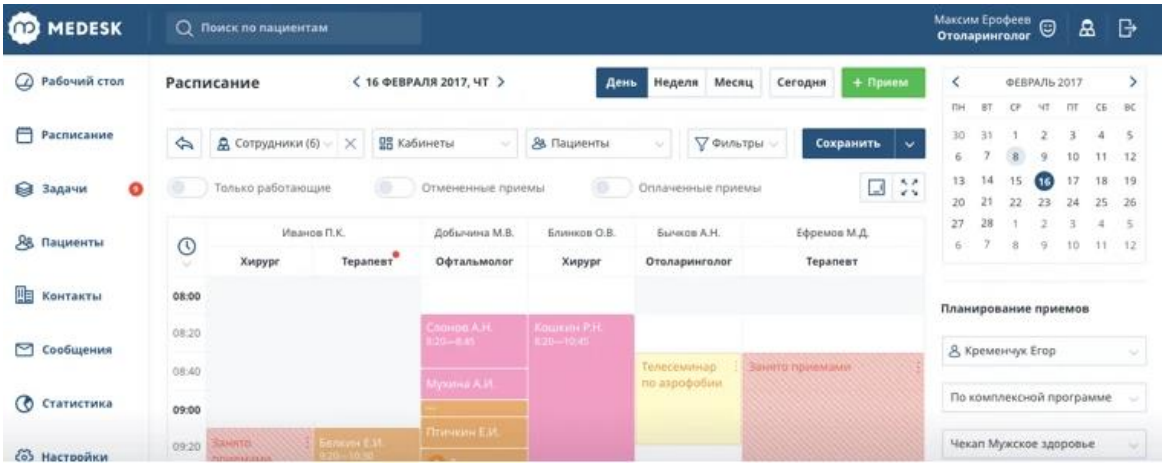

Рисунок 1.1 – Расписание в МИС Medesk

К плюсам этой системы можно отнести:

1) интеграция с другими модулями информационной системы;

2) возможность совместной работы с расписанием;

3) невозможность пересечения посещений.

Недостатками являются:

1) зависимость от МИС Medesk;

2) отсутствие функции автоматической оптимизации расписания;

3) остальные методы ручного составления расписаний.

1.3.3 МИС Медиалог

Данная медицинская информационная система также позволяет настраивать расписания. В отличии от предыдущей системы данная позволяет указывать время отдыха для врача (рисунок 1.2), но работает лишь с фиксированными интервалами времени.

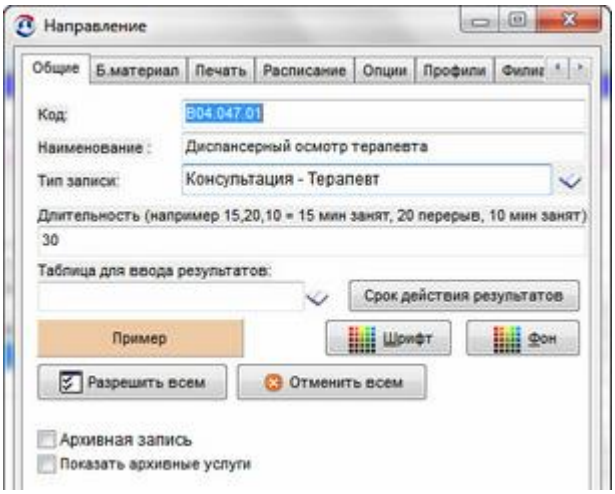

Рисунок 1.2 – Настройка осмотра в МИС Медиалог

Добавление пациентов в сетку расписания производится в ручном режиме (рисунок 1.3).

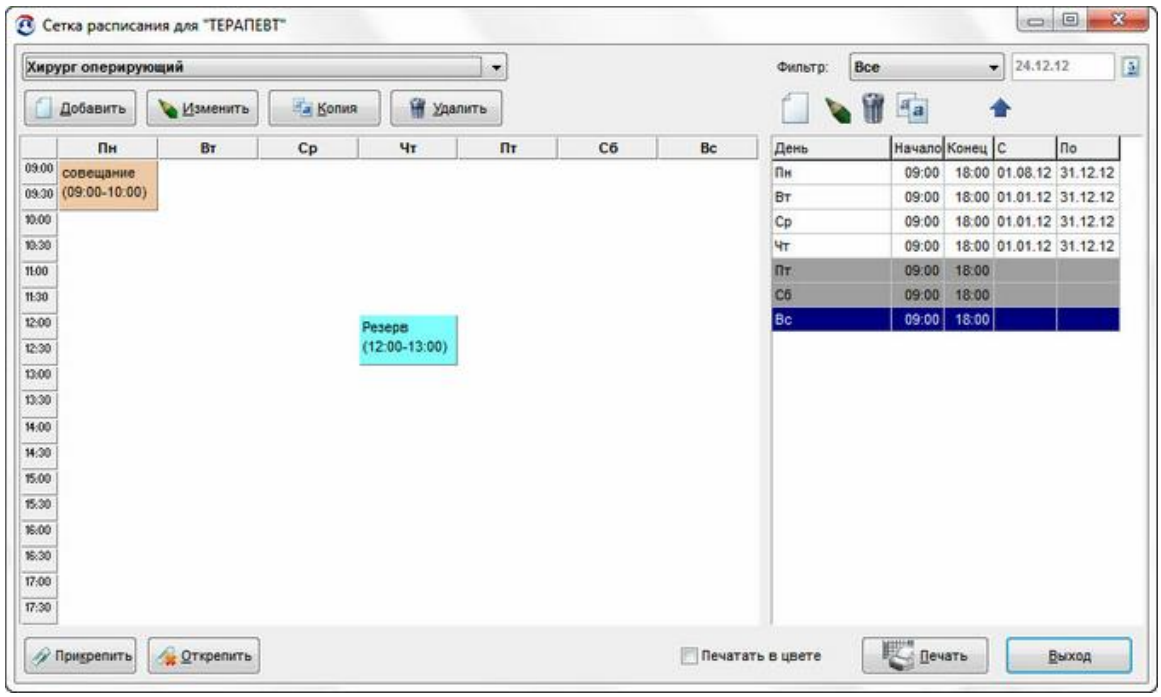

Рисунок 1.3 – Сетка расписания в МИС Медиалог

1.3.4 1С: Автоматизированное составление расписания. Университет

Данная программа не относится к медицинским информационным системам, однако предназначена для построения расписаний и позволяет их оптимизировать. На рисунке 1.4 представлен внешний вид сгенерированного расписания.

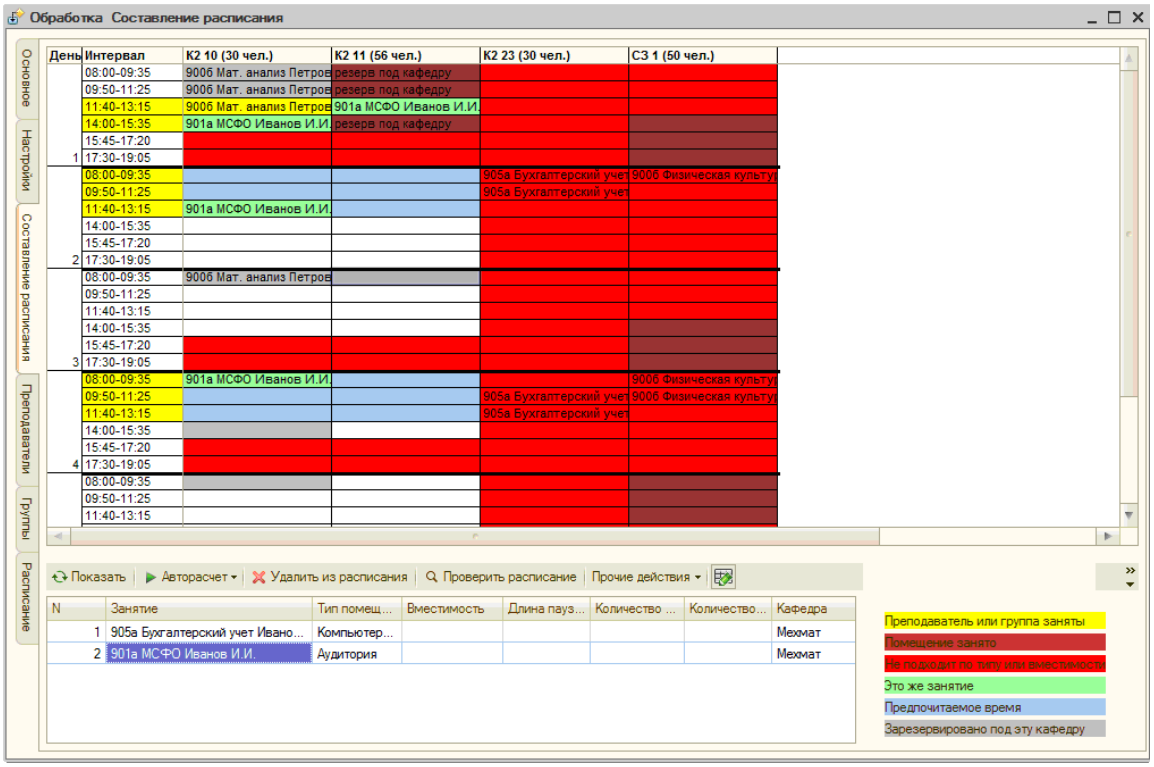

Рисунок 1.4 – Расписание в программе 1С

В задаче оптимизации потока пациентов это решение имеет существенный недостаток – оно не учитывает медицинской специфики. Также минусом является работа с фиксированными отрезками времени.

1.3.5 АВТОРасписание (AVTOR)

На рисунке 1.5 отображён процесс составления расписания в программе AVTOR. Данное программное решение также приспособлено для составления учебных расписаний и обладает функцией оптимизации расписаний. Кроме недостатков предыдущего решения, у этого добавляется невозможность интеграции с существующей медицинской информационной системой.

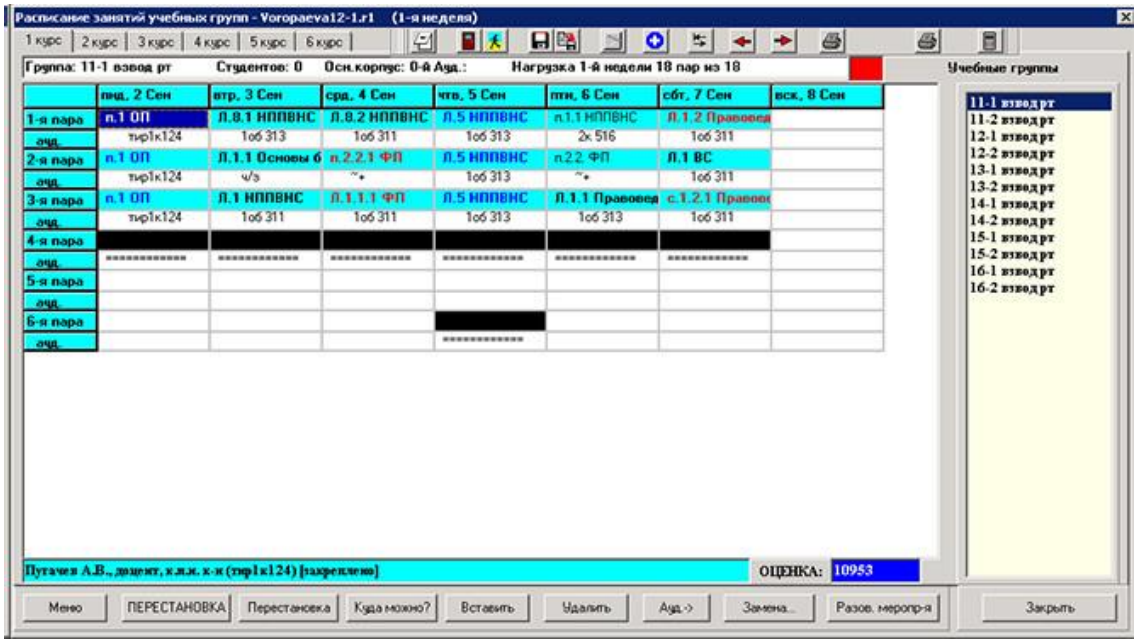

Рисунок 1.5 – Расписание в программе AVTOR

1.4 Теория расписаний

Теория расписаний – раздел дискретной математики, рассматривающий проблемы упорядочения. В общем случае эти проблемы формулируются так: имеется некоторое множество работ (требований) $J = \{J_1, J_2, \dots, J_n\}$ для которых заданы характеристики: длительность обработки, стоимость, момент поступления, срок окончания обслуживания. Задаётся множество машин (приборов) $M = \{M_1, M_2, \dots, M_m\}$, на которых требования должны обслуживаться в соответствии с некоторым порядком [2, 12–13].

Ставится задача дискретной оптимизации: построить расписание, минимизирующее время выполнения работ, стоимость работ и т. п.

Расписание — указание, на каких машинах и в какое время должны обслуживаться требования (выполняться работы).

1.5 Визуализация расписаний

Наиболее наглядным представлением решений задач составления расписаний является диаграмма Гантта. На рисунке 1.6 приведён пример диаграммы для задачи, состоящий из 3 машин и 4 работ. Диаграмма может

отражать порядок выполнения работ на каждой машине (слева) или порядок машин, на которых выполнялась каждая работа (справа). Благодаря такому представлению легко удостовериться, что в каждый момент времени каждая машина занята лишь одной работой и что никакие две операции, принадлежащие одной работе, не выполняются одновременно.

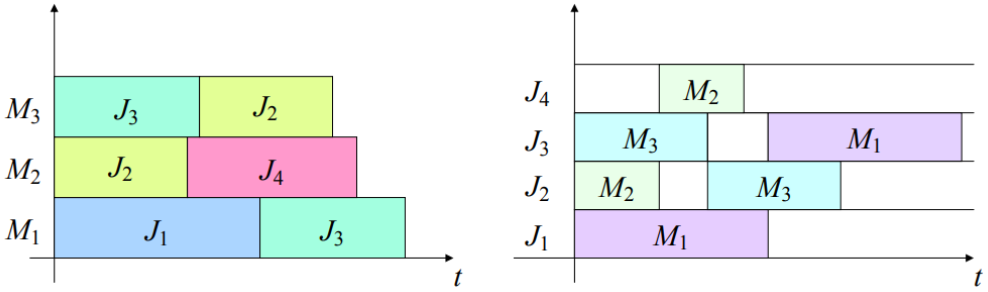

Рисунок 1.6 – Решение задачи для 3 машин и 4 работ

1.6 Классы задач теории расписаний

1.6.1 Класс задач Job Shop

В задачах Job Shop рассматривается множество работ, выполняемых на нескольких машинах. Каждая работа состоит из последовательности операций, которые должны быть выполнены в указанном порядке. Кроме того, каждая операция имеет своё время выполнения и должна обрабатываться на определённой машине. Важными ограничениями этого класса задач являются:

1) несколько операций, относящихся к одной работе, не могут выполняться одновременно;

2) машина может обрабатывать только одну операцию за раз;

3) однажды начатая операция должна быть закончена и не может быть прервана.

Цель задачи Job Shop – минимизировать общее время выполнения: разницу между началом выполнения первой операции и окончанием выполнения последней [18, 20]. Кроме обязательных требований могут быть добавлены дополнительные условия:

1) машинам может требоваться некоторое время между выполнением операций, или, наоборот, отсутствие простоя;

2) между операциями могут быть заданы отношения предшествования;

3) может требоваться оптимизация не только суммарного времени выполнения, но и других критериев;

4) на обработку операции может требоваться фиксированное или вероятностное.

1.6.2 Класс задач Open Shop

Задачи класса Open Shop характеризуются отсутствием заданного порядка выполнения работ. Так же для работ не задано отношение предшествования. Распределение работ может выполняться произвольным образом. В результате решения этой проблемы находят последовательность машин, на которых выполнялась каждая конкретная работа, и, наоборот, последовательность работ, выполненных на каждой конкретной машине.

1.6.3 Класс задач Flow Shop

Для задач Flow Shop изначально задаётся последовательность машин и набор работ, которые должны быть выполнены на этих машинах в строго определённом порядке [4]. Для каждой работы указывается машина, на которой она должна быть выполнена. Целью решения задач данного класса является минимизация времени простоя машин и общего времени обслуживания.

1.7 Методы решения задачи Job Shop

1.7.1 Генетические алгоритмы

Генетические алгоритмы – это семейство вычислительных алгоритмов, основанных на принципах эволюции. Их использование является основным подходом к решению задач Job Shop [10, 15–16]. Особями выступают кандидаты на оптимальное решение задачи. На каждой итерации алгоритма существует поколение, состоящее из множества особей. Функция,

показывающая насколько данное решение близко к оптимальному, называется функцией приспособленности. Алгоритм заключается в создании новых особей на основе текущего поколения применяя к ним операции селекции, кроссинговера и мутации, что позволяет множеству особей эволюционировать от поколения к поколению. Так же к преимуществам использования генетических алгоритмов можно отнести то, что каждая особь является корректным решением задачи Job Shop [25].

1.7.2 Метод удовлетворения ограничений (CSP)

В случае, если для задачи введено ограничение на максимальное время окончание работы, то для решения задачи Job Shop возможно применение алгоритмов решения задач удовлетворения условий [8, 23–24].

1.7.3 Метод применения правил (Dispatching rules)

Данный метод является одним из самых простых методов оптимизации расписаний и широко применяется для решения этого класса задач [7, 11, 14]. Рассмотрим ситуацию, когда машина, которую ожидают несколько задач, освободилась. Метод заключается в установке правил, по которым будет выбрана следующая задача для исполнения [21, 26]. Наиболее известными являются:

1) выбор работы с наибольшим временем ожидания;

2) выбор работы с наименьшим временем выполнения.

Метод применения правил часто используется совместно с генетическими алгоритмами для повышения их эффективности. Сам же метод менее эффективен, чем генетические алгоритмы.

1.7.4 Нейронные сети

Для решения задачи Job Shop применяют следующие типы нейронных сетей:

1) сети Хопфилда;

2) нейронные сети с обратным распространением ошибки.

Решение задачи Job Shop с использованием нейронной сети выглядит следующим образом: на вход подаются свойства операции (время её выполнения, номер операции в работе и т. д.). На выходе нейронная сеть выдаёт приоритет операции [29]. На основе полученных приоритетов с помощью алгоритма Гиффлера-Томпсона строится оптимальное решение [22]. Однако для решения задачи Job Shop применение нейронных сетей менее эффективно, чем использование генетических алгоритмов [6].

1.8 Выбор СУБД и языка программирования

Разрабатываемый модуль оптимизации потока пациентов должен интегрироваться с существующей информационной системой МАУЗ ГКБ №2. Она представляет собой веб-приложение и разработана на языке программирования JavaScript с использованием технологии NodeJS. В качестве СУБД используется PostgreSQL. Таким образом, в качестве языка разработки, целесообразно выбрать JavaScript, а в качестве базы данных – PostgreSQL.

1.9 Выводы по разделу

В данной главе было дано определение и описаны функции медицинской информационной системы. Сформулирована задача оптимизации потока пациентов при проведении медицинских осмотров.

Были рассмотрены существующие программные средства составления расписаний. Часть из них не предоставляет функций для оптимизации расписаний, в других отсутствуют средства интеграции с внешними системами, некоторые содержат функционал для оптимизации расписаний, но не учитывают медицинскую специфику. Практически все программы позволяют работать лишь с фиксированными отрезками времени.

Далее были введены основные определения теории расписаний. Описаны классы задач составления расписаний. Даны краткие описания

некоторых методов их решения. Представлен метод визуализации расписаний в виде диаграммы Гантта.

В качестве метода решения был сделан выбор в пользу генетических алгоритмов. В отличии от нейронных сетей им не требуется обучающая выборка, но по сравнению с другими рассмотренными методами генетические алгоритмы обеспечивают лучшие результаты.

Для создания программной реализации алгоритма был выбран язык программирования JavaScript, а также интегрированная среда разработки WebStorm.

2 ОПТИМИЗАЦИЯ ПОТОКА ПАЦИЕНТОВ С ПОМОЩЬЮ ГЕНЕТИЧЕСКИХ АЛГОРИТМОВ

2.1 Генетические алгоритмы

2.1.1 Механизм работы генетических алгоритмов

Перед применением генетического алгоритма необходимо закодировать задачу в соответствующих терминах. В большинстве комбинаторных задач, решаемых при помощи генетических алгоритмов, особи представляют в виде двоичных векторов. Однако, в случае решения задачи Job Shop этого недостаточно [19]. В задачах составления расписаний особи являются векторами пар, состоящих из номера операции и номера машины, на которой эта операция выполняется.

Следующим этапом применения алгоритма является создание начальной популяции. Она генерируется случайным образом или на основе каких-либо эвристических данных.

2.1.2 Эволюционный процесс

В процессе эволюции для каждой особи определяется величина приспособленности. Чем больше это значение – тем ближе особь к оптимальному решению. Однако, в случае решения задачи Job Shop, наоборот, необходимо минимизировать результат [28]. Поэтому целевая функция преобразуется в следующий вид:

$$Fit(i) = \begin{cases} \bar{F} - F_i(S_i), \text{если } F_i(S_i) < \bar{F}, \\ 0, \text{иначе}, \end{cases}$$

где $F_i(S_i)$ – длительность выполнения всех операций для расписания S_i, полученное на основе особи i, а $\bar{F}$ – длительность выполнения всех операций для некого эвристического решения. Так как значение целевой функции обычно положительно, то в качестве целевой можно рассматривать следующую функцию:

$$Fit(i) = \frac{1}{F_i(S_i)}.$$

2.1.3 Отбор родителей для следующего поколения

В качестве родителей для следующего поколения необходимо выбрать наиболее подходящие особи. Одним из методов такого отбора является пропорциональная выборка. Вероятность $P(i)$ выбора i-ой особи задаётся формулой:

$$P(i) = \frac{Fit(i)}{\sum_{k=1}^{PS} Fit(k)},$$

таким образом, чем ближе особь к оптимальному решению, тем больше шанс, что она станет родителем следующего поколения.

Альтернативным способом отбора является состязание. В качестве кандидатов родителей для следующего поколения случайным образом выбираются несколько особей, а особи с наибольшим значением приспособленности становятся ими.

2.1.4 Кроссинговер

Кроссинговер является основным оператором генетического алгоритма. В результате его применения гены отобранных особей перемешиваются. Параметр P_C характеризует вероятность кроссинговера. Рекомендуется принимать этот параметр $P_C \geq 0.6$ [17, 27].

Существует два вида кроссинговера: n-точечный и однородный. Первый заключается в установке точек кроссинговера внутри хромосом и дальнейшем обмене участками генов между родителями. Во втором случае каждый ген ребёнка выбирается случайным образом из соответствующих генов родителей.

2.1.5 Мутация

Оператор мутации применяется в соответствии с параметром P_M – вероятностью мутации. Мутация независимо применяется к особям, полученным после кроссинговера. Её суть заключается в небольшом случайном изменении генов потомков для внесения разнообразия в новую популяцию.

2.1.6 Формирование нового поколения

В качестве нового поколения выбираются особи, полученные в результате мутации. Одной из модификаций этого процесса, является составление нового поколения из особей текущего поколения, с заменой половины наименее приспособленных родителей наиболее приспособленными потомками [30].

2.1.7 Критерий останова

Работа алгоритма может быть остановлена, если будет выполнено одно из условий:

1) найдено оптимальное решение;

2) достигнут предел количества поколений;

3) за определённое количество поколений результат не улучшился.

2.2 Поток пациентов в терминах генетических алгоритмов

Определимся с терминами, используемыми в процессе работы. В качестве машин будут выступать кабинеты врачей-специалистов или кабинеты, в которых проводятся обследования. Пациент является работой, а врачи и обследования, которые ему необходимо пройти – операциями.

Известно в каких кабинетах оказываются конкретные услуги, а время их оказания определяется на основе статистических данных.

Для каждого расписания будем считать общее время выполнения. Оно выражается как разность между временем окончания выполнения последней операции и временем начала выполнения первой операции. В качестве целевой выберем следующую функцию:

$$Fit(i) = \frac{1}{F_i(S_i)},$$

где $F_i(S_i)$ – длительность выполнения всех операций для расписания S_i, полученное на основе особи i.

В качестве методов отбора в данной работе будут рассмотрены пропорциональная и выборка наиболее приспособленных особей. При

пропорциональной выборке вероятность того, что особь станет родителем для следующего поколения определяется по формуле:

$$P(i) = \frac{Fit(i)}{\sum_{k=1}^{PS} Fit(k)}.$$

Рассмотрим несколько вариантов оператора кроссинговера. В случае одноточечного кроссинговера родительские хромосомы разделяются в одной случайной точке, а затем обмениваются получившимися частями (рисунок 2.1). В итоге такого скрещивания образуется две особи, каждая из которых частично состоит из генов родителей.

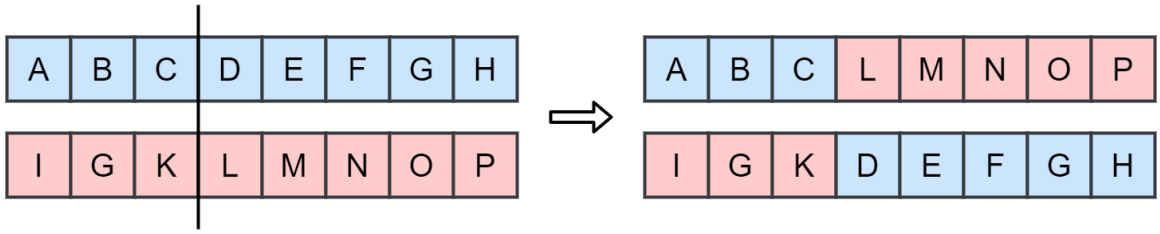

Рисунок 2.1 – Одноточечный кроссинговер

Для создания большего разнообразия особей в работе рассматривается двухточечный кроссинговер. В нём хромосомы родителей разделяются двумя случайными точками, а затем происходит обмен генами (рисунок 2.2).

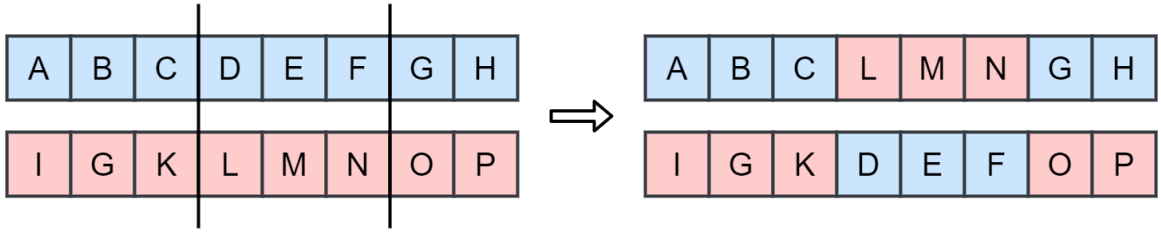

Рисунок 2.2 – Двухточечный кроссинговер

В случае однородного кроссинговера хромосомы родителей не делятся на участки, вместо этого каждый ген рассматривается отдельно (рисунок 2.3). Выбор гена матери или отца происходит случайным образом. В результате такого кроссинговера достигается наибольшее разнообразие особей.

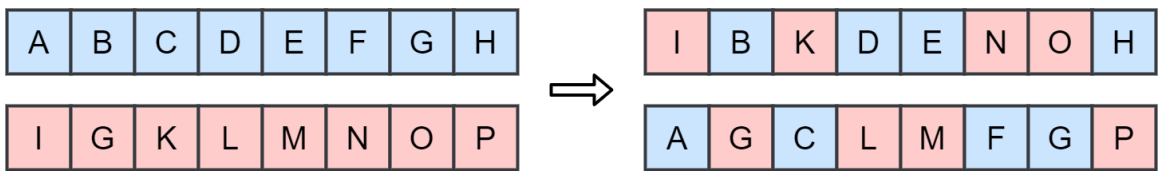

Рисунок 2.3 – Однородный кроссинговер

В работе рассматривается несколько методов мутации. Мутация – это процесс случайного изменения конкретной особи. Она является вторым фактором, вносящим изменения в геном популяции. Простейший случай описывается как изменение у особи единственного гена случайным образом (рисунок 2.4). В терминах расписания это означает, что одна, случайно выбранная операция, будет перенесена на другую, случайно выбранную машину, на которой она может быть выполнена.

Рисунок 2.4 – Изменение одиночного гена

Также будет применён метод мутации, заключающийся в перестановке двух генов цепочки или, иначе говоря, перестановке операций, выполняемых на одной машине (рисунок 2.5).

Рисунок 2.5 – Перестановка генов

Мутация инверсией меняет порядок выполнения операций на обратный (рисунок 2.6). Начало и конец интервала, на которых будет произведена мутация, выбираются случайным образом, затем происходит разворот списка.

Рисунок 2.6 – Мутация инверсии

Мутация циклическим сдвигом похожа на мутацию инверсии. Разница заключается в том, что вместо разворота подсписка выполняется его циклический сдвиг влево на одну позицию (рисунок 2.7). Таким образом первый элемент подсписка становится последним.

Рисунок 2.7 – Мутация циклическим сдвигом

2.3 Выводы по разделу

В данном разделе были описаны основные шаги работы генетических алгоритмов. Рассмотрены используемые варианты критериев останова. Введены определения селекции, кроссинговера и мутации. Приведены разновидности операторов, используемых в данной работе.

Была сформулирована задача оптимизации потока пациентов в терминах генетических алгоритмов. Задана целевая функция.

3 ПРОГРАММНАЯ РЕАЛИЗАЦИЯ МОДУЛЯ ОПТИМИЗАЦИИ ПОТОКА ПАЦИЕНТОВ

3.1 Разработка системы и начальные данные

Разрабатываемый модуль оптимизации потока пациентов должен интегрироваться с существующей информационной системой МАУЗ ГКБ №2. Эта информационная система представляет собой монолитное веб-приложение. В приложении выделены модули проведения медицинских осмотров для организаций и модуль статистического анализа потока пациентов.

Модуль проведения медицинских осмотров обеспечивает создание медосмотров для организаций и добавление в них пациентов, генерацию списков врачей и обследований, необходимых для прохождения медосмотра каждому конкретному пациенту. Этот модуль является поставщиком данных о пациентах и предоставляемых им услугах.

Модуль статистического анализа потока пациентов призван собирать и анализировать время, затрачиваемое пациентами на прохождение медицинского осмотра. Взаимодействие с этим модулем позволяет прогнозировать временные затраты для вновь прибывших пациентов.

При создании медицинского осмотра происходит загрузка списка сотрудников предприятия из файла Excel. Далее специализированный модуль по шифрам вредности каждого сотрудника определяет какие виды осмотров и обследований должны быть включены в план прохождения медосмотра. После этого модуль медосмотров запускает работу модуля оптимизации в фоновом режиме. Модуль оптимизации запрашивает у модуля статистики данные о предполагаемой продолжительности осмотров и обследований для каждой услуги каждого из пациентов с учётом их возраста и пола. На основе этих данных происходит построение оптимального расписания.

3.2 Схема базы данных

3.2.1 Существующая база данных

Для начала работы алгоритму необходимо будет получить информацию о пациентах и планах их медосмотров из базы данных. Рассмотрим часть таблиц существующей схемы базы данных (рисунок 3.1), которые будут использоваться модулем оптимизации.

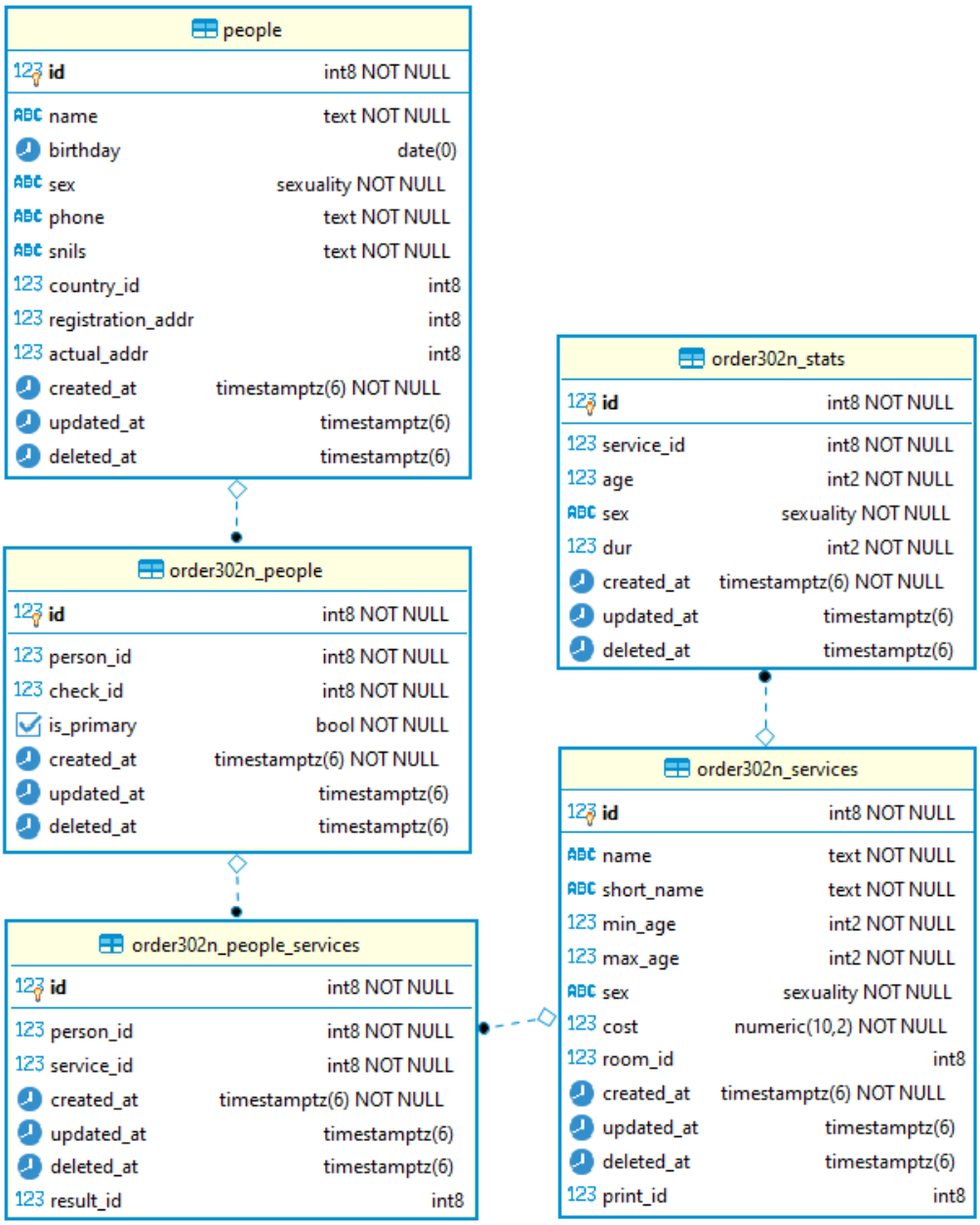

Рисунок 3.1 – Существующая база данных

Опишем используемые модулем таблицы. Таблица «people» хранит персональные данные пациентов и сотрудников (таблица 3.1). Она будет использоваться модулем для получения ФИО, даты рождения и пола пациента.

Таблица 3.1 – Описание таблицы «people»

Наименование	Тип	Комментарий
id	int8	Идентификатор
name	text	ФИО
birthday	date	Дата рождения
sex	sexuality	Пол
phone	text	Номер телефона
snils	text	Номер СНИЛС
country_id	int8	Идентификатор страны
registration_addr	int8	Идентификатор адреса регистрации
actual_addr	int8	Идентификатор фактического адреса
created_at	timestamptz	Дата и время создания
updated_at	timestamptz	Дата и время обновления
deleted_at	timestamptz	Дата и время удаления

Таблица «order302_people» описывает связь между конкретным медосмотром и пациентом, и содержит информацию о том, является ли этот медосмотр для пациента первичным (таблица 3.2). Эта таблица используется модулем для выделения пациентов из всего множества людей в базе.

Таблица 3.2 – Описание таблицы «order302_people»

Наименование	Тип	Комментарий
id	int8	Идентификатор
person_id	int8	Идентификатор человека
check_id	int8	Идентификатор медосмотра
is_primary	bool	Является ли медосмотр первичным
created_at	timestamptz	Дата и время создания
updated_at	timestamptz	Дата и время обновления
deleted_at	timestamptz	Дата и время удаления

Таблица «order302_people_services» связывает пациента с услугами, которые ему необходимо пройти (таблица 3.3). Кроме того, эта таблица хранит идентификатор результата осмотра или обследования. Из этой таблицы модуль получает список услуг для каждого пациента.

Таблица 3.3 – Описание таблицы «order302_people_services»

Наименование	Тип	Комментарий
id	int8	Идентификатор
person_id	int8	Идентификатор человека
service_id	int8	Идентификатор услуги
result_id	int8	Идентификатор результата осмотра
created_at	timestamptz	Дата и время создания
updated_at	timestamptz	Дата и время обновления
deleted_at	timestamptz	Дата и время удаления

Таблица «order302_services» хранит список всех услуг. Она содержит данные о названии, фильтрах, применяемых при добавлении услуг в план медосмотра пациента, стоимости услуги, идентификаторы печатной формы и кабинета. Разрабатываемая программа получает из этой таблицы названия и идентификаторы услуг.

Таблица 3.4 – Описание таблицы «order302_services»

Наименование	Тип	Комментарий
id	int8	Идентификатор
name	text	Название услуги
short_name	text	Краткое название услуги
min_age	int2	Минимальный возраст
max_age	int2	Максимальный возраст
sex	sexuality	Пол
cost	numeric(10,2)	Стоимость
room_id	int8	Идентификатор кабинета

Продолжение таблицы 3.4

Наименование	Тип	Комментарий
print_id	int8	Идентификатор печатной формы
created_at	timestamptz	Дата и время создания
updated_at	timestamptz	Дата и время обновления
deleted_at	timestamptz	Дата и время удаления

Таблица «order302_stats» содержит статистические данные о времени прохождения осмотра или обследования человеком с указанными полом и возрастом. Данные этой таблицы будут основой работы модуля оптимизации.

Таблица 3.5 – Описание таблицы «order302_stats»

Наименование	Тип	Комментарий
id	int8	Идентификатор
service_id	int8	Идентификатор услуги
age	int2	Возраст
sex	sexuality	Пол
dur	int2	Длительность в секундах
created_at	timestamptz	Дата и время создания
updated_at	timestamptz	Дата и время обновления
deleted_at	timestamptz	Дата и время удаления

3.2.2 Проектирование базы данных

После окончания работы модуля необходимо сохранить полученные результаты. Так как работа модуля заключается в упорядочивании услуг в перечне каждого пациента, то разумно использовать структуру данных, похожую на список.

Для реализации списка в реляционной базе данных необходимо создать поле, содержащее идентификатор услуги, идущей в перечне пациента перед текущей. Значение null будет указывать на то, что эта услуга в списке является первой. Так как за план медицинского осмотра пациента отвечает таблица

«order302_people_services», то добавим новое поле в неё. Таблица примет вид, изображённый на рисунке 3.2.

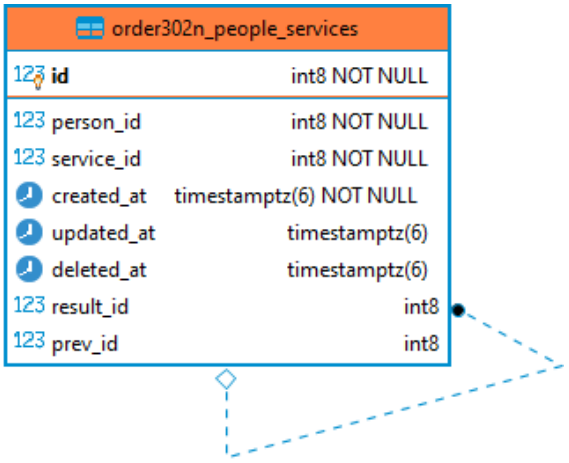

Рисунок 3.2 – Реализация списка услуг в базе данных

3.3 Схема алгоритма

Для разработки модуля был реализован алгоритм, представленный на рисунке 3.3. Этот алгоритм описывает весь функционал модуля оптимизации пациентов. Операторы селекции, кроссинговера и мутации выбираются в соответствии с переданными параметрами при запуске модуля (рисунки 3.4, 3.5 и 3.6).

В качестве критерия останова выбрано предельное количество поколений. Экспериментальным путём было выявлено, что 10000 поколений достаточно для сокращения расписания более, чем в 2 раза.

Работа модуля осуществляется в 4 шага:

1) получение данных о пациентах, планах их медосмотров;

2) получение данных о предполагаемой продолжительности оказания каждому пациенту соответствующих услуг;

3) работа генетического алгоритма по составлению и оптимизации расписания;

4) фиксация в базе данных результатов работы модуля в соответствующей таблице.

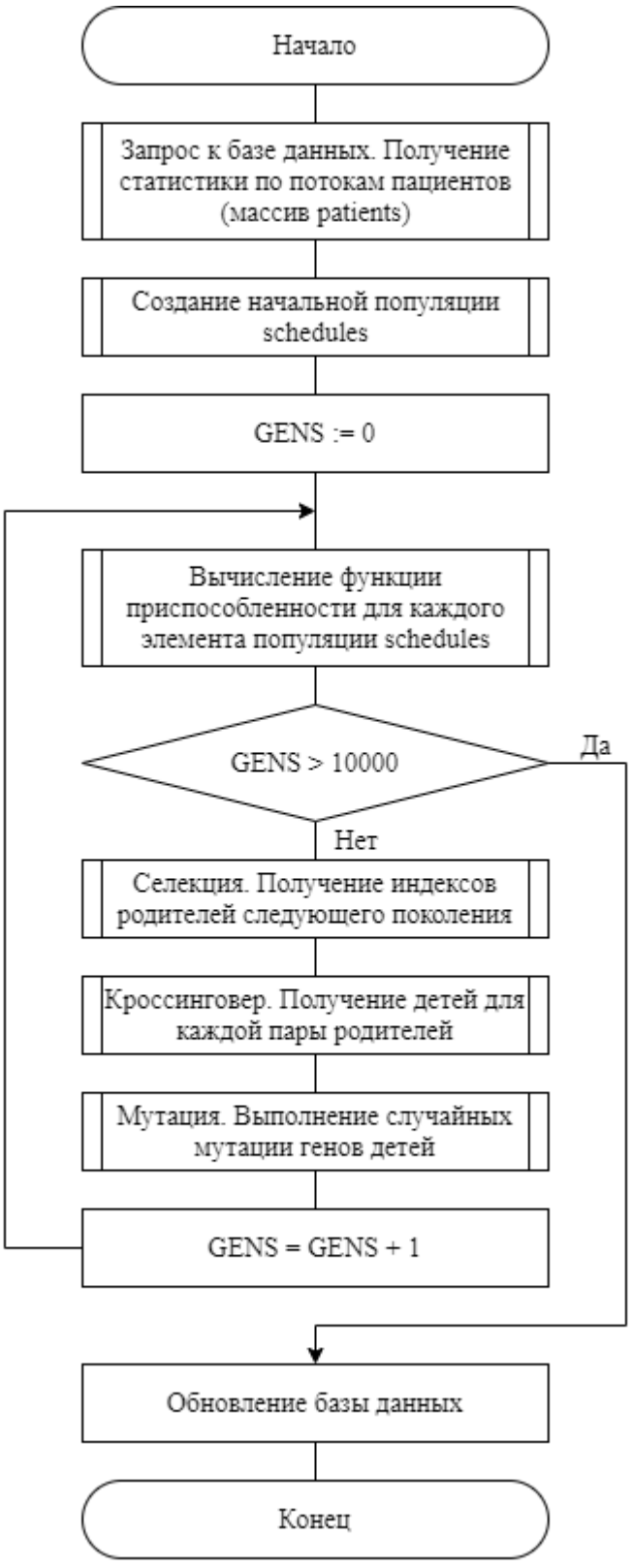

Рисунок 3.3 – Общий алгоритм работы реализуемого модуля

Рассмотрим алгоритм состязательной селекции, представленный на рисунке 3.6. Его суть состоит в случайном выборе нескольких особей текущего поколения и последующем выборе двух наиболее приспособленных.

В данной работе в качестве кандидатов на роль родителей рассматриваются группы из трёх особей.

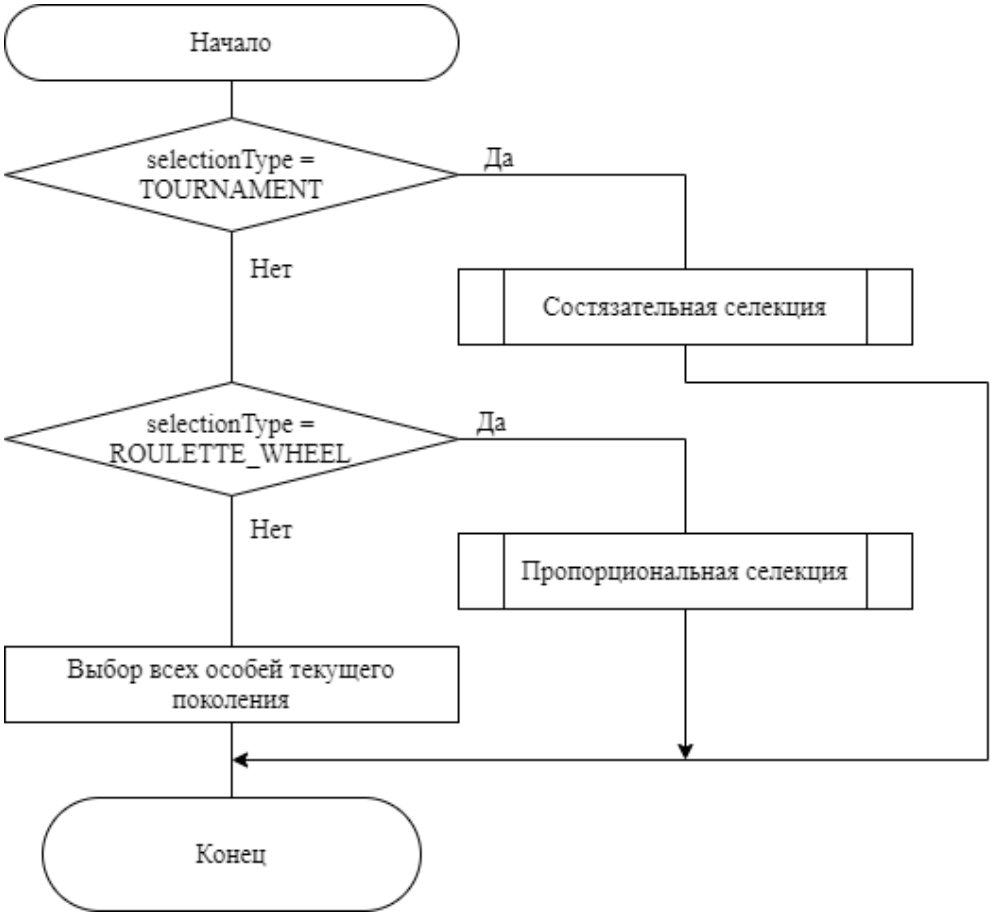

Рисунок 3.4 – Выбор алгоритма селекции

На рисунке 3.7 представлен алгоритм однородного кроссинговера. Процесс кроссинговера определяет какие гены и от какого родителя приобретёт новая особь. В данном случае ребёнок сначала полностью копирует гены матери, а затем, проходя по каждому гену в отдельности, с вероятностью 0.5 перенимает соответствующий ген отца.

Мутация – это процесс, направленный на внесение дополнительного разнообразия популяции, независимо от генов родителей. В данной работе шанс мутации особи равен 0.05. На рисунке 3.8 изображён алгоритм мутации циклическим сдвигом. Первым шагом определяется отрезок генома, на котором будет происходить мутация. Точки начала и конца отрезка выбираются случайным образом. Далее все гены смещаются на одну позицию влево, а первый ген становится последним.

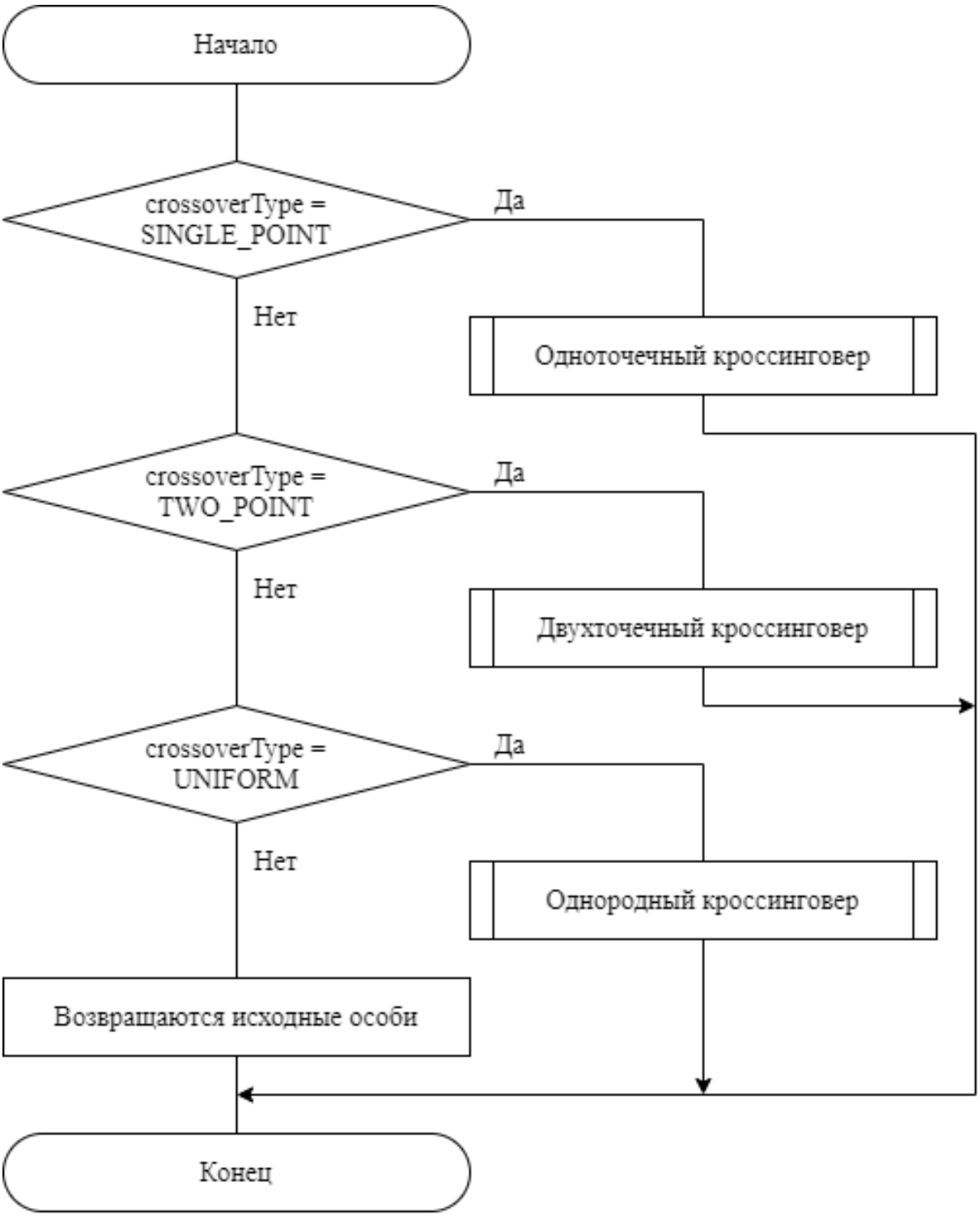

Рисунок 3.5 – Выбор алгоритма кроссинговера

В приложении 1 представлена реализация основного цикла генетического алгоритма и реализации других операторов селекции, кроссинговера и мутации, используемых в данной работе.

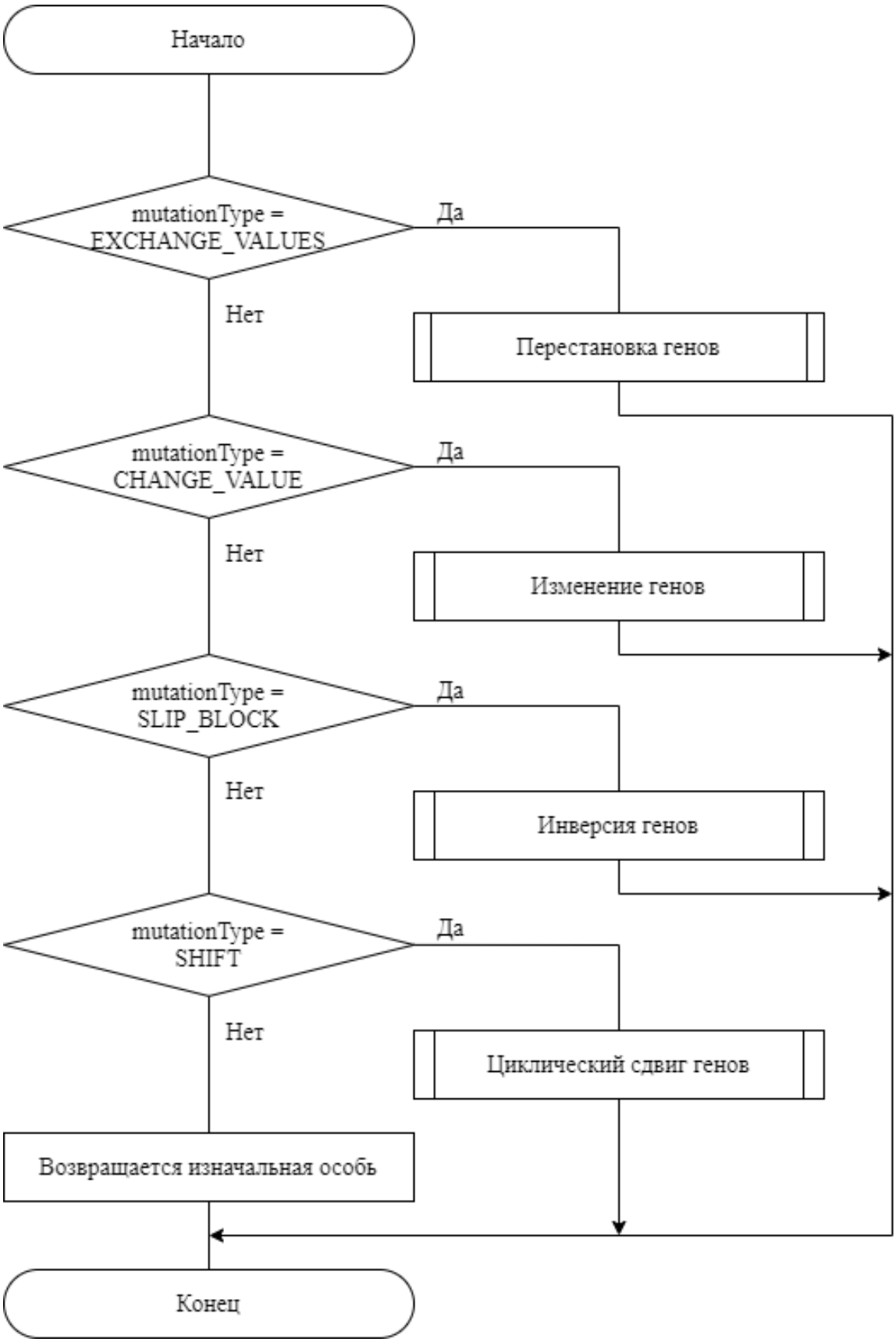

Рисунок 3.6 – Выбор алгоритма мутации

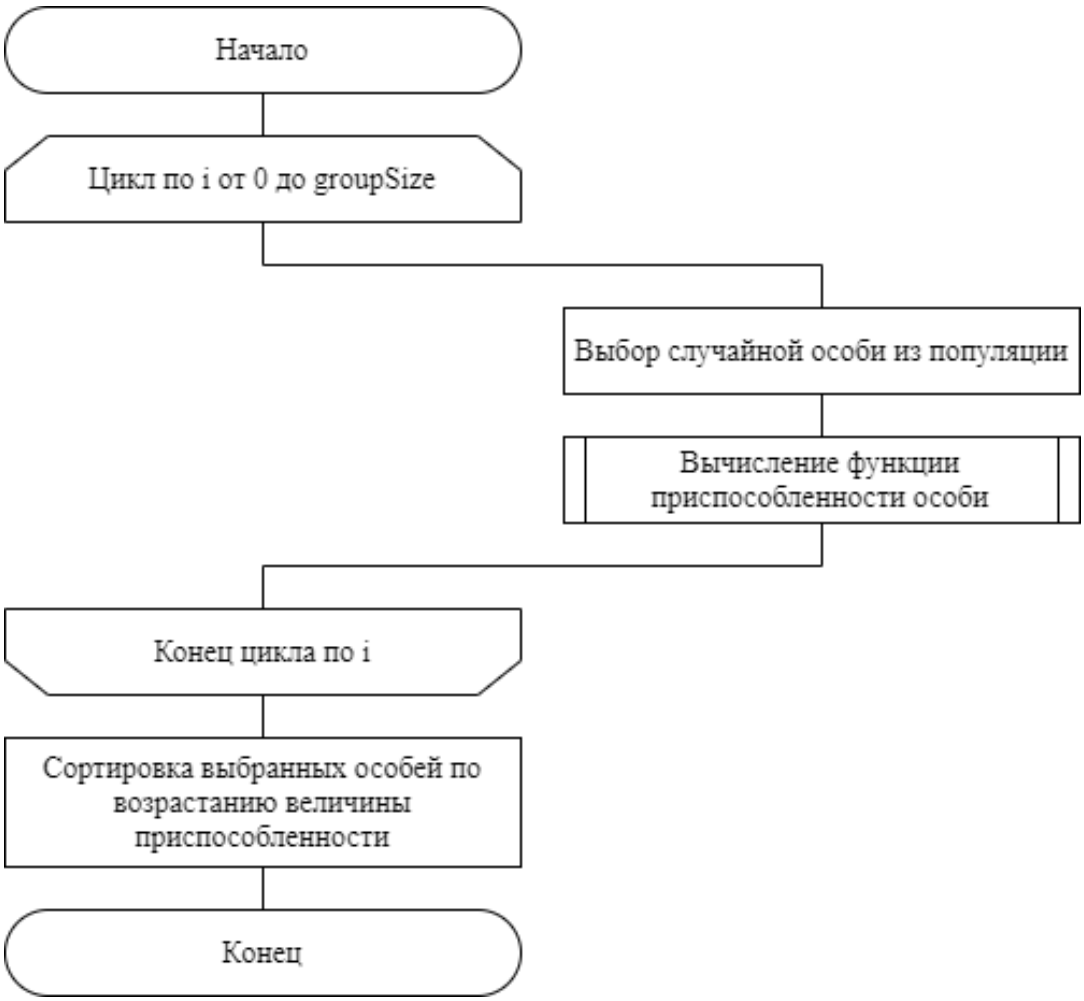

Рисунок 3.6 – Состязательная селекция

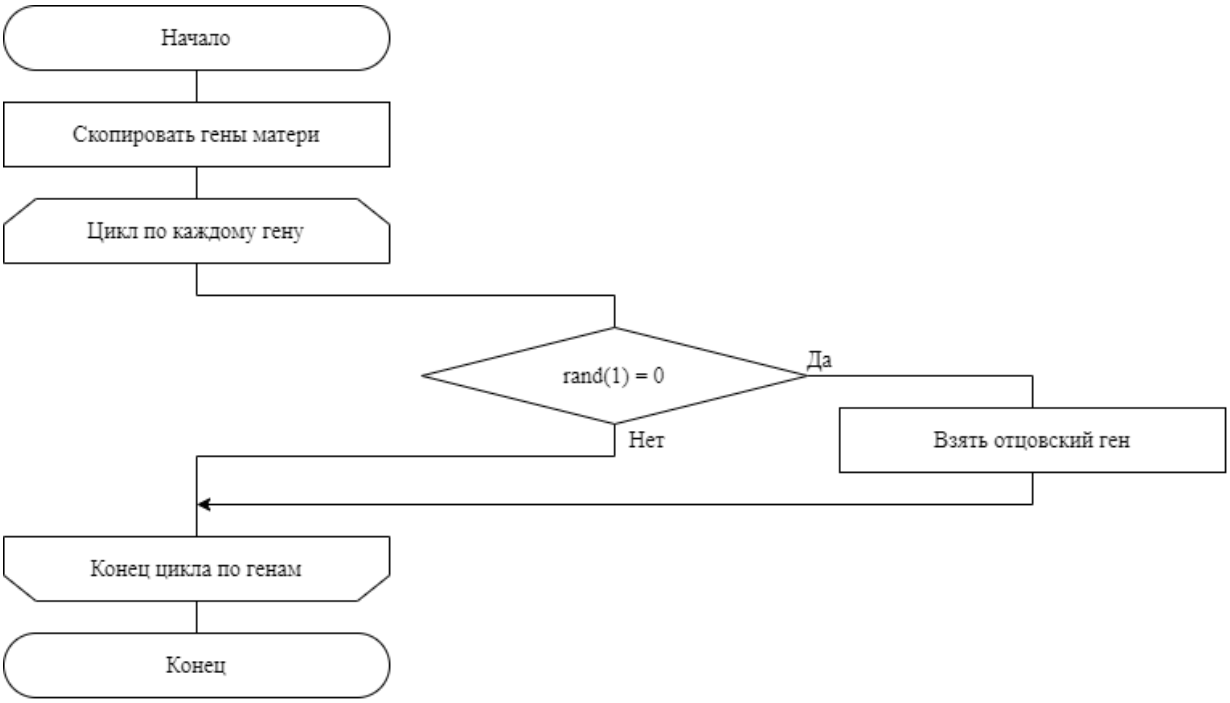

Рисунок 3.7 – Однородный кроссинговер

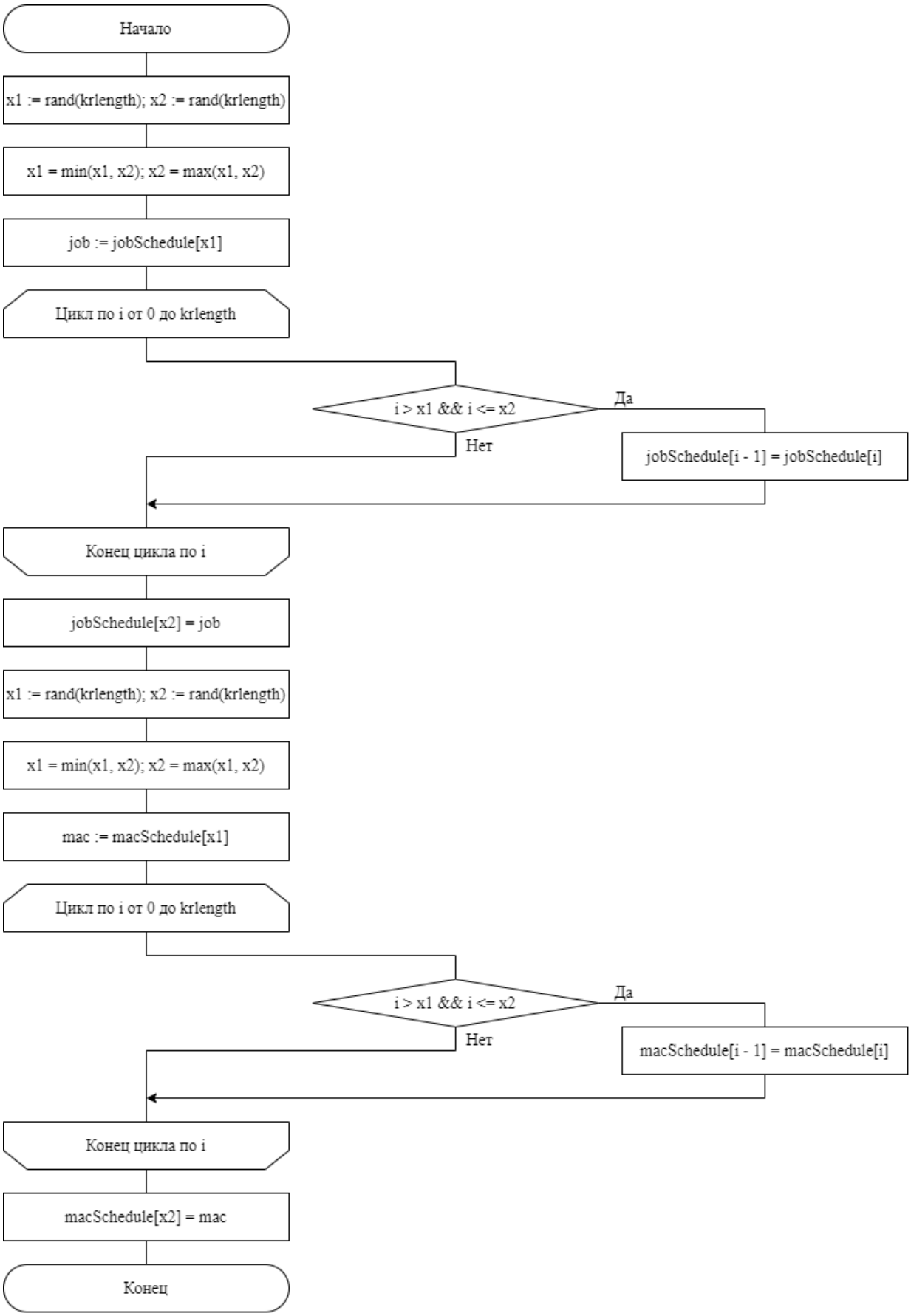

Рисунок 3.8 – Мутация циклическим сдвигом

3.3 Проверка работы алгоритма на экспериментальных данных

Опишем эксперименты, которые были проведены для выявления наиболее оптимальной конфигурации алгоритма.

Применим алгоритм к существующему восьмичасовому расписанию и сравним результаты работы алгоритма, составленного из различных операторов. В качестве оцениваемых параметров примем: наименьшую длину расписания, которую смог обеспечить алгоритм и время работы алгоритма к моменту, когда этот результат был достигнут.

Во всех тестах используется следующая начальная конфигурация:

$PS = 500$ – размер популяции;

$GENS = 10000$ – предельное количество поколений;

$RUNS = 100$ – количество запусков теста.

Была проведена серия тестов с использованием различных операторов селекции: состязательного и пропорционального. Далее представлены результаты тестирования алгоритма с применением состязательной селекции.

В таблице 3.1 представлены размеры наименьших сгенерированных расписаний, выраженные в секундах. Из всех запусков тестов был выбран лучший результат.

Таблица 3.1 – Наименьший размер расписания (состязательная селекция)

	Одноточечный	Двухточечный	Однородный
Перестановка	25080	24690	13130
Изменение	25150	24360	13160
Инверсия	25050	24020	13110
Циклический сдвиг	24110	23390	13140

Наилучшие результаты были получены с использованием мутации инверсией и однородного кроссинговера.

В таблице 3.2 представлено среднее время работы алгоритма в секундах до момента, когда был получен наилучший результат.

Таблица 3.2 – Среднее время работы алгоритма (состязательная селекция)

	Одноточечный	Двухточечный	Однородный
Перестановка	3	12	266
Изменение	96	11	276
Инверсия	75	7	166
Циклический сдвиг	56	9	256

Наиболее быстрым оказался алгоритм, использующий одноточечный кроссинговер и перестановочный метод мутации. Однако результат, полученный этой комбинацией, оказался в 1,9 раза хуже, чем наименьшее расписание. Комбинации с использованием однородного кроссинговера в среднем показывают большее время работы, чем другие, но в то же время они создают расписания наименьшего размера.

Далее была проведена серия тестов аналогичных конфигураций, но с использованием пропорционального метода селекции. В таблице 3.3 представлены наименьшие размеры сгенерированных расписаний в секундах.

Таблица 3.3 – Наименьший размер расписания (пропорциональная селекция)

	Одноточечный	Двухточечный	Однородный
Перестановка	24610	25940	13100
Изменение	25160	23030	13100
Инверсия	25180	24470	13120
Циклический сдвиг	25330	25560	13100

Самые короткие расписания были получены с использованием однородного кроссинговера. При этом метод мутации незначительно влияет на результат. Самое короткое расписание лишь на 10 секунд меньше аналогичного с применением состязательной селекции.

В таблице 3.4 представлено среднее время работы алгоритма в секундах прошедшее с момента начала работы алгоритма до получения наилучшего результата.

Таблица 3.4 – Среднее время работы алгоритма (пропорциональная селекция)

	Одноточечный	Двухточечный	Однородный
Перестановка	87	63	276
Изменение	56	102	214
Инверсия	16	1	273
Циклический сдвиг	176	8	220

В данном случае наибольшую скорость показала комбинация двухточечного кроссинговера и мутации инверсии, которая, так же, показала один из наихудших результатов размера конечного расписания. Алгоритмы, использующие однородный кроссинговер, снова показали худшие результаты времени работы.

3.4 Вывод по разделу

В данном разделе были представлены алгоритмы и диаграммы, связанные с разработкой модуля оптимизации потока пациентов.

Были проведены эксперименты, демонстрирующие работу алгоритма с различными операторами селекции, кроссинговера и мутации.

В результате экспериментов было выявлено, что использование однородного кроссинговера даёт наиболее компактные варианты расписаний. Так же в результатах тестов было выявлено, что именно оператор кроссинговера влияет на результат наибольшим образом.

В качестве итоговой была выбрана конфигурация: состязательная селекция, однородный кроссинговер и мутация инверсией. С её помощью удалось перестроить расписание длиной 8 часов в 3 часа 38 минут.

ЗАКЛЮЧЕНИЕ

Цель данной работы заключалась в разработке модуля оптимизации потока пациентов МАУЗ ГКБ № 2.

В ходе работы были проанализированы существующие программные решения, была составлена математическая модель, рассмотрены методы оптимизации расписаний.

В качестве метода решения было выбрано решение с использованием генетических алгоритмов. В качестве операторов селекции были рассмотрены: пропорциональный и состязательный. В качестве операторов кроссинговера: одноточечный, двухточечный и однородный. Рассмотрены следующие операторы мутации: перестановка, изменение, инверсия и циклический сдвиг.

Для тестирования рассматриваемых конфигураций применялись данные существующих расписаний поликлиники.

Разработанный модуль реализован на языке программирования JavaScript с использованием библиотеки node-postgres для доступа к базе данных, а также интегрированной среды разработки WebStorm.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Беллман, Р.Э. Динамическое программирование / Р.Э. Беллман – М.: Издательство иностранной литературы, 1960. – 400 с.

2. Бурдюк, В.Я. Теория расписаний. Задачи и методы решений / В.Я. Бурдюк, В. В. Шкурба // Кибернетика. – 1971. – № 1. – С. 89–102.

3. Гэри, М. Вычислительные машины и труднорешаемые задачи / М. Гэри, Д. Джонсон – М.: Мир, 2012. – 420 с.

4. Каширских, К.Н. Улучшенный алгоритм решения двухмашинной задачи flow shop с неодновременным поступлением работ / К.Н. Каширских, К.Н. Поттс, С.В. Севастьянов // Дискретный анализ и исследование операций. – 1997. – Т. 4, № 1. – С. 13–32.

5. Корбут, А.А. Методы ветвей и границ. Обзор теорий, алгоритмов, программ и приложений / А.А. Корбут, И.Х. Сигал, Ю.Ю. Финкельштейн // Operations Forsch. Statist., Ser. Optimiz. – 1977. – V. 8, № 2 – P. 253–280.

6. Корбут, А.А. Гибридные методы в дискретном программировании / А.А. Корбут, И.Х. Сигал, Ю.Ю. Финкельштейн // Известия Академии наук. Техническая кибернетик – 1988. – № 1. – С. 65–77.

7. Корбут, А.А. Приближенные методы дискретного программирования / А.А. Корбут, Ю.Ю. Финкельштейн // Известия Академии наук. Техническая кибернетика – 1983. – № 1. – С. 165–176.

8. Лазарев, А.А. Алгоритмы в теории расписаний, основанные на необходимых условиях оптимальности / А.А. Лазарев // Исследования по прикладной математике. Казань: Издательство Казанского государственного университета – 1984. – Выпуск 10. – С. 102–110.

9. Пападимитриу, Х. Комбинаторная оптимизация. Алгоритмы и сложность. / Х. Пападимитриу, К. Стайглиц – М.: Мир, 1985. – 512 с.

10. Севастьянов, С.В. Геометрические методы и эффективные алгоритмы в теории расписаний / С.В. Севастьянов // Диссертация доктора физ.-мат. наук. – Новосибирск: 2000. – 280 с.

11. Сигал, И.Х. Введение в прикладное дискретное программирование: теория и вычислительные алгоритмы. / И.Х. Сигал, Иванова А.П. – М.: Физматлит, 2002. – 240 с.

12. Танаев, В.С. Введение в теорию расписаний / В.С. Танаев, В.В. Шкурба. – М.: Наука, 1975. – 256 с.

13. Танаев, В.С. Теория расписаний. Многостадийные системы / В.С. Танаев, Ю.Н. Сотсков, В.А. Струсевич – М.: Наука. Гл. ред. физ.-мат. лит., 1989. – 328 с.

14. Финкельштейн, Ю.Ю. Приближенные методы и прикладные задачи дискретного программирования. / Ю.Ю. Финкельштейн – М.: Наука, 1976. – 265 с.

15. Хачатуров, В.Р. Комбинаторные методы и алгоритмы решения задач дискретной оптимизации большой размерности. / В.Р. Хачатуров, В.Е. Веселовский, А.В. Злотов и др. – М.: Наука. – 2000. – 360 с.

16. Шкурба, В.В. Задачи календарного планирования и методы их решения. / В.В. Шкруба, Т.П. Подчасова, А.Н. Пшичук, Л.П. Тур – Киев: Наукова думка, 1966. – 154 с.

17. Alon, N. Approximation schemes for scheduling on parallel machines / N. Alon, G.J. Woeginger, T. Yadid // J. of Scheduling. – 1998. – Vol. 1. – P. 55–66.

18. Applegate, D. A computational study of the job-shop scheduling problem. / D. Applegate, W. Cook // ORSA Journal on Computing – P. 149–156.

19. Bierwirth, C. A generalized permutation approach to job shop scheduling with genetic algorithms / C. Bierwirth // OR Spektrum – P. 87–92.

20. Coffman, E.G. Computer and Job Shop Scheduling Theory / E.G. Coffman // John Wiley & Sons, Hoboken – 1976. – 299 с.

21. Fisher, H. Probabilistic learning combinations of local job-shop scheduling rules / H. Fisher, G.L. Thompson // Industrial Scheduling – 1963. – V. 3, № 2 – P. 225–251.

22. Giffler, B. Algorithms for solving production scheduling problems / B. Giffler, G.L. Thompson // Oper. Res. – 1950 – P. 487–503.

23. Hartmann, S. Experimental evaluation of state-of-art heuristics for the resource-constrained project scheduling problem / S. Hartmann, R. Kolisch // European Journal of Operational Research – 2000. – Vol. 127 – P. 394–407.

24. Kolisch, R. Heuristic Algorithms for Solving the Resource-Constrained Project Scheduling Problem: Classification and Computational Analysis / R. Kolisch, S. Hartmann // Manuskripte aus den Instituten für Betriebswirtschafts-lehre, Kiel, Germany – 1998. – 31 c.

25. Mirshekarian, S. Correlation of job-shop scheduling problem features with scheduling efficiency / S. Mirshekarian, D.N. Šormaz // Expert Systems with Applications – 2016. – Vol. 62 – P. 131–147.

26. Nasiri, M.M. A simulation optimisation approach for real-time scheduling in an open shop environment using a composite dispatching rule / M.M. Nasiri, R. Yazdanparast, and F. Jolai // International Journal of Computer Integrated Manufacturing – 2017. – Vol. 30, № 12 – P. 1239–1252.

27. Park, J. An investigation of ensemble combination schemes for genetic programming based hyper-heuristic approaches to dynamic job shop scheduling / J. Park, Y. Mei, S. Nguyen, G. Chen, and M. Zhang // Applied Soft Computing Journal – 2018. – Vol. 63 – P. 72–86.

28. Sevastianov, S.V. Makespan minimization in open shops: A polynomial time approximation scheme / S.V. Sevastianov, G.J. Woeginger // Mathematical Programming – 1998. – Vol. 82. – P. 191–198.

29. Tang, L. A neural network model and algorithm for the hybrid flow shop scheduling problem in a dynamic environment / L. Tang, W. Liu, J. Liu // Journal of Intelligent Manufacturing – 2005. – Vol. 16, № 3 – P. 361–370.

30. Yeo, M.F. Optimising engineering problems using genetic algorithms / M.F. Yeo, E.O. Agyei // Engineering Computations – Vol. 15, № 2 – P. 268–280.

ПРИЛОЖЕНИЕ 1

Операторы селекции, кроссинговера и мутации

```
// Состязательная селекция
TourSelection() {
    const nums = Array(this.groupSize);

    for (let i = 0; i < nums.length; i++) {
        const p = {i: rndNext(0, this.population.length - 1)};
        p.fitness = this.population[p.i].MakeSpan();
        nums[i] = p;
    }
    return nums.sort((a, b) => a.fitness - b.fitness).map(x => x.i);
}

// Пропорциональная селекция
RouletteWheelSelection() {
    let nums = Array(this.groupSize);
    for (let i = 0; i < nums.length; i++) {
        const p = {i: rndNext(0, this.population.length)};
        p.fitness = this.population[p.i].MakeSpan();
        nums[i] = p;
    }
    nums = nums.sort((a, b) => a.fitness - b.fitness).map(x => x.i);

    const reciprocal = new Float64Array(this.groupSize);
    for (let i = 0; i < reciprocal.length; i++) {
        reciprocal[i] = 1 / this.population[nums[i]].MakeSpan();
    }

    for (let j = 0; j < 2; j++) {
        let value = Math.random();

        for (let i = 0; i < reciprocal.length; i++) {
            value -= reciprocal[i];
            if (value <= 0) {
                nums[j] = i;
                break;
            }
        }
    }
```

```
    return nums;
}

// CROSSOVER

// Однородный кроссинговер
UniformCO(mother, father) {
    const macs = mother.macSchedule.slice(0);
    const jobs = mother.jobSchedule.slice(0);
    for (let i = 0; i < this.krlength; i++) {
        if (rndNext(0, 2) === 0) {
            macs[i] = father.macSchedule[i];
        }
        if (rndNext(0, 2) === 0) {
            jobs[i] = father.jobSchedule[i];
        }
    }
    return new Schedule(jobs, macs, this.jobnum, this.procnum, this.macnum);
}

// Двухточечный кроссинговер
TwoPointCO(mother, father) {
    const N = this.krlength;
    let elNum = rndNext(Math.floor(N / 4), Math.floor(3 * N / 4));
    let x = rndNext(0, N - 1);
    const y = x + elNum - 1;
    let tt = rndNext(0, N - 1);

    const gens = new Int32Array(N);
    for (let i = 0; i < N; i++) {
        gens[i] = -1;
    }

    const used = new Int32Array(this.jobnum);
    for (let i = x; i <= y; i++) {
        gens[i % N] = father.jobSchedule[tt % N];
        used[gens[i % N]]++;
        tt++;
    }

    let k = 0;
    for (let i = 0; i < N; i++) {
```

```
        if (gens[i] !== -1) continue;

        while (used[mother.jobSchedule[k]] >= this.procnum)
            k++;
        gens[i] = mother.jobSchedule[k];
        used[gens[i]]++;
    }

    const macs = mother.macSchedule.slice(0);
    elNum = rndNext(Math.floor(N / 4), Math.floor(N / 2));
    x = rndNext(0, N);
    tt = rndNext(0, N);
    const bound = x + elNum;
    while (x < bound) {
        macs[x % N] = father.macSchedule[tt % N];
        tt++;
        x++;
    }
    return new Schedule(gens, macs, this.jobnum, this.procnum, this.macnum);
}

// Одноточечный кроссинговер
SinglePointCO(mother, father) {
    const N = this.krlength;
    let x = rndNext(Math.floor(N / 4), Math.floor(3 * N / 4));

    const gens = new Int32Array(N);
    for (let i = 0; i < N; i++)
        gens[i] = -1;

    const used = new Int32Array(this.jobnum);
    for (let i = 0; i < x; i++) {
        gens[i] = mother.jobSchedule[i];
        used[gens[i]]++;
    }

    let k = 0;
    for (let i = x; i < N; i++) {
        if (gens[i] !== -1) continue;

        while (used[father.jobSchedule[k]] >= this.procnum) {
            k++;
```

```
        }
        gens[i] = father.jobSchedule[k];
        used[gens[i]]++;
    }

    const macAS = new Int32Array(N);
    x = rndNext(Math.floor(N / 4), Math.floor(3 * N / 4));
    for (let i = 0; i < x; i++) {
        macAS[i] = mother.macSchedule[i];
    }
    for (let i = x; i < N; i++) {
        macAS[i] = father.macSchedule[i];
    }
    return new Schedule(gens, macAS, this.jobnum, this.procnum, this.macnum);
}

// MUTATION

// Мутация изменением
ChangeValueMutation(sch) {
    const x1 = rndNext(0, this.krlength);
    const mac = rndNext(0, this.macnum);
    sch.macSchedule[x1] = mac;
}

// Мутация перестановкой
ExchangeValuesMutation(sch) {
    const N = this.krlength;
    let x1 = rndNext(0, N);
    let x2 = rndNext(0, N);

    const temp = sch.jobSchedule[x1];
    sch.jobSchedule[x1] = sch.jobSchedule[x2];
    sch.jobSchedule[x2] = temp;

    x1 = rndNext(0, N);
    x2 = rndNext(0, N);

    const macTemp = sch.macSchedule[x1];
    sch.macSchedule[x1] = sch.macSchedule[x2];
    sch.macSchedule[x2] = macTemp;
}
```

```
// Мутация циклическим сдвигом
ShiftMutation(sch) {
    const N = this.krlength;
    let x1 = rndNext(0, N);
    let x2 = rndNext(0, N);
    if (x1 > x2) {
        const t = x1;
        x1 = x2;
        x2 = t;
    }
    const job = sch.jobSchedule[x1];
    for (let i = 0; i < sch.jobSchedule.length; i++) {
        if (i > x1 && i <= x2) {
            sch.jobSchedule[i - 1] = sch.jobSchedule[i];
        }
    }
    sch.jobSchedule[x2] = job;
    x1 = rndNext(0, N);
    x2 = rndNext(0, N);
    if (x1 > x2) {
        const t = x1;
        x1 = x2;
        x2 = t;
    }

    const mac = sch.macSchedule[x1];
    for (let i = 0; i < sch.macSchedule.length; i++) {
        if (i > x1 && i <= x2) {
            sch.macSchedule[i - 1] = sch.macSchedule[i];
        }
    }
    sch.macSchedule[x2] = mac;
}

// Мутация инверсией
ReverseMutation(sch) {
    const N = this.krlength;
    let el = rndNext(1, N);
    let x1 = rndNext(0, N - el);
    let x2 = rndNext(0, N - el);
    x1 = Math.min(x1, x2);
    x2 = Math.max(x1, x2);
```

```
    const temps = new Int32Array(el);
    let kk = x1;
    for (let i = 0; i < temps.length; i++) {
        temps[i] = sch.jobSchedule[kk];
        kk++;
    }
    let step = x2 - x1;
    const temps2 = new Int32Array(step);
    for (let i = x1 + el; i < x1 + el + step; i++) {
        temps2[i - x1 - el] = sch.jobSchedule[i];
    }
    for (let i = x1; i < x1 + temps2.length; i++) {
        sch.jobSchedule[i] = temps2[i - x1];
    }
    for (let i = x2; i < x2 + temps.length; i++) {
        sch.jobSchedule[i] = temps[i - x2];
    }

    el = rndNext(1, N);
    x1 = rndNext(0, N - el);
    x2 = rndNext(0, N - el);
    x1 = Math.min(x1, x2);
    x2 = Math.max(x1, x2);
    const tempMacs = new Int32Array(el);
    kk = x1;
    for (let i = 0; i < tempMacs.Length; i++) {
        tempMacs[i] = sch.macSchedule[kk];
        kk++;
    }
    step = x2 - x1;
    const tempMacs2 = new Int32Array(step);
    for (let i = x1 + el; i < x1 + el + step; i++) {
        tempMacs2[i - x1 - el] = sch.macSchedule[i];
    }
    for (let i = x1; i < x1 + tempMacs2.length; i++) {
        sch.macSchedule[i] = tempMacs2[i - x1];
    }
    for (let i = x2; i < x2 + tempMacs.length; i++) {
        sch.macSchedule[i] = tempMacs[i - x2];
    }
}
```