ВОРОНЕЖСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

На правах рукописи

Асвад Фирас М.

МОДЕЛИ СОСТАВЛЕНИЯ РАСПИСАНИЯ ЗАНЯТИЙ НА ОСНОВЕ ГЕНЕТИЧЕСКОГО АЛГОРИТМА НА ПРИМЕРЕ ВУЗА ИРАКА

Специальность 05.13.17 – Теоретические основы информатики

ДИССЕРТАЦИЯ
на соискание ученой степени кандидата технических наук

научный руководитель – доктор
технических наук, профессор
Астахова Ирина Федоровна

Воронеж - 2013

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Актуальность темы

Развитие исследований, направленных на решение задачи составления расписания, можно разбить на два этапа. Первый этап имеет начало в 80-е годы и заканчивается в середине 90-х. В этот период масштабно применяются классические методы решения задач целочисленного программирования: метод полного перебора, метод раскраски графа, метод ветвей и границ (Безгинов А.Н., Трегубов С.Ю., Логоша Б.А., Петропавловская А.В., Гусева Н.Я.). Используемые в этих разработках методы имеют высокую степень формализации как самой задачи, так и используемых алгоритмов. Применение классических методов в образовательных системах обучения становится малоэффективным ввиду большой размерности задачи и значительных временных затрат. Это привело к появлению методов, получивших название интеллектуальных, положившему начало второму этапу. В их основе лежит использование различных эвристик и эвристических алгоритмов (Костин Л.А., Клеванский Н.Н., Маслов М.Г.). Решение задачи составления расписания с помощью эвристик не гарантирует нахождения глобального оптимума. Существует ряд работ, использующих для автоматизации составления расписания математический аппарат нечеткой логики (Ханов Г.В., Алабужев Е.В., Борисов А.Н., Алексеев А.В., Меркурьев Е.В. и д.р.). Нечеткая логика позволяет заметно упростить формализацию требований, но часто приводит к построению расписания, имеющего не лучшие характеристики в результате перехода от «жестких» требований к более «мягким». В настоящее время для решения задачи составления расписания применяется ещё один новый подход – нейронные сети (Пилиньский М., Рутковская Д.). Важнейшим недостатком применения этого подхода является сложность выбора начального состояния нейронной сети. В последние годы особое распространение получили исследования методов эволюционного поиска (Ерунов В.П., Морковин И.И., Каширина И.Л., Низамова Г.Ф., Коробкин А.А.). Применение методов

эволюционного поиска приводит к получению хороших результатов, однако имеет место высокая вычислительная трудоёмкость и относительная неэффективность на заключительных этапах эволюции. В работе Низамовой Г.Ф. используются методы системного анализа, генетических алгоритмов и теории важности критериев. На основании анализа и выявленных недостатков существующих разработок в настоящей работе проводится исследование, направленное на решение задачи составления расписания с использованием агрегированного генетического алгоритма, для чего проводится формализация некоторых составляющих учебного процесса.

Цель работы и основные задачи

Целью диссертационной работы является разработка и исследование моделей формализации составления расписания занятий с использованием генетического алгоритма и его применение к вузам Ирака. Для достижения цели необходимо решить следующие задачи:

1. Разработать модели составления расписания занятий в вузе и генетический алгоритм, осуществляющий поиск его квазиоптимального варианта.
2. Построить и исследовать структурные модели информационного процесса составления расписания с помощью Rational Rose.
3. Разработать специальное программное обеспечение, позволяющее составить квазиоптимальное расписание занятий в вузе Ирака.

Объект исследования – модели составления расписания занятий в вузах, **предмет исследования** – генетические алгоритмы составления расписания занятий в вузах Ирака (г. Диала).

Методы исследования

При решении поставленных задач использовались методы системной декомпозиции, математического моделирования, методы и алгоритмы эволюционного моделирования, методы структурного моделирования, методы

объектно-ориентированного программирования и объектно-ориентированного проектирования.

Научная новизна

В работе получены следующие результаты, характеризующиеся научной новизной:

1. Модель для задачи составления расписания занятий, отличительной особенностью которой является агрегированное представление объектов расписания.
2. Генетический алгоритм для нахождения квазиоптимального решения, отличительной особенностью которого является его представление в виде особи, состоящей из трех хромосом со специальными операторами скрещивания и мутации, и использование гена «конфликтов», которые позволяют получить эффективный алгоритм составления расписания.
3. Структурные модели системы информационного процесса составления расписания, основанные на CASE технологии Rational Rose, отличительной особенностью которых является работа с агрегированными объектами.
4. Специальное программное обеспечение, отличающееся работой с агрегированными объектами и геном «конфликтов», сокращающим время составления расписания на ЭВМ.

Теоретическая и практическая ценность

В работе проведена формализация организационных компонентов учебного процесса: составление расписания и построение структурных моделей системы. Практическая ценность работы заключается в возможности использования разработанного программного обеспечения для принятия решений при построении квазиоптимального расписания учебных занятий и его применение для получения квазиоптимального расписания в вузе г. Диала

Ирака. По результатам работы получено свидетельство о регистрации программного комплекса на ЭВМ «Разработка системы составления расписания ВУЗа Ирака» в Федеральном институте промышленной собственности (ФИПС) № 20126118248 от 11 сентября 2012г.

Апробация работы

Основные результаты, представленные в диссертационной работе, докладывались и обсуждались на Международной конференции «Актуальные проблемы прикладной математики, информатики и механики» (Воронеж, 2012), на XIII межд. научно-технической конф. «Кибернетика и высокие технологии XXI века» (Воронеж 2012), на Воронежской весенней математической школе «Понтрягинские чтения» (Воронеж 2010, 2012), на научных сессиях Воронежского государственного университета, (Воронеж, 2011, 2012); на Двенадцатом всероссийском симпозиуме по прикладной и промышленной математике (Москва, 2011 г.), на семинаре каф. «Компьютерное и математическое моделирование» Тамбовского государственного университета (Тамбов, 2013 г.).

Публикации

Результаты диссертации опубликованы в 9 работах. Из совместных работ в диссертацию вошли только результаты, принадлежащие лично диссертанту. Диссертантом получена модель с ограничениями составления расписания, генетический алгоритм с агрегированными объектами, разработаны структура популяции и каждой особи, операторы кроссинговера и мутации, разработаны структурные модели информационного процесса составления расписания, разработан программный комплекс и проведено его применение к вузу Ирака. Списку ВАК соответствуют работы [1-2].

Структура и объём диссертации

Диссертация состоит из введения, 4 глав, разбитых на пункты, заключения, списка используемой литературы из 130 наименований и

приложения. Общий объем диссертации – 118 страниц. Работа содержит 57 рисунков, 2 диаграммы и 14 таблиц.

Область исследований.

Диссертационная работа соответствует следующим пунктам шифра специальности 05.13.17 – Теоретические основы информатики:

1. Исследование, в том числе с помощью средств вычислительной техники, информационных процессов, информационных потребностей коллективных и индивидуальных пользователей.

2. Исследование информационных структур, разработка и анализ моделей информационных процессов и структур.

13. Применение бионических принципов, методов и моделей в информационных технологиях.

На защиту выносятся:

1. Модель составления расписания занятий в вузе и генетический алгоритм, организующий поиск его квазиоптимального варианта.
2. Структурные модели описания информационного процесса составления расписания, основанные на CASE технологии Rational Rose, отличительной особенностью которых является работа с агрегированными объектами.
3. Специальное программное обеспечение, позволяющее составить квазиоптимальное расписание занятий в вузе и применение его для вуза г. Диала в Ираке.

Глава 1. Анализ моделей и методов составления расписания учебных занятий

1.1 Информационные технологии в сфере образования

Информатизация сферы образования – одно из приоритетных направлений процесса развития общества. Задача внедрения новых информационных технологий в сферу образования рассматривается в рамках проекта федеральной целевой программы "Развитие единой образовательной информационной среды на 2010-2015 годы" и считается одной из главных. Она включает в себя создание новых образовательных программ на основе информационных технологий, развитие сети электронных библиотек, модернизация и развитие существующей сетевой инфраструктуры и др. Повышение эффективности обучающего процесса невозможно достичь без использования автоматизированных рабочих мест для решения задач, возникающих в ВУЗах. К таким задачам относятся учет персонала, материально–технических ресурсов, управление документацией, распределение преподавательской нагрузки и т.д. Особо следует отметить важность проблемы создания информационного обеспечения для организационных процессов учебных заведений, от которого в определенной степени зависит эффективность использования научно–педагогического потенциала. В это направление включены задачи назначения и распределения учебной нагрузки студентов и преподавателей – составление расписания занятий и экзаменов. Эта составляющая учебного процесса регламентирует трудовой ритм и влияет на творческую отдачу преподавателей и студентов, поэтому ее можно рассматривать как фактор оптимизации использования ограниченных трудовых ресурсов. Технологию же разработки расписания следует воспринимать не

только как трудоемкий технический процесс или объект автоматизации с использованием ЭВМ, но и как процесс оптимального управления. Таким образом, задачи получения оптимальных расписаний занятий и экзаменов имеют очевидный социальный и экономический эффект [26-27].

1.1.1 Анализ существующих программных комплексов решения задачи составления расписания

Сегодняшняя ситуация на рынке такова, что очень мало программных продуктов соответствует требуемой функциональности, необходимой для составления расписания в ВУЗе. Такое положение вещей легко объясняется тем, что школьное образование на сегодняшний день более "стандартизовано" (в смысле организации учебного процесса), чем вузовское. Такая стандартизация ведет к большому объему потенциального рынка продаж копий программного обеспечения, что приводит к окупаемости продукта.

В случае ВУЗов спрос на системы составления расписаний даже больше, чем для школ, но имеет место большая специфика организации учебного процесса в каждом отдельно взятом ВУЗе. Создать унифицированное программное обеспечение не представляется возможным, а стоимость создания специализированного продукта у сторонних разработчиков оказывается неоправданно велика. Таким образом, пока ни один программный продукт не позволяет достаточно просто управлять учебным процессом даже в период сессии.

Рассмотрим возможности наиболее популярных на российском рынке программных продуктов, реализующих решение задачи составления расписания [26, 27]:

aSc Timetables (IBN). Программа разработана для всех типов начальных и средних учебных заведений. Среди основных характеристик разработчики выделяют: оптимальное использование кабинетов и других школьных

помещений, соблюдение всех существующих психологических и гигиенических требований, устранение возможных ошибок и субъективных факторов при составлении расписаний в школе, учет пожеланий учителей и малое количество окон, комфортный интерфейс, позволяющий как можно более эффективно вводить данные. Кроме автоматического составления расписания по классам, программа составляет отдельные расписания для кабинетов и учителей. Существует возможность архивирования и хранения нескольких копий существующих расписаний, возможность замены учителей на период болезни или отсутствия.

Построение же расписания в ВУЗах этой программой не предусмотрено.

Система *AVTOR-2+*. Предназначена для составления расписаний занятий и сопровождения их в течение всего учебного года. *AVTOR-2+* - универсальная система. Есть несколько версий программы, рассчитанные на любые учебные заведения. *AVTOR-2+* имеет приятный дизайн и дружественный интерфейс. Программа достаточно проста в освоении. Имеется подробное руководство, в котором описаны все возможности и способы работы с программой. Программа работает на любых IBM-совместимых компьютерах, начиная с 486DX с оперативной памятью 4Mb (и выше), занимает около 1 Mb на жестком диске. Операционная система: MS DOS, либо WINDOWS 95/98.

Время работы зависит от размера учебного заведения и мощности компьютера.

Недостатками программы являются не стандартизированный интерфейс, необходимость ручной коррекции готового расписания.

Методист (ИнфоРесурс). Основные возможности программы: распределение и контроль учебной нагрузки, учет методических рекомендаций и личных пожеланий преподавателей ("окна", методические дни, нежелательные уроки и дни недели); составление расписания для любого типа

учебного заведения: недельное или семестровое (цикл составляет от 1 до 23 недель); учет объединения групп (классов) в потоки и разбиения классов на подгруппы, закрепление специальных аудиторий (компьютерные классы, лингафонные кабинеты, спортзал и т.п.); учет времени переходов между корпусами; указание причин "неудачного назначения" занятий (занята желаемая аудитория, преподаватель назначен в нежелательный для него день недели) с возможностью их "ручного" исправления (получается, в этом случае программа не может автоматически решать такие проблемы); возможность многократного "улучшения" расписания (автоматически или "вручную"); возможность указания уровня значимости учитываемых при составлении расписания факторов; возможность введения приоритетов преподавателей - степени учета их индивидуальных пожеланий.

Ограничения функциональности “Методиста”: многосменные расписания ограничены максимальным кол-вом уроков в день – 7; занятия всегда начинаются с первого урока / пары (при необходимости возможно назначение на первую пару "свободного занятия"); не учитывается время перемен (например для проверки возможности перехода между корпусами); не учитывается "уровень сложности" занятий для их рационального распределения по неделе; продолжительность занятий постоянна (невозможно составление расписания для 30 мин. урока в младших и 45 мин. - в старших классах).

Система “*Расписание*”. Программа “Расписание” ориентирована на составление школьного расписания. Использование же в ВУЗ`ах и колледжах возможно лишь с некоторыми оговорками. Составление расписания производится в рамках комплекса условий, которые определяются на шагах ввода исходных данных.

Ректор 3.7.9. Возможности программы: учет санитарных правил и норм, учет распределения классов по сменам, возможность объединения классов в поток, деление класса (потока) на группы, различные формы таблиц

расписания, сокращение "окон" в расписании учителей, представление данных в форматах Excel, Word и HTML, планирование замен. В состав программного продукта входит модуль для автоматического расчета тарификации. Модуль "Тарификация" позволяет: подсчитывать часы по I, II и III ступеням; учитывать руководство кабинетом; подсчитывать часы за проверку тетрадей по I, II и III ступеням; учитывать внеклассную и кружковую работу, обучение на дому; рассчитывать педагогический стаж на дату тарификации; учитывать уровень образования и должность; учитывать квалификацию учителя и даты аттестации; учитывать классное руководство; экспортировать таблицы в форматах Excel, Word и HTML.

Недостатками программы является долгая работа, нереализованная возможность назначения аудиторий (программа отмечает аудиторию, только если она жестко задана для этой нагрузки на этапе ввода данных), множество окон и необходимость ручной коррекции готового расписания.

Таким образом, из приведенного списка только программа "Методист" более или менее соответствует требуемой функциональности программного продукта составления расписания в ВУЗе.

Разработаны автоматизированные системы по управлению учебным процессом. Система управления обучением MOODLE представляет собой среду, которая проектировалась для организации деятельностного обучения, в основе которого лежит взаимодействие всех участников учебного процесса. Система MOODLE позволяет обеспечить:

– многовариантность представления информации;

– интерактивность обучения;

– многократное повторение изучаемого материала;

– структурирование контента и его модульность;

– создание постоянно активной справочной среды;

– самоконтроль учебных действий;

– выстраивание индивидуальных образовательных траекторий;

– конфиденциальность обучения;

– соответствие принципам успешного обучения.

Основами успешного обучения является:

– научность обучения;

– последовательность и систематичность обучения;

– доступность обучения;

– наглядность обучения;

– сознательность и активность в обучении;

– прочность полученных знаний и навыков, умений;

– индивидуализация обучения.

Основными понятиями MOODLE является курс, профиль пользователя. Форум – это модуль, который позволяет общаться участникам дистанционной программы. Роль определяет статус пользователя. Существует возможность преобразования текстовых значений в мультимедиа и гипертекстовое представление. «Банк текстовых заданий» содержит все вопросы данного курса. Существует возможность создания вычисляемого вопроса, числового вопроса, проводить анализ результатов тестирования с помощью теста самоконтроля, проводить тренинг. Можно формировать элемент «Задание», учение отвечает, учитель сохраняет комментарий.

Появились диссертационные работы, позволяющие составить оптимальное расписание [52,58,77]. В работе [77] рассматривается структуризация исходной информации для составления расписания занятий, описывается агрегированный генетический алгоритм на основе этой структуризации, однако решение задачи недоведено до конца, результатов вычислительного алгоритма нет, отсутствует и описание программного комплекса. Однако, идеи, высказанные в этой работе очень полезны для дальнейшего решения задачи составления расписания. В работе [58] приводится

составление расписания для экзаменационной сессии, это упрощение задачи позволило автору получить решение за достаточно короткое время на ЭВМ. В работе [52] с применением идей работы Низамовой [77] составления расписания, в некоторых частных случаях эта модель дает приемлемое решение на ЭВМ, поэтому считать, что задача составления расписания в вузе решена, будет не совсем верно. Т.к. нет программного комплекса протестированного и отлаженного, позволяющего составить расписание в вузе даже при наличии небольшого количества ограничений.

Необходимо также заметить, что ни одна из перечисленных программ не имеет возможности быть использованной как часть существующей уже в ВУЗе информационной системы путем интегрирования в нее и повышения ее эффективности работы.

В нашем же случае разрабатывается модель автоматизированного составления расписания сессий в ВУЗе которая позволила бы эффективно решать задачу и обладала бы гибкостью (от создания базы данных до ее изменений с течением времени) для адаптации системы в рамках конкретной практической задачи, а также имела возможность экспортировать и импортировать существующие данные в систему.

1.1.2 Основные подходы к решению задачи составления расписания

Поскольку интересы участников учебного процесса многообразны, задача планирования и распределения ресурсов – многокритериальная. Система составления расписания в ВУЗе обязательно должна реализовывать ряд основных функций:

- выбор преподавателей, выбор групп студентов, выбор дисциплин,
- закрепление некоторых аудиторий за определенными предметами,
- ввод ограничений на дисциплины по сложности, от которых зависит их периодичность проведения,

- ввод приоритетов и ограничений на дни, в которые необходимо предоставить или нет возможность провести ту или иную дисциплину,
- возможность перераспределить ресурсы по требованию.

Кроме того, существуют еще и специфические требования каждого ВУЗа к функциональным возможностям проекта.

Решение поставленных задач включает использование различных типов алгоритмов. Наиболее распространенным подходом к решению данного типа задач на сегодняшнее время является использование комбинаторных алгоритмов. Следует также заметить появление нового направления для решения широкого класса прикладных задач – использование генетических алгоритмов. Рассмотрим их подробнее.

Комбинаторные алгоритмы. В них выполняется перебор всевозможных моделей из заданного базиса с выбором лучшей них по заданному критерию селекции.

При переборе сложность частичных моделей постепенно наращивается от 1 до максимального числа n (числа аргументов базисного набора функций) [48, 101, 124].

Таким образом, общая схема комбинаторного алгоритма включает следующие операции:

- Определяются коэффициенты всех частных моделей при сложности s=1..n.
- Для каждой из них вычисляется значение внешнего индивидуального или комбинированного критерия сложности селекции.
- Единственная модель оптимальной сложности выбирается по минимальному значению критерия.

Генетические алгоритмы (ГА) представляют собой новое направление. Они способны не только решать и сокращать перебор в сложных задачах, но и легко адаптироваться к изменению проблемы [31].

- Функция ГА генерирует определенное количество случайных возможных решений (хромосом), которые являются начальным решением. Это отличает ГА от стандартных методов, у которых начальное состояние алгоритма всегда одно и то же.
- На следующем шаге для каждой хромосомы вычисляется “коэффициент выживаемости” - некое численное значение, зависящее от её близости к ответу.
- Множество хромосом, имеющих большую выживаемость (близость к ответу), получают возможность "воспроизводить" потомство с помощью "перекрестного скрещивания" с другими хромосомами множества. Это приводит к появлению новых хромосом (возможных решений), которые сочетают в себе некоторые характеристики, наследуемые ими от родителей. Наименее приспособленные хромосомы с меньшей вероятностью смогут участвовать в воспроизведении потомков, так что те свойства, которыми они обладали, будут постепенно исчезать из популяции в процессе эволюции. Так воспроизводится вся новая популяция допустимых решений, выбирая лучших представителей предыдущего поколения, скрещивая их и получая множество новых хромосом. Таким образом, из поколения в поколение, хорошие характеристики распространяются по всей популяции. Скрещивание наиболее приспособленных хромосом приводит к тому, что исследуются наиболее перспективные участки пространства поиска.
- Процесс повторяется до тех пор, пока решение не найдено, или не получено достаточно к нему приближенное.

ГА является достаточно мощным средством и может с успехом применяться для широкого класса прикладных задач, включая те, которые трудно, а иногда и вовсе невозможно, решить другими методам. Однако, ГА, как и другие методы эволюционных вычислений, не гарантирует обнаружения

глобального решения за полиномиальное время. Главным же преимуществом ГА-мов является то, что они могут применяться даже для решения сложных задач, где невозможно применить специальные методы. Даже там, где хорошо работают существующие методики, можно достигнуть их улучшения сочетанием с ГА. Сила генетического алгоритма заключена в его способности манипулировать одновременно многими параметрами, эта особенность ГА использовалась в сотнях прикладных программ, включая проектирование самолетов, настройку параметров алгоритмов и поиск устойчивых состояний систем нелинейных дифференциальных уравнений.

В генетических алгоритмах [31] можно выделить стандартную схему, а каждый следующий алгоритм отличается наполнением этого стандартного в зависимости от задачи, стандартный алгоритм состоит из следующих шагов:

1. Сконструировать начальную популяцию. Ввести точку отсчета t = 0. Вычислить приспособленность каждой хромосомы в популяции, а затем среднюю приспособленность всей популяции.

2. Установить следующее время t=t+1. Произвести выбор двух родителей (хромосом) для реализации оператора кроссинговера. Он выполняется случайным образом пропорционально приспособленности родителей.

3. Сформировать генотип потомков. Для этого с заданной вероятность произвести оператор кроссинговера над генотипами выбранных хромосом. Далее с вероятность 0,5 выбрать один из потомков $P_i(t)$ и сохранить как член новой популяции. После этого к $P_i(t)$ последовательно применить оператор инверсии, а затем - оператор мутации с заданными вероятностями. Полученный генотип потомка сохранить как $P_k(t)$.

4. Определить количество хромосом для исключения их из популяции, чтобы ее размер оставался постоянным. Текущую популяцию обновить заменой отобранных хромосом на потомков $P_k(t)$.

5. Произвести определение приспособленности (целевой функции) т перерасчет средней приспособленности всей полученной популяции.

6. Если $t=t_{заданному}$, то перейти к 7, если нет, то перейти к 2.

7 Конец работы.

Данный алгоритм известен как упрощенный план Холланда. В практических задачах вместо понятия «приспособленность» пользуюся понятием «целевая функция». Каждый конкретный алгоритм отличается только наполнением этапов в зависимости от задачи.

В настоящий момент в работах Кашириной разработано много представлений начальной популяции, операторов кроссинговера на примере задачи коммивояжера. Известно соседское преставление, кроссоверы альтернативных ребер, кроссовер фрагментов, эвристический кроссовер. Имеется порядковое представление, путевое представление. Для них разработаны частично отображающий кроссовер, упорядоченный кроссовер, кроссовер рекомбинации ребер, циклический кроссовер, измененный кроссовер. Разработано несколько операторов мутации: мутация со случайным выбором ген, «жадная » мутация. Придумано матричное представление.

Основными недостатками рассмотренных подходов к решению задачи являются длительность проведения вычислительного эксперимента (в случае комбинаторных алгоритмов), сложность реализации и возможность не найти оптимального по заданному критерию решения во втором случае. Необходимо найти некоторое сочетание применения этих подходов.

1.2 Задача выбора и составление расписания

Задача планирования загрузки преподавательского состава и распределения аудиторного фонда возникает во всех учебных заведениях с определенным числом преподавателей и некоторым числом аудиторий. Непросто организовать учебный процесс в рамках задачи распределения

ресурсов материальных, человеческих, трудовых, да еще и так, чтобы построенная система была оптимальной по многим критериям и не выходила бы за рамки объявленных ограничений.

1.2.1 Общие вопросы, связанные с теорией расписаний

Под расписанием обычно подразумевается набор некоторых действий, выполнение которых нужно расставить в определенный промежуток времени, учитывая разнообразные ограничения, накладываемые на порядок выполнения этих действий [107].

Следует отметить, что задачи составления расписания очень трудоемки. Для их решения привлекался обширный арсенал средств прикладной математики, проводились многочисленные экспериментальные исследования, выдвигались и анализировались различные гипотезы, разрабатывались новые подходы и методы. Тем не менее, значительная доля полученных результатов носит негативный характер и скорее выявляет сложность проблемы, чем намечает конструктивные пути ее решения [110].

На протяжении существования теории расписаний предлагались различные методы решения, задач упорядочения операций обработки деталей машинами и распределительных задач. Это явление объясняется не отсутствием иных моделей, а тем, что уже на стадии решения простейших по постановке задач возникли серьезные затруднения. Математическая сторона вопроса довольно широко освещена в литературе, хотя всесторонней аналитической теории таких систем нет [127]. Обычно большинство задач сводится к задачам линейного программирования, но, как правило, даже самые простейшие имеют большой размер. Как правило, задачи теории расписаний – трудные математические задачи. По существу это задачи упорядочения конечного множества требований в условиях целого комплекса ограничений и нехватки ресурсов для выполнения этих требований. Иногда ставится задача поиска упорядочения, которое

оптимизирует некоторый критерий. Каждая такая задача может быть решена конечным перебором всех имеющихся вариантов. Однако число таких вариантов даже в самых небольших задачах столь велико, что их перебор не может быть осуществлен даже с помощью компьютера в обозримое время. Для решения таких задач нужно использовать математические и эвристические методы, которые сразу позволяют каким-либо образом отсекать заведомо неподходящие варианты [109].

В общем виде задача составления расписания представима как частный случай задачи выбора и распределения ресурсов. Следует также учитывать, что реализации в аналитическом виде для больших размерностей нашей задачи мы не найдем, но есть возможность синтезировать совокупность моделей и алгоритмов из разных областей знания для достижения цели.

Именно методы математического моделирования этого явления позволяют выявить процедуры принятия решения и разрешить задачу управления данным процессом с их помощью.

1.3 Анализ методологий и современных средств проектирования программных комплексов

1.3.1 Структурный подход к разработке программного обеспечения

Сущность структурного подхода к разработке программного обеспечения заключается в ее декомпозиции на автоматизируемые функции [20]: система разбивается на функциональные подсистемы, которые в свою очередь делятся на подфункции, подразделяемые на задачи и так далее. Процесс декомпозиции продолжается вплоть до конкретных процедур. При этом система сохраняет целостное представление, в котором все составляющие компоненты взаимоувязаны.

При разработке системы "снизу-вверх" от отдельных задач ко всей системе целостность теряется, возникают проблемы при информационной стыковке отдельных компонентов. Однако в отдельных случаях такая технология может оказаться целесообразной: срочность, эксперимент и адаптация.

Все наиболее распространенные методологии структурного подхода к разработке программного обеспечения базируются на ряде общих принципов [68]. Таковыми принципами являются:

- принцип разделения - принцип решения сложных проблем путем их разбиения на множество меньших независимых задач, легких для понимания и решения;
- принцип иерархического упорядочения – принцип организации составных частей проблемы в иерархические древовидные структуры с добавлением новых деталей на каждом уровне;
- принцип абстрагирования – выделение существенных аспектов системы и отвлечение от несущественных;
- принцип формализации – необходимость строгого методического подхода к решению проблемы;
- принцип непротиворечивости – обоснованность и согласованность элементов;
- принцип структурирования данных – данные должны быть структурированы и иерархически организованы.

В структурном анализе используются в основном две группы средств, представляющих функции, выполняемые системой и отношения между данными. Каждой группе средств соответствуют определенные виды моделей (диаграмм), наиболее распространенными, среди которых, являются следующие:

- DFD (Data Flow Diagram) диафаммы потоков данных;

- SADT (Structured Analysis and Design Technique) модели и соответствующие функциональные диаграммы;
- ERD (Entity-Relationship Diagrams) диаграммы “сущность-связь”.

Диаграммы потоков данных и диаграммы «сущность-связь» - наиболее часто используемые в CASE-средствах виды моделей. Это связано с рядом преимуществ DFD-методологии перед SADT-методологией.

На стадии проектирования ИС модели расширяются, уточняются и дополняются диаграммами, отражающими структуру программного обеспечения: архитектуру ПО, структурные схемы программ и диаграммы экранных форм.

Перечисленные модели в совокупности дают полное описание программного обеспечения независимо от того, является ли она существующей или вновь разрабатываемой. Состав диаграмм в каждом конкретном случае зависит от необходимой полноты описания системы.

1.3.2 Объектный подход к разработке программного обеспечения

В наиболее общей и классической постановке объектно-ориентированный подход базируется на следующих концепциях:

- объекта и идентификатора объекта;
- атрибутов и методов;
- классов;
- иерархии и наследования классов.

Любая сущность реального мира в объектно-ориентированных языках и системах моделируется в виде объекта. Любой объект при своем создании получает генерируемый системой уникальный идентификатор, который связан объектом все время его существования и не меняется при изменении состояния объекта.

Каждый объект имеет состояние и поведение. Состояние объекта - набор значений его атрибутов. Поведение объекта - набор методов (программный код), оперирующих над состоянием объекта. Значение атрибута объекта - это тоже некоторый объект или множество объектов. Состояние и поведение объекта инкапсулированы в объекте; взаимодействие объектов производится на основе передачи сообщений и выполнении соответствующих методов.

Множество объектов с одним и тем же набором атрибутов и методов образует класс объектов. Объект должен принадлежать только одному классу (если не учитывать возможности наследования). Допускается наличие примитивных предопределенных классов, объекты-экземпляры которых не имеют атрибутов: целые, строки и т.д. Класс, объекты которого могут служить значениями атрибута объектов другого класса, называется доменом этого атрибута.

Допускается порождение нового класса на основе уже существующего класса – наследование. В этом случае новый класс, называемый подклассом существующего класса (суперкласса), наследует все атрибуты и методы суперкласса. В подклассе, кроме того, могут быть определены дополнительные атрибуты и методы. Различаются случаи простого и множественного наследования. В первом случае подкласс может определяться только на основе одного суперкласса, во втором случае суперклассов может быть несколько.

Если в языке или системе поддерживается единичное наследование классов, набор классов образует древовидную иерархию. При поддержании множественного наследования классы связаны в ориентированный граф с корнем, называемый решеткой классов. Объект подкласса считается принадлежащим любому суперклассу этого класса.

Одной из более поздних идей объектно-ориентированного подхода является идея возможного переопределения атрибутов и методов суперкласса в подклассе (перегрузки методов). Эта возможность увеличивает гибкость, но

порождает дополнительную проблему: при компиляции объектно-ориентированной программы могут быть неизвестны структура и программный код методов объекта, хотя его класс (в общем случае - суперкласс) известен. Для разрешения этой проблемы применяется так называемый метод позднего связывания, означающий, по сути дела, интерпретационный режим выполнения программы с распознаванием деталей реализации объекта во время выполнения посылки сообщения к нему. Введение некоторых ограничений на способ определения подклассов позволяет добиться эффективной реализации без потребностей в интерпретации.

Как видно, при таком наборе базовых понятий, если не принимать во внимание возможности наследования классов и соответствующие проблемы, объектно-ориентированный подход очень близок к подходу языков программирования с абстрактными (или произвольными) типами данных.

С другой стороны, если абстрагироваться от поведенческого аспекта объектов, объектно-ориентированный подход весьма близок к подходу семантического моделирования данных (даже и по терминологии). Фундаментальные абстракции, лежащие в основе семантических моделей, неявно используются и в объектно-ориентированном подходе. На абстракции агрегации основывается построение сложных объектов, значениями атрибутов которых могут быть другие объекты. Абстракция группирования - основа формирования классов объектов. На абстракциях специализации/обобщения основано построение иерархии или решетки классов.

При рассмотрении объектно-ориентированных баз данных (БД), наиболее важным новым качеством, которого позволяет достичь объектно-ориентированный подход, является поведенческий аспект объектов. В прикладных системах, основывавшихся на БД с традиционной организацией (вплоть до тех, которые базировались на семантических моделях данных) существовал принципиальный разрыв между структурной и поведенческой

частями. Структурная часть системы поддерживалась всем аппаратом БД, ее можно было моделировать, верифицировать и т.д., а поведенческая часть создавалась изолированно. В частности, отсутствовали формальный аппарат и системная поддержка совместного моделирования и гарантирования согласованности этих структурной (статической) и поведенческой (динамической) частей. В среде объектно-ориентированных БД (ООБД) проектирование, разработка и сопровождение прикладной системы становится процессом, в котором интегрируются структурный и поведенческий аспекты. Конечно, для этого нужны специальные языки, позволяющие определять объекты и создавать на их основе прикладную систему.

Специфика применения объектно-ориентированного подхода для организации и управления БД потребовала уточненного толкования классических концепций и некоторого их расширения. Это определяется потребностями долговременного хранения объектов во внешней памяти, ассоциативного доступа к объектам, обеспечения согласованного состояния ООБД в условиях мультидоступа и тому подобных возможностей, свойственных базам данных. Выделяются три аспекта, отсутствующие в традиционной парадигме, но требующиеся в ООБД.

Первый аспект касается потребности в средствах спецификации знаний при определении класса (ограничений целостности, правил дедукции и т.п.). Второй аспект - потребность в механизме определения разного рода семантических связей между объектами, вообще говоря, разных классов. Фактически это означает требование полного распространения на ООБД средств семантического моделирования данных. Потребность в использовании абстракции ассоциирования отмечается и в связи с использованием ООБД в сфере автоматизированного проектирования и инженерии. Наконец, третий аспект связан с пересмотром понятия класса. В контексте ООБД оказывается более удобным рассматривать класс как множество объектов данного типа, т.е одновременно поддерживать понятия и типа и класса объектов.

1.3.3 Объединение структурного и объектного подхода в новом поколении CASE-средств

В технологическом цикле создания программного обеспечения принято выделять следующие этапы [66]:

- анализ - определение того, что система будет делать,
- проектирование – определение подсистем и их взаимодействие,
- реализация – разработка подсистем по отдельности,
- объединение – соединение подсистем в единое целое,
- тестирование – проверка работы системы,
- установка – введение системы в действие,
- функционирование - использование системы

В работе [43, 66] показано, что наиболее важными являются ранние этапы создания программного обеспечения – этап анализа и этап проектирования, поскольку именно на этих этапах могут быть допущены наиболее опасные и дорогостоящие ошибки. Существуют различные методологии и CASE-средства, обеспечивающие автоматизацию этих этапов. Такие CASE-средства должны выполнять следующие задачи:

- построение модели бизнес-процессов и анализ этой модели,
- создание структурной модели и связывание структуры с функциональной моделью. Результатом такого связывания должно быть распределение ролей и ответственности участников бизнес-процессов,
- создание сценариев выполнения бизнес-функций, подлежащих автоматизации и полного описание последовательности действий (включающее все возможные сценарии и логику развития),
- создание сущностей и атрибутов и построение на этой основе модели данных,

- определение требований к программному комплексу и связь функциональности системы с бизнес-процессами,
- создание объектной модели, на которой в дальнейшем может быть автоматически сгенерирован программный код,
- интеграция с инструментальными средствами, обеспечивающими поддержку групповой разработки, системами быстрой разработки, средствами управления проектом, средствами управления требованиями, средствами тестирования, средствами управления конфигурациями, средствами распространения и средствами документирования.

Практика показывает, что одна отдельно взятая нотация или инструмент не могут в полной мере удовлетворить всем перечисленным требованиям.

Новое поколение CA8E-средств представляет собой набор связанных между собой инструментальных средств, в полной мере обеспечивающих решение всех задач анализа, проектирования, генерации, тестирования и сопровождения информационных систем. В качестве примера таких средств может служить продукт AllFusion Modeling Suite, фирмы Computer Associates. Это интегрированный комплекс CASE-средств, обеспечивающий все потребности разработчиков программного обеспечения, предназначен для проектирования и анализа баз данных, бизнес-процессов и информационных систем. Комплекс включает продукты: AllFusion Process Modeler, AllFusion Erwin Data Modeler, AllFusion Data Model Validator, AllFusion Model Manager, AllFusion Component Modeler,– использование которых позволяет сократить расходы и повысить продуктивность процесса разработки.

Приведем краткую аннотацию некоторых из этих продуктов.

AllFusion Process Modeler – ведущий инструмент для моделирования бизнес-процессов. Являясь стандартом де-факто, Process Modeler поддерживает сразу три нотации моделирования: IDEF0 (федеральный стандарт США), IDEF3 и DFD.

AllFusion Erwin Data Modeler – лидер среди средств моделирования баз данных и хранилищ данных. Позволяет проектировать, документировать и сопровождать базы данных различных типов. Поддерживая прямое и обратное проектирование для 20 типов СУБД, Erwin Data Modeler повышает качество разрабатываемой БД, производительность труда и скорость разработки.

AllFusion Data Model Validator – инструмент для проверки структуры баз данных и создаваемых в ERwin моделей, позволяющий выявлять недочеты и ошибки проектирования. Data Model Validator дополняет функциональность ERwin, автоматизируя трудоемкую задачу поиска и исправления ошибок, одновременно повышая квалификацию проектировщиков баз данных благодаря встроенной системе обучения.

AllFusion Model Manager – среда для работы группы проектировщиков. Обеспечивает совместный доступ и редактирование моделей, повышая эффективность и скорость работы проектировщиков, является интегрирующим звеном для ERwin (моделирование баз данных) и Process Modeler (моделирование бизнес-процессов). Защищает хранимые на собственном сервере модели, позволяя задавать для сотрудников различный уровень доступа к ним. Руководителям же проектов позволяет координировать весь ход работы.

AllFusion Component Modeler – мощное CASE-средство для моделирования компонентов программного обеспечения и генерации объектного кода приложений на основе созданных моделей. Продукт можно использовать как при создании новых приложений, так и при изменении или объединении существующих. Благодаря интеграции с Process Modeler есть возможность использования функциональной модели вместе с объектной. Поддерживает около десятка стандартных нотаций, таких как UML и Booch, интегрируется с технологиями COM/DCOM, CORBAPlus, Visibroker и др., продуктами CA, Microsoft Rational Software и др.

Oracle Designer (входит в Oracle9i Developer Suite) – высоко функциональное средство проектирования программных систем и баз данных, реализующее технологию CASE и собственную методологию Oracle – “CDM”. Позволяет команде разработчиков полностью провести проект, начиная от анализа бизнес-процессов через моделирование к генерации кода и получению прототипа, а в дальнейшем и окончательного продукта.

Rational Rose – средство моделирования объектно-ориентированных программных систем, базирующееся на языке моделирования UML. Rose способен решать практически любые задачи в проектировании информационных систем: от анализа бизнес процессов до кодогенерации на определенном языке программирования. Только Rose позволяет разрабатывать как высокоуровневые, так и низкоуровневые модели, осуществляя тем самым либо абстрактное проектирование, либо логическое.

Итак, аннотация инструментальных средств показывает – применение CASE-средств позволяет наиболее эффективно использовать преимущества как объектного, так и структурного подхода к созданию информационных систем.

1.4 Выводы, цель и задачи исследования

Приведенный анализ работ позволяет сделать следующие выводы: существующие математические модели в общем представлении сложны и громоздки уже для малых размерностей задач и в большинстве своем сводятся к использованию метода проб и ошибок в решении поставленной задачи планирования. Это ведет к рутинным вычислениям, требующим больших затрат машинного времени при организации вычислительного эксперимента.

Анализ существующего программного обеспечения показывает, что нет программных продуктов, включающих синтезированные алгоритмы и представления предметной области, которые позволяли бы решить задачу не просто перебором решений, а выявлением закономерностей и использованием

их для реализации поставленных целей. Нет методик, рекомендаций по созданию программного обеспечения на основе современных методов проектирования и реализации.

На основании сделанных выше выводов сформулируем цель диссертационной работы. Целью диссертационной работы является разработка и исследование моделей формализации составления расписания занятий с использованием генетического алгоритма и его применение к вузам Ирака. Для достижения цели необходимо решить следующие задачи:

1.Разработать модели составления расписания занятий в вузе и генетический алгоритм, осуществляющий поиск его квазиоптимального варианта.

2.Построить и исследовать структурные модели информационного процесса составления расписания с помощью Rational Rose.

3.Разработать специальное программное обеспечение, позволяющее составить квазиоптимальное расписание занятий в вузе Ирака.

Глава II. Решение задачи составления расписания занятий ВУЗа, с использованием генетических алгоритмов

2.1. Постановка математической модели задачи составления расписания

Составление расписания учебных занятий является одной из важнейших задач управления учебным процессом. В связи с этим проблема автоматизации составления расписаний учебных занятий в образовательных системах массового обучения по-прежнему остается одной из актуальных проблем организации учебного процесса.

Постановка задачи : Пусть для расписания R={B,T,A} на множестве временных интервалов T = {1, . . ., n} необходимо множество аудиторий A(j), где j ={1,…, m}. Каждый элемент множества B блок, его можно представить в виде B = <Pz, Gh, Dl,Yy>, где Pz ∈ P = {1, . . ., p} – множество преподавателей, участвующих в расписании , Gh ∈ {1, ..., Gs} – множество студенческих групп, для которых необходимо составить расписание, Dl ∈ D = {1, ..., Dd} – множество дисциплин; Y= {1, ..., Yy} – множество, задающее тип блока [95**]**.

Набор элементов множества аудиторий A зависит от элементов множества D, т.е. в каждый время j количественные и качественные характеристики множества A различны.

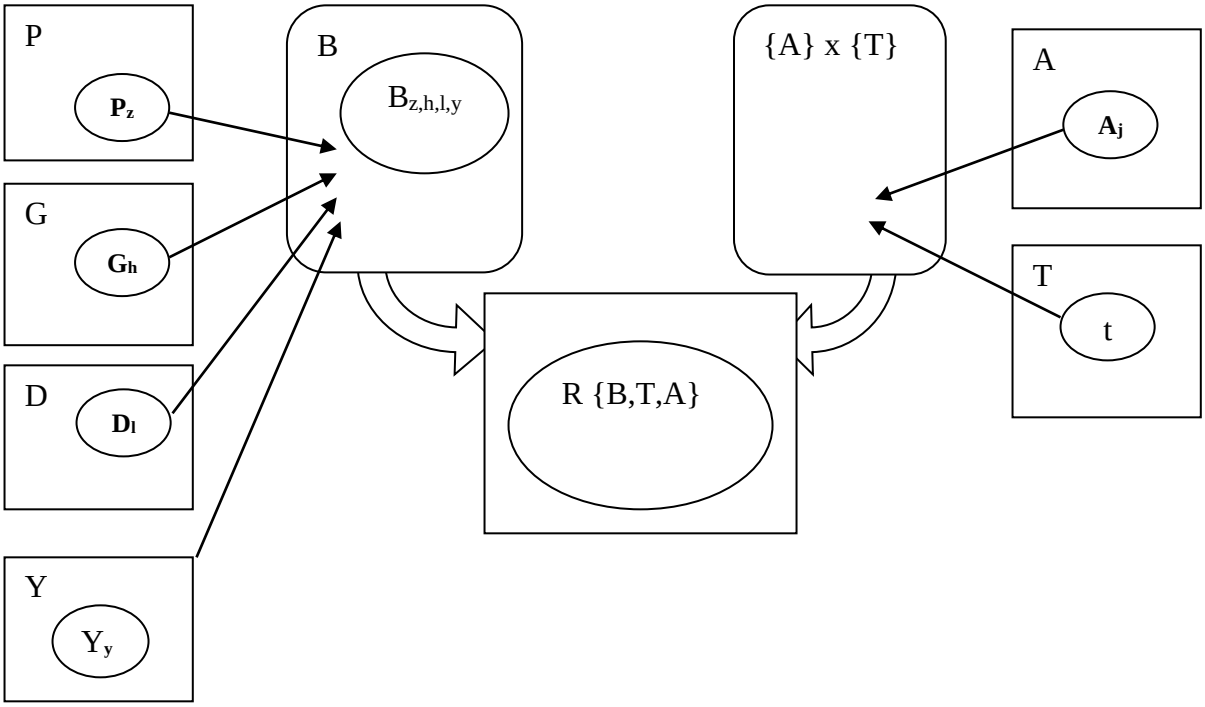


Рис. 2.1 Структура взаимодействия множеств предметной области

Принимая во внимание то, что организация образовательного процесса весьма трудоёмкий процесс, общающийся с огромными объёмами разнородной информации, при составлении расписания целесообразным является использование приёмов агрегирования (композиции) и декомпозиции при представлении исходной информации. Благодаря такому подходу, исходная информация представляется в виде совокупности объектов вместе с указанием существующих между ними связей. Такие модели называют моделями типа «сущность - связь».

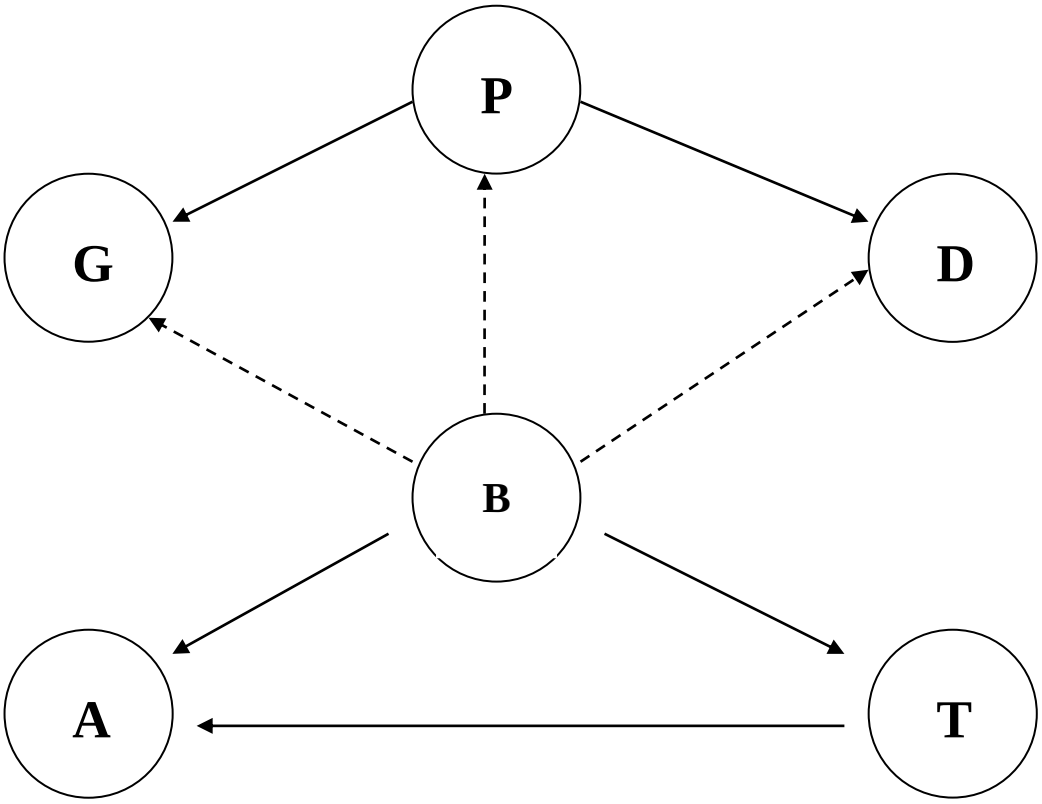
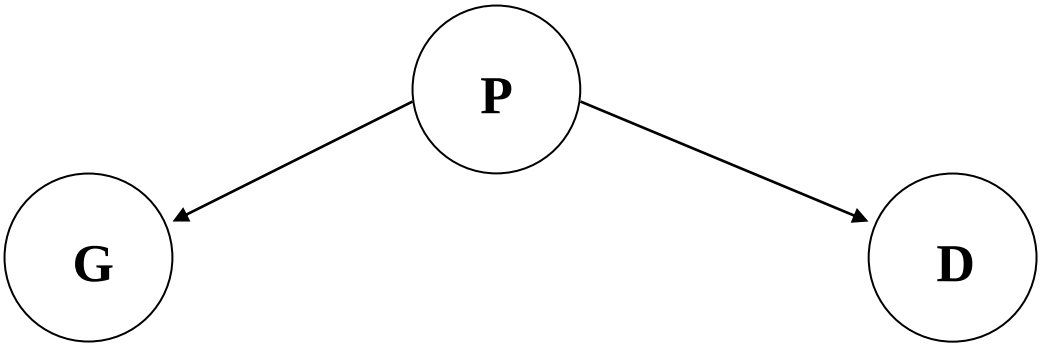


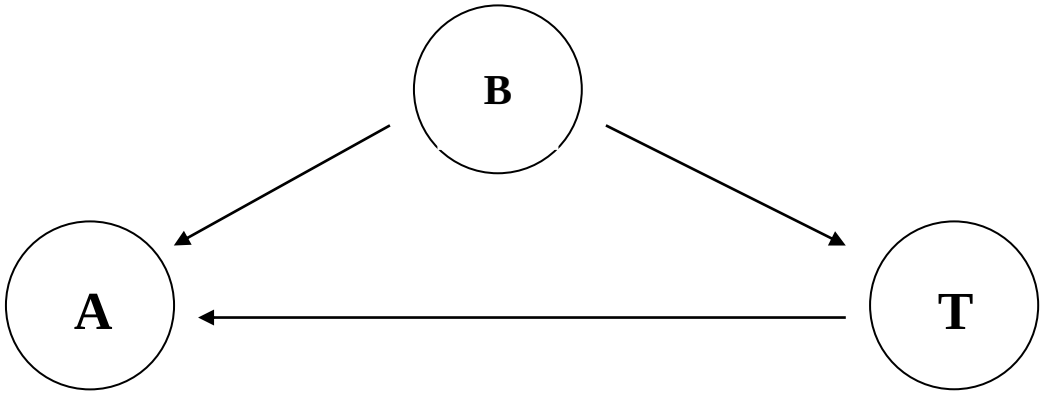
Рис. 2.2 Модель «сущность – связь» исследуемой предметной области

Изображенная на рис. 2.2 частная модель «сущность – связь» представляет собой граф, состоящий из шесть объектов B, G, D, P, A, T, которые имеют между собой различные связи. Связь PG говорит о том, что преподаватель P проводит занятия для группы G.Связь PD говорит о том, что преподаватель P проводит занятия по дисциплине D.Связь BA говорит о том, что Block B проводит занятия в аудитории A. Связь Block B проводит занятия во временной интервал T. То, что во время пары T занятия проводятся в аудитории A, описывает связь TA. Заметим, что связи в этой модели не являются равнозначными. Сплошными линиями на рисунке обозначены существенные связи, а пунктирные линии отображают несущественные связи. При проведении процедуры анализа подобных объектов, несущественными (слабыми) связями принято пренебрегать. В результате удаления слабых (несущественных) связей, рассматриваемый объект распадается на несколько объектов меньшего размера, способных существовать отдельно. Подобное представление сложных объектов

в виде совокупности объектов меньшего размера в теории принято называть декомпозицией.



а)



б)

Рис. 2.3 Декомпозиция исходного объекта.

На рис. 2.3 изображена декомпозиция объекта исследуемой области. Удаление из объекта слабых связей, приводит к делению его на две части меньшего размера. Это позволяет провести их автономный анализ и синтез. Деление на сильные связи и слабые может носить как естественный, так и искусственный характер. Удаление слабых связей и декомпозиция объекта позволяет на этапе проектирования как можно удобнее представить исходную информацию. Объект а) на рис. 2.3 благодаря декомпозиции можно рассматривать с точки зрения расписания. В нём хранится информация о том, какие занятия необходимо проводить в группе. Данный объект можно рассматривать как требование, которое необходимо обслужить. Объект б) на

рис. 2.3 объединяет в себе средства, необходимые для проведения занятий, согласно расписания. В настоящей работе используется временная декомпозиция, т.е. при составлении расписания конструируется сложный объект (расписание) из более простых объектов. В этом случае на каждом этапе синтеза решается менее громоздкая задача оптимизации с меньшим количеством переменных. На использовании данного подхода базируется большая группа методов, получивших название методов последовательной оптимизации [77].

Таким образом, говоря о крупном учебном заведении, выделяются следующие группы объектов-блоков:

1) **Блок** студенческих групп, преподавателей и предметов:
   a) Множество обучающихся групп G.
   b) Множество преподавателей P.
   c) Множество дисциплин D.
   d) тип блока Y.
2) **Блок** аудиторий и времени:
   e) Множество аудиторий A.
   f) Множество учебных пар T (временных интервалов проведения занятий).

Занятия должны проводиться в аудитории, которая как можно больше соответствует роду занятий и размеру групп. Исходя из этого, все аудитории разделяются на несколько групп:

- Аудитории, предназначенные для проведения лекционных занятий, которые способны разместить до 2 групп. Принимается обозначение A_b.
- Аудитории среднего размера, в которых могут проводиться практические занятия 1-2 групп одной специальности. Обозначено A_m.

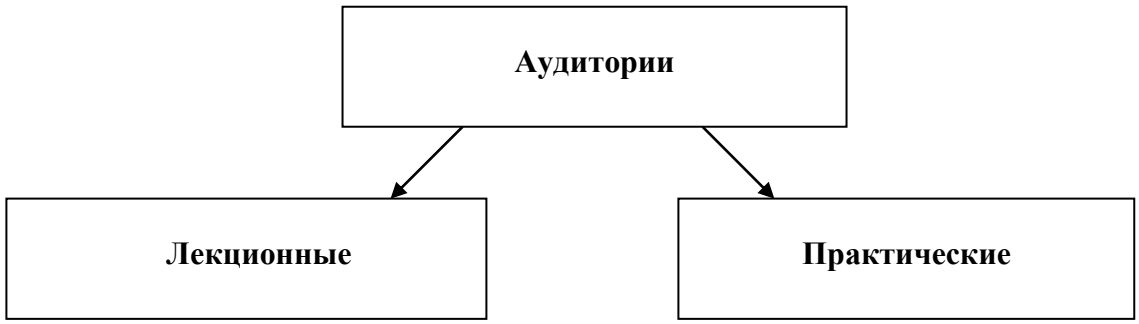


Рис. 2.4. Тип аудиторий

В свою очередь, каждая аудитория имеет свой уникальный номер, присваиваемый ей учебным управлением. Данный номер характеризует расположение данной аудитории, но он не несёт информацию о её конкретном назначении. Для описания аудитории, расположенной в ν – ом учебном корпусе, под номером s и имеющей тип type вводится в рассмотрение трехкомпонентный кортеж:

$$A = \{a_j\},\ a_j = \left(a_j^{\nu}, a_j^{s}, a_j^{type}\right) \qquad (2.1)$$

При описании временных интервалов возможного времени проведения занятий, в терминах теории множеств, учитываются следующие моменты:

Описание временных интервалов проведения предполагает использование сквозной нумерации пар в течение семестра. Временные интервалы описываются множеством T, каждый элемент которого представляет собой трехкомпонентный кортеж вида:

$$t_k = \left(t_k^{id}, t_k^{d}, t_k^{t}\right), \qquad (2.2)$$

где t_k^{id} - номер временного интервала, $t_k^{id} = \overline{1, N_{idps}}$; где t_k^{d} - номер дня, $t_k^{d} = \overline{1, N_{dpw}}$; где t_k^{t} -временной интервал в течение дня, $t_k^{t} = \overline{1, N_{cpd}}$. Здесь N_{idps} - количество времени в неделе , N_{dpw} - число учебных дней в неделе, N_{cpd} - число учебных временных интервалов в течение одного учебного дня.

При описании временных интервалов возможного времени проведения занятий, в терминах теории множеств, учитываются следующие моменты:

- Расписание, как правило, составляется на один семестр.
- Для каждой специальности есть учебный план, в котором содержится информация о количестве учебных дней в неделе, количестве пар в один день, которые можно запланировать.

Описание временных интервалов проведения занятий в таком виде предполагает использование сквозной нумерации пар в течение семестра .

Следующим рассматриваемым объектом является блок (B) занятий, который состоит из следующих объектов:

- D – дисциплины,
- G – группы,
- P – преподаватели,
- T – тип блока.

На основе имеющейся информации о списке учебных групп и учебных планов специальностей можно сказать, что все объекты однозначно определяют, какие занятия должны проводиться в имеющихся учебных группах, какие преподаватели должны преподавать и в каких аудиториях (тип блока).С учетом связи между этими объектами, образуется новая структура – «занятие». Объект «занятие» представляет собой агрегированный объект. При решении поставленной задачи составления расписания имеется дело с множеством блоков занятий. Каждый элемент этого множества определяет дисциплину, по которой требуется провести занятие у конкретной учебной группы.

На основе этих выводов данный объект можно представить в виде множества вектором следующей структуры:

$$B = \{B_i\},\ B_i = (B_i^d, B_i^g, B_i^p, B_i^y), \qquad (2.3)$$

где B_i – элемент множества блоком занятий $B(i = \overline{1, N_{блоков}})$;

$N_{блоков}$ – количество блоков занятий;

B_i^d – дисциплина, преподаваемая в блоке;

B_i^g – группа, у которой проводится занятие;

B_i^p – параметр, определяющий длительность блока;

B_i^y – параметр определения вида занятия.

Рассмотренная выше операция агрегирования применительно к объектам D и G возможна благодаря наличию между ними связи.

Благодаря проведенной процедуре агрегирования удалось уменьшить размерность решаемой задачи. В процессе создания математической модели задачи составления расписания, используется три множества объектов.

$A = \{a_i\}$ – множество аудиторий;

$T = \{t_k\}$ – множество временных интервалов;

$B = \{b_j\}$ – множество блоков занятий.

Для получения расписания необходимо найти три вектора (2.4):

$$\alpha = (\alpha_1, \alpha_2, ..., \alpha_i, ..., \alpha_{N_{блоков}})$$

$$t = (t_1, t_2, ..., t_i, ..., t_{N_{áëîêîâ}}) \qquad (2.4)$$

$$b = (b_1, b_2, ..., b_i, ..., b_{N_{блоков}}), \text{ где}$$

$\alpha_i \in A$ - код аудитории, назначенный блоку занятий $b_i \in B$

$t_i \in T$ - код временного интервала, назначенный первому занятию из блока занятий $b_i \in B$.

Ограничения, налагаемые на расписание, описываются следующим образом: ограничения в Вузах Ирака:

1. расписание составляется только на 5 дней, начиная с воскресенья и заканчивая четвергом;
2. максимальное количество занятий - 3 пары в день для каждой группы студентов любого курса;
3. расписание составляется только для одной недели занятий в течение всего семестра, деления на «числитель/знаменатель» отсутствует;

4. все занятия проходят в одну смену;
5. гендерные проблемы, занятия у студентов мужского пола проходят отдельно от студентов женского пола.

Математическое выражение другой группы ограничений:

1. Математическое выражение, описывающее отсутствие накладок для аудиторий выглядит следующим образом. Для каждой упорядоченной двойки элементов: аудитория и пара, аудитории существует либо единственный блок занятий из множества B, что означает проведение занятия этого блока в этой аудитории в момент данной пары, либо отсутствие блока занятия, указывающее на то, что аудитория свободна.

$$\forall(a_i,t_j)\ a_i \in A\ t_j \in T(\exists\neg b_k\ (a_i = a_k) \wedge (b_k \in B^{t_j})) \vee (\neg\exists b_k\ (a_i = a_k) \wedge (b_k \in B^{t_j})), \quad (2.5)$$

где B^{t_k} – множество блоков занятий, проводимых во время пары t_k.

2. Отсутствие накладок для преподавателей описывает, что для каждой упорядоченной двойки элементов: пара и преподаватель, существует либо единственный блок занятий, которые ведет данный преподаватель во время заданной пары, либо этого блока не существует вообще. Данное ограничение формализуется следующим образом:

$$\forall(p_i,t_j)\ p_i \in P\ t_j \in T(\exists\neg b_k\ (p_i = p_k) \wedge (b_k \in B^{t_j})) \vee (\neg\exists b_k\ (p_i = p_k) \wedge (b_k \in B^{t_j})), \quad (2.6)$$

где B^{t_j} – множество блоков занятий, проводимых во время пары t_k.

3. Отсутствие накладок для учебных групп означает, что для каждой пары элементов: группа и пара, сумма компонент b_i^e вектора b_i блоков из множества $B^{g_j} \cap B^{t_k}$ не превышает единицы. Во время конкретной пары группа находится на одном занятии, или проводится занятие только у одной из подгрупп, либо у обеих, либо занятий нет вообще. Тогда математическое описание данного ограничения имеет вид:

$$\sum b_i^e \le 1, i \in B^{g_n} \cap B^{t_j} \forall(g_n,t_j)\ g_n \in G, t_k \in T \quad , \quad (2.7)$$

где B^{g_n} – множество блоков занятий, в которых присутствует группа g_n, а B^{t_j} – множество блоков занятий, проводящихся во время пары t_k.

4. Соответствие типа аудитории проводимому занятию предполагает, что для каждого блока занятия $b_i, b_i \in B$ аудитория выбирается из допустимого подмножества аудиторий, код этого подмножества хранит компонента b_i^a.

$$\forall b_i \in B\, a_i \in A^{z_i^a}. \quad (2.8)$$

5. Ограничение, налагаемое на количество учебных пар, проводимых в течение одного учебного дня, означает, что для каждой пары элементов: группа и день, число проводимых пар не превышает максимально допустимого – $N_{пар\_\max}$.

$$\sum z_i^e \le N_{пар\_\max}, i \in I_{g_n}^{b_\tau} \forall (z_\tau, g_n)\ z_\tau \in Z, g_n \in G, \quad (2.9)$$

где $Z = \{z_1, z_2, ..., z_{N_{дней}}\}$ – множество учебных дней. Каждый элемент описанного множества, определяется следующим образом:

$z_\tau = \{t_j \in T\ t_j^d = z_\tau\}$.

С учетом (2.5), а также при соблюдении ограничений (2.6) – (2.9), требуется найти такой вариант выбора векторов α, t, b, удовлетворяющее ограничениям (2.6) – (2.9), а также минимизирующее значение критерия потери качества K. Критерий качества основывается на желательных требованиях[81].

$K = \varphi(\alpha, t, b) = \sum_{i=1}^{N} c_i w_i(\alpha, t, b)$, где c_i – значение штрафного коэффициента за невыполнение $i-го$ требования, а w_i – оценка степени невыполнения $i-го$ желательного требования.

К наиболее значимым желательным требованиям относятся:

1. Пожелания преподавательского состава: для формулировки данного требования рассматриваются две матрицы. Первая матрица называется матрицей запретов и выглядит следующим образом:

$$M_{запретов}(i,k) = \begin{cases} 1, запрет\ проведения\ занятия\ для\ i-го\ преподавателя \\ во\ время\ k-ой\ пары, \\ 0, отсутствие\ запрета \end{cases}.$$

Вторая матрица – матрица занятости формируется следующим образом:

$$M_{занятости}(i,k) = \begin{cases} 1, проведение\ занятия\ для\ i-го\ преподавателя \\ во\ время\ k-ой\ пары, \\ 0, отсутствие\ занятий \end{cases}.$$

Требование учета пожеланий преподавателей описывается следующим соотношением:

$$\sum_{p_i \in P} \sum_{t_j \in T} \overline{(M_{запретов}(i,k) \wedge M_{занятости}(i,k))} \to \min .$$

2. Минимизация количества «окон» у преподавателей:

$$\sum\sum (t\_\max_{p_i}^{z_\tau} - t\_\min_{p_i}^{z_\tau}) - N_{блоков} \to \min ,$$

где $t\_\max_{p_i}^{z_\tau}$ – максимальный номер пары в день z_τ у преподавателя p_j, а $t\_\min_{p_i}^{z_\tau}$ – минимальный номер пары в день z_τ у преподавателя p_j.

3. желательное требование равномерности занятий представляется в следующем виде:

$Z = \{z_1, z_2, ..., z_{N_{дней}}\}$ – множество учебных дней.

$M_n^{ср} = \frac{1}{N_{дней}} \sum_{z_\tau \in B} \left| I_{g_n}^{z_\tau} \right|$ – среднее количества занятий в день для группы g_n

Отсюда, требование выглядит так:

$$\sum_{n=1, N_{групп}} D_n^{ср} \to \min$$

На основании описанных требований строится целевая функция на основе минимизации штрафных показателей. Каждое нарушение ограничения или желательного требование увеличивает значение целевой функции в

соответствии с коэффициентом значимости требования k_og_i. В результате целевая функция в общем виде описывается формулой:

$$F_ц = \sum_{i=1..N_{количество ограничений}} og_i \cdot k\_og_i + K$$

производится реализация эволюционного поиска квазиоптимального расписания с использованием генетического алгоритма.

2.2. Систематизация исходной информации

Исходными данными в решении задачи составления расписания плана являются дисциплины обучения, таблица тесноты связей, таблица преподавателей, таблица групп, таблица аудиторий, тип занятий и временные нормы. Данную информацию удобно представить в виде реляционной базы данных. Это позволит организовать довольно простую выборку необходимых при решении задачи данных. А также иметь не сложные способы редактирования и занесения новой информации в базу данных информационной системы. Проектирование описанной базы данных осуществлено с помощью ER – диаграмм. Сущность ER[66] – проектирования заключается в следующем. В предметной области выделяем основные классы объектов и присваиваем им статус сущности. В случае разрабатываемей основной сущностью является дисциплина. Вторая сущность представляет собой класс объектов, описывающих статус дисциплины. Между этими сущностями существует связь «один ко многим», т.е. каждая дисциплина имеет единственный статус, а какой-либо статус может быть поставлен в соответствие нескольким предметам обучения. В результате всех описанных выше рассуждений, база данных учебного плана представима в виде следующей концептуальной модели данных[125-126].

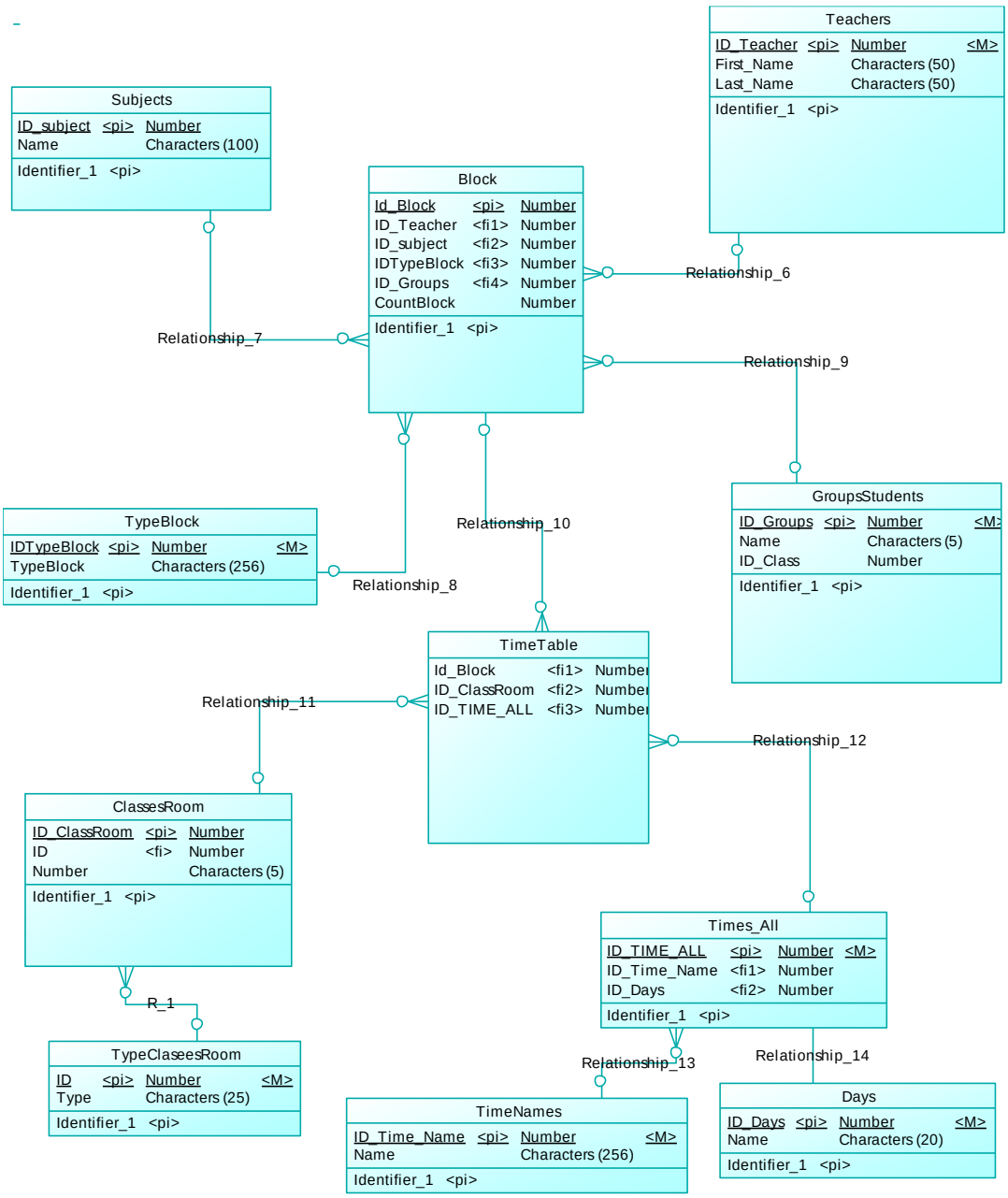


Рис. 2.5 Концептуальная модель базы данных информационной системы.

Данная модель была спроектирована с использование Power Designer. С помощью этого же инструмента мы получаем физическую модель базы данных(совокупность таблиц).

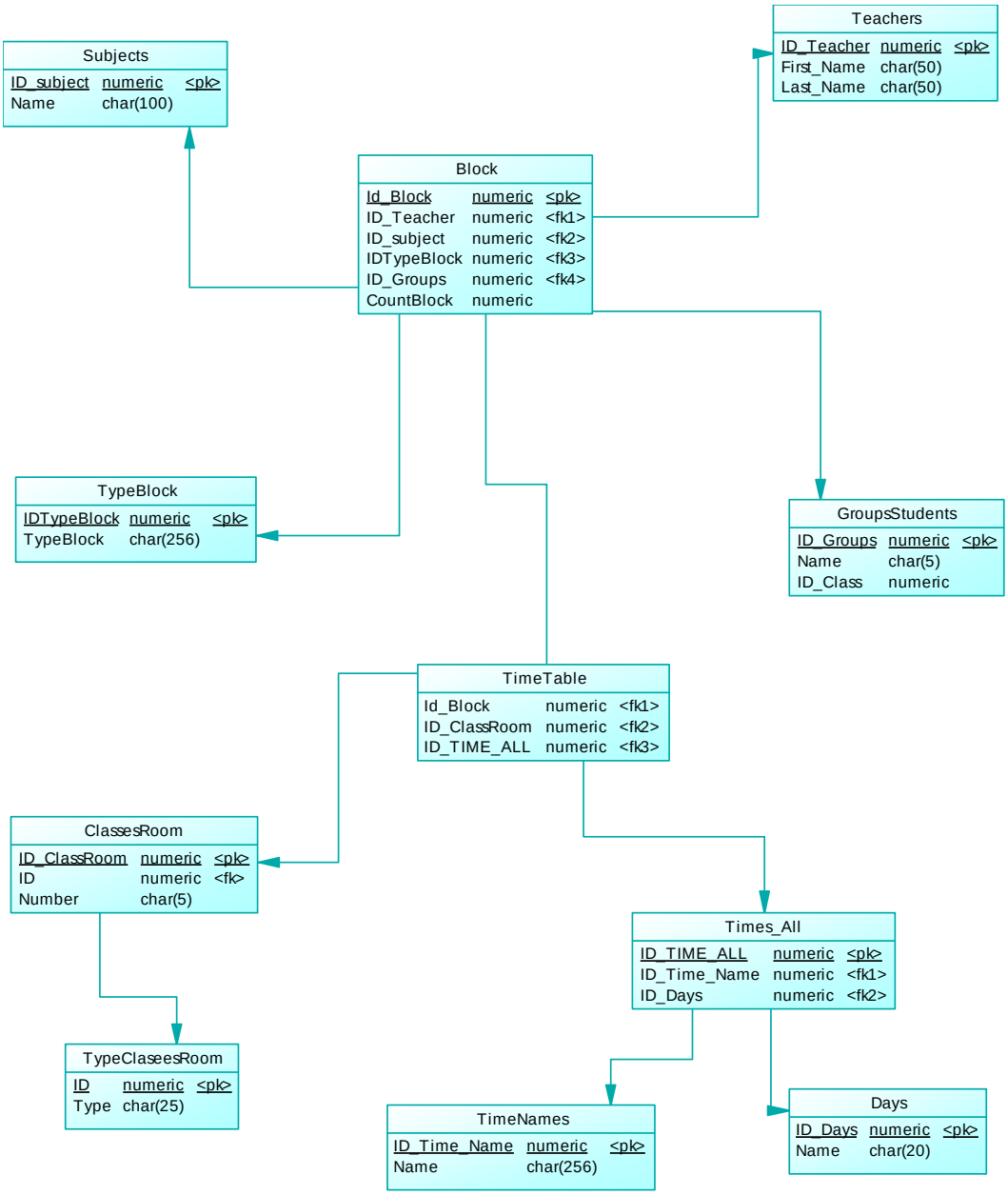


Рис. 2.6 Физическая модель базы данных информационной системы.

Как видно из рисунка, база данных расписания состоит из 5 таблиц. Таблица «Subject» - описывает предметы обучения, «Times_All» содержит информацию о временных нормативах учебного плана. Таблица с именем «Groups» хранит информацию о студентах. Таблица «Teachers» - хранит информацию о преподавателях. «ClassesRoom» содержит информацию о аудиториях. «Block» содержит информацию о занятиях. «TypeBlock» содержит

информацию об ограничениях блока. «Timetable» содержит информацию о расписании. На самом деле последняя таблица имеет наибольшее значение в рамках решаемой задачи. Чтобы получить данные к этой таблице необходимо заполнить все таблицы данными.

Структурирование исходных данных таким образом помогло решить задачу проектирования расписания. Для реализации синтеза расписания ВУЗа в данной работе были применены методы эволюционного поиска. А именно: генетический алгоритм. Генетические алгоритмы описывают не случайный поиск, они эффективно используют информацию, накопленную в процессе эволюции. В отличие от других методов оптимизации эти алгоритмы, как правило, анализируют различные области пространства решений одновременно и поэтому они более приспособлены к нахождению новых областей с лучшими значениями целевой функции. Математически доказано, что не существует идеальной структуры представления, которая позволит кодировать любое возможное решение и производить его оценку, так что для создания хорошей структуры требуется анализ, перебор и эвристические подходы. Возможный вариант представления решения должен позволять проведение различных перестановок в альтернативных решениях.

2.3. Описание разработанного агрегативного генетического алгоритма

Разработанный генетический алгоритм состоит из следующих шагов[31], это стандартная схема генетического алгоритма, отличается она наполнением для этой конкретной задачи:

1. формирование начальной популяции;
2. селекция особей;
3. скрещивание особей случайными значениями функции пригодности;

4. операция мутации над потомством;
5. проверка значения гена «конфликтов» и его ликвидация в случае необходимости;
6. формирование новой популяции;
7. проверка критерия остановки алгоритма;
8. выбор наилучшей особи.

1. Формирование начальной популяции

На первом этапе случайным образом формируется исходная популяция, состоящая из заданного числа N особей, где каждая особь популяции представляет собой отдельный вариант расписания (решение задачи).

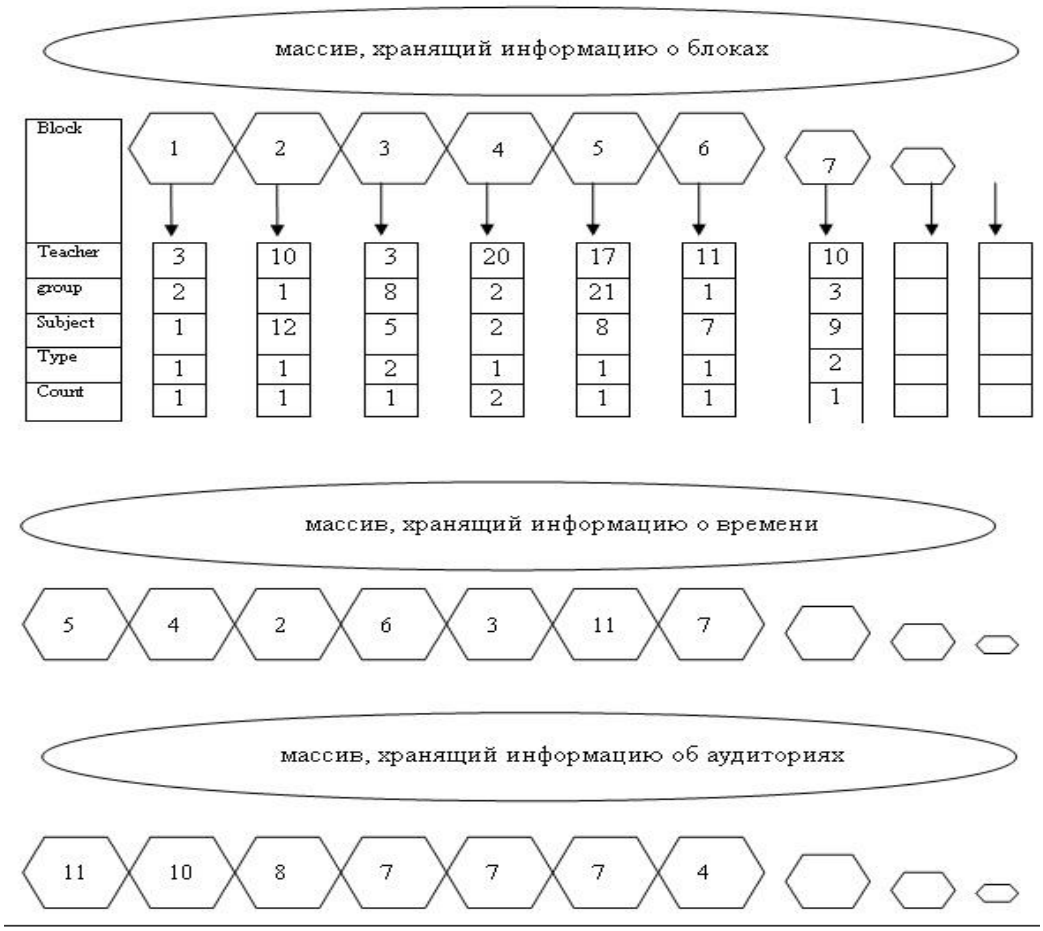


Рис. 2.7 Схема особи.

На рис. 2.7. изображен индивид, который состоит из 3 хромосом – аудитории, времени и блока. Каждая хромосома имеет массив длиной 1-N генов. Эти массивы хранят информацию о хромосомах.

Все особи популяции состоят из трех хромосом. Каждая хромосома особи в свою очередь состоит из генов, количество которых равняется количеству блоков занятий. Информационным наполнением первой хромосомы являются аудитории, для второй хромосомы – учебные нары (временные интервалы), для третьей хромосомы – идентификаторы блока.

Формирование каждой отдельной особи исходной популяции происходит с использованием случайного закона следующим образом: поочередно для каждого гена, условно обозначающего блок занятий, определяется некоторое значение. Для хромосомы аудиторий этим значением является номер аудитории из числа допустимых для проведения данного блока занятия. После чего осуществляется переход к следующему гену данной хромосомы. По достижению последнего гена первой хромосомы и определения его значения, таким же образом заполняются значения генов второй хромосомы. Каждому гену второй хромосомы из общего числа номеров времени, случайным образом присваивается номер пары, допустимый для данного блока занятия. После чего формирование второй хромосомы начальной популяции завершается. Аналогично происходит заполнение третьей хромосомы. Описанный процесс продолжается до момента достижения предельной численности популяции, определяемой в начале алгоритма[79].

В предлагаемом генетическом алгоритме не требуется кодирования варьируемых параметров в виде нулей и единиц как это принято в существующих генетических алгоритмах. Гены принимают целочисленные значения, которые кодируют номера аудиторий, номера времени, а также идентификаторы блоков.

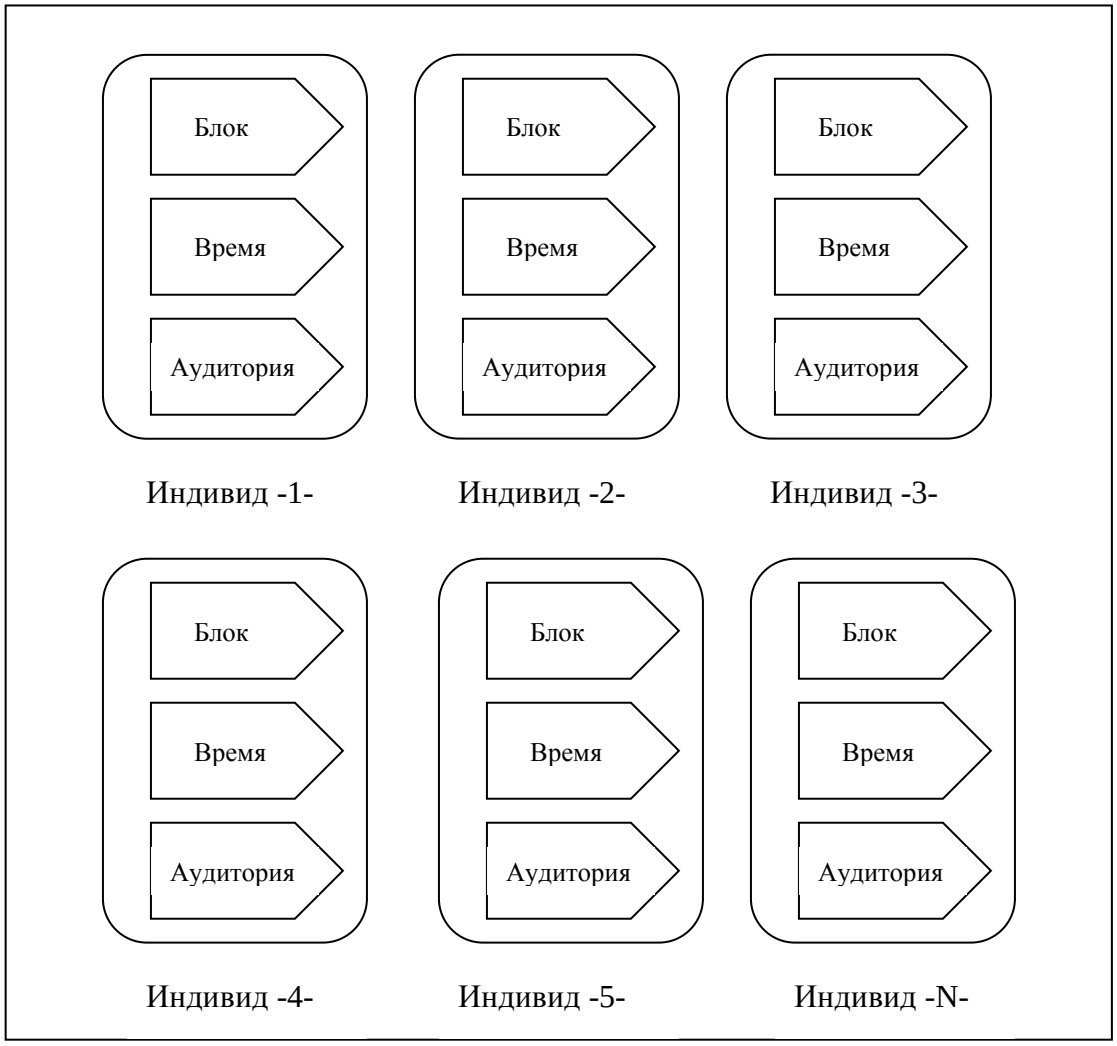


Рис. 2.8 Схема популяции.

2. Селекция особей.

На этапе происходит отбор (селекция) наиболее приспособленных особей (вариантов расписания), имеющих более предпочтительные значения функции пригодности по сравнению с остальными особями.

Оператор отбора является важнейшим фактором, влияющим на эффективность генетического алгоритма. Неудачный выбор метода селекции может привести к сужению области поиска, к потере наиболее приспособленных особей популяции. В данной работе для предотвращения потери лучших особей в качестве метода отбора предлагается использовать

метод, турнирного отбора, дополненный принципом элитизма. Принцип элитизма заключается в следующем. Из предыдущей популяции выбирается L отдельных особей (число L определено изначально, и имеет фиксированное числовое значение), имеющих максимальное значение функции пригодности[9].

3. Кроссинговер

Третьим этапом генетического алгоритма является скрещивание. На этом этапе основную функцию выполняет оператор кроссинговера. Это языковая конструкция, позволяющая на основе скрещивания хромосом родителей создавать хромосомы потомков. Так как структура оператора кроссинговера в основном определяет эффективность генетического алгоритма, то существует огромное число видов этого оператора. На этом этапе используется измененный кроссовер. Выбирается точка сечения, первая часть одного родителя копируется в первого потомка. Во вторую часть потомка копируются гены второго родителя. Если такие гены уже встречаются в потомке, то они пропускаются, а оставшуюся часть потомка дополняют генами первого родителя (рис.2). Заметим, что точки оператора кроссинговера определяются случайным образом. В рамках самого алгоритма это происходит следующим образом. Случайно выбираются две позиции гена от 1 до N. Такие преобразования происходят с двумя хромосомами в особи, третья хромосома-блок переформируется в зависимости от первых двух.

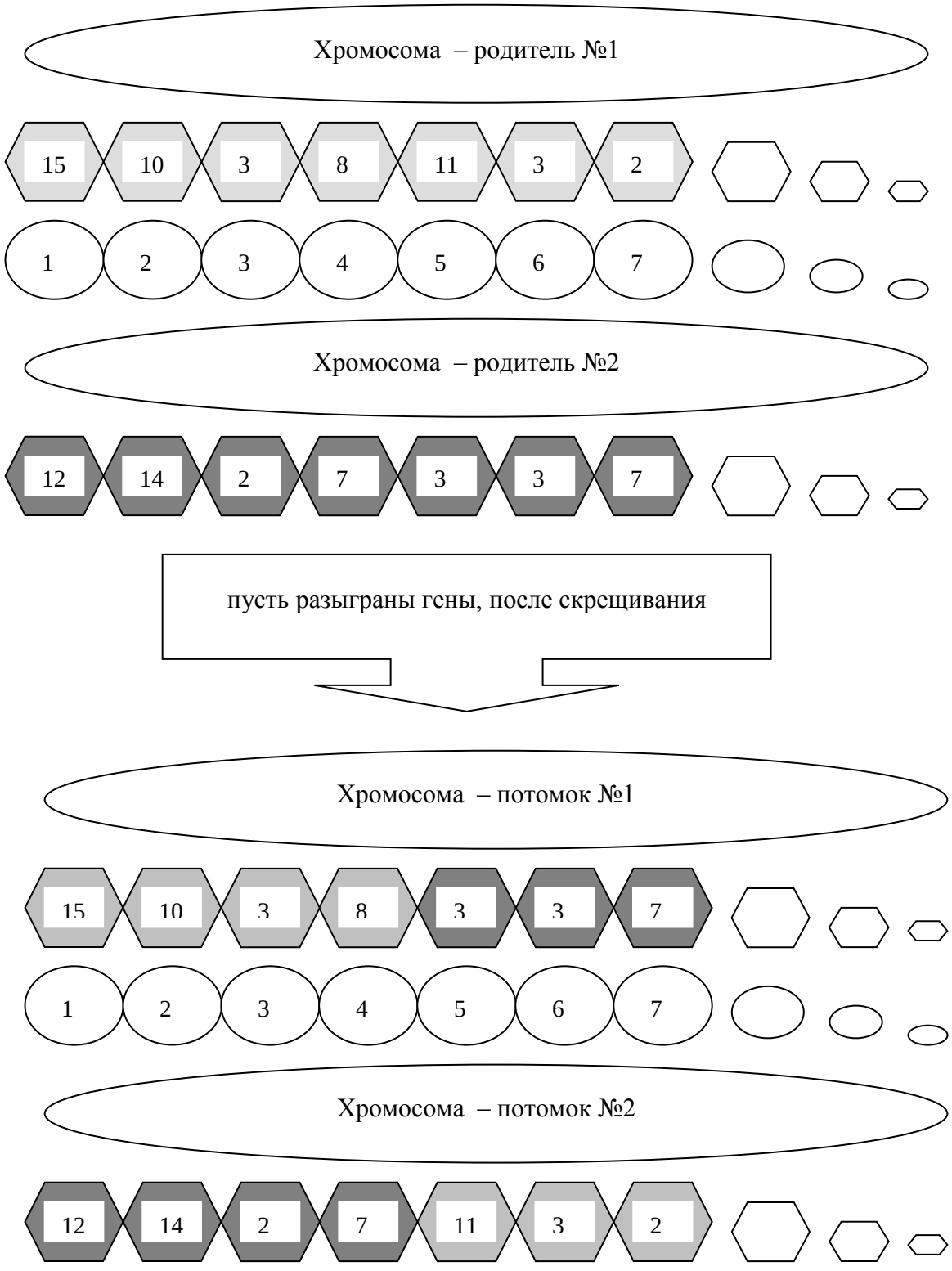


Рис. 2.9 Схема работы оператора кроссинговера на особи

4. Операция мутации

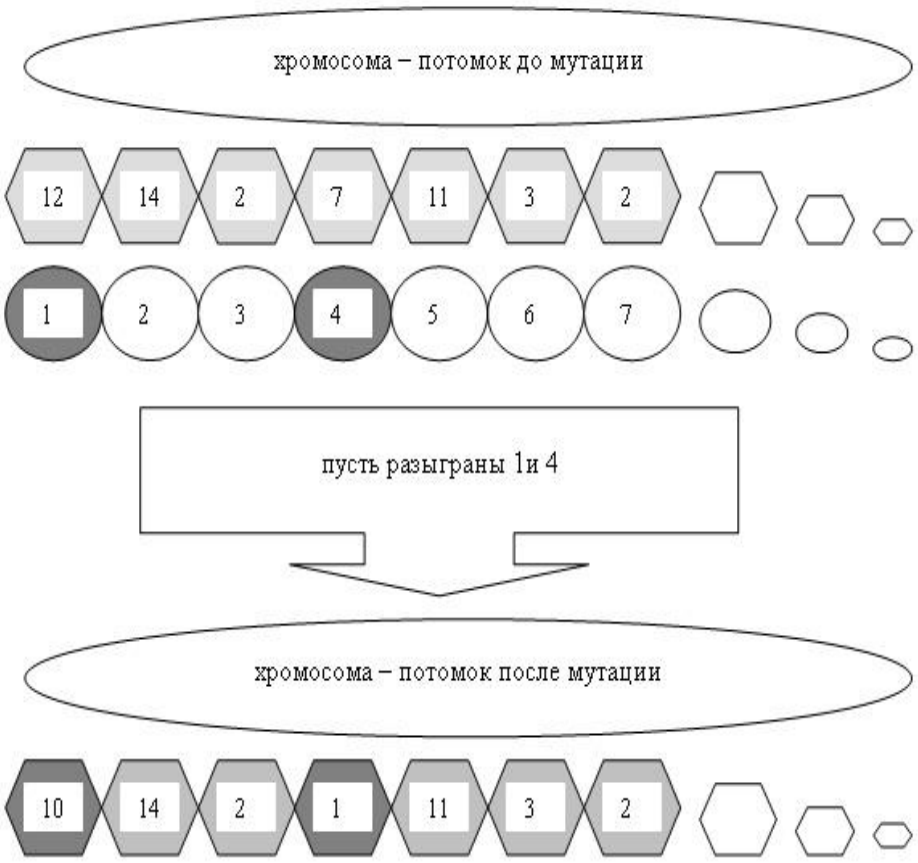


Рис. 2.10 Оператор мутации.

На рис 2.10, к некоторым из особей получившимся после скрещивания применяется оператор мутации. Мутация играет достаточно важную роль в работе генетического алгоритма: вносит дополнительное разнообразие в текущую популяцию и тем самым расширяет пространство поиска оптимального решения. В разработанном алгоритме оператор мутации с некоторой вероятностью изменяет значение нескольких генов в хромосомах некоторой "новой" особи на другие значения, входящие в число допустимых значений данного гена. Например, в результате мутации $i - \tilde{a}\hat{i}$ гена хромосомы аудиторий, где i определяет некоторое лекционное занятие, а значением этого гена (его информационным наполнением) является номер аудитории, предназначенной для проведения лекционных занятий, $i - \grave{i}\acute{o}$ гену будет присвоен номер аудитории, случайно выбранный из подмножества лекционных

аудиторий. Аналогично, для хромосомы временных интервалов в результате выполнения оператора мутации $j - му$ гену будет присвоен номер пары из допустимого подмножества пар, предназначенных для проведения именно данного вида занятия, Такая реализация оператора мутации позволяет избежать появления ошибок, связанных с несоответствие проводимого занятия типу аудитории, или наложению учебных пар[83].

5.На пятом этапе проверка значений функции конфликтов и его решение в случае необходимости.

Если этот ген не равен нулю, значит в хромосоме или одинаковым временным интервалам соответствует одна аудитория, или одной аудитории соответствует одинаковые временные интервалы, такое расписание содержит конфликт, который обязательно должен быть разрешен. Если у особи в хромосоме аудиторий ген «конфликтов» отличен от нуля, то есть одинаковые аудитории, отвечающие одинаковым временным интервалам, выбирается из базы данных другая аудитория с другим номером, а ген «конфликтов» становится на единицу меньше. Если у особи в хромосоме временных интервалов, ген «конфликтов» отличен от нуля, то есть одинаковые временные интервалы, отвечающие одной аудитории, следовательно из базы данных из массива временных интервалов выбирается свободный с другим номером и незанятый в этой аудитории, ген «конфликтов» уменьшается на единицу. Ген «конфликтов» позволяет сократить время проведения вычислительного эксперимента на порядок, то есть он ускоряет процесс сходимости алгоритма и получения квазиоптимального решения, рис. 2.11.

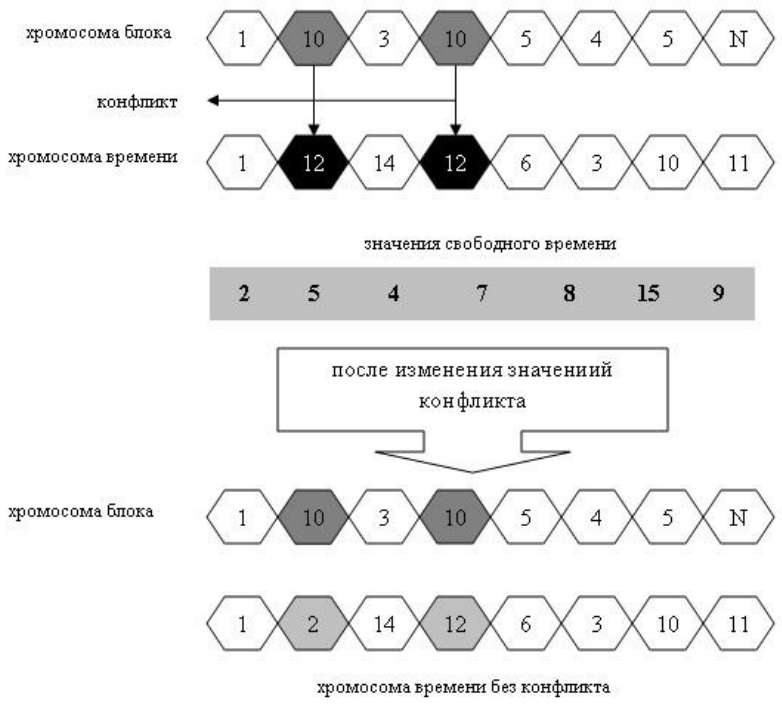


Рис. 2.11 Проверка значений функции конфликтов и его решение в случае необходимости

6. Формирование новой популяциия

Оператор отбора выполняет функции фильтрующего инструмента, который выделяет в составе популяции особи, имеющие низкое значение функции пригодности. Выделенные найденные слабые особи удаляются из популяции до тех пор, пока численность не становится исходной. Новое поколение, сформированное в результате применения операторов отбора, скрещивания, мутации, называемое популяцией потомков, заменяет родительскую популяцию.

7. Проверка условия прекращения работы алгоритма

На этом этапе происходит проверка условия прекращения работы алгоритма. Она основана на использовании приращения функции пригодности, т.е. если в течение нескольких поколений особей приращение значения функции пригодности наиболее "лучшего" индивидуума оказывается незначительным, то работу алгоритма завершается. Критерием остановки может

служить также число заданного количества итераций, либо значение гена «конфликтов», если конфликт присутствует либо по распределению аудиторий, либо по распределению времени значение этого гена равно 1, в противном случае 0. Если значение гена «конфликтов» отлично от нуля, то расписание содержит невыполненные ограничения и количество итераций в ГА увеличивается для получения бесконфликтного расписания.

Работа генетического алгоритма осуществлялось итерационно.

При выполнении заданного в алгоритме условия останова осуществляется переход к следующему этапу, в противном случае происходит переход к этапу селекции и процесс поиска оптимального решения продолжается.

8. Выбор «лучшего» решения На этом этапе среди полученных особей выбирается наилучшая особь, которая и будет являться решением задачи. Под лучшей особью понимается та, у которой значение функции пригодности является минимальным[77].

Использование гена времени и аудиторий (в зависимости от блока), использование блоков при реализации ГА позволяет съэкономить время при выполнении ГА и обеспечить реализацию ГА.

Одна итерация генетического алгоритма выглядит в виде восьми связанных этапов. Во время каждой итерации численность популяции увеличивается вдвое, а потом возвращается в исходное состояние, т.е. поиск оптимального решения производится в боле широком пространстве, нежели размеры самой популяции. Это во многом определяется выбором генетических операторов. Итерационный процесс генетического алгоритма продолжается до тех пор, пока не пройдет заданное число поколений или не выполнится какой-либо иной критерий останова.

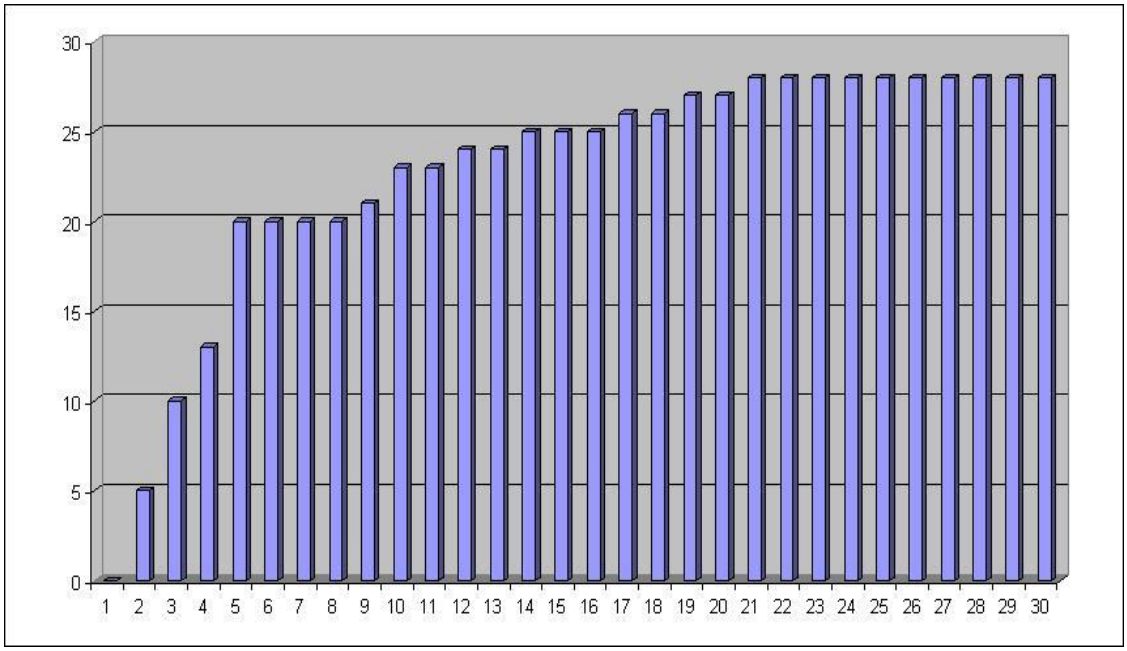


Рис. 2 .12 Сходимость генетического алгоритма

2.4. Результаты практического применения разработанной модели синтеза учебного плана

Описанный алгоритм был реализован с использованием языка C #.[93] Для отладки данного алгоритма использовались экспериментальные данные по факультету компьютерной науки вуза Ирака. В испытаниях принимало участие порядка ста предметов. Между описанными дисциплинами были определены связи, которые характеризуют очередность следования предметов друг за другом. Информация о дисциплинах и связях между ними хранится в базе данных, работа с которой реализуется через интерфейс. На рис 2.12 представлена форма обработки дисциплин, соответствующих специальности «компьютерной науки», где каждая строка является блоком.

Рисунок показывает работу генетического алгоритма, которая состоит из следующих шагов:

1. необходимо выбрать нужные условия из пяти возможных;
2. установить размер популяции (на рис. Это зеачение равно 1000);

3. определить количество повторов алгоритма (на рис. Данное знаачение равно 120, но получил результат раньше);
4. определить размер мутации (на рис. Это значение равно 0.01).

После заполнения всех данных алгоритм начинает работать. По истечении нескольких минут появляется результат составления расписания факультета. Для получения результата, представленного, потребовалось 5 минут.

Скорость сходимости данного алгоритма регулируется несколькими важными характеристиками. Одна из самых важных – численность популяции. На протяжении работы алгоритма («жизни популяции») ее численность не меняется. Это связано с операциями селекции, кроссинговера, мутации и естественного отбора. Это можно проследить на рис 2.12 с учетом того, что численность популяции должна оставаться равной ста[41].

представлен результат работы алгоритма – расписание. Оно состоит из даты, времени, групп, преподавателей, предметов, аудиторий и типа занятий. Это расписание составлено для одного факультета на один семестр, где каждая учебная неделя состояла из пяти дней по 3 пары в день. Для его составления были задействованы 8 групп, 18 преподавателей, 53 предмета, 7 аудиторий.

2.5. Выводы по второй главе

1. Построена математическая модель расписания. Описаны ограничения, налагаемые на расписание.
2. Для хранения информации, необходимой для синтеза расписания, разработана база данных.
3. Описан разработанный агрегативный генетический алгоритм .

4. Описанный алгоритм реализован на ЭВМ, и в заключительной части главы приведены некоторые результаты его работы.

Глава 3. Структура программного комплекса информационной системы в образовании

3.1 Графические диаграммы UML

Этап проектирования и моделирования при создании программного обеспечения занимает до 80% всего рабочего времени. Процесс правильного и грамотного проектирования непротиворечивой модели программного обеспечения является очень важным. Ошибку на этапе кодирования исправить гораздо легче, чем просчет при начальном проектировании. Внутренне несогласованная или неполная модель может привести к полной неудаче проекта, невозможности кодирования и интеграции всех разрабатываемых приложений между собой или во внешнюю среду [15-17].

При проектировании программного комплекса информационной системы в образовании были использованы возможности унифицированного языка моделирования UML и программный пакет Rational Rose для визуального объектно-ориентированного моделирования систем на основе графических диаграмм языка UML.

UML (Unified Modelling Language) – это язык визуального моделирования программного обеспечения, включающий в себя определенную систему обозначений (нотацию), которая предназначена для обозначения идей и решений, выполненных на этапе объектно-ориентированного анализа и проектирования. UML включает в себя набор диаграмм, которые позволяют создавать модели сложных программных систем.

Rational Rose – это программный пакет для визуального объектно-ориентированного моделирования систем на основе классов и их взаимодействия. Другими словами, это визуальный редактор, позволяющий моделировать программные системы любой сложности на основе графических

диаграмм языка UML. Rational Rose позволяет создавать модели будущей системы, удобные для понимания алгоритмов работы, взаимосвязей между объектами, по которым в дальнейшем создается программный каркас будущей программной системы. Сейчас моделирование – одно из средств, позволяющих значительно сократить время разработки, уложиться в бюджет и создать систему с нужным качеством. Модель будущей системы позволяет уже на стадии проектирования получить представление о поведении системы и избежать ошибок в дальнейшем, когда в написание кода вложены значительные силы. Rational Rose как инструмент моделирования выгодно отличается от других и позволяет просто и удобно создавать UML диаграммы, понятные любому разработчику информационных систем. Эти диаграммы позволяют создать при помощи Rational Rose исходный текст программы на различных языках программирования и облегчить тем самым разработчику рутинную работу – написание кода программы. [46]

Преимущества от применения Rational Rose:

- сокращение цикла разработки приложения “заказчик – программист – заказчик”. Теперь заказчику нет необходимости ждать первой демо-версии, чтобы убедиться, что все делается правильно;
- увеличение продуктивности работы разработчиков, уменьшение ручного труда при кодировании;
- способность вести большие проекты и группы проектов;
- возможность повторного использования уже созданного программного обеспечения;
- использование языка UML, служащего универсальным “мостиком” между разработчиками различных направлений.

В распоряжение проектировщика системы Rational Rose предоставляет следующие типы диаграмм, последовательное создание которых позволяет

получить представление обо всей проектируемой системе и об отдельных ее компонентах:

- Use case diagram (диаграммы сценариев);
- Deployment diagram (диаграммы топологии);
- Statechart diagram (диаграммы состояний);
- Activity diagram (диаграммы активности);
- Interaction diagram (диаграммы взаимодействия);
    - Sequence diagram (диаграммы последовательностей действий);
    - Collaboration diagram (диаграммы сотрудничества);
- Class diagram (диаграммы классов);
- Component diagram (диаграммы компонент).

Следует заметить, что для создания различных предметных областей общий порядок работы может несколько отличаться. Для создания нашей системы будем использовать описанный ниже порядок работы.

В первую очередь произведем анализ списка операций, которые будет выполнять система, и определим список объектов системы, которые данные функции выполняют. Таким образом, определим требования к системе и границы предметной области. Для этого будем использовать диаграммы Use case.

Далее определим поведение системы при помощи диаграмм Statechart diagram (диаграммы состояний) и Activity diagram (диаграммы активности).

Дальнейшая детализация взаимодействия будет производиться при помощи Sequence diagram (диаграммы последовательностей действий) и Collaboration diagram (диаграммы сотрудничества).

На основании производимых действий определим компоненты системы при помощи диаграммы компонентов (Component diagram).

Для нашей системы мы не будем проектировать Deployment diagram (диаграмма топологии). Дадим ей краткое описание и укажем причины, по которым не будем ее реализовывать в рамках нашей задачи.

Deployment diagram (диаграммы топологии). Этот вид диаграмм предназначен для анализа аппаратной части системы, то есть “железа”, а не программ. В прямом переводе с английского Deployment означает “развертывание”, но термин “топология” точнее отражает сущность этого типа диаграмм. Для каждой модели создается только одна такая диаграмма, отображающая процессоры, устройства и их соединения. Обычно этот тип диаграмм используется в самом начале проектирования системы для анализа аппаратных средств, на которых она будет эксплуатироваться. Для решения нашей задачи не требуется проектирование данного типа диаграмм, т.к. в нашем комплексе основная роль отводится алгоритмам и пакетам программ, а не физическим устройствам, роль которых для нас несущественна.

3.1.1 Определениетребований к системе

при помощи диаграммы Use Case

Use Case Diagram (диаграмма сценариев). Этот вид диаграмм позволяет создать список операций, которые выполняет система. Часто этот вид диаграмм называют диаграммой функций, потому что на основе набора таких диаграмм создается список требований к системе и определяется множество выполняемых системой функций. Каждая такая диаграмма – это описание сценария поведения системы, которому следуют действующие лица. Данный тип диаграмм используется при описании бизнес процессов автоматизируемой предметной области, определений требований к будущей программной системе. Отражает

объекты как системы, так и предметной области и задачи, ими выполняемые. [76]

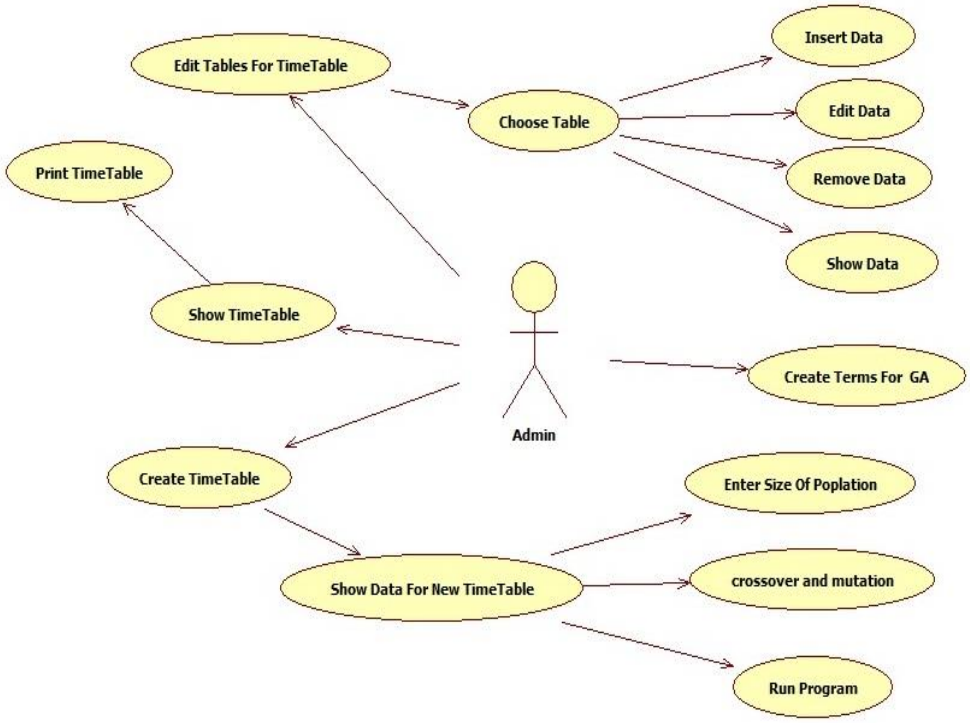


Рис. 3.1. Use Case diagram

В данной системе можно выделить одного актера – пользователя, который отдает команды системе.

- Edit Tables for Time Table: - редактирование сведоние о данных в системе.
    1- Choose Table: - выбор любую таблицу из таблиц системы.
    2- Add Data:- добавление новых данных в таблицу которая выбрана и сохранения их в базе данных
    3- Edit Data: - редактирование данных в таблицу которая выбрана и сохранения их в базе данных.
    4- Remove Data: - удаления данных из таблицы которая выбрана.
    5- Show Data: - вывод данных на экран из таблицы которая выбрана.
- Show Time Table:- - вывод данных на экран.

1- Print Time Table:- распечатать результаты работы системы.

- Create terms for GA: - Создание условий для ГА.

- Create Time Table:- Создание нового расписания.

6- Size Population: - Определение размера популяции.

7- Mutation: - Определение вероятности мутации.

8- Run Program: - запуск программы и получение нового расписания.

3.1.2 Machine Diagram (диаграммы состояний). Создание модели поведения системы при помощи диаграммы Statechart

Каждый объект системы, обладающий определенным поведением, может находиться в определенных состояниях, переходить из состояния в состояние, совершая определенные действия в процессе реализации сценария поведения объекта. Поведение большинства объектов реальных систем можно представить с точки зрения теории конечных автоматов, то есть поведение объекта отражается в его состояниях, и данный тип диаграмм позволяет отразить это графически. Для этого используется два вида диаграмм: Statechart diagram (диаграмма состояний) и Activity diagram (диаграмма активности).

Диаграмма состояний предназначена для отображения состояний объектов системы, имеющих сложную модель поведения. Диаграмма активности – это дальнейшее развитие диаграммы состояний. Фактически данный тип диаграмм может использоваться и для отражения состояний моделируемого объекта, однако, основное назначение диаграммы активности в том, чтобы отражать бизнес-процессы объекта. Этот тип диаграмм позволяет показать не только последовательность процессов, но и ветвление и даже синхронизацию процессов. Этот тип диаграмм позволяет проектировать алгоритмы поведения объектов любой сложности, в том числе может использоваться для составления блок-схем.

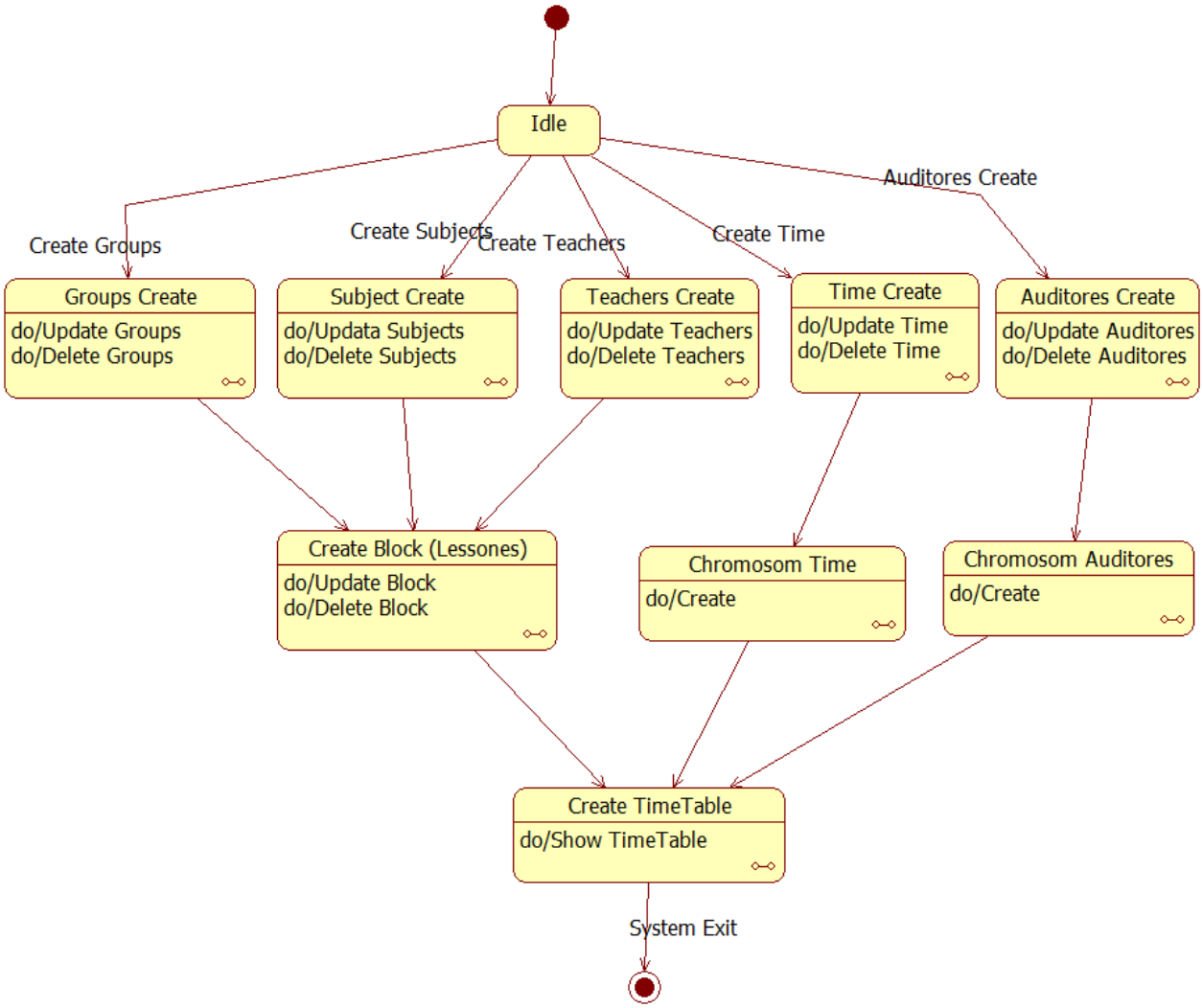


Рис. 3.2. Диаграмма состояния Statechart

Диаграмма Statechart предназначена для описания состояний объекта и условий перехода между ними. Описание состояний позволяет точно представить модель поведения объекта при получении различных сообщений и взаимодействии с другими объектами. Модель поведения позволяет взглянуть на получаемый программный объект со стороны, ведь основное назначение объектно-ориентированного программирования – создавать объекты, наделенные определенным поведением, которые в дальнейшем и будут производить работу в программном коде. А данный тип диаграмм позволяет

четко представить все поведение полученного программного объекта в виде графических значков состояний.

3.1.3 Описание взаимодействия при помощи Sequence diagram

Interaction diagram (диаграммы взаимодействия). Этот тип диаграмм включает в себя диаграммы Sequence diagram (диаграммы последовательностей действий) и Collaboration diagram (диаграммы сотрудничества). Эти диаграммы позволяют с разных точек зрения рассмотреть взаимодействие объектов в создаваемой системе.

Sequence diagram (диаграммы последовательностей действий). Взаимодействие объектов в системе происходит посредством приема и передачи сообщений объектами-клиентами и обработки этих сообщений объектами-серверами. При этом в разных ситуациях одни и те же объекты могут выступать и в качестве клиентов, и в качестве серверов.

Данный тип диаграмм позволяет отразить последовательность передачи сообщений между объектами. Этот тип диаграммы не акцентирует внимание на конкретном взаимодействии, главный акцент уделяется последовательности приема/передачи сообщений. Для того чтобы окинуть одним взглядом все взаимосвязи объектов, служит Collaboration diagram.

Продолжением создания нашей системы будет диаграмма взаимодействия между объектами (Sequence diagram). Кроме сценария поведения каждого объекта системы необходимо точно представлять взаимодействие этих объектов между собой и их обмен сообщениями. Он происходит в определенной последовательности, и Sequence diagram позволяет получить отражение этого обмена во времени.

На данном этапе мы проектируем взаимодействия объектов между собой путем обмена сообщениями. Здесь можно наглядно видеть связи между ними и роли, отведенные им в системе. В нашем случае система предоставляет

активный интерфейс пользователю, т.е. оператор занимает значимое место в системе.

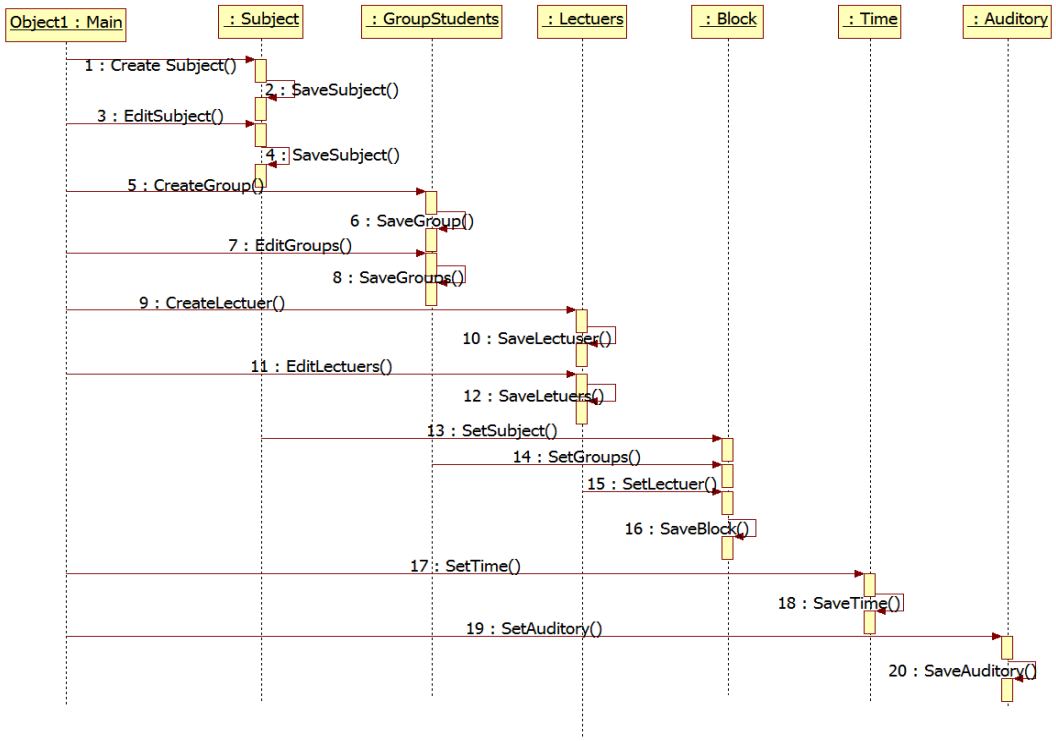


Рис. 3.3. диаграмма последовательностей для интерфейса.

Наша схема состоит из семи объектов. Оператор может редактировать эти объекты (рис. 3.3), в которых приняты обозначения:

CreateSubject() – добавить сведение о новом предмете. Добавление предмета является важной составляющей частью для составления расписания. Для каждой группы существует несколько предметов. Оператор должен имеет возможность добавить их в базу данных.

SaveSubject() – сохранить данные предметов в таблице предметов базы данных.

EditSubject() – редактировать сведения о предмете.

CreateGroup() – добавить сведение о новой группе студентов. При появлении новой группы оператор добавляет данные о вновь созданной группе в таблицу групп.

SaveGroup() – сохранить сведение о группе.

Edit Group() – редактировать сведение о группе.

CreateLectuer()-добавить сведение о новом преподавателе.

SaveLectuer() – сохранить сведение о преподавателе.

EditLectuer()- редактировать сведение о преподавателе.

SetSubjects()- добавление сведений о предмете в блок.

SetGroups()- добавление сведений о группе в блок.

SetLetcuers()- добавление сведений о преподавателе в блок.

SetBlocks()- добавление блока в таблицу. Каждый блок содержит информацию по предмету (практика или лекция), группе и преподавателю. За каждым преподавателем закреплены определенные предметы и занятия.

SaveBlock()- сохранить блок в базе данных.

SetTime()- после формирования блоков происходит добавление времени в произвольном порядке.

SaveTime()- сохранить время.

SetAuditory()- аналогично добавлению времени происходит добавление номеров аудиторий для каждого блока – в произвольном порядке.

SaveAudiory()- сохранить номер аудитории.

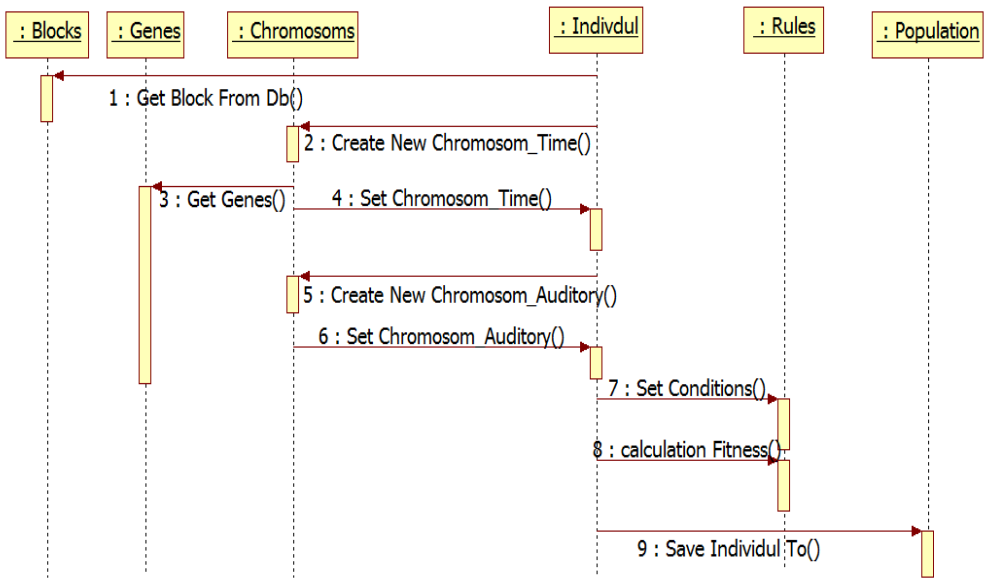


Рис .3.4. Диаграмма последовательностей для организвции генетического алгоритма (ГА)

На рис. 3.4. изображены 6 объектов со следующими функциями.

GetBlockFromDb()-эта функция сохранения всех блоков и их количество из базы данных в память компьютера на время работы программы. Это сделано для упрощения и ускорения работы с программой.

- CreatNewChromosom_Time()- после добавления всех блоков в память компьютера необходимо добавить ген времени для каждого блока. Это время выбирается случайно из базы данных. В результате получается хромосома времени равной количеству блоков.

- GetGenes()- данная операция уточняет равенство количества блоков и количества генов времени в хромосоме. Для каждого блока имеется ген времени.

- SetChromsmoe_time()- сохранение генов времени по порядку как массив в память компьютера, чтобы для каждого блока было свое время.

- CreatNewChromosom_Auditory()- после добавления всех блоков в память компьютера необходимо добавить ген аудитории для каждого блока.

Аудитория выбирается случайно из базы данных. В результате получается хромосома аудиторий по длине равная количеству блоков.

- SetChromsmoe_Auditory()- сохранение генов аудиторий по порядку как массив в память компьютера, чтобы для каждого блока была своя аудитория.

- Set Conditions()- создание двух типов условий – трудных и легких.

- Calculation fitness() - расчет несоответсвий в расписании, которые зависят от условий. Данная функция является самой важной в алгоритме, т.к. позволяет выявить наилучшее расписание. [100]

- Save Individual to()- после формирования расписания мы получаем индивид, который состоит из блоков, хромосомы времени, хромосомы аудиторий и значению функции (или приспособленности). Необходимо добавить этот индивид в популяции, чтобы сравнить с другими индивидами с помощью значения функции фитнеса.

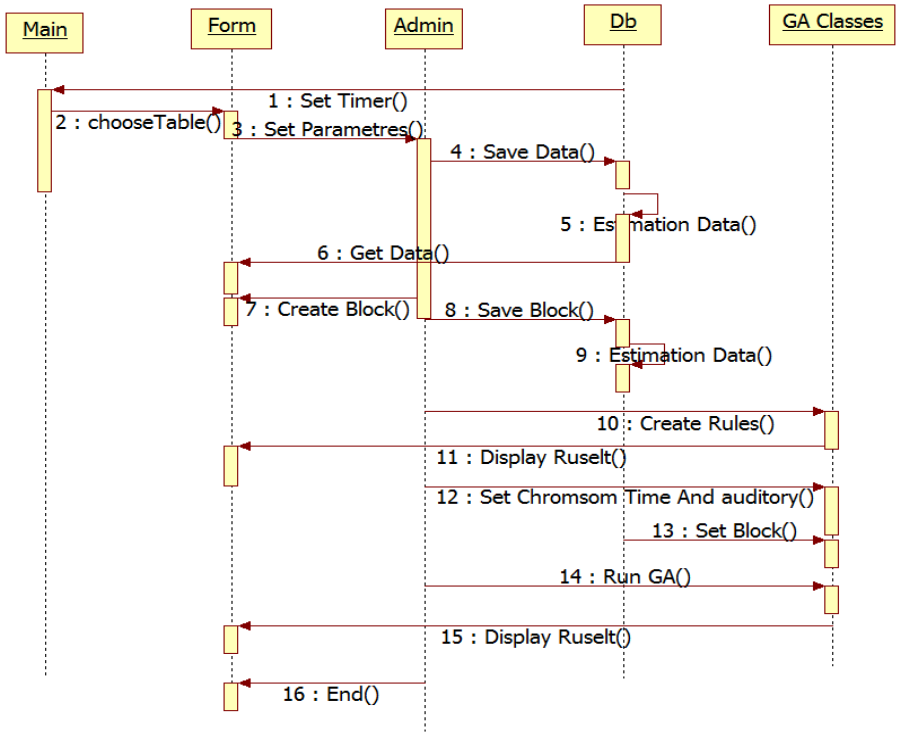


Рис.3.5 Диаграмма последовательностей для всей системы.

На рис. 3.5. изображены 5 объектов со следующими методами:

- SetTimer()-перед началом работы с системой оператор может установить на машине текущую дату, если она не соответствует действительности. Это может облегчить работу с программой, т.к. после запуска комплекса даты начала и конца периодов составления расписания будут предложены автоматически в зависимости от текущего числа. В противном случае, они не будут соответствовать реальной дате и оператору придется проделать несложную, но, возможно, лишнюю работу.

- ChooseTable()- оператор выбирает таблицы и заполняет их. В системе есть 5 таблиц. Оператор заполняет таблицы существующей информацией, которая совпадает с расписанием.

- SetParameters ()- возможно также редактирование пользователем основных констант, используя которые решается задача. После этого редактирования управление передается обработчику (Admin), который запускает алгоритм оценки введенных параметров.

- SaveData()- сохранить данные в базу данных.

- GetData()-после запуска комплекса, оператор может по желанию отредактировать информацию о требованиях и множествах ресурсов, хранимых в архивах базы данных.

- CreateBlock()-создать Блок занятий, представляет собой агрегированный объект, в него входят такие сущности как: Преподаватели,Учебные группы,Предметы,Вид занятий.

- CreateRules()- определение условия остановки генетического алгоритма зависит от его конкретного применения. В оптимизационных задачах, если известно максимальное (или минимальное) значение функции приспособленности, то остановка алгоритма может произойти после достижения ожидаемого оптимального значения, возможно - с заданной точностью. Остановка алгоритма также может произойти в случае, когда его

выполнение не приводит к улучшению уже достигнутого значения. Алгоритм может быть остановлен по истечении определенного времени выполнения либо после выполнения заданного количества итераций. Если условие остановки выполнено, то производится переход к завершающему этапу выбора «наилучшей» хромосомы. В противном случае на следующем шаге выполняется селекция.

- Set Chromosome Time and Chromosome Auditory ()- этот процесс создания гена времени и гена аудитории случайно из данных, содержащихся в этих таблицах. Каждый блок будет иметь ген времени и ген аудитории. Эти гены должны совпадать с условиями алгоритма, в противном случае алгоритм должен создать новые гены.
- Set Block()- это процесс сохранения всех блоков с генами времени и аудитории и значением функции фитнеса в популяции для каждой таблицы, для того, чтобы сравнить таблицы по значению функции фитнеса.
- RunGa()- начало работы генетического алгоритма подразумевает последовательное выполнение некоторых операций. Основными операциями являются Селекция, Кроссовер, Мутация и Репродукция.
- Display Result ()-оператор получает с внешнего накопителя печатную форму с результатами решения задачи.
- End()- происходит корректное закрытие связей с базой данных, сохранение результатов из оперативной памяти программы и завершение работы программного комплекса.

3.1.4 Диаграмма классов

Программа, реализующая генетический алгоритм выполнена в идеологии объектно-ориентированного программирования, следовательно она состоит из классов, диаграмма классов для системы состоит из трех диаграмм:

1- Диаграмма классов для объектов базы данных состоит из 7 классов[75].

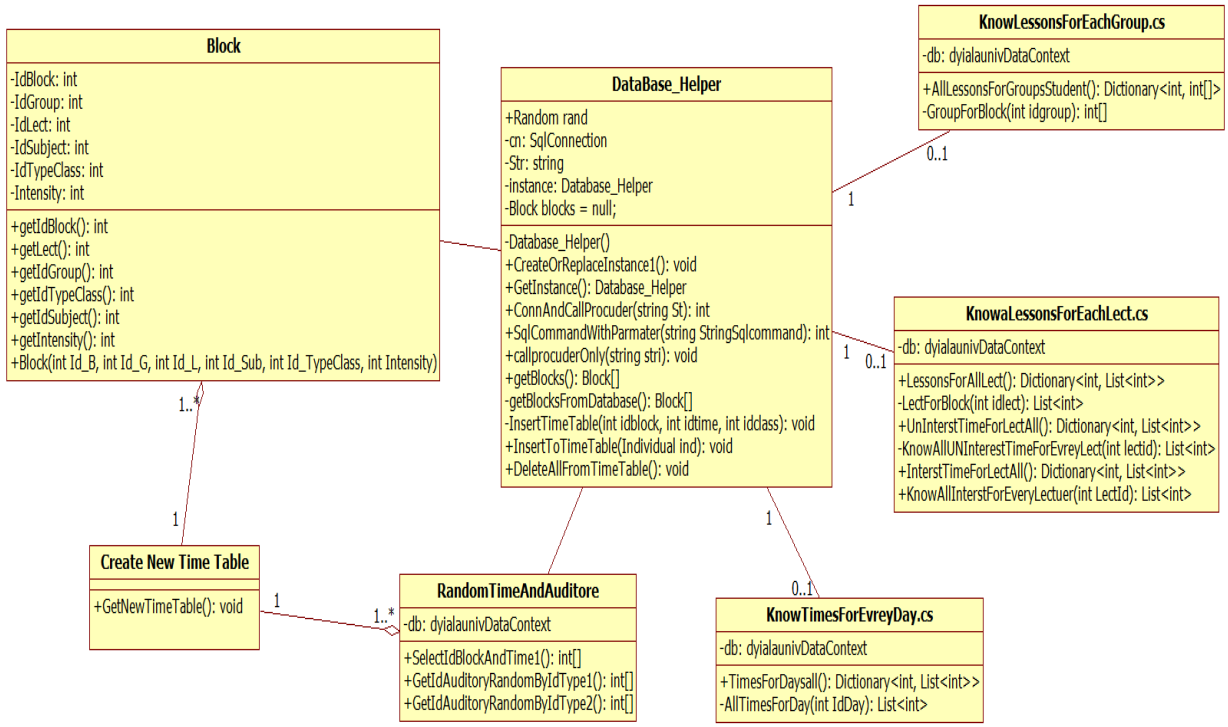


Рис. 3.6 Диаграмма классов для объектов базы данных.

Класс Block- блок занятий представляет собой агрегированный объект, в него входят следующие атрибуты и методы.

Атрибуты:

IdBlock - номер блока присваивается автоматически;

IdGroup - номер группы берется из таблицы группа;

IdLectuer - номер преподавателя берется из таблицы преподаватель;

IdSubject - номер предмета берется из таблицы предмета;

IdTypeLesson - тип занятия: лекция или практика;

Interstiy- количество занятий в неделю.

Методы:

GetIdBlock()-получения и установка номера блока в метод блока;

GetIdGruop()- получения и установка номера группы в метод блока;

GetIdLectuer()- получения и установка номера преподавателя в метод блока;

GetIdSubject()- получения и установка номера предмета в метод блока;

GetIdTypeLesson- получения и установка тип занятия в метод блока ;

GetInterstiy()- получения и установка количество занятий в метод блока;

Block(int Id_B, int Id_G, int Id_L, int Id_Sub, int Id_TypeClass, int Intensity)- сохранение данных значений в базе данных.

Класс DataBase_Helper - этот класс для соединения с базой данных

Данный класс имеет следующие атрибуты и методы:

Атрибуты:

- Cn: SqlConnction- атрибут открытия и закрытия соединения с базой данных.

Методы:

-GetBlocksFromDatabase(): Block[]- функция, которая помогает системе получить все блоки и составить их в массив, чтобы добавить для кажого блока время и аудиторию;

- InsertTimeTable(int idblock, int idtime, int idclass): void -функция ввода и сохранения данных в таблицу расписание. Эти данные состоят из номера блока, номера времени и номера аудитории;

- InsertToTimeTable(Individual ind) :void - функция ввода наилучшего индивида в таблице расписания и удаления старого расписания в этой же таблице. Если значение фитнеса нового расписания имеет малое количество несовпадений, то оно сохраняется вместо старого;

- DeleteAllFromTimeTable():void- функция удаления старого расписания.

Класс KnowLessonsForEachGroup - имеет следующие атрибуты и методы:

Атрибуты:

- DyialaunivDataContext :db- атрибут соединения с базой данных.

Методы:

- AllLessonsForGroupsStudent(): Dictionary<int, int[]>- функция вызова всех занятий для каждой группы студентов и сохранения их в массиве, чтобы

выявить совпадения занятий для каждой группы, проходящих в одно время в одной аудитории.

Класс KnowaLessonsForEachLect.cs- данный класс имеет следующие атрибуты и методы.

Атрибуты:

- DyialaunivDataContext:db - атрибут соединения с базой данных.

Методы:

- LessonsForAllLect(): Dictionary<int, List<int>>- функция вызова всех занятий для каждого преподавателля и сохранения их в массиве, чтобы выявить совпадения занятий для каждого преподавателя, проходящих в одно время в одной аудитории;
- UnInterstTimeForLectAll(): Dictionary<int, List<int>>- одним из условий является время проведения занятий для преподавателей. Система должна выполнить это условие, поэтому эта функция вызова всего нежелательного для преподавателя времени и сохранение этого времени в массив с номером преподавателя для сравнения нежелательного времени с расписанием. На этом этапе отслеживается количество несовпадений;

-InterstTimeForLectAll():Dictionary<int, List<int>>- функция вызова желательного времени для преподавателя и сохранение этого времени в массив с номером преподавателя для сравнения желательного времени с расписанием.

Класс knowTimesAndAuditoryForEvreyDay.cs- имеет следующие атрибуты и методы.

Атрибуты:

- DyialaunivDataContext:db- атрибут соединения с базой данных.

Методы:

- TimesForDaysall(): Dictionary<int, List<int>>- функция, которая помогает системе выявить все время занятий в день, чтобы узнать, есть ли накладка

времени в этот же день для каждой группы и каждого преподавателя. Если есть накладка, нужно поменять это время на другое случайно из таблицы времени;

- AuditoryForDays(): Dictionary<int,List<int>>- функция, которая помогает системе выявить все номера аудиторий в день, чтобы узнать, есть ли накладка номера аудитории в этот же день для каждой группы и каждого преподавателя. Если есть накладка, нужно поменять этот номер аудитории на другой, случайно из таблицы аудитории.

Класс randomTimeAndAuditore.cs- имеет следующие атрибуты и методы.

Атрибуты:

- DyialaunivDataContext:db - атрибут соединения с базой данных.

Методы:

- SelectTimeFromDataBase(): int[]- функция помогает системе вызвать все время из таблицы времени и сохранить его в массив в памяти компьютера, чтобы система смогла соединяться с этим временем;
- GetIdAuditoryRandomByIdType1(): int[]- функция помогает системе вызвать все аудитории для лабораторных работ из таблицы аудиторий и сохранить их в массив в памяти компьютера, чтобы система смогла соединяться с этими аудиториями;
- GetIdAuditoryRandomByIdType2(): int[]- функция помогает системе вызвать все аудитории для практических занятий из таблицы аудиторий и сохранить их в массив на память компьютера, чтобы система смогла соединяться с этими аудиториями.

Класс Delete And Create New Time Table cs- имеет следующие методы:

- GetSetTimeTable(): void- функция, которая сохраняет самое хорошее расписание в базу данных, выбранное из популяции с помощью значения функции фитнеса.

2- Диаграмма классов для объектов генетического алгоритма состоит из 12 классов.

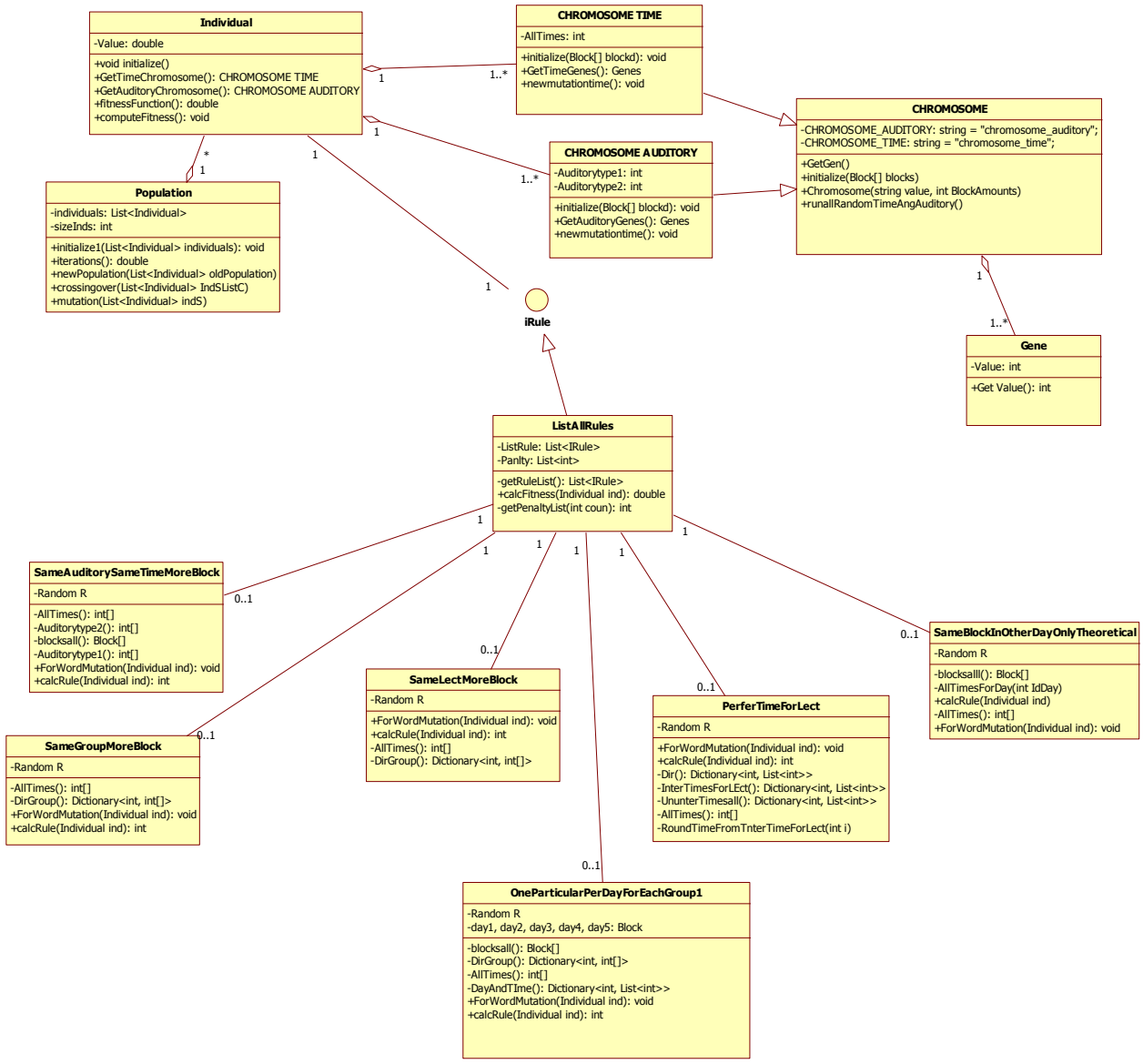


Рис. 3.6 Диаграмма классов для генетического алгоритма.

1- Класс Gen: это значение в хромосоме, имеет метод:

- GetValue():int- метод, которой получает ген и возвращает указатель на массив хромосом.

2- Класс Chromosome – имеет интерфейсную часть.

Класс хромосом имеет две хромосомы – хромосома времени и хромосома аудитории и имеет следующие атрибуты и методы:

Атрибуты:

-CHROMOSOME_AUDITORY: string = "chromosome_auditory":Значение string имеет значение хромосомы аудитории,

- CHROMOSOME_TIME: string = "chromosome_time":Значение string имеет значение хромосомы времени.

Методы:

- GetGen() -возвращает указатель на массив хромосом;

- initialize(Block[] blocks)- метод инициализации блоков;

-Chromosome(string value, int BlockAmounts)- функция получения типа хромосомы и количества блоков;

- runallRandomTimeAngAuditory()- метод, который помогает создать новые гены случайно для каждой хромосомы[70].

- **Класс хромосом CHROMOSOME TIME-** эта хромосома времени, данный класс имеет следующие атрибуты и методы.

Атрибуты:

- AllTimes:int[] - масстив времени.

Методы:

- initialize(Block[] blockd) void- функция инициализации блоков;

- GetTimeGenes() Genes- Метод, которой помогает получить все временные гены;

- newmutationtime()void- мутаци, метод изменения значения гена в хромосоме времени случайно из таблицы времени.

- Класс хромосом CHROMOSOME AUDITORY- эта хромосома аудитории, данный класс имеет следующие атрибуты и методы.

Атрибуты:

- Auditorytype1: int []- массив, который заполнен значениями для аудиторий лабораторных занятий;

- Auditorytype2: int[]- массив, который заполнен значениями для аудиторий практических занятий.

Методы:

- initialize(Block[] blockd)- void:инициализация блоков;

- GetAuditoryGenes() Genes- метод, которой помогает получить все гены аудиторий;

- newmutationtime() void- мутация,метод изменения значения гена в хромосоме аудиторий случайно из таблицы аудиторий.

3- Класс Individual- имеет следующие атрибуты и методы:

Атрибуты:

- Value:double- значение, которое имеет количество несовпадений для индивида.

Методы:

- void initialize()- создание новых генов в случайном порядке для каждой хромосомы;

- GetTimeChromosome() CHROMOSOME TIME- метод вызова всех генов времени;

- GetAuditoryChromosome() CHROMOSOME AUDITORY: метод вызова всех генов аудитории,

- fitnessFunction() double- метод функции фитнеса, которой осуществляет подсчет несовпадений для индивида;

- computeFitness() void - сравнение нового индивида с остальными с помощью значения функции фитнеса.

4- Класс Population- имеет следующие атрибуты и методы:

Атрибуты:

- individuals: List<Individual>- список, состоящий из типов индивида, служит для сохранения всех индивидов;

sizeInds: int- значение размера популяции.

Методы:

- initialize1(List<Individual> individuals) void- инициализация всех индивидов по порядку с помощью фитнеса. Выбор наилучшего индивида и сохранение его в базе данных, удаление старого индивида;

- iterations() double- запуск алгоритма, который состоит из селекции, кроссовера и мутации;

- newPopulation(List<Individual> oldPopulation)- селекция – это выбор половины индивидов из популяции, имеющих наименьшее число несовпадений;

- crossingover(List<Individual> IndSListC)- кроссовер – это скрещивание для получения новых индивидов;

- mutation(List<Individual> indS)- мутация – это изменение одного гена в каждой хромосоме случайно.

5- Класс interface ListAllRules- имеет следующие атрибуты и методы:

Атрибуты:

- ListRule: List<IRule>- массив состоит из условий;

- Panlty: List<int>- массив для сохранения несовпадений для каждого условия.

Методы:

- getRuleList(): List<IRule>- инициализация условий;

- calcFitness(Individual ind)- double метод подсчета несовпадеий для каждого индивида и сохранение значение в индивиде;

- getPenaltyList(int coun)- int метод общего числа несовпадений для каждого индивида.

-Класс интерфейс имеет 6 перечисленных классов. Каждый класс поддерживает какое-то ограничение для функции фитнеса.

1-Класс SameAuditorySameTimeMoreBlock.cs- поддерживает ограничение, что в одной аудитории проводится одна лекция в одно время и имеет следующие атрибуты и методы:

Атрибуты:

- Random R - создание нового значения случайно.

Методы:

- ForWordMutation(Individual ind)- void метод поиска гена, который имеет несовпадения, и поменяет его на другой ген случайно;
- calcRule(Individual ind)- int метод подсчета количества несовпадений для этого условия в данном индивиде;
- AllTimes(): int[]- инициализация всего времени;
-Auditorytype2(): int[]- инициализация всех аудиторий для практических занятий;
-Auditorytype1(): int[]- инициализация всех аудиторий для лабораторных занятий;
- blocksall(): Block[]- инициализация блоков.

2-Класс SameGroupMoreBlock- задает ограничения, что у каждой группы есть одна лекция в одно время и имеет следующие атрибуты и методы:

Атрибуты:

- Random R - создание нового значения случайно.

Методы:

- AllTimes(): int[]- инициализация всего времени;
- calcRule(Individual ind): int- метод подсчета количества несовпадений для этого условия в данном индивиде;
- ForWordMutation(Individual ind): void- метод поиска гена, который имеет несовпадения, и поменяет его на другой ген случайно;
- DirGroup(): Dictionary<int, int[]>- инициализация всех занятий для каждой группы.

3-Класс SameLectMoreBlock- поддерживает ограничения, что каждый преподаватель проводит одну лекцию в одно время, данный класс имеет следующие атрибуты и методы:

Атрибуты:

- Random R- инициализация значения случайно.

Методы:

- AllTimes(): int[]- нициализация всего времени;

- calcRule(Individual ind):int- метод подсчета количества несовпадений для этого условия в данном индивиде.

- ForWordMutation(Individual ind): void- метод поиска гена, который имеет несовпадения, и поменяет его на другой ген случайно;

- DirLect(): Dictionary<int, int[]>- инициализация всех занятий для каждого преподавателя.

4-Класс PerferTimeForLect- этот класс поддердивает ограничения, что каждый преподаватель может обозначить время для проведения занятий, данный класс имеет следующие атрибуты и методы:

Атрибуты:

- Random R - инициализация значения случайно.

Методы:

- ForWordMutation(Individual ind): void - метод поиска гена, который имеет несовпадения, и поменяет его на другой ген случайно;

- AllTimes(): int[]- инициализация всего времени;

- calcRule(Individual ind): int- метод подсчета количества несовпадений для этого условия в данном индивиде;

- Dir(): Dictionary<int, int[]>- инициализация всех занятий для каждого преподавателя;

- InterTimesForLEct(): Dictionary<int, List<int>>- инициализация всего наилучшего времени для каждого преподавателя;

- UnunterTimesall(): Dictionary<int, List<int>>- инициализация всего наихудшего времени для каждого преподавателя;

- RoundTimeFromTnterTimeForLect(int i)- создание гена времени случайно из наилучшего времени и замена наихудшего.

5-Класс OneParticularPerDayForEachGroup1.cs- поддерживает ограничения, что в расписании предусматривается одно практическое занятие в день и имеет следующие атрибуты и методы:

Атрибуты:

- Random R- инициализация значения случайно;
- day1, day2, day3, day4, day5: Block[] :5 значений, каждое значение является массивом блока. Каждому значению соответствует день занятия.

Методы:

- AllTimes(): int[] - инициализация всего времени;
- blocksall(): Block[]- инициализация всех блоков;
- DirGroup(): Dictionary<int, int[]>- инициализация всех занятий для каждой группы;
- DayAndTIme(): Dictionary<int, List<int>>- инициализация всего времени занятий для каждого дня и сохранение в массиве;
- ForWordMutation(Individual ind): void- метод поиска гена, который имеет несовпадения, и поменяет его на другой ген случайно;
- calcRule(Individual ind): int- метод подсчета количества несовпадений для этого условия в данном индивиде.

6-Класс SameBlockInOtherDayOnlyTheoretical- ограничение на пары, данный класс имеет следующие атрибуты и методы:

Атрибуты:

- Random R - инициализация значения случайно .

Методы:

- AllTimes(): int[]- инициализация всего времени;

blocksall(): Block[]- инициализация всех блоков;

AllTimesForDay(int IdDay)- инициализация всего времени блоков по дням;

- ForWordMutation(Individual ind):void- метод поиска гена, который имеет несовпадения, и поменяет его на другой ген случайно;

- calcRule(Individual ind): int -метод подсчета количества несовпадений для этого условия в данном индивиде.

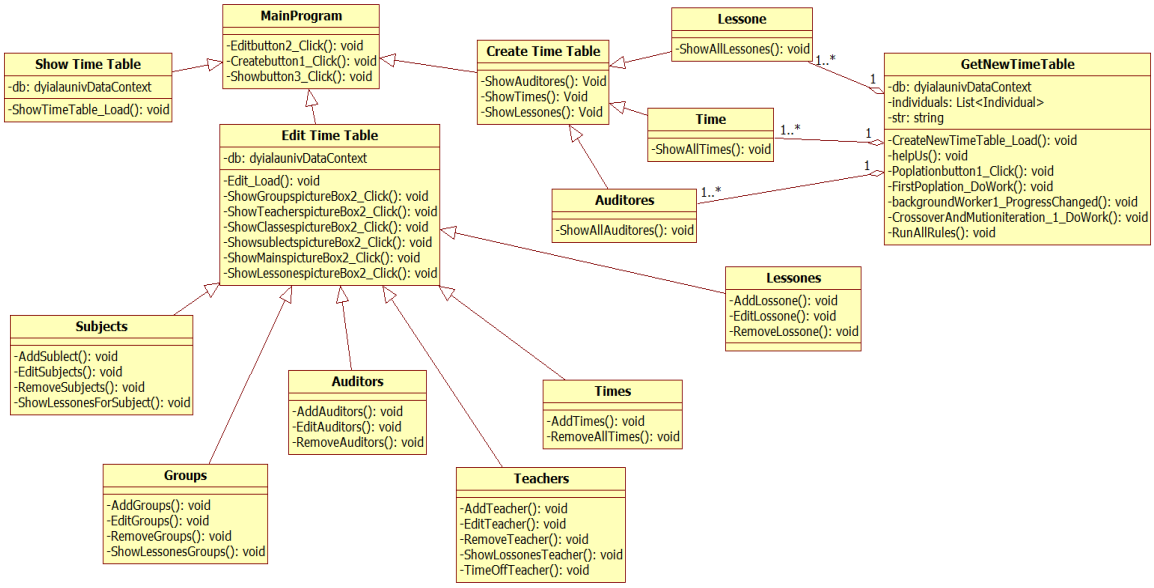


Рис. 3.7. Диаграмма классов для разработки интерфейса

В данной диаграмме приняты следующие обозначения:

- **Класс MainProgram-** реализует главное окно интерфейса.

- Editbutton2_Click() void- ректирование таблиц;

- Createbutton1_Click() void- добавление новых данных;

- Showbutton3_Click() void- вывод данных на экран;

- **Класс Show Time Table-** данный класс имеет следующие атрибуты и методы.

Атрибуты:

- db: dyialaunivDataContext- соединения с базой данных.

Методы:

- ShowTimeTable_Load()void- вывод общих данных таблицы на экран.

- **Класс Create Time Table-** создание расписания, для того, чтобы создать расписание, нужно выяснить, заполнены ли таблицы.

- ShowAuditores() Void- вывод данных таблицы аудиторий;

- ShowTimes() Void-вывод данных таблицы времени;

- ShowLessones() Void-вывод данных таблицы блоков.

- **Класс GetNewTimeTable-** получение расписания, данный класс имеет следующие атрибуты и методы.

Атрибуты:

- db: dyialaunivDataContext - функция соединения с базой данных;

- individuals: List<Individual>-массив имеет все индивиды.

Методы:

- FirstPoplation_DoWork() void- запуск программы и получение нового расписания;

- CrossoverAndMutioniteration_1_DoWork() void- запуск алгоритма(селекция, кроссовера , мутации);

- CreateNewTimeTable_Load(): void- сохранение самый хороший индивид в базу данных.

- **Класс Edit Time Table** – реализует редактирование таблиц, данный класс имеет следующие атрибуты и методы.

Атрибуты:

- db: dyialaunivDataContext - соединения с базой данных.

Методы:

- Edit_Load() Void - редактирование таблиц.

- **Класс Subjects -** предметы, класс имеет следующие методы.

- AddSublect() void- добавление новых данных;

- EditSublect() void:- редактирование данных;

- Remove Sublect() void- удаления данных;

- ShowSublect() void- вывод данных на экран.

- **Класс Auditors-** аудитории, класс имеет следующие методы.

- AddAuditor() void- добавление новых данных;

- EditAuditor() void- редактирование данных;

- Remove Auditor() void- удаления данных.

- **Класс Groups**- группы, класс имеет следующие методы.
- AddGroup () void- добавление новых данных;
- EditGroup () void- редактирование данных;
- RemoveGroup () void- удаления данных;
- ShowLessonesGroup() void: вывод данных на экран.
- **Класс Teachers**- преподавател, класс имеет следующие методы.
- AddTeacher () void- добавление новых данных;
- EditTeacher () void- редактирование данных;
- RemoveTeacher () void- удаления данных;
- ShowLossonesTeacher() void- вывод данных на экран;
- **Класс Times**- время, класс имеет следующие методы.
- AddTimes()void- добавление новых данных;
- RemoveTime()void- удаления данных.
- **Класс Blocks** - имеет следующие методы.
- AddLessone () void- добавление новых данных;
- EditLessone () void- редактирование данных;
- RemoveLessone () void- удаления данных. [117]

3.2 Информационные потоки данных

3.2.1 Описание работы алгоритма.

Для интерфейса:

1. Инициализация

Первоначальная инициализация особей проходит в 2 этапа.

1 этап). Построение допустимых значений, и состоит из следующих пунктов

1. Выборка из базы данных ID аудиторий по типам: лекционные, практические, (по всем уникальным значениям таблицы AudType ID)
2. Сохранение полученных данных в коллекции LectionAuds, PracticalAuds, Эти коллекции содержат допустимые значения для Хромосомы.Аудитории.
3. Выборка всех Time.ID из базы Times, сохранение их в таблице TimeData. Т.о. таблица TimeData содержит все возможные пары за одну неделю.
4. В массив BlockInfo[] Blocks {id_block,id_group,id_subject,id_teacher,id_Type_Lectuer} выбираются соответствующие столбцы из таблицы Block БД. Информация массива Blocks в дальнейшем используется для записи в БД конечного результата;

Хранение данных в оперативной памяти целесообразно, поскольку они выбираются единственный раз и используются для всех особей на каждом этапе работы алгоритма (инициализация, скрещивание, мутация, селекция)[49].

2 этап) Инициализация:

1.Создание n экземпляров класса Особь, где n – указанный в настройках алгоритма размер популяции.

2.Выборка m - общего числа блоков занятий из таблицы Blocks.

3.Вызов конструктор класс Особь с параметром m

4.Конструктор класса Особь создает экземпляр класса Хромосома.Аудитории и Хромосома.Время, каждая из которых имеет m экземпляров класса Ген

5.Конструктор класса Хромосома.Аудитории создает массив из m генов. Номер гена в массиве соответствует номеру блока занятий.

6. Инициализация генов класса Хромосома.Аудитории – по номеру гена (номеру блока занятий) из массива BlockAudtypes – тип подходящей аудитории, по типу подходящей аудитории выбирается таблица с номерами аудиторий. Из этой таблицы случайным образом выбирается номер подходящей аудитории. Если значение соответствующего значения BlockIntencity 1, то для следующего гена из Хромосомы.Время выполняется принудительное назначение того же времени, что и у предыдущего.

7. Инициализция генов класса Хромосома.Время – каждому гену присваивается случайное значений из таблицы TimeData.

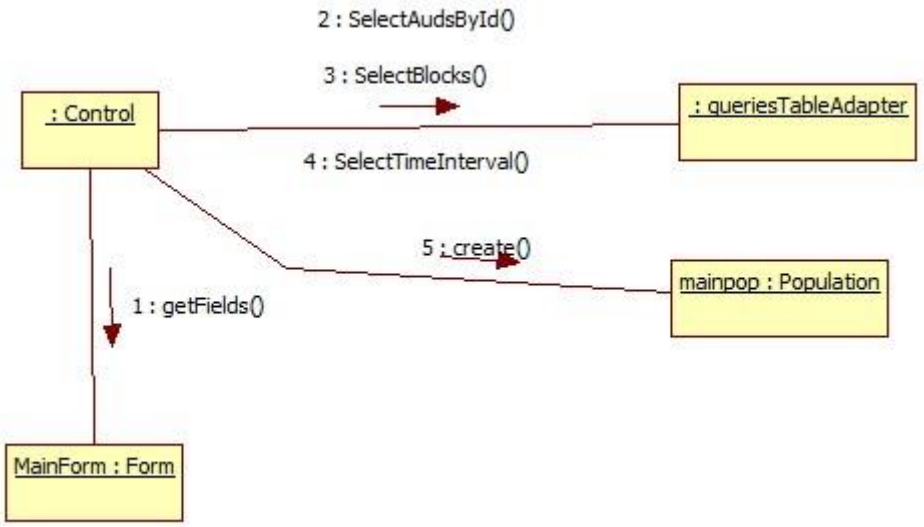


Рис 3.9. . Инициализация.

2- Create the First Population:- на первом этапе случайным образом формируется исходная популяция,состоящая из заданного числа M особей, где каждая особь популяции представляет собой отдельный вариант расписания (решение задачи).

3- Conditions:- определение условия остановки генетического алгоритма зависит от его конкретного применения.

4-Select:- селекция особей На этапе происходит отбор (селекция) наиболее приспособленных особей (вариантов расписания), имеющих более предпочтительные значения функции пригодности по сравнению с остальными особями.

5- Crossover:- (Скрещивание) – скрещивает родителей, чтобы получить нового потомка. Скрещивание происходит с определенной вероятностью. Если скрещивание не выполнилось, то потомок – это точная копия одного из родителей.

6- Mutation:- (Мутация)- изменяет несколько отличительных признаков нового потомка в локусе (участок хромосомы) с определенной вероятностью. Если мутации не произошло, потомок является прямым результатом скрещивания, или копией одного из родителей.

3.2.2 Схема работы программы.

Дадим описание взаимодействия объектов программы между собой и с конечным пользователем.

На жестком диске (или любом другом внешнем носителе) хранится информация, которая когда–либо была введена пользователем с помощью программного комплекса в базу данных (БД). Пользователь вносит информацию в БД только в основные таблицы через программный модуль с помощью пользовательского интерфейса. У нас этот блок обозначен как

“редактирование основных таблиц БД”. Также пользователь имеет право изменять общие константы расписания, используемые модулями решения задачи с помощью соответствующего блока и управлять выполнением программных модулей, т.е. запускать решение задачи, когда ему нужно и в том порядке, в котором это необходимо. Пользователь может видеть результаты решения задачи двумя путями: через интерфейс пользователя – с помощью блока “редактирование основных таблиц БД” и с помощью печатной формы, сохраненной программой на жестком диске (или любом другом внешнем носителе). Второй вид для человека гораздо более удобен. В форму выходных результатов можно даже внести исправления, но следует учитывать, что никакой проверки на целостность программой в этом виде результатов уже не осуществляется, т.е. все исправления, внесенные вручную, остаются на совести оператора, сделавшего изменения. Если же изменения делаются через интерфейс пользователя, то они проходят проверку на целостность результатов и сохраняются в БД. В случае не прохождения проверки, пользователю выводится соответствующее сообщение.

В программном блоке модуль обеспечения интерфейса пользователя, через который и идет все взаимодействие программного комплекса с человеком, который может запускать модуль решения задачи, модуль оптимизации результатов и создание печатной формы. Все эти модули взаимодействуют с БД. Модули оптимизации и решения задачи как берут данные из БД, так и вносят измененные сведения в нее. Как во вспомогательные таблицы, так и в основные. Модуль же вывода печатной формы только берет данные и только из основных таблиц БД. Следует также отметить, что если поддержание целостности в основных таблицах БД происходит и с помощью программных модулей и с помощью ключей и ограничений БД, то во вспомогательных таблицах следит за целостностью данных только программный блок.

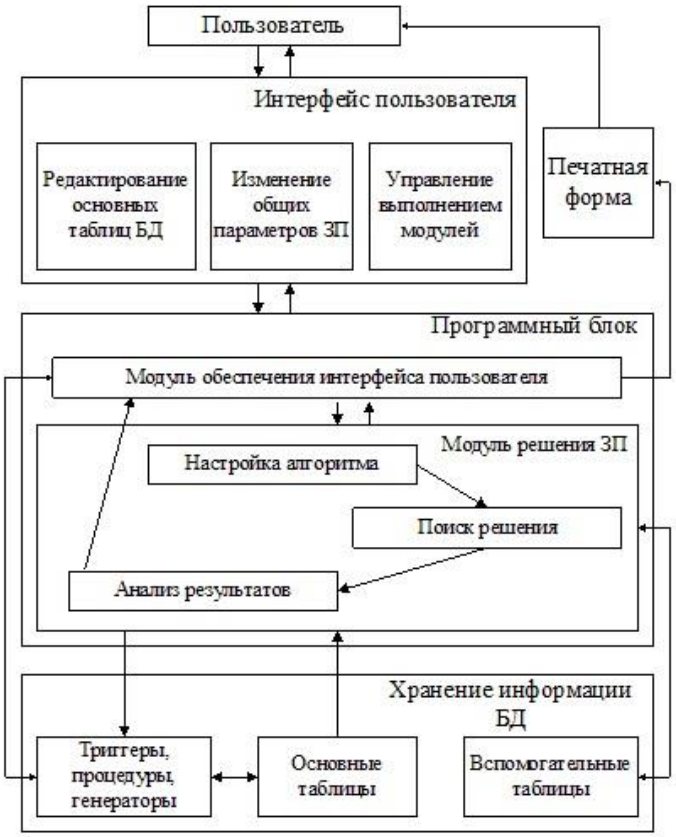


Рис 3.10 Схема работы программы

3.3. Выводы по третьей главе

1. В настоящей главе на основе анализа ПО построены графические диаграммы UML, позволившие определить основные элементы программного комплекса информационной системы.
2. На основе анализа процессов, сформулирован механизм обмена данными для информационного комплекса.
3. Построены блок-схемы работы программного комплекса по составлению расписания .

Глава 4. Использование программного комплекса для решения задачи

4.1 СРЕДСТВА РЕАЛИЗАЦИИ

При разработке программного продукта использовались ниже перечисленные средства реализации. Sql Server R2 2008 была выбрана для управления базой данных. Она выбрана как очень удобная и функциональная среда разработки. Для проектирования базы данных использовалась программа Power Designer 15.0. В качестве сервера приложений был выбран Windows - сервер Sql Server R2 2008. Важными преимуществами Sql Server перед другими web-серверами являются свободное распространение данного продукта, стабильность, высокая производительность, открытый исходный код. Все серверные скрипты Windows-приложения написаны на языке C#. C# – простое и мощное средство для динамического построения интерфейсы на Windows Forms. Вся функциональность клиентских приложений реализована с помощью C#. Для написания кода клиентской и серверной сторон Windows-приложения использовалась среда разработки visual studio 2010.

4.2 ТРЕБОВАНИЯ К АППАРАТНОМУ И ПРОГРАММНОМУ ОБЕСПЕЧЕНИЮ

Аппаратное обеспечение сервера данных: размеры свободного дискового пространства, необходимого для хранения базы данных зависят от количества данных и от версии Sql Server, но не могут составлять менее 300 Мб.

Аппаратное обеспечение программы Visual Studio 2010: чтобы установить программу нужно 2.2GB свободного дискового пространства.

Аппаратное обеспечение windows-сервера: не менее 4Мб свободного дискового пространства для хранения файлов приложения.

Аппаратное обеспечение клиента: полностью определяется выбранным visual studio.

Программное обеспечение сервера данных: Sql server 2008.

4.3 Описание программного комплекса

Для того чтобы представлять работу программы опишем интерфейс, предоставляемый комплексом конечному пользователю. В него включены меню, кнопки, линейки прокрутки, выпадающие списки, страницы и т.д. Также рассмотрим компоненты, не видимые пользователем программы, но имеющие значение для разработчика. Это компоненты, предназначенные для связи программы с базой данных[13].

4.4 Работа с программным комплексом

После запуска программы появляется основное диалоговое окно (Рис. 4.1).

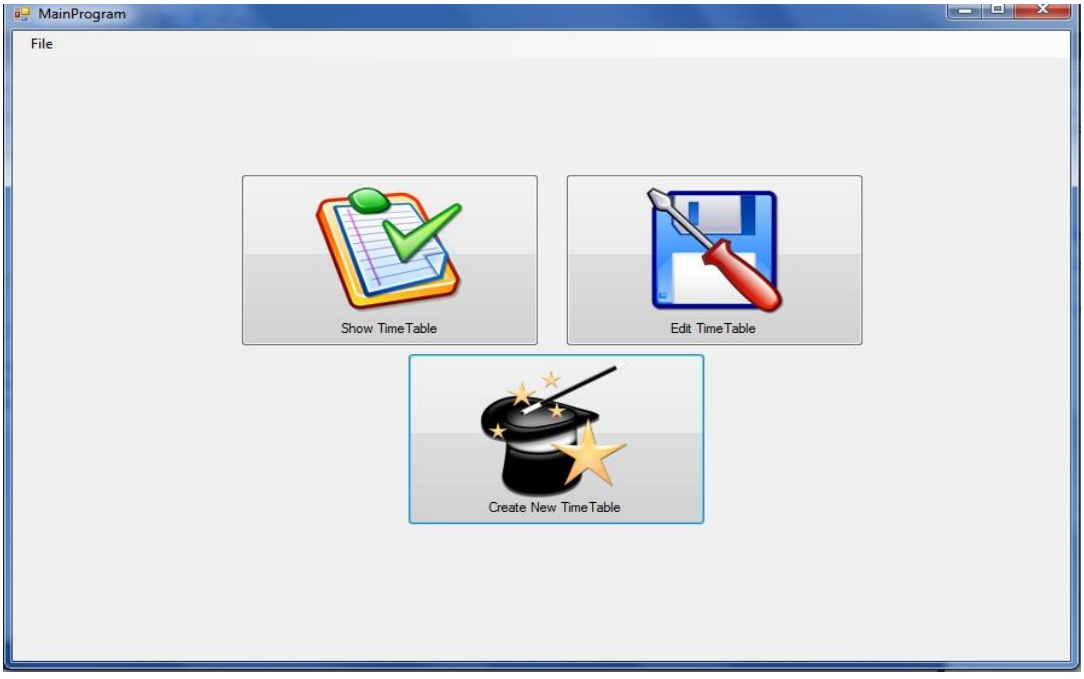


Рис 4.1. Основное диалоговое окно программы.

Скроллинг делает доступными страницы программы: Edit Time Table; Create New Time Table; Show Time Table (Рис. 4.1).

-Edit Time Table: Позволяет добавлять, редактировать и удалить данные в таблицах смстемы. В работе приведены следующие таблицы:

- **Предметы**
  - Добавить/удалить
  - Просмотреть/Редактировать
- **Групы студентов**
  - Добавить/удалить
  - Просмотреть/Редактировать
- **Преподаватели**
  - Добавить/удалить
  - Просмотреть/Редактировать
- **Аудитории**
  - Добавить/удалить
  - Просмотреть/Редактировать
- **Время**
  - Добавить/удалить
  - Просмотреть/Редактировать
- **Блоки**
  - Добавить/удалить
  - Просмотреть/Редактировать
- **Предметы .**

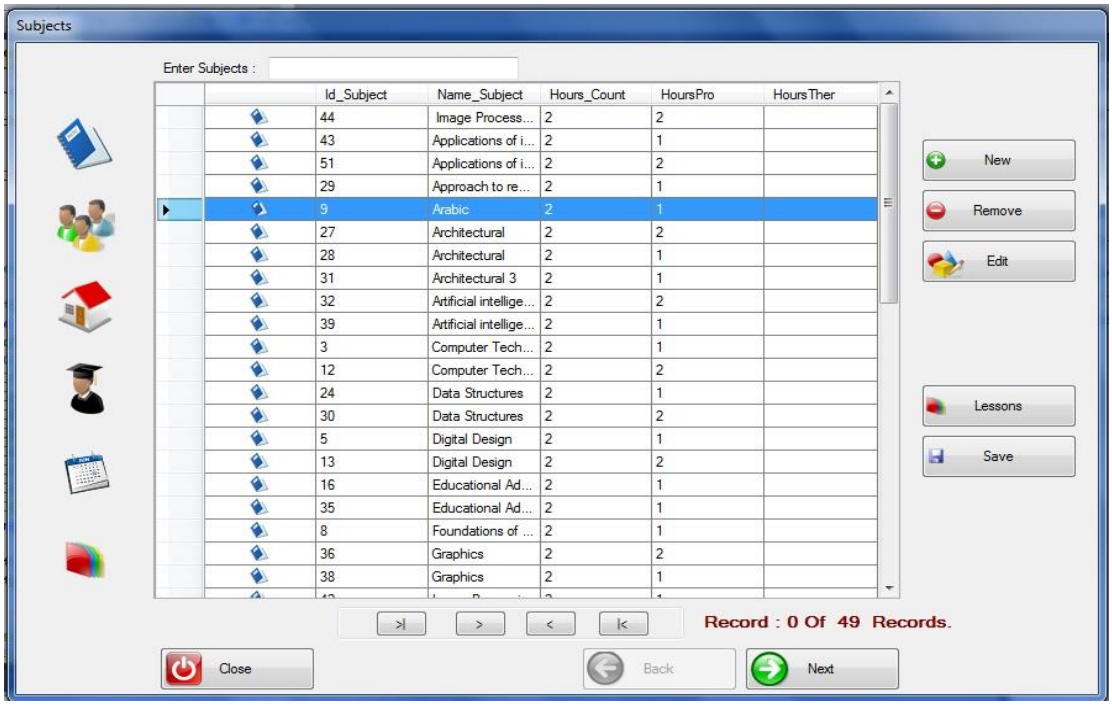


Рис 4.2. Интерфейс управления таблицой предметов.

На рис. 4.2. изображена таблица предметов, где пользователь может редактировать данные. Этот интерфейс состоит из следующих форм[1].

- **Enter Subject** – форма фильтрации. При введении в строку поиска наименование предмета, данная форма позволяет найти данные этого предмета.

Форма фильтрации

- New, Remove, Edit, Lessone , Save- эта форма позволяет редактировать данные предмета.

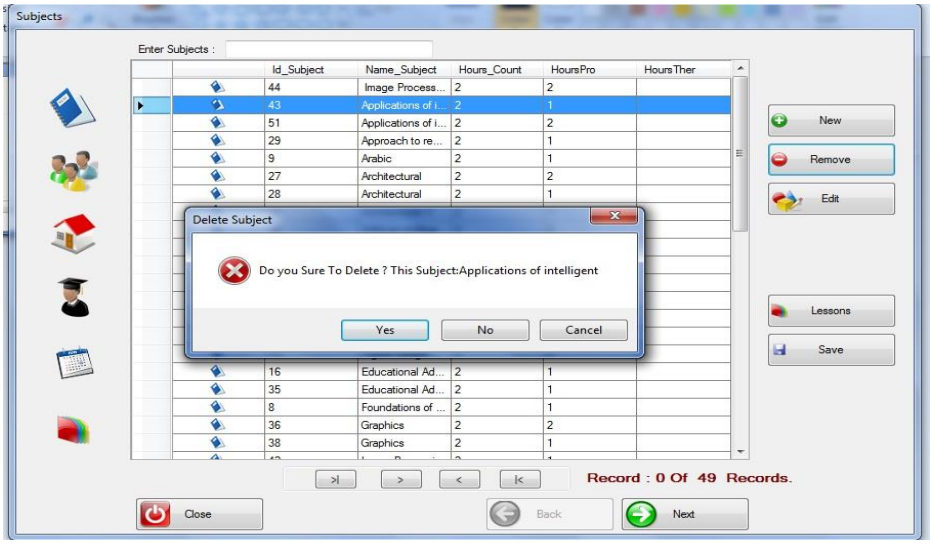
Форма Добавить/удалить/ Редактировать

- New : форма добавления предмета.

4.3. Форма добавления предмета

Чтобы добавить предмет необходимо
• Указать название предмета
• Указать количество часов в неделю
• Указать количество лекционных и практических часов занятий

• Remove : форма удаления предмета.



4.4. Форма удаления предмета

Чтобы удалить предмет необходимо
• Выбрать предмет из списка на рис. 4.2

• Нажать кнопку Remove
• Нажать кнопку yes для подтверждения удаления предмета

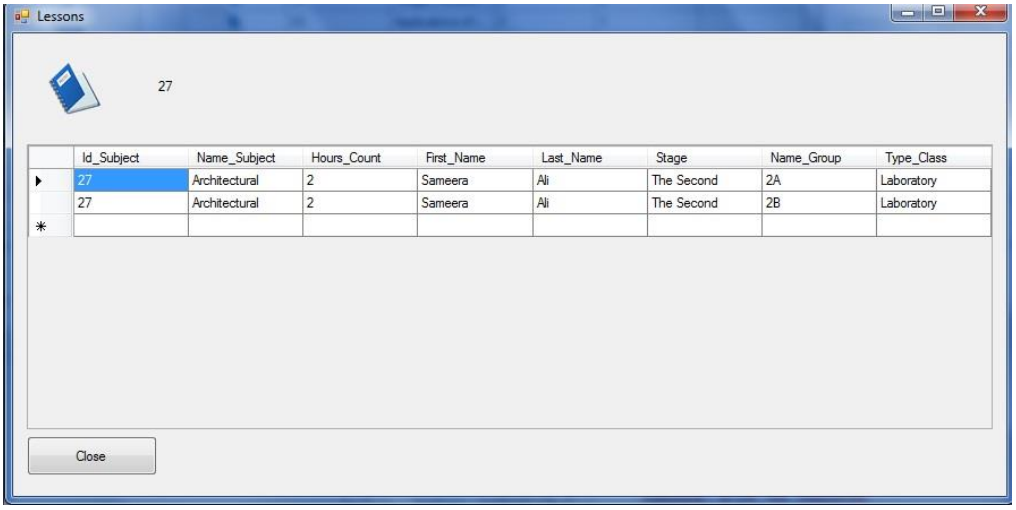
• Edit : форма редактирования.

4.5. Форма редактирование предмета

Чтобы редактировать предмет необходимо
• Выбрать предмет из списка на рис 4.2
• Нажать кнопку Edit
• Редактировать предмет

• Lessons: данная кнопка предназначена для более подробного просомтра предмета, т.е. можно узнать информацию о том, для кого проводится предмет и кто ведет этот предмет.

Чтобы просмотреть предмет необходимо
• Выбрать предмет из списка на рис 4.2
• Нажать кнопку Lessons
• показать предмет



4.6. Форма просмотра предмета

• Save : кнопка предназначена для сохранения измененных данных.

- **Групы студентов**

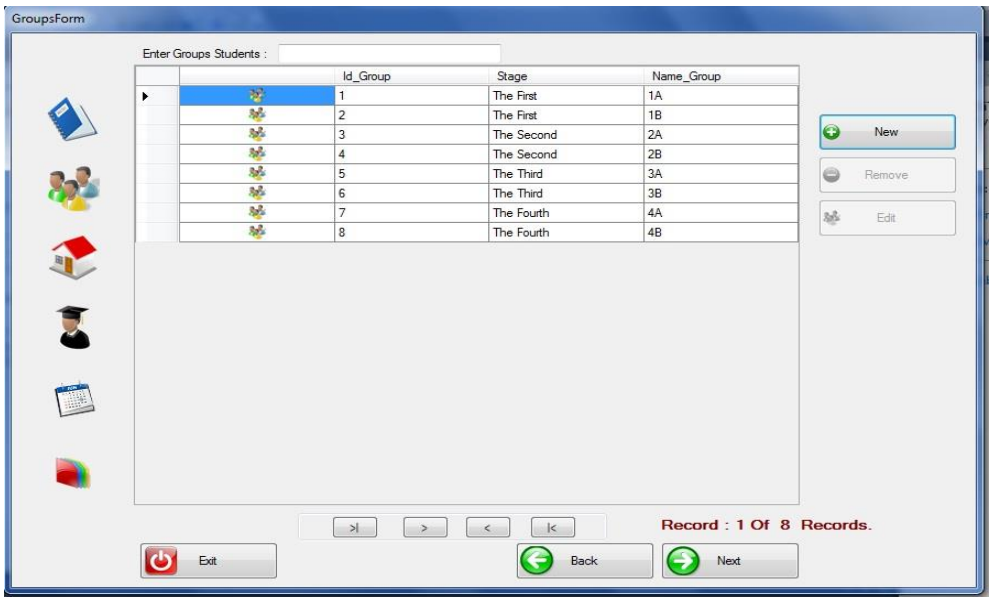


Рис. 4.7. Интерфейс управления таблицой групп студентов

В этом интерфейсе будут рассмотрены следующие функции:

- Добавить - добавление новой группы студентов.
- Удалить - удаление группы студентов.
- Просмотреть - просмотр всех групп.
- Редактировать - редактирование групп студентов.

- **Преподаватели.**

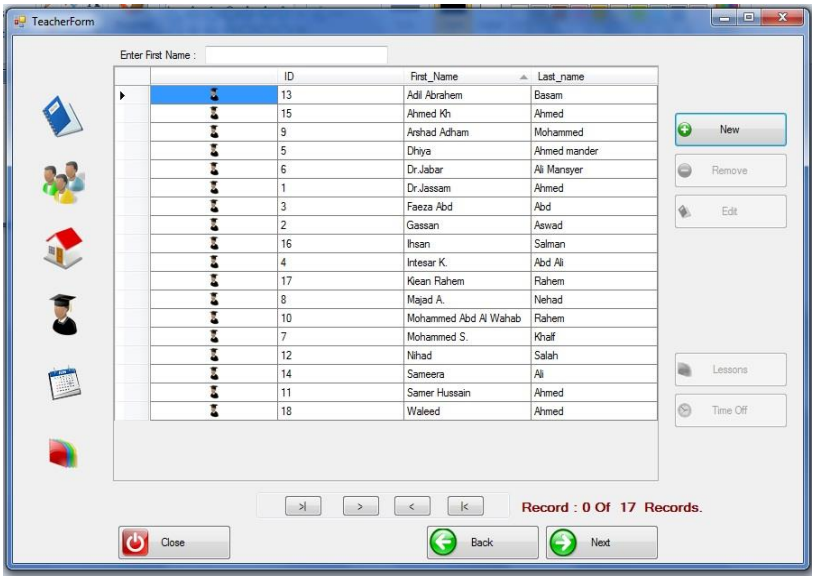
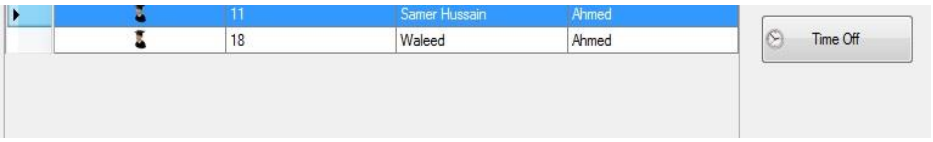


Рис. 4.8. Интерфейс управления таблицой преподавателей

В этом интерфейсе будут рассмотрены следующие функции:

- Добавить - добавление преподавателей.
- Удалить - удаление преподавателей.
- Просмотреть - просмотр преподавателей.
- Редактировать - редактирование преподавателей.
- Ограничить время - ограничение времени работы для преподавателей. Такое ограничение представлено на форме ниже.



Форма ограничения времени для преподвателей

Чтобы ограничить время для преподавателей необходимо
• Выбрать преподавтель из списка на рис. 4.4.
• Нажать кнопку Time Off
• После нажатия кнопки появится новое окно (рис. 4.5.), на которм можно ограничить время, выбрав подходящее время для работы преподавателей.

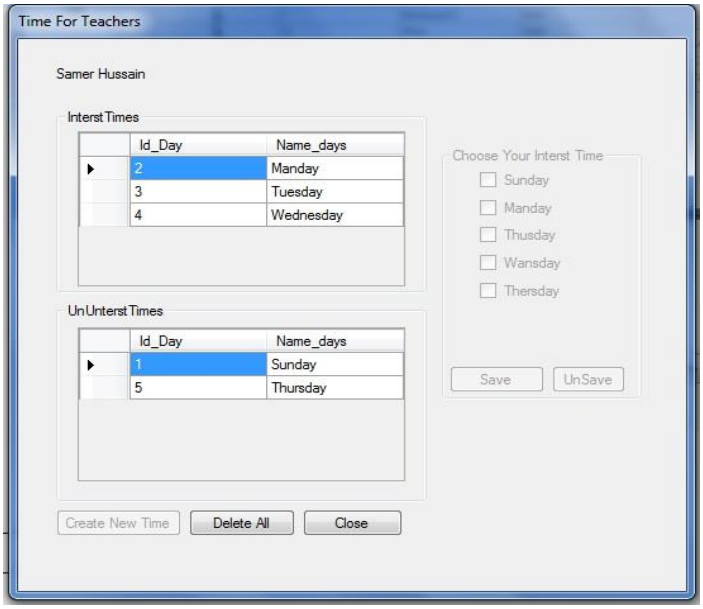


Рис.4.9 Интерфейс для заполнения ограничения времени работы для преподавателей.

- **Аудитории**

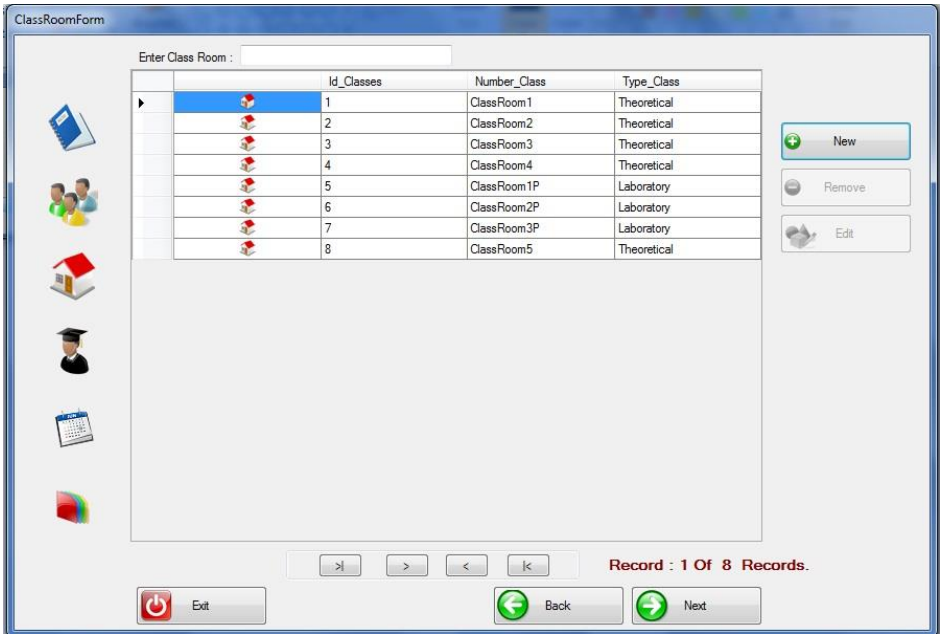


Рис. 4.10. Интерфейс управления таблицой аудиторий

В этом интерфейсе будут рассмотрены следующие функций:

- Добавить - добавление аудиторий.
- Удалить - удаление аудиторий.
- Просмотреть - просмотр аудиторий.

- Редактировать - редактирование аудиторий.

- **Время:**

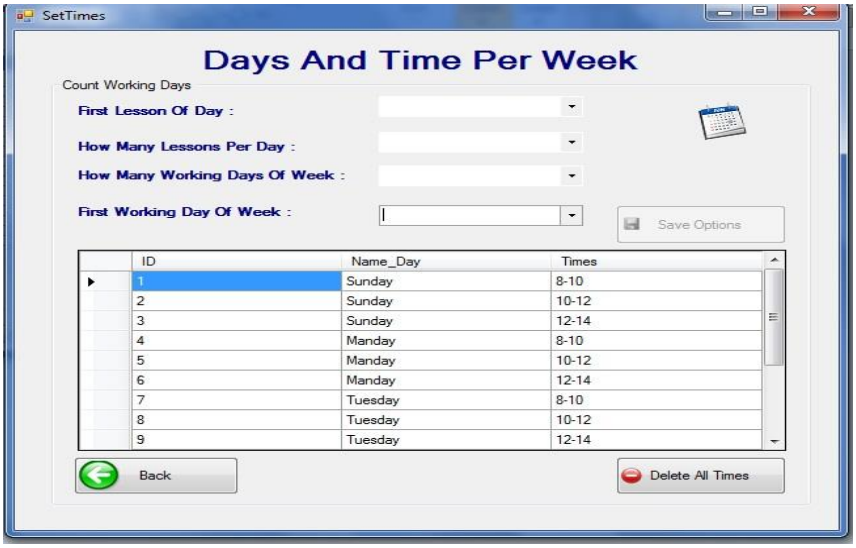


Рис. 4.11. Интерфейс управления таблицой времени

В этом интерфейсе будут рассмотрены следующие функции

- Добавить - добавление времени занятий для факультета. При добавлении времени необходимо выбрать настройки времени, как показано на рис. 4.11

Чтобы настроить параметры времени нужно заполнить следующие поля:

1- First Lesson Of Day - время начала первого занятия,

2- How Many Lessons Per Day - количество занятий в день,

3- How Many Days Of Week - количество дней в недели,

4- First Working Day Of Week -первый рабочий день недели,

5- Save Options - сохранение настроек в таблице времени.

- Delete All Times - удаление всех настроек времени.

- Просмотреть - просмотр результата как показано на рис. 4.8

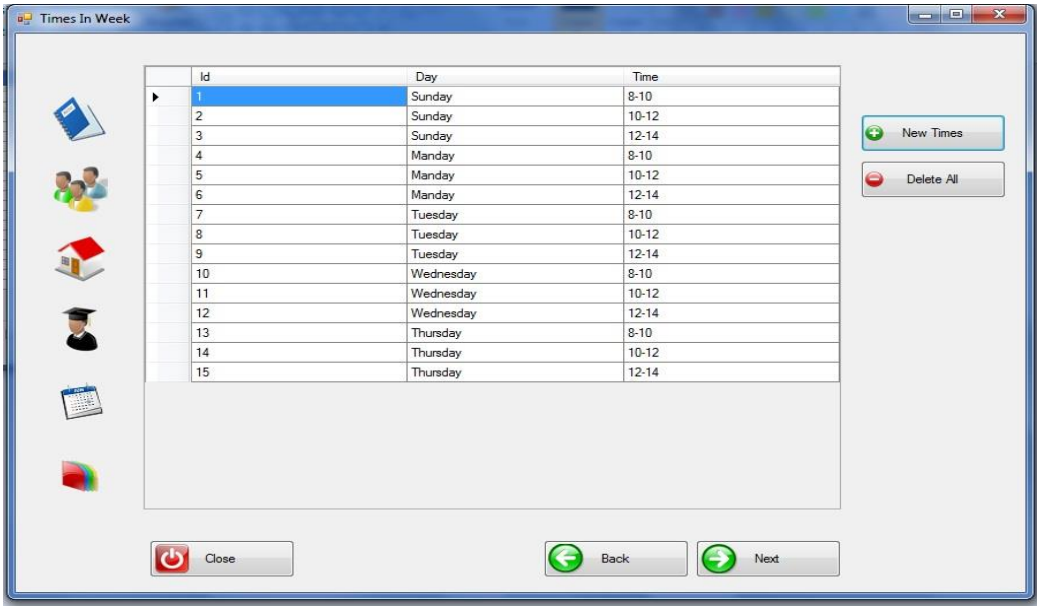


Рис. 4.12. Интерфейс просмотра результата времени

- **Блоки**

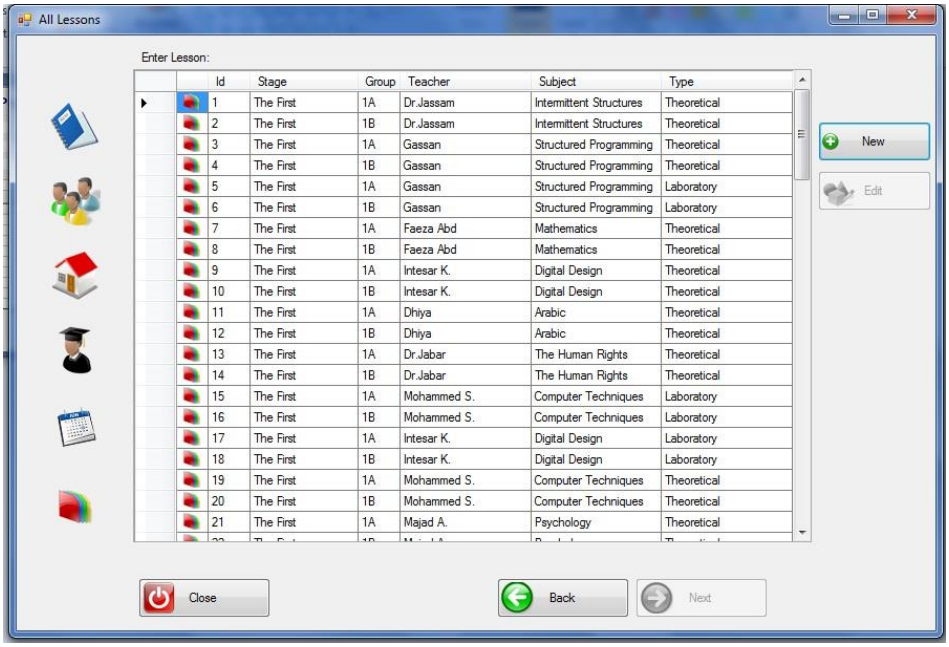
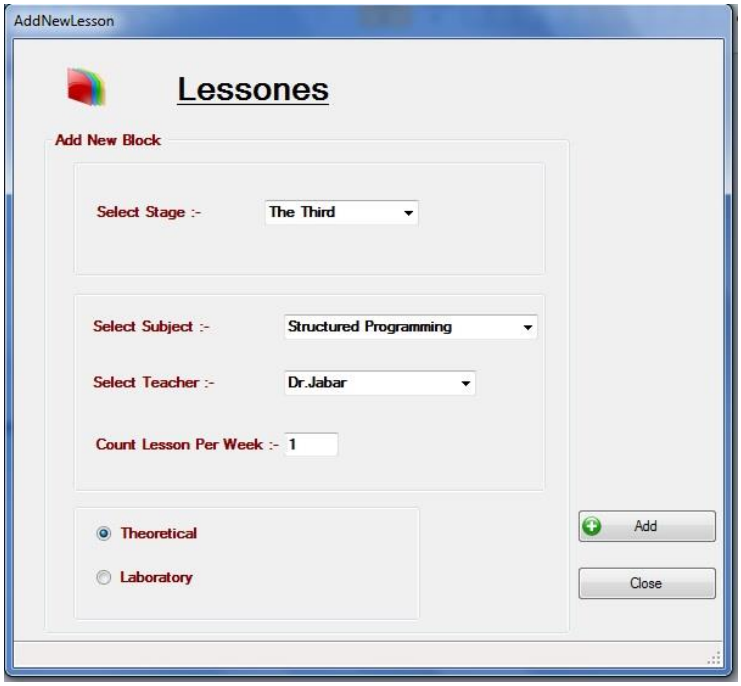


Рис. 4.13. Интерфейс управления блоками.

В этом интерфейсе будут рассмотрены следующие функции

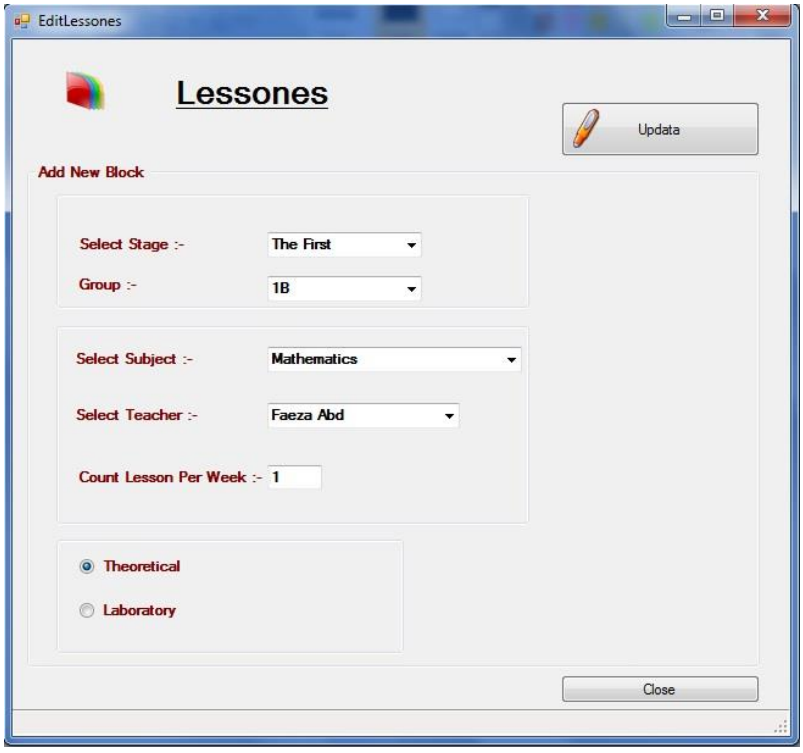
- Добавить - добавление нового блока



4.14. Форма добавления блока

Чтобы добавить блок необходимо
• Выбрать кнопку New на рис 4.9.
• Select stage - выбор курса.
• Select subject - выбор предмета
• Select teacher - выбор преподавателя.
• Count Lesson Per Week - количество блоков в неделю.
• Theoretical, Laboratory - тип блока.
• Add - принажатии этой кнопко блок добавится в таблицу блоков.

- Просмотреть - просмотр результата
- Редактировать - редактирование блока как показано на следующей форме.



4.15.Форма редактирования блока

Чтобы добавить блок необходимо
• Выбрать кнопку Edit на рис 4.9.
• Select stage - выбор курса.
• Select group - выбор группы студентов
• Select subject - выбор предмета
• Select teacher - выбор преподавателя.
• Count Lesson Per Week - количество блоков в неделю.
• Theoretical, Laboratory - тип блока.
• Update - при нажатии этой кнопки блок сохранится в таблице блоков.

4.4.1 Описание создания расписания.(Create Time Table)

после заполнения таблиц данными можно создать расписание, как показано на рис. 4.16. Этот интерфейс состоит из 4-х форм.

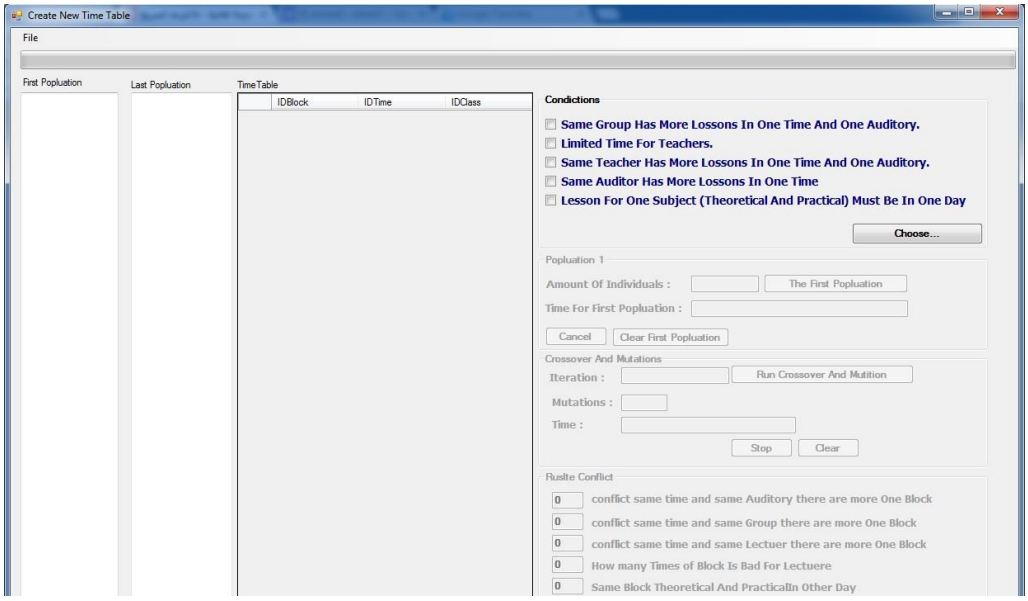


Рис.4.16 .Интерфейс Описание создания расписания.

1-я форма - это условия. Есть пять условий для составления расписания. Пользователь может самостоятельно выбрать необходимые условия из имеющихся. На рис. показаны следующие условия: одна группа имеет одну пару в одно время; ограниечение времени для преподавателья; один преподаватель имеет одно занятие в одной аудитории в одно время; в одной аудитории должна проводиться одна пара в одно время; порядок проведения лабораторных и практических занятий по одному предмету.

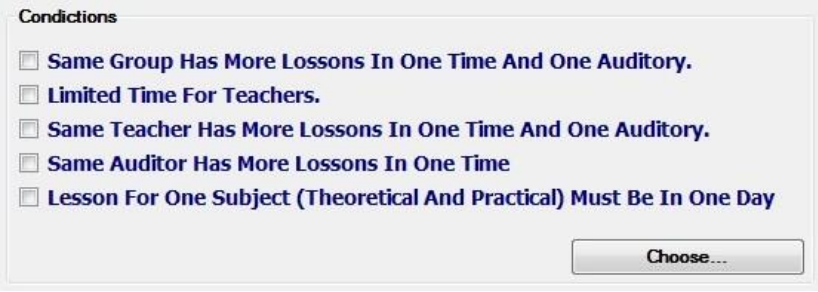


Рис. 4.17. Условия для создания расписания.

2-я форма - популяция. Для этой формы нужно указать размер популяции и нажать кнопку the first population (как показано на рис. 4.17.) и после этого получим первую популяцию

Рис. 4.18 Указание размера популяции.

На рис 4.18 показана первая популяция, которая состоит из пятисот индивидов. Каждый индивид имеет номер фитнеса, который показывает количество конфликтов. Индивиды расположены в порядке возрастания.

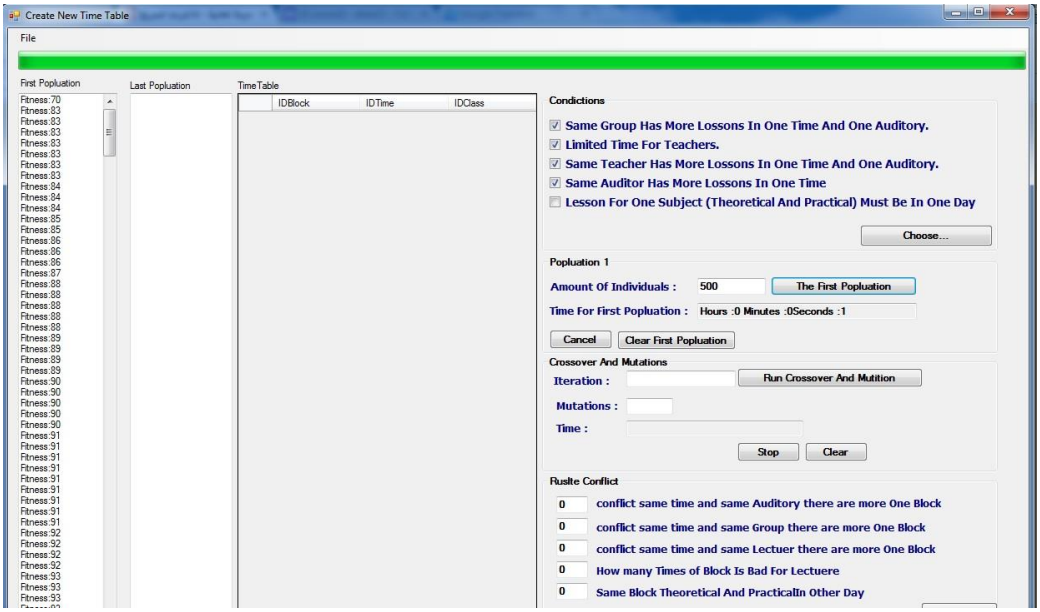


Рис. 4.19. Получение первой популяции.

3-я форма - кроссовер и мутация. После получения првой популяции, нужно указать размер мутации и количество повторов алгоритма, затем нажать кнопку Run crossover And Mutation, как показано на рис. 4.20

Рис. 4.20 Кроссовер и мутация.

На рис. 4.21 показан процесс поиска расписания без конфликтов после нажатия кнопки Run crossover And Mutation.

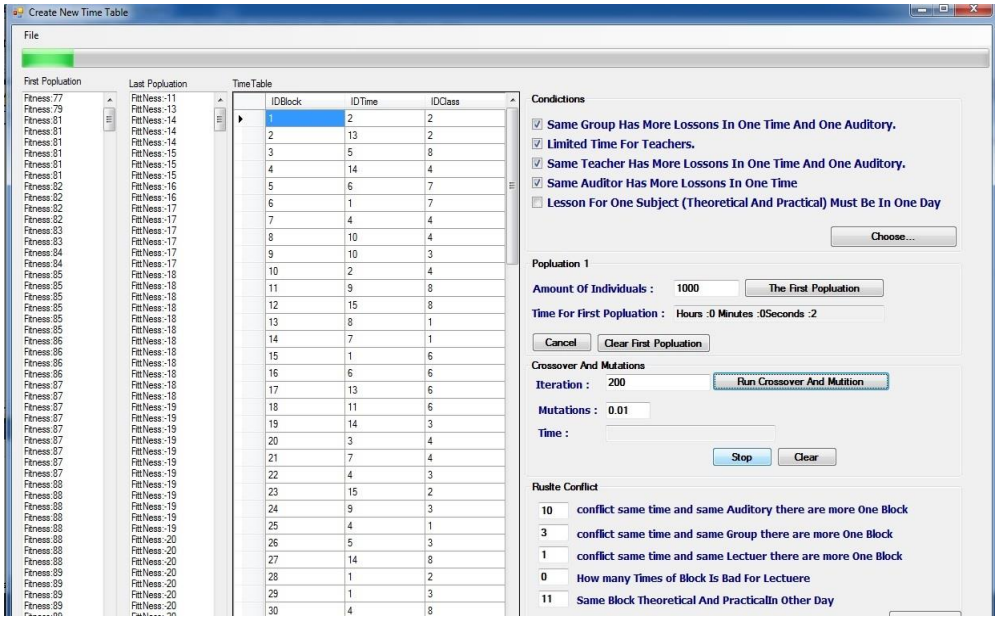


Рис. 4.21. процесс поиска расписания.

Процесс поиска расписания длится несколько минут, в данном случае одна минута. После этого мы получим индивид без конфликтов, который яляется расписанием. Скорость получения расписания зависит от технических характеристик компьютера. На компьютере, который имеет невысокие технические характеристики, время на поиск результата увеличивается и может сосавять до 5 минут.

4-я форма - результат расписания по каждому условию отдельно. Данная форма показывает количество конфликтов для каждого условия отдельно. Значения для каждого условия должны быть нулевыми, в противном случае имеются конфликты, которые нужно исправить при поторном запуске алгоритма.

Рис. 4.22 результат расписания по каждому условию отдельно

На рис. 4.23 показано готовое расписание без конфликтов, где выбраны все условия и указаны размер популяции и количество повторов алгоритма, выбран размер мутации.

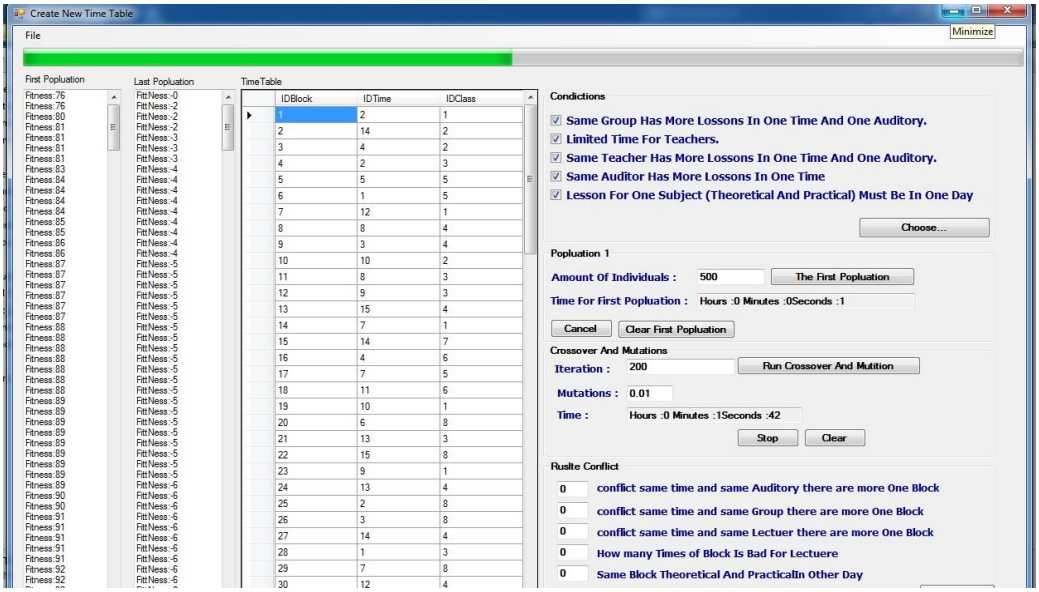


Рис. 4.23. готовое расписание.

4.4.2 Невидимые компоненты C # для связи программы с БД

Для связи программного комплекса с базой данных используется специальный модуль ClassDiagram с набором невидимых пользователю компонент C #, осуществляющих взаимодействие с базой данных[24].

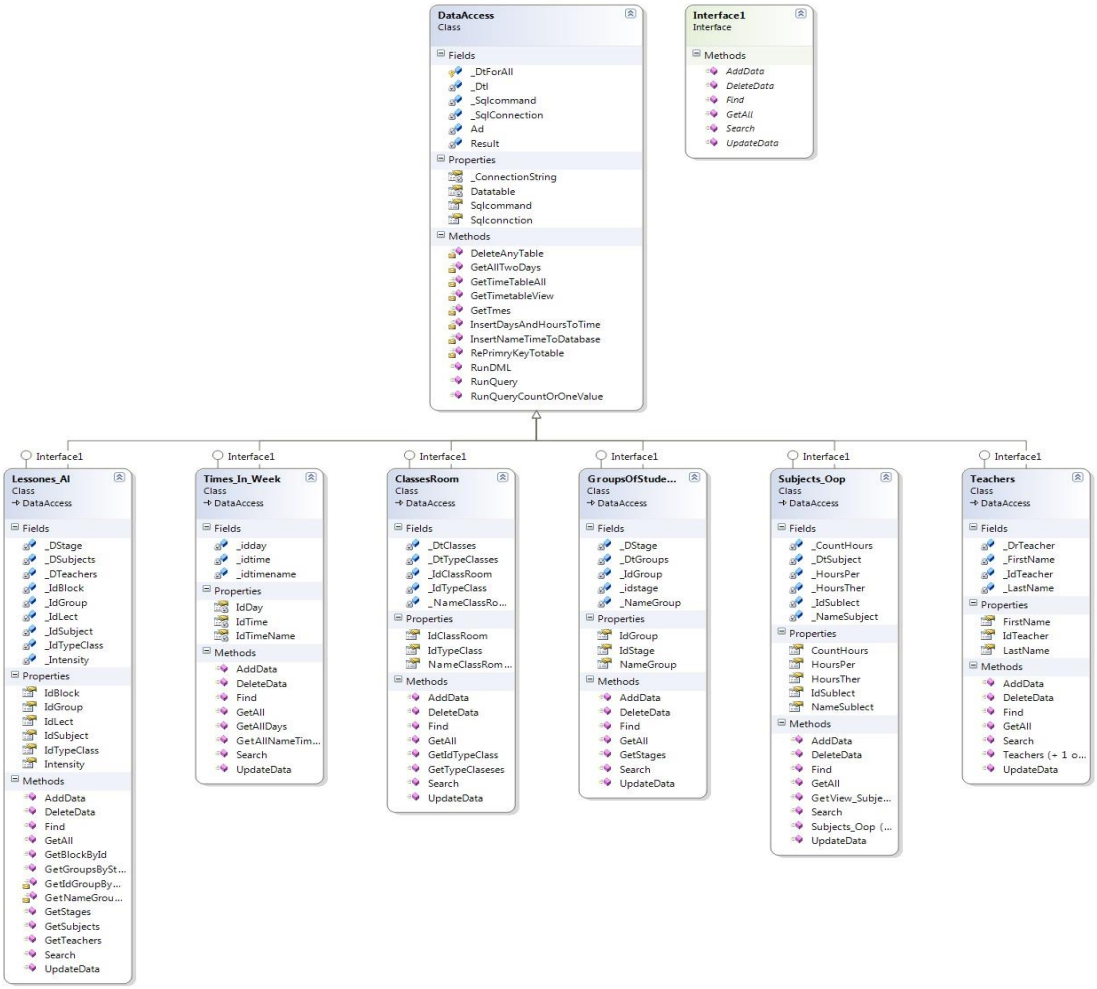


Рис 4.24. Модуль ClassDiagram с набором невидимых пользователю компонент C#.

4.5 Show Time Table : Результаты работы

На рис.4.25 Показан интерфейс расписания, где можно посмотреть расписание для каждого объекта отдельно или выбрать совокупность этих объектов.

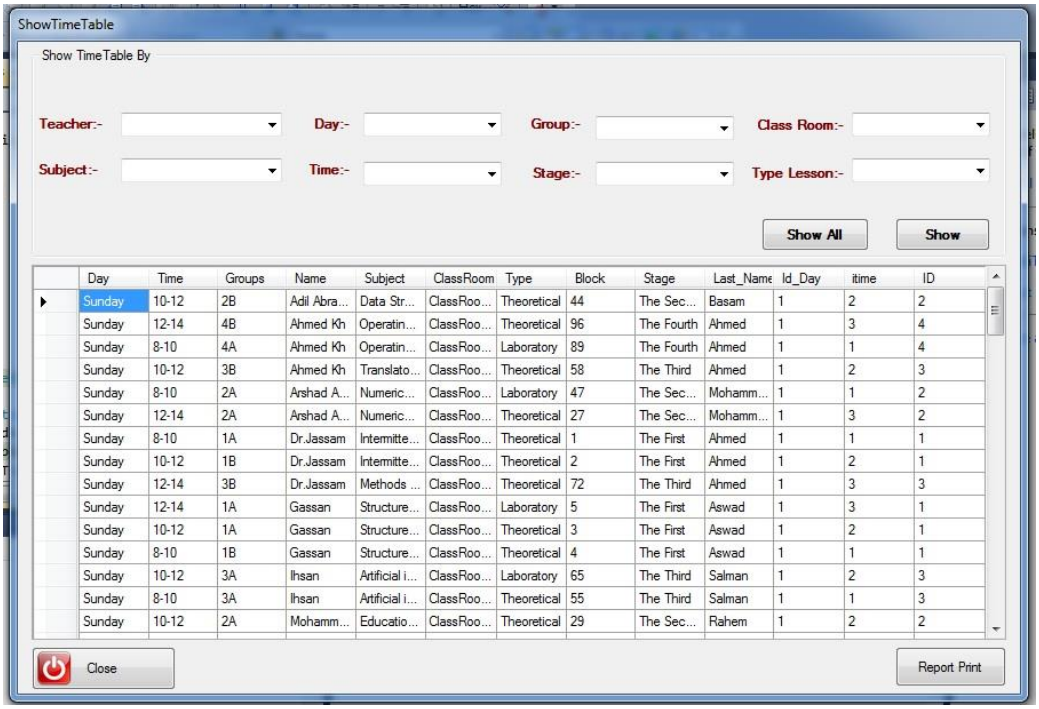
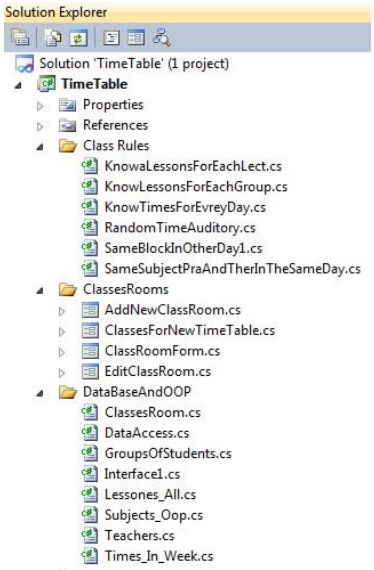


Рис. 4.25 Интерфейс расписания

4.6 Структура проекта

На рис. 4.27 приведена файловая структура проекта. Здесь более подробно рассказывается о том, как и из чего строятся интерфейс страницы, как вообще функционирует windows-приложение[61].



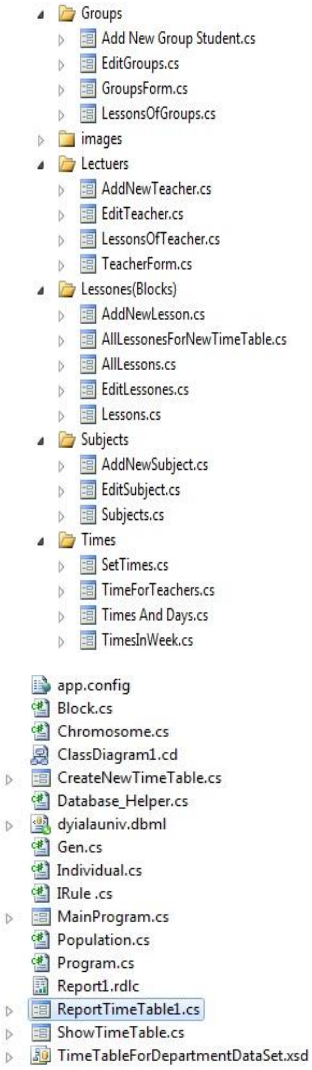


Рис 4.27. Файловая структура проекта.

Папка «DataBaseAndOOP» означает связь с базой данных и состоит из нескольких классов, каждый из которых является таблицей. Класс[64] DataAccess.cs является основным классом, который связывает базу данных с системой. Папка «ClassRules» состоит из классов, каждый из которых является условиемя, которое выбирается из системы. Это условие помогает получить расписание без конфликтов. Папка «ClassRooms» отвечает за аудитории. Она содержит операции по добавлению, редактированию, удалению аудиторий и состоит из нескольких интерфейсов - каждый для отдельной операции. Папка «ClassGroups» отвечает за группы. Она содержит операции по добавлению,

редактированию, удалению групп и состоит из нескольких интерфейсов - каждый для отдельной операции. . Папка «ClassLectuares» отвечает за преподаватлей. Она содержит операции по добавлению, редактированию, удалению преподавателей и состоит из нескольких интерфейсов - каждый для отдельной операции. . Папка «ClassSubjects» отвечает за предметы. Она содержит операции по добавлению, редактированию, удалению предметов и состоит из нескольких интерфейсов - каждый для отдельной операции. . Папка «ClassTimes» отвечает за время. Она содержит операции по добавлению, редактированию, удалению времени и состоит из нескольких интерфейсов - каждый для отдельной операции. В папке «images» - рисунки для интерфейсов. Файл app.confg - этот файл имеет код связи с базой данных. Класс Block инициализирует все блоки. Класс Chromosom - инициализирует хромосомы времени и хромосомы аудитории для каждого блока. Класс Diagram - является структурой связи между классами. Интерфейс Create TimeTable - интерфейс получения расписания. Класс DataBase_helper - сохраняет лучшее расписание, полученное с помощью алгоритма в базе данных и удаляет предыдущую старую версию расписания. Класс Individial - инициализирует индивид, который состоит из блока, хромосомы времени и хромосомы аудитории. Класс IRue - считает количество конфликтов в каждом индивиде. Класс Population - делает скрещивание и мутацию, чтобы создать новые потомки. Интерфейс Main Programm - главный интерфейс в системе. Интерфейс Show TimeTable - показывает расписание. Интерфейс Report TimeTable - распечатывает расписание. Файл TimeTable DataSet - файл хранит расписание для просмотра.

4.7 Подключение к БД

Подключение к базе данных с именем DATABASE пользователя с логином LOGIN и паролем PASSWORD осуществляется с помощью следующей функции:[82]

```
<?xml version="1.0"?>
<configuration>

  <connectionStrings>

    <add name="TimeTable.Properties.Settings.TimeTableForDepartmentConnectionString"
        connectionString="Data Source=firas-pc;Initial Catalog=TimeTableForDepartment;Integrated Security=True"
        providerName="System.Data.SqlClient" />

  </connectionStrings>

  <startup>
    <supportedRuntime version="v4.0" sku=".NETFramework,Version=v4.0"/>
  </startup>

</configuration>
```

```
private string _ConnectionString
    {
      get
      {
        return
TimeTable.Properties.Settings.Default.TimeTableForDepartmentConnectionString;
      }
    }
```

Функция _ConnectionString устанавливает соединение с сервером sql server 2008 r2 .

4.8 Описание программного продукта

Данная глава посвящена описанию программного продукта в целом, а также конкретным решениям в рамках решаемых задач.

В качестве средств реализации поставленных задач была выбрана связка c#+ Sql Server 2008. C# – это язык программирования[88], позволяющий

организовать удобный доступ к базам данных. Взаимодействие C# предоставляет неплохие возможности представления информации пользователю. Благодаря тому, что sql server является серверным языком сценариев, разработанные алгоритмы выполняются на сервере, выдавая пользователям конкретные результаты. Это позволяет получить доступ к разработанному ПО с любой ЭВМ сети инфраструктуры расписания.

Схематически, функциональную схему программного комплекса можно представить следующим образом.

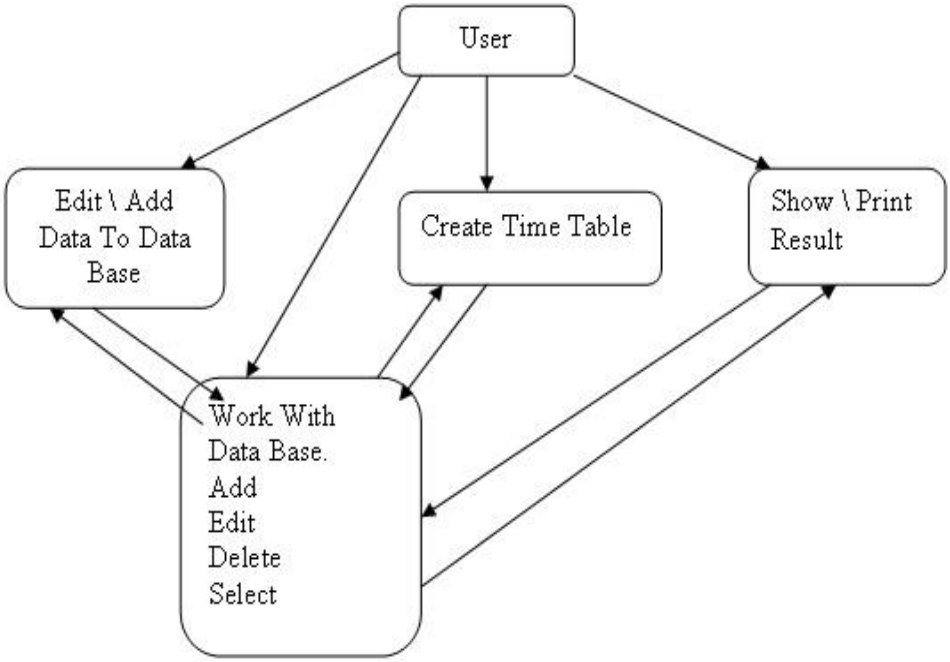


Рис. 4.28 Функциональная схема программного комплекса

На основе функциональной схемы разработана структурная схема взаимодействия основных модулей.

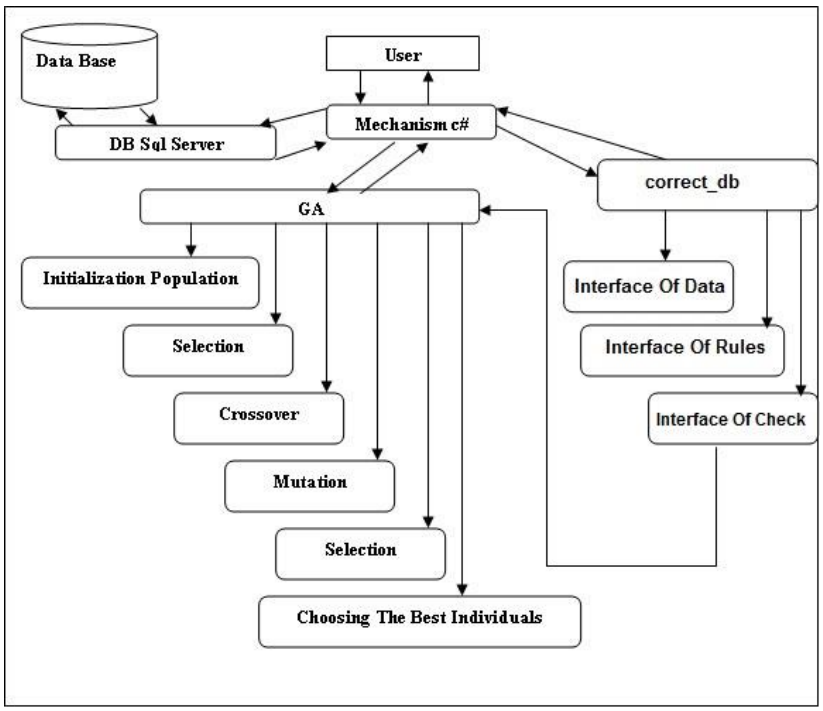


Рис. 4.29 Структурная схема взаимодействия основных модулей

Структура основного модуля работы с базой данных с применением языка C# и СУБД SQL SERVER имеет следующий вид:

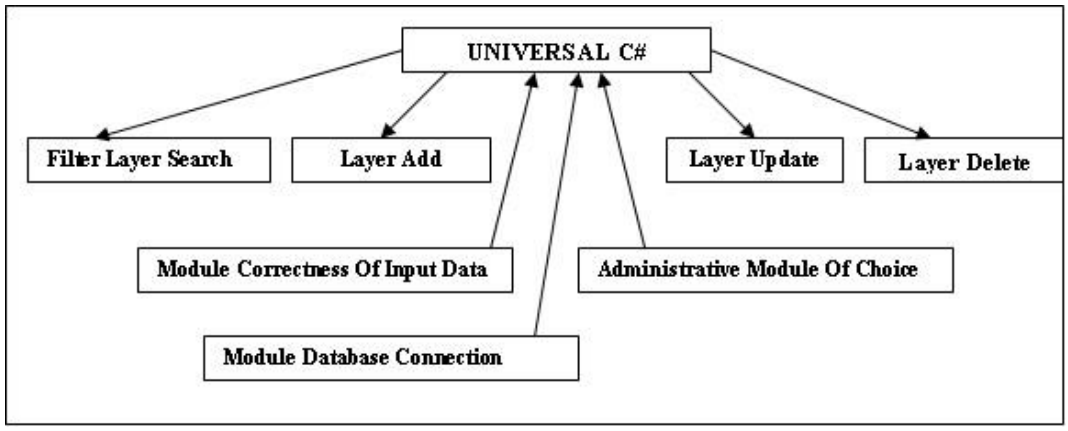


Рис. 4.30 Базовый модуль для работы с БД

4.9 Выводы по четвертой главе

1. В настоящей главе всесторонне рассмотрены принципы взаимодействия пользователя с программным комплексом.

2. Поставлены и проведены решения задачи построения расписания. Результаты приведены в виде таблиц.

ЗАКЛЮЧЕНИЕ

Заключение отражает основные результаты, полученные в ходе выполнения диссертационной работы.

Основным результатом настоящей диссертационной работы является модель формализация задачи составления расписания в вузе. Среди наиболее важных результатов можно выделить:

1. Проведенный анализ существующих методов решения задачи составления расписания занятий в вузах.
2. Разработку модели составления расписания с агрегированными объектами.
3. Создание генетического алгоритма, осуществляющего поиск квазиоптимального варианта расписания, основными отличиями которого является представление решения в виде особи, состоящей из трех хромосом со специальными операторами скрещивания и мутации, гена «конфликтов», повышающего эффективность алгоритм составления расписания.
4. Разработку программного комплекса, обеспечивающего нахождение квазиоптимального расписания с учетом всех ограничений и его применение для составления расписания в университете г. Диала Ирака.

СПИСОК ЛИТЕРАТУРЫ

1. Microsoft Corporation. Разработка Windows-приложений на Visual Basic .NET и Visual C# .NET. Учебный курс MCAD/MCSD. / – Пер. с англ. – М.: Русская редакция, 2003 г., 512 с.
2. Айвазян С. А. Классификация многомерных наблюдений / С.А. Айвазян, З.И. Бежаева, О.В. Староверов. - М. : Статистика,1974.-383 с.
3. Айзерман Н.А. Выбор вариантов: основы теории. / Н.А. Айзерман, Ф.Т. Алескеров.-М.:Наука. Гл.ред. физ-мат. лит., 1990.-240с.
4. Акофф Р.Л. Планирование в больших экономических системах / Р.Л. Акофф; Пер. с англ.-М.:Сов. Радио 1972.-223 с.
5. Астахова И. Ф. Применение объектно-ориентированной технологии для создания гибкой автоматизированной системы / И.Ф. Астахова, Т.В. Курченкова, В.Н. Стариков // Математика. Образование. Экология. Гендерные проблемы. : Материалы межд. конф.– Т.1.– М.: Прогресс-Традиция, 2003.– С. 95-96.
6. Астахова И. Ф. Разработка информационной системы построения расписания / И.Ф. Астахова, Т.В. Курченкова // Математика. Образование. Экология. Гендерные проблемы. : Материалы межд. конф.– Т.2.– М.: Прогресс-Традиция, 2001.– С. 287-290.
7. Багриновский К.А. Интеллектная система в отраслевом планировании / К.А. Багриновский, В.В. Логвинец.-М.:Наука,1989.-136 с.
8. Барон Д. Рекурсивные методы в программировании. / Д. Барон: пер. с англ. Мартынюка В.В., под ред. Любимского Э.З.-М.:Мир, 1974.-79 с.
9. Батищев Д.И. Генетические алгоритмы решения экстремальных задач / Нижегородский университет. – Нижний Новгород: 1995 г., 62 с.
10. Бердж Вильям Методы рекурсивного программирования / Вильям Бердж.-М.:Машиностроение, 1983.-248 с.
11. Берзин Е. А. Оптимальное распределение ресурсов и теория игр / Е.А. Берзин.-М.:Радио и связь, 1983.-216 с.

12. Берзин Е. А. Оптимальное распределение ресурсов и элементы синтеза систем / Е.А. Берзин.-М.:Сов.радио, 1974.-304 с.
13. Биллиг В. А. Основы программирования на C#. — М.: Изд-во «Интернет-университет информационных технологий — ИНТУИТ.ру», 2006. — 488 с.
14. Блауберг И. В. Становление и сущность системного подхода / И.В. Блауберг, Э.Г. Юдин.-М.:Наука, 1973.-270 с.
15. Боггс У. UML и rational rose / У. Боггс, М. Боггс.-М.: Лори, 2000.-581 с.
16. Бусленко Н. П. Моделирование сложных систем / Н.П. Бусленко.-М.:Наука, 1978.- 400 с.
17. Буч Г. Язык UML: Рук. пользователя / Г. Буч, Д. Рамбо, А. Джекобсон.-М.: ДМК Пресс, 2001.- 429 с.
18. Вагнер Г. Основы исследования операций. / Г. Вагнер:пер. с англ. Вавилова Б.Т.-М.:Мир, 1977. т.1.-335 с., т.2.-488 с., т.3.-501 с.
19. Ватсон К. C#. — М.: Лори, 2004. — 880 с.
20. Вендров А. М. Case-технологии: Современные методы и средства проектирования информационных систем./ А.М. Вендров.–М.: Финансы и статистика, 1998.–175.
21. Вентцель Е. С. Исследование операций: задачи, принципы, методология / Е.С. Вентцель.-М.: Наука, 1988.-206 с.
22. Вентцель Е. С. Теория вероятностей и ее инженерные приложения: Учеб. пособие для студентов высших техн. учеб. заведений / Е.С. Вентцель, Л.А. Овчаров.-М.: МГУ, 1992.-400 с.
23. Вербовецкий А. А. Основы проектирования баз данных / А.А. Вербовецкий.-М.: Радио и связь, 2000.- 85 с.
24. Вирт Н. Алгоритмы и структуры данных. — Спб: Невский диалект, 2001 г. — 352 с.
25. Власов В. В. Общая теория решения задач (рациология) / В.В. Власов.-М.: Статистика, 1983.-187 с.
26. Володин В. Офисный софт: Расписание – это просто или не очень просто / В. Володин, И. Володина. – (http://tech.stolica.ru/article.php?id=2003082903).

27. Володин В. Офисный софт: Расписание – это просто или не очень просто. Часть II / В. Володин, И. Володина. – (http://tech.stolica.ru/article.php?id= 2003091401).
28. Гафт М. Г. Принятие решений при многих критериях / М.Г. Гафт.-М.: Знание, 1979.-64 с.
29. Гиг Дж. Прикладная общая теория систем / Дж. Гиг: пер. с англ.-М.: Мир, 1981.-733 с.
30. Гилл Ф. Практическая оптимизация. / Ф. Гилл, У. Мюррей, М. Райт; пер. с англ. Лебедева В.Ю. под ред. Петрова А.А.-М.:Мир, 1985.-509 с.
31. Гладков Л.А. Генетические алгоритмы / Гладков Л.А., Курейчик В.В., Курейчик В.М. – М.: ФИЗМАТЛИТ, 2006.- 320 с.
32. ГОСТ 19.701-90 (ИСО 5807-85). Единая система программной документации. Схемы алгоритмов, программ, данных и систем. Условные обозначения и правила выполнения. – М.: Издательство стандартов, 1991. – 26 с.
33. Гуннерсон Э. Введение в C#. Библиотека программиста. — СПб.: Питер, 2001. — 304 с.
34. Гурин Л. С. Задачи и методы оптимального распределения ресурсов / Л.С. Гурин, Я.С. Дымарский, А.Д. Меркулов.-М.:Сов.радио, 1968.-463 с.
35. Давыдов Э. Г. Игры, графы, ресурсы / Э.Г. Давыдов.-М.:Радио и связь, 1981.-112 с.
36. Данциг Д. Линейное программирование / Д. Данциг.-М.:Прогресс, 1966.-600 с.
37. Джонс Дж.К. Методы проектирования / Дж.К. Джонс: пер. с англ.-М.:Мир, 1986.-326 с.
38. Дружинин В. В. Введение в теорию конфликта / В.В. Дружинин, Д.С. Конторов, М.Д. Конторов.-М.:Радио и связь, 1989.-288 с.
39. Дубов Ю. Я. Многокритериальные модели формирования и выбора вариантов систем / Ю.Я. Дубов, С.И. Травкин, В.Н. Якимец.-М.: Наука, 1986.-296 с.

40. Йордон Э. Структурные модели в объектно-ориентированном анализе и проектировании / Э. Йордон, К. Аргила; под ред. В. Алеева.–М.: Лори, 1999.–264 с.
41. Исаев А. Генетические Алгоритмы – http://algolist.manual.ru.
42. Калашников В. В. Сложные системы и методы их анализа / В.В. Калашников.-М.:Знание, 1980.-312 с.
43. Калянов Г.Н. CASE. Структурный системный анализ (автоматизация и применение)/ Г.Н. Калянов.–М.: ЛОРИ, 1996.–242 с.
44. Канторович Л. В. Математические методы организации и планирования / Л.В. Канторович.-Л.:Изд.-во ЛГУ, 1939.-68 с.
45. Карлин С. Математические методы в теории игр, программировании и экономике / С. Карлин; Пер. с англ.-М.:Мир, 1964.–838 с.
46. Кватрани Т. Rational Rose 2000 и UML. Визуальное моделирование / Т. Кватрани: пер. с англ.–М.:ДМК Пресс, 2001.–175 с.
47. Кини Р. Л. Принятие решений при многих критериях: предпочтения и замещения / Р.Л. Кини, Г. Райфа: Пер. с англ.-М.:Радио и связь, 1981.-560 с.
48. Кнут Дональд Э. Искусство программирования для ЭВМ. / Дональд Э. Кнут; Пер. с англ. – М.: Мир. Т. 4. Комбинаторные алгоритмы, 1979, 844 с.
49. Ков О. UML. Мета-язык проектирования и моделирования программного обеспечения/ О. Ков. – (www.metod.square.spb.ru), 2001.
50. Конвей Р. В. Теория расписаний / Р.В. Конвей, В.Л. Максвелл, Л.В. Миллер.-М.:Наука, 1975.-360 с.
51. Корн Г. Справочник по математике (для научных работников и инженеров) / Г. Корн, Т. Корн.-М.:Наука, 1977.-832 с.
52. Коробкин А.А. Модели и методы искусственного интеллекта. Использование генетических алгоритмов и аппарата нечеткой логики для организации учебного процесса в ВУЗах./ И.Ф.Астахова, А.А.Коробкин-Berlin: LAP LAMBERTAcademic Publishing GmbH& CO. KG, 2012. – 137 с.
53. Кофман А. Сетевые методы планирования / А. Кофман, Г. Дебазей; пер. с фр.-М.: Прогресс, 1968.-181 с.

54. Коффман Э. Г. и др. Теория расписаний и вычислительные машины. / Э.Г. Коффман, Р. Сети, Дж. Л. Бруно и др.; Под ред. Э. Г. Коффмана – М.:Наука, 1984.-334 с.
55. Краснощеков П. С. Принципы построения моделей / П.С. Краснощеков, А.А. Петров.-М.:Изд.-во МГУ, 1983.-264 с.
56. Кренке Д. Теория и практика построения баз данных / Д. Кренке : пер. с англ.–8-е изд.–СПб.: Питер, 2003.–799 с.
57. Кузнецов Ю. Н. Математическое программирование: Уч.пособие / Ю.Н. Кузнецов, В.И. Кузубов, А.В. Волощенко.-М.:Высш.шк.,1980.-300 с.
58. Курченкова Т. В. Разработка информационной системы построения расписания экзаменов / Т.В. Курченкова // Труды молодых ученых ВГУ. Вып. 1.– Воронеж: Воронеж. госуниверситет, 2002.– С. 18-19.
59. Ларичев О. И. Наука и искусство принятия решений / О.И. Ларичев.-М.:Наука, 1979.-200 с.
60. Левченков В. С. Алгебраический подход к теории выбора / В.С. Левченков.-М.:Наука, 1990.-167 с.
61. Либерти Д. Программирование на C#. — СПб.: Символ-Плюс, 2003. — 688 с.
62. Льюис К. Метод программирования экономических показателей / К. Льюис.-М.:Финансы и статистика,1986.-130 с.
63. Льюс Р. Д. Игры и решения / Р.Д. Льюс, Х. Райфа: Пер. с англ.-М.:ИЛ, 1961.-642 с.
64. Майо Д. C#. Искусство программирования. Энциклопедия программиста. — Киев: «ДиаСофт», 2002. — 656 с.
65. Майо Дж. C# Builder. Быстрый старт. — М.: Бином, 2005. — 384 с.
66. Маклаков С. В. ERwin и BPwin. CASE-средства разработки информационных систем./ С.В. Макланов – М.:ДИАЛОГ-МИФИ, 2000 – 256 с.
67. Мамиконов А.Г. Методы разработки автоматизированных систем управления / А.Г. Мамиконов. – М.:Энергия, 1973. – 336с.

68. Марка Д.А. Методология структурного анализа и проектирования./ Д.А. Марка, К. Мак Гоуэн. – М.: МетаТехнология, 1993. –235 с.
69. Марчук Г. И. Методы вычислительной математики: Учеб. пособие для вузов по спец. “Прикл. математика” / Г.И. Марчук.– М.:Наука, 1989.-608 с.
70. Мацяшек Л.А. Анализ требований и проектирование систем. Разработка информационных систем с использованием UML / Л.А. Мацяшек; пер. с англ. и ред. В.М. Неумоина. – М.: Вильямс, 2002.–428 с.
71. Месарович М. Общая теория систем: математические основы / М. Месарович, Я. Такахара.-М.:Мир, 1978.-311 с.
72. Методы и средства объектно-ориентированного программирования: Сб. научн. тр. АН УССР. Ин-т кибернетики им.В.М.Глушкова.-Киев:ИК, 1991.-68 с.
73. Микелсен К. Язык программирования C#. Лекции и упражнения. Учебник. — Киев: «ДиаСофт», 2002. — 656 с.
74. Мороз А. И. Курс теории систем / А.И. Мороз.-М.:Высш.шк., 1987.-412 с.
75. Мюллер Дж. Базы данных и UML: Проектирование / Роберт Дж. Мюллер: Пер. с англ.-М.: ЛОРИ, 2002.-420 с.
76. Нейбург Э.Д. Проектирование баз данных с помощью UML / Э.Д. Нейбург, Р.А. Максимчук.–М.: Изд. Дом “Вильямс”, 2002.–288 с.
77. Низамова Г.Ф. Математическое обеспечение составления расписания учебных занятий на основе генетических алгоритмов // Диссертация на соискание ученой степени кандидата наук – Уфа: УГАТУ, 2006, 135с.
78. Никлаус Вирт Алгоритм+структуры данных=программы./ Вирт Никлаус; пер. с англ. Иоффе Л.Ю., под ред. Подшивалова Д.Б. – М.: Мир, 1985. – 404 с.
79. Николаев В.И. Систематехника:методы и приложения / В.И. Николаев, В.М. Брук.-Л.:Машиностр.-е, 1985.-199 с.
80. О проекте федеральной целевой программы "Развитие единой образовательной информационной среды на 2001-2005 годы" / Министерство образования Российской Федерации: Решение коллегии от

24.04.2001 № 9/2.– (http://www.informika.ru/text/goscom/struk/kolleg/resh/01/04/9-2.html).

81. Орел Е.Н. Моделирование процессами управления проектами при ресурсных ограничениях И/ИЛИ / Е.Н. Орел, Т.Я. Орел // Эволюционная информатика и моделирование.-М.:ИФТП, 1994.-с. 165-185.
82. Павловская Т. А. Паскаль. Программирование на языке высокого уровня. Учебник для вузов. — СПб.: Питер, 2003. — 393 с.
83. Панченко Т.В. Генетические алгоритмы / под.ред.Ю.Ю.Тарасевича. – Астрахань: «Астраханский университет», 2007 г., 87 с.
84. Перегудов Ф. И. Введение в системный анализ / Ф.И. Перегудов, Ф.П. Тарасенко.-М.:Высш.шк,1989.-367 с.
85. Петер Пин-Шен Чен. Модель “сущность-связь” – шаг к единому представлению данных./ Петер Пин-Шен Чен.// Системы управления базами данных.–N 3.–Б.м.–1995.–С. 137-158.
86. Подиновский В. В. Оптимизация по последовательно применяемым критериям / В.В. Подиновский, В.М. Гаврилов.-М.:Сов.радио, 1975.-192 с.
87. Поспелов Г.С. Программно-целевое планирование и управление (Введение) / Г.С. Поспелов, В.А. Ириков.-М.:Сов.радио,1976.-440 с.
88. Прайс Д., Гандэрлой М. Visual C#.NET. Полное руководство. — Киев: «Век», 2004. — 960 с.
89. Применение ЭВМ в учебном процессе / Под ред. А.И. Берга.-М.,1969.-248 с.
90. Пугачев В. С. Теория вероятности и математическая статистика: Учебн. / В.С. Пугачев. –2-е изд., испр. и доп.-М.: Физматлит, 2002.-496 с.
91. Райфа Г. Анализ решений / Г. Райфа.-М.:Наука, 1977.-402 с.
92. Роб П. Системы баз данных: проектирование, реализация и управление. / П. Роб.–5-е изд.–СПб.: БХВ-Петербург, 2004.
93. Робинсон С., Корнес О. и др. C# для профессионалов, в 2-х томах. / - Пер. с англ. – М.: Лори, 2005. – 1002 с.

94. Романовский В. И. Математическая статистика / В.И. Романовский. –М.-Л.: ОНТИ, 1938.-528 с.
95. Саати Т. Аналитическое планирование. Организация систем / Т. Саати, К. Кернс; пер. с англ.-М.:Мир, 1991.-224 с.
96. Самарский А. А. Математическое моделирование: Идеи. Методы. Примеры. / А.А. Самарский, А.П. Михайлов.-2. изд., испр.-М.: Физматлит, 2002.-316 с.
97. Сергиенко И.В. Математические модели и методы решения задач дискретной оптимизации / И.В. Сергиенко.-Киев: Наук. Думка, 1988.-471 с.
98. Сингх М. Системы: декомпозиция, оптимизация и управление / М. Сингх, А. Титли: Пер. с англ.-М.:Машиностроение, 1986.-496 с.
99. Системы управления и информационные технологии: Межвуз. сб. науч. трудов. – Воронеж: Изд-во Воронеж. техн.ун-та, 1998.-200 с.
100. Скотт К. UML: Основные концепции / К. Скотт; пер с англ. О.А. Лещинского; под ред. А.Ю. Шелестова.–М.: Вильямс, 2002.–138 с.
101. Смоляр Л. И. Оперативно-календарное планирование: модели и методы / Л.И. Смоляр.-М.: Экономика, 1979.-135 с.
102. Соболь И. М. Выбор оптимальных параметров в задачах со многими критериями / И.М. Соболь, Р.В. Статников.-М.:Наука, 1981.-111 с.
103. Солодовникова О. С. Информационные технологии в задачах составления расписания в ВУЗах / О.С. Солодовникова, В.С. Солодовников // Кабардино-Балкарский университет, Нальчик. – (http://www.mnogosmenka.ru/drugoe/ telematika.htm).
104. Стивенс Р. Программирование баз данных / Р. Стивенс: Пер. с англ.; Под ред. С.М. Молявко.-М.: Бином, 2003.-383 с.
105. Сухарев А. Г. Курс методов оптимизации / А.Г. Сухарев, А.В. Тихонов, В.В. Федоров.-М.: Наука, 1986.-325 с.
106. Сысоев В.В. Системное моделирование. / В.В.Сысоев. – Воронеж: ВГТА, 2000. – 74 с.

107. Танаев В. С. Теория расписаний. Многостадийные системы / В.С. Танаев, Ю.Н. Сотсков, В.А. Струсевич. – М.:Наука, 1989.-327 с.
108. Танаев В. С. Теория расписаний. Одностадийные системы / В.С. Танаев, В.С. Гордон, Я.М. Шафранский. – М.: Наука, 1984.-381 с.
109. Танаев В.С. Введение в теорию расписаний. / В.С. Танаев, В.В. Шкурба – М.:Наука, 1975. – 256 с.
110. Танаев В.С. Теория расписаний./ В.С. Танаев. – Знание. 1988. №2. - 32с.
111. Таха Х. Введение в исследование операций:В 2-х кн.Кн.1. / Х. Таха:пер. с англ.-М.:Мир, 1985.-479 с.
112. Таха Х. Введение в исследование операций:В 2-х кн.Кн.2. / Х. Таха:пер. с англ.-М.:Мир, 1985.-496 с.
113. Телло Э. Р. Объектно-ориентированное программирование в среде Windows / Э.Р. Телло: Пер. с англ. Д.М. Арапова, А.К. Петренко.-М.: Высш.шк., 1993.-347 с.
114. Троелсен Э. C# и платформа .NET. Библиотека программиста. — СПб.: Питер, 2002. — 796 с.
115. Трофимов С.А. CASE-технологии: Практическая работа в Rational Rose / С. А. Трофимов.–2-е изд.–М.: Бином-Пресс, 2002.–288 с.
116. Турчак Л. И. Основы численных методов / Л.И. Турчак.-М.:Наука, 1987.-318 с.
117. Фаулер М. UML в кратком изложении: Применение стандартного языка объектного моделирования / М. Фаулер, К. Скотт; Пер. с англ. А.М. Вендрова; Под ред. Л.А. Калиниченко.-М.: Мир, 1999.-191 с.
118. Федотова Д.Э. CASE-технологии / Д.Э. Федотова, Ю.Д. Семенов, К.Н. Чижик.–М.: Горячая линия-Телеком, 2003.–157 с.
119. Фишберн П. С. Теория полезности для принятия решений / П.С. Фишберн.-М.:Наука, 1978.-352 с.
120. Фридман А.Л. Основы объектно-ориентированной разработки программных систем / А.Л. Фридман.–М.: Финансы и статистика, 2000.–190 с.

121. Фролов А. В., Фролов Г. В. Язык C#. Самоучитель. — М.: Диалог-МИФИ, 2003. — 560 с.
122. Хансен Г. Базы данных: разработка и управление / Г. Хансен, Дж. Хансен: Пер. с англ.; Под ред. С. Каратыгина.–М.: Бином, 1999.–699 с.
123. Харрингтон Д. Проектирование объектно-ориентированных баз данных: Эволюция технологий хранения информации / Д. Харрингтон.: Пер. с англ.-М.: ДМК Пресс, 2001.-269 с.
124. Хахулин Г. Ф. Постановка и методы решения задач дискретного программирования: Учеб. пособие / Г.Ф. Хахулин.-М.: МАИ, 1992.-59 с.
125. Чекалов А. П. Базы данных: от проектирования до разработки приложений / А.П. Чекалов.-СПб.: БХВ-Петербург, 2003.-380 с.
126. Чери С. Логическое программирование и базы данных / С. Чери, Г. Готлоб, Л. Танка: Пер. с англ.; под ред. Л.А.Калиниченко.-М.:Мир, 1992.-352 с.
127. Чудаков А.Д. Автоматизированное оперативно-календарное планирование в гибких комплексах механообработки / А. Д. Чудаков, Б. Я. Фалевич.–М.:Машиностроение, 1986. – 222 с.
128. Шаллоуей А. Шаблоны проектирования: Новый подход к объектно-ориентированному анализу и проектированию / А. Шаллоуей, Трот Дж. Р.-М.: Изд. дом “Вильямс”, 2002.-288 с.
129. -Шилдт Г. C#: учебный курс. — СПб.: Питер, 2002. — 512 с.: ил.
130. Шилдт Г. Полный справочник по C#. — М.: Издательский дом «Вильямс», 2004. — 752 с.
131. Шрейдер Ю. А. Системы и модели / Ю.А. Шрейдер, А.А. Шаров.-М.:Радио и связь, 1982.-152 с.
132. Эддоус М. Методы принятия решений / М. Эддоус, Р. Стэнсфилд: Пер. с англ.-М.:Аудит, 1997.-590 с.
133. Энкарначчо Ж. Автоматизированное проектирование. Основы понятия и архитектура систем./ Ж. Энкарначчо, Э. Шлехтендаль. -М.:Радио и связь, 1986.-288 с.

ПРИЛОЖЕНИЕ

РОССИЙСКАЯ ФЕДЕРАЦИЯ

СВИДЕТЕЛЬСТВО

о государственной регистрации программы для ЭВМ

№ 2012618248

«Разработка системы составления расписания ВУЗа Ирака»

Правообладатель(ли): ***федеральное государственное бюджетное образовательное учреждение высшего профессионального образования «Воронежский государственный университет» (RU)***

Автор(ы): ***Астахова Ирина Федоровна, Асвад Фирас М. (RU)***

Заявка № **2012615867**

Дата поступления **12 июля 2012 г.**

Зарегистрировано в Реестре программ для ЭВМ

11 сентября 2012 г.

Руководитель Федеральной службы по интеллектуальной собственности

Б.П. Симонов