Тимешов А.С., Раскатова М.В. Формирование модели генетического алгоритма для решения задачи составления расписания занятий в школе // Международный журнал информационных технологий и энергоэффективности. – 2022. – Т.7 №2(24) с.32-39

Международный журнал информационных технологий и энергоэффективности

Сайт журнала:
http://www.openaccessscience.ru/index.php/ijcse/

УДК 004.896

ФОРМИРОВАНИЕ МОДЕЛИ ГЕНЕТИЧЕСКОГО АЛГОРИТМА ДЛЯ РЕШЕНИЯ ЗАДАЧИ СОСТАВЛЕНИЯ РАСПИСАНИЯ ЗАНЯТИЙ В ШКОЛЕ

Тимешов А.С., Раскатова М.В.
Федеральное государственное бюджетное образовательное учреждение высшего образования «Национальный исследовательский университет «МЭИ», Россия (111250, г.Москва, ул. Красноказарменная, д.14); e-mail: chel.ed@yandex.ru

В статье рассматривается проектирование модели генетического алгоритма, решающего задачу составления расписания занятий, и ее программная реализация. Программная реализация написана с использованием языка программирования Python и библиотеки Deap.

Ключевые слова: генетические алгоритмы, модель алгоритма, расписание занятий, школа, библиотека Deap

A MODEL OF A GENETIC ALGORITHM FORMATION FOR SOLVING THE PROBLEM OF SCHOOL SCHEDULES

Timeshov A.S., Raskatova M.V.
National Research University “Moscow Power Engineering Institute”, Russia (111250, Moscow, Krasnokazarmennaya street, 14); e-mail: chel.ed@yandex.ru

The article discusses the design of a genetic algorithm model that solves the problem of scheduling classes, and its software implementation. The software implementation is written using the Python programming language and the Deap library.

Keywords: genetic algorithms, algorithm model, decision evaluation strategy, school schedule, school, Deap library.

Человечество на протяжении всей своей истории сталкивалось с необходимостью составления 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 [1].

Каждый человек, зная, к какому сроку требуется выполнить определенное действие и, предполагая, как много времени и ресурсов потребуется на его выполнение, занимается планированием своего личного расписания. Чаще всего составление такого расписания не вызывает трудностей. Они возникают лишь в том случае, когда задач становится много, когда появляется необходимость принять во внимание множество дополнительных факторов и условий, когда появляется необходимость составления расписания не для одного человека, а

для целого коллектива[2]. Таким коллективом могут быть участники учебного процесса в учебном заведении - преподаватели и учащиеся.

Составление расписания занятий в учебном заведении является очень трудоёмкой задачей, потому как интересы участников учебного процесса многообразны, а все факторы, которые могут повлиять на расписание, практически невозможно учесть. Поэтому правильно подобранный алгоритм составления расписания позволит наиболее рационально и эффективно распределить рабочее время и ресурсы учебного процесса. Некоторыми из возможных вариантов решения подобной задачи являются применения таких алгоритмов как: алгоритм метода раскраски графа, алгоритм градиентного спуска, жадные алгоритмы и генетические алгоритмы.

Генетический алгоритм – это последовательность управляющих действий и операций, имитирующая эволюционные процессы в природе такие как: скрещивание, естественный отбор и мутация, для решения задач поиска и оптимизации.

В генетическом алгоритме потенциальное решение задачи, в данном контексте, это расписание занятий, представляется хромосомой, т.е. в закодированном виде. И каждая такая хромосома несет в себе набор генов, др. словами элементы решения или его свойства.

Генетический алгоритм постепенно развивает популяцию, то есть набор потенциальных решений задачи. Эти решения называются индивидуумами или особями, они итеративно оцениваются и используются для создания нового поколения индивидуумов [3].

Особи, которые получили наивысшую оценку, с большей вероятностью могут пройти отбор и, скрещиваясь с другими особями, передать свои свойства будущему поколению. Таким образом, потенциальные решения задачи постепенно эволюционируют и в конечном итоге, находится лучшее решение [4]. Периодически случайным образом популяция обновляется, меняются сочетания генов в хромосомах, то есть происходит мутация. Целью мутации является стимулирование алгоритма на поиск неисследованных областей потенциальных решений.

Общую схему работы генетического алгоритма можно представить в виде схемы, приведенной на рисунке 1.

Рисунок 2 - Общая схема работы генетического алгоритма

В каждом учебном заведении можно выделить следующие множества объектов:

- Учебные классы (множество C = $\{c_0, c_1, \ldots, c_{k-1}\}$, где c_i – учебный класс, $0 \leq i \leq k-1$, k - количество учебных классов в школе).
- Учителя (множество T = $\{t_0, t_1, \ldots, t_{l-1}\}$, где t_i - учитель, $0 \leq i \leq l-1$, l - количество учителей в школе).
- Дисциплины (множество D = $\{d_0, d_1, \ldots, d_{m-1}\}$, где d_i - дисциплина, $0 \leq i \leq m-1$, m – количество дисциплин, преподаваемых в школе).
- Аудитории (множество A = $\{a_0, a_1, \ldots, a_{n-1}\}$, где d_i - дисциплина, $0 \leq i \leq n-1$, n – количество аудиторий в школе).

- Временные интервалы проведения учебных занятий (уроков) (множество L = $\{l_0, l_1, \ldots, l_{p-1}\}$, где l_i – временной интервал урока, $0 \leq i \leq p-1$, p – количество уроков, доступных для проведения занятий за одну неделю для всей школы).

Чтобы решить задачу, важно определить ключевые множества, для которых будет строиться решение, и которые обязательно нужно включить в структуру гена.

В случае если в учебном заведении каждому учителю присвоена отдельная аудитория, а наименование дисциплин не имеет значения для составления расписания, важно только сопоставить учителей и учебные классы в определенный момент времени, то в качестве обязательных множеств, необходимых для построения гена, можно выбрать учителей, учебные классы и уроки.

Каждому классу, учителю и уроку может быть присвоен свой номер. Таким образом, формируя каждый ген в виде последовательности {T, C, L}, где T - номер учителя, C - номер класса, L - номер урока, получается следующая модель хромосомы, представленная на рисунке 2.

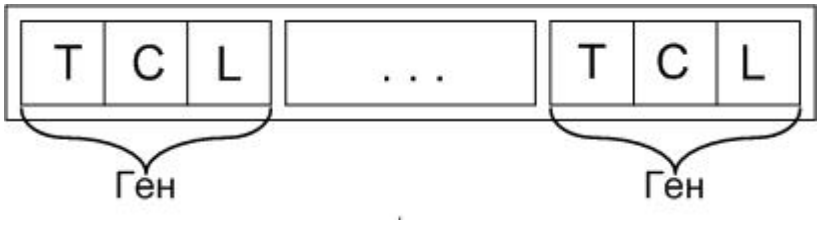

Рисунок 2 - Модель хромосомы

В данной хромосоме каждый ген несет в себе всю необходимую информацию для его корректного декодирования, а его расположение в хромосоме не имеет абсолютно никакого значения, что позволяет применить для данной модели большее число возможных моделей скрещивания.

Начальная популяция решений является отправной точкой, откуда начинается поиск решения. От того, насколько удачно она составлена, может зависеть время, которое будет затрачено на поиск, и качество конечного результата. Поскольку связь между учителем и классом является статической и не может каким-либо образом разрываться или меняться в процессе всей работы алгоритма, то единственной составляющей гена, которая может хоть как-то изменяться, является номер урока.

В общем случае количество допустимых уроков в неделю (которое можно выделить на проведение учебных занятий) для каждого класса в общеобразовательном учреждении равно 40 (8 уроков на каждый будний день), то для начальной популяции достаточно будет создать расписания, в каждом из которых будет только один из сорока возможных номеров урока для всех генов. Очевидно, что такие расписания будут неприемлемыми, но такой подход способен охватить сразу всю область поиска уже на начальном этапе работы алгоритма. Модель начальной популяции решений представлена на рисунке 3.

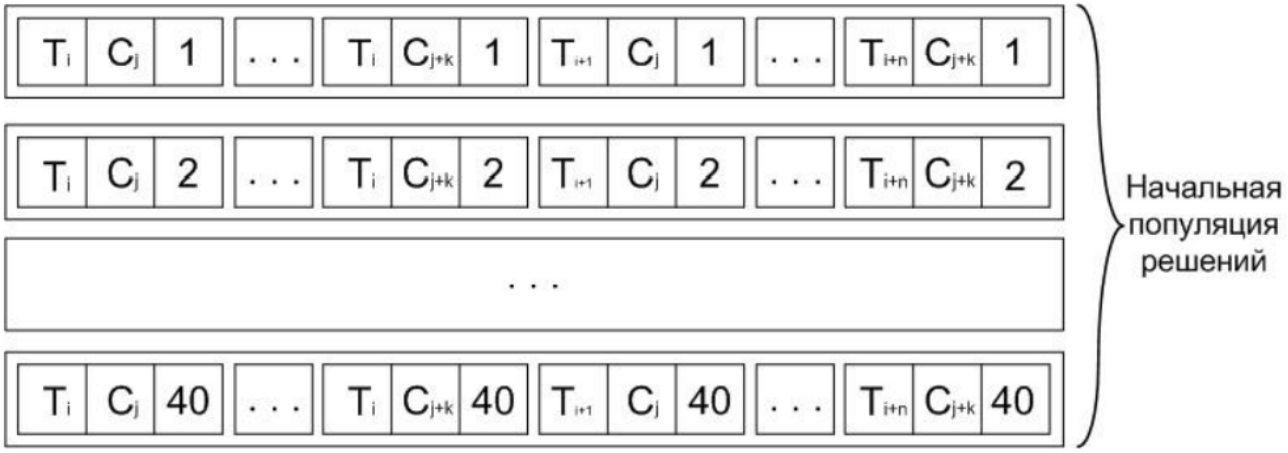

Рисунок 3 - Начальная популяция решений

Где T_i – номер учителя, 0 ≤ i ≤ n, где n – количество учителей в школе – 1, C_j – номер класса, 0 ≤ j ≤ k, где k – количество учителей в школе – 1.

Такая стратегия выбора начальной популяции чревата тем, что в процессе эволюции потенциально хорошие номера уроков могут исчезать, поэтому крайне важно хотя бы на ранних поколениях с некоторой периодичностью добавлять в текущую популяцию решений значения из начального поколения. Помимо этого в целях предотвращения сходимости алгоритма к локальному экстремуму имеет смысл со временем изменять значение вероятности возникновения мутации, которая представляет собой случайную замену номеров уроков всех занятий, проводимых каким-либо учителем.

В процессе формирования модели алгоритма следует учитывать и тот факт, что нарушение определенных правил недопустимо ни в коем случае. Такие правила называются жесткими ограничениями. В контексте рассматриваемой задачи это могут быть такие правила как: отсутствие разрывов между занятиями для учащихся, отсутствие коллизий в расписании и др. А пожелания участников учебного процесса, например, отсутствие разрывов между занятиями у учителей, можно считать мягкими ограничениями. Алгоритм будет стремиться выполнить эти требования и минимизировать число их нарушений, тем не менее, их наличие не будет оцениваться алгоритмом как недопустимое решение.

Существует несколько стратегий оценки приспособленности решений [5], например:

- Найти такое представление хромосомы, которое полностью или частично исключало бы саму возможность нарушения ограничений. Это существенно бы упростило решение задачи, но в рамках задачи по составлению расписания занятий такого представления нет.
- В процессе оценки решений отбрасывать нарушающие хотя бы одно жесткое ограничение. Такой подход может привести к возникновению ситуации, при которой ценная информация, содержащаяся в этих решениях, может быть навсегда утеряна.
- В процессе оценки решений исправлять те, которые нарушают хотя бы одно жесткое ограничение. Такой подход также может привести к потере ценной информации.
- В процессе оценки решений штрафовать те решения, которые нарушают жесткие ограничения. Такой подход меняет жесткое ограничение на мягкое, но с более высоким штрафом. При этом он лишь делает оценку плохих решений хуже, не

исключая полностью их из рассмотрения. Получается, что такие решения становятся менее желательными, но, тем не менее, информация, хранящаяся в них, не теряется. Единственной сложностью такого подхода является определение модели назначения штрафов, поскольку неправильно подобранный штраф может привести к тому, что недопустимое решение либо окажется оптимальным, либо полностью исключится из рассмотрения, то очень важно подобрать правильные значения штрафов за каждое нарушение жестких ограничений.

Так как рассматриваемая задача имеет большое количество параметров, критериев и условий, то решение задачи расположено в огромном пространстве потенциальных решений и, следовательно, необходимо использовать максимально широкую область поиска. Потеря ценной информации будет существенна и только сузит эту область. Поэтому целесообразно при решении задачи выбрать последний подход.

Чрезвычайно важно не допускать излишнюю потерю потенциально хороших генов во время эволюции и стараться давать возможность особям с низкой приспособленностью проходить отбор и, скрещиваясь с другими индивидуумами, в том числе и с наиболее приспособленными, передавать свою генетическую информацию новым поколениям. Для выполнения данной цели как раз подойдут такие методы отбора как: случайный, турнирный отбор, отбор по правилу рулетки, стохастическая универсальная выборка и ранжированный отбор. Эти виды селекции как раз подразумевают подобную особенность эволюции.

Хотя перечисленные методы селекции позволяют сохранить наличие некоторого количества потенциально хороших генов в процессе эволюции, в любой момент эволюции может произойти такая ситуация, при которой лучшие решения в текущем поколении не пройдут отбор и исчезнут. Потеря, скорее всего, будет временная, но возникновение таких потерь может увеличить время поиска лучшего решения может, а гарантия «возвращения» утерянных решений, на самом деле, отсутствует. Для того чтобы избежать подобных ситуаций, модель алгоритма можно реализовать с использованием стратегии элитизма. Суть данной стратегии заключается в том, что какое-то небольшое, заранее заданное число копий лучших (элитных) индивидуумов обязательно, до этапов отбора, скрещивания и мутации, переходит в будущее поколение [7]. Схема эволюционного цикла алгоритма с применением стратегии элитизма представлена на рисунке 4.

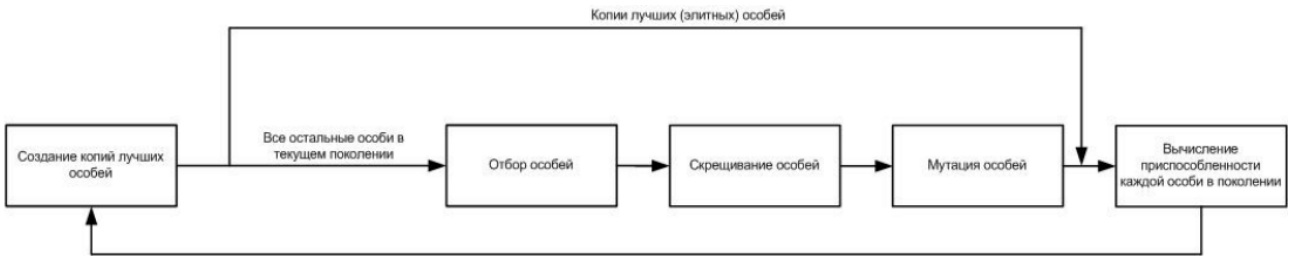

Рисунок 4 - Эволюционный цикл алгоритма с применением стратегии элитизма

В качестве модели скрещивания для выбранной модели хромосомы можно применить следующие методы скрещивания: одноточечное, двухточечное и равномерное скрещивание [6]. Перечисленные виды рекомбинации не нарушают структуру хромосомы и не создают

недопустимые гены в процессе своей работы. При этом они способны поддерживать многообразие комбинаций генов для всей популяции решений.

На рисунке 5 представлена зависимость максимальной и средней приспособленности от поколения для выбранной модели алгоритма с турнирным методом отбора и размером турнира 2, и равномерным скрещиванием. Красной линией представлена зависимость максимальной приспособленности, зелёной – зависимость средней. Модель генетического алгоритма была программно реализована с применением языка программирования Python и библиотеки Deap.

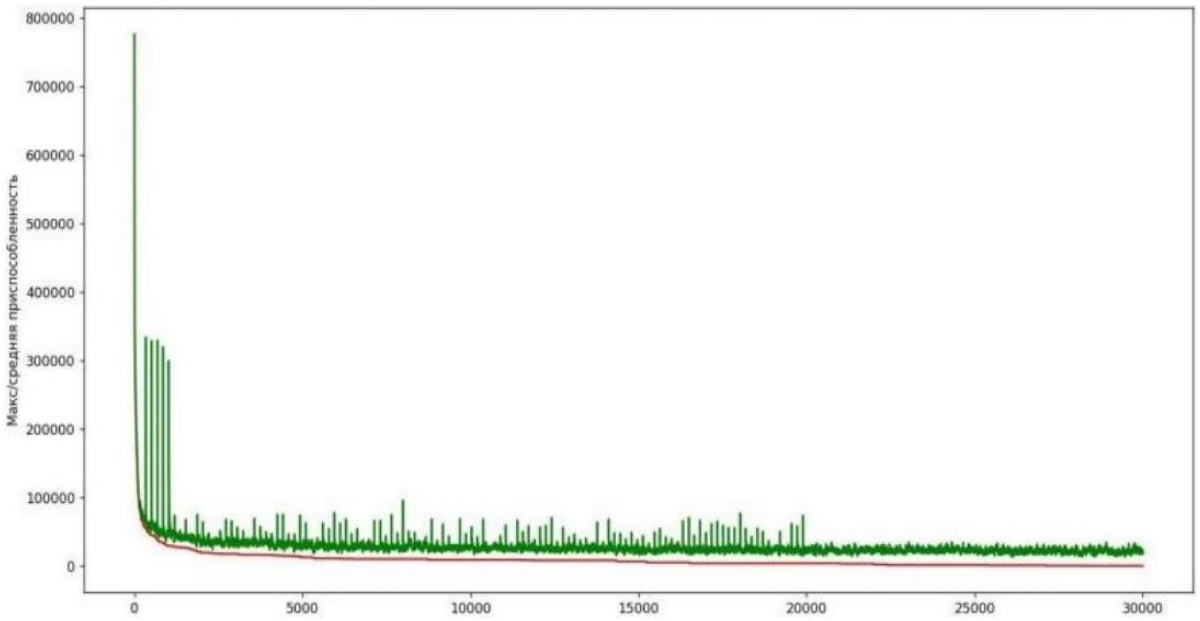

Рисунок 5 - График эволюции генетического алгоритма

Из графика видно, что алгоритм достаточно быстро преобразует начальное поколение, состоящее из решений, насчитывающих огромное количество коллизий, в решения с различными комбинациями номеров уроков. Но такому быстрому спуску графика сопутствует потеря различного числа номеров занятий для некоторых пар учитель-класс, вследствие чего поиск замедляется. На графике заметны резкие скачки зеленой линии – это участки эволюции, на которых проводилась процедура добавления в популяцию генотипов из начального поколения, а также промежутки увеличения значения графика средней приспособленности - повышение вероятности возникновения мутации для поддержания многообразия генов. Но буквально через несколько поколений после таких операций средняя приспособленность решений возвращалась к своему прежнему уровню, а максимальная улучшалась. Это свидетельствует о том, что выбранная стратегия не ухудшает популяцию, а вносит в нее утерянные в процессе эволюции комбинации генов, тем самым расширяя область поиска. Алгоритм, преодолевая преграды, возникшие на его пути, борется со стагнацией прогресса, поддерживает многообразие популяции потенциальных решений, избегает преждевременной сходимости к локальному экстремуму и находит решение за приемлемое время.

Разработанная коллективом авторов модель генетического алгоритма позволяет решать задачу составления расписания занятий в школе. Применение такого алгоритма для решения подобных задач позволит существенно сократить загруженность организаторов учебного

процесса и оптимизировать их рабочее время, а учебный процесс, построенный на основе такого расписания, будет протекать непрерывно и с оптимальными нагрузками на учащихся. Однако при формировании модели генетического алгоритма следует учитывать тот факт, что применение генетических алгоритмов на практике влечет за собой большой объем счетных операций и вероятность преждевременной сходимости к локальному экстремуму.

Данная модель может быть оптимизирована для различных учебных заведений путем добавления в представление хромосомы различных компонент, таких как наименование дисциплин и номера учебных аудиторий и других, в случае, если их влияние на составление расписание существенно.

Список литературы

1. Танаев, В.С. Введение в теорию расписаний / В.В., Шкурба – М.: Наука, 1975. – 256с.
2. Танаев, В.С. Теория Расписаний. Многостадийные системы / Ю.Н., Сотсков, В.А., Струсевич – М.: Наука, 1989. – 327с.
3. Вирсански, Э. Генетические алгоритмы на Python – М.: ДМК Пресс, 2020. – 286 с.
4. Джоши, П. Искусственный интеллект с примерами на Python / – СПб.: ООО «Диалектика», 2019. – 448с.
5. Рутковская, Д. Нейронные сети, генетические алгоритмы и нечеткие системы / М. Пилиньский, Л. Рутковский. – М.: Горячая линия - Телеком, 2006. – 452 с.
6. Т.В., Панченко Генетические алгоритмы / под ред. Ю. Ю., Тарасевича. – Астрахань: Издательский дом «Астраханский университет», 2007. – 87 с.
7. Amita Kapoor. Hands-On Artificial Intelligence for IoT. – Packt Publishing Ltd., 2019. – 370с.
8. Deap Documentation [Электронный ресурс] / Сайт с документацией по библиотеке Deap. URL: https://deap.readthedocs.io/en/master/index.html

References

1. Tanaev, V.S. Introduction to scheduling theory / V.V., Shkurba – M.: Science, 1975. – 256p.
2. Tanaev, V.S Schedule Theory. Multistage systems / Y.N., Sotskov, V.A., Strusevich - M.: Science, 1989. - 327p.
3. Wirsansky, E. Hands-On Genetic Algorithms with Python – M.: DMK Press, 2020. – 286 p.
4. Joshi, P. Artificial intelligence with examples in Python / - St. Petersburg: Dialectika, 2019. - 448 p.
5. Rutkovskaya, D. Neural networks, genetic algorithms and fuzzy systems / M. Pilinsky, L. Rutkovsky. - M.: Hot line - Telecom, 2006. - 452 p.
6. T.V., Panchenko Genetic algorithms / ed. Y. Y., Tarasevich. - Astrakhan: Astrakhan University Publishing House, 2007. - 87 p.
7. Amita Kapoor. Hands-On Artificial Intelligence for IoT. – Packt Publishing Ltd., 2019. – 370p.
8. Deap Documentation [Electronic resource] / Deap library documentation site. URL: https://deap.readthedocs.io/en/master/index.html