

УДК 004

**ИССЛЕДОВАНИЕ ПРИМЕНЕНИЯ ГЕНЕТИЧЕСКИХ АЛГОРИТМОВ  
ДЛЯ ГЕНЕРАЦИИ ПРОГРАММНОГО КОДА РЕШЕНИЯ ЗАДАЧИ  
КОММИВОЯЖЁРА**

Томашевский Н.А., студент гр. ИС-2 09.03.02, III курс

Научный руководитель: Значко В.Н., старший преподаватель

Кубанский государственный университет

г. Краснодар

Оптимизационные задачи – задачи, заключающиеся в нахождении значений некоторых входных параметров заданной целевой функции, при которых целевая функция достигает значения экстремума. Существует ряд оптимизационных методов, предназначенных для решения оптимизационных задач. Однако все методы имеют ряд проблем, например, объём времени, тратящийся на процесс вычислений, для наиболее точных и применимых методов, как правило, слишком велик, а при получении результата и нахождении экстремума не всегда можно быть уверенным, что экстремум является глобальным. По этой причине проводится поиск новых оптимизационных алгоритмов.

Генетические алгоритмы (ГА) — это стохастические методы поиска, основанные на теории эволюции Ч. Дарвина. ГА осуществляет поиск экстремального значения функции путём случайного подбора, комбинирования искомых параметров и успешно реализуют принципы естественного отбора и наследования.

В данной работе ГА будут рассмотрены в задаче генерации программного кода, используемого для решения классической замкнутой задачи коммивояжёра [1,3]. ГА будет работать над популяцией программных кодов, которые должны решать задачу коммивояжёра. Для упрощения генерации программного кода он будет представлен последовательностью инструкций на разработанном специализированном байт-коде. Таким образом, хромосомы будут наборами команд байт-кода вместе с аргументами; каждая особь будет исполняться как рабочая программа.

Спецификация байт-кода и исходный текст итоговой программы на алгоритмическом языке C++ представлен по адресу [https://github.com/MrTomashevsky/genetic\\_alg\\_konsole.git](https://github.com/MrTomashevsky/genetic_alg_konsole.git).

Входными данными у программы был набор графов, а выходными – получаемое при помощи генетического алгоритма программное решение на байт-коде, следование которому при обходе графа приведет к решению задачи коммивояжера.

Алгоритм просмотра одной особи:

- 1) Передаем особи ссылку на новый граф в массиве графов;
- 2) Интерпретируем программный код особи для этого графа;
- 3) Особь возвратила массив из вершин графа, проход по которым особь предполагает решением задачи коммивояжера. Проходимся по данным вершинам, суммируя пути, получаем общий путь;
- 4) Записываем в массив путей полученный общий путь. Чем ближе длина пути, рассчитанного особью, к лучшей длине пути, тем больше функция приспособленности;
- 5) Если не просмотрены все графы, то переход на пункт 1.

Исследование проводилось для популяций в 50, 100, 150, 200 особей. Применялся 1, 2, 3, 4 и 5-точечный кроссинговер. Использовались следующие операторы выбора родителей: панмиксия, турнирный отбор, рулеточный отбор, инбридинг, аутбридинг [2]. Вероятность мутации равнялась 10, 20 и 30%: в случайном месте хромосомы особи менялась или удалялась команда, либо вставлялась новая. Начальная популяция создавалась случайно: каждая особь собиралась с нуля, генерируя программный код на байт-коде случайным образом. Любая сгенерированная особь гарантированно синтаксически верна: запускаемый сгенерированный программный код не обращается к недоступным ячейкам памяти, не путает типы данных и т.д.

Слишком малое количество точек в кроссинговере вело к очень быстрому затуханию популяции, так как хромосомы родителей недостаточно радикально перемешивались, в результате дети этих родителей оказывались слишком похожими на родителей. По этой причине далее рассматриваться не будут результаты исследования при применении 1- и 2-точечных кроссинговеров.

Было проведено несколько десятков запусков программы с различными входными данными. Далее на рисунках будут приведены графики изменения популяции пошагово при разных входных данных алгоритма (под графиком указаны следующие величины, использованные в качестве входных данных:  $s$  – величина начальной популяции индивидуумов,  $stos$  – количество точек в кроссинговере,  $pm$  – шанс мутации в процентах).

На рисунках 1, 2, 3, 4 представлены графики изменения популяций пошагово при начальной популяции в 50, 100, 150 и 200 индивидуумов с использованием 3-точечного кроссинговера. При таких входных данных при любом выбранном операторе выбора родителей популяция очень быстро увядала; популяция колебалась лишь при инбридинге.

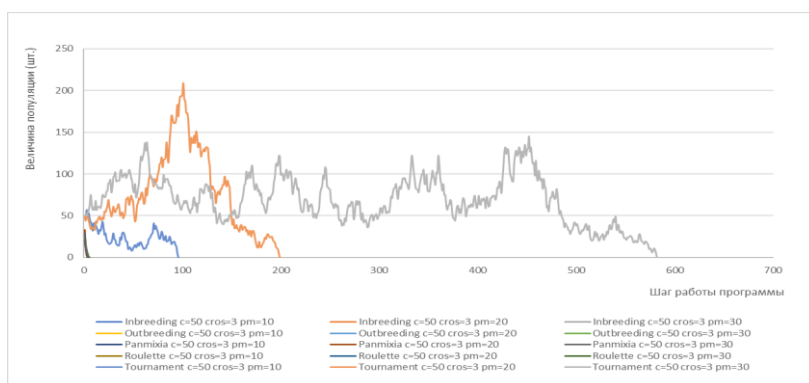


Рисунок 1 – график изменения популяций пошагово при начальной популяции в 50 индивидуумов с использованием 3-точечного кроссинговера

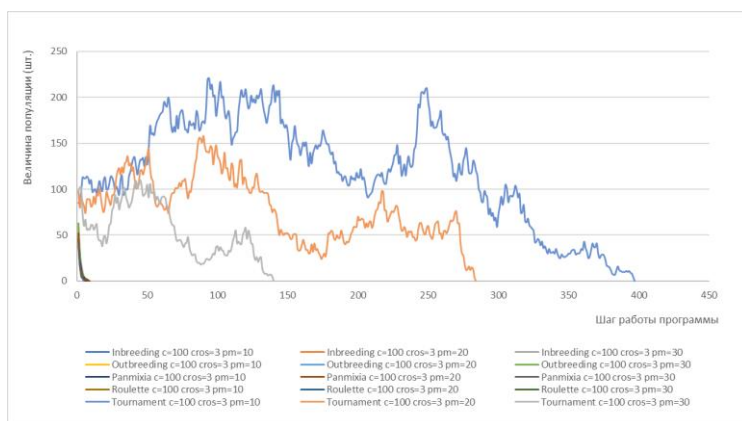


Рисунок 2 – график изменения популяций пошагово при начальной популяции в 100 индивидуумов с использованием 3-точечного кроссинговера

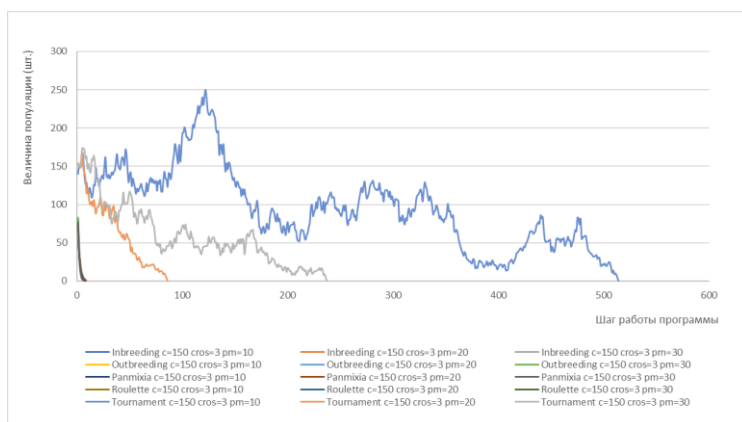


Рисунок 3 – график изменения популяций пошагово при начальной популяции в 150 индивидуумов с использованием 3-точечного кроссинговера

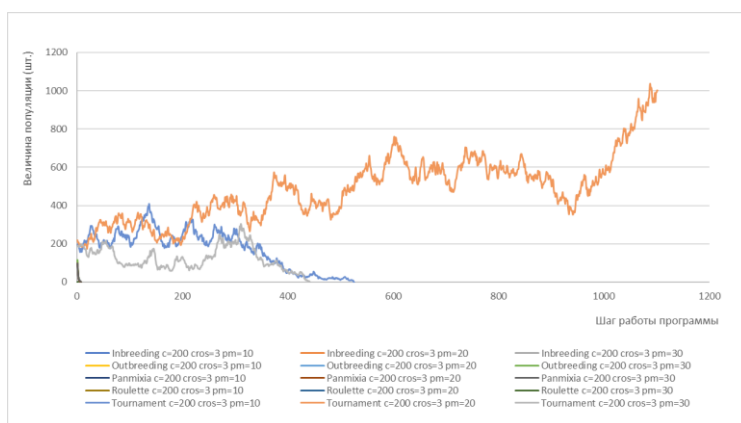


Рисунок 4 – график изменения популяций пошагово при начальной популяции в 200 индивидуумов с использованием 3-точечного кроссинговера

На рисунках 5, 6, 7, 8 представлены графики изменения популяций пошагово при начальной популяции в 50, 100, 150 и 200 индивидуумов с использованием 4-точечного кроссинговера. При таких входных данных при любом выбранном операторе выбора родителей популяция увядала ещё быстрее, чем при применении 3-точечного кроссинговера. Однако при турнирном отборе популяция быстро увеличивалась. Вероятнее всего, популяция заполнялась очень похожими между собой особями, произошедшими от нескольких самых лучших в начале исследования. После достижения размера популяции больше 500000 индивидуумов наблюдение останавливалось.

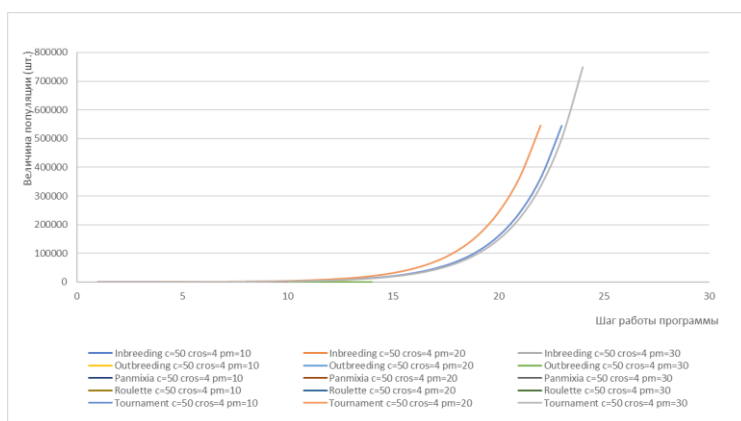


Рисунок 5 – график изменения популяций пошагово при начальной популяции в 50 индивидуумов с использованием 4-точечного кроссинговера

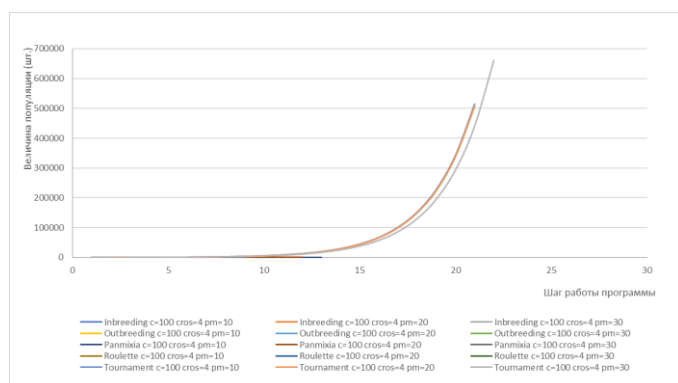


Рисунок 6— график изменения популяций пошагово при начальной популяции в 100 индивидуумов с использованием 4-точечного кроссинговера

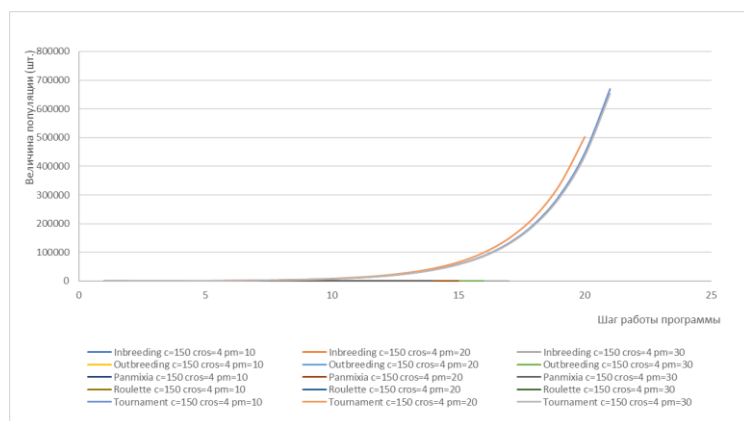


Рисунок 7 — график изменения популяций пошагово при начальной популяции в 150 индивидуумов с использованием 4-точечного кроссинговера

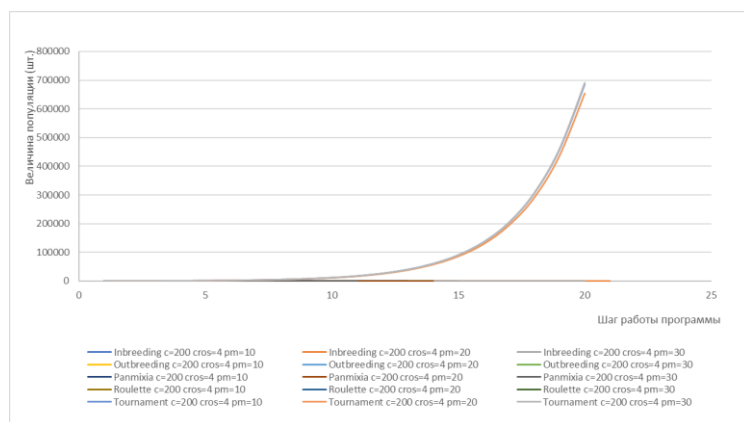


Рисунок 8 — график изменения популяций пошагово при начальной популяции в 200 индивидуумов с использованием 4-точечного кроссинговера

На рисунках 9, 10, 11, 12 представлены графики изменения популяций пошагово для разных входных данных при начальной популяции в 50, 100, 150 и 200 индивидуумов с использованием 5-точечного кроссинговера. При таких входных данных при турнирном отборе популяция увеличивалась ещё быстрее (поэтому, чтобы пронаблюдать поведение популяции для других операторов выбора родителей, поведение популяции для турнирного отбора на графике представлено только на первых шагах популяции). При рулеточном отборе популяция умирала фактически на первых двух шагах популяции: это следствие того, что для рулетного отбора желательно наличие в популяции индивидуумов с самого старта большим значением функции приспособленности, что маловероятно при случайной генерации начальной популяции.

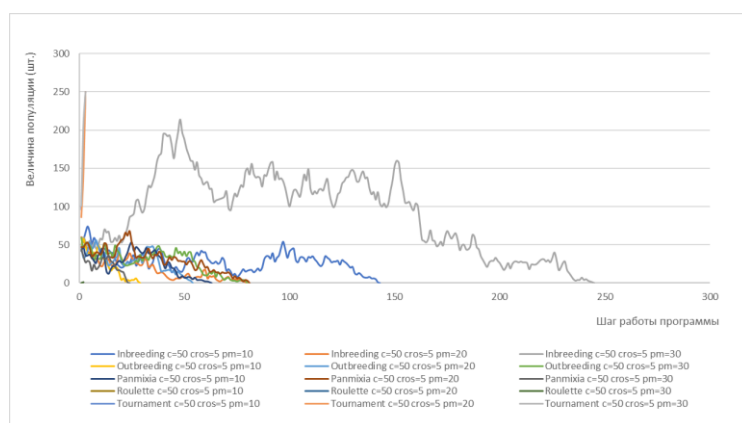


Рисунок 9 – график изменения популяций пошагово при начальной популяции в 50 индивидуумов с использованием 5-точечного кроссинговера

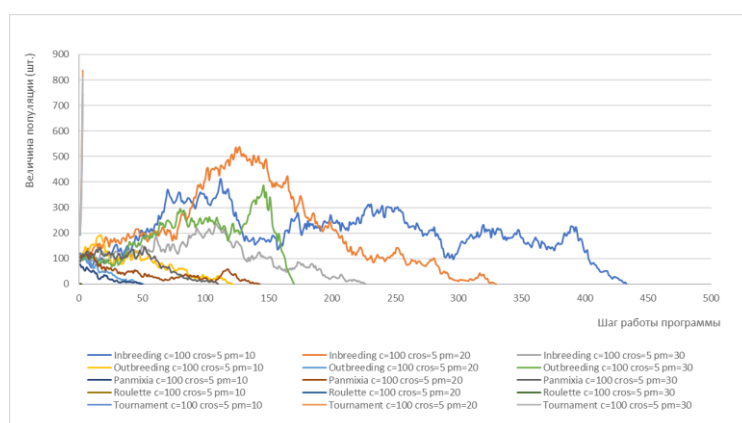


Рисунок 10 – график изменения популяций пошагово при начальной популяции в 100 индивидуумов с использованием 5-точечного кроссинговера

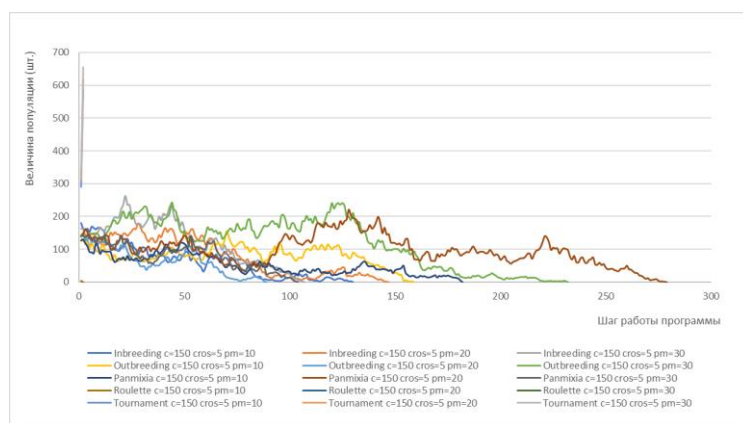


Рисунок 11 – график изменения популяций пошагово при начальной популяции в 150 индивидуумов с использованием 5-точечного кроссинговера

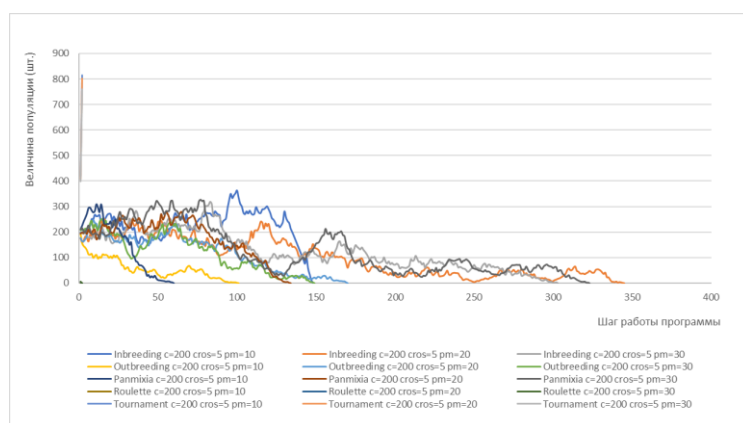


Рисунок 12 – график изменения популяций пошагово при начальной популяции в 200 индивидуумов с использованием 5-точечного кроссинговера

Для остальных операторов родителей популяции активно колебались, притом особенно заметно, что при аутбридинге и инбридинге популяция могла на разных шагах то увеличиваться вчетверо относительно изначальной популяции, то быстро уменьшаться вполовину. При панмиксии исследование проходило очень быстро, тогда как инбридинг и аутбридинг могли увязнуть на одном шаге на более долгое время.

## Заключение

В ходе исследования используемый метод генерации особей при помощи ГА столкнулся со следующими проблемами:

- В начальной популяции у особей не всегда генерируются команды, взаимодействующие с массивом вершин. Такие особи большей частью отбраковываются, но какая-то часть даёт потомство. Неудачные особи - норма для ГА, но в данном исследовании они намного сильнее влияют на популяцию, так как особи являются программами.

- Алгоритм работает с длинными сгенерированными программами, и кроссинговер часто создаёт из них синтаксически неверные. Специально для минимизации данной проблемы и был разработан байт-код, но алгоритм все равно тратит весомую часть времени на недееспособные особи, так как на каждом шаге их рождается много.

Из-за этих проблем в результате исследования не было получено особей с рабочим программным кодом на байт-коде, решающим задачу коммивояжёра. Выбранный метод генерации программ признан недееспособным. В нулевом поколении необходимы осмысленные рабочие особи для того, чтобы алгоритм возвращал какие-то результаты.

### Список литературы

1. Семенов, С. С. Анализ трудоемкости различных алгоритмических подходов для решения задачи коммивояжера [Текст] / С. С. Семенов, А. В. Педан, В. С. Воловиков, И. С. Климов // Системы управления, связи и безопасности. – 2017. №1. – С. 116–131.
2. Панченко, Т. В. Генетические алгоритмы учебно-методическое пособие [Текст] – Астрахань: Издательский дом «Астраханский университет», 2007. – 87 с. – ISBN 5-88200-913-8.
3. Рейндогльд, Э. Комбинаторные алгоритмы решения задачи коммивояжера. Теория и практика [Текст] / Э. Рейндогльд, Н. Део – М.: Мир, 2000. – 480 с.