


РОССИЙСКАЯ АССОЦИАЦИЯ ИСКУССТВЕННОГО ИНТЕЛЛЕКТА
РОССИЙСКАЯ АССОЦИАЦИЯ НЕЧЕТКИХ СИСТЕМ И МЯГКИХ ВЫЧИСЛЕНИЙ
АДМИНИСТРАЦИЯ ГОРОДА КОЛОМНЫ
РОССИЙСКИЙ ФОНД ФУНДАМЕНТАЛЬНЫХ ИССЛЕДОВАНИЙ
МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ ИМ. Н.Э.БАУМАНА
УЛЬЯНОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

VI-я Международная
научно-практическая конференция

# «ИНТЕГРИРОВАННЫЕ МОДЕЛИ И МЯГКИЕ ВЫЧИСЛЕНИЯ В ИСКУССТВЕННОМ ИНТЕЛЛЕКТЕ»

**(16-19 МАЯ 2011 г.)**

## СБОРНИК НАУЧНЫХ ТРУДОВ

**Том 2**

***К 90-летию Лотфи Заде***

Москва
Физматлит
2011

# ГЕНЕРАЦИЯ КОНЕЧНЫХ АВТОМАТОВ С ПОМОЩЬЮ ГЕНЕТИЧЕСКИХ АЛГОРИТМОВ ДЛЯ РЕШЕНИЯ ЗАДАЧ НАВИГАЦИИ*

*Буздалов М.В.*, *м.н.с.*
*Санкт-Петербургский государственный университет информационных технологий, механики и оптики*
*e-mail: buzdalov@rain.ifmo.ru*

## 1. ВВЕДЕНИЕ

Задачи навигации возникают в различных отраслях современной науки, включая робототехнику. Так, на практике часто возникают задачи о перемещении робота из одной точки в другую в различном пространстве конфигураций – от простейшего случая передвижения робота на плоскости среди препятствий до поиска оптимальной стратегии перемещения в пространстве робота-манипулятора со множеством степеней свободы. При этом различные виды роботов в различных условиях могут хранить подробную карту местности, на которой ему предстоит работать, обладать лишь частичной информацией о местности или не знать о местности ничего, кроме положения конечной точки. Роботы также могут обладать различными средствами получения информации о внешнем мире – от простейших контактных сенсоров до систем видеонаблюдения, в том числе и в инфракрасном диапазоне. Среда работы может быть неизменной или, напротив, иметь изменяющиеся составляющие или параметры.

Среди всего множества задач навигации обычно рассматривают некоторые крайние случаи, такие как, с одной стороны, обладание полной информацией о местности и, с другой стороны, отсутствие возможности запоминать что-либо, кроме нескольких чисел. В реальности эти случаи комбинируются, и системе управления роботом необходимо использовать доступную ему информацию наиболее эффективным способом, одновременно корректно обрабатывая нештатные ситуации. Сложность требований, предъявляемых к системе управления мобильным роботом, очень высока. Использование автоматного программирования [1], по крайней мере в некоторых частях системы управления, позволит уменьшить число ошибок, что приведет к повышению надежности системы.

*Работа выполнена при финансовой поддержке РФФИ (проект №10-01-00654)

Для более эффективного использования автоматного программирования, однако, необходимо исследовать возможность реализации алгоритмов, решающих задачи навигации, автоматными методами. Принимая в расчет сложность этих алгоритмов, автор данного исследования считает целесообразным исследовать автоматическое построение автоматов, решающих некоторые задачи навигации.

## 2. ЗАДАЧА ПОИСКА ЦЕЛИ В ОБЛАСТИ С ПРЕПЯТСТВИЯМИ

Рассмотрим следующую задачу навигации. Имеется некоторая двумерная область с препятствиями. Препятствия могут быть произвольной формы, однако никакие два препятствия не должны иметь общих точек. Кроме того, любой круг конечного радиуса пересекает конечное число препятствий, и любое препятствие покрывается кругом конечного радиуса.

В области, удовлетворяющей описанным требованиям, дана точка – цель. Агент в начальный момент времени находится в произвольной точке области, не принадлежащей ни одному препятствию. Задача агента заключается в том, чтобы добраться до цели или, если она недостижима, сообщить об этом. При этом агент знает свои координаты и координаты цели. Агент не знает, где именно находятся препятствия, но может определить, если ли препятствие в непосредственной с ним близости. Агент обладает O(1) дополнительной памяти, по этой причине он не может запоминать уже посещенную часть области, но может хранить некоторый заранее определенный объем информации (например, координаты некоторого числа точек).

Известны алгоритмы, решающие данную задачу. К ним относятся алгоритмы семейства *Bug*. Общая идея этих алгоритмов такова: пока агент может двигаться по направлению к цели, он это делает. Если агент находит препятствие, которое не дает ему двигаться, он начинает обход этого препятствия. Обход прекращается и продолжается движение к цели, когда выполнено какое-либо условие, специфичное для конкретного алгоритма. В процессе этого обхода алгоритм также может прийти к выводу, что цель недостижима.

Первые алгоритмы семейства *Bug* (*Bug1*, *Bug2*) были впервые описаны в работе [2]. Обобщение алгоритмов семейства *Bug* на случай сложных сенсоров было проведено в работе [3]. Полученный результат был существенно улучшен в работе [4], в которой был описан алгоритм *Tangent Bug*. Этот алгоритм неявно строит на видимых

участках препятствий граф касательных [5] – упрощенный вариант графа видимости – и уже с использованием этой информации прокладывает локально оптимальный путь.

## 3. ПОСТАНОВКА ЗАДАЧИ

Несмотря на то, что описанные алгоритмы решают задачу поиска цели в области с препятствиями, представляет интерес автоматическое построение конечного автомата, который также решает данную задачу. Построенный конечный автомат может быть использован в качестве части системы управления робота, перемещающегося в области с препятствиями, если эта система спроектирована с использованием парадигмы автоматного программирования [1]. Кроме того, данная задача интересна с точки зрения оценки применимости эволюционных методов для построения конечных автоматов для решения задач навигации.

Упростим ранее сформулированную задачу поиска цели в области с препятствиями. Определим область как бесконечное клетчатое поле. Некоторые клетки этой области заняты препятствиями, причем таких клеток конечное число. Цель находится в одной из клеток, не занятых препятствием. Агент также занимает одну клетку. Агент может перемещаться только в клетки, смежные по стороне с текущей и не занятые препятствиями.

Агент знает свои координаты, а также координаты цели. Будем также считать, что у агента есть ориентация в пространстве и, как следствие, выделенное направление «вперед».

Агент управляется конечным автоматом. Конечный автомат принимает ряд входных воздействий и формирует на их основе ряд выходных воздействий. Также у агента есть O(1) дополнительной памяти – помимо собственных координат, координат цели и информации об ориентации в пространстве, он может хранить координаты еще одной точки и число, определяющее ориентацию автомата. Доступ к координатам агента, цели и сохраненной точки осуществляется с помощью входных и выходных воздействий автомата.

### *3.1. Описание воздействий конечного автомата*

Опишем входные воздействия конечного автомата:

- `x1` – верно ли, что в клетке, находящейся впереди агента, нет препятствия.
- `x2` – верно ли, что если агент перейдет на одну клетку вперед, то он станет ближе к цели (без учета препятствий).

- x3 – верно ли, что агент находится у цели.
- x4 – верно ли, что агент находится в сохраненной точке с сохраненной ориентацией.
- x5 – верно ли, что агент находится ближе к цели, чем сохраненная точка.

Опишем выходные воздействия конечного автомата.

- z1 – ничего не делать.
- z2 – передвинуться на одну клетку вперед.
- z3 – повернуться по часовой стрелке.
- z4 – повернуться против часовой стрелки.
- z5 – сохранить текущую точку и ориентацию.
- z6 – остановиться и доложить, что агент достиг цели.
- z7 – остановиться и доложить, что цель недостижима.

Были выбраны именно данные воздействия, так как, с одной стороны, они не оперируют с известными агенту данными на низком уровне (иначе воздействий было бы слишком много, построение автомата заняло бы чрезмерно много времени, а получившийся автомат был бы слишком сложен для восприятия), а с другой стороны, не являются слишком высокоуровневыми (что привело бы к усложнению неавтоматной части алгоритма, а также к уменьшению числа классов алгоритмов, выражаемых с помощью этих воздействий).

### *3.2. Описание генетического алгоритма с применением коэволюции*

В данном исследовании для автоматического построения конечного автомата применяется генетический алгоритм [6]. Функция приспособленности каждого конкретного автомата оценивается исходя из того, как этот автомат проходит некоторый набор полей – тестов. В свою очередь, этот набор полей также эволюционирует параллельно с эволюцией популяции автоматов, таким образом, применяется коэволюция [6]. Цель применения коэволюции – обеспечить отсутствие сходимости генетического алгоритма к неверным решениям вследствие неполноты набора тестов, а также создание благоприятных условий для работы генетического алгоритма.

Общая схема генетического алгоритма такова. На основе имеющегося поколения создается множество потомков. Потомки создаются парами. Выбор родителей для каждой пары потомков осуществляется оператором селекции. Потомки создаются в результате действия оператора скрещивания на родителей, при этом родители остаются неизменными. Далее, для каждой из имеющихся

особей создается ее измененная копия, полученная с помощью оператора мутации. Из всех имеющихся особей в следующее поколение проходят лучшие особи.

После обработки поколения в случае необходимости к набору тестов применятся оператор мутации тестов. Это происходит в двух случаях – когда лучшая особь в поколении прошла все тесты и когда в течение длительного времени значение приспособленности лучшей из особей не изменяется. В первом из этих случаев цель оператора мутации тестов – получить тест, который не проходит ни одна текущая особь, если это невозможно, то считается, что генетический алгоритм достиг своей цели. Во втором из этих случаев оператор мутации тестов предназначен для того, чтобы сдвинуть генетический алгоритм с «мертвой точки».

В тех случаях, когда текущий набор тестов пройден, перед тем, как генерировать новый набор, особи, проходящие все тесты, дополнительно некоторое время оптимизируются по более жестким критериям. Это делается для того, чтобы «обобщить» алгоритмы, инкапсулируемые этими особями, путем уменьшения числа активных состояний, устранения излишних операций или упрощения логики алгоритмов.

### *3.2.1. Особь генетического алгоритма*

В качестве представления состояния автомата используется дерево решений [7]. По сравнению с такими способами представления состояния, как полные таблицы переходов [8], деревья решений требуют меньше памяти для их хранения, а по сравнению с сокращенными таблицами переходов [9] деревья решений при соразмерной потребляемой памяти, проще в реализации. Кроме того, многие из операций над деревьями решения точнее моделируют модификацию способов рассуждения человека, чем операции над таблицами переходов.

В листьях деревьев решений находятся пары чисел – номер выходного воздействия и номер состояния, в которое нужно перейти. В каждом узле находится номер входного воздействия, на основе которого принимается очередное решение.

Автомат полностью описывается массивом деревьев решений, каждое из которых описывает соответствующее состояние. Начальным состоянием автомата является состояние с номером ноль.

### *3.2.2. Операторы генетического алгоритма*

В качестве оператора селекции используется турнирный отбор. Для того, чтобы выбрать особь, из имеющихся особей выбирается

восемь особей. Особи разбиваются на пары, из каждой пары с вероятностью 0,9 победителем выходит более приспособленная, а с вероятностью 0,1 – менее приспособленная особь. Победители вновь разбиваются на пары, между ними вновь происходит отбор и так далее до тех пор, пока не останется ровно одна особь, которая и считается выбранной.

В настоящем исследовании используются два оператора скрещивания. Первый из этих операторов для каждого из номеров состояний с вероятностью 0,2 меняет местами автоматы потомков. Второй оператор для каждого из номеров состояний производит обмен поддеревьями в соответствующих деревьях.

Используемый в данной работе оператор мутации в каждом состоянии с вероятностью 0,1 заменяет случайно выбранное поддерево дерева решения на случайно сгенерированное дерево той же высоты.

### *3.2.3. Функция приспособленности*

Рассмотрим прохождение одной особью (автоматом) одного теста (поля). Поведение особи на тесте характеризуется качественными и количественными характеристиками. Качественные характеристики определяют то, корректно ли пройден тест, а если некорректно, то какого рода ошибку совершил автомат. Количественные характеристики предназначены для сравнения особей, имеющих одинаковые качественные характеристики, и выявления лучшей из них.

Качественной характеристикой поведения особи является *тип результата*. Он может быть одним из следующих:

- Тест пройден корректно.
- Произошло нарушение правил (автомат «врезался» в препятствие).
- Дан неверный ответ (воздействие `z6` в том случае, когда агент не находится у цели, или воздействие `z7` в том случае, когда цель достижима).
- Автомат не останавливается. Данный результат, в свою очередь, может быть классифицирован как:
  - Зацикливание. Граф переходов агента в пространстве состояний вошел в цикл.
  - Убегание. Агент удаляется от цели на все большее расстояние.

Заметим, что на типах результата, вообще говоря, невозможно ввести отношение полного порядка достаточно «разумным» образом. Однозначным образом можно только утверждать, что пройти тест

корректно лучше, чем пройти тест некорректно. Сравнивать же некорректные прохождения теста можно различными способами, но ни один из них не является полностью верным. Так, автомат, который крутится на месте старта, определенно «хуже» автомата, который доходит до цели и «забывает» об этом доложить, хотя оба они имеют одинаковый тип результата – зацикливание. Автомат же, который в целом проходит тесты верно, но, оказавшись в «тупике», выдает сообщение о том, что цель недостижима, значительно лучше первого из «зациклившихся» автоматов, но хуже второго.

Для различения автоматов, обладающих одинаковыми или близкими по смыслу типами результата, предназначены две количественные характеристики: *минимальное достигнутое расстояние до цели* и *суммарное расстояние до цели по пути следования*. Из общих соображений следует, что если из двух автоматов первый смог приблизиться к цели на меньшее расстояние, чем второй, то первый почти наверняка лучше второго. При равных же минимальных расстояниях, чем меньше сумма расстояний до цели по пути следования, тем лучше считается автомат.

Описанная функция приспособленности предназначена для того, чтобы оценивать особей, если тестирование проводится на одном тесте. При использовании многих тестов уже нельзя однозначно сказать, лучше ли один автомат, чем другой, поскольку на разных тестах преимущество может быть у разных автоматов. Данная проблема носит общий характер и, в той или иной степени, относится ко всем задачам многокритериальной оптимизации.

Для дальнейшего описания функции приспособленности целесообразно выделить в описываемом генетическом алгоритме две стадии. Первая из этих стадий имеет место, когда лучшая из особей текущего поколения не проходит все тесты, и цель генетического алгоритма – добиться прохождения всех тестов. Вторая из этих стадий наступает после того, как все тесты пройдены лучшей особью, и продолжается некоторое время. По истечении этого времени к множеству тестов применяется оператор мутации тестов, и процесс снова переходит к первой стадии.

В первой стадии необходимо добиться того, чтобы все тесты были пройдены, поэтому функция приспособленности предназначена для того, чтобы давать преимущество особям, проходящим тесты лучше. Будем считать, что в данный момент времени имеется $N$ тестов, и они упорядочены по времени их появления. Пусть автомат $A_1$ прошел первые $k_1$ тестов без ошибок и допустил ошибку на тесте с номером

$k_1+1$ (если все тесты пройдены без ошибок, то $k_1 = N$), а автомат $A_2$ прошел первые $k_2$ тестов без ошибок и допустил ошибку на тесте с номером $k_2+1$. В этом случае функция приспособленности устроена следующим образом:

- если $k_1 < k_2$, то $A_2$ лучше $A_1$;
- если $k_1 > k_2$, то $A_1$ лучше $A_2$;
- если $k_1 = k_2$, то автоматы сравниваются по функции приспособленности на тесте $k_1+1$ (если $k_1 = N$, то автоматы считаются равными).

Во второй стадии существует, по крайней мере, одна особь, которая проходит все имеющиеся тесты. Однако, почти наверняка способ их прохождения неоптимален, например, особь делает много «лишних» движений. Чтобы устранить этот недостаток, требуется оптимизировать способ прохождения этих тестов. Можно сказать, что при попытке пройти новые тесты в дополнение к предыдущим особь «изобретает» новую стратегию прохождения, а с помощью оптимизации существующих стратегий происходит «обобщение» новых и старых стратегий. Будем считать, что в данный момент времени имеется $N$ тестов, и они упорядочены по времени их появления. Пусть автомат $A_1$ прошел первые $k_1$ тестов без ошибок и допустил ошибку на тесте с номером $k_1+1$ (если все тесты пройдены без ошибок, то $k_1 = N$), а автомат $A_2$ прошел первые $k_2$ тестов без ошибок и допустил ошибку на тесте с номером $k_2+1$. В этом случае функция приспособленности устроена следующим образом:

- если $k_1 < k_2$, то $A_2$ лучше $A_1$;
- если $k_1 > k_2$, то $A_1$ лучше $A_2$;
- если $k_1 = k_2$, то для всех пройденных тестов параметры «минимальное достигнутое расстояние до цели» и «суммарное расстояние до цели по пути следования» складываются для обоих автоматов. Если первые из параметров у автоматов не равны, то лучшим считается тот автомат, у которого первый параметр меньше. Иначе лучше тот автомат, у которого меньше второй параметр.

### *3.2.4. Оператор мутации тестов и коэволюция*

Как было обосновано ранее, оценка особи на фиксированном наборе тестов может привести к тому, что особь, проходящая все тесты, может не являться корректным решением задачи. С другой стороны, если ни одна особь долгое время не проходит все тесты, то

такая система тестов может быть настолько «трудной», что ландшафт функции приспособленности является слишком крутым и изломанным для большинства алгоритмов направленного поиска.

Для решения этой проблемы естественным кажется введение некоторого способа изменения набора тестов. Тесты должны усложняться вместе с улучшением среднего или лучшего решения, и, возможно, упрощаться, если решения ухудшаются или долгое время остаются неизменными по качеству.

В области генетических алгоритмов реализацией этой идеи является коэволюция. Впервые коэволюция была применена в работе [10], где с ее помощью была частично решена проблема преждевременной сходимости генетического алгоритма к локальному, но не глобальному оптимуму. Сущность коэволюции в этой работе заключается в том, что функция приспособленности определяется не только для особи (на основе тестов), но и для теста (на основе особей), и операторы генетических алгоритмов введены не только для особей, но и для тестов (возможно, эти операторы различны или действуют независимо друг от друга). Тем самым, если особи генетического алгоритма стремятся к некоторому локальному оптимуму, не являющемуся глобальным, то тесты, «вытесняющие» алгоритм из этого оптимума, получают большее преимущество, и этот оптимум с течением времени покидается.

В данном исследовании было решено упростить алгоритм, изменяющий набор тестов, с тем, чтобы развитие тестов не доминировало над развитием особей, и дополнить его возможностью упрощения тестов при прекращении развития поколения особей.

В данном исследовании тесты упорядочены в список. Функции приспособленности, описанные ранее, являются «префиксными» – если одна особь проходит корректно больший префикс тестов, чем другая, то она является лучшей. Следовательно, манипуляция сложностью тестов может происходить путем добавления, удаления или изменения последнего теста.

В начале работы генетического алгоритма набор тестов состоит из одного теста, сгенерированного случайным образом. В ситуациях, когда текущий набор тестов проходится хотя бы одной особью, и проведена предварительная оптимизация таких особей, на основе одного из уже существующих тестов случайными мутациями генерируется тест, который не проходится ни одной из имеющихся особей. Этот тест добавляется в конец текущего списка тестов, и работа генетического алгоритма продолжается.

Если за некоторое, заранее фиксированное число поколений ни одна особь не сумела пройти имеющиеся тесты, то делается вывод о том, что последний из этих тестов является «слишком сложным». Из этого следует, что по крайней мере лучшая особь проходит все тесты, за исключением последнего. Следовательно, чтобы продолжить работу генетического алгоритма на более простых тестовых данных, необходимо сгенерировать способом, описанным ранее, новый тест, предназначенный для замены последнего из имеющихся тестов.

Данная стратегия коэволюции позволяет в некотором смысле обеспечить полноту тестов, в то же время избегая долговременных периодов стагнации генетического алгоритма.

## 4. ПОЛУЧЕННЫЕ РЕЗУЛЬТАТЫ

С помощью описанного генетического алгоритма и метода генерации тестов, при котором цель всегда была достижима, был сгенерирован конечный автомат всего из трех состояний, решающий задачу при условии, что цель всегда достижима. Его граф переходов изображен на 0.

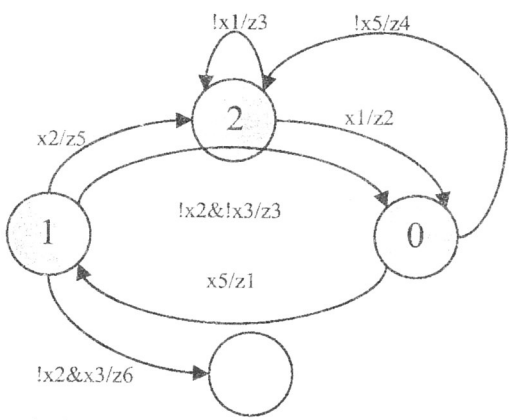


**Рис.1. Автомат для случая достижимой цели**

Данный автомат весьма элегантно производит проверку всех необходимых условий, движение в сторону цели, когда это возможно, и обход препятствия. На 0 продемонстрирована траектория движения агента, управляемого данным автоматом. На основе анализа траекторий агента, а также графа переходов управляющего автомата можно доказать корректность реализованного алгоритма.

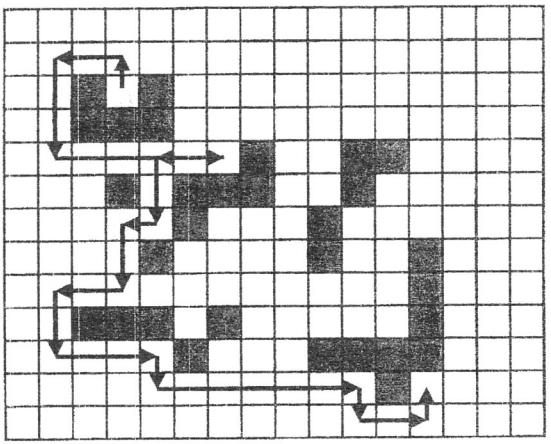

**Рис.2. Траектория движения агента, управляемого автоматом**

Полученный алгоритм напоминает алгоритм *Distant Bug* при том условии, что в сведении задачи на клетчатое поле используется манхэттэнская метрика, в отличие от более традиционной евклидовой.

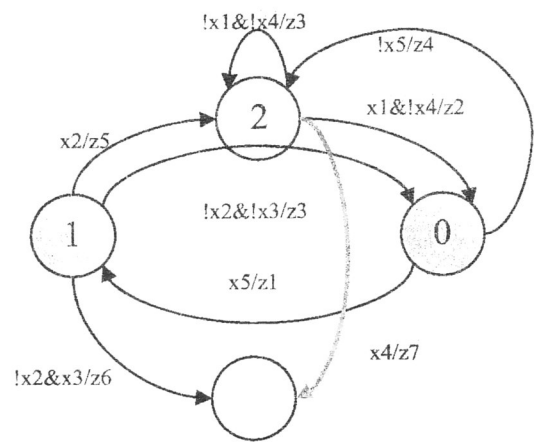


**Рис.3. Автомат для произвольного случая**

Полученный автомат достаточно просто дополнить одним переходом и немного модифицировать условия в состоянии 2, чтобы получить полное решение задачи. Этот автомат изображен на 0.

## 5. ЗАКЛЮЧЕНИЕ

Генетические алгоритмы с использованием коэволюции можно применять для генерации конечных автоматов, используемых при решении тех задач навигации, где применима парадигма автоматного программирования. При этом полученные решения обладают аналитически доказуемой корректностью, а в процессе их поиска компьютер находит новые нетривиальные методы решения встречающихся задач.

## Литература


1. Поликарпова Н.И., Шалыто А.А. Автоматное программирование. – СПб.: Питер, 2010.
2. Lumelsky V.J., Stepanov A.A. Path Planning Strategies for a Point Mobile Automaton Moving amidst Uknown Obstacles of Arbitrary Shape// Algorithmica. – 1987. – Vol.2. – P.403-430.
3. Lumelsky V.J., Skewis T. Incorporating Range Sensing in the Robot Navigation Function// IEEE Transactions on Systems, Man, and Cybernetics. – 1990. – Vol.SMC-20, №5: – P.1058-1068.
4. Kamon I, Rimon E, Rivlin E.A New Range-Sensor Based Globally Convergent Navigation Algorithm for Mobile Robots. CIS – Center of Inteligent Systems 9517, Computer Science Department, Technion, Israel, 1995.
5. Liu Y. H., Arimoto S. Path Planning Using a Tangent Graph for Mobile Robots among Polygonal and Curved Obstacles// International Journal of Robotic Research. – 1992. – Vol. 11, №4. – P.376-382.
6. Holland J.H. Adaptation in Natural and Artificial Systems. – Ann Arbor: University of Michigan Press, 1975
7. Данилов В.Р. Технология генетического программирования для генерации автоматов управления со сложным поведением. – СПб: СПбГУ ИТМО, 2007. Бакалаврская работа. http://is.ifmo.ru/papers/danilov_bachelor
8. Лобанов П.Г., Шалыто А.А. Использование генетических алгоритмов для автоматического построения конечных автоматов в задаче о флибах //Известия РАН. Теория и системы управления. – 2007. – №5. – С.127–136
9. Поликарпова Н.И., Точилин В.Н., Шалыто А.А. Метод сокращенных таблиц для генерации автоматов с большим числом входных переменных на основе генетического программирования //Известия РАН. Теория и системы управления. – 2010. – №2. – С.100 -117
10. Hillis W.D. Co-Evolving Parasites Improve Simulated Evolution as an Optimization Procedure// Phys. D. – 1990. – Vol.42, Issue 1-3. – P.228–234