

Мирча Дмитрий Валерьевич

Тема: программная модель мобильного робота.

Утверждена приказом по университету № 2572/2 от 13 ноября 2000 г.

МАГИСТЕРСКАЯ ДИССЕРТАЦИЯ

по направлению высшего профессионального образования
552800 “Информатика и вычислительная техника”

Факультет Автоматики и Вычислительной Техники

Руководитель:

Губарев В.В.,
д.т.н., проф. каф. ВТ НГТУ

Научный консультант:

Гаврилов А.В.,
к.т.н., доц. каф. ВТ НГТУ

Новосибирск, 2003 г.

Министерство общего и профессионального образования
Российской Федерации

НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

УТВЕРЖДАЮ
Заведующий кафедрой ВТ
д.т.н., профессор
Губарев В.В.

“ ___ ” _____ 2003 г.

ЗАДАНИЕ

на магистерскую диссертацию

студенту *Мирча Д. В.* факультета Автоматики и Вычислительной Техники, обучающемуся по направлению 552800 “Информатика и вычислительная техника”

Магистерская программа (специализация) 552819, “Компьютерный анализ и интерпретация данных”

Тема: Программная модель мобильного робота

Цель работы: Разработать и реализовать программную модель мобильного робота для исследования эффективности его работоспособности в режимах одиночной и совместной деятельности

Руководитель МД:

Губарев В.В.,
д.т.н., проф. каф. ВТ НГТУ

Научный консультант:

Гаврилов А.В.,
к.т.н., доц. каф. ВТ НГТУ

(подпись)

“ ___ ” _____ 2003 г.

Содержание	
Введение	4
Глава 1	6
Роботы с коллективным режимом поведения	7
Игра в футбол	7
Проект Microsoft - “Террариум”	17
Программные модели роботов с индивидуальным поведением	21
Пример Анимата	23
Глава 2	26
Математическое описание для одиночного робота	26
Обзор интересных задач	27
Исследование задачи средней продолжительности жизни («Длительность жизни»)	29
Исследование задачи максимальной продолжительности жизни («Популяция»)	31
Исследование задачи быстрого сбора ресурсов («Все ресурсы»)	34
Выводы	38
Глава 3. Описание программы	39
Схема работы программы Robots:	40
Описание программы Robots	41
Общая структура программы	46
Функционирование программы	47
Выводы	51
Глава 4. Работа с программой	52
Работа с программой. Интерфейс	52
Достоинства и недостатки разработанной системы	57
Выводы	58
Заключение	59
Используемая литература:	60
ПРИЛОЖЕНИЕ 1: НЕКОТОРЫЕ РЕСУРСЫ В ИНТЕРНЕТЕ	62
ПРИЛОЖЕНИЕ 2. Текст программы «Robots»	65
ПРИЛОЖЕНИЕ 3. Полный текст программы «Robots» в MIME-кодировке	65

Закладка не определена.

Введение

В последнее время задача построения эффективно функционирующего искусственного интеллекта стала весьма актуальной, роботы все чаще используются при применении различных задач, до сих пор считавшихся недоступным для них – например эффективный сбор ресурсов. Я исследовал проблему построения эффективного алгоритма поиска ресурсов и проблему выживания популяции в условиях ограниченности ресурсов.

Целью работы является создание программной модели коллективного поведения роботов на примере решения задачи поиска источника питания в случае кооперации с другими роботами и без кооперации, а также исследование эффективности коллективного поведения.

В качестве метода исследования был выбран сравнительный анализ результатов работы программы при задании различных параметров окружающей среды.

Научная новизна данного подхода заключается в необычности постановки задачи, я не нашел постановок или решений данного типа задач, где бы рассматривалось именно коллективное поведение роботов в собирании пищи (в данном случае можно сказать - энергии) для поддержания жизнедеятельности.

Так же новым был сам подход решения задачи с прямым доступом результатов в Internet, поскольку обычно задача решается только на локальных машинах и лишь при наличии программы-симулятора. Исследовалась возможность предоставления максимальной информативности и интерактивности полученных данных с учётом ограничений накладываемых на объём и структуру информации передаваемой по сетям Internet и Intranet.

Научная значимость решаемой задачи заключается в исследовании необходимости и целесообразности режима коллективного поведения в условиях поиска ресурсов и выживания колонии, практическая же ценность в создании механизма решения задачи получения требуемого уровня «общительности» для различных типов начальных условий.

Практическая ценность данной работы заключается также и в том, что результаты исследования коллективного поведения могут быть использованы при построении роботов, предназначенных для поиска различных объектов при чрезвычайных ситуациях, а также «геологических» роботов, созданных для эффективного сбора ресурсов и полезных ископаемых.

В дальнейшем результаты работы планируются использовать для постановки лабораторных работ в рамках постановки специальности «интеллектуальные системы» для магистратуры факультета АВТ НГТУ.

Выкладывание на общий доступ рабочей версии программы произведено на веб-сервере <http://dimmx.olmisoft.com/ii/>, откуда каждый желающий может провести собственные исследования на тему взаимодействия роботов и интерактивном режиме на работающей программе. Планируется разместить и программу и описание также на университетский сервер <http://ermak.cs.nstu.ru> для проведения лабораторных работ в рамках постановки специальности «интеллектуальные системы» для магистратуры.

Диссертация состоит из введения, четырёх разделов, заключения, списка литературы из 23 наименований и трех приложений. Работа содержит 60 страниц основного текста, 14 иллюстраций и 10 диаграмм.

В первой главе приведён обзор существующих систем моделирования коллективного поведения роботов, проведён анализ достоинств и недостатков этих систем, введена классификация существующих систем по наиболее существенным признакам.

Во второй главе приведены методы и инструменты исследования, проводится анализ поставленной проблемы. Строится формальная модель системы и производится выбор инструментария для реализации поставленной задачи. Ставятся эксперименты на различных состояниях системы и делаются выводы.

В третьей главе описываются модули, интерфейсы, приведены системные соглашения по структурам данных. В данной главе производится описание разработанной системы на структурном уровне.

В четвёртой главе приводится описание функционирования системы, описывается взаимодействие с пользователем, рассматриваются достоинства и недостатки разработанной системы.

Глава 1

На различных этапах технологического развития люди стремились создать человекоподобную машину, в двадцатом веке она получила название робот. В последней четверти двадцатого века роботостроение сформировалось как новая отрасль промышленности, были построены миллионы роботов, которые успешно используются на различных производствах прежде всего для автоматизации ручных работ и в экстремальных условиях.

Теперь уже робот совсем не обязательно является человекоподобной машиной, но он всегда остается электромеханическим устройством, выполняющим какие-то действия, определенные задачи. Каждая модель робота предназначена для выполнения определенных типов задач, которые он будет выполнять с наибольшей эффективностью. Под эффективностью я понимаю коэффициент полезного действия (КПД), то есть степень выполнения задачи за определенное время и при определенных затратах. Чем выше эффективность, тем робот полезнее для общества. Поэтому множество исследований посвящено повышению эффективности работы роботов.

Каждый робот может функционировать по определенному, жестко заданному алгоритму, а может иметь гибкую логику и использовать адаптивное поведение к окружающим условиям. Поведение робота всегда зависит его управляющего центра, его «мозга», который может быть реализован в виде конечного автомата – на уровне элементов схем, так делают для простейших роботов, для которых не требуется адаптация к условиям окружающей среды. Для более продвинутых моделей обычно используют электронно-вычислительные машины со сложными алгоритмами работы, либо нейронные сети, реализованные аппаратно или программно.

Для современных нужд робототехники все чаще требуются сложные управляющие устройства последнего типа, поэтому задаче разработки управляющих устройств уделяется столь сильное внимание со стороны исследователей всего мира.

Наиболее экономичным и дешевым методом разработки управляющего механизма является программное моделирование на персональных компьютерах, поскольку можно имитировать любую среду обитания и запускать имитационный процесс сколь угодно раз без реальных затрат на разработку и переконструирования управляющего механизма.

Темой этой диссертации является создание программной модели робота, который может выполнять задачу сбора ресурсов на определенной карте, в простейшем случае генерируемой случайным образом. Роботы имеют определенную длительность жизни, которую они могут увеличить, собирая больше ресурсов (еды), на перемещение и поддержание жизнедеятельности их ресурс тратится.

Основной целью диссертации является исследование поведения модели в различных случаях и установках окружающей среды, длительность жизнедеятельности, среднюю продолжительность жизни в случаях индивидуального режима поведения и коллективного.

Хочу заметить, что существует множество исследований и имитационных моделей мобильных роботов, но, как правило, в этих моделях не учитывается возможность сотрудничества между отдельными роботами в рамках выполнения общей задачи. Лишь малая часть исследований касается кооперативного режима работы роботов.

Роботы с коллективным режимом поведения

Рассмотрим некоторые существующие ныне системы моделирования искусственного интеллекта для коллектива мобильных роботов.

Игра в футбол

Поскольку в настоящее время формируется новый этап развития робототехники на основе микромеханики и нанотехнологий, появилась реальная возможность существенно повысить адаптационные возможности роботов и расширить сферы их применения. В качестве пробной задачи для проверки возможностей систем искусственного интеллекта и робототехники была выбрана игра в футбол. На первом этапе развития в качестве такой пробной задачи выбиралась игра в шахматы, в 1948г Клод Шеннон начал компьютерные эксперименты по игре в шахматы и положил начало периоду внутриутробного развития шахматных программ, в 1974г состоялся первый шахматный чемпионат программ в Стокгольме, на котором победила советская программа Каисса, после этого шахматные программы успешно развивались и программа машины Deep Blue (IBM) победила мирового



Рисунок 1. Футбол роботов. Лига роботов среднего размера.

чемпиона Г.Каспарова, в настоящее время развитие шахматной теории в основном осуществляется на основе компьютеров. Но игра в шахматы включает только двух игроков и очень формализована. Мировому научно-техническому сообществу предлагается в качестве следующего теста игра в футбол, где задействованы сразу две команды по 11 игроков и где уровень формализации значительно ниже.



*Рисунок 2. роботы
«Brainstormers Tribots»,
лига роботов среднего размера*

Нетрудно понять, что для компьютера игра в футбол является значительно более сложной, чем игра в шахматы. Она происходит в реальном времени, в реалистичных физических условиях, при неполной информации. К тому же, имеет (в силу своей непрерывности) значительно большее число комбинаторных вариантов при принятии решения. Построение живого робота-футболиста требует решения многих задач, начиная от компьютерного зрения и распознавания образов до координации и управления группой роботов.

Компьютерное моделирование роботов-футболистов концентрируется на вопросах стратегического и тактического управления группой объектов в реальном времени находящихся во внешнем, изменяющемся окружении (так называемые мультиагентные системы). Эти задачи находят многочисленные применения в военных приложениях, спасательных операциях, мониторинге экологической ситуации микророботами и т.п.

В нашей стране наиболее серьезной командой является лига моделирования СНГ [2].

Лига СНГ

Суть модели, лежащей в основе лиги моделирования СНГ, очень проста. Футболисты и мяч представляют собой массивные, квазиупругие двухмерные цилиндрические объекты с соответствующей физикой столкновений. Футбольное поле ограничено отражающими стенками, и в этом смысле правила скорее хоккейные, чем футбольные. Каждый игрок обладает полной информацией о координатах и скоростях всех объектов на поле. В качестве управляющих используются два параметра - линейное ускорение и угловая скорость футболиста. Начиная с версии 2.0 (2003 г.) добавляется так же вектор удара по мячу. Тем самым моделируются колесные роботы, наиболее популярные на RoboCup.

Организаторы соревнований СНГ-лиги предоставляют серверную программу, выполняющую все необходимые расчеты физики столкновений и выводящую на экран игровую ситуацию на поле. Задача участников соревнований - подготовить dll модуль, состоящий, по крайней мере, из одной функции, которая должна в каждый такт времени возвращать изменение линейной скорости и угла поворота для каждого игрока в команде. Существует подробная документация по подготовке и компиляции этого модуля, а также обо всех функциях SDK [6].

SDK разработано для языков C++ и Pascal и работает на различных операционных платформах. В поставке SDK приведен пример простейшей команды, футболисты которой просто бегут на мяч. С исходниками на C++ и алгоритмом работы реальной команды p-th.com (Днепр) можно ознакомиться в интернете [7].

Рассмотрим основные действия игрока при игре в футбол. Он может бежать с мячом, изменять направление движения, отдавать пас, бить по воротам, нападать и отбирать мяч и в роли вратаря ловить мячи. Игроки, контролируемые компьютером, как правило, придерживаются довольно тупой тактики. Поскольку для удовлетворительной имитации человеческого поведения даже в простых ситуациях требуется слишком большие вычислительные мощности, обычно игроки, контролируемые компьютером, используют упрощенную тактику действий.

Например, в футболе команда компьютера придерживается зонной системы, когда каждый игрок перемещается только в пределах некоего прямоугольного участка поля, нападая на появляющегося в зоне игрока противника и оставляя его, когда он уходит за пределы зоны. В баскетболе иногда используется персональная опека, когда каждому из пяти игроков предоставляется опекун, постоянно мешающий вашему баскетболисту. Пока алгоритмы этой группы спортивных компьютерных игр не оформились окончательно и фирмы предлагают совершенно отличные друг от друга методы управления спортивной борьбой. Тогда общий алгоритм игры в футбол [4] будет таким:

Инициализация параметров игры. Выбрать некоторого спортсмена в качестве текущего.

для текущего (сновного) игрока выбрать действие:

- перемещение: переместить выбранного спортсмена;
- дриблинг: перемещать спортсмена с проверкой сопротивления игроков противника; при необходимости отдать мяч;

- пас: в соответствии с длиной паса и индивидуальными характеристиками спортсмена определить точку приземления мяча;
- оборона: рассчитать вероятность отбора мяча у игрока противника в соответствии с индивидуальными характеристиками;
- удар по воротам: с учетом положения игрока по отношению к воротам и их защищенностью игроками противника рассчитать вероятность попадания в створ ворот; по индивидуальным характеристикам вратаря рассчитать вероятность ловли мяча.

Для i -го игрока, управляемого компьютером, выполнить простейшие действия:

- если мяч или игрок команды соперников с мячом в зоне контроля данного игрока, то перемещаться к нему и попытаться отобрать мяч;
- если мяч у игрока своей команды, то перемещаться вперед до границы своей зоны;
- если мяч у игрока чужой команды, то перемещаться назад до границы своей зоны;
- если мяч у самого игрока, то в зависимости от функций (нападающий, защитник) и распределения вероятностей рассчитать необходимость паса/дриблинга/удара по воротам.

Вернуться к п.2.

Математическая постановка задачи планирования действий коллектива роботов при игре в футбол

Математически задача планирования действий коллектива роботов на примере игры в футбол может быть сформулирована следующим образом. Пусть имеется две одинаковые группы по n игроков в каждой: $P=[P_0, P_1, \dots, P_{i-1}, P_i, P_{i+1}, \dots, P_{n-1}]$ - игроки команды и $T=[T_0, T_1, \dots, T_{i-1}, T_i, T_{i+1}, \dots, T_{n-1}]$ - игроки противника. Каждая команда состоит из одного игрока-вратаря P_0 или T_0 и из $n-1$ игроков, выполняющих роль защитника/нападающего в зависимости от текущей ситуации: $[P_1, \dots, P_{i-1}, P_i, P_{i+1}, \dots, P_{n-1}]$ или $[T_1, \dots, T_{i-1}, T_i, T_{i+1}, \dots, T_{n-1}]$. Рабочая зона представляет собой футбольное поле прямоугольной формы с твердым покрытием и размерами $l_{\text{length}}^{\text{field}} \times l_{\text{width}}^{\text{field}}$. Каждый игрок может перемещаться вперед/назад и поворачивать влево/вправо с равномерной скоростью и без пробуксовок. Поле делится на две

одинаковые основные зоны: зона своей команды и зона противника, которые разделены центральной линией. В каждой такой зоне находятся соответствующие ворота с ограничивающей зоной вратаря. Для осуществления вбрасывания мяча, на центральной линии имеется центр поля или зона вбрасывания мяча. Для определения местоположения игроков вводится декартова система координат. Все футбольное поле разбито на m квадратных участков S_j с размерами соизмеримыми размерам игроков и образующие полное множество участков поля $S = [S_0, S_1, \dots, S_{j-1}, S_j, S_{j+1}, \dots, S_{m-1}]$ (где $m > 2n$). Каждый участок характеризуется координатами по осям OX , OY и принадлежностью к одной из выше перечисленных зон (см. рисунок 2). Во время игры, каждому игроку известно местоположение игроков обеих команд, а также мяча. Действия игроков строго синхронизированы по времени. Через определенные периоды времени, каждый игрок принимает решение об очередном своем действии в составе команды, после чего он выполняет его. Время, в течение которого игрок принимает решение, называется временем принятия решения ($T_{пр}$), а время, затрачиваемое на его выполнение - время выполнения решения ($T_{вр}$). Скорость перемещения мяча больше скорости игрока, поэтому за время $T_{вр}$ мяч "пролетит" расстояние - l_{max}^{ball} , которое больше максимального расстояния l_{max}^{player} , на которое может переместиться игрок за тоже время. Игра каждого члена команды представляет собой как простые действия, такие как перемещение по полю, так и сложные - взаимодействие с мячом или с другими игроками обеих команд. Опирируя с мячом, игрок может "вести" его, то есть перемещаться вместе с ним по полю, либо ударить по нему, для паса игроку своей команды или для атаки ворот противника. При взаимодействии с игроками своей командой, игрок может принять пас от одного из "своих" либо находиться от них на определенном расстоянии, а при взаимодействии с противником - перехватывать мяч либо создавать помехи для перемещения игроков противника. Для организации игры вводятся правила, которые ограничивают перемещения игроков в зависимости от текущего состояния системы, принадлежности к той или иной команде и их функционального назначения.

Планирование действий рассматривается в двух аспектах: применительно ко всей команде - стратегия поведения и непосредственно к каждому игроку - тактика.

Тактика поведения i -го игрока определяется задачей выбора его очередного целевого положения (положение в которое переместиться игрок в следующий период времени), из множества участков поля $S = [S_0, S_1, \dots, S_{j-1}, S_j, S_{j+1}, \dots, S_{m-1}]$ ($m > 2n$), при попадании игрока в который некоторая совокупная величина $\gamma_{i,j}$, являющаяся некоторым критерием

эффективности его действий в текущей ситуации, стремилась бы к максимальному значению.

Выбор целевого положения для каждого робота-игрока в текущей ситуации осуществляется в соответствии с алгоритмом, аналогичным приведенному в докладе Каляева И.А., Капустяна С.Г., Усачева Л.Ж. "Способ динамического распределения целей в задаче группового применения мобильных роботов специального назначения". [14]

Проблема состоит в формировании некоторых комплексных оценок тех или иных целенаправленных действий каждого робота с точки зрения коллективной эффективности.

Значение такой оценки каждого возможного j -го целевого участка для каждого i -го робота - $\gamma_{i,j}$ может быть определено как:

$$\gamma_{i,j} = a_{i,j} (k_1^{\text{стр}} \gamma_1^{i,j} + k_2^{\text{стр}} \gamma_2^{i,j} + k_3^{\text{стр}} \gamma_3^{i,j} + k_4^{\text{стр}} \gamma_4^{i,j} + k_5^{\text{стр}} \sum_{v=0, v \neq i}^n \gamma_5^{i,j,v} + k_6^{\text{стр}} \sum_{w=0, w \neq i}^n \gamma_6^{i,j,w} + k_7^{\text{стр}} \gamma_7^{i,j})$$

где $a_{i,j}$ - коэффициент "игрок-зона"; $\bar{\varphi}_1$ - фактор "игрок-цель¹"; $\gamma_2^{i,j}$ - фактор "цель¹-ворота противника"; $\gamma_3^{i,j}$ - фактор "цель¹-ворота команды"; $\gamma_4^{i,j}$ - фактор "цель¹-мяч"; $\gamma_5^{i,j,v}$ - фактор "цель¹-игрок команды"; $\gamma_6^{i,j,w}$ - фактор "цель¹-игрок противника"; $\gamma_7^{i,j}$ - фактор "цель¹-центральная линия";

$k_1^{\text{стр}} - k_7^{\text{стр}}$ - стратегические коэффициенты соответствующих факторов. [15]

Коэффициент $a_{i,j}$ разрешает или запрещает i -му игроку перемещение в j -й участок поля согласно правилам игры. Коэффициент принимает нулевое значение в случае запрещенных положений игрока, а единичное - в случае разрешенных. Таким образом, этот

коэффициент определяет тактику игры каждого игрока. Факторы - $\bar{\varphi}_1$, $\gamma_2^{i,j}$, $\gamma_3^{i,j}$, $\gamma_4^{i,j}$, $\gamma_5^{i,j,v}$, $\gamma_6^{i,j,w}$, $\gamma_7^{i,j}$ позволяют производить оценку возможного j -го целевого положения i -го игрока относительно расстояний до объектов (игрок, ворота противника, ворота команды, мяч, игрок команды, игрок противника и центральнона линия соответственно) и принимают значения в диапазоне от 0 до 1. Коэффициенты $k_1^{\text{стр}} - k_7^{\text{стр}}$ устанавливают приоритеты и

влияния соответствующих факторов в зависимости от стратегии, определяемой состоянием мяча. Своим значением коэффициент определяет приоритет фактора, а знаком - его положительное/отрицательное влияние. Варьируя ими можно задавать стратегию игры в зависимости от текущей ситуации.

Стратегия поведения команды определяется состоянием мяча, которое наиболее полно отражает состояние всей системы.

Существует 6 состояний мяча: вбрасывание мяча (начало игры), мяч свободен, мяч в воротах команды (гол команде), мяч в воротах противника (гол противнику), мяч у игроков команды, мяч у игроков противника. Система последовательно переходит из одного состояния в другое под действиями игроков обеих сторон (см. рисунок 3).

В случае когда команда обладает мячом (состояние “мяч у игроков команды”), игроку владеющим им необходимо оценить возможное следующее целевое положение мяча и определить действия над ним (удар по воротам, проход с мячом или пас).

Для определения возможности удара по воротам строится одномерный массив весов гола $U = |u_0, u_1, \dots, u_{z-1}, u_z, u_{z+1}, \dots, u_{r-1}|$. Элементами массива являются значения весовых коэффициентов показывающие возможность попадания мяча на z-й участок поля из множества $S = [C_0, C_1, \dots, C_{z-1}, C_z, C_{z+1}, \dots, C_{r-1}]$ принадлежавшего воротам (где $C \in S$) в случае нанесения по ним удара. Чем выше значение весового коэффициента, тем больше вероятности забить гол в данном направлении, поэтому задача оптимизации удара по воротам и определения возможности его нанесения сводится к отысканию наибольшего значения коэффициента, который определяется как:

$$u_z = u_z^{\text{цель}^2} \times u_z^{\text{вп}}$$

где $u_z^{\text{цель}^2}$ - фактор “мяч-цель²”, $u_z^{\text{вп}}$ - фактор “цель²-вратарь противника”.

Факторы - $u_z^{\text{цель}^2}$, $u_z^{\text{вп}}$ позволяют производить оценку z-го участка ворот противника относительно расстояний до объектов (текущее положения мяча и текущее положение вратаря соответственно) и принимают значения в диапазоне от 0 до 1.

Для определения возможности и организации паса, строится одномерный массив весов паса:

$$D = |d_0, d_1, \dots, d_{j-1}, d_j, d_{j+1}, \dots, d_{m-1}|$$

Элементами массива являются значения весовых коэффициентов, показывающие возможность передачи паса на j -й участок поля из множества $S=[S_0, S_1, \dots, S_{j-1}, S_j, S_{j+1}, \dots, S_{m-1}]$. Чем больше значение коэффициента, тем более благоприятная ситуация для передачи паса. Значение коэффициента определяется как:

$$d_j = d_j^{\text{цель}^3} \times d_{v,j}^{\text{ик}} \times d_j^{\text{напр}} \times q_j^{\text{прот}}$$

где $d_j^{\text{цель}^3}$ - фактор "мяч-цель³"; $d_{v,j}^{\text{ик}}$ - фактор "цель³ - игрок команды"; $d_j^{\text{напр}}$ - фактор "цель³ - направление"; $q_j^{\text{прот}}$ - признак присутствия противника.

Факторы - $d_j^{\text{цель}^3}$, $d_{v,j}^{\text{ик}}$ и $d_j^{\text{напр}}$ позволяют производить оценку возможного j -го целевого положения мяча относительно расстояний до объектов (текущее положение мяча, текущее положение игрока своей команды и центр ворот противника соответственно) и принимают значения в диапазоне от 0 до 1. Признак $q_j^{\text{прот}}$ показывает на наличие или отсутствие игроков противника около возможного j -го целевого положения мяча. Признак принимает нулевое значение в случае нахождения игрока противника в непосредственной близости от целевого положения мяча и единичное - в противном случае.

Матрицы гола - U и паса - D строятся только для случая "мяч у игроков команды", в то время как матрица коэффициентов эффективности Γ - для всех случаев.

Таким образом, алгоритм планирования действий коллектива роботов, играющих в футбол, сводится к следующему:

1⁰ определяется состояние системы;

2⁰ каждому игроку задаются соответствующие стратегические коэффициенты $K_i^{\text{стр}}$ - $K_j^{\text{стр}}$, определяющие стратегию игры команды для текущего состояния;

3⁰ для каждого игрока команды определяется тактика игры, которая заключается в определении всех допустимых целевых положений и их приоритетов;

4⁰ в случае, если система находится в состоянии "мяч у игроков команды", для игрока, владеющего мячом, строится матрица гола U для определения возможности нанесения удара по воротам. В случае невыполнения удара по воротам, для игрока с мячом строится матрица паса D для определения возможности передачи паса. Если пас возможен, игрок с мячом передает номер участка поля, куда будет послан мяч, игроку который должен его принять;

5⁰ независимо от состояния системы для каждого игрока команды строится матрица коэффициентов эффективности Γ для определения его очередного действия. В случае если осуществляется пас, игрок, принимающий его, имеет в качестве очередного действия должен выбрать переход в положение, наиболее выгодное для приема мяча;

6⁰ после отработки запланированных действий, алгоритм повторяется.

Игра роботов в футбол является наиболее иллюстративным примером использования принципов коллективных действий в условиях динамически изменяющейся ситуации, поэтому алгоритм планирования действий разрабатывался применительно к коллективу роботов, играющих в футбол. Суть алгоритма заключается в том, что в каждой текущей ситуации роботы выбирают очередную промежуточную цель и, исходя из этой цели, определяют действия из набора возможных. Алгоритм является быстродействующим, так как не содержит сложных вычислительных процедур. При изменении ситуации в результате действий самих роботов и роботов противника процедура повторяется. Выбор целей роботами, осуществляется таким образом, чтобы был достигнут максимальный эффект в текущей ситуации. В этом алгоритме рассмотрены также способы задания стратегии поведения коллектива и тактики отдельных игроков коллектива.

Robocup – мировой чемпионат по футболу среди роботов

В 1999 году прошел первый чемпионат мира по футболу роботов в Стокгольме в двух лигах - в лиге роботов (robot league) и в лиге моделирования (simulation league) и выявил большой интерес к этому начинанию [1, 16]. Конечной целью ставится создание команды человекообразных роботов, которые в 2050 году смогут сразиться в футбол с чемпионской командой людей (рис 3).



Рисунок 3. Соревнования по футболу между роботами ASIMO

Сейчас чемпионаты мира среди роботов-футболистов проводятся в нескольких лигах, одна из которых – компьютерное моделирование игры (соревнование алгоритмов). В начале декабря 2001 г. в рамках Всероссийского научно-технического [фестиваля](#) молодежи Мобильных Роботов в МГУ прошел первый чемпионат СНГ по компьютерному моделированию игры в футбол [5].

В 2002 году с 19 по 25 июня прошел международный чемпионат по футболу между роботами. Было представлено более 120 команд из 25 стран. Соревнования проводились в пяти следующих лигах [9]:

Лига роботов небольшого размера. Команды состоят из пяти колесных роботов, диаметр которых не превышает 18 см в диаметре. Роботы полностью автономны и не имеют связи в реальном времени со своими создателями. По словам Осады, игры будут проводиться в очень быстром темпе. В 2001 году в этой лиге победила команда из Ngee Ann Polytechnic (Сингапур).

Лига роботов среднего размера. Команда состоит из четырех колесных роботов диаметром менее 50 см. Эти роботы также автономны, поэтому модели этой лиги и лиги роботов небольшого размера проходят аппаратные и программные тесты. На чемпионате 2001 года победителем стала команда из CS Freiburg (Германия).

Лига четырехногих роботов Sony. Эти роботы напоминают созданного корпорацией Sony робота-собаку Aibo, поэтому, как считает Осада, данное соревнование — это в первую очередь соревнование по программированию. Чемпионом в 2001 году была названа команда Университета штата Новый Южный Уэльс (Австралия).

Лига моделей. В турнире этой лиги не принимают участие физические роботы, а вся борьба происходит на экране компьютера, поэтому, по существу, это битва двух

конкурирующих стратегий. Это самая старая лига RoboCup; в 2001 году победителем в ней стала команда из Университета Тсингхуа (Китай).



Рисунок 4. Лига гуманоидных роботов

Лига человекообразных роботов. В играх этой лиги могут принимать участие любые роботы, перемещающиеся на двух ногах. Для этой категории пока не установлены детальные правила, однако известно, что человекообразные роботы движутся все еще очень медленно. Организаторы предполагают, что игры будут проходить между командами, состоящими из одного или двух роботов, и все сведется к сериям

пенальти. На участие в этих соревнованиях подали заявку восемь команд.

На этих соревнованиях 2002 года Япония стала победителем в наиболее реалистичной и приближенной к условиям настоящего футбола лиге «роботов среднего размера». В лиге роботов малого размера победила команда США, Россия же выступала только в лиге моделей.

Следующий фестиваль "Мобильные роботы-2003" [5] пройдет уже в ноябре 2003 года.

Проект Microsoft - "Террариум"

"Террариум" (Terrarium) - это многопользовательская игра, представляющая собой симулятор природной экосистемы, разработанный средствами .NET Framework. Разработчики могут создавать собственных существ и добавлять их в игру на своих клиентских компьютерах, а специальные телепортеры перемещают их создания между всеми клиентами одноранговой сети "Террариума".

Создавая новых существ, игрок может выбирать между травоядными («коровами»), которые питаются растениями, и плотоядными («хищниками»), которые могут питаться как травоядными, так и другими плотоядными (см. рисунок 5). Как только создание игрока попадает в экосистему "Террариума" и вступает в соперничество за ее ресурсы, он может использовать данный Web-сайт для сравнения статистики его естественного движения с показателями других обитателей экосистемы. Главная цель игры - создание существа, способного пережить всех остальных.



Рисунок 5. Рабочий экран программы Terrarium

Рассмотрим основные принципы функционирования игры. Игра может работать в двух режимах [11]:

- Локальный режим Terrarium.** В этом режиме пользователю доступны две опции. Пользователь может играть один, без взаимодействия с другими клиентскими приложениями Terrarium. В этом случае экосистема, отображаемая на экране, представляет собой всю экосистему. В этом режиме удобно отлаживать своих созданий.. Пользователь может также присоединиться к группе компьютеров, на которых запущены приложения Terrarium, образующие экосистему. Это осуществляется через подключение к общему каналу следующим образом: каждый участвующий пользователь выбирает специальную, частную сеть, вводя согласованное название сети в поле channel на консоли Terrarium. После ввода в это поле название сети/канала, приложение Terrarium на данном компьютере взаимодействует только с теми компьютерами, в приложениях на которых пользователи ввели то же самое название.
- Режим Экосистемы.** Это - стандартный режим, в котором приложение Terrarium, запущенное на данном компьютере представляет собой маленькую

часть общей экосистемы, которая охватывает все участвующие компьютеры, работающие через общий Terrarium сервер. Т.е. общая экосистема состоит из всех запущенных в данный момент времени приложений Terrarium.

При программировании своих созданий разработчики имеют полный контроль над всем набором генетических черт (зрение, скорость, защитная мощь, мощь нападения, и т.д.); над поведением (алгоритмы для обнаружения добычи, перемещения, нападения, и т.д.); над воспроизводством (как часто создание воспроизводит себе подобных и какая генетическая информация, если таковая вообще имеется, будет передана потомству). После завершения процесса разработки код компилируется в сборку (assembly - динамически связанная библиотека, или DLL) которая может быть загружена в локальную часть экосистемы, просматриваемую через Terrarium приложение. Когда создание первоначально представлено в Режиме Экосистемы, десять экземпляров этого создания рассеиваются по локальной части экосистемы. Ни одного экземпляра данного создания больше не может быть загружено в экосистему ни пользователем его создавшим, ни любым другим пользователем в сети, пока все экземпляры создания не умерли в процессе эволюции. Напротив, в локальном режиме Terrarium, неограниченное число экземпляров данного создания может быть введено в среду

Как только создание загружено в Terrarium, его поведение обуславливается тем как оно запрограммировано. Каждому созданию предоставляется от 2 до 5 миллисекунд (в зависимости от производительности компьютера) для выполнения действия, прежде чем оно будет разрушено. Это предотвращает от захвата процессора одним созданием и приостановки/блокировки игры (например, как в случае с бесконечным циклом).

Синий шар – телепортер - перекатывается случайным образом в пределах каждого приложения Terrarium. Если Terrarium взаимодействует с другими компьютерами (или в Режиме Экосистемы или при использовании частного разделяемого канала в локальном Режиме Terrarium), всякий раз, когда этот синий шар пересекает создание, оно транспортируется в отобранный случайным образом компьютер, относящийся к той же экосистеме.

Центральный (master) сервер Terrarium обеспечивает возможностями обнаружения компьютеров в рамках глобальной экосистемы или рамках экосистемы на одном общем канале. Так же этот сервер обеспечивает возможностями сбора статистики [12].

С этой игрой связан один курьезный факт, давно вошедший в анекдоты. В чемпионате 2002 года по игре Terranium российские программисты заняли первое и второе места в общем конкурсе, написав очень оригинальный алгоритм существования.

Отечественным разработчикам пришли в голову две уникальные идеи, на которые представители других стран оказались просто неспособны. Первая идея заключалась в следующем. Когда в пределах видимости российской коровы оказывались коровы исключительно чужой породы, наша соотечественница быстро съедала всю доступную траву - и погибала героической смертью, обрекая на смерть и "чужих". Эдакий Иван Сусанин с наклонностями к суициду и ксенофобии. Заметьте, вред делали не хищникам, а просто чужим коровам - пусть лучше погибнут, чем размножатся и выиграют.

Вторая идея, которая, по словам организаторов игры, и определила успех российских коров, была вообще предельно проста. Многие разработчики задумывались - как обучить своих коров обмениваться информацией. Формальные правила запрещали непосредственный обмен данными между животными. Но научить одну корову показывать другим, где есть много травы или где находится хищник (видимость в террариуме была ограничена), было бы очень полезно для всего подвида.

Один из российских программистов сумел решить эту задачу. Решение было очень простым: если одна корова видела другую, быстро бегущую в каком-то направлении, она сама начинала бежать в том же направлении.

Поскольку у компьютерных травоядных было всего два интереса в жизни - есть траву и не быть съеденной, то наличие в окрестностях бегущей коровы, как правило, означало одно из двух. Либо там, куда бежит корова, есть трава, либо там, откуда она бежит, есть хищники. В обоих случаях для другой коровы было очень логично последовать поданному примеру. И стадо коров-чемпионок бойко срывалось с места при возникновении опасности или исчерпании корма.

Благодаря этим двум идеям наша команда выиграла [12].

Таким образом мы видим, что данная система хоть и не позволяет напрямую общение между особями, некоторое подобие сотрудничества реализовать можно, что и было успешно проделано русскими программистами.

Программные модели роботов с индивидуальным поведением

Рассмотрим, в рамках темы диссертации, следующие модели мобильных роботов, известных как «Аниматы».

Аниматами называются автономные агенты, поведение которых должно следовать принципам поведения животных. Они представляют собой полезный инструмент для изучения поведения животных. Эксперименты с такими агентами дают возможность подвергать критической проверке существующие представления о этих принципах, а также формулировать новые гипотезы, которые можно проверить на живых организмах. Результаты, полученные в этих экспериментах, уже позволили сформулировать некоторые принципы адаптивного поведения. В то же время, пока нет общей теоретической основы для создания аниматов. [17]

Рассматриваемый подход сформировался во второй половине 80-х – начале 90-х гг. Его основные положения изложены в работах [19,20], а некоторые его достижения в применении к изучению поведения животных можно найти в обзоре [21]. Как и следовало ожидать, к настоящему времени в рамках общего подхода сформировались различные направления, перекрывающиеся с бионикой, математическим моделированием поведения животных и искусственным интеллектом. Чтобы составить представление о разнообразии этих направлений, полезно начать с ресурсов Интернета, указанных в Приложении 1. Однако, несмотря на это разнообразие, можно выделить то главное, что делает создание аниматов самостоятельной дисциплиной:

- От бионических разработок аниматы отличаются тем, что в них должны быть воплощены не отдельные «патенты природы», а, как уже отмечено выше, фундаментальные принципы, определяющие поведение живых организмов, - те принципы, в силу которых это поведение отличается от поведения машин. Например, можно создать аппарат с шестью ногами и сенсорами, имитирующими глаза муравья, но, если управление походкой и навигацией аппарата запрограммированы, исходя из некоторых «общих соображений», то это – не анимат. И наоборот, робот может передвигаться на колесах и воспринимать внешнюю среду с помощью радара, но если его действия подчиняются тем же закономерностям, что и у животных, - это анимат.
- Даже виртуальный анимат отличается от «обычных» математических моделей поведения животных. Отличие состоит в том, что для модели допустимо воспроизводить только одну сторону поведения без всякой связи с целями

животного и с той средой, в которой осуществляется поведение. Например, ритм суточной активности может моделироваться осциллятором, переключающимся между активной и неактивной фазами, без всяких указаний на то, что именно делает организм во время активной фазы и как это способствует адаптации организма к среде. Напротив, анимат должен быть целостным агентом, т.е. он должен выполнять все действия, необходимые для достижения определенной цели в конкретной, пусть и виртуальной среде. В применении к суточному ритму это означает, что анимат должен решать определенные задачи в течение активной фазы, например, успевать восстанавливать свои энергетические ресурсы в среде с определенным распределением пищи.

- От традиционных разработок в области искусственного интеллекта аниматы отличаются не только ориентацией на принципы поведения животных. Не менее важно то, что создание аниматов ориентировано в конечном итоге на решение ими нечетко сформулированных задач в плохо предсказуемой среде – т.е. таких, с которыми приходится иметь дело живым организмам. Аниматы создаются не для игры в шахматы, а для таких задач, как исследование поверхности планет в условиях, когда время и ресурсы робота ограничены, а местность не изучена.

Эти особенности аниматов открывают новые возможности для биологии. В самом деле, лучшая проверка наших знаний о механизмах адаптивного поведения животного в его естественной среде – это воссоздать по возможности полный аналог животного и посмотреть, насколько поведение такого аналога и в самом деле адаптивно в той же самой среде. И, хотя выполнение этой исследовательской программы в полном объеме нереально, даже её частичное осуществление (например, если виртуальный агент, только частично воспроизводящий поведение живых прототипов, изучается в виртуальной среде, частично имитирующей реальную), обещает быть перспективным. Можно надеяться, что эксперименты с аниматами позволят:

- подвергнуть проверке существующие представления о фундаментальных механизмах поведения животных.
- сформулировать новые представления, которые в дальнейшем могут быть проверены в экспериментах с реальными организмами.

Пример Анимата

Приведем пример использования анимата для изучения механизма ориентации самцов тутового шелкопряда в струе феромона самки (Kuwana et al., 1996a,b). Самец воспринимает запах с помощью рецепторов, расположенных на двух симметричных антеннах. Классическая модель ориентации предполагает, что самец всегда поворачивает в сторону той антенны, рецепторы которой возбуждены сильнее. Эта модель была проверена следующим образом.

На первом этапе авторы создали нейронную сеть, соответствующую классической модели. В этой сети сигналы от левой антенны возбуждали эффекторы на правой стороне тела самца, и наоборот. Однако компьютерные эксперименты с ориентацией в струе запаха показали неэффективность этого механизма по сравнению с поведением реальных бабочек: виртуальные самцы часто выходили за пределы струи и не могли найти её снова (рис. 1).

На втором этапе с помощью генетического алгоритма была создана более сложная нейронная сеть, позволившая в компьютерном эксперименте воспроизвести наблюдаемую у реальных самцов успешную ориентацию.

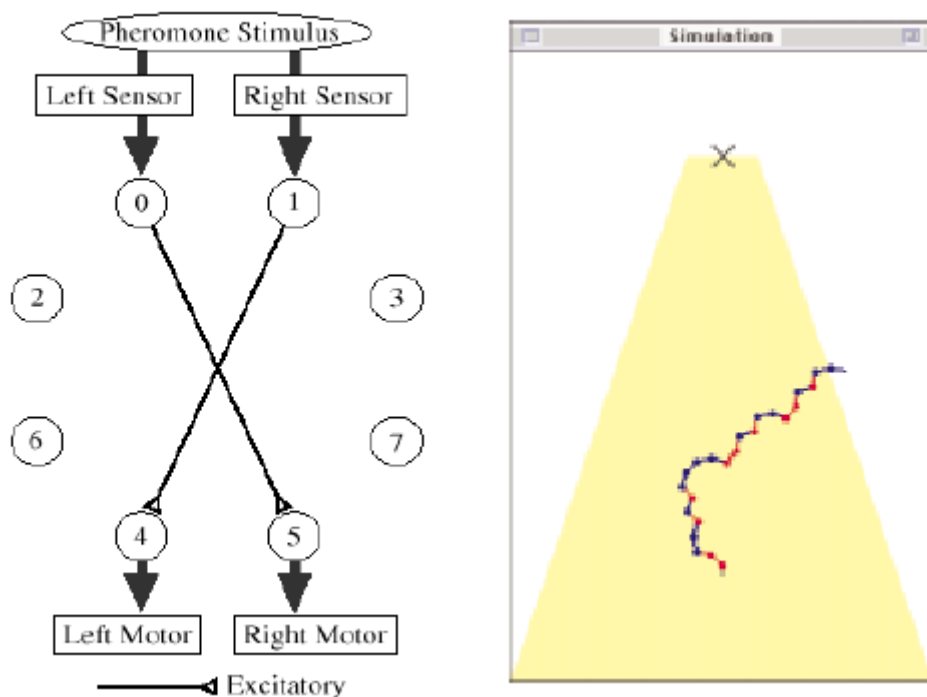


Рис. 6. Слева: сеть, соответствующая классическому представлению об ориентации

бабочек. Справа: траектория в струе запаха (желтый цвет), типичная для этой сети.

«Бабочка» выходит за пределы струи и не находит источник запаха X.

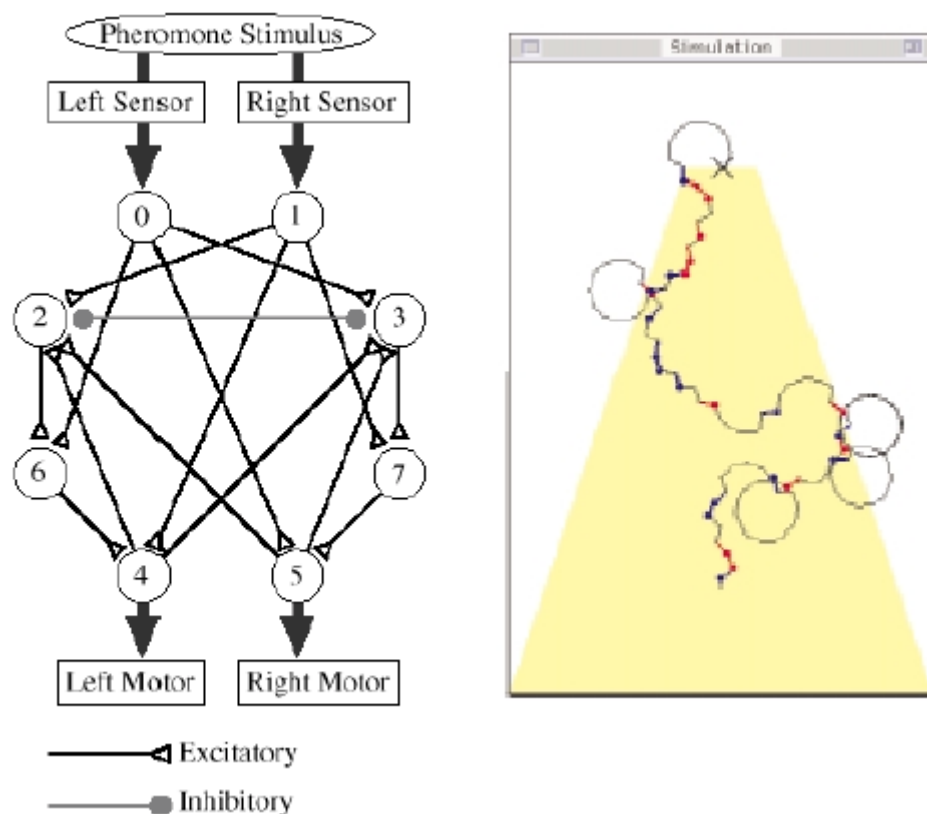


Рис. 7. Новая сеть и соответствующая ей траектория «бабочки» в струе запаха (там же).

На третьем этапе авторы проверили, насколько эффективно полученная нейронная сеть обеспечивает ориентацию физического робота, получавшего сигналы от установленных на нем настоящих антенн шелкопряда. Нейронная сеть передавала преобразованный сигнал колесам робота. Проверка показала, что, в отличие от компьютерного эксперимента, робот ориентировался гораздо хуже, прежде всего потому, что в потоке, несущем феромон, всегда имеет место турбулентность, которая не может не влиять на ориентацию, но не воспроизводится реалистично в компьютерных экспериментах.

Наконец, сеть была модифицирована так, что, в отличие от классической модели, ориентация включала не только повороты в сторону наибольшего раздражения, но и спонтанные, независимые от раздражения зигзаги при движении в струе запаха, а также петли, описываемые самцами, вышедшими за пределы струи (Рис. 2). Сравнение с экспериментальными данными показало, что такое поведение характерно и для реальных бабочек. С помощью модифицированной сети робот успешно находил источник запаха, переносимого струей воздуха. Таким образом, использование робота и реального потока воздуха в данном исследовании имело принципиальное значение для понимания ориентации бабочек.

Полученные результаты могут стать основой для дальнейшего изучения механизма ориентации шелкопрядов уже на нейрофизиологическом уровне. Например, можно попытаться обнаружить в нервной системе бабочек механизм, соответствующий нейронной сети анимата.

Это исследование показывает, как могут дополнять друг друга эксперименты с аниматами и реальными организмами. В результате появляется возможность создать робота, пригодного, в принципе, для практических целей, а также сформулировать гипотезы для лучшего понимания механизмов поведения живых организмов. Следует отметить, что аналогичные исследования механизмов ориентации и поискового поведения, последовательно сочетающие эксперименты с живыми организмами, виртуальными моделями и физическими роботами, стали распространенными в последнее время [21]. Одна из причин быстрого развития таких исследований – практические задачи, например, поиск мин на минных полях, а также обнаружение источников химического загрязнения в водоемах [18].

Глава 2

Поскольку задача трудно формализуема и решение содержит огромное количество параметров, влияющих на ход решения, то в качестве основного метода исследования я использовал имитационное моделирование. Для этого была написана программа на компьютере, которая пошагово, с помощью итераций, проводит моделирование ситуации в заданных изначально условиях.

Математическое описание для одиночного робота

Математически же упрощенную задачу выживания отдельной особи популяции в случае, когда она одна на карте, можно представить так. Изначально особь имеет n -ное количество еды. Каждый ход он тратит на поддержку своего существования 1 единицу еды. Робот движется хаотически и каждый шаг может переместиться по одной из осей x или y на одну клетку, в которой может оказаться запас еды, а может и не оказаться. Общий запас еды распределяется по клеткам поля по закону нормального распределения. Таким образом, робот каждый ход имеет некоторую вероятность найти еду, математическое ожидание вероятности нахождения еды на текущем шаге можно описать как:

$$M\{F_i\} = Q \cdot p_i + 0 \cdot (1 - p_i) = Q \cdot p_i, \text{ где}$$

Q – значение количества еды в данной клетке,

p_i – плотность распределения еды по карте.

Тогда текущий запас еды (энергии) на любой момент времени k можно определить по формуле:

$$Stock = S_0 + \sum_{i=1}^k M\{F_i\} - D \cdot k, \text{ где}$$

- S_0 - начальный уровень (запас) еды
- k - текущий номер шага
- F_i - найденное количество еды
- $M\{F_i\}$ - мат. ожидание найденного количества еды.
- D - съедаемое за день количество еды

Решением задачи является максимизация числа k , при котором $Stock > 0$

Написание математической модели для сообщества роботов, имеющих возможность общаться между собой и обмениваться информацией, представляется задачей реалистичной, но очень сложной, поскольку нужно будет учитывать координаты роботов, отслеживать состояние клеток матрицы, по которой они ходят, чтобы определить время завершения итерационного процесса.

Обзор интересующих задач

Теперь мы можем поставить задачи, которые хотелось бы исследовать с помощью программы в рамках исследования эффективности режимов сотрудничества между роботами и одиночного существования.

Напомню, что по условиям задачи, роботы собирают еду на карте, это дает им возможность дальнейшего существования, стало быть сбор как можно большего количества еды есть стимул к выживанию. В случае одиночного существования каждый робот бродит по карте и ищет источники пищи, находя – собирая все сам. В случае же коллективного режима существования в случае нахождения пищи робот может также позвать остальных роботов, поделившись, таким образом, пищей с остальными. Роботы же могут, в свою очередь, отказаться от подарка, если по их подсчетам они не успеют подойти в заданную точку вовремя – учитывая скорость поедания пищи (сбора ресурсов) и расстояние до точки.

- Во-первых, сам собой напрашивается вопрос – увеличит ли продолжительность жизни вариант с коллективным существованием? Конечно ответ на этот вопрос зависит и от параметров карты, ее насыщенности едой и от вероятности вызова

других роботов этим. Задачей исследования является проверка, до каких значений вышеупомянутых параметров коллективное существование является выгодным, а при каких – вредным, приводящем к снижению средней продолжительности жизни робота? Назовем эту задачу «длительность жизни». Решение этой задачи реализовано в программе.

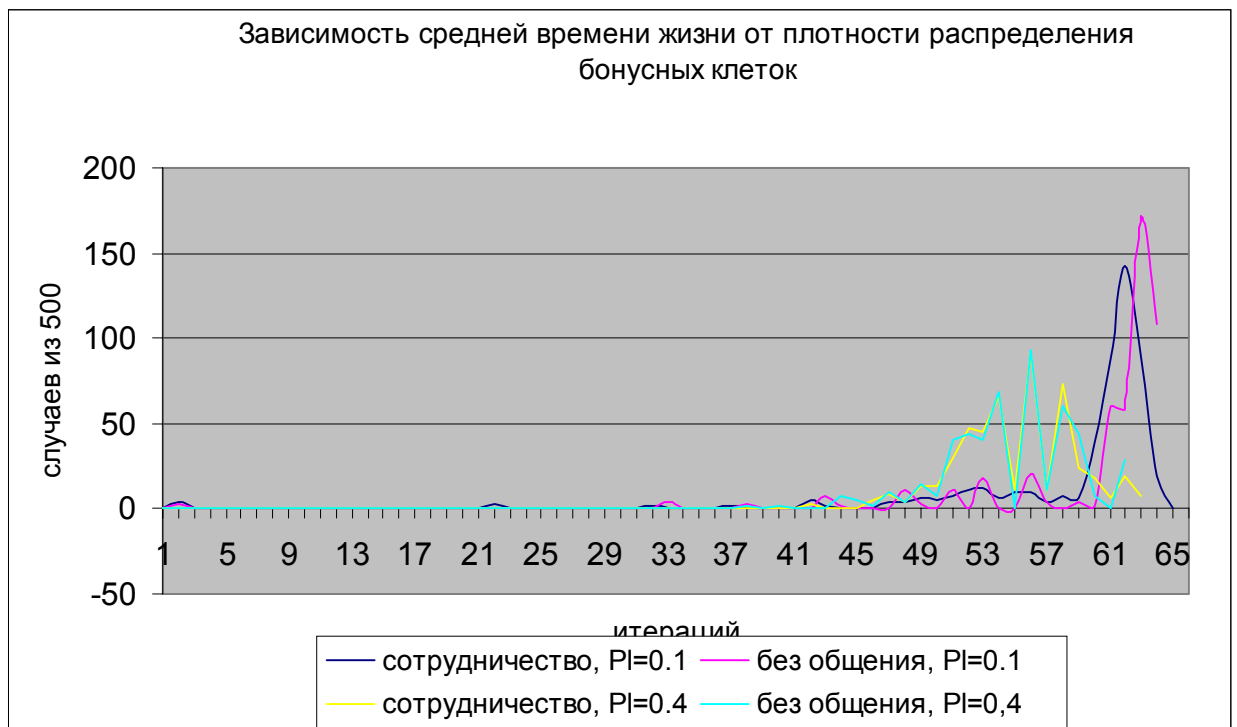
- Во-вторых, интересует максимальная продолжительность жизни робота – при каких параметрах все еще эффективно увеличить вероятность долгожительства путем коллективного существования роботов? Какова должна быть вероятность вызова роботов этим роботом при заданной карте, чтобы увеличить продолжительность жизни? Назовем эту задачу «популяция». Решение задачи реализовано в программе.
- В-третьих, существует очень важная задача сбора ресурсов, когда главной целью является сбор всех ресурсов за минимальное время. При этом количество энергии робота считается несущественным и неограниченным, важно лишь как можно быстрее собрать ресурсы. Задача, думаю, будет интересна для исследователей-геологов и робототехников, проектирующих геологических роботов. Задача актуальна еще и потому, что коллективная работа позволяет минимизировать время поиска и максимизировать время сбора ресурсов, повышая таким образом КПД работы. Но до какой степени богатости ресурсов коллективное поведение все еще является выгодным? Поскольку наступает момент, когда вместо поиска они будут друг друга гонять по карте, зовя, а ведь пока робот подходит, ресурс может и закончиться. Назовем эту задачу «все ресурсы». Решение этой задачи реализовано в программе.
- Интересно исследовать также
 - зависимость эффективности общения от скорости сбора ресурсов, при прочих одинаковых параметров,
 - зависимость продолжительности жизни от размеров карты при равной плотности распределения пищи и равном количестве еды всего на карте при коллективном поведении и при одиночном.
 - или времени сбора ресурсов от скорости сбора ресурса каждым роботом при коллективном поведении. Начиная с какой скорости коллективное поведение будет препятствовать эффективности сбора?

- Размеры карты, при котором роботам будет эффективнее заниматься самостоятельным поиском, чем постоянно передвигаться друг к другу при вызове, учитывая что он может «опоздать».

Подобных задач можно придумать множество, некоторые из них я исследую сам и попытаюсь сделать выводы.

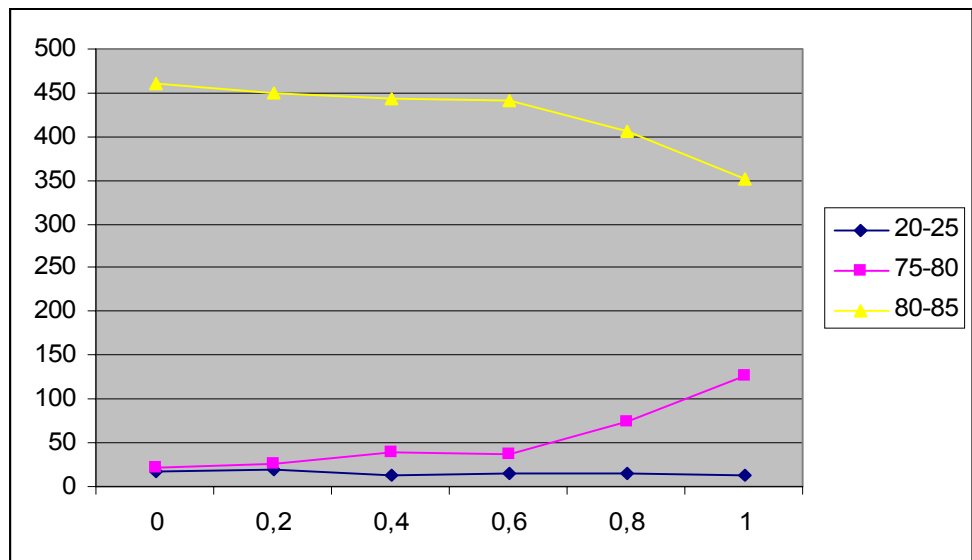
Исследование задачи средней продолжительности жизни («Длительность жизни»)

Параметры запуска: Рабочее поле [8x8], плотность распределения пищи $Pf=0.4$, еды на карте: 200, скорость поедания $s=4$, роботов на поле =4, начальная энергия роботов - 20, режим коммуникаций включен, вероятность вызова других роботов $Pt=1$



В результате не заметно увеличения вредного времени жизни роботов.

Тогда рассмотрим следующие условия : рабочее поле [6x6], плотность распределения пищи $Pf=0.1$, Еды на карте: 200, скорость поедания $s=4$, Роботов на поле =4, начальная энергия роботов - 20, режим коммуникаций меняется от 0 до 1 с шагом 0.2

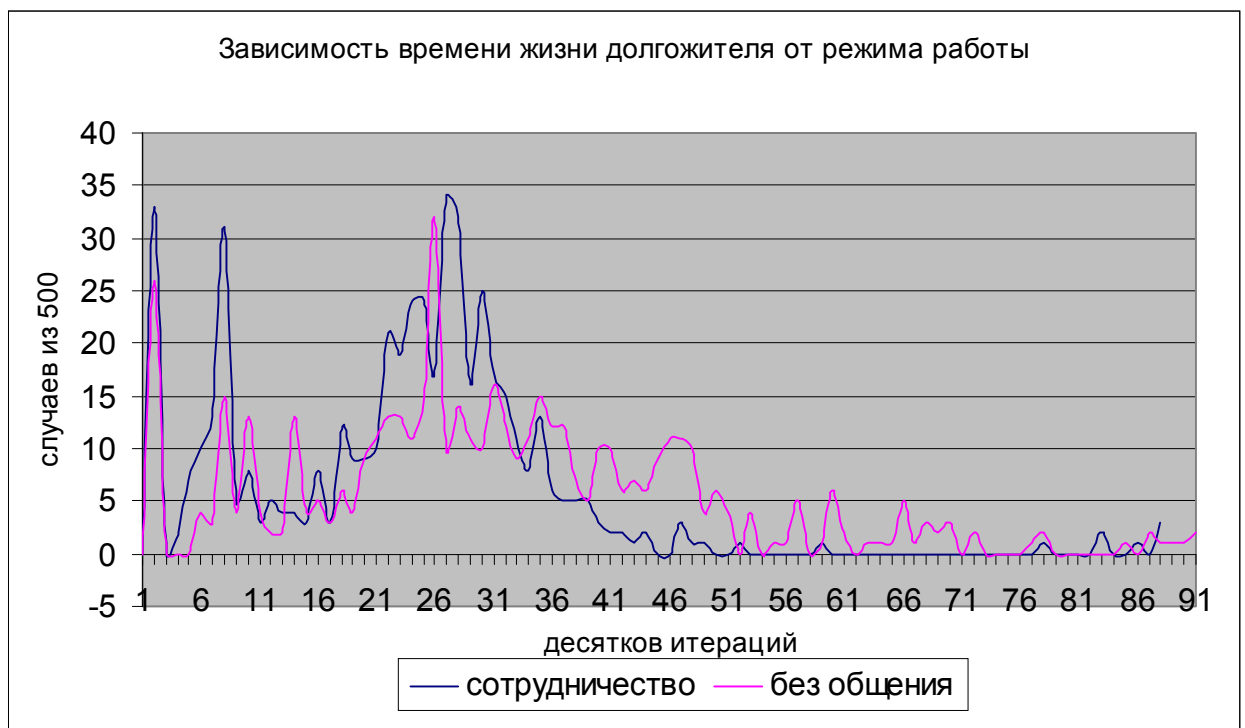


Можно сделать вывод, что включение режима кооперативного поведения мало влияет на среднюю продолжительность жизни, но влияет и в сторону ее уменьшения. Таким образом можно сделать вывод, что на данных условиях – на карте редкие и богатые ресурсы режим сотрудничества неэффективен.

Исследование задачи максимальной продолжительности жизни («Популяция»)

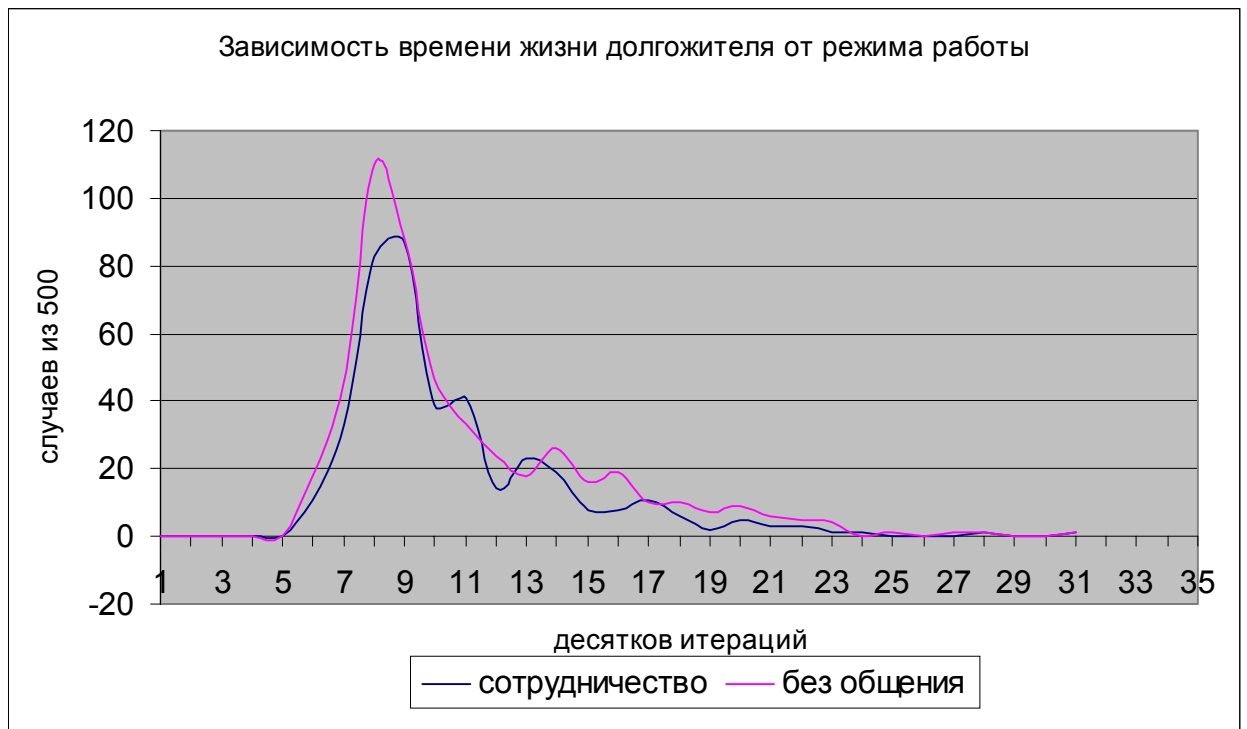
Эта задача показывает, насколько может измениться продолжительность жизни наиболее удачливого индивидуума в случае коллективного поведения и в случае индивидуального поведения роботов.

Рассмотрим случай дефицита ресурсов. Возьмем карту размерами 8*8, плотность распределения ресурсов 0.05, еды на карте: 200, скорость поедания $s=4$, Роботов на поле =4, начальная энергия роботов - 20, режим коммуникаций включен, вероятность вызова других роботов $Pr=1$

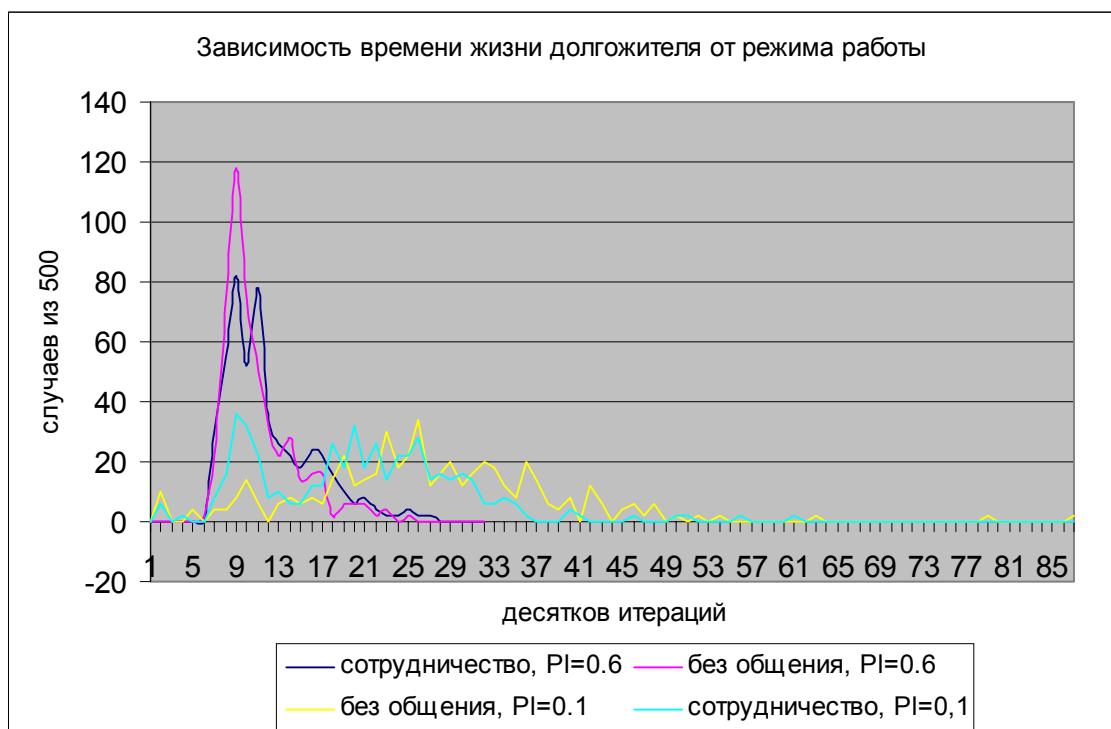


Чтож, как видим, в таких скудных условиях режим сотрудничества не приносит реальной пользы, более того, он приводит к снижению максимальной продолжительности жизни, достаточно взглянуть на достаточно показательный диапазон итераций от 40 до 75.

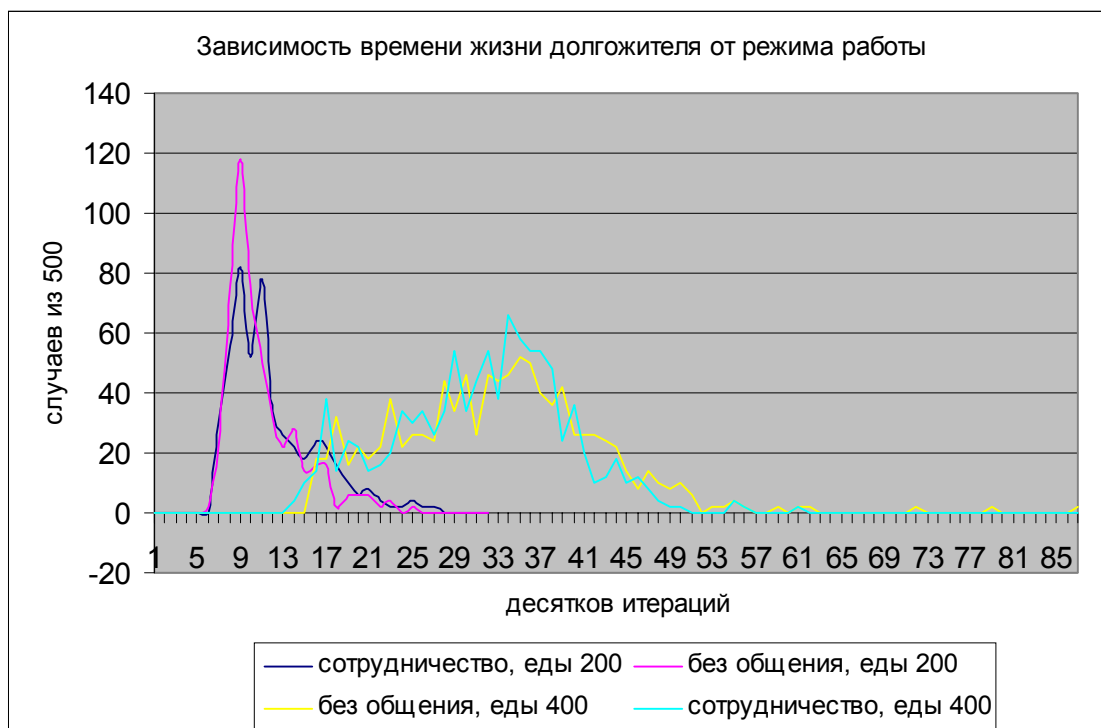
Если рассматривать случай большей плотности, $Pr=0.6$, то графики будут таковыми



(условия: Рабочее поле [8x8], плотность распределения пищи $PI=0.1$, Еды на карте: 200, скорость поедания $s=2$, Роботов на поле 4, начальная энергия роботов - 20, режим коммуникаций включен, вероятность вызова других роботов $P_r=1$)



Попробуем изменить количество пищи на карте, используем предыдущие условия но добавим пищи в 2 раза, с 200 до 400 и проведем эксперимент



Таким образом видно, что простое включение режима сотрудничества между роботами в общем не дает индивидуумам прироста длительности жизни ни в случае слабой ни в случае большой равномерности распределения ресурсов на карте.

Исследование задачи быстрого сбора ресурсов («Все ресурсы»)

В задаче сбора ресурсов главным критерием эффективности работы является время сбора всех ресурсов на карте. При решении задачи я использовал следующие допущения:

- Число роботов-сборщиков есть число конечное и известное.
- Роботы имеют неограниченный запас энергии для жизнедеятельности.
- Передвигаться роботы могут только в 4 направлениях, согласно моделируемому миру.
- Роботы могут передавать друг другу информацию о количестве еды в данной клетке и могут делать это каждый ход, независимо от расстояния друг от друга.
- Получив информацию об очередной найденной точке с ресурсами, робот может решить, что, учитывая расстояние до нее и скорость сбора, он гарантированно не успеет и отказаться от предложения, продолжив свой путь далее.
- Если же робот уже движется по направлению к вызванной точке и принимает новое предложение, он может вычислить, до какой из точек ему идти быстрее, а главное – выгоднее, учитывая оставшееся количество ресурсов на ней. Робот может соответственно поменять направление.
- Количество ресурсов на карте во время экспериментов одинаковое
- Скорость сбора ресурсов устанавливается заранее, до начала экспериментов
- Все параметры роботов во время эксперимента неизменны

Таким образом я смоделировал ситуацию сбора «полезных ископаемых» на определенной территории без препятствий, где роботы могут перемещаться свободно по любой из осей декартовых координат.

Учитывая малую эффективность свободного поиска, где направление выбирается случайным образом, следует ожидать, что взаимопомощь роботов может существенно повлиять на результат, причем как в положительную сторону, так и в отрицательную.

Рассмотрим два крайних случая:

1. Плотность точек с наличествующими ресурсами очень велика, а точки ресурсами бедны. Тогда эффективность общения может оказаться

отрицательной – роботы будут постоянно звать друг друга, практически не занимаясь поиском, как таковым. Более того, возрастает вероятность, что робот не успеет дойти до точки, как все ее ресурсы уже будут вычерпаны.

2. Плотность точек с ресурсами очень мала, точки очень богаты ресурсами. Тогда большую часть времени занимает малоэффективный свободный поиск. Но редкие вызовы оказываются очень полезными – на вызов приходят сразу много роботов и совместно они вырабатывают ресурс точки очень скоро, после чего разбредаются вокруг с свободным поиске, если за время работы не поступило еще вызова, конечно.

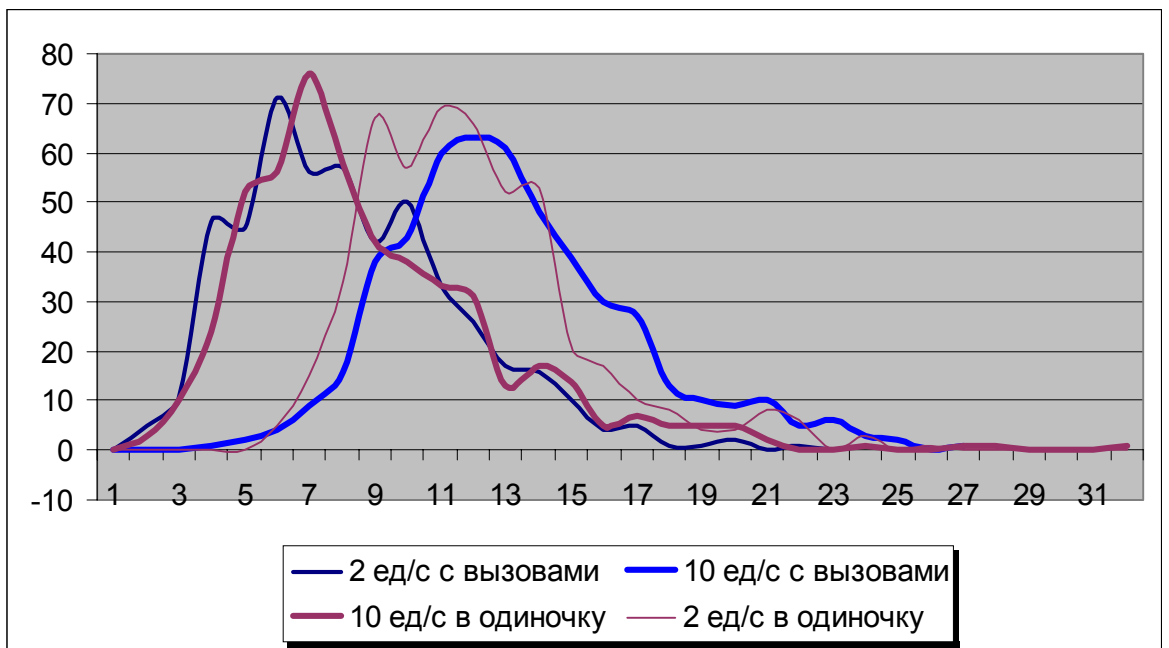
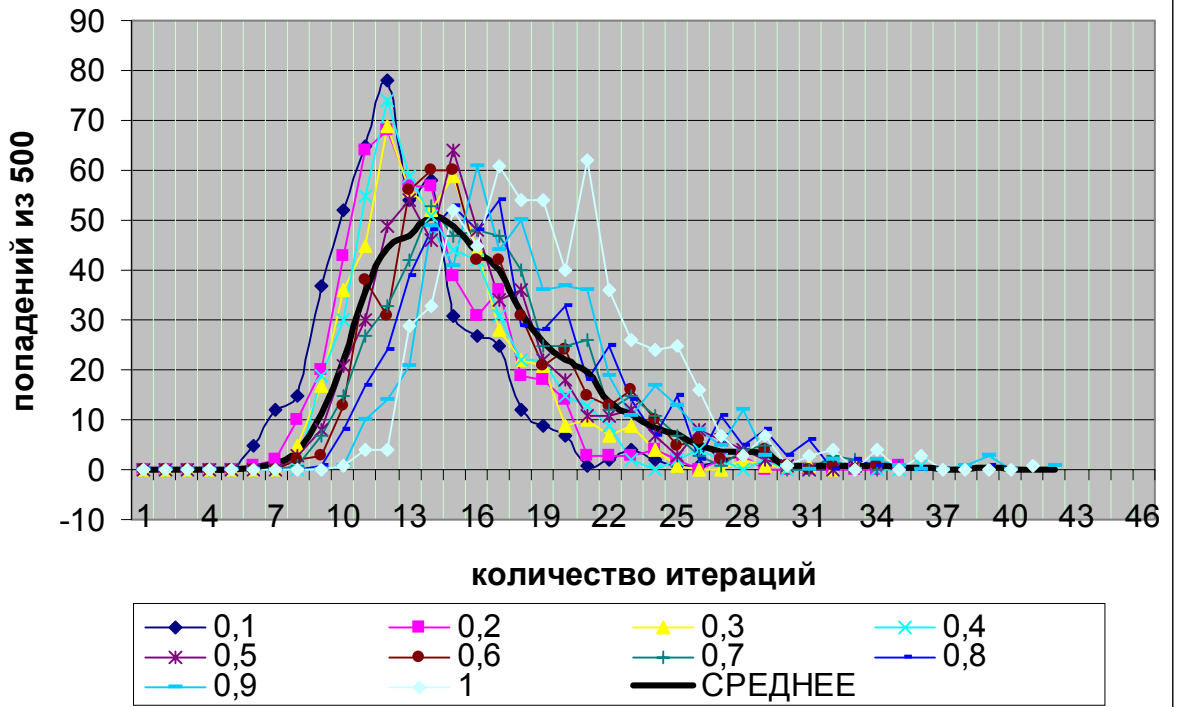
Здесь возникает закономерный вопрос – насколько богатым или бедным должно быть месторождение, чтобы оправдать метод коллективного поиска? Какова должна быть вероятность вызова роботов для определенного месторождения, чтобы эффективность физических роботов была максимальной?

Проведем для этого моделирование на разработанной мною и описанной в следующих главах программе.

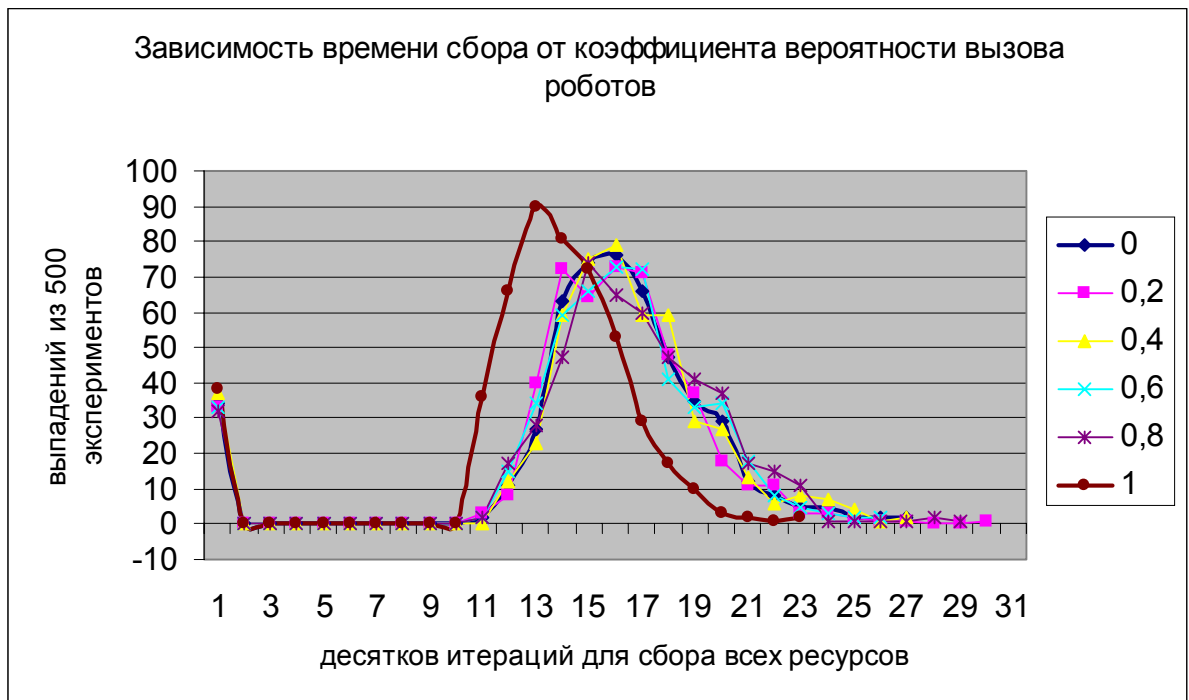
В качестве начальных условий возьмем поле размерами 10×10 , количество ресурсов на всем поле пусть ожидается в 300 единиц, количество роботов, участвующих в поиске равно четырем, проведем серию в 500 экспериментов.

Для ответа на вопрос о богатости месторождения, используем скорость сбора 2 ед/итерацию (это будут более дешевые роботы, не столь быстрые как при скорости 10 ед/итерацию), для режима общения выберем вероятность вызова в 100%.

Зависимость времени сбора от "богатости" месторождения при равных прочих параметрах



На решении этой задачи видно, что для реальной задачи сбора ресурсов реальными роботами, имеющими ограниченную скорость сбора становится эффективным использовать режим общения. Для уточнения проведем следующий эксперимент:



богатое месторождение (400 ед. ресурсов на карте) с очень редкими залежами, между которыми достаточно большое расстояние. Рабочее поле [6x6], плотность распределения пищи $P_l=0.07$, скорость сбора $s=2$, то есть роботы простые, роботов на поле 4, режим коммуникаций включен, вероятность вызова других роботов P_r меняется от 0 до 1 с шагом 0.2

Здесь уже хорошо видна эффективность режима общения – месторождения подобного рода следует разрабатывать с использованием кооперативного режима, и оно будет разработано быстрее, то есть за меньшее время.

Выводы

В ходе серии экспериментов, проведенных с целью установить эффективность режима сотрудничества между роботами, было выяснено, что:

1. Режим кооперативного поведения не приносит пользы в случае задачи увеличения продолжительности жизнедеятельности, причем результат этот не зависит
2. Режим кооперативного поведения не приносит заметной пользы в случае задачи увеличения средней продолжительности жизни индивидуума, лишь наоборот, уменьшает.
3. Режим кооперативного поведения очень полезен в задаче сбора ресурсов, причем тем более, чем реже встречаются залежи ресурсов на исследуемой территории.
4. В задаче сбора ресурсов большую роль играет скорость сбора ресурсов каждым роботом. Действительно, при большой скорости робот может сам выработать богатую клетку месторождения еще до того, как вызванные им роботы подойдут на помощь, следовательно они будут идти зря, что понижает продуктивность. Поэтому данный режим особенно эффективен для не скоростных роботов.
5. В имитационной программе не учитывается разница в скорости быстрого следования и изучения поверхности, в обоих случаях роботы движутся с одной скоростью, изучая поверхность на своем пути. Возможно, что если увеличить реалистичность и сделать скорость следования по вызову выше скорости изучения клетки, которая сейчас равна одной итерации, то эффективность работы роботов значительно возрастет.

Глава 3. Описание программы

В третьей главе описываются модули, интерфейсы, приведены системные соглашения по структурам данных. В данной главе производится описание разработанной системы на структурном уровне.

Для реализации проекта мною была написана программа, моделирующая поведение роботов, с помощью которой можно ставить эксперименты и моделировать различные ситуации.

Программа эта написана на языке PHP 4.01, который в результате своей работы выдает код на языке HTML, который, в свою очередь интерпретируется браузером пользователя – то есть любой программой просмотра Интернет - страниц. Таким образом не требуется специальной среды (специальной имитационной программы) или особенного клиентского приложения на стороне пользователя.

Программа состоит из одного *.php файла под названием **robots.php**, подкаталога \css со дополнительным файлом стилей для отображения на экран и подкаталога \img с набором графических файлов, служащих для облегчения восприятия на экране. Общая структура программы такова:

```
\
|-\css
|--- style.css           - вспомогательный файл, требуется для HTML
|-\img
|--- bot*_*.gif         - изображения роботов
|--- bg_lev*.jpg        - изображения клеток фона
|- robots.php           - запускаемый файл
```

где в bot*_*.gif вместо звездочек указаны цифры, первая цифра от 0 до 3 – номер робота, вторая цифра от 0 до 4 – направление движения робота на картинке; в файлах bg_lev*.jpg вместо звездочки уровень насыщенности клетки карты в числовом значении, от 0 до 5, подробное описание приведено ниже.

На данный момент вся программа расположена на сервере <http://dimmx.olmisoft.com/ii/robots.php>, и доступна для всех желающих работать с ней.

В программе была использована технология объектно-ориентированного программирования и широко использовались структуры и классы объектов.

Схема работы программы Robots:

Рисунок 8. Архитектура программы robots

На рисунке 8 показана схема работы программы. Программа, размещенная по определенному адресу в Интернете на компьютере, называемом сервером всегда находится в ожидании запуска. Любой пользователь Интернета, набрав в своем клиентском приложении (браузере) адрес этого сервера и имя программы, может запустить программу и получить результат работы. Соединение устанавливается через сеть Интернет. Пользователь – исследователь может запускать программу столько раз, сколько ему будет нужно. Поскольку браузер, клиентская часть де-факто есть у каждого пользователя компьютера,

подключенного к интернету, то разработанную мною программу можно считать доступной для любого человека.

Описание программы Robots.

Работа программы начинается с инициализации класса `cl_game`. Это основной класс программы, который состоит из 19 функций:

- `function Start ($display);`

.....
Функция запуска итерационного процесса. предварительно следует подготовить карту и роботов функцией `Prepare ()`

.....
Функция принимает бинарный параметр, означающий отображать процесс графически или нет. Возвращает количество итераций при заданных условиях.

- `function Prepare ();`

.....
Функция подготовки рабочего поля. Расставляет бонусы по клеткам в зависимости от параметров, размещает случайным образом роботов по карте и задает им начальное направление.

- `function LiveCriterion ();`

.....
Функция проверки, следует ли продолжать итерации дальше либо завершить процесс на текущей итерации.

- `function BotActions ($i);`

.....
Функция, реализующая выбор действий для *i*-го робота. В этой функции происходит выбор следующего действия.

- `function AddToStack ($i, $call_to_x, $call_to_y, $field_energy)`

.....
Функция, добавляющая в очередь вызовов *\$i* робота точку с координатами [*\$call_to_x*][*\$call_to_y*.] с учетом расстояния до нее, скорости сбора энергии(еды) и количества энергии(еды) в ней. Именно в этой функции заложен механизм отказа от вызова, если расстояние слишком велико и робот гарантированно не успеет ничего собрать.

- `function DeleteCallFromStack ($bot, $number_of_call)`

.....
Функция, удаляющая ненужную запись заданному роботу из его очереди вызовов.

- `function ResortCalls ($i)`

.....
Функция, оптимизирующая работу алгоритма поведения роботов при сотрудничестве. Все текущие вызовы сортируются по степени их «полезности». Эта полезность вычисляется по формуле:

.....
Полезность = ресурсов в точке / (расстояние * скорость сбора)

Если полезность меньше единицы, точка вызова считается неактуальной и удаляется.

- `function MakeChoice($i, $field_energy);`

Функция выбора - звать роботов или нет, зависит от текущего состояния, от коэффициента вероятности вызова и от количества энергии на клетке.

- `function SelectNewDirection ($i);`

Функция выборки нового направления в зависимости от текущего состояния - идет или стоит и поглощает пищу/энергию. В последнем случае если еще есть что собирать, робот остается на месте.

- `function Eat ($i, $AmountToEat);`

функция реализующая потребление пищи (увеличение энергии у робота, уменьшение на карте)

- `function CallBots ($current_robot, $mode);`

функция обращения к другим ботам. Тип обращения задается переменной \$Mode, реализованы обращения:

`$Mode = "callevryone"` - сообщить всем, что в клетке есть еда

`$Mode = "negative"` - сообщить всем, что заданная клетка уже пуста

- `function CalculateRandomDirection ($i);`

функция меняет направление движения робота случайным образом, при этом следит чтобы робот не мог пересечь границы поля и не двигался бесцельно назад, в сторону клетки откуда он только что пришел.

- `function MoveByDirection ($i);`

Функция реализует передвижение i-го робота на одну клетку в заданном направлении.

- `function CalculateDistance ($x1, $y1, $x2, $y2);`

Функция возвращает расстояние между двумя точками, расстояние должно вычисляться по формуле разностей квадратов, но в данном упрощенном случае, когда робот может выбирать направление только вдоль осей декартовых координат и препятствий на карте нет, расстояние то вычисляется разностью координат.

- `function CalculateDirection ($i, $to_x, $to_y);`

Вычисляет направление для i робота при движении к определенной точке.

- `function DisplayHeader ();`

Функция отображает на экране начальные строки HTML-документа по стандартам Интернета.

- `function DisplayForm ();`

..... Функция отображает на экране форму с текущими настройками, позволяя их изменять

- `function DisplayTask ();`

..... Функция отображает на экране условия задачи.

- `function DisplayFooter ();`

..... Функция отображает последние строчки HTML-документа согласно стандартам интернета.

- `function DisplayField ();`

..... Функция отображает текущее состояние поля в графическом режиме.

- `function DisplayBot ($x, $y);`

..... Функция отображает в клетке с координатами \$x, \$y всех роботов, которые там находятся, если они там есть.

- `function DisplayInitialPage ();`

..... Функция отображает начальное состояние программы - форму данных и условия задачи

Далее, класс содержит структуру `varOptions`, которая хранит текущие параметры запуска, которые можно изменить с помощью формы, вызванной функцией `DisplayForm ()`;

- `$this->varOptions->xx=6;`

..... Размерность поля по координате x, число должно быть целым и положительным.

- `$this->varOptions->yy=6;`

..... размерность поля по координате y, число должно быть целым и положительным.

- `$this->varOptions->P1=0.2;`

..... вероятность появления пищи в клетке. Измеряется от 0 до 1

- `$this->varOptions->Qmax = 300;`

..... Максимальное количество пищи на карте. Может быть немного меньше, но общее количество стремится к этому числу. Число должно быть положительным.

- `$this->varOptions->Nbots = 2;`

..... Количество роботов на карте, меняется от 1 до 4

- `$this->varOptions->DisplayGr=1;`

..... Показывать ли первую (пробную) итерацию графически? Если равно единице, то да, если нулю – не показывать.

- `$this->varOptions->InitialEnergy = 20;`

..... Начальная энергия робота

- `$this->varOptions->Iterations = 20;`

..... Сколько проводить итераций для достижения статистических данных. Меняется от нуля до 1000.

- `$this->varOptions->Communicate = 0;`

..... Переменная определяет - разрешить ли режим коммуникации между роботами или запретить. Если равна нулю – режим отключен. Если единица – включен.

- `$this->varOptions->ver = 0.9;`

..... Вероятность вызова другого робота при нахождении клетки с едой. Имеет смысл только при включенном режиме коммуникации. Изменяется от 0 до 1.

- `$this->varOptions->task="population";`

..... Исследуемая задача. Может принимать значения

"geological" – исследуется задача быстрого сбора ресурсов

"population"– исследуется задача наиболее продолжительной жизнедеятельности роботов из популяции.

- `$this->varOptions->Speed=4;`

..... скорость поглощения пищи/сбора ресурсов из клетки. Обозначает сколько единиц пищи робот может собрать за одну итерацию. Изменяется от 1 до 10000 (где 10000 - аналог бесконечно быстрого сбора).

Также программа использует некоторые глобальные переменные, жизненно важные для работоспособности программы.

- `$this->Iteration=0;`

..... номер текущей итерации

- `$this->Total=0;`

..... общее оставшееся количество энергии на карте, меняется в течение работы программы.

- `$arField = Array (Array());`

двухмерный массив чисел, содержащий состояние поле, значение каждой клетки массива содержит количество энергии на соответствующем поле

- `$this->field_message=""` ;

системное сообщение, отображаемое для пользователя о состоянии поля

- `$this->action_message=""` ;

системное сообщение, отображаемое для пользователя содержащее комментарии к действиям роботов на каждом шаге

Вся текущая информация о роботах содержится в структуре `varBot`, объявленной глобально и доступной для любой функции класса. Описание структуры `varBot` приведено ниже:

- `$this->varBot[$i]->x`

положение робота по горизонтали, координата X

- `$this->varBot[$i]->y`

положение робота по вертикали, координата Y

- `$this->varBot[$i]->direction`

направление движения робота. Изменяется от 0 до 4 :

0 – робот стоит, используется в случае, когда робот собирает ресурс

1 – направление робота налево, в сторону уменьшения координаты X

2 – направление робота вверх, в сторону уменьшения координаты Y

3 – направление робота направо, в сторону увеличения координаты X

4 – направление робота вниз, в сторону увеличения координаты Y

- `$this->varBot[$i]->Energy`

Энергия робота на текущей итерации. Изначально равняется значению параметра опций `$this->varOptions->InitialEnergy`, потом может изменяться в обе стороны – как увеличения, так и уменьшения. При задаче выживания на каждый шаг робот тратит 1 единицу энергии при поиске и 0 единиц при сборе пищи, при задаче сбора ресурсов энергия робота не играет никакой роли (в данном варианте написания программы).

- `$this->varBot[$i]->CallStackX[]`

Одномерный массив координаты Y очереди вызовов.

Вполне реальна ситуация, когда робота позвали, он спешит к источнику пищи, в это время его зовет другой робот, тогда следующий вызов он отложит в очередь, сходит к первому и потом уже пойдет на второй вызов, если его не отменят к тому времени (то есть если там не соберут уже все).

- `$this->varBot[$i]->CallStackY[]`
..... Одномерный массив координаты Y очереди вызовов.
- `$this->varBot[$i]->CurrentCall;`
..... Указатель на координаты в массиве актуального на данный момент вызова
- `$this->varBot[$i]->Calls;`
..... Общее количество вызовов. Если общее количество совпадает с текущим вызовом (с учетом нулевого), значит робота никто не зовет и он ищет пищу самостоятельно.

Общая структура программы

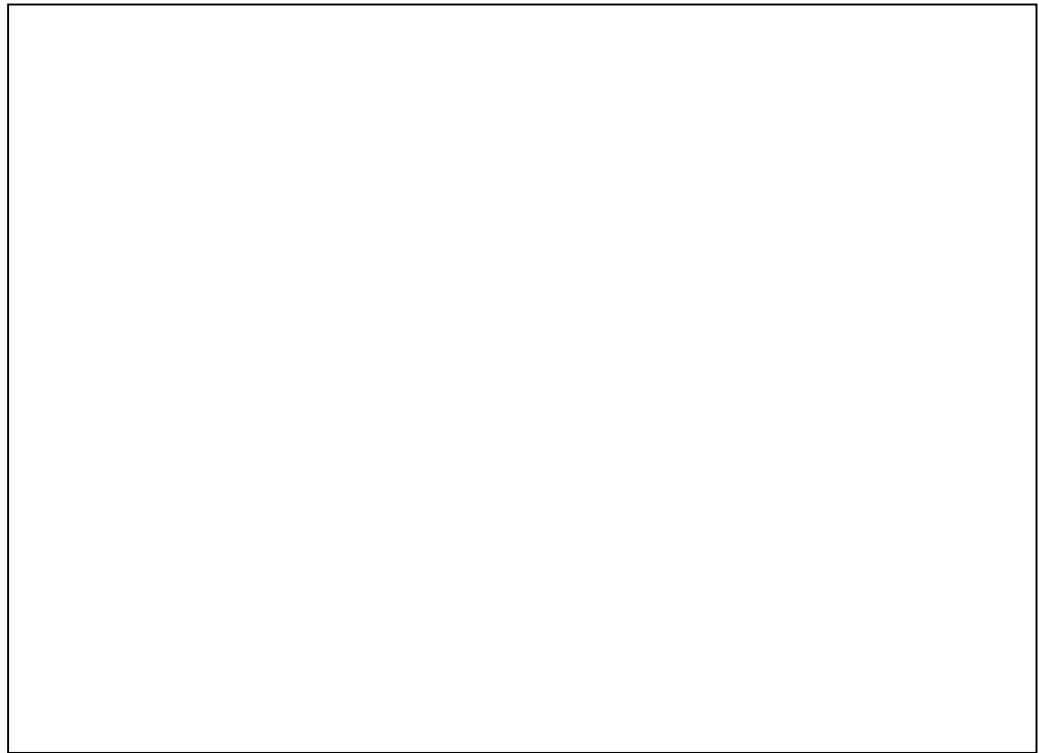


рисунок 9. Блок-схема работы программы

Программа состоит из трех основных модулей (рисунок 9):

- Подготовки
- Обработки данных
- Вывода результатов на экран

Обработка данных выполняется в цикле, пока необходимость в продолжении итераций все еще существует. В цикле на каждой итерации для каждого из роботов выполняются некоторые действия и состояние системы изменяется.

Функционирование программы

Программа начинается с инициализации класса `cl_game`:

```
$game= new cl_game;
```

далее идет проверка, запущена программа первый раз или уже была нажата кнопка “Go”, в первом случае вызывается функция `DisplayInitialPage ()`, во втором случае запускается серия экспериментов с параметрами, определенными на первом шаге.

Каждый эксперимент производится последовательным вызовом функций

```
$game->Prepare ();
```

```
$game->Start(0);
```

после чего результат работы функций сохраняется в переменной ассоциативного массива `$ITOG`.











В дальнейшем содержимое массива обрабатывается и выдается на экран в виде графика, в котором по оси *y* отображается число итераций, требуемых для завершения процесса на очередном эксперименте, а на оси *x* отображается частота случаев повторения данного результата при этих параметрах.

Рисунок 10. Пример графика, выводимого программой.

При этом на экран выводится форма, в которой можно увидеть, при каких параметрах был получен данный результат, изменить параметры или тип решаемой задачи и запустить эксперимент снова (Рисунок 12, Глава 4).

В случае графического отображения процесса деятельности роботов на экран выводится поле и комментарии к каждой итерации, подобные изображенным на рисунке 33

При этом графические обозначения приняты следующие:

-  - робот 0,
-  - робот 1,
-  - робот 2,
-  - робот 3
-  - клетка поля, не содержащая еды
-  - клетка поля, содержащая 1-10 единиц еды
-  - клетка поля, содержащая 11-20 единиц еды
-  - клетка поля, содержащая 21-30 единиц еды
-  - клетка поля, содержащая 31-40 единиц еды
-  - клетка поля, содержащая более 40 единиц еды

Как видно роботы выкрашены в разные цвета, чтобы было легко идентифицировать их, клетки поля тоже отображают некоторую информацию о своем содержимом.

Графики было применено очень мало, в основном текст. Это было сделано в целях оптимизации, ведь каждую картинку пользователю приходится скачивать на свою машину,

чтобы увидеть, а это как правило оплачивается. Поэтому в целях минимизации трафика (объема скачанной информации) количество изображений было сведено к минимуму.

Рисунок 11. Пример работы графического режима программы.

Выводы

Использование языка программирования PHP позволило в разумные сроки создать программу моделирования поведения мобильных роботов и среду обитания для них со свободно модифицируемыми параметрами. Применение технологии программирования под web позволило расширить аудиторию пользователей программы, таким образом больше людей, исследующих аналогичные темы в направлении создания искусственного интеллекта смогут воспользоваться моими наработками. В процессе создания программы были углублены познания в языках программирования. В дальнейшем планируется использование технологии динамического HTML для оживления взаимодействия с пользователем и расширение функциональных возможностей программы для моделирования более реалистичных условий взаимодействия роботов, а также возможно для исследования других возможностей кооперативного режима работы роботов.

Хотя программа позволяет исследовать сразу три типа задач, тем не менее есть еще большой запас по планируемой к реализации функциональности, то есть еще множество идей не реализовано и, благодаря прозрачности кода, их легко можно добавить позднее.

Например графический вывод результатов в виде диаграммы, добавление типов решаемых задач, усложнение окружающей среды, введение препятствий, усложнение алгоритма случайного поиска, оптимизация алгоритмов выбора целенаправления, общая оптимизация производительности, запись результатов и процесса в форматированный файл, чтение исходных данных из специального файла.

Глава 4. Работа с программой

В этой главе приводится описание функционирования системы, описывается взаимодействие с пользователем, рассматриваются достоинства и недостатки разработанной системы.

Работа с программой. Интерфейс.

Чтобы работать с программой **robots**, пользователю необходимо запустить Интернет-браузер (клиентскую программу для работы с Интернет-страницами) и набрать адрес, на котором расположена программа. На данный момент это адреса:

- ✓ <http://dimmx.olmisoft.com/ii/robots.php>
- ✓ <http://193.125.239.205/robots.php>

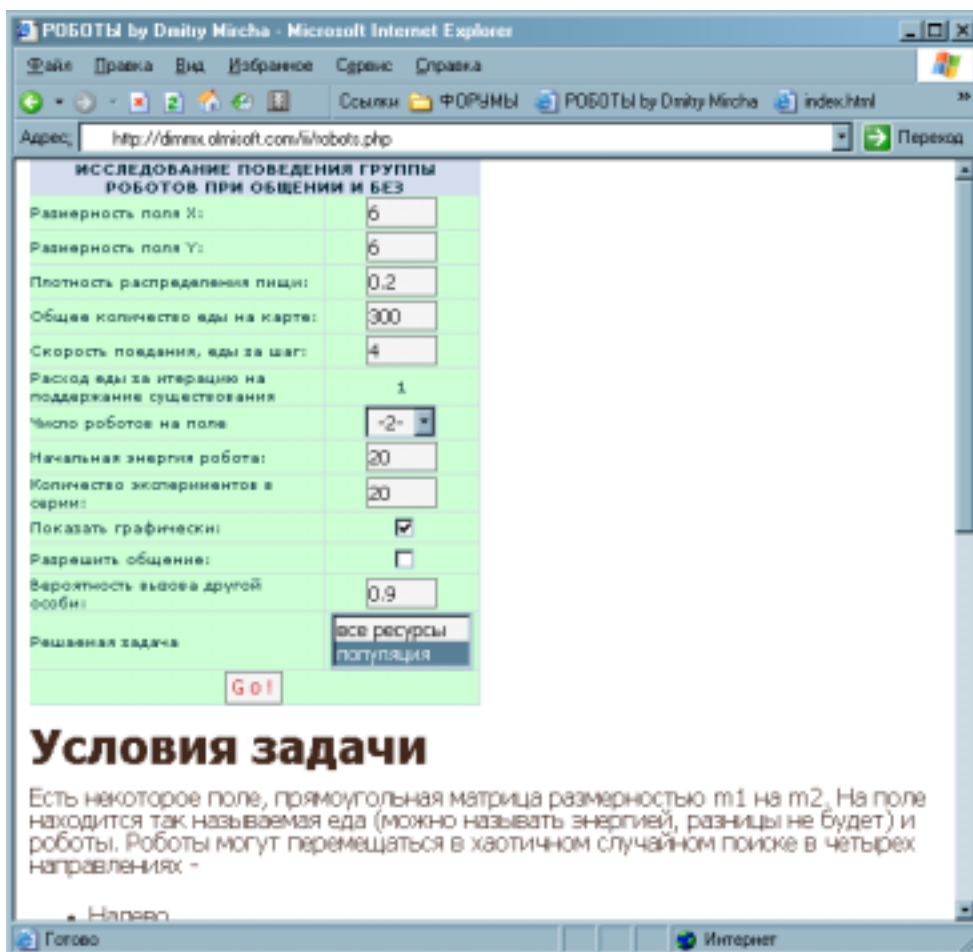


Рисунок 12. Начальный вид работы программы.

Набрав его, пользователь увидит форму (Рисунок 33), где он может оставить изначальные или изменить параметры поля и выбрать тип исследуемой задачи. Там же, на первой странице кратко описаны типы задач и находится краткое объяснение параметров, к чему приводит изменение каждого из них.

- **Размерность поля X:**
 -
Задается в условных единицах размер поля по ширине (координата X).
 -
Значение от нуля до 50.
- **Размерность поля Y:**
 -
Задается в условных единицах размер поля по высоте (координата Y)
 -
Значение от нуля до 50.
- **Плотность распределения пищи:**
 -
Задается в вероятность нахождения на клетке пищи, значение от 0 до 1.
- **Общее количество еды на карте:**
 -
Задается общий запас еды на всей карте, независимо от количества точек с едой. Это количество равномерно распределяется по тем клеткам карты, в которых вероятность ее появления выше заданной.
- **Скорость поедания, еды за шаг:**
 -
Для решения задачи геологических раскопок требуется такое приближение к реальности – робот не может за раз собрать все ресурсы на клетке, ибо иначе пропадает смысл вызова другого робота на эту клетку. Данный параметр сделан регулируемым для увеличения гибкости системы. Значение от 1 до 1000
- **Расход еды за итерацию на поддержание существования**
 -
Данный параметр сделан фиксированным и равным единице, но предусмотрено отображение в списке параметров, для большей понятности в работе.
- **Число роботов на поле**
 -
На поле может быть один, два или больше роботов. Соответственно меняются все результаты. Может быть от 1 до четырех.
- **Начальная энергия робота:**
 -
Роботы для поиска ресурсов (еды) используют механизм случайного выбора направления, чтобы они не погибали на первых же итерациях, им сделан запас еды, во время пользования которым они могут найти новые источники пищи и пополнить этот запас.
- **Количество экспериментов в серии:**

После отображения графически первой итерации, алгоритм запускается указанное этим параметром раз вновь, но уже без подробной детализации. Графическое отображение первой итерации можно и отключить, см. описание следующего параметра.

- Показать графически:

Иногда не важно как именно происходил эксперимент, нужна лишь статистика, для этого предусмотрено отключение графического показа первой итерации, что позволяет сэкономить и время, и трафик пользователя.

- Разрешить общение:

Ключевой и очень важный параметр. При отключении этого параметра роботы теряют возможность взаимодействовать друг с другом и так можно легко проследить отличие коллективного поведения при заданных параметрах от одиночного, где каждый действует сам по себе.

- Вероятность вызова другой особи:

Вероятность, что один робот позовет другого, задается коэффициентом

- Решаемая задача

Тип решаемой задачи. На текущий момент реализованы задачи эффективного поиска ресурсов (геологическая задача) за минимальное время и задача выживания популяции, где критерием эффективности является среднее время жизни особи.

После установки всех параметров и нажатия на кнопку “Go” происходит запуск функций моделирования, и если был разрешен графический вывод на экран, то каждый шаг итерации первого раунда выводится на экран с необходимыми пояснениями относительно происходящего в этом раунде. Пример графического вывода показан на рисунке 13

В конце работы программы выводится таблица с результатом исследования – статистические данные по запуску серии экспериментов (рисунок 14).

Рисунок 13. Графическое отображение работы программы.

Рисунок 14. результат работы программы – статистические данные.

Достоинства и недостатки разработанной системы

Среди основных недостатков разработанной программы можно отметить следующие:

1. При отсутствии у исследователя доступа к сети Интернет, он не сможет воспользоваться программой. Хотя существует возможность установки у себя локальной копии сервера и программы, этот путь редко реализуем на практике.
2. Если у пользователя очень медленное соединение с сетью Интернет, то при задании слишком большого количества экспериментов он может не дожидаться результатов – его клиентская программа прекратит прием данных, решив, что соединение потеряно. Решением является настройка браузера либо переход на другой канал доступа к Интернету.
3. Программа не обладает возможностью динамического показа пользователю последовательности событий, в силу того, что результат работы PHP-скрипта отображается как статическая страница.
4. На данный момент программа не поддерживает возможности переключаться на другие языки, например, на английский, во время работы. Хотя сама она может быть переведена целиком.
5. Программа слабо оптимизирована по производительности с целью сохранения прозрачности кода для будущих разработчиков.
6. Производительность программы существенно зависит от производительности сервера, на котором она расположена и от числа одновременно работающих с ним пользователей.

Среди основных достоинств разработанной программы можно отметить следующие:

1. То, что программа доступна из сети Интернет, существенно расширяет аудиторию пользователей, большее число пользователей может ознакомиться с исследованиями на данную тему и провести собственные эксперименты без установки или скачивания дополнительных программных модулей.
2. Программа написана на современном языке программирования PHP, который легко сопровождать и редактировать.

3. Программный код очень хорошо документирован мною и во всех ключевых местах расположены комментарии на русском языке.
4. Программа написана с использованием структур и классов, что облегчает чтение кода и понимание сути работы.
5. Программа использует стандартный интерфейс, понятный любому пользователю сети Интернет.
6. Есть возможность сменить тип решаемой задачи и изменить все ключевые параметры среды и роботов, что обеспечивает гибкость в исследованиях.
7. Графическое отображение ситуации с комментариями происходящего позволяет наглядно понять суть взаимодействия и оценить работу алгоритма.
8. Запуск многократной серии экспериментов с теми же параметрами, но без графического отображения позволяет оценить статистически эффективность поведения роботов в заданной ситуации.
9. Вывод результатов работы в виде гистограммы повышает наглядность, вывод же в числовом значении можно использовать в дальнейшей работе.

Выводы

Написанная программа обладает дружественным интерфейсом, легким для понимания и восприятия даже начинающим пользователем компьютера, имеющим навыки работы с сетью интернет, поскольку использует те же инструменты отображения что и обычные интернет-страницы. Комментарии при включенном графическом режиме позволяют хорошо представить суть происходящего на карте. Отображение гистограммы, наряду с числовыми данными повышает наглядность и увеличивает воспринимаемость результатов.

Программа, однако имеет ряд недостатков, в числе наиболее значимых из них – необходимость доступа к Интернету для работы с ней и статичность действий, нет возможности наглядно, в динамике, показать процесс работы, только «покадрово».

Тем не менее программа функционирует, выдает результаты и позволяет успешно исследовать поставленную задачу путем анализа табличных данных.

Заключение

В работе рассмотрена задача использования коллективного поведения роботов при решении задачи сбора ресурсов и выживания вида. Исследовалась степень необходимости использования кооперативного режима для разных начальных условий и были сделаны соответствующие выводы.

Для решения задачи была написана программа, моделирующая поведение роботов в итерационном процессе. Решение задачи программным методом вместо математического моделирования позволило сэкономить много времени для постановки множества экспериментов, имея требования к построению роботов, полученные программно для решения задачи при заданных условиях можно приступить к созданию физических роботов.

Программа была написана на языке PHP для серверного компьютера. Такой подход позволяет любому исследователю, подключенному к сети Интернет, использовать программу, запуская ее на сервере в своих нуждах, без установки ее на своем компьютере. Очень важным является тот факт, что программа является очень ресурсоемкой и требует весьма производительного компьютера для решения задачи в приемлимое время, что имеется не у каждого исследователя.

В программе создана система, моделирующая окружающую среду, роботов и реализацию взаимодействия мобильных роботов, предусмотрена возможность исключать возможность общения между ними и задавать параметры окружающей среды. При разработке этой системы были выбраны и реализованы механизм доступа к данным, механизм генерации сообщений и их распознавания, механизм отображения текущего состояния, был выработан общий подход к применению Internet-технологий для создания программ. Принципиальным моментом является то, что система не требует наличия у пользователя специальных программ, данных и знаний для работы с программой моделирования ИИ, что расширяет сферу её возможного применения. Применение современного языка программирования PHP для генерации динамических HTML-страниц позволяет легко добавлять новые элементы в систему и расширять её функциональные возможности.

В работе был произведен также обзор существующих систем моделирования поведения мобильных роботов на примере игры роботов в футбол и проекта «Террариум». Был рассмотрен пример программной модели робота с индивидуальным поведением – Анимата - автономного агента с поведением, схожим с принципами поведения животных.

Используемая литература:

1. Чемпионаты мира среди роботов по футболу - <http://www.robocup.org/>
2. Виртуальный футбол роботов - <http://n-th.com/soccer/>
3. М.Б.Игнатъев, Самоорганизующиеся робототехнические системы и игра в футбол. - <http://dept44.aanet.ru/Homepages/Ignatiev/Ignat9.htm>
4. Алгоритмы, используемые в играх - <http://www.codenet.ru/progr/other/games/08.php>
5. Всероссийский научно-технический фестиваль молодежи "Мобильные роботы-2003" - <http://users.imec.msu.ru/fmrobot/>
6. алгоритмы работы реальной команды n-th.com (Днепр) - <http://n-th.com/soccer/Dnepr.htm>
7. <http://n-th.com/soccer/manual.zip> - документация по подготовке и компиляции модуля, а также обо всех функциях SDK для модели лиги СНГ.
8. <http://www.ai.tsi.lv/soccer/> - Виртуальный футбол роботов – сервер с исходниками сервера футбола роботов -
9. Роботы играют в футбол - http://www.osp.ru/cw/2002/20/028_1_1.htm
10. Microsoft Terrarium - <http://www.windowsforms.net/default.aspx?tabIndex=6&tabId=42>
11. Описание игры Terranium - http://www.gotdotnet.ru/default.aspx?s=doc&d_no=586&c_no=37
12. Результаты игры в Terranium - <http://www.terrariumgame.net/terrarium/charts/chartnew.aspx>
13. Алгоритмы планирования коллективных действий при игре в футбол, Стоянов С.В., НИИ МВС ТРГУ - http://www.rtc.neva.ru/konf_10/Stoyan/stoyanov.htm
14. Каляев И.А., Капустян С.Г., Усачев Л.Ж. Основы построения распределенных систем управления коллективами роботов // Информационные технологии.-М.: Машиностроение, Й5, 1998.- с.13-18.
15. Капустян С.Г., Усачев Л.Ж., Стоянов С.В. Метод оптимального распределения целей в коллективе роботов // Информационные технологии.- М.: Машиностроение, №4, 1998.- с.29-34.
16. RoboCup // RoboCup papers at ICRA-98 and DARS-98. RoboCup Federation, Spring, 1998
17. В.А. Непомнящих. Поиск общих принципов адаптивного поведения живых организмов и аниматов // AI News – Новости искусственного интеллекта, №5(50), 2002.- с.48-53.

18. Gelenbe E., Schmajuk N., Staddon J., Reif J. Autonomous search by robots and animals: A survey. *Robotics and Autonomous Systems*. 1997. V.22. №1. P.23-34
19. Maes P. Modeling adaptive autonomous agents. *Artificial Life Journal*. 1994. V.1. №1-2. P. 135-162. <http://pattie.www.media.mit.edu/people/pattie/cv.html#publications>
20. Wilson S.W. (1991). The Animat Path to AI. In: Meyer J.A., Wilson S.W. (Eds.), *From animals to animats: Proceedings of the First International Conference on Simulation of Adaptive Behavior*. Cambridge, MA: The MIT Press/Bradford Books, 1991. P. 15-21. <http://world.std.com/~sw/>
21. Webb B. What does robotics offer animal behaviour? *Animal Behaviour*. 2000. V.60. №5. P.545-558. <http://www.stir.ac.uk/Departments/HumanSciences/Psychology/Staff/bhw1/index.htm>
22. Smithers T. Autonomy in Robots and Other Agents. *Brain and Cognition*. 1997. V.34. №1. P.88–106. <http://www.eteo.muni.es/dptos/informatica/tsmithers/index.php>
23. Документация языка программирования PHP - www.php.net

ПРИЛОЖЕНИЕ 1:

НЕКОТОРЫЕ РЕСУРСЫ В ИНТЕРНЕТЕ

International Society for Adaptive Behavior (ISAB) <http://www.isab.org.uk/>

Страницы исследовательских лабораторий

Adaptive Systems Research Group <http://homepages.feis.herts.ac.uk/~comqcln/ASRG.html>

AI Lab, Zurich Univ., Switzerland <http://www.ifi.unizh.ch/groups/ailab/>

AnimatLab, Paris, France http://www-poleia.lip6.fr/ANIMATLAB/animat_accueil_eng.html

Bio-Robotic Lab, Case Western Univ., Cleveland, USA <http://biorobots.cwru.edu/>

COGs, Sussex Univ., UK <http://www.cogs.susx.ac.uk/index.html>

Laboratoire Bioinformatique, Lausanne, Switzerland <http://diwww.epfl.ch/Khepera/>

Los Alamos National Lab, USA <http://www.nsl.lanl.gov/projects/robot/>

Mobile Robot Lab, Georgia Inst. Tech., USA <http://www.cc.gatech.edu/ai/robot-lab/>

3 Sigma Robotics <http://www.3sigmarobotics.com>

Персональные страницы

Randall Beer <http://vorlon.cwru.edu/~beer/>

Rodney Brooks <http://www.ai.mit.edu/people/brooks/brooks.html>

Marco Dorigo <http://iridia.ulb.ac.be/~mdorigo/dorigo.html>

Dario Floreano <http://dmtwww.epfl.ch/isr/east/team/floreano/>

Auke Jan Jjspeert <http://rana.usc.edu:8376/~ijspeert/index.html>

Thiemo Krink <http://www.daimi.au.dk/~krink>.

Orazio Miglino <http://gral.ip.rm.cnr.it/orazio/>

George Mobus <http://www.tacoma.washington.edu/css/gmobus/index.html>

Stefano Nolfi <http://gral.ip.rm.cnr.it/nolfi/>

Luc Steels <http://arti.vub.ac.be/steels/>

Peter Todd <http://www.mpib-berlin.mpg.de/users/ptodd/>

Xiaoyuan Tu <http://www.dgp.toronto.edu/people/tu/>

Stewart Wilson <http://world.std.com/~sw/>

Tom Ziemke <http://www.ida.his.se/ida/~tom/>

Библиографии со ссылками на электронные публикации

Biomorphic Robotics:

<http://www.iguana-robotics.com/RobotUniverse/BiomorphicRobots.htm>

Online ALife Publications <http://www.cogs.susx.ac.uk/users/ezequiel/alife-page/alife.html>

Zooland: The Artificial Life Resource <http://alife.ccp14.ac.uk/zooland/zooland/>

Complexity Papers Online <http://www.calresco.org/papers.htm>

Статьи и учебники

Нейронные сети – основные модели. И.В. Заенцев, уч. пособие, ivz@ivz.vrn.ru

Amit's Game Programming Information –

<http://www-cs-students.stanford.edu/~amitp/gameprog.html>

Game AI: The State of the Industry -

http://www.gamasutra.com/features/20001101/woodcock_01.htm

Designing Need-Based AI for Virtual Gorillas -

http://www.gamasutra.com/features/designers_notebook/20001222/adams_pfv.htm

Hierarchal AI -

<http://www-cs-students.stanford.edu/~amitp/Articles/HierarchalAI.html>

Machine Learning in Games - <http://satirist.org/learn-game/>

Map AI - http://people.mw.mediaone.net/markeverson/map_ai.htm

Practical Guide to Building a Complete Game AI: Volume I

http://www.lupinegames.com/articles/prac_ai.html

Strategy and Tactics: DreamWeaver's Thoughts -

<http://www-cs-students.stanford.edu/~amitp/Articles/StrategyAndTactics.html>

The Game AI Page Building Artificial Intelligence into Games -

<http://www.gameai.com/ai.html>

ПРИЛОЖЕНИЕ 2.

Текст программы «Robots»