УДК 519.876.5+004.896+681.5
DOI: 10.15827/0236-235X.123.581-590

Дата подачи статьи: 29.01.18
2018. Т. 31. № 3. С. 581–590

Система моделирования поведения групп робототехнических агентов с элементами социальной организации Кворум

В.Э. Карпов [1, 2], *к.т.н., доцент, руководитель Отделения нейрокогнитивных наук и интеллектуальных систем, karpov.ve@gmail.com*
М.А. Ровбо [1], *инженер-исследователь, rovboma@gmail.com*
Е.Е. Овсянникова [1], *инженер-исследователь, eeovsyan@yandex.ru*

[1] *Национальный исследовательский центр «Курчатовский институт» , г. Москва, 123182, Россия*
[2] *Московский физико-технический институт (государственный университет), г. Долгопрудный, 141701, Россия*

В работе описана архитектура среды многоагентного моделирования Кворум, разработанной для реализации конструктивных решений и алгоритмов индивидуального поведения агентов-роботов, а также для моделирования поведения больших групп роботов. Основной задачей системы является инструментальное обеспечение исследований в области применения методов социальной организации в групповой робототехнике.

Кворум – это прототип системы имитационного и агентного моделирования, которая должна адекватно абстрагировать сложность физической системы, обеспечивая удобные интерфейсы и библиотечные модули для моделирования групп мобильных роботов и отдельных внутренних структур агента. Она была построена как модульная, легко расширяемая система, ориентированная на моделирование систем с большим количеством агентов за счет упрощения физических и других эффектов, а также благодаря способности выполнять моделирование с использованием параллельных вычислений.

Показано, что предлагаемый метод моделирования подобных систем применим к широкому спектру задач из области групповой робототехники с элементами социальной структуры (а также роевой и групповой робототехники в целом). Некоторые особенности архитектуры и модели, используемые в Кворум, позволяют расширить ее в последующих итерациях для полной поддержки моделирования на параллельных вычислительных системах.

Одной из важных особенностей Кворум является ориентация на управление реальными робототехническими системами. Для этого система обеспечивает использование одинаковых интерфейсов управления как для вычислительной модели, так и для реальных роботов.

Система была апробирована на ряде характерных для групповой робототехники задач, особенно тех, которые используют механизмы социальной структуры, а также на группе лабораторных роботов разных типов, поддерживающих интерфейс системы моделирования Кворум.

Ключевые слова: *агентное моделирование, моделирование социального поведения, групповая робототехника, имитационное моделирование, инструменты разработки, кластерные вычисления, биоподобные системы, эусоциальность, робот, искусственная жизнь.*

Важной задачей групповой робототехники является построение такой системы, которая давала бы новые качества и улучшенные характеристики за счет взаимодействия многих отдельных агентов, обладающих относительно простыми правилами. Основным методом исследования этих систем является агентное моделирование, для которого исследователи используют различные среды и библиотеки.

Одним из направлений исследований в области групповой робототехники является изучение биологических систем с целью применения принципов их организации для построения команды роботов, поскольку они демонстрируют множество желательных качеств, таких как устойчивость к внешним возмущениям, умение работать в неизвестных условиях и эффективность. Новым и перспективным направлением в этой области является подход, основанный на применении в групповой робототехнике моделей и методов социального поведения [1]. Далее искусственные группы, в которых применяются эти механизмы, будем называть группами с элементами социальной структуры. Составляющими этих групп являются агенты. Под ними будут пониматься реальные или виртуальные объекты, вычислительные системы, которые существуют в некоторой сложной динамичной среде, могут воспринимать окружающую среду через датчики и автономно действовать на нее с помощью эффекторов на основе полученной информации [2], тем самым достигая комплекса целей или задач, для которых они были разработаны [3].

Для полноценного всестороннего изучения искусственных коллективов, строящихся по подобным принципам, важно наличие такой системы имитационного моделирования, которая предоставляет средства реализации основных элементов коллектива с социальной структурой, обеспечивает масштабируемость системы, возможность удобного ее расширения, поддержку сбора данных для последующего анализа. В обзоре [4] была показана необходимость и сформулированы требования к специализированной системе моделирования, пригодной для решения задач коллективной робототехники, ориентированных на создание социальных структур. Коротко остановимся на некоторых

характерных системах моделирования, а также их особенностях в связи с рассматриваемой проблемой.

Gazebo – это программа и набор библиотек, позволяющих моделировать поведение роботов с учетом разнообразных физических эффектов [5]. Когда предоставляемых системой моделируемых законов оказывается недостаточно, ее возможности могут быть расширены с помощью доступных плагинов или путем написания новых плагинов: добавление инфракрасных дальномеров, видеокамеры или физических эффектов движения через жидкость. Однако Gazebo плохо подходит для моделирования систем с большим количеством агентов из-за довольно высокой вычислительной стоимости.

AnyLogic [6] и Repast [7] являются мощными инструментами для моделирования, однако относятся к категории универсальных средств (AnyLogic больше подходит для моделирования макросистем, в то время как Repast – для моделирования многоагентных систем, состоящих из элементов с двигательной активностью), что делает их плохо приспособленными для исследования роботизированных систем с элементами социальной структуры, так как не предоставляют никаких специальных библиотек для моделирования систем в этой области.

В отличие от систем, поддерживающих детальное моделирование физических процессов, или универсальных систем агентного и имитационного моделирования существуют библиотеки моделирования, предоставляющие набор средств, более подходящих для построения социальных систем. Примерами являются NetLogo [8], Myrmedrome [9] и AntMe! [10]. NetLogo, несмотря на мощь самого языка, обладает весьма ограниченной функциональностью для моделирования роботизированных систем с элементами социальной структуры, а две другие программы не являются библиотеками моделирования, а, скорее, иллюстрируют и имитируют некоторые элементы поведения муравьев. Таким образом, возникла необходимость в создании специализированной системы моделирования. Прототип такой системы, получившей название Кворум, представлен в данной работе.

В статье описываются архитектура и некоторые экспериментальные результаты, демонстрирующие возможности моделирования различных задач, с которыми сталкиваются исследователи в этой области. Несмотря на то, что этот прототип сам по себе не предназначен для моделирования больших систем с использованием распределенных вычислений, его архитектура и особенности дают возможность протестировать некоторые концепции на базовом уровне, что позволяет в дальнейшем построить на нем специализированную имитационную систему, использующую параллельные вычисления.

Модели социального поведения

Согласно подходу к построению искусственных коллективов с элементами социальной структуры, агенты должны реализовывать ряд механизмов, необходимых для формирования и эффективного функционирования социума, например, стремление держаться группой, контагиозное и агонистическое поведение, образование коалиций [1].

Согласно работе [4], специализированная система моделирования также должна в той или иной мере поддерживать создание моделей внутреннего и внешнего мира, понятие пространства, времени, среды, в которой взаимодействуют агенты, предоставлять возможность модульной реализации отдельных аспектов агента, стигмергию как передачу информации через изменение внешней среды (например, с помощью феромонов), уметь поддерживать модель из нескольких взаимодействующих коллективов в силу распространенности задач деления и слияния групп роботов [11]. Использование стандартных поведенческих модулей позволяет сосредоточиться на содержательной части изучаемой проблемы.

Таким образом, для реализации и исследования искусственных коллективов агентов на основе элементов социальных сообществ необходима система моделирования со следующими особенностями [4]:

– масштабируемость, возможность использования параллельных вычислительных систем;

– поддержка определенных особенностей агентных систем в целом и их реализация в виде библиотек: понятие агента, взаимодействие агентов, окружающая среда, законы взаимодействия с окружающей средой, пространством, временем;

– основные структуры и механизмы социального поведения: индивидуальные психические различия, дифференциация функций, локальное взаимодействие индивидов и языковая коммуникация, формирование коалиций и иерархической структуры;

– поддержка создания моделей внутреннего и внешнего мира агента;

– поддержка моделей с несколькими взаимодействующими (но отличными) группами агентов.

Разработанная система Кворум реализует ряд вышеперечисленных свойств системы моделирования в дополнение к масштабируемости для распределенной вычислительной системы. Поскольку Кворум является экспериментальным прототипом, то библиотеки элементов высокого уровня, например, для механизмов формирования коалиции, разработчик модели должен реализовывать самостоятельно, исходя из ограничений конкретной задачи, вместо использования готового модуля, однако расширяемость системы позволяет дополнить ее необходимой функциональностью в дальнейшем и

предоставить такие базовые механизмы в составе ПО системы моделирования.

Архитектура системы моделирования Кворум

Система Кворум предназначена для моделирования поведения больших групп физических (робототехнических) агентов. Пользователь может выбрать несколько типов агентов и любое количество агентов в каждом типе. Каждый агент снабжен виртуальными датчиками, которые имитируют реальные датчики на роботах, такие как локаторы, суперлокаторы, датчики положения и т.д.

Кворум представляет собой программный инструмент, который имитирует роботов и их окружение. Данная система предназначена, в первую очередь, для мобильных роботов. Одной из ее особенностей является наличие интерфейсов, позволяющих управлять как виртуальными агентами, так и реальными техническими устройствами. Виртуальный агент – это агент без физического тела, например, используемый в компьютерном моделировании роботизированного агента. Конфигурация архитектуры для конкретной системы управления осуществляется в соответствующих пользовательских программных модулях. Например, конфигурация для архитектуры TMU определяется в модуле tmurobot.py (архитектура TMU реализует многопроцессорную расширяемую систему управления малых мобильных роботов).

В отличие от систем моделирования, позволяющих реализовывать различные физические эффекты, Кворум использует упрощенную физическую модель для экономии вычислительных ресурсов и ускорения разработки моделей агентов и среды. В частности, исходя из особенностей предметной области, 2D-модель мира была выбрана в качестве основы, в которой движение агентов и многие взаимодействия геометризированы. Например, работа ультразвуковых и инфракрасных радиолокаторов моделируется путем расчета лучей и оценки зоны видимости датчика. Такой подход позволяет моделировать большие группы агентов, сохраняя важные для области элементы и абстрагируя их физические детали.

Система моделирования работает под управлением ОС Linux и использует платформу ROS [12], которая передает данные между системными модулями. Модули, скрипты, подсистемы приложений могут быть написаны на любом поддерживаемом ROS языке. Основным требованием является реализация интерфейсов через систему сообщений – так называемые топики.

Система основана на двух модулях – *KVORUM_m* и *KVORUM_v*, которые составляют ее ядро, а также на программах прикладного уровня. Модули ядра написаны на языке Python. Поскольку одной из основных задач ядра системы является моделирование поведения группы реальных технических объектов, интерфейс модулей ядра определяется таким образом, чтобы максимально абстрагировать управляющую программу от объекта управления.

Абстракция управляющей программы от объекта управления (виртуального или реального робота, или группы роботов) осуществляется посредством взаимодействия компонентов ядра и пользовательского приложения, а именно путем обмена сообщениями (рис. 1).

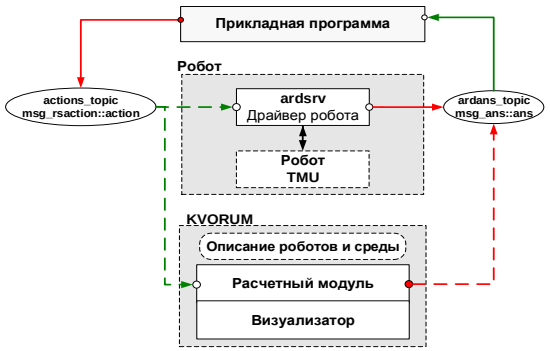


Рис. 1. Ядро системы

Fig. 1. The system core

Симуляция Кворум состоит из ядра (расчетный модуль, визуализатор, описание робота и окружающей среды), модели роботов и прикладных программ. Ядро системы взаимодействует с имитационной моделью робота и программы через топики ввод (*action_topic*) и вывод (*ardans_topic*). Модель робота состоит из узла управления и логической модели робота TMU (один из используемых в лаборатории типов робота, определяющий протокол связи с ним). Прикладные программы создаются пользователем в зависимости от имеющейся задачи.

На рисунке 2 представлен пример организации модулей системы моделирования Кворум для трех разных управляющих алгоритмов. Программа Demo считывает конфигурационную информацию из файлов описания карты, среды и агентов и реализует логику агента в единственном модуле demo.py. На примере алгоритма *SwHunt* показано, как алгоритм управления группой агентов может пользоваться сторонним описанием агентов, в том числе включающих специфичные для робота элементы. *tmusrv* и *temper* вместе позволяют построить более сложную логику агента с модулями, реализующими механизм эмоций. Аналогично идее, использованной в системе ARGoS [13], все эти модули управления пользуются одним и тем же интерфейсом *ardans_topic* и *actions_topic*, связывающим их с ядром системы моделирования (*kvorum_m*, *kvorum_v*) или же с драйвером реальных роботов *Ardsrv*.

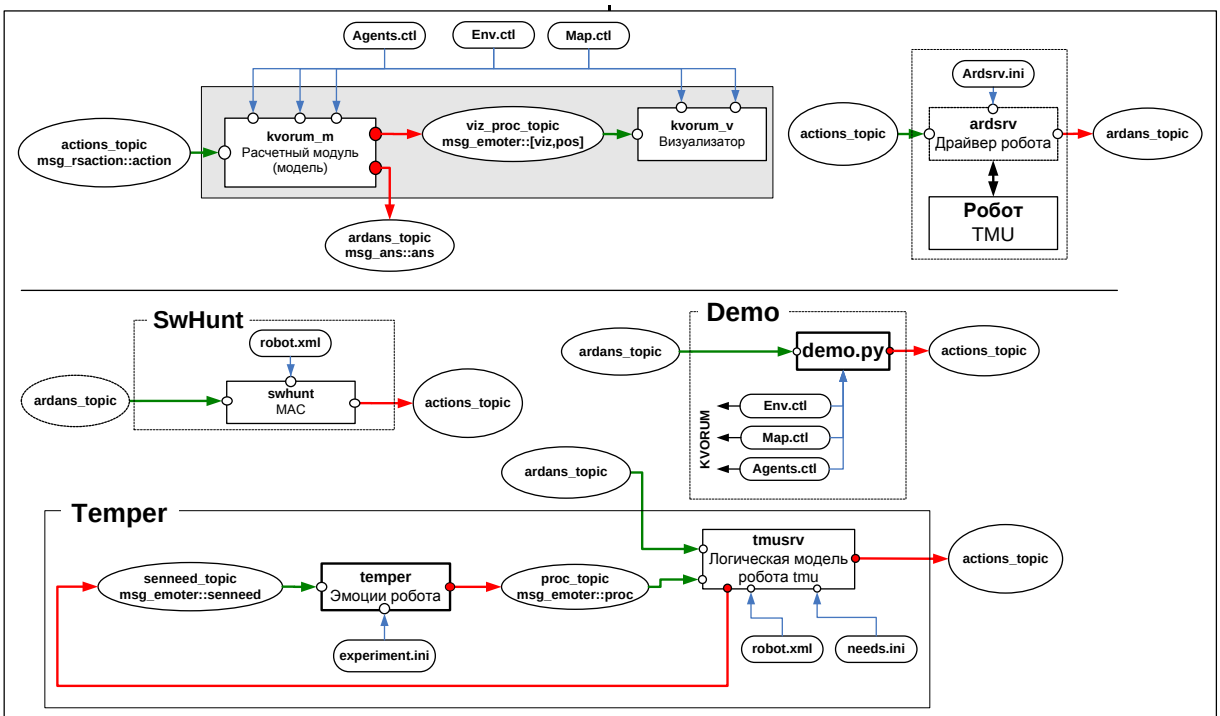
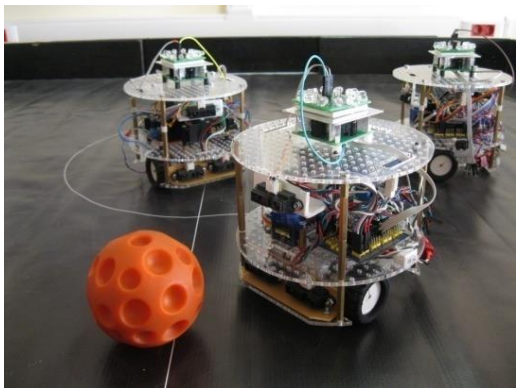


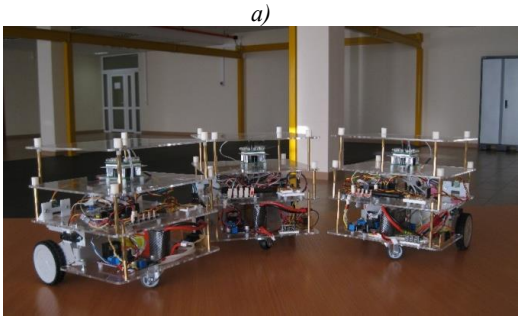
Рис. 2. Пример организации модулей системы Кворум для трех задач – SwHunt, Demo и Temper

Fig. 2. The example of organizing the Quorum system modules for three tasks: SwHunt, Demo and Temper

Такая архитектура имеет важное свойство переносимости кода элементов модели между симулятором и реальными роботами. В частности, роботы серий YARP-2 и YARP-3 (рис. 3), разработанные в лаборатории робототехники НИЦ «Курчатовский институт», имеют архитектуру TMU и поэтому работают по аналогичной схеме управления.



а)



б)

Рис. 3. Роботы серии YARP-1 (а) и YARP-2 (б)

Fig. 3. Robots of YARP-1 (a) and YARP-2 (б) series

Система Кворум основана на двухмерном представлении пространства. Это удобное упрощение, так как многие задачи групповой робототехники рассматриваются в контексте наземных мобильных роботов. Разделение этого пространства на отдельные элементы, например ячейки, позволяет дополнительно сократить количество необходимых вычислений, запрашивая и обрабатывая только те объекты, которые находятся в соответствующих ячейках. Подобное разбиение позволяет сократить объем информации, передаваемой между узлами системы распределенных вычислений, например, похожая схема используется в Repast HPC [14]. Использование только непрерывных координат привело бы к необходимости поиска среди большого количества объектов для сравнения их координат с данными ограничениями. В то же время, несмотря на деление на ячейки, непрерывные координаты остаются параметрами объектов и позволяют не ухудшать точность при моделировании смещения и оценке расстояний между объектами.

При работе с дискретной сеткой работа датчиков, таких как дальномеры, моделируется особым образом. Рассмотрим ультразвуковой дальномер. Для получения измерения от робота строится луч в том направлении, в котором осуществляет поиск дальномер. Ограничим длину луча некоторым максимальным значением. Клетки, которые он пере-

сек, отмечены с использованием алгоритма Брезенхэма. Выбираем все объекты, расположенные в этих ячейках, из памяти системы. Если они удовлетворяют условиям, проверяемым датчиком (в данном случае они должны отражать ультразвуковой сигнал), то возвращается наименьшее расстояние из вычисленных. Стоит отметить, что вместо луча можно построить конус и выделить все клетки, пересекаемые им. Таким образом, подобный подход дает некоторую гибкость в описании работы датчиков.

Среда, в которой живут агенты, представляет собой множество слоев, каждый из которых отвечает за представление определенного физического или логического знака. Это означает, что каждая точка пространства (x, y) описывается определенным вектором. Система представляет среду обитания в виде многомерного массива с размерами [x, y, *level*]. Например, интерпретация слоев (значения уровня) в зависимости от задачи может выглядеть так:

0 (*LEVEL_LIGHT*) – освещенность;

1 (*LEVEL_COLOR*) – цвет поверхности;

2 (*LEVEL_IR*) – эфир (инфракрасный сигнал);

3 (*LEVEL_GROUND*) – уровень поверхности;

4 (*LEVEL_ONBOARD*) – бортовой датчик (виртуальный уровень) и т.п.

Имена слоев определяются в словаре системы.

Такая организация данных позволяет разделить различные характеристики между различными областями памяти и выделить независимость псевдофизических характеристик среды. В дополнение к более структурированному представлению о мире подобное разделение имеет определенные технические преимущества: разные датчики, как правило, работают с различными характеристиками мира, и поэтому в разных слоях измерения и действия могут выполняться параллельно в отдельных потоках. Это также в значительной степени унифицирует операции, позволяя аналогичным функциям обрабатывать различные характеристики.

На рисунке 4 показано представление поля в виде слоев. Каждый датчик имеет свой слой, содержащий информацию о существующих агентах и объектах на нем. При создании датчиков можно изменять значения слоя. Оси координат представлены декартовой системой. Координаты точки всегда положительные, углы лежат в первом квартале

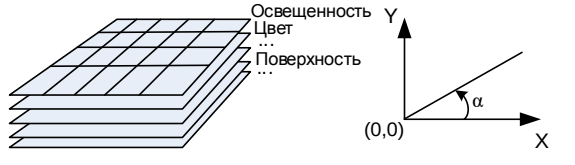


Рис. 4. Представление поля модели в виде слоев

Fig. 4. Representation of a model field as layers

и отсчитываются против часовой стрелки от горизонтали (оси X).

Основными типами датчиков являются точечный детектор, дальномер, локатор и суперлокатор.

Дальномер возвращает расстояние до объекта, найденного на слое *LEVEL_GROUND*.

Локатор – датчик, способный регистрировать окружающие объекты в определенном секторе, возвращая результаты как вектор значений. В зависимости от типа локатора вектор возвращаемых значений может содержать или расстояние до обнаруженных объектов на данном слое, или значения обнаруженных объектов.

Суперлокатор – локатор, возвращаемые значения которого представляют собой вектор из пар вида (значение сигнала, расстояние). Это позволяет определять не только расстояния до объектов, но и сами объекты. Как и для локатора, необходимо определить соответствие угла и индекса в массиве показаний.

Модуль расчета отвечает за пошаговое моделирование состояния системы и обеспечивает интерфейс между агентами и запросами пользователей: получение данных от датчика агента, настройку скорости движения агента вперед и назад, удаление, инициализацию агентов, рендеринг и инициализацию системы моделирования. Модуль визуализации представляет карту и агентов на экране.

Чтобы перенести систему управления на реальных роботов, необходимо выполнить дополнительную настройку и использовать специфичный для робота узел, обеспечивающий необходимый интерфейс. Такой узел называется *Ardsrv*. Он переводит унифицированные команды, передаваемые через *actions_topic*, в сигналы, воспринимаемые роботом. В случае YARP-2 и YARP-3 эти команды переводятся в соответствующий формат и передаются по беспроводной связи бортовым компьютерам, выполняющим их. Ответ, поступающий от роботов на узел *Ardsrv*, переводится обратно в унифицированный формат и передается в систему управления через *ardans_topic*. Таким образом, модули системы моделирования заменяются модулем драйвера робота, а топики коммуникации остаются неизменными вместе с остальной частью системы.

Описание конфигурации

Для работы пользовательских приложений и модулей ядра необходимо иметь описание виртуальной среды, поля и агентов. Эти описания представлены в виде файлов – программ, написанных на языке Python.

Описание среды (*Env.ctl*) вынесено в отдельный файл в силу специфики инициализации управляющих структур системы. Будучи центральной глобальной структурой как для пользовательской программы, так и для компонент ядра, конструктор

класса «Среда» *TEnv* требует отдельной предварительной инициализации.

Файл, описывающий среду, в которой находится робот, представляет собой фрагмент программы на языке Python, интерпретируемой при инициализации экземпляра класса *TEnv* («Среда») – глобальной переменной *Env*.

В файле конфигурации определяются размер рабочего окна визуализации (*MAX_SCR_X* и *MAX_SCR_Y*), размер виртуального поля (*DIM_X* и *DIM_Y*), топология поля (признак топологии тора *UseTorusRegime*).

Если значение параметра *UseTorusRegime* равно *True*, это означает «склейку» сторон поля, то есть образование поверхности типа тор. В противном случае среда будет представлять собой прямоугольную область заданного размера, ограниченную по периметру препятствиями. Скрипт выполняется во время инициализации экземпляра класса и является фрагментом конструктора.

Описание карты, на которой будут находиться роботы, представляет собой программу (скрипт). В этом скрипте осуществляется работа с глобальной переменной *Env*, обозначающей среду, в которой обитают агенты.

Она состоит из множества слоев, каждый из которых отвечает за представление того или иного физического или логического признака: каждая точка пространства (x, y) описывается некоторым вектором.

Система представляет среду обитания в виде трехмерного массива с измерениями [x, y, *level*]. Слои (*LEVEL_LIGHT*, *LEVEL_COLOR* и т.д.) были описаны выше.

Создание агента. Файл *Agents.ctl* содержит описание агентов. Процесс создания агентов включает в себя следующие шаги: создание экземпляра агента, определение параметров агента, регистрация агента в системе.

Необходимо создать экземпляр класса *TAgent* (описан в *pylib/agents.py*). Для этого вызывается соответствующий конструктор *TAgent(cid, cpos, cshape, cenv, csize)*.

Здесь *cid* – идентификатор агента (число); *cpos* – начальные координаты агента ([x, y, a]), где a – угол ориентации робота относительно оси абсцисс X; *cshape* – имя формы (*turtle*, *elefant*, …); *cenv* – указатель на среду (экземпляр класса *TEnv*), в которой работает агент (обычно это глобальная переменная *Env*); *csize* – размер фигуры (масштабный коэффициент относительно исходного размера изображения агента).

При этом углы в системе задаются в градусах. Нормализация значений происходит автоматически, поэтому пользователю нет необходимости приводить эти значения к тому или иному диапазону.

Система поддерживает несколько видов датчиков, различаемых по характеру возвращаемых значений (вектор или скаляр), а также типу сигнала и принципу действия.

Датчики представлены классом *TSensor* модуля *pylib/agent.py*.

Тип датчика:

– *gdic.ST_USONIC* – аналог ультразвукового дальномера, возвращает величину, пропорциональную расстоянию до объекта;

– *gdic.ST_SHARP* – аналог инфракрасного дальномера, возвращающего величину, равную разности между радиусом действия датчика и расстоянием до объекта;

– *gdic.ST_DETECTOR* – детектор; возвращает значение поля соответствующего слоя;

– *gdic.ST_CONST* – фиктивный датчик; возвращает постоянное значение, применяется для эмуляции считывания внутренних параметров робота (например, напряжения питания).

Тип возвращаемого значения:

– *gdic.RST_SCALAR* – скалярная величина;

– *gdic.RST_VECTOR* – вектор значений;

– *gdic.RST_SUPER_VECTOR* – вектор пар значений вида [*расстояние*, *значение_сигнала*].

На рисунке 5 представлена окружность, на которой изображено деление обзора суперлокатора на сектора. Для определения местоположения объекта в секторе берутся граничные значения сектора и далее осуществляется проход по этому сектору (по элементам массива) для нахождения детектированных локатором значений.

Детектирование множественных сигналов. Предположим, что агенту необходимо зарегистрировать и проанализировать множество источников сигналов. Например, принять и обработать сигналы от окружающих его роботов, то есть определить, какие источники находятся вокруг, их местоположение, тип генерируемого сигнала.

Очевидно, что для этого нецелесообразно использовать обычные датчики, которые регистрируют значение первого попавшегося сигнала.

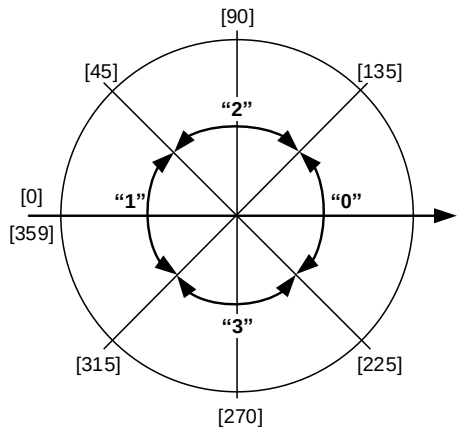


Рис. 5. Углы суперлокатора

Fig. 5. The superlocation corners

Для таких задач удобнее использовать локатор. При этом тип локатора – детектор (*gdic.ST_DETECTOR*). В этом случае система определит значение сигнала и его азимут (по индексу в векторе). Если же, помимо значения сигнала и азимута, нас интересует и расстояние до источника, то вместо локатора предпочтительнее использовать суперлокатор. Вектор возвращаемых суперлокатором значений, как уже говорилось выше, представляет собой пары вида (*значение_сигнала*, *расстояние*). Итак, когда нужно различать агентов, рекомендуется использовать суперлокатор, который зарегистрирует вокруг агента все что возможно.

Эксперименты и тестовые задачи

Опишем использование системы Кворум с группировкой реальных роботов и приведем примеры моделирования некоторых характерных задач.

Управление группой роботов. Интерфейс системы моделирования, позволяющий использование кода системы управления как в симуляции, так и на реальных роботах, был апробирован на группе из нескольких роботов YARP. Для передачи сигналов с управляющего устройства группе роботов использовалось устройство радиосвязи. В силу определенных особенностей канала связи пропускная способность была весьма ограниченной и количество пакетов, отправляемых роботам, необходимо было регулировать, а также равномерно распределять ресурсы между ними. Для этого был создан диспетчер сообщений, принимающий сообщения, предназначенные общему командному каналу, от всех роботов и организующий их в очередь, при этом управляя скоростью и очередностью пересылки сообщений для распределения пропускной способности в равных долях между всеми участниками (рис. 6).

Новые сообщения, получаемые по *callback*, помещаются в очередь ожидания, по одной очереди на каждого робота. Тип таких очередей *FIFO (first-in-first-out)*, они фиксированного размера. Старые сообщения выбрасываются из очереди, когда она переполняется. Для обработки сообщений, находящихся в очередях ожидания, в программе по таймеру вызывается обработчик, каждый вызов которого обрабатывает и посылает не более одного сообщения из очередей. Частота вызова этого обработчика обратна пропускной способности канала, что гарантирует соответствие пропускной способности выходного канала.

Функция обработки и отправки сообщения по мере вызовов последовательно проходит по всем очередям, посылая из них по одному сообщению: $R_1 - R_2 - \ldots - R_N - R_1$. Здесь R_i – сообщение i-го робота. При появлении сообщений от нового робота динамически создается другая очередь, и порядок отсылки сообщений может временно нарушиться.

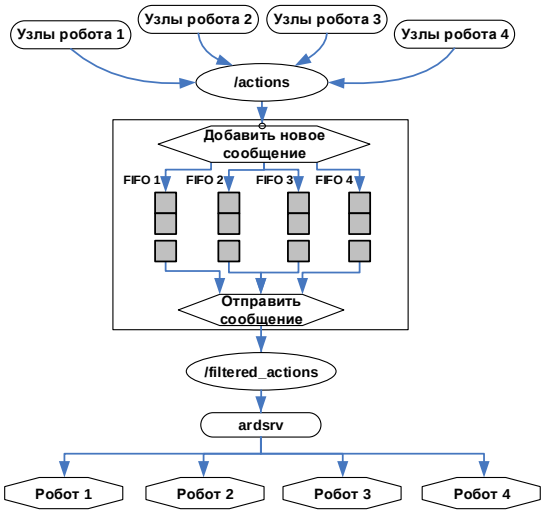


Рис. 6. Диспетчер сообщений

Fig. 6. The Message Manager

Если в просматриваемой очереди не осталось сообщений, то она пропускается и проверяется следующая очередь. Система предназначена для работы на основе топиков ROS [12] (*/actions, /filtered_actions*).

Эксперименты показали работоспособность предложенной архитектуры системы: реальные роботы управлялись той же программой, что и их виртуальные модели в симуляции без модификаций управляющего кода.

Коммуникация. Один из важных элементов построения групповых роботизированных систем – это языковое взаимодействие роботов [1]. Важным элементом являются символические структуры, на основе которых также можно строить системы управления, реализующие некоторые базовые механизмы систем с социальной структурой (например, подражательное или контагиозное поведение), что иллюстрируется работой [15]. Особенностью задачи является моделирование связи между агентами, в частности, с использованием элементов языковой коммуникации.

В Кворум связи между агентами определены явно следующим образом: агенты снабжены как источниками сигналов, так и соответствующими приемниками. Связь осуществляется путем интерпретации генерируемых сигналов. Коммуникационные возможности применялись в работе [16], где группа агентов использовала их для координации усилий в задаче стайной охоты (см. http://www.swsys.ru/uploaded/image/2018-3/2018-3-dop/18.jpg).

В рассмотренной задаче выбирается лидер группы, а затем группа разделяется на роли, определяющие, с какой стороны агенты-охотники будут атаковать жертву.

Следование пути по визуальным ориентирам. В работе [17] предложен метод построения пути вдоль визуальных ориентиров мобильным роботом. Эта задача относится к категории моделирова-

ния внутреннего представления внешнего мира в агенте.

Описание маршрута в указанной работе основано на пространственных связях, а результаты подтверждены проведением имитационных экспериментов с использованием системы моделирования Кворум. Робот-разведчик использует алгоритм случайного блуждания, чтобы найти объект заданного цвета на полигоне $n \times m$ ячеек, по пути формируя описание маршрута. После достижения цели описание переносится на второго робота, который повторяет маршрут, следуя полученным ориентирам (рис. 7).

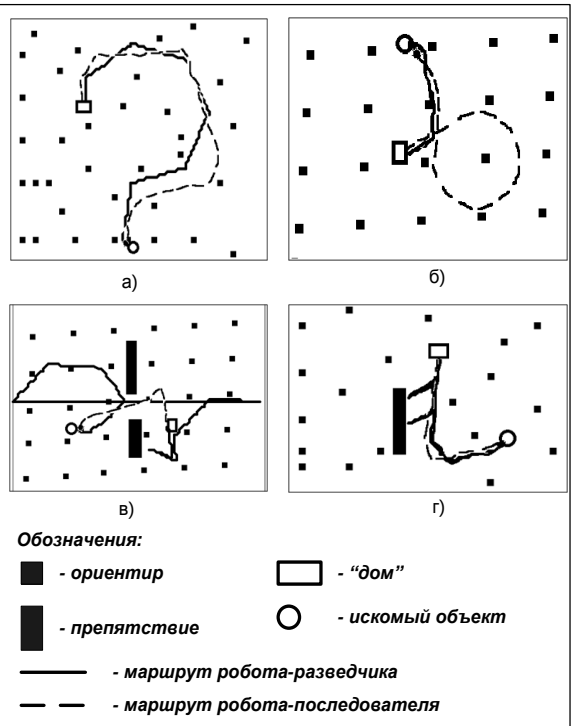


Рис. 7. Следование робота вдоль маршрута, обнаруженного другим роботом

Fig. 7. The robot going along the route detected by another robot

Поддержание энергетической автономности группы роботов. Одной из проблем коллективной робототехники является энергетическая независимость команды. В то же время можно рассматривать как задачу выполнения целевой функции одновременно с поддержанием необходимого энергетического уровня агентов, так и задачу сбора энергии из различных источников. Пример постановки задачи приведен в работе [18], где сформулирована многокритериальная оценка целей робота. Особенностью задачи с точки зрения системы моделирования является динамическая природа взаимодействий агентов и мира: сбор различных ресурсов, изменение параметров робота (энергетического уровня), генерация энергии в центральном гнезде. В этой модели агенты взаимодействуют через стигмергию в виде направленных феромонов [19]. Индивидуальное параметрическое регулирование поведения используется, чтобы управлять деятельностью группы. Оно реализуется путем назначения двух ролей и переключения между ними. Каждая роль соответствует различным характеристикам, в частности, способности переносить различное количество пищи и времени, по истечении которого агент возвращается в гнездо, если пища не была собрана. Механизм, используемый в этой работе, является по существу стохастическим автоматом с двумя состояниями φ_1 и φ_2 с соответствующими стратегиями переключения ролей, которые определяются с помощью специальных матриц для возвращения с пищей, и для возврата без еды, где используются параметры вероятности переключения роли разведчика на роль собирателя и наоборот.

В работе [18] рассматривается проблема поддержания достаточного уровня энергии для функционирования группы с необходимостью использования нескольких ресурсов и централизованного генератора энергии. На тестовой карте размещено несколько объектов: источники ресурсов и базовая станция. Базовая станция определяет потребности в ресурсах и сообщает о них агентам, которые находятся поблизости. Агенты активируются и начинают искать ресурсы, извлекать их и доставлять до базы. Агент может собирать только один ресурс за один цикл работы. Каждый агент имеет ограниченный запас энергии, который он использует во время поиска. Если запас иссякает до того, как агент достигнет базы, он умирает (деактивируется). База играет роль зарядной станции коллектива, вырабатывающей энергию из ресурсов, добытых агентами (рис. 8). Используется модель принятия решения на основе оптимизации по Парето. Агенты оценивают обнаруженные источники пищи по следующим факторам: сколько времени проводился поиск; насколько острой была потребность в этом типе ресурса в гнезде; каким был его целевой ресурс.

Влияние факторов определяется коэффициентами в формуле. В общем виде целевая функция строится следующим образом: $f_e = c_t f_t(t) + c_n f_n(n, s) + c_g f_g(s, g)$, где c_t, c_n и c_g являются коэффициентами влияния соответствующих факторов, при этом $c_t + c_n + c_g = 1$.

На рисунке 8 представлены 4 различных типа ресурсов, обозначенных цветными квадратами. Квадрат в центре, выделяющийся среди других по цвету, является базой, где агенты хранят собранные ресурсы. Маленькие черепахи обозначают роботов.

Модель системы была протестирована с помощью Кворум. Вычислительный эксперимент позволил оценить динамику количества роботов в команде по различным параметрам системы управления (по «характерам» агентов).

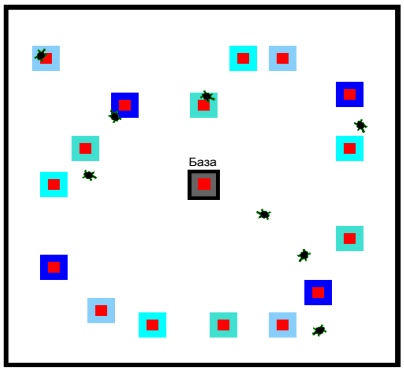


Рис. 8. Гомогенная группа агентов, собирающая различные ресурсы для базы

Fig. 8. A homogeneous group of agents that collects various resources for a database

Заключение

Разработан и апробирован прототип системы моделирования Кворум на некоторых репрезентативных задачах групповой робототехники, подтвердивший жизнеспособность предложенных концепций и моделей, которые станут основой специализированной системы моделирования коллективной робототехники с использованием параллельных вычислений. Еще одним важным свойством предлагаемой и реализованной архитектуры является переносимость кода элементов модели между симулятором и реальными роботами. Это позволяет развернуть тот же управляющий код, который был протестирован в моделировании, в реальных системах, откуда следует более четкая интерпретация результатов, полученных с помощью такой системы моделирования. Эта возможность была проверена на реальных роботах и прототипе симулятора.

Разработанный прототип удовлетворяет следующим функциональным требованиям:

– в архитектуру заложены масштабируемость, возможность моделирования с применением вычислительных кластеров;

– поддержка определенного функционала, свойственного агентным системам, и его предоставление в виде библиотек: агент, взаимодействие между агентами, среда, законы взаимодействия со средой, пространство, время;

– реализация некоторых базовых механизмов социального поведения и структур и поддержка дополнения и расширения соответствующих механизмов (индивидуальные психические различия, дифференциация функций и т.п.);

– поддержка модели из нескольких взаимодействующих коллективов.

Следующий шаг в развитии системы – полная поддержка параллельных вычислительных систем и тестирование на кластере компьютеров. Ожидается, что потенциальное увеличение скорости вычислений позволит моделировать большие группы (десятки тысяч) локально взаимодействующих агентов в режиме реального времени, сохраняя при этом адекватное представление моделируемого мира.

Работа выполнена при частичной финансовой поддержке грантов РНФ № 16-11-00018, РФФИ № 16-29-04412 офи_м.

Литература

1. Карпов В.Э. Модели социального поведения в групповой робототехнике // Управление большими системами. 2016. № 59. С. 165–232.
2. Russell S.J. and Norvig P. Artificial intelligence: a modern approach, 3rd ed., Prentice Hall, 2010, 1152 p.
3. Maes P. Artificial life meets entertainment: lifelike autonomous agents. Commun. ACM, 1995, vol. 38, no. 11, pp. 108–114.
4. Ровбо М.А., Овсянникова Е.Е., Чумаченко А.А. Обзор средств имитационного моделирования коллективов роботов с элементами социальной организации // Программные продукты и системы. 2017. Т. 30. № 3. С. 425–434.
5. Koenig N. and Howard A. Design and use paradigms for gazebo, an open-source multi-robot simulator, Proc. 2004 IEEE/RSJ Int. Conf. Intell. Robot. Syst. (IEEE Cat. No. 04CH37566), 2004, vol. 3, pp. 2149–2154.
6. Borshchev A. The Big book of simulation modeling: multimethod modeling with AnyLogic 6. Amazon Digital Services LLC, 2015, 614 p.
7. Argonne National Laboratory: Repast Simphony, 2016. URL: http://repast.sourceforge.net/repast_simphony.php# (дата обращения: 10.01.2018).
8. Wilensky U. NetLogo ants model. Center for Connected Learning and Computer-Based Modeling, 1997. URL: http://ccl.northwestern.edu/netlogo/models/Ants (дата обращения: 10.01.2018).
9. Cacace S., Cristiani E. and D'Eustacchio D. Myrmedrome: simulating the life of an ant colony. In Imagine Math 2: between culture and mathematics, M. Emmer (Ed.), Springer Milan, 2013, pp. 201–210.
10. Wendel T. AntMe. 2014. URL: https://service.antme.net/ (дата обращения: 10.01.2018).
11. Воробьев В.В. Алгоритм кластеризации коллектива роботов // Беспилотные транспортные средства с элементами искусственного интеллекта (БТС-ИИ-2016): сб. тр. III Всерос. науч.-практич. семинара. М.: Перо, 2016. С. 50–59.
12. Koubaa A. Robot operating system (ROS), Springer, Cham, 2017, vol. 707, no. 2, 665 p.
13. Pinciroli C., Trianni V., O'Grady R., Pini G., Brutschy A., Brambilla M., Mathews N., Ferrante E., Di Caro G.A., Ducatelle F., Stirling T., Gutiérrez Á., Gambardella L.M., and Dorigo M. ARGoS: a modular, multi-engine simulator for heterogeneous swarm robotics. Proc. IEEE Int. Conf. Intell. Robot. Syst., 2011, pp. 5027–5034.
14. Collier N. and North M. Parallel agent-based simulation with Repast for High Performance Computing. Simulation, 2013, 22 p.
15. Карпов В.Э. Сенсорная модель подражательного поведения роботов // OSTIS-2016: матер. VI Междунар. науч.-технич. конф. Минск: Изд-во БГУИР, 2016. С. 471–476.
16. Karpov V. and Karpova I. Leader election algorithms for static swarms. Biol. Inspired Cogn. Archit., 2015, vol. 12, pp. 54–64.
17. Карпова И.П. К вопросу о представлении маршрута мобильного робота на основе визуальных ориентиров // Мехатроника. Автоматизация. Управление. 2017. Т. 18. № 2. С. 81–89.
18. Ровбо М.А., Малышев А.А. Комплекс для изучения энергетически автономных коллективов роботов // Открытое образование. 2017. № 2. С. 68–77.
19. Boissard E., Degond P., and Motsch S. Trail formation based on directed pheromone deposition, J. Math. Biol., 2013, 35 p., vol. 66, iss. 6, pp. 1267–1301. URL: https://link.springer.com/article/10.1007%2Fs00285-012-0529-6 (дата обращения: 10.01.2018).

Software & Systems
DOI: 10.15827/0236-235X.123.581-590

Received 29.01.18
2018, vol. 31, no. 3, pp. 581–590

Kvorum – the system for modeling the behavior of robotic agent groups with elements of social organization

V.E. Karpov [1, 2], *PhD., associate professor, Head of the Department of Neurocognitive Sciences and Intelligence Systems, karpov.ve@gmail.com*
M.A. Rovbo [1], *Research-Engineer, rovboma@gmail.com*
E.E. Ovsyannikova [1], *Research-Engineer, eeovsyan@yandex.ru*

[1] *National Research Center «Kurchatov Institute», Moscow, 123182, Russian Federation*
[2] *Moscow Institute of Physics and Technology (State University), Moscow Region, Dolgoprudny, 141701, Russian Federation*

Abstract. The paper describes the Kvorum architecture, which is a multi-agent modeling environment designed to develop constructive solutions and algorithms of robot agents' individual behavior, as well as for modeling behavior of large groups of robots. The main objective of the system is to provide tools for research in applying social organization methods in group robotics.

Kvorum is a prototype simulation and agent modeling system that abstracts the complexity of a physical system and provides convenient interfaces and library modules for modeling groups of mobile robots and internal structures of individual agents. It was built as a modular, easily extensible system focused on modeling systems with a large number of agents by simplifying physical and other effects, and also by the ability to perform simulations using parallel computations.

It is shown that the proposed method for modeling such systems is applicable to a wide range of problems from the field of group robotics with social structure elements (as well as for common robotics and group robotics problems in general). Some features of architecture and models in Kvorum allow it to be expanded in subsequent iterations to fully support simulation on parallel computing systems.

One of the important Kvorum features is the focus on controlling real robotic systems. For this purpose the system provides the same control control interfaces of both computational models and real robots.

The system was tested on some specific problems of group robotics, in particular those that use social structure mechanisms, as well as on a group of laboratory robots of different types that support the simulation system interface.

Keywords: agent modeling, social behavior modeling, group robotics, simulation tools, development tools, cluster computing, bioinspired systems, eusociality, robot, artificial life.

Acknowledgements. *This work has been partially supported by RNF grants no. 16-11-00018, RFBR no. 16-29-04412 ofi_m.*

References

1. Karpov V.E. Models of social behavior in group robotics. *Managing Big Systems*. 2016, no. 59, pp. 165–232 (in Russ.).
2. Russell S.J., Norvig P. *Artificial Intelligence: A Modern Approach*. 3rd ed. Prentice Hall Publ., 2010, 1152 p.
3. Maes P. Artificial life meets entertainment: lifelike autonomous agents. *Commun. ACM*. 1995, vol. 38, no. 11, pp. 108–114.
4. Rovbo M.A., Ovsyannikova E.E., Chumachenko A.A. Review of simulation modeling tools for obot groups with social organization elements. *Software & Systems*. 2017, no. 3, pp. 425–434 (in Russ.).
5. Koenig N., Howard A. Design and use paradigms for gazebo, an open-source multi-robot simulator. *2004 IEEE/RSJ Int. Conf. Intell. Robot. Syst. (IEEE Cat. no. 04CH37566)*. 2004, vol. 3, pp. 2149–2154.
6. Borshchev A. *The Big Book of Simulation Modeling: Multimethod Modeling with AnyLogic 6*. Amazon Digital Services LLC, 2015, 614 p.
7. *Repast Simphony*. Argonne National Laboratory. 2016. Available at: http://repast.sourceforge.net/repast_simphony.php# (accessed January 10, 2018).
8. Wilensky U. *NetLogo Ants Model*. Center for Connected Learning and Computer-Based Modeling, Northwestern Univ., 1997. Available at: http://ccl.northwestern.edu/netlogo/models/Ants (accessed January 10, 2018).
9. Cacace S., Cristiani E., D'Eustacchio D. Myrmedrome: Simulating the life of an ant colony. *Imagine Math 2: Between Culture and Mathematics*. M. Emmer (Ed.). Milano, Springer Milan Publ., 2013, pp. 201–210.
10. Wendel T. *AntMe*. 2014. Available at: https://service.antme.net/ (accessed January 10, 2018).
11. Vorobev V.V. Algorithm for robot team clustering. *Proc. 3rd All-Russ. Sci.-Pract. Workshop "Unmanned Vehicles with the Elements of Artificial Intelligence" (BTS-II-2016)*. Moscow, Pero Publ., 2016, pp. 50–59 (in Russ.).
12. Koubaa A. *Robot Operating System (ROS)*. Springer Publ., Cham, 2017, vol. 707, no. 2, 655 p.
13. Pinciroli C., Trianni V., O'Grady R., Pini G., Brutschy A., Brambilla M., Mathews N., Ferrante E., Di Caro G.A., Ducatelle F., Stirling T., Gutiérrez Á., Gambardella L.M., Dorigo M. ARGoS: A modular, multi-engine simulator for heterogeneous swarm robotics. *IEEE Int. Conf. Intell. Robot. Syst*. 2011, pp. 5027–5034.
14. Collier N., North M. Parallel agent-based simulation with repast for high performance computing. *Simulation*. 2013, no. 89, pp. 1215–1235.
15. Karpov V.E. Sensory model of robots imitative behavior. *Proc. 6th Intern. Sci. and Tech. Conf. "Open Semantic Technologies for Intelligent Systems" (OSTIS-2016)*. Minsk, BGUIR, 2016, pp. 471–476 (in Russ.).
16. Karpov V., Karpova I. Leader election algorithms for static swarms. *Biol. Inspired Cogn. Archit*. 2015, vol. 12, pp. 54–64.
17. Karpova I.P. About the problem of representing the route of a mobile robot based on visual landmarks. *Mechatronics. Automation. Control*. 2017, vol. 18, no. 2, pp. 81–89 (in Russ.).
18. Rovbo M.A., Malyshev A.A. Complex for studying energetically autonomous collectives of robots. *Open Education*. 2017, no. 2, pp. 68–77 (in Russ.).
19. Boissard E., Degond P., Motsch S. Trail formation based on directed pheromone deposition. *J. Math. Biol*. 2013, vol. 66, iss. 6, pp. 1267–1301. Available at: https://link.springer.com/article/10.1007%2Fs00285-012-0529-6 (accessed January 10, 2018).