УДК 004.4
DOI 10.53403/9785951505071_2022_206

ПРОБЛЕМЫ СОЗДАНИЯ ИНСТРУМЕНТАЛЬНЫХ СРЕДСТВ ПРОГРАММИРОВАНИЯ ЛОГИКИ ПОВЕДЕНИЯ АГЕНТОВ В МУЛЬТИАГЕНТНЫХ СИСТЕМАХ ИМИТАЦИОННОГО МОДЕЛИРОВАНИЯ ДВУХСТОРОННИХ БОЕВЫХ ДЕЙСТВИЙ

К. В. Иванов, М. В. Галкин, А. И. Сайфуллин, Р. Н. Сайфуллина, Д. В. Девятых

Российский федеральный ядерный центр –
Всероссийский НИИ экспериментальной физики, Саров

В докладе рассмотрены ключевые аспекты создания инструментальных средств программирования логики агентов в мультиагентных системах и озвучены проблемы, возникающие при разработке. Выполнен краткий обзор существующих инструментальных средств программирования логики агентов, в котором отмечены как зарубежные, так и отечественные программные продукты.

Основной акцент доклада – это имитационное моделирование двухсторонних боевых действий, ориентированное на специалистов предметной области. Для этого выделены три типа агентов: «тактическая единица», «тактическая группа» и «объединение». Показана структура их связи и взаимодействия, сформулированы особенности и специфика для каждого типа агентов. Для каждого типа агентов предложено решение, определяющее облик наиболее подходящего инструментального средства для программирования логики их поведения, который был бы удобен для применения специалисту предметной области.

Ключевые слова: имитационное моделирование, агент, мультиагентная система, программирование, логика поведения, система управления.

Введение

В современной литературе встречается множество определений понятия «мультиагентная система». Однако, в контексте затрагиваемых данным докладом проблем, целесообразно обратиться к определению Дж. Люгера, который под мультиагентной системой (МАС) понимал вычислительную программу, решатели которой расположены в некоторой среде и каждый из них способен к гибким, автономным и социально-организованным действиям в направлении предопределенной цели [1]. В данном докладе рассматривается подкласс мультиагентных систем – мультиагентные системы имитационного моделирования (МСИМ) двухсторонних боевых действий (ДБД).

Термин «агент» происходит от латинского глагола «agere», что означает «действовать», «двигать», «править», «управлять». В энциклопедическом словаре Ф. А. Брокгауза и И. А. Ефрона дано следующее определение: «агент – деятель, лицо, действующее по поручению или полномочию другого» [2]. Это определение правильно выражает суть агента МСИМ, который может функционировать автономно от имени своего владельца (человека-оператора) и решать разнообразные задачи по обработке информации. Для успешной работы агент должен обладать достаточными интеллектуальными способностями, должен иметь возможности взаимодействия с владельцем для получения заданий и передачи результатов, должен ориентироваться в среде своего существования и принимать необходимые решения.

Две базовые характеристики – автономность и целенаправленность – позволяют отличать агента от других программных и аппаратных объектов (модулей, подпрограмм, процедур и т. п.). Авто-

номность агента означает его способность взаимодействовать со средой без прямого участия других агентов, для чего он должен уметь контролировать свое внутреннее состояние и выполняемые действия. Целенаправленность означает, что соответствующий агенту программный код выполняется не автоматически, а по решению самого агента в зависимости от ситуации и согласно поставленным перед агентом целям.

Наличие целесообразности поведения требует, чтобы агент обладал свойством реактивности. Такой уровень интеллекта соответствует рефлекторному поведению животного. Если же агент обладает знаниями о среде, собственных целях и способах их достижения, то такой агент может быть назван разумным (когнитивным). Таким образом, может быть проведена граница между интеллектуальными и неинтеллектуальными агентами.

Еще одна характеристика агента – коммуникабельность, предполагает необходимость решения своих задач совместно с другими агентами и наличие развитых протоколов коммуникации.

Обзор существующих инструментальных средств программирования логики агентов

Перечень систем, ориентированных на решение задач, связанных с программированием логики агентов, довольно обширен. Приведем наиболее известные проекты: AgentBuilder [3], Ascape [4], Bee-gent [5], Cable [6], Decaf [7], FIPA-OS [8], Grasshopper [9], Gypsy [10], JADE [11], JASON [12], JATLite [13], JAFMAS [14], MAML [15], ProcessLink [16], Swarm [17], Zeus [18].

Среди отечественных разработок большую известность получили инструментальные средства группы интеллектуальных систем Санкт-Петербургского института информатики и автоматизации РАН, возглавляемой В. И. Городецким [19]. Концептуальной основой развиваемой коллективом В. И. Городецкого технологии разработки МАС является выделение «общности» реализуемых функций и представление их в виде классов и структур данных, являющихся составной частью программного инструментария. Реализуемая в этом подходе идея состоит в выделении наибольшего количества классов и структур данных, повторно используемых в агентных приложениях. В системе MAS DK создана конструкция «Типовой агент» («Generic Agent»), являющаяся базой для последующей специализации классов и структур данных, клонирования экземпляров агентов и индивидуальной доработки специфических компонентов. Этот процесс поддерживается соответствующим набором редакторов.

Отметим, что подавляющее большинство из приведенных выше систем ориентированы на разработку МАС “с нуля” и, по сути, представляют собой платформы для создания мультиагентной системы разработчиком-программистом. Нашей же целью является определение принципов и подходов к созданию инструментария, позволяющего не разработчику, а конечному пользователю МСИМ БДБ производить наполнение системы агентами, обладающими требуемой логикой поведения. При этом пользователь не должен отвлекаться на решение сопутствующих, общих проблем, связанных с созданием МАС (разработка пользовательских интерфейсов, средств коммуникаций агентов и т. п.), и может не являться программистом высокой квалификации. Исходя из этого, указанные системы целесообразно рассматривать с точки зрения языков и семантик, применяемых для программирования логики агентов.

Так, например, в системе JASON реализована операционная семантика, формально определенная для языка AgentSpeak(L). Данный язык определяет агента через спецификацию множества базовых убеждений и множества планов. Атомарное убеждение является обычным предикатом первого порядка. Атомарные убеждения или их отрицания обозначаются соответствующими литералами. Начальное множество убеждений является набором базовых атомарных убеждений. В языке AgentSpeak(L) определяются два типа целей: достижимые цели и тестовые цели. Достижимая цель устанавливается тогда, когда агент стремится достичь определенного состояния мира, в котором связанный предикат является истинным. Практически для этого инициируется выполнение вложенных планов. Тестовая цель возвращает унификацию для связанного предиката с одним из убеждений агента, в случае неудачи такая цель отвергается. Переключающее событие может инициировать выполнение некоторого плана. Событие может быть внутренним, когда достигнута определенная

подцель, или внешним, порождаемым в результате модификации убеждений после восприятия состояния внешней среды. Планы используют базовые действия, которые агент способен выполнить в среде. Действия также определяются как предикаты первого порядка, но обозначаются специальными символами. План образуется переключающим событием, за которым следует конъюнкция литералов, образующих контекст. Контекст должен быть последовательностью таких текущих убеждений агента, для которых данный план применим. Остаточная часть плана есть последовательность базовых действий и целей, которые агент достигает (или проверяет), когда план выбран для исполнения.

Тело плана – это последовательность операций, реализующих предназначение плана:

– математических операций;

– вызовов внутренних функций JASON;

– действий, направленных на объекты среды функционирования МАС;

– коммуникационных актов.

Обобщенная синтаксическая конструкция плана выглядит следующим образом:

«Метка

Имя события : Контекстные ограничения <- Тело плана.»

Программы агентов интерпретируются циклически. В каждом цикле обновляется список событий, порождаемых либо перцепцией из внешней среды, либо выполнением подцелей, определенных в теле плана. Далее интерпретатор унифицирует выбранное событие с переключающими событиями в заголовках планов, определяя, таким образом, множество релевантных планов. Проверяя содержимое контекста этих планов, интерпретатор определяет множество применимых планов и выбирает один применимый план из этого множества. Когда все формулы в теле плана оказываются выполненными, то цель считается достигнутой и план также удаляется из очереди планов. Цикл работы интерпретатора повторяется [20].

Ниже приведен пример программы на языке AgentSpeak(L) в которой вычисляется факториал числа 5:

```
fact(0,1).
+fact(X,Y): X < 5
<- +fact(X+1, (X+1)*Y).
+fact(X,Y): X == 5
<- .print("fact 5 == ", Y).
```

Программа содержит два разных плана, которые активируются при возникновении одного и того же убеждения fact(X,Y), но в разных контекстных условиях - «X < 5» и «X == 5». Благодаря наличию начального убеждения «fact(0,1).» и первому плану, у агента будут последовательно формироваться следующие убеждения:

```
fact(1,1).
fact(2,2).
fact(3,6).
fact(4,24).
fact(5,120).
```

При возникновении последнего убеждения первый план уже не активируется, зато активируется второй план, который выведет результат вычисления факториала.

Типы агентов в мультиагентных системах имитационного моделирования двухсторонних боевых действий

Рассмотренный выше язык AgentSpeak(L), безусловно, является мощным средством программирования логики агентов. Однако, в отличие от языков программирования «общего назначения» (C++, Java, Python и т. д.), язык AgentSpeak(L) имеет высокую степень специализации применительно к рассматриваемой предметной области, достаточно универсален в рамках этой предметной области, и, как следствие, довольно сложен в практическом применении.

Для того чтобы определить требования к инструментарию программирования логики агентов МСИМ БДБ, который был бы удобен для применения специалисту данной предметной области, попытаемся систематизировать типы применяемых агентов и определить характерные решаемые ими задачи. Рассмотрим наиболее общий случай, когда необходимо моделировать действия каждой отдельно взятой тактической единицы в связке с многоуровневой системой управления (СУ) силами. В качестве тактической единицы будем рассматривать элементарную, с точки зрения моделируемой системы управления, сущность. Такими сущностями могут являться корабль, самолет, рота (взвод, отделение – в зависимости от того, какая требуется степень детализации) солдат. Тактической группой будем называть несколько однотипных тактических единиц, объединенных под единым командованием для решения общей тактической задачи. Тактические группы и тактические единицы могут входить в состав объединений, которые представляют собой формирования, создаваемые для решения задач оперативно-тактического, оперативного и оперативно-стратегического уровней. Одни объединения могут входить в состав других.

Таким образом, можно выделить три типа агентов, между которыми выстроена иерархия с точки зрения системы управления (см. рис. 1): тактические единицы, тактические группы, объединения.

Рис. 1. Иерархия типов агентов МСИМ БДБ

Охарактеризуем задачи, которые решают агенты указанных типов в процессе моделирования.

Тактическая единица, действуя самостоятельно или в составе тактической группы, решает тактические задачи, поставленные перед ней вышестоящим командованием. В роли командира могут выступать либо агенты типа «Тактическая группа» и «Объединение», либо человек, который формирует команды, задавая исходные данные моделирования (сценарии). Человек также может выступать в роли командира тактической единицы и отдавать приказы на выполнение различных тактических задач, если применяется автоматизированный режим моделирования, который допускает интерактивное вмешательство человека в процесс моделирования. Таким образом, тактическая единица должна уметь решать устанавливаемые ей тактические задачи, действуя автономно (принимая самостоятельно решения в установленных рамках, планируя и осуществляя действия, меняющие окружающую среду), целенаправленно и коммуникабельно (см. «Введение»). Поскольку автономность подразумевает способность агента взаимодействовать со средой, тактическая единица должна обладать сенсорами, с помощью которых она сможет получать информацию о состоянии среды. Тактическая единица должна обладать способностью восприятия и анализа информации, поставляемой сенсорами, в процессе выполнения тактической задачи. Целенаправленность подразумевает, что информация, полученная агентом от сенсоров или других тактических единиц, используется для анализа текущей ситуации и выбора способа действия для достижения цели, заданной в рамках выполняемой тактической задачи. Коммуникабельность подразумевает способность агента решать свои задачи совместно с другими агентами, например, получая приказы от агента типа «тактическая группа» или «объединение».

Тактическая группа представляет собой объединение нескольких тактических единиц. Исходя из того, что в нашем случае каждая тактическая единица представлена отдельным агентом, т. е. сущностью, имеющей собственную логику поведения, тактическую группу целесообразно рассматривать в виде агента, формирующего и уточняющего индивидуальные задачи для тактических единиц. При этом агент действует в соответствии с приказами, получаемыми от вышестоящих в иерархии системы управления агентов, а также использует информацию о среде, получаемую от собственных сенсоров, сенсоров подчиненных тактических единиц, либо поставляемую другими агентами.

Объединения представляют собой формирования, состоящие из тактических единиц, тактических групп и других объединений. Поэтому, как и в случае с тактическими группами, их целесообразно представлять в виде агентов, формирующих и уточняющих задачи для подчиненных агентов. Логика поведения агента «объединение» в некотором смысле дублирует логику поведения агента «тактическая группа» – и в том и в другом случае производится анализ информации, поступающей от подчиненных тактических единиц, средств и системы управления, и на основе этого анализа формируются задачи для подчиненных объектов. Различаются лишь масштабы собираемых данных и подчиненных сил.

Таким образом, мы видим, что агенты всех выделенных нами типов призваны решать задачи, поставляемые вышестоящими в иерархии системы управления агентами. В процессе решения этих задач агенты типа «объединение» и «тактическая группа» осуществляют управление подчиненными агентами, а агенты типа «тактическая единица» управляют собственным состоянием, задавая параметры своего передвижения в пространстве, а также параметры приданных им средств обнаружения, поражения и связи.

Программирование логики агентов типа «тактическая единица»

Поскольку назначение агента «тактическая единица» состоит в решении тактических задач, поставленных перед ним вышестоящим командованием, будем считать, что логика поведения агента определена в том случае, если заданы алгоритмы решения всех возможных (в рамках конкретного модельного эксперимента) тактических задач. Как было отмечено выше, алгоритм решения тактической задачи для агента данного типа должен определять порядок перемещения тактической единицы в пространстве и управления ее системами в интересах достижения поставленных в задаче целей. При этом необходимо обеспечить анализ информации о текущей обстановке и реакцию на ее изменение в процессе решения задачи.

Применительно к рассматриваемой нами предметной области, упомянутые выше алгоритмы, как правило, строго регламентированы боевыми уставами. Например, алгоритмы движения подводных лодок (ПЛ) и использования ими вооружения в процессе выполнения поставленных задач четко описаны и легко представимы в виде блок-схемы. То же самое касается алгоритмов поведения ПЛ в ситуациях, которые могут возникнуть в процессе выполнения поставленных задач в результате изменения боевой обстановки. Введем термин «поведенческие модели», объединяющий алгоритмы поведения агента в рамках решения конкретных тактических задач и алгоритмы реакций на изменение обстановки.

Учитывая вышесказанное, для программирования поведенческих моделей удобным представляется инструмент, который бы позволил оператору формировать алгоритмы в виде графической схемы, основными элементами которой являются:

– процессоры – реализуют микропрограммы управления ТЕ («движение с заданными параметрами» и т. п.);

– функции – обеспечивают расчет входных параметров для процессоров («расчет параметров движения» и т. п.);

– связи – определяют условия и порядок передачи управления между процессорами и функциями.

Собранная по такому принципу схема может выглядеть следующим образом:

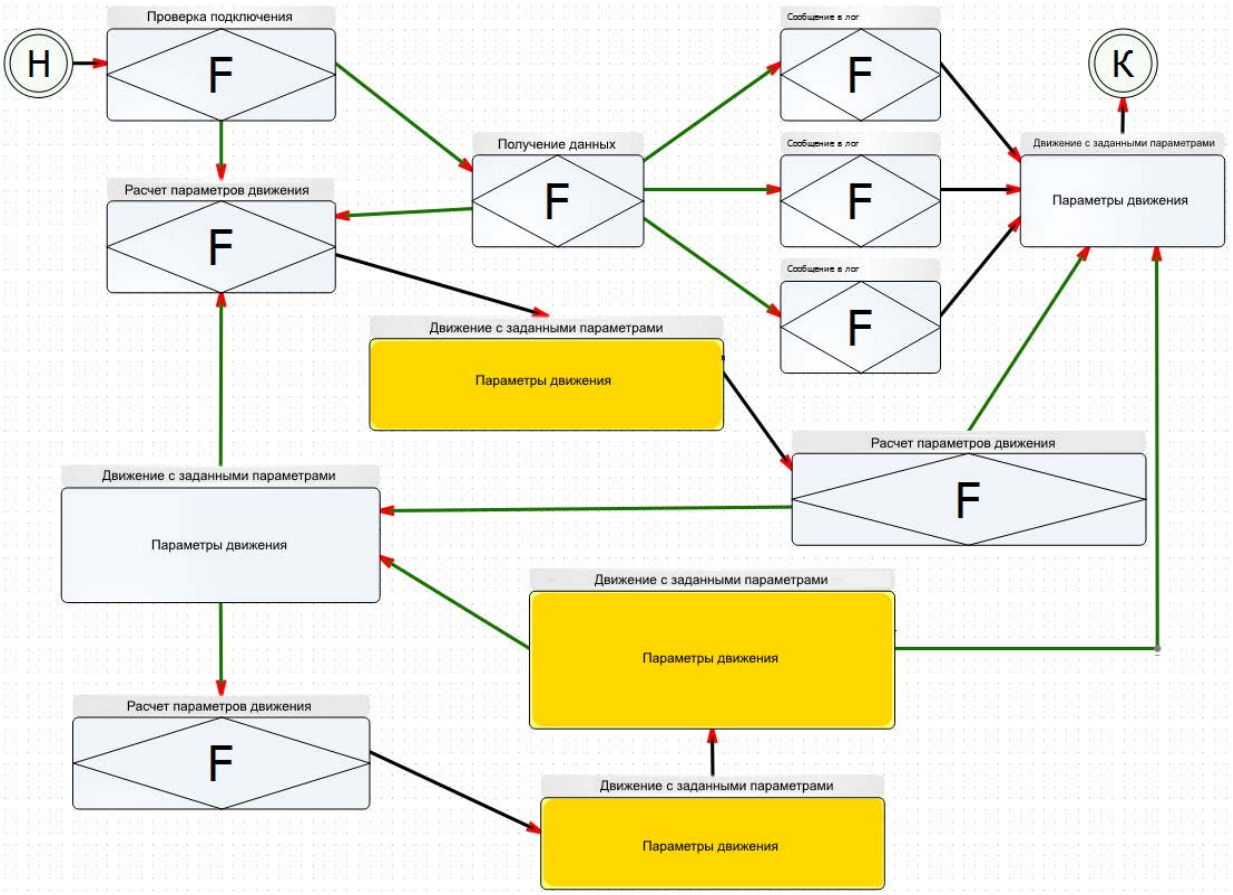

Рис. 2. Пример схемы алгоритма поведения агента типа «тактическая единица»

Для полного описания логики работы агента «тактическая единица», кроме алгоритмов его поведения в рамках решения конкретных тактических задач и реакций, необходимо задать правила обработки информации об изменении окружающей обстановки, поступающей от сенсоров, и активации того или иного алгоритма решения задачи или реакции. Назовем эти правила термином «логика системы управления». В процессе работы системы управления можно выделить следующие этапы:

1. Обработка информации, поступающей от сенсоров, и распознавание событий, имеющих семантическое значение в рамках описания логики работы агента (получение приказа, обнаружение цели и т. п.)

2. Анализ информации о произошедших событиях, текущем состоянии агента и окружающей среды, выбор актуальной поведенческой модели

3. Активация актуальной поведенческой модели

4. Алгоритмы этапов 1 и 2 могут быть довольно сложными, требующими для удобного описания таких языковых конструкций как циклы, ветвления, множественный выбор и т. п. Поэтому в качестве инструмента программирования логики системы управления целесообразно рассматривать язык общего назначения (например, Python), расширенный библиотеками проблемно-ориентированной направленности. В нашем случае эти библиотеки могут содержать такие функции как:

– опрос локальных сенсоров;
– отправка сообщения по каналу радиосвязи;
– получение сообщений по каналу радиосвязи;
– получение текущей активной поведенческой модели;
– активация поведенческой модели;
– расчет расстояния до точки на карте;
– получение модельного времени;

и т. п.

Очевидно, что рассмотренные ранее поведенческие модели дополняют логику системы управления и вместе с ней определяют полную логику поведения агента «тактическая единица». Тем не менее, принимая во внимание указанные особенности алгоритмизации, авторам представляется целесообразным выделение для программирования поведенческих моделей и логики системы управления отдельных интерфейсных модулей, обладающих вышеприведенными свойствами.

Программирование логики агентов типа «тактическая группа»

Введенные нами понятия «поведенческие модели» и «логика системы управления» можно применить и к агентам типа «тактическая группа». Задачей агентов данного типа является формирование и уточнение индивидуальных задач для подчиненных тактических единиц, исходя из получаемых приказов и информации о среде. Учитывая это, в качестве поведенческих моделей агентов «тактическая группа» будем рассматривать алгоритмы назначения частных задач подчиненным тактическим единицам и расчета входных параметров для этих задач, а в качестве логики системы управления - алгоритмы обработки приказов и информации, получаемой от подчиненных тактических единиц, активации поведенческих моделей.

При таком подходе, относительно желаемого образа инструмента программирования логики системы управления агентов типа «тактическая группа» можно сделать выводы, аналогичные выводам, сделанным для агентов «тактическая единица». Так же, как и в случае с агентами «тактическая единица», здесь целесообразно применение языка общего назначения, расширенного библиотеками проблемно-ориентированной направленности. Состав функций этих библиотек должен быть аналогичен составу, приведенному для агентов типа «тактическая единица», но дополнен функциями, позволяющими обрабатывать информацию о расположении, свойствах и текущем состоянии подчиненных тактических единиц:

– получение списка подчиненных тактических единиц;
– получение информации о расположении подчиненной тактической единицы;
– получение текущей задачи подчиненной тактической единицы;
– получение текущего состояния подчиненной тактической единицы;

и т. п.

Поскольку в процессе выполнения поведенческих моделей агентов данного типа в основном производятся различного рода вычисления, подразумевающие оценку текущей обстановки и расчеты значений входных параметров подчиненных тактических единиц, для программирования их логики также целесообразно применение языка общего назначения с расширениями, обеспечивающими доступ к геоинформационным данным, информации о тактико-технических характеристиках и текущем состоянии подчиненных тактических единиц.

Программирование логики агентов типа «объединение»

Как было отмечено выше, задачи агентов типа «объединение» схожи с задачами агентов типа «тактическая группа». Но если агент типа «тактическая группа» решает задачи тактического уровня и при принятии решения оценивает обстановку в рамках нескольких подчиненных тактических единиц, то агент типа «объединение» решает задачи оперативно-тактического, оперативного и оперативно-стратегического уровней и при принятии решений оценивает обстановку в рамках нескольких тактических групп.

С повышением уровня управления возрастает сложность формализации действий агента, осуществляющего управление. Если для тактического уровня алгоритмы управления и действий в большинстве случаев четко определены боевыми уставами, то в оперативном искусстве определения таких алгоритмов отсутствуют. Основной целью оперативных маневров является создание превосходства над противником на главном направлении в операции за счет изменения состава группировок, оперативного построения сил, использования резервов, переноса усилий на другое

операционное направление. Таким образом, можно сказать, что критерием достижения основной цели оперативных маневров является обеспечение превосходства над противником на главном направлении, причем этот критерий не меняется от одной задачи к другой, в отличие от случая с тактическими маневрами, в которых критерии достижения основной цели определяются индивидуально по каждой задаче. В процессе моделирования агенту типа «объединение» необходимо непрерывно производить анализ всей имеющейся информации о своих силах и силах противника с целью проверки выполнения условия достижения основной цели – превосходства над противником на главном направлении, и если обнаружено, что это условие не выполнено, обеспечить его выполнение вышеперечисленными способами.

Тем не менее, основной проблемой при создании инструмента программировании логики поведения агента типа «объединение» остается отсутствие возможности четкой формализации действий при принятии решения и определения детерминированных алгоритмов. В качестве решения данной проблемы может быть рассмотрено применение математического аппарата нечеткой логики и построенных на его основе нейронных сетей. Задачу управления силами можно декомпозировать на две основные подзадачи: анализ текущего соотношения сил на главном направлении и принятие компенсирующих мер для обеспечения выгодного для себя соотношения сил на главном направлении, исходя из имеющихся сил и средств. Первая подзадача может быть решена посредством простого сравнения сил по выделенным критериям, либо с применением более сложной методики оценки боевой эффективности. Для решения второй задачи целесообразно применение экспертной системы, дополненной нейронной сетью. Экспертная система (ЭС) позволит выработать решение на основе базы данных решающих правил. Если ЭС столкнется с ситуацией, которую эксперты не рассмотрели заранее, нейронная сеть может отнести непредвиденную ситуацию к одной из рассмотренных ранее. В том случае, если ситуацию не удалось отнести к одной из ранее рассмотренных, нейронная сеть, исходя из имеющихся сил и средств, а также доступной информации о противнике, определит наиболее выгодную, с точки зрения решаемой оперативной задачи конфигурацию сил, при которой будет обеспечено превосходство над противником на главном направлении. В этом случае будет определено новое правило.

В то время как экспертные системы используют правила импликации и логический вывод, нейронные сети имеют способность к обучению. Эта совокупность качеств делает нейронные сети и экспертные системы достойными претендентами на формирование гибридной интеллектуальной системы [21].

Таким образом, технология программирования логики поведения агента будет сводиться к процессам заполнения базы данных решающих правил и обучения нейронной сети.

Для экспертной системы должны быть определены сущности предметной области (цели, способы решения, возможные варианты событий и др.) и измеряемые характеристики (вероятности наступления событий, коэффициенты значимости целей, приоритетность решений и др.). При описании правил целесообразно использовать следующие базовые понятия: объект, показатель (признак) и процедура сравнения. В нашем случае объектами являются агенты типа «объединение», «тактическая единица», «тактическая группа». В качестве показателей могут быть использованы пространственно-временные характеристики, физические свойства объектов (степень поражения объекта, количество боеприпасов и т. д.). Процедура сравнения должна учитывать: причинно-следственные связи между объектами, степень влияния одних объектов на другие.

Основными типами алгоритмов обучений нейронных сетей [22] являются: обучение с учителем, обучение с подкреплением, обучение без учителя. В нашем случае целесообразно применение именно алгоритмов обучения с подкреплением. Данный подход позволит решить проблемы отсутствия данных о крупных боевых столкновениях, ограниченности доступа к результатам и эффективности учений потенциального противника, что делает обучение с учителем принципиально невозможным без размеченных наборов данных, отвечающих требованиям репрезентативности. Анализ литературных источников подтверждает высокий уровень интеллектуальных способностей агентов, обучаемых с подкреплением (системы AlphaGo [23], DeepMind [24], OpenGym AI [25]). И хотя агенты, обученные с помощью перечисленных программных продуктов, не использовались для детального моделирования двусторонних боевых действий, те принципы, на которых строились

модели принятия решений агентов, могут быть реализованы и для решения актуальных задач имитационного моделирования с использованием библиотек для машинного обучения с открытым исходным кодом и не накладывающих лицензионных ограничений.

В ходе обучения с подкреплением программный агент совершает наблюдения и предпринимает действия внутри среды моделирования, возвращающей награды и штрафы. Цель агента – максимизация долгосрочных наград и сокращение штрафов, это довольно абстрактная установка, конкретная реализация которой может достигаться с использованием таких приемов машинного обучения как градиенты политики и глубокие Q-сети. Ниже приведен распространенный вариант алгоритма REINFORCE (REward Increment = non-Negative Factor × Offset Reinforcement × Characteristic Eligibility):

- Запуск моделирования, при котором для политики в форме нейросетевой модели вычисляются (но не применяются) градиенты, определяющие изменение вероятности совершения выбранного действия;
- После прогона нескольких эпизодов вычисляются оценки каждого действия, например, с использованием алгоритма дисконтной ставки;
- Перемножение вектора градиента и оценки действия для изменения вероятностей совершения действия;
- Применение результирующих векторов-градиентов для изменения весовых коэффициентов нейросетевой модели с использованием алгоритмов оптимизации (в простейшем случае градиентного спуска).

Одной из проблем данного подхода является оценка действий, ввиду задержки в получении наград от среды. Совершив ряд действий и, достигнув определенного результата, агент должен оценить, какие из совокупности действий приводили к достижению результата, а какие мешали. Очевидно, что конечный результат определяется не последним действием, а всей совокупностью. Одним из распространенных подходов решения проблемы назначения доверительных коэффициентов является использование дисконтной ставки, при которой оценка совокупности действий определяется по следующей формуле:

$$N = \sum_{t} n_t \times r^t,\ i = 0 \ldots (M-1), 0 \leq r \leq 1.$$

где N – оценка совокупности действий; M – количество совершенных агентом действий, n_i – награда за отдельное действие, r – значение дисконтной ставки. Чем выше значение дисконтной ставки, тем более значимыми будут предыдущие действия.

Заключение

В результате проведенного анализа подходов к построению инструментальных средств программирования логики поведения агентов в мультиагентных системах имитационного моделирования двухсторонних боевых действий была выполнена систематизация типов агентов и выстроена иерархия с точки зрения системы управления. Установлено, что с повышением уровня иерархии возрастает степень абстрактности применяемых алгоритмов. Для каждого типа агентов предложено решение, определяющее облик наиболее подходящего инструментального средства для программирования логики их поведения:

Для агентов типа “тактическая единица” – конструктор на основе графических схем;

Для агентов типа “тактическая группа” – язык общего назначения, расширенный библиотеками проблемно-ориентированной направленности:

Для агентов типа “объединение” – заполнение базы данных решающих правил и обучение нейронной сети.

Литература

1. Люгер Д. Ф. Искусственный интеллект: стратегии и методы решения сложных проблем: Пер. с англ. – М.: Издательский дом «Вильямс», 2003.

2. Энциклопедический словарь Ф. А. Брокгауза и И. А. Ефрона. – СПб.: Полрадис, 1993.

3. AgentBuilder: An Integrated Toolkit for Constructing Intelligent Software Agents [Electronic Resourse] / Reticular Systems, 1999. – Mode of access : http://www.agentbuilder.com.

4. Brookings R. S. Ascape: An Agent Based Modeling Framework in Java [Elec-tronic Resourse] / R. S. Brookings. – [Electronic Data]. - Brookings Center on Social and Economic Dynamics, 2000. – Mode of access : http://www.brook.edu/es/dynamics/models/ascape/.

5. Bee-gent Multi-Agent Framework [Electronic Resourse] / Toshiba Corporation Systems and Software Research Laboratories, 2000. [Electronic Data]. – Mode of access: http://www2.toshiba.co.jp/beegent/index.htm.

6. Dee, C. CABLE: A multi-agent architecture to support military command and control / C. Dee, P. Millington, B. Walls, T. Ward // Proceedings of PAAM 2000. -Manchester, UK, April 2000. P. 322–330.

7. Graham, J. Tools for Developing and Monitoring in Distributed Multi-Agent Systems / J. Graham, V. Windley, D. McHugh, F. McGeary, D. Cleaver, K. Decker // Pro-ceedings of the Workshop on Agents in Industry. Barcelona, Spain, June 2000. – http://www.eecis.udel.edu/~decaf/.

8. Poslad, S. J. The FIPA-OS agent platform: Open Source for Open Standards [Electronic Resourse]/ S. J. Poslad, S. J. Buckle, R. Hadingham // Proceedings of PAAM 2000. – Electronic Data. – Manchester, UK, April 2000. – P. – Mode of access : http://fipa-os.sourceforge.net.

9. Grasshopper, Release 2.2. Basics and Concepts (Revision 1.0), March 2001. – IKV++GmbH. – 70 p.

10. Jazayeri M. Gypsy: A Component-based Mobile Agent System [Electronic Re-sourse] / M. Jazayeri, W. Lugmayr // Proceedings of the 8th Euromicro Workshop on Parallel and Distributed Processing (PDP2000). – Electronic Data. – Rhodos, Greece, 2000. – http://www.infosys.tuwien.ac.at/Staff/lux/Gypsy.

11. Bellifemine, F. JADE – A FIPA-compliant agent framework [Electronic Re-sourse] / F. Bellifemine, A. Poggi, G. Rimassa // Proceedings of PAAM 2000. – Electronic Data. – Manchester UK, April 2000. – P. – Mode of access : http://sharon.cselt.it/projects/jade.

12. Bordini R. H., Hubner J. F., Wooldridge M. Programming Multi-Agent Systems in AgentSpeak with Jason. – Jonh Wiley&Sons: Chichester, 2007. – 294 p.

13. JATLite – http : // java.stanford.edu/java-agent/html.

14. JAFMAS – http://www.ececs.uc.edu/~abaker/JAFMAS.

15. Raphael M. J. A knowledge base for knowledge-based multiagent system con-struction / M. J. Raphael, S. A. Deloach. // National Aerospase and Electronics Conf. (NAECON), Dayton, OH, October 10-20, 2000. – http://www.cis.ksu.edu/~sdeloach/publications/Conference/arams-naecon.pdf .

16. Kouadri, G. A generic framework for context-based distributed authorizations/ G. Kouadri, P. Brezillon // Proc. Of the fourth Int. Conf. on Modeling and Using Context, Stanford, California (USA), June 23–25, 2003. P. 326–333.

17. Reference Guide for Swarm 2.1.1. [Electronic Resourse] / Swarm Develop-ment Group. – Electronic Data. – Mode of access : http:// www.santafe.edu/projects/swarm/swarmdocs/set/book930.html.

18. Collis, J. The Zeus Agent Bilding Toolkit : ZEUS Technical Manual [Electron-ic Resourse] / J. Collis, D. Ndumu // Intelligent Systems Research Group, BT Labs, Release 1.0. 1999. – Electronic Data. – Mode of access : http://193.113.209.147/projects/agents/index.htm.

19. Городецкий В. И. MAS DK: инструментарий для разработки многоагентных систем и примеры приложений / В. И. Городецкий, О. В. Карсаев, И. В. Хотенко, А. В. Хабалов // Труды Международного конгресса «Искусственный интеллект в XXI веке» (ICAI 2001), 3–8 сентября 2001 г. М.: Физматлит, 2001. С. 249–262.

20. Швецов А. Н. Агентно-ориентированные системы: от формальных моделей к промышленным приложениям. // Всероссийский конкурсный отбор обзорно-аналитических статей по приоритетному направлению «Информационно-телекоммуникационные системы», 2008. С. 75–76.

21. Алдошина А. Н. Экспертная система на основе нейросетевых технологий для мониторинга и диагностики корпоративной локальной сети // Молодой ученый – 2016. № 18(122). С. 37.

22. Hastie T., Tibshirani R. and Friedman J. “Elements of Statistical Learning”, Springer, 2009.

23. Silver, D., Huang, A., Maddison, C. et al. Mastering the game of Go with deep neural networks and tree search. Nature 529, 484–489 (2016) doi:10.1038/nature16961.

24. Irina Higgins, Arka Pal, Andrei Rusu, Loic Matthey, Christopher Burgess, Alexander Pritzel, Matthew Botvinick, Charles Blundell, and Alexander Lerch-ner. 2017. DARLA: Improving Zero-Shot Transfer in Reinforcement Learning. In Proceedings of the 34th International Conference on Machine Learning (Proceedings of Machine Learning Research), Vol. 70. PMLR, 1480–1490.

25. Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. 2016. OpenAI Gym.(2016). arXiv:cs.LG/1606.01540.

PROBLEMS OF DEVELOPING TOOLS FOR THE AGENT BEHAVIOR LOGIC PROGRAMMING IN MULTIAGENT SYSTEMS FOR SIMULATION OF TWO-SIDED MILITARY OPERATIONS

K. V. Ivanov, M. V. Galkin, A. I. Sayfullin, R. N. Sayfullina, D. V. Devyatykh

Russian Federal Nuclear Center –
All-Russian Scientific Research Institute of Experimental Physics, Sarov

The paper considers the key aspects of developing tools for the agent behavior logic programming in multiagent systems and problems encountered in the development process are articulated. A brief review of available agent logic programming tools, including both foreign and Russian software products, is given.

The paper is focused on the simulation of two-sided military operations for experts in this subject. Three types of agents are identified: a tactical unit, a tactical group, and a combination. The agent communication and interaction structure, as well as the specific features of each agent type are presented. The solution is suggested for each type of agents, which determines the most relevant and easy to use tool for experts in this area to program the agent behavior logic.

Key words: simulation, agent, multiagent system, programming, behavior logic, control system