Модел. и анализ информ. систем. Т. **18**, № **2** (2011) 113–128

УДК 004.896 + 519.713

Задача о роботах на Марсе (мультиагентный подход к задаче Дейкстры)

Бодин Е.В., Гаранина Н.О.[1], Шилов Н.В.[2]

Институт Систем Информатики им. А.П. Ершова СО РАН

e-mail: {bodin, garanina, shilov}@iis.nsk.su

получена 9 февраля 2011

Ключевые слова: мультиагентная система, распределённый алгоритм, задача о назначениях, планирование перемещений

Изучаются мультиагентные алгоритмы для так называемой задачи о роботах на Марсе. Эту задачу можно рассматривать как задачу из теории графов (дискретная математика), как комбинаторную геометрическую задачу (теоретическое программирование) или как частный случай задачи планирования перемещений (искусственный интеллект). Наши алгоритмы основаны на эвристическом поиске, предложенном Э. Дейкстрой. В статье представлен ряд новых мультиагентных алгоритмов, решающих задачи о роботах на Марсе, доказана их корректность, приведены результаты проверки на модели некоторых из этих алгоритмов, предложены направления дальнейших исследований. Новизна представленной работы состоит в том, что в отличие от теоретико-графового и комбинаторно-геометрического подходов, ориентированных на централизованное решение задачи, мы развиваем мультиагентный подход, но, в свою очередь, наша работа отличается от работ по планированию перемещений математической строгостью доказательств корректности предложенных алгоритмов.

Я, робот.
А. Азимов

1. Введение

Мультиагентные системы — это распределённые системы [12], состоящие из агентов. Агент — это автономный реагирующий объект (в объектно-ориентированном смысле) с внутренним состоянием, которое можно охарактеризовать в терминах

[1]Работа поддержана грантом РФФИ 10-01-00532-а.
[2]Работа поддержана интеграционным проектом РАН № 2/12.

“мнений”, “представлений”, “убеждений” или “веры” (Believes), целей (Desires) и намерений (Intentions) агента. Вера[3] агента является совокупностью его мнений о себе и других агентах, а также о среде, которые могут быть неполными, несогласованными и вообще неверными. Цели агента — это его долгосрочные задачи и обязанности, которые также могут быть несогласованны. Намерения агента используются для краткосрочного планирования. Реагирующим агент является в том смысле, что он может изменить свою веру, цель или намерение после взаимодействия с другими агентами или средой, но каждый агент всё-таки автономен, что означает, что изменение его внутреннего состояния зависит только от него, но не от окружения. Агенты описанного типа обычно называются BDI-агентами [7]. Мультиагентный алгоритм — это распределённый алгоритм [12], который решает какую-либо задачу совместными усилиями агентов в мультиагентной системе.

В данной статье мы исследуем мультиагентные алгоритмы для следующей задачи о Роботах на Марсе (Mars Robot Puzzle — MRP).

> На плоском участке поверхности Марса находятся $n > 1$ автономных роботов и столько же укрытий в общем положении[4]. Местоположение всех укрытий фиксировано и известно каждому роботу. Каждый робот знает о существовании всех остальных роботов, но не знает их координаты. Роботы могут взаимодействовать каждый с каждым попарно (Peer to Peer). В некоторый момент времени все роботы останавливаются, фиксируют свои текущие позиции, и затем они все должны выбрать укрытия, чтобы двинуться к ним по прямому маршруту. Ясно, что роботам нельзя сталкиваться; поэтому каждый отдельный робот может двигаться к укрытию, только когда он знает (абсолютно уверен), что на своём пути он не столкнётся с другим роботом. Задача: разработать мультиагентный алгоритм, гарантирующий, что каждый робот когда-нибудь будет точно знать, что его маршрут к выбранному укрытию не пересекается с маршрутами остальных роботов.

Данная задача — это мультиагентная интерпретация следующей задачи Дейкстры [5, 6]:

> Разработать алгоритм, соединяющий попарно данные N чёрных и N белых точек в общем положении на плоскости таким образом, чтобы отрезки, связывающие точки, не пересекались.

Задачу Дейкстры можно рассматривать как частный (“геометрический”) случай задачи о назначениях из теории графов [5, 6, 10], как частный случай комбинаторной геометрической задачи [5, 6, 8, 13], тогда как MRP может также рассматриваться как частный случай задачи планирования перемещений [3].

Некоторые предварительные результаты по MRP были опубликованы в [2]. В цитируемой статье для агентов предлагается три совершенно различных алгоритма.

[3]Мы предпочитаем использовать именно термин “вера”, а не “мнения”, “представления” или “убеждения”, которые иногда используются в литературе на русском языке [9, 14].

[4]т. е. никакие три объекта не лежат на одной прямой.

Первый алгоритм, самый простой, подходит для "прыгающих" роботов (способных перемещаться мгновенно). Но гипотеза "прыгающих" роботов — слишком сильное допущение. Второй алгоритм очень строго синхронизирует независимых роботов. Это тоже не очень хорошее решение, поскольку роботы в определенной степени теряют автономность. В третьем алгоритме роботы асинхронно избирают лидера, затем посылают ему свои данные, после чего лидер решает задачу Дейкстры (или даже задачу о назначениях) любым подходящим детерминированным способом. Но последнее решение получается псевдо-мультиагентным, поскольку оно скрывает детерминированный алгоритм за выборами лидера. Поэтому в данной статье мы хотели бы представить новое мультиагентное решение, лишённое перечисленных недостатков.

Оставшаяся часть статьи организована следующим образом. Следующий раздел описывает наш базовый мультиагентный алгоритм для MRP. Этот алгоритм основан на допущении справедливости и отсутствии тупиков в процессе обмена сообщениями и использует заданное начальное (произвольное, но взаимно-однозначное) назначение укрытий. Третий раздел посвящён задаче обеспечения справедливого и беступикового планирования контактов (с помощью введения предопределённых приоритетов для роботов или глобального времени) и генерации начального назначения укрытий (в трёх вариантах). Последний раздел — заключение, в котором мы обсуждаем направления дальнейших исследований.

2. Алгоритм поиска безопасного маршрута

Целью каждого робота является выбор индивидуального укрытия так, чтобы избежать столкновения на пути к нему. Эта цель достигается посредством кооперативного поиска бесконфликтного/безопасного (т. е. без пересечений) множества прямолинейных маршрутов к выбранным укрытиям. В каждый момент времени в намерениях каждого робота содержится его текущее укрытие-назначение. Вера каждого робота содержит два целочисленных счётчика: NC для "числа собственных конфликтов" и CFR для "числа бесконфликтных роботов"; NC представляет верхнюю оценку числа роботов, с которыми данный робот может иметь конфликт, а соответственно CFR — нижняя оценка числа роботов, которые вообще конфликтов не имеют. Кроме того, пусть у каждого робота есть несколько переменных для вещественных чисел, он умеет складывать, вычитать, сравнивать вещественные значения и вычислять расстояния (от своей позиции на плоскости до укрытий).

Теперь можно представить алгоритм для каждого отдельного робота (он называется SMEx - Safe Mars Explorer). В этом алгоритме и следующих будем использовать псевдокод, в основном совпадающий с псевдокодом алгоритмов из [12]. Кроме того, условимся описывать явно в алгоритмах только наиболее существенные переменные, а описания вспомогательных переменных опущены, но ясны из контекста.

```
algorithm SMEx::
var NC, CFR : integer;
var s11, s12, s21, s22 : real;
var contacts : множество роботов;
```

```
begin
1: NC:= (n - 1); CFR:= 0;
   текущее назначение := начальное назначение для данного робота;[5]
2: repeat
3:    if NC > 0 then NC:= (n - 1);
4:    contacts:= множество всех остальных роботов;
5:    repeat
6:      partner := робот из contacts, который готов к общению;[6]
7:      начать переговоры с партнёром partner:
8:      send <NC> to partner; receive <NC партнёра> from partner;
9:      send <текущее назначение> to partner;
        receive <назначение партнёра> from partner;
10:     s11:= расстояние до своего текущего назначения;
11:     s12:= расстояние до текущего назначения партнёра;
12:     send <s11,s12> to partner; receive <s22,s21> from partner;
13:     if s11 + s22 > s12 + s21
14:     then{ обменяться назначениями с партнёром; //операция FLIP
             NC:= (n - 1); CFR:= 0 }
15:     else if NC > 0
16:         then { NC:= (NC - 1); CFR:= 0 }
17:         else if NC = 0
18:             then if NC партнёра = 0
19:                  then CFR:= CFR + 1
20:                  else { NC:= (n - 1); CFR:= 0 }
21:     завершить переговоры с партнёром;
22:     contacts:= удалить партнёра из списка контактов;
23:   until множество контактов не опустеет;
24: until (NC = 0 ∧ CFR = 2*(n - 1) )
25: выбранное укрытие:= текущее назначение;
end.
```

Для каждого робота данный алгоритм делит время своего исполнения на раунды — интервалы, в течение которых робот должен иметь ровно один контакт с каждым другим роботом; согласно описанию алгоритма, раунд соответствует внутреннему циклу **repeat-until** (строки 3–23). Кроме того, мы различаем два типа раундов: процесс и пост-процесс. Робот во время процесса подсчитывает в своей переменной NC, сколько ещё конфликтов может быть с другими роботами, т. е. "формирует" веру, имеет ли он конфликты (строки 14–16). Во время пост-процесса каждый робот, который верит, что у него конфликтов нет (т. е. его $NC = 0$), накапливает в своём счётчике CFR, сколько других роботов тоже считают себя бесконфликтными (т. е. их собственный $NC = 0$) (строки 17–19). Заметим, что процесс и пост-процесс могут чередоваться (так как NC и CFR могут реинициализироваться в строке 20).

[5]Реализация этого присваивания обсуждается в разделе 2.

[6]Реализация этого присваивания обсуждается в разделе 1.

Лемма 1. *Пусть мультиагентная система состоит из $n > 1$ роботов, которые исполняют алгоритм SMEx. Тогда в каждый момент времени работы мультиагентной системы прямолинейные маршруты к текущим назначениям-укрытиям от позиций тех роботов, которые только что закончили раунд (т. е. исполняют строку 24) с персональным счётчиком $NC = 0$, не пересекаются друг с другом.*

Доказательство. Предположим противное, что в некоторый момент времени пересекаются маршруты двух роботов с нулевыми значениями индивидуальных счётчиков NC. Рассмотрим последний контакт между этими роботами. Он произошёл на самом последнем раунде обоих роботов.

Заметим, что для каждого робота, если значение его счётчика NC убывает от $(n-1)$ до 0 в течение раунда, то в этом раунде робот не выполнял операцию $FLIP$ (строка 14) и не имел конфликтов с партнёрами во время контакта с ними (т. к. в противном случае условие в строке 13 было бы выполнено хотя бы раз в этом раунде, NC реинициализировался бы в строке 14 и не мог бы достичь значения 0).

Следовательно, в течение своих последних раундов оба робота обнаруживают, что их маршруты не пересекаются с маршрутами других роботов (на момент контакта). В частности, они считают, что в момент их последнего контакта их маршруты не пересекались. Тем не менее, если их маршруты пересеклись после контакта, из этого следует, что хотя бы один из них выполнил операцию $FLIP$ после их контакта, и, следовательно, значение персонального счётчика NC этого робота не может быть 0 в конце раунда (как было показано в предыдущем абзаце). Противоречие. ■

Будем говорить, что общение между роботами удовлетворяет условию справедливости и беступиковости, если в любой момент времени для любого робота, желающего контактировать в этот момент с каким-либо другим роботом-партнёром, обязательно наступит в будущем момент времени[7], когда партнёр будет готов к общению.

Лемма 2. *Пусть мультиагентная система состоит из $n > 1$ роботов в общем положении, которые исполняют алгоритм SMEx, все роботы имеют взаимно-однозначно назначенные начальные укрытия, а общение между роботами удовлетворяет условию справедливости и беступиковости. Тогда система когда-нибудь обязательно остановится (т. е. все роботы завершат исполнение алгоритма).*

Доказательство. Предположим противное: пусть есть роботы, которые будут "работать вечно", т. е. не завершат исполнение алгоритма SMEx. В силу условия, что все роботы имеют взаимно-однозначно назначенные начальные укрытия, а общение между роботами удовлетворяет условию справедливости и беступиковости, каждый оператор алгоритма SMEx выполняется за конечное время (т. е. не ждёт бесконечно долго реакции партнёра). Поэтому незавершение алгоритма SMEx для таких роботов означает, что у них условие $(NC = 0 \wedge CFR = 2(n-1))$ в строке 24 всегда ложно. Это влечёт, что они будут исполнять $FLIP$ бесконечное число раз (также как и реинициализацию NC и CFR в строке 14). Но каждая операция $FLIP$ уменьшает значение общей суммы длин прямолинейных маршрутов всех роботов к их текущим укрытиям, и имеется всего $n!$ возможных значений этой суммы. Противоречие. ■

[7]будущий по отношению к текущему моменту времени.

Теорема 1. *Пусть мультиагентная система состоит из $n > 1$ роботов в общем положении, исполняющих алгоритм SMEx, все роботы имеют взаимно-однозначно назначенные начальные укрытия, а общение между роботами удовлетворяет условию справедливости и беступиковости. Тогда система когда-нибудь завершит свою работу, и в этом состоянии не будет пересекающихся путей роботов к их выбранным назначениям.*

Доказательство. Остановка системы доказана в лемме 2. Осталось доказать, что это "заключительное" состояние свободно от конфликтов. Для этой цели зафиксируем произвольного робота R и докажем, что он не имеет конфликтов с другими роботами.

Заметим, что два последних раунда его деятельности — это пост-процессные раунды, где значение его NC все время равно 0, а значение его CFR монотонно возрастает от 0 до конечного значения $2(n-1)$. (Иначе его значение CFR не сможет достичь своего конечного значения $2(n-1)$, как следует из строк алгоритма 17 - 20.)

В течение первого из этих двух раундов R уведомляет всех других роботов, что его $NC = 0$, получает аналогичные уведомления от всех остальных роботов, и монотонно наращивает значение своего CFR от 0 до $(n-1)$; но то, что он получил 0 в качестве значения NC партнёра, означает, что этот партнёр выполняет свой собственный пост-процессный раунд. Следовательно (в соответствии с леммой 1), после этого раунда робот R знает, что на момент его переговоров с каким-либо роботом их маршруты не пересекаются.

В течение второго из этих двух раундов R уведомляет всех других роботов, что его $NC = 0$, получает аналогичные уведомления от всех остальных роботов и монотонно наращивает значение своего CFR от $(n-1)$ до конечного значения $2(n-1)$. Но то, что он получил 0 в качестве значения NC партнёра, означает, что этот партнёр выполняет свой собственный пост-процессный раунд, который тоже должен быть вторым пост-процессным раундом. Следовательно, в этом раунде все роботы сообщают роботу R (послав 0 как значение их NC), что они не меняли своего назначения в предыдущем раунде. Согласно лемме 1, маршрут робота R не пересекается с маршрутами всех других роботов. ■

Таким образом, Задача о Роботах на Марсе может быть решена алгоритмом Safe Mars Explorer при предположении, что каким-то образом гарантировано свойство справедливости и беступиковости общения между роботами и что роботам известно взаимно-однозначное начальное назначение укрытий.

3. Вспомогательные алгоритмы

Пусть роботы имеют уникальные имена. Для простоты мы считаем, что имена — это натуральные числа от 1 до n. Каждый робот "знает" свой индивидуальный номер-имя (и хранит его в переменной i). Для удобства представления алгоритмов мы вынесем в отдельную процедуру $change(i, j)$ часть алгоритма SMEx (строки 8 - 20), которая задаёт общение и обмен укрытиями между i-м и j-м роботом:
procedure `change(i, j)::`

```
const partner = j : робот;
var NC, CFR : integer;
var s11, s12, s21, s22 : real;
begin
1: send <NC> to partner; receive <NC партнёра> from partner;
2: send <текущее назначение> to partner;
   receive <назначение партнёра> from partner;
3: s11:= расстояние до своего текущего назначения;
4: s12:= расстояние до текущего назначения партнёра;
5: send <s11,s12> to partner; receive <s22,s21> from partner;
6: if s11 + s22 > s12 + s21
7: then{ обменяться назначениями с партнёром; //операция FLIP
        NC:= (n - 1); CFR:= 0 }
8: else if NC > 0
9:     then { NC:= (NC - 1); CFR:= 0 }
10:    else if NC = 0
11:        then if NC партнёра = 0
12:            then CFR:= CFR + 1
13:            else { NC:= (n - 1); CFR:= 0 }
end.
```

1. Планирование контактов

Общение между агентами возможно или непосредственное синхронное, или опосредованное синхронное. При непосредственном общении агент, послав запрос на общение другому агенту, ждёт ответа, не предпринимая никаких других действий, пока тот не откликнется. При опосредованном общении агент посылает партнёру запрос на общение с указанием времени, удобного для переговоров, после чего может не ждать немедленного ответа. Партнёр может согласиться или назначить другое время. Очевидно, что во втором случае мультиагентную систему придётся расширить глобальными часами, которые "видны" всем агентам.

Рассмотрим замкнутую систему агентов, обладающих абсолютно одинаковыми возможностями. В случае непосредственного общения справедливое планирование в такой системе организовать невозможно, поскольку может возникнуть тупиковая ситуация, когда один агент ждёт ответа от второго, второй от третьего, а третий от первого. В случае опосредованного общения даже двое агентов не смогут договориться о времени общения, поскольку эта ситуация сводится к задаче об атакующих генералах [1], в которой гонец (среда) оказывается ненадёжным в том смысле, что время прочтения письма адресатом произвольно.

Однако справедливое и беступиковое планирование контактов легко организовать посредством введения в систему приоритетов или внешнего планирования взаимодействия агентов (например, с помощью "диспетчера"). В этом разделе будут представлены два варианта справедливого беступикового планирования контактов: с использованием приоритетов и с использованием заранее определённых возможностей назначить время контакта, что является ослабленным вариантом внешнего

планирования.

Предположим, что индивидуальное имя-номер задаёт приоритет робота, и, таким образом, для каждого робота с номером i роботы с номерами от 1 до $(i-1)$ — это "младшие" роботы, а роботы с номерами от $(i+1)$ до n — это "старшие" роботы. В следующей модификации алгоритма SMEx общение между роботами непосредственное, а приоритеты влияют на возможность посылать запрос на общение (младшие могут запрашивать старших):

```
algorithm SMEx2::
const i : робот; // Собственный номер робота
var NC, CFR : integer;
var Senior, Junior : множество роботов;
var Ready_Junior[1..(i -1)] : boolean; // Ready_Junior[k]=true означает,
          // что Junior с номером k готов к контакту, false -- не готов.
begin
1: NC:= (n - 1); CFR:= 0;
   текущее назначение := начальное назначение для робота i;
2: repeat
3:    if NC > 0 then NC:= (n - 1);
4:    Junior:= множество всех младших роботов;
5:    Senior:= множество всех старших роботов;
6:    repeat
7:      if Senior ≠ empty then
8:      { Senior.select(j);       //Выбор старшего партнёра для общения j.
9:        send <запрос на общение> to j;          //i хочет общаться с j.
10:       wait until receive <готов> from j;        //i ожидает ответа j.
11:       change( i, j);                       //Общение, обмен укрытиями.
12:       Senior.remove(j);                //Удаление j из списка старших.
        },
13:     if {j : j∈ Junior ∧ Ready_Junior[j]} ≠ empty then
14:     { Junior.select( j: Ready_Junior[j]);
                                  //Выбор ожидающего младшего партнёра j.
15:       change( i, j);                         //Общение, обмен укрытиями.
16:       Junior.remove(j);                //Удаление j из списка младших.
        }
17:   until Senior ≠ empty ∧ Junior ≠ empty;
18: until (NC = 0 ∧ CFR = 2*(n - 1) )
end.
```

(Обратите внимание, что после строки 12 стоит запятая, означающая, что разделённые ею блоки операторов 8-12 и 13-16 могут выполняться последовательно в произвольном порядке, но не параллельно.)

Теорема 2. *Пусть мультиагентная система состоит из $n > 1$ роботов в общем положении, исполняющих алгоритм SMEx2, все роботы имеют взаимнооднозначно назначенные начальные укрытия. Тогда общение между роботами удовлетворяет условию справедливости и беступиковости, а система когда-нибудь*

завершит свою работу, и в этом состоянии не будет пересекающихся путей роботов к их выбранным назначениям.

Доказательство следует из утверждения 1, если показать, что общение между роботами, работающими по алгоритму SMEx2, удовлетворяет условию справедливости и беступиковости. Это свойство обеспечено тем, что приоритеты в отношении возможностей послать запрос на общение гарантируют отсутствие "циклических" запросов и ожиданий ответа, то есть тупиков. ■

Рассмотрим теперь вариант планирования, в котором роботы не имеют приоритетов, но имеют доступ к глобальным часам. В таком случае возможно опосредованное общение агентов. Для назначения времени контакта пара роботов может использовать "шпионский" протокол. Его идея состоит в том, чтобы назначать не конкретное время, а множество времён для встреч (например, "жду тебя каждый чётный вторник в 8:00, каждый нечётный — в 20:00"). Для двух агентов эта схема будет работать, однако в случае, если агентов больше двух, то могут возникнуть тупики. Если хотя бы три агента назначат друг другу свидания из одного и того же (регулярного) множества времён, то снова возможен тупик, как в случае непосредственного общения в замкнутой системе без приоритетов. Агенты могут всегда приходить в одно и то же время на встречу не к тому, кто их ждёт: например, первый агент приходит ко второму, второй к третьему, а третий к первому.

Чтобы тупиков такого сорта не возникало, для нескольких агентов (больше двух) можно предложить следующее. Пусть каждый агент имеет собственное уникальное заранее определённое множество времён для встреч[8]. Поскольку такое множество заранее определено, то это в некотором роде оказывается внешним планировщиком, но так как агент имеет возможность самостоятельно выбирать время для разных встреч (возможно, одно и то же), то "зависимость" агента от внешнего планирования оказывается не очень жёсткой.

Ниже приведена модификация алгоритма SMEx, в которой используется "шпионское" планирование контактов. Пусть $T(i)$ — множество моментов времени, в которые робот i может назначать встречи, а $t(i,j)$ — множество моментов времени из $T(i)$, в которые робот i ждёт встречи с роботом j. Хотя набор таких множеств $T(i)$ заранее определён, робот i в процессе исполнения алгоритма сам назначает время для встречи с j.

```
algorithm SMEx3::
const i : робот;                                  //Собственный номер робота.
var NC, CFR : integer;
var contacts : множество роботов;
var ItoJ[1..n], JtoI[1..n] : множество времён;[9]
begin
1: NC:= (n - 1); CFR:= 0 ;
   текущее назначение := начальное назначение для робота i;
```

[8] Простейший пример такого множества: $T(i) = \{a \times n \times t + i \mid a$ — произвольное натуральное число (константа), t — номер периода, n — число роботов$\}$.

[9] //`ItoJ` — множество времён встреч, назначенных самим роботом своим партнёрам, а `JtoI` — множество времён встреч, назначенных партнёрами данному роботу.

```
2: repeat
3:   if NC > 0 then NC:= (n - 1);
4:   contacts:= множество всех роботов;
5:   repeat
6:     { contacts.select(j)( j: ItoJ[j] = empty);
                    //Выбор партнёра, которому время ещё не назначено.
7:       send <t(i,j)> to j;              //i посылает время встречи j.
8:       ItoJ[j] := t(i,j);            //i записывает время встречи с j.
9:     },
10:    { contacts.select( j: nearest_time(ItoJ[j], JtoI[j]));
                         //Выбор робота j с ближайшим временем встречи.
11:      if come(j)          // если робот j вышел на связь, то происходит
12:      then { change( i, j);              //общение и обмен укрытиями.
13:             contacts.remove(j);}           //Удаляет j из контактов.
       }
14:   until contacts≠empty;
15: until (NC = 0 ∧ CFR = 2*(n - 1) )
end.
```

(Здесь снова запятая в строке 9 разделяет блоки, выполняющиеся последовательно в произвольном порядке, но не параллельно.)

Теорема 3. *Пусть мультиагентная система состоит из $n > 1$ роботов в общем положении, исполняющих алгоритм SMEx3, все роботы имеют взаимно-однозначно назначенные начальные укрытия. Тогда общение между роботами удовлетворяет условию справедливости и беступиковости, а система когда-нибудь завершит свою работу, и в этом состоянии не будет пересекающихся путей роботов к их выбранным назначениям.*

Доказательство аналогично доказательству теоремы 2. Покажем, что общение между роботами, работающими по алгоритму SMEx3, удовлетворяет условию справедливости и беступиковости. Справедливость следует из того, что роботы назначают время встречи каждому из списка контактов. Беступиковость обеспечена уникальностью множеств времён контактов для каждого робота, так как тупики при опосредованном общении возможны только в случае, когда больше, чем двое, агентов пытаются установить контакт в один и тот же момент времени. ■

2. Поиск начального укрытия

Теперь рассмотрим три версии алгоритма, устанавливающих взаимно-однозначное начальное назначение укрытий. Назовём их алгоритмы начального соединения IC (Initial Connection).

Первый алгоритм IC1 очень прост и явно использует приоритеты роботов (их номера). Идея IC1 — это эстафета: первый по порядку робот выбирает ближайшее укрытие, затем передаёт второму роботу право выбрать укрытие, информирует его о занятом им укрытии и потом ждёт, пока все роботы не выберут себе по укрытию;

затем второй робот выбирает ближайшее свободное укрытие, передаёт третьему роботу право выбрать укрытие, информирует его о занятых укрытиях и потом ждёт, пока все роботы не выберут себе по укрытию; и так далее. В конце цепочки роботов последний робот "выбирает" оставшееся укрытие и посылает предыдущему роботу сигнал, что все сделали свой выбор. Мы не будем приводить псевдокод этого алгоритма из-за его идейной простоты.

Второй алгоритм IC2 установления начального назначения сложнее, но менее синхронизован, чем эстафета. Сущность IC2 состоит в том, что каждый робот в произвольное время выбирает какое-нибудь укрытие, но если какой-нибудь младший робот захочет занять это укрытие, то старшему придётся подыскать себе другое. Описание этого алгоритма на псевдокоде выглядит следующим образом.

```
algorithm IC2 ::
const i : робот; //Собственный номер робота.
var k : укрытие;
const Senior = множество всех младших роботов : множество роботов;
const Junior = множество всех старших роботов : множество роботов;
var Shlt_Junior[1..i-1][2] : integer×boolean;
    //Shlt_Junior[j][1] -- номер текущего укрытия для младшего робота j,
                //a Shlt_Junior[j][2] -- флаг состояния выбора робота j:
                 //true -- окончательный выбор, false -- предварительны).
var Done_Senior[i+1..n] : boolean;
    //Done_Senior[j] -- флаг состояния выбора укрытия старшим роботом j:
                            //true -- робот сделал выбор, false -- нет.
begin
1: choose any k in [1..n];          //Робот i выбирает какое-либо укрытие.
2: send <k, false> to all Senior;
                  //Уведомляет старших о предварительном выборе укрытия.
3: repeat
4:      if found k in Shlt_Junior
                       //Если младший робот претендует на это же укрытие,
5:      then { k:= new in [1..n];    //то робот i выбирает другое укрытие
             send <k, false> to all Senior;}
                                  //и сообщает о новом выборе всем старшим.
6: until (all j∈Junior: Shlt_Junior[j][2])       //Младшие сделали выбор.
7: send <k, true> to all Senior;
                //Сообщает всем старшим об окончательном выборе укрытия.
8: send <true> to all Junior;
           //Уведомляет младших роботов, что сделал окончательный выбор.
9: wait until (all j∈Senior: Done_Senior[j]);
                         //Ждёт пока старшие сделают окончательный выбор.
end.
```

Для описания третьего алгоритма IC3 нам понадобится процедура $cede(i, j)$, которая напоминает процедуру $change(i, j)$, но отличается от неё тем, что старший робот уступает младшему конкуренту своё укрытие.

```
procedure cede(i, j)::
```

```
const partner = j : робот;
var NC, CFR : integer;
var k_i, k_j : укрытие;
begin
1: send <NC> to partner; receive <NC партнёра> from partner;
2: send <текущее назначение k_i> to partner;
   receive <назначение партнёра k_j> from partner;
3: if k_i = k_j
4: then{ старший уступает укрытие младшему, и выбирает новое[10];
        NC:= (n - 1); CFR:= 0 }
5: else if NC > 0
6:     then { NC:= (NC - 1); CFR:= 0 }
7:     else if NC = 0
8:         then if NC партнёра = 0
9:             then CFR:= CFR + 1
10:            else { NC:= (n - 1); CFR:= 0 }
end.
```

Алгоритм IC3 получается из алгоритма SMEx2 либо SMEx3 в результате замены "начального назначения" на "ближайшее укрытие" и процедуры $change(i, j)$ — процедурой $cede(i, j)$.

Теорема 4. *Пусть мультиагентная система состоит из $n > 1$ роботов, исполняющих алгоритм IC1, или IC2, или IC3. Тогда система когда-нибудь придёт в стабильное состояние, и в этом состоянии все роботы будут иметь разные назначения*[11].

Доказательство корректности алгоритмов IC1 и IC2 очевидно, а для IC3 проходит аналогично доказательству корректности алгоритмов SMEx2 и SMEx3 в теоремах 2 и 3. ■

Выглядит привлекательной идея комбинации алгоритмов SMEx2 и IC3. Пусть в процедуре $contract(i, j)$ роботы i и j обмениваются укрытиями, как в процедуре $change(i, j)$, или старший робот уступает укрытие младшему, как в процедуре $cede(i, j)$, но что именно происходит — зависит от ситуации: уменьшается ли суммарная длина маршрутов роботов или роботы конкурируют из-за укрытия. Но, к сожалению, получившийся алгоритм не будет решать даже задачу Дейкстры, что показывает следующий контрпример, в котором этот алгоритм зацикливается.

Пусть в некоторый момент у нас есть ситуация, приведённая на рис. 1.a: роботы 1 и 2 выбрали укрытие 1, роботы 3 и 4 — укрытие 2. Пусть робот 2 уступает укрытие 1 роботу 1 и выбирает себе новое укрытие 2, а робот 4 уступает укрытие 2 роботу 3 и выбирает укрытие 1 (см. рис. 1.b). Тогда (напомним, что роботы действуют асинхронно и следующий сценарий вполне возможен) роботы 2 и 4 решают разрешить свой конфликт и обменяться укрытиями 1 и 2. В результате чего мы снова получим ситуацию 1.a.

[10] из ранее не использованных этим роботом.

[11] т. е. отношение робот-укрытие будет взаимно-однозначным.

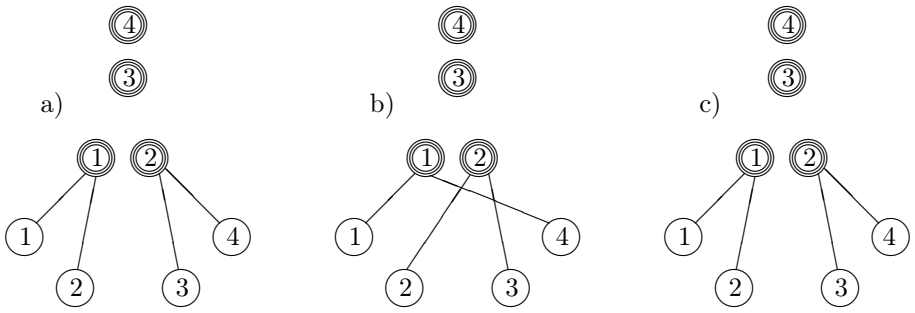

Рис. 1. Зацикливание комбинации SMEx2 и IC3

4. Заключение

Мы представили и доказали корректность ряда мультиагентных алгоритмов для задачи о Роботах на Марсе. Каждый из приведённых выше алгоритмов является волновым [12], поскольку удовлетворяет следующим трём свойствам:

1. Завершение. Действительно, все варианты SMEx'а завершаются в силу убывания суммы расстояний до укрытий, а алгоритмы начального выбора завершатся в силу конечности и упорядоченности роботов.

2. Решение. В алгоритмах SMEx событием решения является осознание того, что можно двигаться к укрытиям.

3. Зависимость. Поскольку во всех алгоритмах каждый из роботов общается с остальными, и это общение влияет на дальнейшие шаги, то все эти события в жизни каждого робота влияют на событие решения.

Однако пока остаётся открытым вопрос о сложности наших алгоритмов. В алгоритме SMEx для каждого робота в каждом его раунде, кроме последних трёх (один процессный и два пост-процессных), система выполняет (хотя бы) один FLIP в промежутке между первыми переговорами робота в этом раунде и первыми переговорами робота в следующем. (Иначе этот раунд должен быть последним процессным раундом перед парой последних пост-процессных). Следовательно, число раундов для каждого робота равно $O(T(n))$, где $T(n)$ — сложность эвристического алгоритма Дейкстры [5, 6] (т. е. самое большое число FLIPов). К сожалению, нам неизвестна верхняя граница для $T(n)$, меньшая $O(n!)$. Если считать, что числовые вычисления, приём и передача сообщений осуществляются за постоянное время, то сложность каждого раунда зависит от сложности справедливого планирования контактов. В алгоритме SMEx2 сложность справедливого планировщика оказывается равной $O(n^2)$, поскольку каждый робот может ждать отклика старшего робота $O(n)$ шагов. Таким образом, общая временная сложность алгоритма SMEx2 равна $O(n^2 \times T(n))$. Однако сложность относительно количества контактов для алгоритма SMEx с планировщиком и без остаётся одинаковой, а именно $O(n \times T(n))$. Такая же временная и "коммуникативная" сложность у третьего алгоритма установления

первоначального соединения IC3. Алгоритм IC1 имеет линейную временную сложность $O(n)$ и "контактную" сложность $O(1)$. Алгоритм IC2 более трудоёмкий, чем IC1, но, видимо, проще, чем IC3: его временная сложность $O(n^2)$, поскольку $O(n)$ младших могут заставить робота извещать $O(n)$ старших о перемене укрытия, и такая же "контактная" сложность $O(n^2)$ по той же причине.

В целом мы не можем сказать, что мы полностью удовлетворены вышеприведёнными алгоритмами, поскольку имеется ряд вопросов, требующих более подробного исследования.

Первый вопрос связан с использованием предположения справедливости контактов в алгоритме SMEx и использовании предопределённых приоритетов в алгоритме SMEx2. Использование предположения о справедливости нехорошо тем, что его нельзя обеспечить без введения в систему внешнего агента-планировщика или системы приоритетов агентов. Кроме того, любая система приоритетов агентов так же приводит к тому, что можно выделить самого "главного" агента с наивысшим приоритетом. Это означает, что в обоих случаях система агентов не "плоская", а имеет структуру со специальным агентом (организатором или главным). Если таковой имеется, то почему бы не делегировать индивидуальные права выбора этому специальному агенту, который эффективно сможет распределить укрытия среди всех роботов? Таким образом возникает естественный вопрос: можно ли разработать мультиагентный алгоритм, решающий задачу о Роботах на Марсе без использования предположения о справедливости контактов в алгоритме SMEx и без использования предопределённых приоритетов?

Следующий вопрос связан с обобщением мультиагентных алгоритмов для MRP на оптимизационные задачи теории графов, и, в частности, на задачу о назначениях. Очевидно, задача Дейкстры — это частный случай задачи о назначениях, где вес ребра — это евклидово расстояние между чёрными и белыми точками. В следующей таблице приведена матрица весов для двудольного графа, состоящего из вершин $\{C_1, C_2, C_3\}$ для потребителей и $\{P_1, P_2, P_3\}$ для поставщиков.

	P_1	P_2	P_3
C_1	0	1	3
C_2	3	0	1
C_3	1	3	0

Пусть есть следующее начальное соответствие (строка 1 алгоритма SMEx) продавцов потребителям: $C_1 \leftarrow P_2$, $C_2 \leftarrow P_3$ и $C_3 \leftarrow P_1$. При этих условиях FLIP (строка 14) не может работать, и, следовательно, роботы (в этом случае потребители) завершат алгоритм с неизменившимся соответствием, но оно не оптимально. Так что остаётся невыясненным, могут ли роботы решить задачу о назначениях мультиагентным способом. Мы не знаем ни такого алгоритма, ни утверждения, доказывающего, что это невозможно.

Третий вопрос связан с задачами планирования перемещений. В парадигме искусственного интеллекта "объекты" (т.е. роботы и укрытия в MRP) никогда не представляются геометрическими точками, а скорее представлены "безопасными областями" (как правило, сферами с положительным радиусом в какой-то метрике). При этих условиях "прямой путь" — это не совсем прямая линия, а скорее цилиндр,

“множество безопасных путей” — не набор прямых отрезков без пересечений, а множество непересекающихся цилиндров. В этом разница между MRP и задачей планирования перемещений в ИИ. Мы хотели бы исследовать более реалистичную версию MRP, в которой все объекты представлены безопасными областями. Мы также планируем изучить полезность нашего подхода и алгоритмов для задачи избегания столкновений в воздушном пространстве.

Четвёртое возможное направление исследований — это верификация представленных алгоритмов посредством симуляции и проверки на модели свойств безопасности (“роботы никогда не будут сталкиваться”) и живости (“роботы когда-нибудь двинутся с места”) с помощью инструментов SPIN и SMV. Сейчас это является нашим текущим исследованием. К настоящему моменту мы проверили алгоритм SMEx2 в системе SPIN для 5 агентов в режиме симуляции и для 3 агентов в режиме верификации.

Список литературы

1. **Fagin R., Halpern J.Y., Moses Y., Vardi M.Y.** Reasoning about Knowledge. London: MIT Press, 1995.

2. **Garanina N.O., Shilov N.V., and Konyaev L.E.** Can Robots Solve the Assignment Problem? // Proc. Int. Workshop on Concurrency, Specification, and Programming.Kraków-Przegorzały, Poland, 2009. V. 1. P. 154–163.

3. **LaValle S.M.** Planning Algorithms. Cambridge University Press, 2006.

4. **Shilov N.V., Garanina N.O.** Modal Logics for reasoning about Multiagent Systems // Encyclopedia of Artificial Intelligence. J.R. Rabucal, J. Dorado, A.P. Sierra, editors. Information Science Reference, 2008. P. 1089–1094.

5. **Shilov N.V., Shilova S.O.** Etude on theme of Dijkstra // ACM SIGACT News. 2004. V. 35(3). P. 102–108.

6. **Shilov N.V., Shilova S.O.** On Mathematical Contents of Computer Science Contests // Proc. the 1st KAIST International Symposium on Enhancing University Mathematics Teaching. Daejeon, Korea, 2005. P. 223–233.

7. **Wooldridge M.** An Introduction to Multiagent Systems. John Willey & Sons Ltd, 2002.

8. **Болтянский В.Г., Солтан П.С.** Комбинаторная геометрия различных классов выпуклых множеств. Кишинев: Штиинца, 1978.

9. **Бугайченко Д.Ю., Соловьев И.П.** Абстрактная архитектура интеллектуального агента и методы её реализации // Системное программирование. СПб, 2005. С. 36–66.

10. **Емеличев В.А., Мельников О.И., Сарванов В.И., Тышкевич Р.И.** Лекции по теории графов. М.: Наука, 1990.

11. **Кормен Т. Х., Лейзерсон И. Ч., Ривест Р. Л., Штайн К.** Алгоритмы: построение и анализ. М.: Вильямс, 2006. 1296 с.

12. **Тель Ж.** Введение в распределенные алгоритмы. М.: МЦНМО, 2009.

13. **Хадвигер Г., Дебруннер Г.** Комбинаторная геометрия плоскости. М.: Наука, 1965.

14. **Черемисинов Д., Черемисинова Л.** Подход к программированию агентов в мультиагентных системах // International Book Series "Information Science and Computing". 2008. № 4. С. 141–147.

Mars Robot Puzzle (a Multiagent Approach to the Dijkstra Problem)

Bodin E.V., Garanina N.O., Shilov N.V.

Keywords: multiagent system, distributed algorithm, assignment problem, path-planning problem

We continue our study of multiagent algorithms for a problem that we call the Mars Robot Puzzle. This problem could be considered as a special case of a graph-theoretic problem (Discrete Mathematics), as a combinatorial geometry problem (Computer Science), or as a very special case of a path-planning problem (Artificial Intelligence). Our algorithms grew up from a local search (heuristic) solution of the problem suggested by E.W. Dijkstra. In the paper we present a series of new multiagent algorithms solving the problem, prove (manually) their correctness, model check some of these algorithms, and discuss further research topics. All our algorithms are multiagent in contrast to "centralized" graph and combinatorial algoritms; correctness of our algorithms is formally proven, while the testing is used for validation of path-planning algorithms.

Сведения об авторах:
Бодин Евгений Викторович,
Институт систем информатики им. А.П. Ершова СО РАН,
научный сотрудник;
Гаранина Наталья Олеговна,
Институт систем информатики им. А.П. Ершова СО РАН,
канд. физ.-мат. наук, научный сотрудник;
Шилов Николай Вячеславович,
Институт систем информатики им. А.П. Ершова СО РАН,
канд. физ.-мат. наук, старший научный сотрудник;
Новосибирский государственный университет,
Новосибирский государственный технический университет, доцент