УДК 519.711
ББК 22.18

ОРГАНИЗАЦИЯ СТРОЯ АГЕНТОВ С ПОМОЩЬЮ КЛЕТОЧНОГО АВТОМАТА

Кузнецов А. В.[1],
(*Воронежский государственный университет, Воронеж*)

Рассматривается алгоритм распределенной организации заданного графом строя агентов и численная симуляция такого алгоритма. Описывается клеточный автомат, моделирующий движение агентов, и исследуются его особенности в связи с типом ландшафта, по которому перемещаются агенты. Указанный клеточный автомат имеет два представления – одномерное и двумерное.

Ключевые слова: клеточный автомат, автономные агенты, рефлексивные агенты, управление строем агентов.

1. Введение

Статья является итогом работ автора по исследованию поиска оптимального пути группами автономных агентов по ландшафтам разной степени сложности в рамках клеточно-автоматного подхода. В работах [1, 4, 14] был сконструирован клеточный автомат, позволяющий моделировать разные виды взаимодействия между иерархически организованными агентами, движущимися по ландшафту переменной сложности, и исследованы некоторые характеристики этих взаимодействий. Эти работы возникли в результате переосмысления автором моделей, предложенных в [6] и [12]. Однако, в отличие от [6], агенты двигались иерархически вложенными друг в друга роями по местности, состоящей из препятствий разной проходимости, причем

[1] *Александр Владимирович Кузнецов, кандидат физико-математических наук, доцент* (*avkuz@bk.ru*).

способ их передвижения отличался гораздо большей самостоятельностью, чем в [12], где агенты двигались, в сущности, по набору опорных точек.

В данной работе анализируется эффективность предложенного клеточного автомата для поиска оптимального маршрута в ландшафтах разного вида. Также предлагается алгоритм организации строя агентов с учетом необходимости обхода препятствий. Отметим, что автомат, предложенный автором в цитированных выше работах, уже предполагал возможность самоорганизации агентов в иерархические рои с помощью аналога социального потенциала, предложенного в [13]. В отличие от работ [7, 16, 18, 19], посвященных построению строя роботов с использованием одномерного клеточного автомата, используется автомат, имеющий одновременно и одномерное, и двумерное представление в зависимости от типа решаемой задачи. Также, в отличие от упомянутых работ, рассматривается строй, порожденный графом, а не набором математических функций, а в качестве окрестностей в одномерном представлении клеточного автомата рассматриваются окрестности, порожденные геодезическим расстоянием на графе строя. Наконец, агенты в настоящей статье не обмениваются координатами для поддержания строя и не безусловно следуют за лидером (что может быть крайне невыгодным на пересеченной местности), а стараются поддерживать строй, выбирая при этом направление движения самостоятельно, исходя из своих предположений о передвижении других агентов в наблюдаемой ими окружающей среде.

2. Непрерывная постановка задачи

В работе далее будут использоваться следующие стандартные обозначения: $\mathbb{R}$, $\mathbb{Z}$ и $\mathbb{N}$ – множества вещественных, целых и натуральных чисел соответственно, $\mathbb{R}_{\geqslant 0}$, $\mathbb{Z}_{\geqslant 0}$ – множества неотрицательных вещественных и целых чисел, $\mathbb{R}^n$ – пространство n-мерных векторов с вещественными координатами, $\|\cdot\|$ – произвольная (например, евклидова) норма в $\mathbb{R}^n$.

Пусть заданы

1. Область $\Omega \subset \mathbb{R}^2$.

2. Множество агентов $Ag = \{ag_k | k = \overline{1,n}\}$.

3. Точки начала и конца движения агента $ag_k \in Ag$ A_k, $B_k \in \Omega$, $k = \overline{1,n}$

4. Функция проходимости $\mathrm{u}^c : [0,T] \times \Omega \to \mathbb{R}_{\geqslant 0}$, $\mathrm{u}^c(t,r)$ – это максимально возможная норма скорости в момент времени $t \in [0,T]$ в точке $r \in \Omega$.

5. Желаемый граф строя $\Phi = (Ag, E, \mathrm{coo}^c_\Phi)$, $E \subseteq Ag^2$ – множество ребер, $\mathrm{coo}^c_\Phi : [0,T] \times Ag \to \Omega$ – функция, дающая желаемые для сохранения строя координаты агента в Ω.

6. Фактический граф строя $\Gamma = (Ag, E, \mathrm{coo}^c_\Gamma)$, $\mathrm{coo}^c_\Gamma : [0,T] \times Ag \to \Omega$ – функция, дающая реальные координаты агента в Ω.

Траектория k-го агента $r_k : [0,T] \to \Omega$ отвечает условиям

$$\|\dot r_k(t)\| = \mathrm{u}^c(t, r_k(t)), \tag{1}$$

$$r_k(0) = A_k, \quad r_k(t) = B_k, \quad t \geqslant T_k, \quad T_k \leqslant T, \tag{2}$$

$$r_k(t) \neq r_l(t), \quad k \neq l, t \in [0,T], \tag{3}$$

где

$$T_k(r_k) = \min\{t \in \mathbb{R}_{\geqslant 0} | r_k(t) = B_k\},$$

$\|\dot r_k(t)\|$ – норма скорости k-го агента в момент времени t,
$\mathcal{Y} \subset C([0,T];\Omega)$ – множество допустимых траекторий. В явном виде выписать функционалы T_k в общем случае невозможно. Отметим, что уравнение (1) означает, что агент движется в каждой точке области с максимальной (по норме) возможной в этот момент времени скоростью.

Необходимо найти такие траектории агентов, чтоб время $T = \max_{k=\overline{1,n}} T_k$ было минимальным, т.е. надо минимизировать все функционалы T_k, сопоставляющие траектории r_k агента ag_k продолжительность прохождения по этой траектории. Более формально, нужно решить задачу минимизации

$$T(r_1, \ldots, r_n) = \max_{k=\overline{1,n}} T_k(r_k) \to \min, \quad r_k \in \mathcal{Y}, k = \overline{1,n}. \tag{4}$$

Также необходимо, чтоб в любой момент времени $t \in [0,T]$ отличие графов Φ и Γ было бы минимально:

$$\mathrm{dist}(\Gamma(t), \Phi(t)) \to \min. \tag{5}$$

Формальное описание функции несходства графов dist будет дано ниже в разделе 6. Таким образом, необходимо найти траектории $r_k \in \mathcal{Y}, k = \overline{1,n}$, являющиеся решениями задачи оптимизации с двумя целевыми функциями (4)–(5).

3. Дискретная модель

Для поиска приближенных решений задачи оптимизации, приведенной в предыдущем параграфе, будет использоваться клеточный автомат, или, в отечественной терминологии, однородная структура со входами и выходами [2, с. 159].

Определение 1. *Однородной структурой со входами и выходами называется набор* $(\mathbb{Z}^k, \mathscr{W}, V)$, *где* $\mathscr{W}$ *– конечный автомат с* n *основными и* p *боковыми входными каналами,* m *основными и* p *боковыми выходными каналами,* l *состояниями, помещенный в каждом узле целочисленной решетки* $\mathbb{Z}^k$ *и задаваемый системой уравнений:*

$$\begin{cases} y(t) = \varphi_1(x(t), b(t), \mathfrak{s}(t)), \\ a(t) = \varphi_2(\mathfrak{s}(t)), \\ \mathfrak{s}(t+1) = \varphi(x(t), b(t), \mathfrak{s}(t)), \end{cases}$$

$$\varphi_1 : \mathscr{E}^n \times \mathscr{E}^p \times \mathscr{S} \to \mathscr{E}^m,$$

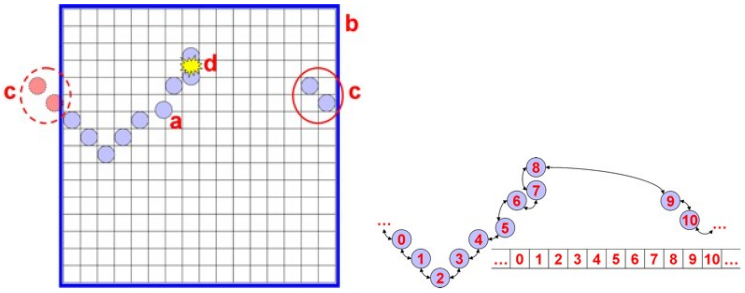
$$\varphi_2 : \mathscr{S} \to \mathscr{E}^p,$$

$$\varphi : \mathscr{E}^n \times \mathscr{E}^p \times \mathscr{S} \to \mathscr{E}^m,$$

где $x = (x_1, \ldots, x_n)$, $b = (b_1, \ldots, b_p)$, $a = (a_1, \ldots, a_p)$, $y = (y_1, \ldots, y_m)$ *обозначают соответственно основной вход, боковой вход, боковой выход и основной выход,* $\mathfrak{s}$ *– состояние автомата,* $t = \overline{0,N}$ *– такт дискретного времени,* $\mathscr{E}$ *– входной и выходной алфавит,* $\mathscr{S}$ *– множество состояний автомата* $\mathscr{W}$, $V = (\alpha_1, \ldots, \alpha_p)$, $\alpha_i \in \mathbb{Z}^k$, $i = \overline{1,p}$, *– шаблон соседства: для автомата в узле* α *каждый* $\alpha_i \in V$ *определяет конечный автомат с координатами* $\alpha + \alpha_i$, *с* i*-м боковым выходным каналом которого соединен* i*-й боковой входной канал автомата в узле* α.

В клеточных автоматах, моделирующих движение агентов, конечному автомату $\mathscr{W}$ будет соответствовать ячейка «местности», содержащая агента, или пустая ячейка, боковым каналам будут соответствовать связи между ячейками, задающие закономерности перемещения агентов между ячейками, а основным каналам – например, модели каналов связи между агентами.

Клеточные автоматы, описывающие движение автономных агентов, можно условно поделить на два типа. Первый из них – двухмерный клеточный автомат (так называемый *world-space cellular automaton*, рис. 1a [17]), например, с правилами, близкими к *Game of Life* Конвея, приведенный в работе [13] для моделирования образования роя агентов.



(a) (b)

Рис. 1: *World-space cellular automaton и robot-space cellular automaton*

Второй тип – это одномерный клеточный автомат (так называемый *robot-space cellular automaton*), каждая клетка которого содержит координаты агента на плоскости, его реальное и желаемое взаимное расположение с соседними агентами (см. рис. 1b [19]).

Такие автоматы применяются в работах для построения строя реальных роботов и для компьютерного моделирования данного процесса, например в [19].

Автор в своих предыдущих работах подробно описывал конструкцию клеточного автомата (далее – КА), используемого им

для моделирования движения агентов. КА, предлагаемый в настоящей работе, незначительно отличается от раннее описанных, поэтому приведем его краткую характеристику. Пусть заданы

1. Замощение Ω_h области $\Omega \subset \mathbb{R}^2$ (для простоты можно считать, что это правильный квадратный паркет $\Omega_h = \{\omega_{ij} | i, j \in Q_h \subset \mathbb{Z}\}$ и не различать клетку ω_{ij} и ее координаты (i, j)).

2. Взаимно-однозначная функция получения уникального идентификатора (УИд) агента $\mathrm{uid} : Ag \to \mathbb{N}$.

3. Множество возможных направлений движения агентов $\mathcal{D} = \{(i, j) | i, j = \overline{-1, 1}\}$.

4. Дискретная функция непроходимости $\mathrm{u} : \mathbb{Z}_{\geqslant 0} \times \Omega_h \to \mathbb{Z}$, устанавливающая, сколько тактов функционирования КА необходимо для преодоления клетки (i, j). При этом предполагается, что для совершенно непроходимой в момент времени t клетки (i, j) $\mathrm{u}(t, (i, j)) = +\infty$. Эта функция получается, по своей сути, дискретизацией функции проходимости, например:

$$u_{ij}(t) = \mathrm{u}(t, (i, j)) = \left\lfloor \frac{\max_{r \in \Omega} \mathrm{u}^c(\tau t, r)}{\min_{r \in \omega_{ij}} \mathrm{u}^c(\tau t, r)} \right\rfloor,$$

где $\lfloor x \rfloor$ означает целую часть x, а τ – продолжительность одного такта.

5. Дискретные функции, дающие желаемые для сохранения строя, предусмотренного графом Ф, координаты агента в текущий момент дискретного времени $\mathrm{coo}_\Phi : \mathbb{Z}_{\geqslant 0} \times Ag \to \Omega_h$ и реальные координаты агента $\mathrm{coo}_\Gamma : \mathbb{Z}_{\geqslant 0} \times Ag \to \Omega_h$.

6. Шаблон формации $\Phi_h(ag) = \{(i, j, agId) | (i, j) \in \mathbb{Z}^2, agId \in \mathrm{uid}(Ag) \cup \{0\}\}$, содержащий относительные координаты соседей агента по строю (строгое определение соседства дано в разделе 6), причем $agId = 0$, если нет требований к тому, какой именно сосед должен находиться, и $agId > 0$, если в клетке с координатами $\mathrm{coo}(t, ag) + (i, j)$ в момент времени t должен находиться именно сосед с УИд $agId$. Таким образом, для элемента $(i, j, agId) \in \Phi_h(ag)$, $agId \neq 0$, справедливо соотношение

$$(i, j) + \mathrm{coo}_\Phi(ag) = \mathrm{coo}_\Phi(ag'), \quad \mathrm{uid}(ag') = agId.$$

Пусть $r_h \subseteq \Omega_h$ – маршрут (на языке перечислительной комбинаторики – решеточный путь), начальная клетка маршрута $r_h[1] = (i,j) + d_1$, $d_1 \in \mathcal{D}$, k-я клетка маршрута $r_h[k] = r_h[k-1] + d_k$, $d_k \in \mathcal{D}$, $k > 1$. Обозначим $(k-1)$-ю клетку маршрута r_h с координатами (i', j'), в которой агент находится в момент времени t, как $r_h(t)$. Тогда дискретным аналогом уравнения (1), определяющего движение агента, будет уравнение

$$\|r_h(t+k) - r_h(t)\| = \|d_k\| \left\lfloor \frac{k}{\|d_k\| u_{i'j'}(t)} \right\rfloor, \quad k \leqslant \|d_k\| u_{i'j'}(t). \tag{6}$$

Детальное формальное определение функций локального такового функционирования описываемого КА (без задания весов, необходимых для поддержания строя) приведено в работе [4]. Вкратце, агент на каждом такте выбирает случайный маршрут движения, удовлетворяющий уравнению (6) и отвечающий ряду других ограничений (например, два агента не могут одновременно находиться в одной клетке). Распределение случайного выбора маршрута таково, что вероятность выбрать минимизирующий нижеприведенные функционалы (7), (9) или (10) маршрут максимальна.

В связи с особенностями реализации, КА должен быть представим как в виде *robot-space cellular automaton*, так и в виде *world-space cellular automaton*, так как первый тип автомата удобен для построения строя, а второй – для работы алгоритма поиска оптимального маршрута и для построения тестовых ландшафтов. Поэтому агент (как ячейка *robot-space cellular automaton*) $ag \in Ag$ в момент времени $t \in \mathbb{Z}_{\geqslant 0}$ представляет собой объект вида

$$ag = \{selfId, leadId, lead, w, \mathcal{D}', trgtAct, trgtTmp, formTmpl, @cell\},$$

где $selfId = \mathrm{uid}(ag)$ – УИд агента; $leadId \in \mathbb{N}$ – УИд лидера данного агента; $lead \in \{true, false\}$ указывает на то, что агент является ведущим; w – количество тактов, которое агент уже простоял в текущей клетке; $\mathcal{D}'$ – упорядоченный по убыванию желательности список возможных направлений из $\mathcal{D}$ для

агента на следующий такт; $trgtAct \in \Omega_h$ – текущая целевая клетка агента; $trgtTmp \in \Omega_h$ – временная целевая клетка агента; $formTmpl = \Phi_h(ag)$ – шаблон строя; $@cell$ – указатель на ячейку, в которой находится агент ag в момент времени t. Ячейка $cell$, соответствующая клетке $(i, j) \in \Omega_h$ для *world-space cellular automaton*, представляет собой

$$cell = \{u, \mathrm{coo}, @ag\},$$

где $u = \mathrm{u}(t, \mathrm{coo})$ – значение функции непроходимости клетки (т.е. минимальное количество тактов, нужное для преодоления клетки); $\mathrm{coo} = (i, j)$; $@ag$ – указатель на агента, находящегося в клетке (i, j) в момент времени t или нуль, если в (i, j) в момент времени t нет агента. Будем обозначать агента, на которого указывает $@ag$ как ag_{ij} и писать

$$ag_{ij} = \{selfId_{ij}, leadId_{ij}, w_{ij}, \mathcal{D}_{ij}, trgtAct_{ij}, trgtTmp_{ij}, formTmpl_{ij}, @cell_{ij}\}.$$

Если $@ag_{ij} = 0$, то для единообразия будем считать, что

$$ag_{ij} = \{0, 0, 0, 0, 0, 0, 0, 0\}.$$

Также будем обозначать ячейку, на которую ссылается $@cell_k$, как $cell_k$ и писать $cell_k = (u_k, \mathrm{coo}_k, @ag_k)$. Помимо этого будем обозначать, что $cell_{ij} = \{u_{ij}, (i, j), @ag_{ij}\}$.

В результате использования ячеек, содержащих указатели, возможно при реализации КА сэкономить объем памяти, нужной для хранения конфигурации КА, так как количество агентов обычно существенно меньше количества клеток и при хранении состояния агента в каждой клетке *world-space cellular automaton* потребовалось бы хранение большого количества нулевых значений. Также возможно увеличить быстродействие КА, так как при перемещении агента в новую клетку не нужно копировать его состояние целиком, а нужно лишь изменить значение пары указателей, и при доступе к ячейке $cell$ уже сразу известен адрес

$@ag$ и наоборот. В псевдокоде алгоритмов мы также будем обозначать поля объектов $cell$ и ag принятым в языках программирования способом, т.е. в формате «имя_объекта.имя_поля», опуская числовые индексы.

Будем обозначать как $V_r(i,j)$ окрестность клетки (i,j) радиуса r

$$V_r(i,j) = \{c \in \mathbb{Z}^2 | \|c - (i,j)\| \leqslant r\}.$$

Определим функцию

$$\theta(x) = \begin{cases} 0, x \leqslant 0, \\ 1, x > 0. \end{cases}$$

Введем дискретный аналог функционалов T_k:

$$T_h(r_h) = \alpha_0 \sum_{(i,j)\in r_h} \|d_{ij}\| u_{ij} + \alpha_1 \sum_{(i,j)\in r_h} \theta(agId_{ij}), \tag{7}$$

где α_0, $\alpha_1 \geqslant 0$ – параметры модели, отвечающие за то, предпочитает ли агент менее непроходимые или же менее населенные другими агентами маршруты, $d_{ij} = d_k$, если $r_h[k] = (i,j)$.

Обозначим множество всех маршрутов $\mathcal{M}_d(i,j;i_r,j_r)$ агента ag_{ij}, соединяющих (i,j) и (i_r,j_r), причем (i,j) не входит в маршрут и если $(i_r,j_r) = trgtAct_{ij}$, то (i_r,j_r) не входит в маршрут, $d = r_h[1] - (i,j)$.

Общая идея состоит в том, чтоб каждый агент в клетке (i,j) искал оптимальный по времени маршрут в некоторой своей окрестности $V_r(i,j)$, а потом собирал из таких локально оптимальных маршрутов квазиоптимальный маршрут. Поскольку непроходимость области Ω_h может меняться со временем, а также не быть полностью разведанной и проходы могут блокироваться другими агентами, то поиск глобального оптимального маршрута может быть невозможен или лишен смысла. Тактовое функционирование *world-space cellular automaton*, в котором агенты ищут кратчайший маршрут до точки назначения, описывается с помощью алгоритма 1, в который входят алгоритмы 3 и 4, приведенные в Приложении.

Следует подчеркнуть, что для агента движение в сторону скорейшего маршрута наиболее вероятно, но не обязательно. Точнее, агент в клетке (i, j) назначает каждому направлению $d \in \mathcal{D}$ вес $weights[d]$:

(8) $$weights[d] = \min_{r_h \in \mathcal{M}_d(i,j;i_r,j_r)} T_h(r_h).$$

Далее $\mathcal{D}$ перемешивается в соответствии с весами из $weights$. Процедура взвешенного случайного перемешивания нетривиальна: используется предложенный в [9] алгоритм. Таким образом, получается упорядоченный список направлений. Если движение в первом по данному списку направлении маршрута невозможно (например, путь заблокирован), то агент выбирает следующий по времени прохождения маршрут и т.д. Это требуется для разрешения коллизий между агентами и для симуляции элемента случайности в передвижении агентов.

Алгоритм 1 (Поиска квазиоптимального по времени пути)**.**

1: **for all** $ag \in Ag$ **do**
2: $weights =$ sortDirections(ag) ▷ Сортировка направлений, алгоритм 3
3: Случайно перемешать $ag.\mathcal{D}'$ в соответствии с весами из $weights$
4: **end for**
5: **for all** $ag' \in Ag$ **do**
6: Actualize(ag') ▷ Актуализация, алгоритм 4
7: **end for**
8: t = t+1

Можно обратить внимание, что в алгоритме разделен поиск наилучшего (в смысле кратчайшего по времени маршрута) направления движения агентов (алгоритм 3) и действительное обновление конфигурации КА. Это сделано для двух целей: для возможности вставки между этими блоками алгоритма, отвечающего за построение строя, и для параллельного вычисления наилучших направлений перемещения. При этом поиск маршрута является самым трудным вычислительно моментом работы КА и занимает, при вычислении алгоритмом Дейкстры, минимум

$O((2r+1)^4)$ времени каждый ход для каждого агента, так как по сути кратчайший путь в окрестности $V_r(i,j)$ – это кратчайший путь на графе с $(2r+1)^2$ вершинами для всех агентов.

Поиск локально оптимального маршрута может повторяться каждый такт времени, как в предыдущих работах автора, альтернативно агент ag_{ij} может рассчитать, например, алгоритмом Дейкстры или A^* локально оптимальный в $V_r(i,j)$ маршрут, следовать по нему и производить повторный пересчет лишь при достижении (i_r, j_r) или при обнаружении по ходу маршрута непроходимых препятствий или скоплений других агентов. Очевидно, в этом случае список предпочитаемых направлений движения $ag_{ij}.\mathcal{D}'$ заменяется на цепной путь $d_1, d_2, \ldots d_p$, соединяющий (i,j) и (i_r, j_r). Выбор между этими методами зависит от скорости изменения ландшафта. Также возможно производить поиск пути с учетом знаний агента о местности и предыдущего опыта агента, как это показано в [14]. В этом случае, помимо слоя разделяемой всеми агентами реальности $Cell = \{cell\}$, для каждого агента задается свой слой «субъективной реальности». В данный слой, который также описывается как множество ячеек, по мере поступления агент сохраняет информацию об уже посещенных клетках, количестве и времени посещений клеток. При поиске локально оптимального пути необходимо, например, увеличивать «веса» переходов в уже посещенные клетки, для чего рассматривать вместо функционала (7) функционал

$$\tilde{T}_h(r_h) = T_h(r_h) + \alpha_2 \sum_{(i,j)\in r_h} visit_{ij}, \quad \alpha_2 \geqslant 0, \tag{9}$$

где $visit_{ij}$ – количество предыдущих посещений клетки (i,j), аналогично [14], моделировать конфликт агентов, как в [4] и т.д. Если агенту известно расстояние от coo_n – ближайшей к клетке назначения клетке из множества $r_h \cap V_r(i,j)$ до точки назначения coo_{target}, то для повышения качества алгоритма поиска можно рассматривать вместо (9) функционал

$$\hat{T}_h(r_h) = \tilde{T}_h(r_h) + \alpha_3 \|\mathrm{coo}_n - \mathrm{coo}_{target}\|, \quad \alpha_3 \geqslant 0. \tag{10}$$

3.1. Автоматы и теория категорий

Отметим, что агенты, движущиеся по решетке, могут соответствовать не только реальным роботам, перемещающимся по пересеченной местности, но и моделировать некоторый процесс неклассических вычислений. Можно рассматривать конфигурации *world-space cellular automaton* $Cell = \{cell\}$ как категорию, в которой объектами являются ячейки, а морфизмами – решеточные пути между ячейками. Аналогично можно рассмотреть конфигурации *robot-space cellular automaton* Ag как категорию, в которой объектами являются состояния агентов ag_i в момент времени t, а морфизмами – отображения $\gamma_{t_0} : ag_i(t) \mapsto ag_i(t + t_0)$.

Операция @ порождает пару функторов между этими категориями:

$$\mathcal{F}_1 : ag \mapsto \{u, \mathrm{coo}, @ag\},$$
$$\mathcal{F}_2 : cell \mapsto \{selfId, leadId, lead, w, \mathcal{D}', trgtAct, trgtTmp, formTmpl, @cell\}.$$

Далее, в рамках теории категорий всякую клетку $(i, j) \in \Omega_h$ можно интерпретировать как *итератор*. В этом случае $ag \in Ag$ является *коитератором* [11], который порождает поток клеток, через которые он проходит.

4. Случайные ландшафты

Назовем ландшафтом в момент времени t множество непроходимостей клеток определенного подмножества замощения $\mathcal{L}(\Omega_h, l) = \{u_{ij} | u_{ij} = \mathrm{u}(t, \omega_{ij}), \omega_{ij} \in \Omega_h\}$, такого что u принимает на $\{\omega_{ij} | i = \overline{1, n}, j = \overline{1, m}\}$ не более l значений, причем к классу i принадлежит N_i клеток, т.е. $\sum_{i=1}^{l} N_i = M$, $M = |\mathcal{L}(\Omega_h, l)|$. Введем следующие характеристики ландшафта, известные из ландшафтной экологии (см., например, [8, 10, 20]).

Определение 2. Конфигурационная энтропия ландшафта $\mathcal{L}(\Omega_h, l)$ *определяется как*

$$S(\mathcal{L}(\Omega_h, l)) = -\sum_{i=1}^{l} \frac{N_i}{M} \ln \frac{N_i}{M}.$$

Определение 3. Total Edge (TE) определяется как общее количество соприкосновений сторон клеток в $\mathcal{L}(\Omega_h, l)$*, принадлежащих к разным классам. Будем далее обозначать Total Edge ландшафта* $\mathcal{L}(\Omega_h, l)$ *как* $TE(\mathcal{L}(\Omega_h, l))$.

Определение 4. Total Edge Density (TED) для ландшафта $\mathcal{L}_{n\times m}(l)$ *определяется как отношение* $TE(\mathcal{L}(\Omega_h, l))$ *к общему количеству клеток* $\mathcal{L}(\Omega_h, l)$*:*

$$TED(\mathcal{L}(\Omega_h, l)) = TE(\mathcal{L}(\Omega_h, l))/M.$$

Специально для исследования алгоритмов нахождения кратчайшего пути и построения строя в [15] были разработаны методы построения случайных ландшафтов с заданной конфигурационной энтропией, TE и TED и исследованы зависимости между упомянутыми характеристиками.

5. Поиск квазиоптимальной траектории

Получившийся в результате работы алгоритма 1 маршрут сравнивался с глобально оптимальным, найденным с помощью алгоритма Дейкстры. Для разных типов ландшафтов результаты сравнения получились довольно разными. Определим отклонение длины Λ маршрута $r_h(t) = (i(t), j(t))$ от оптимального по времени маршрута $r_{opt}(t) = (i_{opt}(t), j_{opt}(t))$, $t \in \mathbb{Z}_{\geqslant 0}$:

$$\delta\Lambda = \frac{|\Lambda(r) - \Lambda(r_{opt})|}{\Lambda(r_{opt})} \cdot 100\%,$$

и отклонение времени T прохождения маршрута r_h от времени T_{opt} прохождения оптимального маршрута r_{opt}:

$$\delta T = \frac{|T - T_{opt}|}{T_{opt}} \cdot 100\%.$$

Зададим функцию $N(S) = 0{,}00160518\mathrm{e}^{4{,}22769S}$ зависимости количества препятствий от конфигурационной энтропии ландшафта. Для ландшафтов одного типа это будет в действительности количество препятствий, для ландшафтов другого типа эта величина применяется для того, чтобы их можно было легче сравнивать с другими.

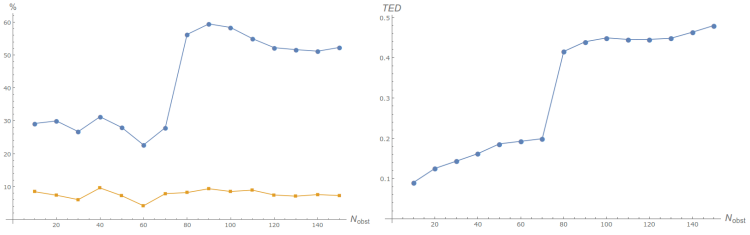
Исследовались квадратные случайные ландшафты $\mathcal{L}(n \times m, l)$ со сторонами в $n = m = 100$ клеток, с $l = 9$ различными классами клеток. Агент имел радиус обзора $r = 5$ и функционал поиска оптимального маршрута вида (10).

Было произведено по 1000 экспериментов для каждого значения $N(S)$ в случайном ландшафте $\mathcal{L}(100 \times 100, 9)$. В каждом эксперименте агент перемещался из клетки $(1, 1)$ в клетку $(98, 98)$[2]. Оказалось, что при возрастании конфигурационной энтропии S в ландшафте, состоящем из равномерно распределенных клеток разной непроходимости (см. рис. 7c в Приложении), среднее отклонение длины и времени от оптимальных (рис. 2a) медленно убывает, причем в районе $N(S) = 70$ наблюдается небольшая ступенька (более выраженная у ландшафтов малой площади), очевидно связанная со ступенькой в зависимости среднего значения TED от $N(S)$ (рис. 2b).

Однако при возрастании количества препятствий в ландшафте, состоящем из нескольких максимально труднопроходимых клеток, которые окружены менее непроходимыми, которые, в свою очередь, окружены еще менее непроходимыми и т.д. (рис. 7a, 7b в Приложении), характеристики алгоритма поиска несколько иные. При росте количества максимально непроходимых клеток N_{obst} и соответствующем росте конфигурационной энтропии (для такого ландшафта справедливо соотношение $N_{obst} = N(S)$) среднее отклонение длины и времени от оптимальных медленно убывает, причем в районе $N_{obst} = 70$ наблюдается резкий скачок (рис. 3a и 3b), после чего опять наблюдается

[2] *Данные доступны по адресу https://www.researchgate.net/publication/319998893_Eksperimentalnye_dannye_po_sravneniu_kvazioptimalnoj_i_optimalnoj_traektorij_agenta.*

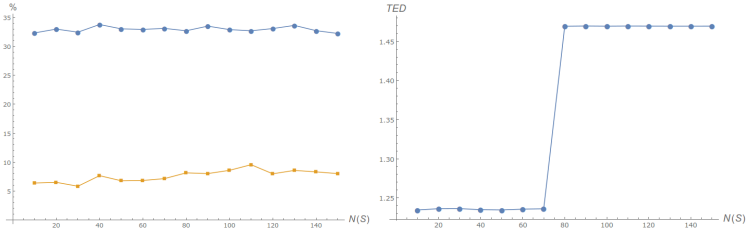
медленное убывание среднего отклонения. Автор полагает, что именно такие ландшафты, а не полностью равномерно случайные, более соответствуют встречающимся в приложениях и что отклонение от оптимального маршрута не более 10% по длине и порядка 30% по времени вполне достаточно для многих практических нужд. Следует учесть, что при поиске локально оптимального маршрута алгоритмом Дейкстры требуется всего примерно $O((2r+1)^4)$ времени. При «склейке» квазиоптимального маршрута длиной L клеток из локально оптимальных временная сложность будет, таким образом, примерно $O((2r+1)^4 L)$ при пересчете направления маршрута каждый ход, при котором изменяется положение агента, или $O((2r+1)^3 L)$ при пересчете маршрута лишь по достижении границы окрестности вместо $O((nm)^2)$ при поиске глобально оптимального маршрута, что дает определенный выигрыш при $r^2 \ll nm$.



(a) (b)

Рис. 2: *Проигрыш времени (круглые маркеры) и отклонение от траектории (квадратные) в процентах по сравнению с оптимальным маршрутом для равномерного случайного ландшафта*

На рис. 4а приводятся гистограммы распределения времени прибытия в конечную точку маршрута для равномерного ландшафта ($N_{obst} = 130$). Слева на рисунке изображена гистограмма распределения времени прибытия в конечную точку для оптимального маршрута, найденного алгоритмом Дейкстры, справа – для квазиоптимального маршрута. Данные гистограммы при-



(a) (b)

Рис. 3: *Проигрыш времени (круглые маркеры) и отклонение от траектории (квадратные) в процентах по сравнению с оптимальным маршрутом для «естественного» случайного ландшафта*

мерно соответствуют (с p-значениями около 0,7) распределению Нортона–Райса с функцией плотности

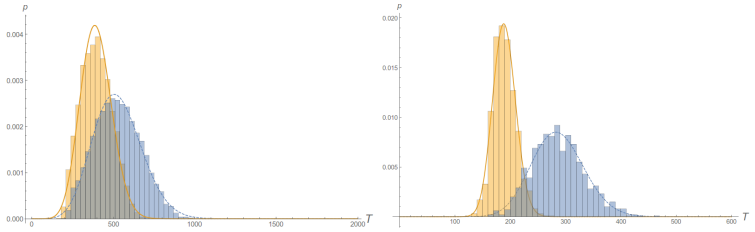
$$p(m,\alpha,\beta;x)=\begin{cases}\dfrac{\alpha m\left(\frac{x}{\alpha}\right)^m e^{-\frac{m\left(\alpha^2+x^2\right)}{2\beta^2}} I_{m-1}\left(\frac{mx\alpha}{\beta^2}\right)}{\beta^2}, & x>0,\\ 0, & x\leqslant 0,\end{cases}$$

где I_k – модифицированная функция Бесселя первого рода.

Аналогичные гистограммы приведены на рис. 4b для распределения времени прибытия в конечную точку маршрута для «естественного» ландшафта ($N_{obst}=130$).

6. Метрика сходства графов и метрика, порожденная графом

Определение 5. *Определим расстояние* $\mathrm{gd}: Ag^2\to\mathbb{R}_{\geqslant 0}$, *следующим образом. Если между агентами* ag_1, $ag_2\in Ag$ *существует в* Φ *кратчайший путь длиной* l, *то* $\mathrm{gd}(ag_1,ag_2)=l$, *иначе* $\mathrm{gd}(ag_1,ag_2)=\infty$, $\mathrm{gd}(ag_1,ag_1)=0$.



(a) (b)

Рис. 4: *Гистограммы распределения времени прибытия в конечную точку маршрута для равномерного (a) и «естественного» (b) ландшафтов*

Это расстояние принято называть геодезическим расстоянием на графе. Будем считать окрестностью агента $ag \in Ag$ радиуса ρ множество $W_\rho(ag) \subseteq Ag$, такое что $\forall(ag' \in W_r(ag)) \quad \mathrm{gd}(ag, ag') \leqslant \rho$. Соответственно, будем считать соседями агента ag всех агентов $ag' \in W_\rho(ag)$.

Пусть $\Gamma_i = (Ag_i, E_i, \mathrm{coo}_i)$, $Ag_i \subseteq Ag$, $E_i \subseteq Ag^2$, $\mathrm{coo}_{\Gamma_i} : \mathbb{Z}_{\geqslant 0} \times Ag_i \to \Omega_h$ – функция соответствия вершины графа Γ_i и клетки Ω_h, $i = 1, 2$, – два графа, описывающие строи агентов. Определим расстояние несходства между ними в момент времени t как

$$\mathrm{dist}(\Gamma_1(t), \Gamma_2(t)) = m_0 + m_1 + \kappa \sum_{ag \in Ag_c} \|\mathrm{coo}_{\Gamma_1}(t, ag) - \mathrm{coo}_{\Gamma_2}(t, ag)\|,$$

где $Ag_c \subseteq Ag$ – множество вершин наибольшего общего подграфа Γ_1 и Γ_2, $m_0 = |Ag_0 \setminus Ag_c|$, $m_1 = |Ag_1 \setminus Ag_c|$, $\kappa = (\mathrm{diam}(\Omega_h))^{-1}$ – диаметр множества Ω_h в клетках. Это расстояние в своей сущности является частным случаем ранее введенного в [3] расстояния графов.

7. Синтез локальной функции перехода

Обратим внимание на то, что у двумерного *robot-space cellular automaton* окрестность $V_r(i,j)$ каждой ячейки в клетке (i,j) порождена окрестностью (i,j) в обычной евклидовой метрике $\mathbb{R}^2$, и алгоритм нахождения скорейшего маршрута учитывает такую окрестность. Однако алгоритм построения строя основан именно на *robot-space cellular automaton* и в качестве множества ячеек рассматривает Ag, а в качестве окрестности агента $ag \in Ag$ – окрестности $W_\rho(ag)$, порожденные геодезическим расстоянием на графе строя Ф.

Доработаем алгоритм 1 так, чтоб агенты стремились в каждый момент времени поддерживать формацию, заданную графом Ф. Отметим, что без существенных потерь во времени прохождения поддерживать в точности заданную формацию можно лишь на ландшафтах с относительно малым количеством препятствий (иначе говоря, с малой конфигурационной энтропией).

Если предположить, что строй агентов в момент времени t описывается графом $\Gamma(t)$, то агенты стараются выбрать такое направление движения, которое было бы компромиссным между направлением, минимизирующим $\mathrm{dist}(\Phi(t),\Gamma(t))$ (точнее, минимизирующим расстояние между графом строя и той части реального графа строя, которая известна агенту) и направлением в сторону кратчайшего по времени пути.

В рамках такой логики агентов, существуют следующие виды взаимодействия:

1. $\mathrm{predict} : Ag \to \Omega_h$ – функция, с помощью которой агент предсказывает новое положение другого агента на следующем такте функционирования КА. Поскольку каждый агент каждый такт вычисляет и сортирует возможные направления движения по желательности (алгоритмы 3 и 7), то $\mathrm{predict}(ag)$ может возвращать, например, $d_0 + (i,j)$, где d_0 – самое желательное для агента в клетке (i,j) направление. Проще говоря, все агенты содержат одинаковый алгоритм поведения. Поэтому каждый агент может «мысленно» подставить себя на место другого и просчи-

тать, куда бы пошел он сам в этой ситуации. Результат такого подсчета и возвращает функция $\mathrm{predict}$. Более формально, агент вызывает функцию вычисления весов не со своими координатами на плоскости и не со своей позицией в графе строя, а с координатами и позицией в строю того агента, для которого он хочет предсказать положение.

2. Рекурсивный запрос остановки. Агент просит остановиться своего лидера, который просит остановиться своего лидера и т.п., пока запрос не дойдет до агента, не имеющего лидера.

3. Рекурсивный запрос продолжения движения. Агент просит продолжить движение своего лидера, который просит продолжить движение у своего лидера и т.п., пока запрос не дойдет до агента, не имеющего лидера.

Отметим, что поскольку $\mathrm{dist}(\Phi, \Gamma) \geqslant 0$ и $T_h(r_h) > 0$ для любых графов Φ, Γ и маршрута r_h, то для скаляризации вышеупомянутых целевых функций их проще всего сложить. Идея алгоритма состоит в том, что

1. Находящиеся в одной окрестности $V_r(i,j)$ и в одной окрестности $W_\rho(ag)$ агенты выбирают лидера (алгоритм 6), за которым следуют, причем так, чтоб лидер агентов одной окрестности обязательно бы следовал за лидером агентов в другой, кроме лидера самого высокого уровня. Множество агентов из $W_\rho(ag)$, найденных агентом ag в окрестности $V_r(i,j)$, обозначим как $\mathit{found}(ag)$.

2. Каждому возможному направлению $d \in \mathcal{D}$ движения агента ag в клетке (i,j) присваиваются веса, учитывающие как направление в сторону кратчайшего пути, аналогично (8), так и в сторону сохранения строя, например

$$weights[d] = \min_{r_h \in \mathcal{M}_d(i,j;i_r,j_r)} T_h(r_h) + \\ + \sum_{ag' \in found(ag)} \|\mathrm{coo}_\Phi(t, ag') + d - \mathrm{predict}(ag'))\|.$$

3. Множество $\mathcal{D}$ случайно перемешивается в соответствии с весами $weights$, получая множество $\mathcal{D}'$. Используется предложенный в [9] алгоритм случайного взвешенного перемешивания. Без перемешивания возможна ситуация, когда веса направлений движения в нескольких направлениях будут равны и неясно, какой вес выбрать. Также во многих случаях агенту будет тогда неясно, что предпочесть – удерживать строй и проиграть в скорости или же отклониться от строя и выиграть в скорости. Более того, перемешиванием моделируется ситуация, когда люди, например, неверно угадывают направление движения соседа и сталкиваются с ним.

4. Агент выбирает лучшее направление из перемешанного $\mathcal{D}'$, если не слишком отстал от лидера. Поскольку перемешивание не равномерно, а учитывает веса (т.е. самые оптимальные маршруты имеют самый большой шанс оказаться первыми) с помощью особого алгоритма, то поддержание строя от такого перемешивания практически не страдает. У экспоненциального распределения, которое используется для перемешивания, очень легкий хвост и вероятность выбора направления, сильно нарушающего строй, близка к нулю.

5. Если агент слишком отстал, то для движения за лидером агент в клетке (i, j) вычисляет положение лидера (i_f, j_f) согласно своему шаблону формации, и ищет в окрестности $V_\varepsilon(i_f, j_f)$ своего лидера. При обнаружении лидера ag_l в $(i_l, j_l) \in V_\varepsilon(i_f, j_f)$ агент выбирает такое направление движения d, чтоб минимизировать $\|(i, j) + d - \mathrm{predict}(ag_l)\|$.

6. Если в окрестности $V_r(i, j)$ ведущего агента ag слишком много агентов, не принадлежащих $W_\rho(ag)$ или, наоборот, не хватает агентов, принадлежащих $W_\rho(ag)$, то ag пытается выбрать другое направление (строка 24 алгоритма 2) или запросить «лишних» или «недостающих» агентов выбрать другое направление. Подробный разбор механизма обмена сообщениями в среде «Психоход» [5] выходит за рамки настоящей работы, но, в целом, когда агент обнаруживает, что ему пришло сообщение типа «запрос другого направления», то он генерирует веса направлений

заново, причем устанавливает вес ранее выбранного направления в 0, делая невозможным выбор этого направления.

Рекурсивные запросы от ведомых агентов к лидерам нужны для восстановления сильных нарушений строя или на очень неоднородных ландшафтах, когда самоорганизация для необщающихся агентов не работает. При этом, если лидер ag заметил отклонение от строя ведомых агентов из множества *found*, то он вместо остановки выбирает направление движения

$$\mathrm{def}(ag) = \frac{\sum_{ag' \in found}(\mathrm{coo}_\Gamma(ag') - \mathrm{coo}_\Phi(ag'))}{\| \sum_{ag' \in found}(\mathrm{coo}_\Gamma(ag') - \mathrm{coo}_\Phi(ag'))\|}.$$

Подробно вышеуказанное поведение агентов может быть описано как алгоритм 2, в который входят алгоритмы 5-7, приведенные в Приложении.

Алгоритм 2 (Поддержания строя)**.**

for all $ag \in Ag$ **do**
  $weights =$ SORTDIRECTIONS(ag) ▷ Алгоритм 3
  $predicted, less, more, found =$ FINDNEIGHBOURS(ag, ε)▷ Алгоритм 5
  if t == 0 **then**
    FINDLEADER(ag, $found$) ▷ Алгоритм 6
  end if
  $distances =$ FINDDIRECTION(ag, $found$, $predicted$) ▷ Алгоритм 7
  Случайно перемешать $ag.\mathcal{D}'$ в соответствии с весами из $weights$ и $distances$.
  $d_0 = ag.\mathcal{D}'[1]$ ▷ Лучшее направление
  if $ag.lead \wedge distances[d_0] \gg r_0$ **then** ▷ r_0 – допустимое расхождение со строем
    $trgtTmp = trgtAct$ ▷ Назначить временную цель
    $trgtAct = (ag.@cell \to coo) + \mathrm{def}(ag)$
    Рекурсивный запрос остановки агента с УИд $ag.leadId$
  else if $ag.lead \wedge distances[d_0] \approx r_0 \wedge$ был рекурсивный запрос остановки **then**

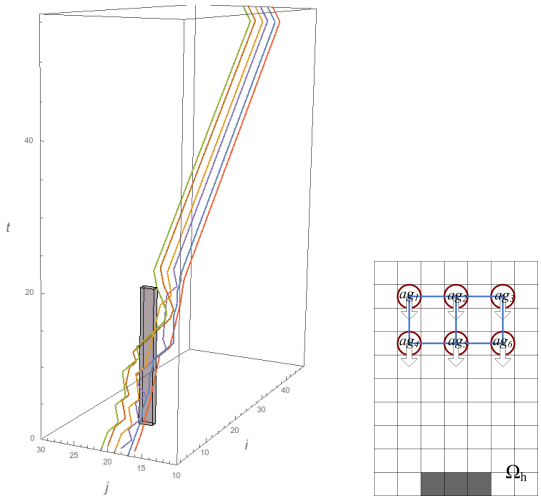
Продолжить движение
Рекурсивный запрос продолжить движение агента с УИд $ag.leadId$
else if $\neg ag.lead \wedge distances[d_0] \gg r_0$ **then**
Рекурсивный запрос остановки агента с УИд $ag.leadId$
$trgtTmp = trgtAct$ ▷ Назначить временную цель
$trgtAct = found[leadId] - (i_s, j_s)$ ▷ $(i_s, j_s, ag.selfId) \in ag.formTmpl$ – собственное положение агента в строю
else if $distances[d_0] \approx r_0 \wedge$ была назначена временная цель **then**
$trgtAct = trgtTmp$
Рекурсивный запрос продолжить движение агента с УИд $ag.leadId$
else if длина $more$ или $less$ слишком велика **then**
Случайно перемешать $ag.\mathcal{D}'$ в соответствии с весами из $weights$ и $distances$.
end if
end for
for all $ag' \in Ag$ **do** ▷ Актуальное перемещение агентов
ACTUALIZE(ag') ▷ Алгоритм 4
end for
t = t+1

8. Результаты симуляции

Предложенный алгоритм организации строя был смоделирован в программе «Психоход»[3], разработанной в [5], и при наличии небольших одиночных препятствий агенты ведут себя так, как показано на рис. 5a. На указанном рисунке изображено шесть агентов ag_i, $i = \overline{1,6}$, имеющих граф строя, показанный на рис. 5b. Для ag_2, ag_4 лидером является агент ag_1, для ag_3, ag_5 – агент ag_2, для ag_6 – агент ag_3. Агенты пытаются наименее времязатратным

[3] *https://bitbucket.org/bokohodteam/bokohod.*

образом обойти препятствие (показанное в виде прямоугольника), после чего восстанавливают строй.



(a) (b)

Рис. 5: *Обход небольшого одиночного препятствия строем из шести агентов*

Однако когда ландшафт становится более неоднородным, эффективность построения строя снижается. Причины у этого следующие: агенты теряют способность адекватно предсказывать направления движения друг у друга и агентам негде развернуть правильный строй после его нарушения из-за препятствия, так как препятствия встречаются слишком часто.

Обозначим

$$\Delta\Phi = \frac{1}{T} \sum_{ag \in Ag} \sum_{t=\overline{1,T}} \|\mathrm{coo}_\Phi(t, ag) - \mathrm{coo}_\Gamma(t, ag)\|,$$

где $\mathrm{coo}_\Phi(t, ag)$ – желаемое положение агента ag в такт $t = \overline{1, T}$ согласно графу строя Φ, $\mathrm{coo}_\Gamma(t, ag)$ – реальное положение агента

ag в такт $t = \overline{1,T}$. Моделирование[4] вышеупомянутого строя из шести агентов на «естественном ландшафте», типа указанного в разделе 5, дает результат, приведенный на рис. 6. На этом рисунке показана зависимость $\Delta\Phi/6$ (множитель 1/6 взят, потому как агентов шесть, и смысл величины $\Delta\Phi/6$ – это среднее отклонение агента от своей позиции в строю) от количества препятствий N_{obst}, которая аппроксимируется с коэффициентом детерминации $R^2 = 0{,}992729$ как

$$\Delta\Phi/6 \approx 0{,}901073 \ln(N_{obst} + 1) - 0{,}673763.$$

Выяснилось, что уменьшить отклонение агентов от строя

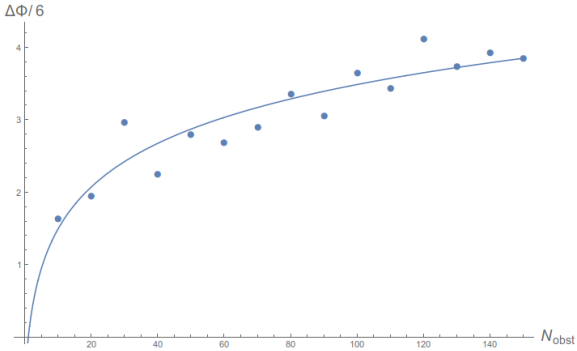


Рис. 6: *Зависимость среднего отклонения позиции агентов от предусмотренной строем и количества препятствий*

возможно, увеличивая параметр ε, указывающий, в насколько широкой окрестности $V_\varepsilon(i_f, j_f)$ заданной шаблоном строя, $(i_f, j_f, agId)$ следует искать агента с УИд $agId$.

[4] *Данные доступны по адресу https://www.researchgate.net/publication/318663064_Eksperimentalnye_dannye_otnositelno_obhoda_prepatstvij_stroem_iz_sesti_agentov.*

9. Заключение

Получены зависимости эффективности построения квазиоптимального маршрута от типа ландшафта и временная вычислительная сложность этого построения. Разработан алгоритм построения и поддержания строя агентов, двигающихся по ландшафту с препятствиями, в котором в качестве критерия близости получившегося строя к заданному используется определенная метрика несходства графов. Также описана новая методика тестирования алгоритмов нахождения оптимального маршрута с помощью случайных ландшафтов с заданными характеристиками типа конфигурационной энтропии.

Клеточный автомат, сконструированный в статье, может быть использован как часть искусственного интелекта наземного робота, передвигающегося в составе строя других подобных ему роботов. Хочется подчеркнуть, что клеточный автомат был положен в основу предлагаемой модели в силу нелинейности уравнений (1)–(3), определяющих движение агента, и невозможности получить в явном виде функционал времени прохождения к цели T для меняющегося со временем ландшафта. Перечисленные обстоятельства, а также тот факт, что функция проходимости u^c обычно задана в табличном виде (т.е. изначально разрывна) сильно препятствуют применению классических методов.

Далее планируется получить результат, аналогичный приведенному в статье, для организации телекоммуникационной *ad hoc* сети агентов, движущихся по пересеченной местности.

Приложение

В данном разделе приведены примеры ландшафтов различного вида. Подробное описание способов генерации ландшафтов приведено в работах [14, 15]. Также приведен псевдокод алгоритмов, необходимых для построения строя и поиска оптимального по времени маршрута.

Алгоритм 3 (Сортировки направлений движения)**.**

function SORTDIRECTIONS(ag)
  $(i, j) = ag.@cell \rightarrow coo$
  $weights = \{\}$
  if $ag.w > 0$ **then** $ag.w = ag.w - 1$
  end if
  if $ag.w == 0 \wedge (i, j) \neq ag.trgtAct$ **then**

$$direction = \frac{(i, j) - ag.trgtAct}{\|(i, j) - ag.trgtAct\|}$$

    $(i_r, j_r) =$ клетка, в которой с границей $V_r(i, j)$ пересекается луч с началом в (i, j) и направленный в направлении $direction$
    $ag.\mathcal{D}' = \mathcal{D}$
    for all $d \in \mathcal{D}$ **do**
      $weights[d] = \min_{r_h \in \mathcal{M}_d(i,j;i_r,j_r)} J(r_h)$
    end for
  end if
  return $weights$
end function

Алгоритм 4 (Актуализации новых положений)**.**

function ACTUALIZE(ag')
  $k = 1$
  $(i, j) = ag'.@cell \rightarrow coo$
  repeat
    $(i_{new}, j_{new}) := (i, j) + ag'.\mathcal{D}'[k]$
    $k = k + 1$
  until $ag'_{i_{new},j_{new}} \neq 0 \vee (i_{new}, j_{new}) \neq (i, j)$
  $cell_{i_{new},j_{new}}.@ag = @ag'$
  $ag'.@cell = @cell_{i_{new},j_{new}}.coo$
end function

Алгоритм 5 (Обнаружения соседей)**.**

function FINDNEIGHBOURS(ag, ε)
  $predicted = \{\}$, $less = \{\}$, $more = \{\}$, $found = \{\}$
  $(i, j) = ag.@cell \to coo$
  for all $(s, p, agId) \in ag.formTmpl$ **do** ▷ Поиск в окрестности $W_\rho(ag)$
    Искать в окрестности $V_\varepsilon(i + s, j + p)$ агента
    if агент ag' найден и $agId == 0$ **then**
      $agId = ag'.selfId$
      $predicted[agId] = \text{predict}(ag')$
      Добавить в $found$ $agId$
    else if агент ag' найден и $agId == ag'.selfId$ **then**
      $predicted[agId] = \text{predict}(ag')$
      Добавить в $found$ $agId$
    else if агент ag' найден и $agId \neq ag'.selfId \wedge agId \neq 0$ **then**
      Добавить в $more$ $ag'.selfId$ ▷ длина $more$ – это m_1 из определения метрики
    end if
    if Длина($ag.formTmpl$) > Длина($predicted$) **then**
      Добавить в $less$ УИд не найденных агентов ▷ длина $less$ – это m_0 из определения метрики
    end if
  end for
  return $predicted$, $less$, $more$, $found$
end function

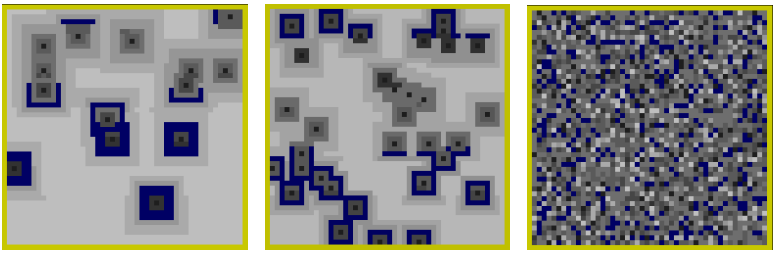
Алгоритм 6 (Назначения и обнаружения лидера)**.**

function FINDLEADER(ag, $found$)
  $ag.leadId = 0$
  for all $id \in found$ **do** ▷ агент выбирает лидером агента с самым маленьким УИд из своих соседей
    if $id < ag.selfId$ **then**
      $ag.leadId = id$
    end if
  end for

8: **end function**

Алгоритм 7 (Поиска направления для поддержания формации)**.**

1: **function** FINDDIRECTION(ag, $found$, $predicted$)
2: $dist = 0$, $distances = \{\}$
3: $(i, j) = ag.@cell \rightarrow coo$
4: **for all** $d \in \mathcal{D}$ **do**
5: **for all** $id \in found$ **do**
6: $dist = dist + \|d + (i, j) - predicted[id]\|$
7: **end for**
8: $distances[d] = dist$
9: **end for**
10: **return** $distances$
11: **end function**



(a) $S = 1{,}6$, $N_{obst} = 14$ (b) $S = 1{,}8$, $N_{obst} = 33$ (c) $S = 1{,}84877$

Рис. 7: *Примеры ландшафтов. Чем клетка темнее, тем больше ее непроходимость*

Литература

1. КУЗНЕЦОВ А.В. *Модель совместного движения агентов с трехуровневой иерархией на основе клеточного автомата* // Журнал вычислительной математики и математической физики. – 2017. – Т. 57, № 2. – С. 339–349. – URL: https://elibrary.ru/item.asp?id=28918677.
2. КУДРЯВЦЕВ В.Б., ПОДКОЛЗИН А.С., БОЛОТОВ А.А. *Основы теории однородных структур.* – М.: Наука, 1990. – 296 с.
3. КУЗНЕЦОВ А.В. *Мера несходства на множестве графов и ее приложения* // Вестник ВГУ. Серия: Системный анализ и информационные технологии. – 2017. – Т. 1. – С. 125–131. – URL: https://elibrary.ru/item.asp?id=29185115.
4. КУЗНЕЦОВ А.В. *Упрощенная модель боевых действий на основе клеточного автомата* // Известия РАН. Теория и системы управления. – 2017. – Т. 56, № 3. – С. 59–71. – URL: https://elibrary.ru/item.asp?id=29369822.
5. КУЗНЕЦОВ А.В., ЛЕЩЕВ А.С. *Программная среда многоагентного моделирования «Психоход»*. Пр. для ЭВМ № 2017619605. Дата регистрации: 28.08.2017, номер и дата поступления заявки: 2017616880 11.07.2017. Правообл. А.В. Кузнецов, А.С. Лещев // Программы для ЭВМ. Базы данных. Топологии интегральных микросхем. – 2017. – № 9.
6. МАЛИНЕЦКИЙ Г. Г., СТЕПАНЦОВ М. Е. *Применение клеточных автоматов для моделирования движения группы людей* // Журнал вычислительной математики и математической физики. – 2004. – Т. 44, № 11. – С. 2094–2098.
7. BEER B., MEAD R., WEINBERG J.B. *A Distributed Method for Evaluating Properties of a Robot Formation* // Proc. of the 24th Conference on Artificial Intelligence (AAAI-2010), July 11–15, 2010, Atlanta, Georgia, USA. – URL: http://www.aaai.org/ocs/index.php/AAAI/AAAI10/paper/view/1677.

8. CUSHMAN S.A. *Calculating the configurational entropy of a landscape mosaic* // Landscape Ecology. – 2016. – Vol. 31, No. 3. – P. 481–489. – URL: http://dx.doi.org/10.1007/s10980-015-0305-2.
9. EFRAIMIDIS P., SPIRAKIS P. *Weighted Random Sampling* // Encyclopedia of Algorithms / Ed. by Ming-Yang Kao. – Boston, MA: Springer US, 2008. – P. 1–99. – ISBN: http://isbndb.com/search-all.html?kw=978-0-387-30162-4978-0-387-30162-4. – URL: https://doi.org/10.1007/978-0-387-30162-4_478.
10. *FRAGSTATS: Spatial Pattern Analysis Program for Categorical Maps. Documentation.* – URL: http://www.umass.edu/landeco/research/fragstats/documents/fragstats_documents.html.
11. GEUVERS H. *Inductive and coinductive types with iteration and recursion* // Proc. of the 1992 workshop on Types for Proofs and Programs. – Bastad: Chalmers University of Technology, 1992. – P. 183–207. – URL: http://www.cs.ru.nl/ herman/PUBS/BRABasInf_RecTyp.pdf.
12. ILACHINSKI A. *Artificial War: Multiagent-Based Simulation of Combat.* – Singapore: World Scientific Publishing Company, 2004. – P. 747.
13. JONES M.P., DUDENHOEFFER D.D. *A formation behavior for large-scale micro-robot force deployment* // Winter Simulation Conference. – Vol. 01. – Los Alamitos, CA, USA: IEEE Computer Society, 2000. – P. 972–982. – URL: doi.ieeecomputersociety.org/10.1109/WSC.2000.899900.
14. KUZNETSOV A.V. *Cellular automata-based model of group motion of agents with memory and related continuous model* // Mathematical Modeling. Information Technology and Nanotechnology – 2017 / Ed. by S. Sazhin, E. Shchepakina, V. Sobolev, D. Kudryashov. – CEUR Workshop Proc. No. 1904. – Aachen, 2017. – P. 223–231. – URL: http://ceur-ws.org/Vol-1904/paper38.pdf.

15. KUZNETSOV A. *Generation of a Random Landscape by given Configuration Entropy and Total Edge* // Computational Technologies. – 2017. – Vol. 22, No. 4. – P. 4–10. – URL: https://elibrary.ru/item.asp?id=30053459.
16. LONG R.L., MEAD R., WEINBERG J.B. *Distributed Auction-Based Initialization of Mobile Robot Formations* // Proc. of the 24th Conference on Artificial Intelligence (AAAI-2010), July 11-15, 2010, Atlanta, Georgia, USA. – 2010. – URL: http://www.aaai.org/ocs/index.php/AAAI/AAAI10/paper/view/1673.
17. MEAD R. *Cellular Automata for Control and Interactions of Large Formations of Robots.* – 2008. – URL: http://robotics.usc.edu/ rossmead/docs/2008/2008Mead_Thesis.pdf.
18. MEAD R., WEINBERG J.B. *2-Dimensional Cellular Automata Approach for Robot Grid Formations* // Proc. of the 23rd Conference on Artificial Intelligence (AAAI-2008), July 13-17, 2008, Chicago, Illinois, USA. – 2008. – P. 1818–1819. – URL: http://www.aaai.org/Library/AAAI/2008/aaai08-299.php.
19. MEAD R., WEINBERG J.B. *A Single- and Multi-Dimensional Cellular Automata Approach to Robot Formation Control* // Proc. of IEEE Int. Conference on Robotics and Automation (ICRA-08). – 2008. – URL: http://robotics.usc.edu/ rossmead/docs/2008/2008WeinbergMead_ICRA08.pdf.
20. RODRÁGUEZ-ITURBE I., D'ODORICO P., RINALDO A. *Configuration entropy of fractal landscapes* // Geophysical Research Letters. – 1998. – Vol. 25, No. 7. – P. 1015–1018. – URL: http://dx.doi.org/10.1029/98GL00654.

ORGANIZATION OF AN AGENTS' FORMATION THROUGH A CELLULAR AUTOMATON

Alexander Kuznetsov, Voronezh State University, Voronezh, Cand.Sc., associate professor (avkuz@bk.ru).

Abstract: The article deals with the algorithm for a distributed organization of the agents' formation defined by a graph and the numerical simulation of such algorithm. Agents move through terrain with many random obstacles ("random landscape"). At first, we describe the continuous statement of the two-criteria minimization problem. The first criterion is the agent's route time. The second criterion is the closeness of agents' formation to the desired one. Next, we introduce a cellular automaton simulating the movement of agents for obtaining quasi-optimal solutions of the problem. The cellular automaton has one-dimensional and two-dimensional representations. Agents use reflexion to predict the motion of other agents. Then we compare obtained solutions with optimal ones for different types of random landscapes via numerical experiment. At this point, we obtain the empirical distribution for the time of an agent's exit to finish point. Finally, we find the relation between a type of random landscape through which agents move and the quality of agents' formation.

Keywords: cellular automaton, autonomous agents, reflexive agents, agents' formation control.

Статья представлена к публикации
членом редакционной коллегии А.А. Шевляковым.

Поступила в редакцию 08.08.2017.
Дата опубликования 30.11.2017.