К.С. Яковлев

HGA*: эффективный алгоритм планирования траектории на плоскости

Аннотация. В статье рассматриваются существующие и перспективные подходы и методы планирования траектории на плоскости. Проводится анализ и дается качественная оценка существующих алгоритмов планирования. Описывается новый, эффективный алгоритм планирования траектории – HGA*, позволяющий осуществлять поиск плана при недостатке временных и вычислительных ресурсов, а также при дополнительных ограничениях (динамическое изменение и частичная наблюдаемость среды планирования). Приводятся результаты экспериментов, иллюстрирующие превосходство HGA* над имеющимися аналогами.

Ключевые слова: интеллектуальные системы управления, автоматическое планирование, планирование траектории, A*.

Введение

Одной из ключевых проблем, возникающих при разработке систем управления (СУ) беспилотными транспортными средствами (БТС), является проблема создания эффективного планировщика – программно-аппаратного модуля, отвечающего за построение плана достижения поставленных целей в автоматическом режиме. В современных СУ процесс планирования рассматривается на двух уровнях – стратегическом (уровне целей) и тактическом (уровне перемещений) [1]. И хотя оба данных уровня тесно связаны между собой в рамках СУ БТС [2], методы их решения различны.

В работе рассматривается задача планирования на уровне перемещений, которая заключается в построении оптимальной (по некоторой совокупности критериев) траектории перемещения БТС. Особое внимание уделяется наиболее сложным, приближенным к практике, случаям, а именно тем случаям, когда временные и вычислительные ресурсы ограничены, отсутствует полная (априорная) информация об окружающей среде, среда динамически изменяется в процессе выполнения плана. В качестве модельного примера рассматривается задача построения траектории перемещения БТС типа «вертолет», при осуществлении маловысотного полета в городских условиях. При этом предполагается, что БТС может осуществлять облет препятствий лишь в горизонтальной плоскости, т.е. рассматривается задач построения траектории в двумерном пространстве.

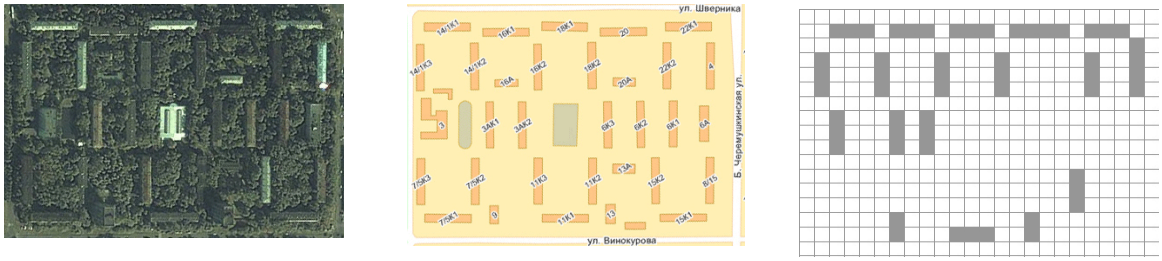
Для решения указанной задачи используются графы специальной структуры – метрические, топологические (МТ-графы), описанные в [3]. МТ-графы позволяют формализовать знания о предметной области и использовать их при планировании с целью снижения временных затрат и экономии вычислительных ресурсов. Основываясь на принципах, изложенных в [3], в работе описывается такой алгоритм построения траектории БТС, который позволяет учитывать динамику и частичную наблюдаемость среды уже на стадии планирования, а не исполнения плана.

1. Постановка задачи

Будем рассматривать задачу планирования траектории на плоскости, как задачу поиска пути на графе особой структуры – МТ-графе [3].

МТ-граф – это неупорядоченная пара:

$$\text{MT-Gr}=<A, d>, \qquad (1)$$



а) б) в)

Рис.1. Представление карты местности в виде МТ-графа
а) спутниковый снимок местности, б) карта местности, в) МТ-граф

где A –множество клеток, представляющее собой матрицу $A_{m\times n}=\{a_{ij}\}$: $a_{ij}=0\vee 1$, $\forall i, j$: $0\le i<m$, $0\le j<n$, m, $n \in \boldsymbol{N}\setminus\{0\}$; d – метрика на множестве $A^{+}=\{a_{ij}|a_{ij}\in A, a_{ij}=0\}$.

Клетку МТ-графа будем называть проходимой, если $a_{ij}=0$; непроходимой, если $a_{ij}=1$.

Графически МТ-граф можно представить в виде таблицы, содержащей m строк и n столбцов. Ячейки таблицы соответствуют клеткам a_{ij}, при этом проходимым клеткам соответствуют светлоокрашенные ячейки, а непроходимым – темноокрашенные (Рис. 1).

Две различные клетки a_{i1j1}, a_{i2j2} будем называть смежными, если: $\forall i_1,j_1,i_2,j_2$: $0\le i_1,i_2<m$, $0\le j_1,j_2<n \rightarrow |i_1-i_2|\le 1 \vee |j_1-j_2|\le 1$. Множество клеток, смежных с a_{ij} будем обозначать как $adj(a_{ij})$. Множество смежных непроходимых клеток МТ-графа будем называть препятствием: $Obs=\{a_{i0j0}, a_{i1j1}, a_{i2j2}, \ldots, a_{isjs}|a_{ikjk}=1, a_{ikjk}\in adj(a_{ik\text{-}1jk\text{-}1})\ \forall k=0,1,2,\ldots,s, s\in\boldsymbol{N}\}$.

Пусть $ADJ\subset A\times A$ – множество всех пар смежных клеток, c_{hv}, $c_d\in R^{+}=\{r|r\in\Re, r>0\}$, а функция c: $ADJ \rightarrow \{c_{hv}, c_d, +\infty\}$ – коммутативная функция, определяющая веса переходов следующим образом: $c(a_{ij}, a_{lk})=c_{hv}$, если $a_{ij}=0 \wedge a_{lk}=0$ и клетки a_{ij}, a_{lk} являются горизонтально или вертикально смежными; $c(a_{ij}, a_{lk})=c_d$, если $a_{ij}=0 \wedge a_{lk}=0$ и клетки a_{ij}, a_{lk} являются диагонально смежными; $c(a_{ij}, a_{lk})=+\infty$, если $a_{ij}=1 \vee a_{lk}=1$. При этом положим $c_d=\sqrt{2}\cdot c_{hv}$.

В качестве метрики d будем использовать функцию:

$$H(a_{ij}, a_{lk}) = \begin{cases} c_d\Delta_j + c_{hv}(\Delta_i-\Delta_j), & \text{если } \Delta_i(a_{ij},a_{lk})\ge\Delta_j(a_{ij},a_{lk}) \\ c_d\Delta_i + c_{hv}(\Delta_j-\Delta_i), & \text{если } \Delta_i(a_{ij},a_{lk})<\Delta_j(a_{ij},a_{lk}) \end{cases}, \quad (2)$$

известную в литературе по искусственному интеллекту как диагональная эвристика [4]. Здесь $\Delta_i=\Delta_i(a_{ij}, a_{lk})=|i-l|$, а $\Delta_j=\Delta_j(a_{ij}, a_{lk})=|j-k|$.

Пусть a_{ij}, $a_{lk} \in A_{m\times n}$, при этом $a_{ij}\ne a_{lk}$, $a_{ij}\ne 0$, $a_{lk}\ne 0$. Путем из a_{ij} в a_{lk} будем называть последовательность смежных проходимых клеток МТ-графа $\pi=\{a_{i0\,j0}, a_{i1\,j1}, a_{i2\,j2}, \ldots, a_{is\,js}\}$: $a_{i0\,j0}=a_{ij}$, $a_{is\,js}=a_{lk}$. Будем обозначать путь как $\pi(a_{ij}, a_{lk})$ или просто π. Клетку a_{ij} будем называть начальной, a_{lk} – целевой. Вес пути π определяется как сумма весов переходов по всем смежным клеткам, входящим в данный путь:

$$c(\pi)=\sum_{v=1}^{s} c(a_{i_{v-1}j_{v-1}}, a_{i_v j_v}). \quad (3)$$

Используя приведенное выше описание МТ-графа, задача планирования может быть формально представлена в виде тройки:

$$PTask=<\text{MT-Gr}, a_{startI\,startJ}, a_{goalI\,goalJ}>, \quad (4)$$

и сформулирована следующим образом. Пусть на МТ-графе зафиксированы начальная $a_{startI,startJ}$ и целевая $a_{goalI,goalJ}$ клетки, такие что $a_{startI,startJ}\ne a_{goalI,goalJ}$, $a_{startI,startJ}\ne 1$, $a_{goalI,goalJ}\ne 1$. Необходимо найти путь $\pi(a_{startI,startJ}, a_{goalI,goalJ})$, т.е. последовательность клеток МТ-графа $\pi=\{a_{i0j0}, a_{i1j1}, a_{i2j2}, \ldots, a_{isjs}\}$ такую, что $a_{i0j0}=a_{startI,startJ}$, $a_{isjs}=a_{goalI,goalJ}$, $\forall v: 1\le v<s, (a_{iv-1\,jv-1}, a_{iv\,jv})\in ADJ$.

2. Методы решения

Поскольку любому МТ-графу может быть поставлен в соответствие граф [3], то можно утверждать, что для МТ-графов справедливы те же алгоритмы поиска пути, что и для графов, при этом в плане практической реализации МТ-графы являются более простыми структурами. Ниже будут описаны алгоритмы итерационного обхода графа, традиционно применяющиеся для решения поставленной задачи в искусст-

венном интеллекте, а также будет предложен новый алгоритм построения пути на МТ-графе, использующий знания о предметной области.

2.1. Алгоритмы семейства A*

A* - это принятое в литературе по искусственному интеллекту обозначение для семейства алгоритмов эвристического поиска пути на графе. Алгоритмы семейства A* представляют собой различные модификации алгоритма Дейкстры, основанные на использовании эвристик для сокращения пространства поиска. Основная идея, лежащая в основе A*-поиска (применительно к МТ-графам), заключается в следующем.

Предположим, что для любой клетки МТ-графа a_{ij} известен вес кратчайшего пути в эту клетку из $a_{startI,startJ}$. Обозначим эту величину как $g^*(a_{ij})$ и будем ссылаться на нее как на g^*-значение клетки a_{ij}. Тогда для того, чтобы отыскать кратчайший путь из $a_{startI,startJ}$ в $a_{goalI,goalJ}$, достаточно осуществить регрессивный поиск по следующему алгоритму: начиная с $a_{goalI,goalJ}$ в искомый путь добавляются такие клетки a_{kl} смежные с текущей клеткой a_{ij}, для которых $\text{argmin}_{akl \in adj(aij)}(g^*(a_{ij})+c(a_{ij}, a_{kl}))$. Таким образом, задача планирования траектории сводится к задаче расчета g^*-значений на МТ-графе.

Очевидно, что расчет g^*-значений для всех клеток МТ-графа – трудоемкий процесс. Фактически данная задача – есть задача обхода МТ-графа. Для сокращения множества клеток, подлежащих обходу, алгоритмы поиска семейства A* используют эвристические функции, оценивающие вес пути из текущей клетки в целевую, следующим образом. Начиная с $a_{startI,startJ}$ обходятся лишь те клетки МТ-графа (выбираемые из некоторого множества кандидатов *OPEN*), которые минимизируют величину $f(a_{ij})=g(a_{ij})+h(a_{ij})$, где $g(a_{ij})$ – вес кратчайшего пути из $a_{startI,startJ}$ в a_{ij}, вычисленный к текущей итерации процедуры обхода, $h(a_{ij})$ – эвристическая оценка веса пути из a_{ij} в $a_{goalI,goalJ}$. Процесс обхода завершается либо при выборе клетки $a_{goalI,goalJ}$, что свидетельствует о нахождении пути, либо при исчерпании множества-кандидатов *OPEN*, что свидетельствует о невозможности построить путь из $a_{startI,startJ}$ в $a_{goalI,goalJ}$ на заданном МТ-графе.

Таким образом, можно говорить о том, что в процессе эвристического поиска вычисляются не g^*-значения вершин графа, а их оценки – g-значения. При этом всегда $g(a_{ij}) \le g^*(a_{ij})$, где $g(a_{ij})$ – вес кратчайшего пути из $a_{startI,startJ}$ в a_{ij}, вычисленный к текущей итерации процедуры обхода. Величина $f(a_{ij})$ является эвристической оценкой веса пути из $a_{startI,startJ}$ в $a_{goalI,goalJ}$, проходящего через a_{ij}.

Процедуры расчета g-значений, восстановления пути по g-значениями, а также схема алгоритма A* приведены на Рис.2.

Алгоритмы семейства A* исследуются достаточно давно. Для них изучена взаимосвязь между используемыми эвристиками и весом найденного пути; установлены критерии нахождения кратчайшего пути. Разработаны модификации алгоритма A*, предназначенные для решения частных задач [5, 6] и др. Однако для задачи планирования траектории, подход, используемый при A*-поиске, а именно подход итерационного рассмотрения вершин МТ-графа, представляется неэффективным по ряду причин.

Во-первых, алгоритмы семейства A* принципиально неспособны избежать рассмотрения

```
Procedure GetPathFromGValues()
a = a_goalI,goalJ                                   (1)
path = a_goalI,goalJ                                (2)
while (a≠a_startI,startJ)                           (3)
        a=argmin_a'∈adj(a)(g(a)+c(a',a))            (4)
        path = a + path                             (5)

procedure ComputeGValues()
a=argmin_a∈OPEN f(a)                                (1)
while (a≠a_goalI,goalJ)                             (2)
        для всех a' из adj(a) выполнить:            (3)
            if g(a') > g(a) + c(a, a') then         (4)
                g(a')=g(a) + c(a, a')               (5)
                добавить (обновить) a' в OPEN с
                соответствующим значением
                f(a')=g(a')+h(a')                   (6)
        a=argmin_a∈OPEN f(a)                        (7)

procedure ASearch()
g(a_startI,startJ)=0                                (1)
g(a)= +∞, ∀a ≠ a_startI,startJ                      (2)
OPEN={a_startI,startJ t}                            (3)

ComputeGValues()                                    (4)
GetPathFromGValues()                                (5)
```

Рис.2. Процедура восстановления пути по g-значениям, расчета g-значений и общая схема алгоритма A*

чрезмерно большого числа клеток при поиске пути на частично проходимых МТ-графах [7]. В частности, когда цель расположена «за» препятствием (Рис. 3а). Именно это проблема общепризнанно считается наиболее серьезной [8] и ограничивающей применение большинства существующих алгоритмов на практике.

Традиционный (и единственно возможный в рамках итерационного подхода) путь к снижению числа рассматриваемых клеток заключается в применении взвешенных эвристик, т.е. эвристик вида $w{\cdot}h(a_{ij})$, $w{\geq}1$, при поиске. Формально это выражается в замене выражения $f(a')=g(a')+h(a')$ на $f(a')=g(a')+w{\cdot}h(a')$ в строке 6 процедуры ComputeGValues (Рис. 2). Алгоритмы, использующие взвешенные эвристики в рамках итерационного подхода принято обозначать как WA*. Доказан тот факт, что использование эвристики с весом w гарантирует построение пути с весом, не более чем в w раз большим веса кратчайшего пути [4]. При этом число рассмотренных клеток как правило сокращается. Однако, как показывают многочисленные эксперименты, данное сокращение не является значительным и не решает проблему (Рис 3б, 3в). Можно утверждать, что все существующие алгоритмы итерационного поиска не способны избежать излишнего рассмотрения клеток, что существенно ограничивает возможности их применения для решения практических задач (в частности – задач навигации БТС).

Во-вторых, итерационный подход при поиске пути влечет принципиальную невозможность создания такой модификации алгоритма А*, которая *одновременно* бы учитывала динамику окружающей среды, ее частичную наблюдаемость, а также позволяла строить неоптимальное решение задачи за короткий промежуток времени и постепенно улучшать его при наличии временных и вычислительных ресурсов. Предлагаемый автором подход позволяет создать такой алгоритм.

2.2. Алгоритм HGA*

Основная идея предлагаемого алгоритма поиска пути HGA* (Hierarchical A* for MT-Graphs) заключается в выделении на МТ-графе частично-упорядоченного множества таких клеток, называемых опорными, что путь между любыми двумя опорными клетками может быть получен стандартными методами. Например, путем построения дискретной прямой по алгоритму Бре-

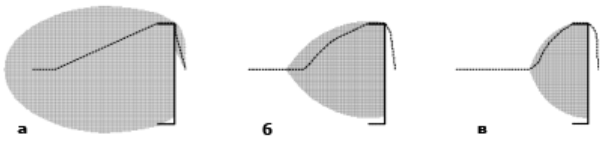


Рис. 3. Число рассмотренных вершин для различных методов итерационного поиска

а) поиск А*
б) поиск с взвешенной эвристикой WA*, w=2
в) поиск с взвешенной эвристикой WA*, w=5

зенхема [9]. Подобный подход имеет много общего с идеями иерархического концептуального планирования, когда исходная задача рекурсивно разбивается на иерархическую сеть подзадач, решение которых известно [10].

Перед тем, как описать алгоритм HGA* введем несколько определений.

Будем говорить, что клетка a_{ij} расположена левее (правее) клетки a_{lk}, если $j<k$ ($j>k$). Будем говорить, что клетка a_{ij} расположена ниже (выше) клетки a_{lk}, если $i<l$ ($i>l$). Заметим, что без ограничения общности можно рассматривать ситуацию, когда целевая клетка МТ-графа расположена не правее начальной. Здесь и далее будем предполагать это условие выполненным.

Секцией будем называть упорядоченную пару клеток МТ-графа $<a_{ij}, a_{lk}>$. Секция $<a_{ij}, a_{lk}>$ – проходима, если на МТ-графе существует хотя бы один путь $\pi(a_{ij}, a_{lk})$. В противном случае, секцию будем считать непроходимой.

Нуль-траекторией между двумя различными клетками a_{ij} и a_{lk} будем называть последовательность смежных клеток МТ-графа $tr(a_{ij}, a_{lk})=\{a_{i0j0}, a_{i1j1}, a_{i2j2}, \ldots, a_{irjs}\}$, такую что:

1. $a_{i0j0}=a_{ij}$, $a_{lk}=a_{irjs}$.
2. Если a_{ij} расположена не правее a_{lk}, то $a_{iv\,jw}$ расположена не правее $a_{iv+1\,jw+1}$ для $\forall\ 0{\leq}v<r$, $0{\leq}w<s$.
3. Если a_{ij} расположена не левее a_{lk}, то $a_{iv\,jw}$ расположена не левее $a_{iv+1\,jw+1}$ для $\forall\ 0{\leq}v<r$, $0{\leq}w<s$.
4. Если $k{\neq}j$, то $\forall\ a_{iv\,jw} \in tr(a_{ij}, a_{lk})$ удовлетворяет соотношению $i_v{=}\lfloor K{\cdot}j_w{+}B\rfloor$, $0{\leq}v{\leq}r$, $0{\leq}w{\leq}s$, где $K=\dfrac{l-i}{k-j}$, $B=l-k{\cdot}K$, а $\lfloor x\rfloor$ - целая часть числа x.
5. Если $k=j$, то $\forall a_{iv\,jw}{\in}\, tr(a_{ij}, a_{lk})$ удовлетворяет соотношению $j_w=j=k$, $i_v=i_{v\text{-}1}+\delta$ для $\forall 0<v{\leq}r$, $0{\leq}w{\leq}s$, где δ=-1 если a_{ij} расположена выше a_{lk}, δ=1 если a_{ij} расположена ниже a_{lk}.

Несмотря на некоторую громоздкость определения, нуль-траектория представляет собой всего лишь отрезок дискретной прямой, проходящей через клетки a_{ij} и a_{lk} (Рис. 4).

Весом нуль-траектории будем называть величину $c(tr(a_{ij}, a_{lk}))$, определяемую аналогично весу пути $\pi(a_{ij}, a_{lk})$. Заметим при этом, что всегда $c(tr(a_{ij}, a_{lk}))= H(a_{ij}, a_{lk})$ и $c(tr(a_{ij}, a_{lk}))\leq c(\pi(a_{ij}, a_{lk}))$.

Модифицируем теперь понятие проходимости секции следующим образом. Секцию $<a_{ij}, a_{lk}>$ будем считать проходимой, тогда и только тогда, когда нуль-траектория $tr(a_{ij}, a_{lk})$ содержит лишь проходимые клетки МТ-графа.

Пусть $a_{ij}, a_{lk} \in A_{m\times n}$, при этом $a_{ij}\neq a_{lk}$, $a_{ij}\neq 0$, $a_{lk}\neq 0$. Частичным путем из a_{ij} в a_{lk} будем называть последовательность клеток МТ-графа $PP=\{a_{i0\ j0}, a_{i1\ j1}, a_{i2\ j2}, \ldots, a_{is\ js}\}$, такую что $a_{i0\ j0}=a_{ij}$, $a_{is\ js}=a_{lk}$. Если при этом каждая из секций $<a_{i0j0}, a_{i1j1}>, <a_{i1j1}, a_{i2j2}>, \ldots < a_{is-1\ j-1s}, a_{is\ js}>$ является проходимой, то частичный путь *PP* будем называть допустимым, в противном случае – недопустимым. Клетки, входящие в частичный путь будем называть опорными. Вес допустимого частичного пути c(*PP)* определим как сумму весов нуль-траекторий между всеми смежным опорным клеткам *PP*, вес недопустимого пути положим равным ∞.

Таким образом, задача поиска пути на МТ-графе сводится к задаче построения допустимого частичного пути, или, что то же самое, к задаче выделения множества опорных клеток на МТ-графе. Для выделения указанного множества предлагается использовать следующую гипотезу о предметной области (в предположении (см. выше), что целевая клетка расположена не левее начальной).

Если начальная и целевая клетки расположены по разные стороны некоторого препятствия, то искомый путь необходимо содержит клетки, расположенные либо выше, либо ниже препятствия.

Использование МТ-графа в качестве модели предметной области, позволяет выразить эту гипотезу на формальном языке и использовать ее для решения задачи построения траектории. Опишем теперь основные принципы функционирования алгоритма HGA*, при этом будем считать, что все препятствия на МТ-графе имеют вид прямоугольников.

Перед началом работы алгоритма происходит формирование частичного пути

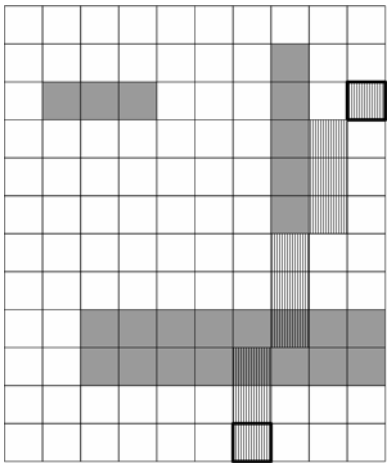
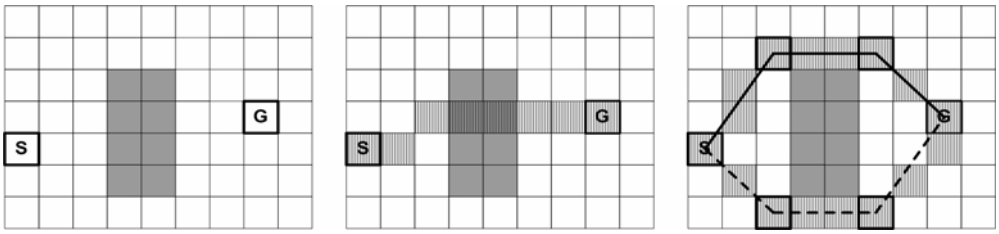


Рис. 4. Нуль-траектория на МТ-графе (помечена вертикальным штрихом)

$PP_0=\{a_{startI,startJ}, a_{goalI,goalJ}\}$ и множества частичных путей – кандидатов на дальнейшее рассмотрение $PC=\{PP_0\}$.

На каждой итерации алгоритма происходит выбор частичного пути PP_i из множества PC и выбор двух опорных клеток a_{ij}, a_{kl} данного пути для построения нуль-траектории между ними (по алгоритму Брезенхема). Если нуль-траектория проходима, то секция $<a_{ij}, a_{kl}>$ помечается как проходимая, осуществляется выбор других опорных клеток из PP_i и цикл построения нуль-траектории повторяется. Если на каком-то этапе все секции PP_i оказываются помеченными как проходимые, то искомый путь считается найденным, и алгоритм прекращает свою работу.

Если нуль-траектория непроходима, то выявляется препятствие, расположенное ближе других к клетке a_{ij}; во временный список заносятся четыре клетки *a*, *b*, *c*, *d*, граничащие с углами препятствия и формируются две проходимые секции $<a, b>$ и $<c, d>$ либо $<a, d>$ и $<b, c>$ в зависимости от направления движения. Используя данные секции, строится два варианта расширения текущего частичного пути, а именно $PP^{(1)}=\{a_{startI,startJ} \ldots, a_{ij}, a, b, \ldots, a_{goalI,goalJ}\}$, $PP^{(2)}=\{a_{startI,startJ} \ldots, a_{ij}, c, d, \ldots, a_{goalI,goalJ}\}$ (либо $PP^{(1)}=\{a_{startI,startJ} \ldots, a_{ij}, a, d, \ldots, a_{goalI,goalJ}\}$, $PP^{(2)} = \{a_{startI,startJ} \ldots, a_{ij}, b, c, \ldots, a_{goalI,goalJ}\}$). PP1 PP2 заменяют в списке кандидатов PC частичный путь PPi и основной цикл алгоритма повторяется. Иллюстративно принципиальная схема работы алгоритма приведена на Рис. 5.



а) б) в)

Рис. 5. Основные шаги алгоритма HGA*

а) выбор опорных клеток для отыскания пути между ними
б) построение нуль-траектории
в) формирование частичных путей – вариантов обхода препятствия

Процедура расширения частичного плана и основная процедура алгоритма HGA* показаны на Рис. 6.

Как видно из схем, приведенных выше, алгоритм HGA* предполагает использование следующих настроечных параметров: CrSort – критерий упорядочивания частичных путей; K – число хранимых в оперативной памяти частичных путей; CrSecSel – критерий выбора секции текущего частичного пути; CrStop – критерий останова.

2.3. Свойства алгоритма HGA*

Прежде чем приступить к анализу описанного алгоритма, хочется отметить его чрезвычайную гибкость. Меняя настроечные параметры, можно добиться не только лучшей «приспособленности» алгоритма для решения конкретной задачи, но и реализовать различные подходы к планированию траектории. Так, при K=1 HGA* представляет собой классический «жадный» метод отыскания решения, который характеризуется чрезвычайно низкими затратами вычислительных ресурсов. С помощью критерия выбора секции CrSel можно менять логику планирования и реализовывать в рамках HGA* такие подходы, как планирование от цели, планирование к цели, двунаправленный поиск [11]. CrSort представляет собой аналог *f*-эвристики из A-поиска, но в отличие от последней может использоваться не только для лучшей адаптации к среде планирования, но и для учета ограничений объекта управления. Так, в некоторых практических задачах целесообразно выбирать частичный путь не с наименьшим весом, а с наименьшим числом секций или – с наибольшим числом «длинных» секций. CrStop – критерий останова – может быть использован для этой же цели. Иногда на этапе планирования траектории не обязательно получать полностью проходимый набор секций (так как все равно в процессе исполнения плана среда может измениться), достаточно построить такой частичный путь, что все его секции имеют вес не больше некоторого Δ.

Логика работы HGA* естественным образом реализует принцип «отсечения по времени» [12]. Т.е. алгоритм удовлетворяет тому требованию, что за короткий промежуток времени строится «приближенное» решение (частичный путь) и постепенно улучшается при наличии временных и вычислительных ресурсов. При этом если соответствующие модификации A* [12] не гарантируют «быстрое» построение даже приближенной траектории на больших графах, то HGA* лишен подобного недостатка, так как на любом этапе работы алгоритм хранит в памяти множество частичных путей кандидатов, «лучший» из которых и представляет собой искомое решение в каждый конкретный момент времени.

Принцип планирования траектории, реализуемый алгоритмом HGA*, приспособлен для решения задач планирования в динамической среде. Более того, для учета динамики окружающей среды в принципиальную схему HGA* не требуется вносить никаких модификаций. Достаточно лишь на этапе сортировки частичных путей учитывать следующие моменты:

- если некоторые ранее проходимые клетки MT-графа стали непроходимыми, необходимо заново разметить все секции частичных путей из множества кандидатов;
- если некоторые ранее непроходимые клетки MT-графа стали проходимыми (или если изменилась целевая клетка), то все частичные планы кандидаты должны быть дополнены секцией <текущая клетка, целевая клетка>.

```
Procedure HGA*(MT-Gr, a_startI startJ, a_startI startJ, K, CrStop,
CrSort, CrSecCel)
A = a_startI,startJ                                          (1)
B = a_goalI,goalJ                                            (2)
BestPath = <A, B>                                            (3)
Paths = {BestPath}                                           (4)
while(BestPlanIsNotGoodEnough(CrStop, BestPlan))             (5)
      Paths = Paths ∪ Extend(BestPlan, CrSecSel)             (6)
      Sort(Paths, CrSort)                                    (7)
      If|Paths|>K                                            (8)
            Truncate(Paths, K)                               (9)
      BestPath=Paths[0]                                      (10)

function Extend(Path, CrSecSel): Paths
resPaths = {∅}                                               (1)
obstSection = ∅                                              (2)
obstSections = {∅}                                           (3)
curPath = ∅                                                  (4)
curSection = ChooseSection(Path, CrSecSelect)                (5)
nullTraj = GetNullTrajectory(CurSection)                     (6)

if(IsFree(NullTraj))                                         (7)
      resPaths = {Path}                                      (8)
else
      obstSections = GetSections(nullTraj)                   (10)
      ForAll(obstSection IN obstSections)                    (11)
            curPath = MakePath(Path, obstSection)            (12)
            resPaths = resPaths ∪ curPath                    (13)

return resPaths                                              (14)
```

Рис. 6. Основные процедуры алгоритма HGA*

Что касается частичной наблюдаемости окружающей среды, то опять же, схема предлагаемого алгоритма не требует принципиальных изменений для учета этого факта. Достаточно лишь на этапе построения нуль-траектории ограничивать ее «длину» (вес) настраиваемым параметром, который следует передавать на вход алгоритму наряду с другими настройками.

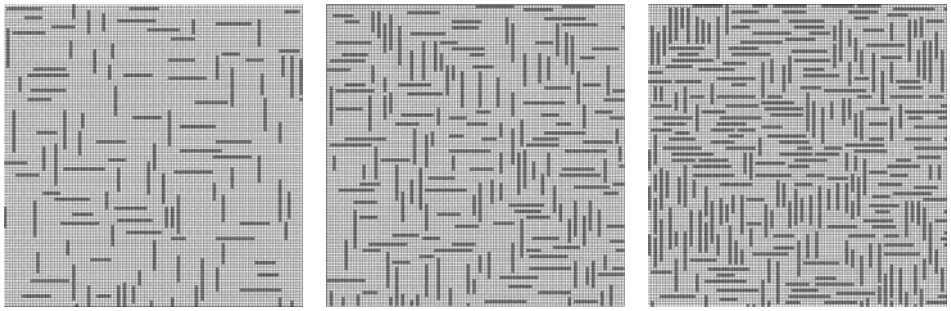
Таким образом, HGA* представляет собой гибкий инструмент, позволяющий эффективно решать задачу планирования траектории на плоскости. При этом в отличие от существующих методов и подходов HGA* позволяет осуществлять планирование в динамической, частично наблюдаемой среде, а также учитывать ограничения объекта управления и соответственно строить оптимальные траектории перемещения не только с учетом единственного критерия – «длины», как в большинстве существующих подходов, а по совокупности критериев.

3. Эксперименты

Для сравнения эффективности существующих алгоритмов планирования траектории на плоскости и предлагаемого был проведен ряд экспериментов. На языке программирования C++ были реализованы алгоритмы HGA*, A* и WA*. С помощью разработанных программных средств были сгенерированы MT-графы размером 50×50, 100×100, 250×250, 500×500, 1000×1000 с разной степенью заполнения препятствиями (СЗП). Препятствия представляли собой прямоугольники шириной в одну клетку и длиной, распределенной равномерно со средним L=10 клеток. Степень заполнения MT-графа препятствиями рассчитывалась по формуле:

$$\lambda=((L*2)+4)*N \qquad (5)$$

Используя выражение (5) для каждого MT-графа определялось необходимое количество препятствий N для достижения параметра λ



а) б) в)

Рис. 7. Примеры МТ-графов с различными значениями параметра λ

а) λ=0,3; б) λ= 0,5; в) λ=0,8

равным 0,3, 0,5 и 0,8. Примеры сгенерированных МТ-графов приведены на Рис. 7.

Всего было создано 150 МТ-графов (Табл.1). Построенные графы представляют собой модельные примеры важной практической задачи – построения траектории перемещения малого беспилотного летательного аппарата (БЛА) типа «вертолет» в городских условиях, когда для обеспечения максимальной скрытности траектория вертолета должна быть эквидистантна земной поверхности и проходить ниже уровня высотных городских строений (т.е. БЛА должен использовать лишь маневры в горизонтальной плоскости для достижения цели).

В ходе экспериментов на каждом МТ-графе случайным образом выбирались из множества крайних клеток начальная и целевая (так чтобы глубина решения - число клеток в кратчайшем возможном пути - *d* равнялась 50, 100, 250, 500 и 1000 соответственно). Для отыскания пути между выбранными клетками применялись следующие алгоритмы:

- A* (в таблицах помечен как «A Search 1»);
- WA* с весом эвристики равным 3 (в таблицах помечен как «A Search 3»);
- WA* с весом эвристики равным 5 (в таблицах помечен как «A Search 5»);
- HGA* (в таблицах помечен как «HGA»).

Как было сказано выше, основным недостатком существующих подходов к построению траектории на плоскости является чрезмерное рассмотрение клеток МТ-графа, которые необходимо хранить в оперативной памяти. Именно поэтому за основной показатель эффективности сравниваемых алгоритмов был взят параметр Q – число рассмотренных (и сохраненных в памяти ЭВМ) клеток МТ-графа. Также отслеживался параметр W – вес построенного пути.

Табл. 1. Характеристики МТ-графов для проводимых экспериментов

СЗП	Размер	Количество МТ-графов
λ=0,3	50 x 50	10
λ=0,3	100 x 100	10
λ=0,3	250 x 250	10
λ=0,3	500 x 500	10
λ=0,3	1000 x 1000	10
λ=0,5	50 x 50	10
λ=0,5	100 x 100	10
λ=0,5	250 x 250	10
λ=0,5	500 x 500	10
λ=0,5	1000 x 1000	10
λ=0,8	50 x 50	10
λ=0,8	100 x 100	10
λ=0,8	250 x 250	10
λ=0,8	500 x 500	10
λ=0,8	1000 x 1000	10
	ИТОГО	**150**

Известно, что алгоритм A* строит кратчайший путь к цели, поэтому помимо абсолютных показателей Q и W рассматривался также интегральный показатель эффективности алгоритма E=(Q/W)/(Q1/W1), где Q1 – число рассмотренных клеток алгоритмом A* для построения пути с весом W1, являющимся кратчайшим. Данный коэффициент позволяет судить о том, во сколько раз рассматриваемый алгоритм эффективней использует вычислительные ресурсы при сохранении «относительного качества» решения (в таблицах с результатами параметр E отражен в процентном отношении).

Результаты экспериментов представлены в Табл. 2- Табл.4.

Анализ полученных данных позволяет сделать два основных вывода.

1. Построение кратчайших путей нецелесообразно на практике из-за чрезмерного расходования вычислительных ресурсов.

Табл. 2. Результаты экспериментов на МТ-графах с параметром λ=0,3

Алгоритм	**Выходные параметры**	**Размер МТ-графа**				
		50x50	**100x100**	**250x250**	**500x500**	**1000x1000**
Asearch 1	Вес пути (W1)	596	1183	2827	5689	11034
	Обработано вершин (Q1)	363	1404	7150	17348	79535
	(Q1/W1)/(Q1/W1)	100%	100%	100%	100%	100%
Asearch 3	Вес пути (W3)	643	1282	3109	6275	12254
	Обработано вершин (Q3)	206	391	947	1907	3757
	(Q3/W3)/(Q1/W1)	53%	26%	12%	10%	4%
Asearch 5	Вес пути (W5)	650	1296	2950	6343	12477
	Обработано вершин (Q5)	206	391	941	1894	3747
	(Q5/W5)/(Q1/W1)	52%	25%	13%	10%	4%
HGA	Вес пути (WH)	608	1206	2908	5937	11490
	Обработано секций (QH)	42	107	252	544	1013
	(QH/WH)/(Q1/W1)	11%	7%	3%	3%	1%

Табл. 3. Результаты экспериментов на МТ-графах с параметром λ=0,5

Алгоритм	**Выходные параметры**	**Размер МТ-графа**				
		50x50	**100x100**	**250x250**	**500x500**	**1000x1000**
Asearch 1	Вес пути (W1)	629,4	1158	2984	6085	11813
	Обработано вершин (Q1)	525	1478	8332	39590	117558
	(Q1/W1)/(Q1/W1)	100%	100%	100%	100%	100%
Asearch 3	Вес пути (W3)	703,2	1261	3339	6803	13592
	Обработано вершин (Q3)	211,8	383	977	1936	3854
	(Q3/W3)/(Q1/W1)	36%	24%	10%	4%	3%
Asearch 5	Вес пути (W5)	709,4	1290	3389	6894	13789
	Обработано вершин (Q5)	208,3	380	961	1926	3807
	(Q5/W5)/(Q1/W1)	35%	23%	10%	4%	3%
HGA	Вес пути (WH)	651	1235	3209	6577	12827
	Обработано секций (QH)	70	187	545	930	2176
	(QH/WH)/(Q1/W1)	13%	12%	6%	2%	2%

Табл. 4. Результаты экспериментов на МТ-графах с параметром λ=0,8

Алгоритм	**Выходные параметры**	**Размер МТ-графа**				
		50x50	**100x100**	**250x250**	**500x500**	**1000x1000**
Asearch 1	Вес пути (W1)	628,4	1235	3024	5938	12377
	Обработано вершин (Q1)	656,1	1733	10281	32410	158328
	(Q1/W1)/(Q1/W1)	100%	100%	100%	100%	100%
Asearch 3	Вес пути (W3)	701,2	1416	3552	6911	14182
	Обработано вершин (Q3)	206,5	384	951	1791	3779
	(Q3/W3)/(Q1/W1)	28%	19%	8%	5%	2%
Asearch 5	Вес пути (W5)	720,6	1442	3662	7008	14405
	Обработано вершин (Q5)	203,8	381	933	1763	3745
	(Q5/W5)/(Q1/W1)	27%	19%	7%	5%	2%
HGA	Вес пути (WH)	682,6	1334	3381	6574	13911
	Обработано секций (QH)	115,9	265	608	1311	2646
	(QH/WH)/(Q1/W1)	16%	14%	5%	4%	1%

2. Алгоритм HGA* позволяет отыскивать пути с меньшим весом быстрей, чем имеющиеся аналоги независимо от размера МТ-графа и степени его заполнения препятствиями.

Отметим особо тот факт, что результаты, показанные алгоритмами WA*-3 и WA*-5, практически идентичны, из чего можно заключить, что дальнейшее повышение веса эвристики не приведет к сокращению множества рассматриваемых клеток. Следовательно, можно предположить, что алгоритм HGA* в среднем в 1,5-4 раза эффективней *любых* имеющихся аналогов, в основу которых заложен принцип итерационного A*-поиска.

Заключение

Задача планирования траектории на плоскости является одной из ключевых при разработке систем управления беспилотными транспортными средствами нового поколения. Для решения этой задачи традиционно используются методы искусственного интеллекта, и именно – различные вариации A*-поиска на графах, в основу которых заложен принцип итерационного обхода с использованием эвристик. К настоящему времени разработаны и исследованы различные модификации алгоритма A* для решения частных задач. Однако ни один из предложенных алгоритмов не в состоянии обеспечить планирование траектории за произвольно короткий промежуток времени, наряду с учетом динамики окружающей среды и ее частичной наблюдаемостью. Также общепризнано, что применение итерационного подхода при построении траектории характеризуется чрезмерным использованием вычислительных ресурсов. Данные обстоятельства существенно ограничивают применение известных методов и алгоритмов планирования траектории на практике.

Для решения указанных проблем предлагается использовать графы специальной структуры – MT-графы, которые позволяют формализовать знания о предметной области и использовать их при планировании. В статье описан новый алгоритм построения траектории на плоскости HGA*, использующий MT-графы в качестве формальной модели. Алгоритм HGA* реализует принципы иерархического планирования и представляет собой чрезвычайно гибкий и эффективный метод построения траектории. При этом HGA* позволяет осуществлять планирование в динамической, частично наблюдаемой среде, а также учитывать ограничения объекта управления. Вычислительная эффективность HGA* и превосходство алгоритма над существующими аналогами были подтверждены результатами проведенных экспериментов.

Литература

1. Осипов Г.С. Методы искусственного интеллекта и задачи управления. Пленарные доклады международной мультиконференции «Теория и системы управления» Москва 26-30 января 2009, М.: Институт проблем управления им. В.А. Трапезникова РАН, 2009.
2. Осипов Г.С., Тихомиров И.А., Хачумов В.М., Яковлев К.С. Интеллектуальные системы управления автономными транспортными средствами: стандарты, проекты, реализации // Авиакосмическое приборостроение №6, 2009, М.: НАУЧТЕХЛИТИЗДАТ, 2009.
3. Яковлев К.С. Графы специальной структуры в задачах планирования траектории. Труды III международной конференции «Системный анализ и информационные технологии САИТ-2009». М: ИСА РАН, 2009.
4. Pearl J. Heuristics: Intelligent Search Strategies for Computer Problem Solving, 1984. Addison-Wesley.
5. Koenig S. et al. Lifelong planning A* // Artificial Intelligence Journal, 155 (1–2), 2004. Elsevier.
6. Likhachev M., Gordon, G., Thrun, S. ARA*: Anytime A* with provable bounds on sub-optimality. In Advances in Neural Information Processing Systems (NIPS), 16, 2003. MIT Press.
7. Яковлев К.С. Методы решения проблемы локального минимума при планировании траектории. Труды IX международной научной конференция им. Таран Т.А. «Интеллектуальный анализ информации ИАИ-2009», Киев: ПРОСВІТА, 2009.
8. Edelkamp S. et al. External-Memory Graph Search. In Proceedings of 18th International Conference on Automated Planning and Scheduling September 14-18, 2008, Sydney, Australia.
9. Bresenham J. E. Algorithm for computer control of a digital plotter // IBM Systems Journal, 4 (1), 1965. IBM.
10. Sacerdoti E. The nonlinear nature of plans. In Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI), 1975.
11. Korf R. E. Artificial Intelligence Search Algorithms. Algorithms and Theory of Computation Handbook. 1996. CRC Press.
12. Likhachev M., et al. Anytime Search in Dynamic Graphs // Artificial Intelligence Journal, 172 (14), 2008. Elsevier.

Яковлев Константин Сергеевич. Инженер-исследователь Института системного анализа РАН. Окончил Российский университет дружбы народов в 2006 году. Автор более 10 печатных работ. Область научных интересов: интеллектуальные системы управления, автоматическое планирование. Адрес электронной почты: yakovlev@isa.ru.