ПРИМЕНЕНИЕ МУРАВЬИНЫХ АЛГОРИТМОВ ДЛЯ ИМИТАЦИОННОГО МОДЕЛИРОВАНИЯ ЗАДАЧИ КОММИВОЯЖЕРА

Д. А. Черкасов (Санкт-Петербург)

Задача коммивояжера

Найти кратчайший гамильтонов цикл в полном взвешенном симметричном графе, где вершины графа – города, а ребра – дороги между городами. Весам ребер соответствуют расстояния c_{ij}.

Задача коммивояжера встречается:

- при решении транспортных проблем;
- при проектировании интегральных схем с большим уровнем интеграции, где требуется соединить между собой компоненты;
- на производстве, где может, например, потребоваться определить кратчайший путь движения детали по конвейеру.

Муравьиные алгоритмы

В основу муравьиных алгоритмов оптимизации положена имитация жизнедеятельности муравьиных колоний. При этом колония рассматривается как многоагентная система, в которой каждый агент (муравей) действует автономно по несложным правилам. Несмотря на простоту поведения каждого агента в отдельности, поведение всей системы получается очень разумным.

Муравьиные алгоритмы основаны на вероятностном выборе. С большей вероятностью будет выбрана вершина (город), если расстояние до нее меньше, чем до других и на пути к ней отложено больше феромона. Кроме того, в ходе алгоритма муравьи откладывают на ребрах феромон, который испаряется со временем. Количество феромона на ребре, ведущем к вершине, также влияет на вероятность ее выбора.

Поскольку в основе алгоритма лежит моделирование передвижения муравьёв по различным путям, такой подход может стать эффективным способом поиска рациональных решений для задач оптимизации, допускающих графовую интерпретацию.

Реализация имитационной модели

Имитационная модель была реализована в среде AnyLogic [5], предоставляющей удобную среду для создания имитационных моделей.

Среда AnyLogic использует язык программирования Java, что позволяет реализовать модели любой сложности. Среда предоставляет удобные средства разработки разных типов моделей, в том числе многоагентных. Кроме того, в AnyLogic встроены богатые средства визуализации результатов.

При разработке модели был использован многоагентный подход [6]. Применение данного подхода позволило описать поведение каждого муравья как агента, который движется по графовой структуре, задаваемой списком вершин и узлов, инициализирующимися при запуске программы.

В модели агентами являются муравьи. Каждому агенту соответствует один муравей.

Состояние агента

Каждый агент обладает собственным состоянием, определяемым полями:

- список вершин, которые необходимо посетить;
- вершина, с которой муравей начал цикл;
- текущая вершина (на этапе выбора следующей вершины);

- вершина, к которой движется муравей;
- ребро, по которому муравей производит движение;
- список посещенных ребер;
- оставшееся время жизни.

Действия на шаге

В системе AnyLogic можно задавать действия для каждого объекта на каждом шаге моделирования. Обновлением «лучшего маршрута» занимается муравей, его нашедший. При достижении агентом вершины вычисляется уровень феромона для только что пройденного ребра и обновляется список посещенных вершин.

Реализация действий агента на шаге приведена ниже (ЯП Java).

```
if ( isActive == false )
    return ;
// Если все вершины пройдены - замкнуть маршрут
if ( startNode != null && nodes_to_visit . size () == 0) {
    nodes_to_visit . add ( startNode );
    startNode = null ;
}
else // Маршрут готов
if ( startNode == null && nodes_to_visit . size () == 0) {
    route . length = pathLen ;
    Main m = ( Main ) getOwner ();
    if (m. shortestP . length > pathLen ) {
        // Обновить кратчайший маршрут, если он найден
        m. shortestP = route ;
        m. isFreshPath = true ;
    }
    if (-- lifeTime == 0) {
        isActive = false ;
    }
    else // Перезапустить муравья для следующей итерации
        reinit ();
    return ;
}
if ( isOnEdge )
    moveTo ( destNode .cx , destNode .cy);
else
{
    nodes_to_visit . remove ( curNode );
if ( nodes_to_visit . size () == 0)
    return ;
// choose the next node
destNode = chooseNode ();
curEdge = getInc ( curNode , destNode );
isOnEdge = true ;
}
```

Выбор следующей вершины

Выбор следующей вершины осуществлется на основе вероятностного правила.

Вероятность перехода муравья из города i в город j :

$$P_{ij,k} = \begin{cases} \dfrac{[\tau_{ij}(t)]^{\alpha} \cdot [\eta_{ij}(t)]^{\beta}}{\sum\limits_{l \in J_{i,k}} [\tau_{il}(t)]^{\alpha} [\eta_{il}(t)]^{\beta}}, & j \in J_{i,k} \\ 0, & j \notin J_{i,k} \end{cases} \qquad 1)$$

α – это параметр, задающий вес следа феромона [3]. При $\alpha = 0$ алгоритм вырождается до жадного алгоритма (будет выбран ближайший город). Параметр β определяет вес расстояния при расчете вероятности.

Ниже приведена процедура выбора вершины (ЯП Java).

```
double P;
ArrayList an = nodes_to_visit ;
ArrayList Pi = new ArrayList ();
Node ni = curNode ;
for (int j = 0; j < an. size (); j ++) {
     double denum = 0;
     Node nj = ( Node ) an. get (j);
     for (int l = 0; l < an. size (); l ++) {
          Node nl = ( Node ) an. get (l);
          denum += pow ( tau (ni ,nl), alpha ) * pow (nu(ni ,nl), beta );
     }
     double num = pow ( tau (ni ,nj), alpha ) * pow (nu(ni ,nj), beta
     );
     Pi. add ( num / denum );
}
probs = Pi;
int n = randSel (Pi);
return ( Node ) an. get (n);
```

Когда муравей прошел все вершины, он «перезапускается». Останов произойдет, когда он пройдет число итераций, равное времени жизни колонии.

На рис. 1 показана работа модели.

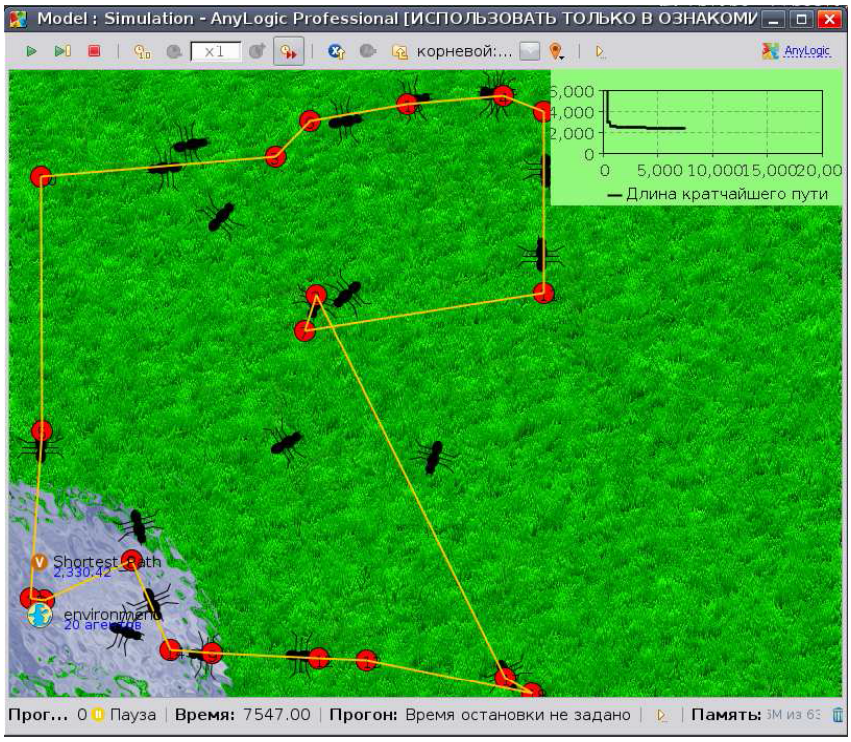

Рис. 1

Эксперимент

Цель эксперимента – установить **скорость сходимости алгоритма** (длину кратчайшего пути) и **зависимость времени решения задачи от количества вершин**.

Входные данные эксперимента

Параметры эксперимента можно разделить на **параметры задачи** и **параметры алгоритма**.

К **параметрам задачи** относится количество вершин и их расположение:

- *случайное;*
- *близкое к выпуклому;*
- *предопределенное.*

К **параметрам алгоритма** относится расположение и количество муравьев, коэффициенты α и β, а также коэффициент испарения феромонов.

В [4] дано оптимальное количество муравьев, равное 10, которое и используется в эксперименте.

Для эксперимента были заданы следующе входные параметры:

- число муравьев – 10;
- расположение муравьев – случайное;
- α = 0.5; β = 2; ρ (коэффициент испарения) = 0.001;
- время жизни колонии – 100;
- расположение вершин – предопределенное.

Скорость сходимости

На рис. 2 можно увидеть как быстро алгоритм сходится к оптимальному решению. Число вершин равно 40.

По оси OX на графике приведено модельное время, а по OY – длина кратчайшего пути.

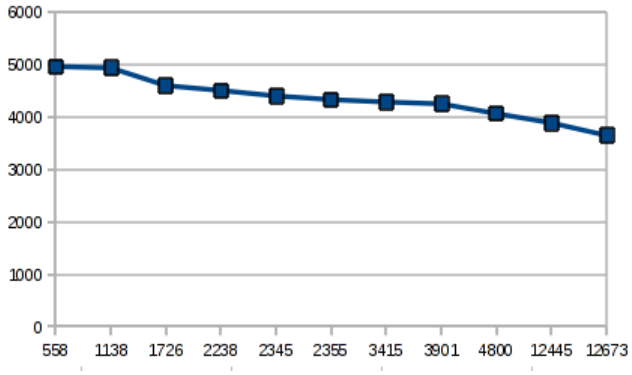

Рис. 2

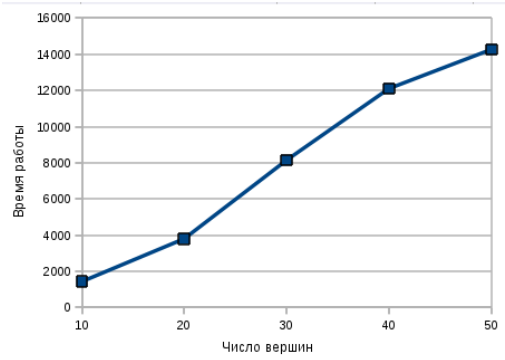

Рис. 3

Зависимость времени решения от размерности задачи

Размерность задачи – число вершин изменялось от 10 до 50 с шагом 10. Каждое моделирование проводилось до тех пор, пока не истекало время жизни колонии. Под временем работы понимается модельное время системы в момент последнего улучшения маршрута.

Зависимость времени работы от размерности задачи близка к линейной (рис. 3). В силу вероятностных свойств алгоритма полученные решения не являются оптимальными, однако достаточно хорошо аппроксимируют идеальные решения (разница составляет не более 10%).

Выводы

В данной работе показана возможность применения многоагентного имитационного подхода для решения поставленной задачи. Результаты экспериментов свидетель-

ствуют об эффективности данного подхода. Разработанная модель наглядно демонстрирует работу алгоритма. Естественными улучшениями данной работы может служить введение ограничений на граф, возможность его изменения на этапе работы модели, улучшения алгоритма, а также реализация других алгоритмов.

Литература

1. **Denebourg J. L., Pasteels J. M. et Verhaeghe J. C.** Probabilistic Behaviour in Ants : a Strategy of Errors? // Journal of Theoretical Biology. 1983. No 105.
2. **Dorigo M., Caro G. Di & Gambardella L. M.** Ant Algorithms for Discrete Optimization //. Artificial Life, 1999. 5 (2). P. 137–172.
3. **Кирсанов М. Н.** Графы в Maple. Задачи, алгоритмы, программы. М: Физматлит, 2007.
4. **Dorigo M. Gambardella L. M.** IEEE Transactions on Evolutionary Computation, Vol.1, No.1, 1997 Ant Colony System: A Cooperative Learning Approach to the Traveling Salesman Problem.
5. Сайт производителя AnyLogic http://www.xjtek.ru/
6. **Карпов Ю. Г.** Имитационное моделирование систем. Введение в моделирование с AnyLogic 5.