УДК 004.048, 004.023
МРНТИ 28.23.19, 28.23.35, 28.23.29

ПОИСК НАИБОЛЕЕ ОПТИМАЛЬНОГО АЛГОРИТМА ДЛЯ РЕШЕНИЯ ПРОБЛЕМЫ ДОСТАВКИ С ВРЕМЕННЫМИ ОКНАМИ

БЕЙБИТУЛЫ М.

Казахстанско-Британский технический университет

Аннотация: *Необходимость оптимизации обслуживания клиентов, снижения эксплуатационных расходов и негативного воздействия на окружающую среду, которое может возникнуть в результате неоптимального планирования транспортных средств и их маршрутов, привлекает внимание научного сообщества к решению этой проблемы в течение последнего десятилетия. Разработка эффективных инструментов для решения проблем в транспортной отрасли может привести к значительному снижению затрат и эффективному потреблению ресурсов. В данной работе рассматриваются и сравниваются алгоритмы, способные обеспечить поиск оптимального маршрута, удовлетворяющего условиям решения проблемы маршрутизации доставки с временными окнами (VRPTW). Представлен анализ наиболее подходящих путей решения, способных уменьшить количество вычисляемых путей и сократить время решения задачи. Были изучены эвристические алгоритмы поиска по графу, а именно: поиск в ширину, алгоритм Дейкстры, Жадный алгоритм и алгоритм A*. Также были представлены способы улучшения каждого алгоритма на языке программирования Python, определены сильные и слабые стороны каждого из предложенных решений и наиболее подходящее для решения проблемы маршрутизации доставки с временными окнами. Результаты работы могут применяться для создания программного обеспечения, решающего проблемы маршрутизации доставки в нефтегазовой индустрии, способствуя оптимизации логистических процессов, что в долгосрочной перспективе положительно скажется на снижении затрат, рациональном потреблении ресурсов и благоприятном влиянии на окружающую среду.*

Ключевые слова: *маршрутизация, доставка, алгоритмы, A*, Python*

УАҚЫТША ТЕРЕЗЕНІ ЖЕТКІЗУ МӘСЕЛЕСІН ШЕШУ ҮШІН ОҢТАЙЛЫ АЛГОРИТМ ІЗДЕУ

Аңдатпа: *Ғылыми қоғамдастықтың назарын соңғы онжылдықта клиенттерге қызмет көрсетуді оңтайландыру, пайдалану шығындары мен көлік құралдарын және олардың жолын субоңтайлы жоспарлау нәтижесінде пайда болатын қоршаған ортаға тигізетін кері әсерді азайту қажеттілігі туралы мәселелерді шешуге аударады. Көлік саласындағы проблемаларды шешудің тиімді құралдарын жасау, шығындарды едәуір төмендетуге және ресурстарды тиімді тұтынуды жүзеге асырады. Бұл жұмыста уақытша терезелерімен (VRPTW) маршруттық жеткізу мәселесін шешудің шарттарын қанағаттандыратын оңтайлы маршрутты іздейтін алгоритмдерді қарастырамыз және салыстырамыз. Есептелген жолдардың санын азайтуға және есепті шешу уақытын қысқартуға болатын, ең қолайлы шешім жолдарының талдауы келтірілген. Эвристикалық графикалық іздеу алгоритмдері зерттелді, атап айтқанда: кеңдік бойынша бірінші іздеу, Дийкстра алгоритмі, ашкөз алгоритм және A * алгоритмі. Сондай-ақ Python бағдарламалау тіліндегі әр алгоритмді жетілдіру жолдары ұсынылды, аталған шешімдердің әрқайсысының күшті және әлсіз жақтары анықталды және уақытша терезелеріне бағыттауды шешуге қолайлылығын көрсетті. Жұмыстың нәтижелері болашақта шығындарды төмендетуге, ресурстарды ұтымды тұтынуға және қоршаған ортаға*

жағымды әсерін тигізетін логистикалық процестерді оңтайландыруға ықпал ететін, мұнай-газ саласындағы жеткізілімдерді бағыттау проблемаларын шешетін бағдарламалық жасақтаманы жасауға пайдаланылуы мүмкін.

Түйінді сөздер: *бағдарлау, жеткізу, алгоритм, A *, Python*

FINDING THE MOST OPTIMAL ALGORITHM TO SOLVE THE DELIVERY PROBLEMS WITH TIME WINDOWS

Abstract: *The need to optimize customer service, reduce operating costs and negative environmental impact that may arise as a result of suboptimal planning of vehicles and their routes attracts the attention of the scientific community to solving this problem over the past decade. The development of effective tools to solve problems in the transport industry can lead to significant cost reductions and efficient resource consumption. In this paper, we consider and compare algorithms that can provide the search for the optimal route that satisfies the conditions for solving the problem of routing delivery with time windows (VRPTW). The analysis of the most suitable solution paths that can reduce the number of calculated paths and reduce the time of solving the problem is presented. Heuristic graph search algorithms were studied, namely: breadth-first search, Dijkstra's algorithm, Greedy algorithm and A * algorithm. It also presented ways to improve each algorithm in the Python programming language, identified the strengths and weaknesses of each of the proposed solutions and the most suitable for solving the problem of routing delivery with time windows. The results of the work can be used to create software that solves the problems of delivery routing in the oil and gas industry, contributing to the optimization of logistics processes, which in the long run will have a positive effect on cost reduction, rational consumption of resources and a favorable impact on the environment.*

Key words: *routing, delivery, algorithms, A *, Python*

Решение проблемы маршрутизации транспортных средств (VRP) и связанных с ней вариантов лежит в основе научных исследований по оптимизации логистического планирования. Одним из важных вариантов VRP является проблема получения и доставки (PDP). В PDP, как правило, требуется найти один или несколько маршрутов с минимальной стоимостью, чтобы обслуживать множество клиентов, при этом два типа услуг могут выполняться в месте нахождения клиента.

Например, большой мебельный магазин может использовать несколько грузовиков для доставки мебели домой. Специализированная компания по утилизации отходов может направлять грузовые автомобили для сбора отходов в ресторанах. Служба здравоохранения может планировать ежедневные визиты осмотра для каждого из своих врачей, проводящих осмотр.

Общей задачей в вышеперечисленных примерах является задача выбора маршрута транспорта. Каждой организации требуется определить, какие заказы, дома, рестораны или пункты осмотра должны обслуживаться каждым маршрутом, грузовиком или врачом и в какой последовательности заказы должны выполняться. Основной целью является наилучшее обслуживание заказов и минимизация общих затрат на эксплуатацию парка транспортных средств [1].

В решении проблемы маршрутизации транспортных средств также существует проблема доставки с временными окнами (VRPTW), когда обслуживание клиента может начаться в пределах временного окна, определенного самым ранним и самым поздним временем, когда клиент разрешит начало обслуживания. Данная работа посвящена исследованию путей решения этой проблемы.

Представим таксопарк или компанию, которая занимается доставкой. Все клиенты

компании нуждаются в одинаковом типе услуг. Каждый маршрут транспортного средства должен начинаться и заканчиваться в одной точке – автопарке, а каждый клиент должен быть посещен ровно один раз.

Для решения этой проблемы необходимо учитывать два пункта.

1. Назначение каждому транспортному средству подмножества узлов из точек, которые указывают клиенты.
2. Планирование маршрутизации, которое включает в себя создание минимальной стоимости маршрута для каждого транспортного средства, чтобы посетить его назначенные запросы. Маршрут должен учитывать пропускную способность и временные ограничения.

Методология интеллектуального решения должна эффективно обрабатывать эти два аспекта и учитывать заданные ограничения, следовательно наиболее подходящим вариантом решения будут эвристические алгоритмы – не всегда точные, но находящие оптимальные решения для задач высокой вычислительной сложности.

Для построения маршрута стоит использовать алгоритмы поиска по графу. Граф – это совокупность непустого множества вершин и связей между вершинами. В математической теории и информатике объекты графа являются вершинами или узлами, а связи – дугами или ребрами. В разных условиях графы могут различаться направленностью, ограничением по количеству связей и дополнительными данными о вершинах и ребрах [2].

Представим сеть автомобильных дорог графом, где перекрестки – это вершины графа, ребра – это дороги, а значения ребер – расстояния по этим дорогам. Существует множество алгоритмов, работающих с графами. В этой работе мы рассмотрим поиск в ширину, алгоритм Дейкстры, жадный алгоритм и A*.

Поиск в ширину исследует заданную область равномерно во всех направлениях. Он часто используется в картах расстояний, но не создает путей, а говорит, как посетить все точки на карте.

Описание алгоритма на Python:

```
frontier = Queue()
frontier.put(start )
visited = {}
visited[start] = True

while not frontier.empty():
   current = frontier.get()
   for next in graph.neighbors(current):
      if next not in visited:
         frontier.put(next)
         visited[next] = True
```

Его можно использовать для решения проблемы маршрутизации, задав значение для отслеживания первоначальной точки относительно каждой посещенной точки. Таким образом простейший алгоритм поиска путей станет указывать путь от заданной цели к началу, где в качестве пути выступает последовательность направления ребер графа.

```
frontier = Queue()
frontier.put(start )
came_from = {}
came_from[start] = None

while not frontier.empty():
   current = frontier.get()
   for next in graph.neighbors(current):
      if next not in came_from:
         frontier.put(next)
         came_from[next] = current
```

Далее нам необходимо находить не все пути, а один наиболее оптимальный. Поэтому для начала нужно прекратить расширять границу поиска по достижению цели.

```
frontier = Queue()
frontier.put(start )
came_from = {}
came_from[start] = None

while not frontier.empty():
   current = frontier.get()

   if current == goal:
      break

   for next in graph.neighbors(current):
      if next not in came_from:
         frontier.put(next)
         came_from[next] = current
```

Но этот метод все равно не подходит, потому что он не учитывает стоимость пути, которую в реальной жизни представляют неровности ландшафта, изгибы дорог и автомобильные пробки. Поэтому нам нужен поиск с равномерной стоимостью.

Алгоритм Дейкстры позволяет следить за стоимостью перемещения относительно начальной точки и оптимизировать маршрут по наименьшим участкам. Для этого необходимо задать приоритеты для очереди исполнения, но в этом случае одна точка может посещаться несколько раз с разной стоимостью. Чтобы этого избежать необходимо изменить логику: теперь вместо добавления к границе ни разу не посещенной точки будет добавляться точка, которая уменьшит стоимость пути [2].

```
frontier = PriorityQueue()
frontier.put(start, 0)
came_from = {}
cost_so_far = {}
came_from[start] = None
cost_so_far[start] = 0

while not frontier.empty():
   current = frontier.get()

   if current == goal:
      break

   for next in graph.neighbors(current):
      new_cost = cost_so_far[current] + graph.cost(current, next)
      if next not in cost_so_far or new_cost < cost_so_far[next]:
         cost_so_far[next] = new_cost
         priority = new_cost
         frontier.put(next, priority)
         came_from[next] = current
```

Но даже в алгоритме Дейкстры граница все равно расширяется по всем направлениям. Чтобы изменить ситуацию можно добавить эвристическую функцию, которая будет определять насколько алгоритм близок к достижению цели.

```
def heuristic(a, b):
   # Manhattan distance on a square grid
   return abs(a.x - b.x) + abs(a.y - b.y)
```

Однако это все еще не соответствует условию оптимального нахождения маршрута за наименьшую единицу времени. Тут нужен другой подход.

Жадный алгоритм, вместо поиска первого наилучшего совпадения, сразу оценивает расстояние до цели. Тут можно использовать очередь с приоритетами из поиска в ширину, но убрать cost_so_far, оставшийся от алгоритма Дейкстры [3].

```
frontier = PriorityQueue()
frontier.put(start, 0)
came_from = {}
came_from[start] = None

while not frontier.empty():
   current = frontier.get()

   if current == goal:
      break

   for next in graph.neighbors(current):
      if next not in came_from:
         priority = heuristic(goal, next)
         frontier.put(next, priority)
         came_from[next] = current
```

Однако, если жадный поиск встречает на перспективном направлении непреодолимые препятствия – затяжную пробку, перекрытую дорогу или стену – он превращается в поиск по ширине, пока не найдет обходной путь, что снова не является кратчайшим и оптимальным решением. Для его поиска необ-

ходимо объединить вычисление расстояния до начальной точки из алгоритма Дейкстры и оценку расстояния по первому наилучшему совпадению из жадного алгоритма [4].

A* не тратит время на исследование бесперспективных направлений и использует оцененное расстояние до цели.

Главное отличие A* в том, что он не использует эвристику для поиска подходящего ответа, потому что эвристика не оценивает расстояния повторно. A* использует эвристику для изменения порядка узлов, чтобы найти узел цели как можно быстрее.

```
frontier = PriorityQueue()
frontier.put(start, 0)
came_from = {}
cost_so_far = {}
came_from[start] = None
cost_so_far[start] = 0

while not frontier.empty():
   current = frontier.get()

   if current == goal:
      break

   for next in graph.neighbors(current):
      new_cost = cost_so_far[current] + graph.cost(current, next)
      if next not in cost_so_far or new_cost < cost_so_far[next]:
         cost_so_far[next] = new_cost
         priority = new_cost + heuristic(goal, next)
         frontier.put(next, priority)
         came_from[next] = current
```

Таким образом алгоритм A* выполняет условие планирования маршрутизации, которое включает в себя создание минимальной стоимости маршрута для каждого транспортного средства, чтобы посетить назначенные клиентами точки. И, при его доработке, способен укладываться в ограничения, заданные проблемой временных окон:

- если клиент обслуживается после верхней временной границы, выбранное решение считается неприемлемым;
- машина ожидает наступления нижней временной границы, если прибыла раньше срока;
- опоздание добавляет штрафное значение к стоимости достижения цели.

A* можно оптимизировать для повышения эффективности и точности выбора путей, посредством анализа проблемы и определения маршрута через представление узлов и установки стоимости в виде статичных и динамических препятствий, изменения расстояния и дополнительных затрат времени, сопутствующих окружающей обстановке [5].

ЛИТЕРАТУРА

1. Ittai Abraham, Daniel Delling, Andrew V. Goldberg, and Renato F. Werneck: Alternative Routes in Road Networks//Journal of Experimental Algorithmics. 2013, 23-34.
2. Rolf H. Mohring and Heiko Schilling: Partitioning Graphs to Speedup Dijkstra's Algorithm// Journal of Experimental Algorithmics. 2007, 189-202.
3. Dominik Schultes: Fast and Exact Shortest Path Queries Using Highway Hierarchies // ESA. 2005, 68-79.
4. Goldberg, A.V., Werneck, R.F.: Computing Point-to-Point Shortest Paths from External Memory //Proceedings of the 7th Workshop on Algorithm Engineering and Experiments. 2005, 26-40.
5. Reinhard Bauer, Daniel Delling, Peter Sanders, Dennis Schieferdecker, Dominik Schultes, and Dorothea Wagner: Combining Hierarchical and Goal-Directed Speed-Up Techniques for Dijkstra's Algorithm?//International Workshop on Experimental and Efficient Algorithms. 2008, 303-318.