Содержание

Аннотация

В работе описывается программа для построения оптимального плана выполнения заданий несколькими аппаратами. Работа выполнена в рамках системы управления АНПА для подводных аппаратов ДВФУ и ИПМТ ДВО РАН. Рассмотрен ряд алгоритмов для реализации такого построения. Проведено их тестирование на сгенерированных и составленных в ручную тестах, после чего наиболее эффективные алгоритмы для различных размеров входных данных внедрены в систему управления.

1 Введение

1.1 Глоссарий

- АНПА – Автономный необитаемый подводный аппарат.
- ИПМТ – Институт проблем морских технологий ДВО РАН.
- MTSP – Multiple Traveling Salesman Problem или множественная задача коммивояжера.
- ГА – генетический алгоритм.

1.2 Описание предметной области

Одна из областей применения автономных необитаемых подводных аппаратов заключается в решении обзорно-поисковых задач. В рамках таких задач аппаратами покрывается некоторая площдь под

водой с целью, например, построения карты с нанесенными результатами измерений, либо с целью поиска и обследования заданных объектов.

В ИПМТ ДВО РАН и ДВФУ давно ведутся исследования по разработке более эффективных методов решения обзорно-поисковых задач с использованием подводных аппаратов.

В настоящее время ведутся исследования методов, которые позволят более эффективно решать обзорно-поисковые задачи засчет интеллектуализации аппаратов. Все больше методов концентрируются на эффективном использовании группы АНПА для решения подобных задач.

1.3 Неформальная поставнока задачи

Первоначальная задача заключалась в разработке и исследовании методов и алгоритмов организации работы в группе АНПА в рамках математической модели, предложенной в работе [].

1.4 Обзор существующих решений

Существует большое множество алгоритмов, использующихся для решения задачи централизованного планирования заданий для группы аппаратов.

Все найденные подходы используют алгоритмы решения известной задачи дискретной оптимизации, MTSP, в разных ее вариантах. Такие подходы описаны например в []. В частности, в работе [] описан подход, основанный на применении целочисленного программи-

рования к решению MTSP с различными исходными положениями аппаратов.

Однако, большинство методов находят приближенное решение множественной задачи коммивояжера, так как точные алгоритмы работают медленно. В работе [] к решению MTSP применен генетический алгоритм, а в [] описаны эвристические методы распределения заданий между аппаратами с последующим решением задачи коммивояжера для каждого из них в отдельности.

В ИПМТ ДВО РАН и ДВФУ для решения таких задач используют решение, описанное в работе []. Данное решение является точным в рамках используемой модели, но работает слишком медленно для необходимого количество заданий и аппаратов. В связи с этим необходимо исследование других подходов к задаче группового управления.

Заказчиком также была предложена новая модель для постановки исходной задачи, в рамках которой и работает вышеупомянутое решение. Она также описана в работе []. Ни один из найденных методов ее не рассматривает, поэтому некоторые из них решено было доработать для решения задачи в рамках новой модели.

2 Математические методы

2.1 Постановка задачи

Предполагается, что имеется m аппаратов и n заданий. Планирование происходит перед началом миссии. Известно, что q-ый аппарат в начальный момент времени находится в точке s_q и готов

к выполнению заданий. Вводится функция $d_q(\mathbf{a}, \mathbf{b})$, обозначающая время перехода АНПА от точки $\mathbf{a}$ к точке $\mathbf{b}$. Она принимает вид $d_q(\mathbf{a}, \mathbf{b}) = d(\mathbf{a}, \mathbf{b}) = |\mathbf{a} - \mathbf{b}|$. Для i-го задания дано v_i вариантов его выполнения и j-ый вариант характеризуется тройкой $(\mathbf{a}_{i,j}, \mathbf{b}_{i,j}, l_{i,j})$, обозначающей соответственно точку начала задания, точку окончания и время его выполнения.

Планом аппарата назван кортеж пар $p = (i_1, j_1), (i_2, j_2), ..., (i_{|p|}, j_{|p|})$ такой, что $i_k \in 1..n, j_k \in 1..v_{i_k}$, для всех $k \in 1..|p|$. Выполнение плана q-м аппаратом начинается в точке $\mathbf{s}_q$ и заканчивается в точке $\mathbf{b}_{i_{|p|}, j_{|p|}}$. Время выполнения аппаратом с номером q плана p, составляет:

$$t_q(p_q) = d(\mathbf{s}_q, \mathbf{a}_{i_1,j_1}) + \sum_{k=2}^{|p|} d_q(\mathbf{b}_{i_{k-1},jk-1}, \mathbf{a}_{i_k,j_k}) + \sum_{k=1}^{|p|} l_{i_k,j_k} \qquad (1)$$

Общим планом является кортеж планов $P = (p_1, p_2, ..., p_m)$ такой, что каждое здание встречается среди его планов один раз и в одном варианте. Время выполнения общего плана определяется выражением:

$$t(P) = \max_{q \in 1..m} t_q(p_q) \qquad (2)$$

Задача состоит в том, чтобы при данных стартовых позициях $\mathbf{s}_q$, известных заданиях и вариантах их выполнения найти общий план P, минимизирующий $t(P)$.

3 Требования

Программа должна:

- Реализовывать и сравнивать несколько алгоритмов решения задачи планирования
- Быть совместимой с уже имеющимся API системы планирования заданий для группы подводных аппаратов ДВФУ.

Лучшие из реализованных алгоритмов для различных размеров входных данных требуется внедрить в уже использующуся систему планирования заданий для подводных аппаратов ДВФУ, работающую под операционными системами семейства Linux. В результате внедрения новых алгоритмов система должна:

- Осуществлять планирование до 100 заданий для группы до 5 аппратов за приемлимое время (до 1 минуты).
- Выбирать более эффективный алгоритм в зависимости от размеров входных данных для планирования.
- Допускается отклонение решения от оптимального на некоторых входных данных, в случае если алгоритм, дающий точное решение неэффективно на них работает.

4 Проект

4.1 Объектно-ориентированная структура

Система состоит из следующих модулей

- Task. Содержит классы:

  - pnt. Содержит основной набор функций для работы с двумерными векторами в евклидовом пространстве.
  - assigned_task. Содержит информацию об использованном варианте задания (начальная, конечная точки и время выполнения).
  - task_t. Содержит набор функций для работы с заданиями.
  - tasks_type. Контейнер для хранения заданий. Содержит функционал для добавления и доступа к заданиям.

- Solver. Содержит единственный класс basic_solver, являющийся базовым ко всем классам, инкапсулирующим различные алгоритмы решения задачи планирования заданий для группы АНПА.

- Plan. Содержит единственный класс plan_t, который инкапсулирует работу с общим планом для всех аппаратов.

- Data. Содержи тединственный класс problem_data, содержащий входные данные к задаче и предоставляющий к ним доступ.

- Visualize. Содержит класс visual_frame, который служит для вывода заданного общего плана для группы аппаратов на экран.

- MILPSolver. Содержит класс milp_solver, в котором находится реализация алгоритма, основанного на целочисленном программировании.

- GreedySolver. Содержит класс greedy_solver, реализующий жадный подход к решению задачи планирования.

- GeneticSolver. Содержит класс genetic_solver, реализаующий генетический алгоритм для решения исходной задачи.
- HKSolver. Содержит класс, содержащий реализацию алгоритма, описанного в asdfasdf

4.2 Алгоритмы

4.2.1 Целочисленное программирование

В настоящем разделе описывается алгоритм нахождения точного решения к поставленной задаче с помощью алгоритмов, использующих методы целочисленного программирвоания. В сравнении с вариантной моделью был сделан ряд упрощений:

- Все задания имеют ровно один вариант выполнения
- У каждого задания начальная и конечная точки совпадают
- Временные затраты на выполнение самих заданий аппаратами пренебрежимо малы
- Скорости аппаратов одинаковы
- Аппараты в начальный момент времени готовы к выполнению миссии
- Для описания координат аппаратов и заданий используется двумерная декартова система координат

4.2.1.1 Сведение исходной задачи к MTSP

Для постановки задачи MTSP необходимо ввести ориентированный граф.

Вершинами графа будут являться задания, стартовые вершины аппаратов и одна фиктивная вершина, необходимая только для постановки задачи коммивояжера.

От фиктивной вершины проведены ребра к стартовым вершинам аппаратов. От стартовых вершин аппаратов проведены ребра к каждой вершине заданий. Вершины заданий соединены ребрами между собой, образуя полный подграф. То есть ребра проведены между всеми парами вершин в обоих направлениях. Также, проведены ребра от вершин заданий к фиктивной вершине.

Весом каждого ребра будет расстояние между координатами заданий либо стартовых вершин. Вес ребер, инцидентных, фиктивной вершине равен нулю.

Решением задачи MTSP будет являться система циклов, удовлетворяющих требованиям MTSP, при этом стоимость максимального цикла будет минимально возможной. Каждому аппарату соответствует ровно один цикл. Двигаясь в этом цикле по направленным ребрам от стартовой вершины и заканчивая в фиктивной вершине, добавляя каждую вершину в план, мы получим оптимальный план этого аппарата.

Вставить пример графа.

4.2.1.2 Поставновка задачи целочисленного программирования

Вводим бинарные переменные $x_{i,j,k}$, где i, j – номера вершин графа, k – номер аппарата. Полагаем $x_{i,j,k}$ равным единице, если в цикле, соответствующему аппарату k, есть ребро (i, j), и нулю – в противоположном случае. Вершина с индексом 0 - фиктивная.

Добавляем следующие ограничения:

- Каждый аппарат должен иметь у себя в цикле ровно одно ребро, входящее в фиктивную вершину и выходящее из нее:

$$\forall k \sum_j x_{0,j,k} = \sum_i x_{i,0,k} = 1 \tag{3}$$

- Сумма ребер, выходящих из любой вершины, кроме фиктивной, должна быть равна сумме ребер, входящих в нее и равна единице:

$$\forall i \neq 0 \sum_j \sum_k x_{i,j,k} = \sum_j \sum_k x_{j,i,k} = 1 \tag{4}$$

Обозначим вес ребра (i, j) как $w_{i,j}$. Тогда стоимость маршрута каждого аппарата:

$$c_k(\mathbf{x}) = \sum_i \sum_j w_{i,j} x_{i,j,k} \tag{5}$$

- В первом варианте алгоритма минимизируемым функционалом является

$$\widetilde{c}(\mathbf{x}) = \max_k \{c_k(\mathbf{x})\} \tag{6}$$

- Во втором варианте мы вводим переменную C_{max}, оптимальное значение котоорой ищем бинарным поиском. На каждой итерации бинарного поиска проверяем существование решения задачи

целочисленного программирования, в которую добавлены ограничения $c_k(\mathbf{x}) \leq C_{max}$ и минимизируемым функционалом является сумма:

$$\widetilde{c}(\mathbf{x}) = \sum_k c_k(\mathbf{x}) \tag{7}$$

$$\widetilde{c}(\mathbf{x}) \to \min \tag{8}$$

4.2.1.3 Неравенства циклов

При вышеописанной постановке задачи целочисленного программирования мы легко можем получить в качестве решения не один, а систему циклов для одного или нескольких аппаратов. Чтобы этого избежать, необходимо вводить дополнительные линейные ограничения исключающие появление систем циклов. В данной работе использовался следующий метод:

1. Ожидаем решения задачи целочисленного программирования.
2. Анализируем решение на наличие систем циклов. Для этого используем поиск в глубину для каждого аппарата.
3. Если поиск в глубину находит цикл, длина которого меньше суммарного количества ребер, задействованных данным аппаратом, значит у данного аппарата в машруте есть несколько циклов.
4. Пусть k – аппарат, у которого найден вышеописанный цикл, A - множество пар (i, j) таких, что ребро (i, j) является частью

цикла. Тогда для данного аппарата добавляем к задаче (3) - (8) следующее линейное ограничение:

$$\sum_{(i,j)\in A} x_{i,j,k} \leq |A| - 1 \tag{9}$$

5. Решаем задачу (3) - (9), переходим к шагу 1.

4.2.1.4 Реализация

Для формулировки задачи целочисленного программирования на языке программирования C++ использовался фреймверк FlopCpp []. Данный фреймверк может использовать различные решатели данных задач. В данной работе использовался решатель SYMPHONY [].

4.2.2 Жадный алгоритм

Жадный алгоритм действует следующим образом.

1. Для каждого задания будем определять, какому аппарату его выгоднее всего отдать на данный момент.

2. Для этого будем назначать задание аппарату q, с текущим планом p_q в том случае, если существует такой вариант задания, что при вставке его в некоторую позицию плана p_q, мы получим новый план $\widetilde{p_q}$ такой, что время его выполнения будет минимально возможным для всех возможных вставок данного задания в план.