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 [1].
- СПУ – Система программного управления.
- ИПМТ – Институт проблем морских технологий ДВО РАН.
- MTSP – Multiple Traveling Salesman Problem или множественная задача коммивояжера [2].
- ГА – генетический алгоритм [3].

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

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

Например, в работе [4] описывается алгоритм использования группы подводных аппаратов для обнаружения шлейфов теплой воды с атомной электростанции в морской среде. Имеется также опыт успешного использования АНПА для поиска затонувших объектов. Так, в работе [5] описывается случай успешного использования автономных подводных аппаратов для обнаружения авиалайнера, потерпевшего крушение в водах Атлантического океана, спустя год после происшествия.

В России исследования по разработке более эффективных методов решения обзорно-поисковых задач с использованием подводных аппаратов ведутся в ИПМТ ДВО РАН и ДВФУ. Первые исследования методов, основанных на централизованном управлении группой АНПА, описаны в работах [6] и [7]. В этих работах аппараты используются для обследования локальных неоднородностей морской среды.

1.3 Неформальная постановка задачи

В настоящее время в ДВФУ организация работы группы аппаратов осуществляется следующим образом:

1. Заранее составляются задания для аппаратов. Отдельное задание может заключаться в прохождении от одной точки подводной среды к другой с заданной скоростью, возможно, с заданной траекторией, с целью съемки дна гидролокатором, замера параметров водной среды и т.п.

2. Далее, система управления аппаратами, запущенная с настольного ПК или ноутбука, определяет множество аппаратов, между которыми необходимо оптимальным образом распределить вышеописанные задания.

3. Затем СПУ запускает алгоритм-планировщик для поиска оптимального распределения заданий между аппаратами. Этот алгоритм для каждого аппарата определит план (последовательность заданий), который аппарату необходимо выполнить. Каждое задание может выполнять только один аппарат, единственный раз.

4. Индивидуальные планы рассылаются аппаратам по сети, после чего начинается их выполнение.

Во время выполнения планов могут возникать непредвиденные ситуации, такие как:

- Появление нового аппарата, готового к выполнению заданий

- Выход аппарата из строя. Это может произойти как по причине поломки, так и ввиду того, что все аппараты используют батареи для электропитания, которые разряжаются в течение нескольких часов.

- Могут появиться новые задания, по причине, например, обнаружения аппаратами новых локальных неоднородностей.

- Существующие задания могут измениться. Может измениться время, которое аппарату необходимо потратить на выполнение задания, могут измениться начальные и конечные точки в виду перемены подводных течений, либо помех из за появления новых объектов в морской акватории.

В случае возникновения любой из вышеперечисленных непредвиденных ситуаций, СПУ составляет план заново, учитывая занятость аппаратами выполнением заданий, которые нельзя прерывать, и принимая во внимания изменившиеся данные для планирования. Обновленные планы аппаратов рассылаются по сети. Каждый аппарат начинает выполнять новый план по завершению последнего задания, в котором был задействован.

В существующей СПУ алгоритм поиска оптимального плана выполнения заданий работает слишком медленно. По этой причине лабораторией необитаемых подводных аппаратов и их систем в ДВФУ была предложена задача, разработать и исследовать новые алгоритмы планирования заданий между группой АНПА с целью их последующего внедрения в систему управления.

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. В такой задаче каждому заданию соответствует единственная точка пространства, которую необходимо посетить единственный раз, только одному аппарату.

Один из вариантов постановки задачи MTSP выглядит следующим образом:

- Имеется граф $G(V, E)$
- $V = v_0, v_1, ..., v_n$ – множество вершин (v_0 - единственная стартовая позиция всех аппаратов, $v_1, ..., v_n$ – координаты точек-заданий)
- E – множество ребер $\{(v_i, v_j)|i \neq j\}$. Каждое ребро (v_i, v_j) означает наличие перехода от задания v_i к заданию v_j. Применительно к подводным аппаратам граф G, как правило, является полным.
- C – матрица весов $c_{i,j}$ каждого ребра (стоимость перехода между заданиями)
- R_k – маршрут k-го аппарата, $k = 1..m$ (неразрывная последовательность ребер)
- $\widetilde{C}(R_k) = \sum\limits_{(v_i,v_j) \in R_k} c_{i,j}$ - стоимость маршрута R_i

Задача состоит в определение такого множества из m маршрутов, что наибольшая стоимость маршрута является минимально возможной:

$$\max_{k=1..m} \widetilde{C}(R_k) \to \min \tag{1}$$

В других вариантах может меняться оптимизируемый функционал, стартовых позиций может быть несколько, и так далее. Подробнее о данной задаче и о методах ее решения можно узнать в работе [2]. В частности, в [8] описан метод, основанный на применении целочисленного программирования к решению MTSP с различными исходными положениями аппаратов.

Однако, большинство подходов находят приближенное решение множественной задачи коммивояжера, так как точные алгоритмы работают медленно. В работе [9] к решению MTSP применен генетический алгоритм, а в [10] 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 задачи планирования используют решение, описанное в работе [6]. 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 [6]. 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

В данном разделе описывается модель, использующаяся для решения задачи планирования в СПУ заказчика.

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 $d_q(\mathbf{a}, \mathbf{b}) = |\mathbf{a} - \mathbf{b}|/u_q$, где u_q - максимальная скорость q-го аппарата. 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.

Время выполнения $l_{i,j}$ может зависеть как от координат $\mathbf{a}_{i,j}$ и $\mathbf{b}_{i,j}$, так и от других параметров. Алгоритм решения задачи планирования никак не учитывает зависимости $l_{i,j}$ от каких-либо параметров, предполагается, что эти величины известны на этапе составления заданий и будут даны на вход алгоритму.

Планом аппарата названа последовательность пар $p = ((i_1, j_1), (i_2, j_2), ..., (i_{|p|}, j_{|p|}))$ такая, 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:

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

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. $|P| = m$ (количество планов совпадает с количеством аппаратов). Время выполнения общего плана определяется выражением:

$$t(P) = \max_{q \in 1..m} t_q(p_q) \qquad (3)$$

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

$$t(P) \to \min \qquad (4)$$

2.2 Алгоритмы

2.2.1 Целочисленное программирование

В настоящем разделе описывается алгоритм нахождения точного решения к поставленной задаче с помощью алгоритмов, использующих методы целочисленного программирования. В сравнении с основной моделью (2) – (4) 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.

Данные упрощения были сделаны для сведения задачи к MTSP (1).

Сведение исходной задачи к MTSP

Введем ориентированный граф $G(V, E)$ аналогично (1.4). И рассмотрим задачу оптимизации (1). В нашем случае множество вершин состоит из m стартовых позиций аппаратов и n заданий. Из всех стартовых позиций существуют переходы к заданиям. Существуют также переходы между любыми двумя различными заданиями. Для сохранения задачи в варианте (1) при различных стартовых позициях, необходимо также ввести фиктивную вершину v_0, из которой проведены ребра к стартовым позициям и в которую ведут ребра из каждого задания:

$$
\begin{aligned}
&E = \{(v_0, v_j) | j = 1..m\} \cup \\
&\cup \{(v_i, v_0) | i = m+1..m+n\} \cup \\
&\cup \{(v_i, v_j) | i \neq j; i, j = m+1..m+n\} \cup \\
&\cup \{(v_i, v_j) | i = 1..m, j = m+1..m+n\}
\end{aligned}
$$

Весом каждого ребра будет расстояние между координатами точек, соответствующих вершинам. Вес ребер, инцидентных, фиктивной вершине равен нулю. На построенном графе необходимо решить задачу (1).

Постановка задачи целочисленного программирования

Вводим бинарные переменные $x_{i,j,k}$, где $i, j = 0..m + n$ – номера вершин графа G, $k = 1..m$ – номер аппарата. Полагаем $x_{i,j,k}$ равным единице, если в цикле, соответствующему аппарату k, есть ребро (v_i, v_j), и нулю – в противоположном случае.

Добавляем следующие ограничения:

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

$$\forall k \sum_j x_{0,j,k} = \sum_i x_{i,0,k} = 1 \tag{5}$$

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

$$\forall i \neq 0 \sum_j \sum_k x_{i,j,k} = \sum_j \sum_k x_{j,i,k} = 1 \tag{6}$$

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

$$c_k(\mathbf{x}) = \sum_i \sum_j w_{i,j} x_{i,j,k} \tag{7}$$

- В первом варианте алгоритма минимизируемым функционалом является
$$\widetilde{c}(\mathbf{x}) = \max_k \{c_k(\mathbf{x})\} \tag{8}$$

- 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:
$$\widetilde{c}(\mathbf{x}) = \sum_k c_k(\mathbf{x}) \tag{9}$$

$$\widetilde{c}(\mathbf{x}) \to \min \tag{10}$$

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 маршруте 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 (5) – (10) следующее линейное ограничение:

$$\sum_{(i,j) \in A} x_{i,j,k} \leq |A| - 1 \tag{11}$$

5. Решаем задачу (5) – (11), переходим к шагу 1.

Аналогичные условия для исключения циклов использовались, например, в работе [11].

Реализация

Решением линейной задачи оптимизации ((10) – (**??**) являются значения переменных $x_{i,j,k}$, доставляющие максимум в целевой функционал (10). Значение этих переменных могут вовсе не быть целыми, хотя мы предполагаем, что $x_{i,j,k}$ принимают только целые значения 0 или 1. Данную проблему решает метод ветвей и границ, позволяющий решать линейные задачи оптимизации в целых числах. Этот метод давно известен (о нем можно прочитать в работе [12]) и реализован в различных библиотеках для языка программирования C++. В данной работе использовался фреймверк SYMPHONY [13], реализующий данный метод. Для формулировки задачи целочисленного программирования на языке программирования C++ использовался фреймверк FlopCpp [14].

2.2.2 Жадный алгоритм

Жадный алгоритм действует следующим образом:

1. Переберем все задания и их варианты.

2. Каждый вариант задания попытаемся вставить в план каждому аппарату.

3. Выберем тот аппарат, после вставки данного варианта задания которому, время выполнения его плана будет наименьшим.

4. Вставим это задание выбранному аппарату в план, в наиболее подходящее место.

5. После вставки методом динамического программирования определим оптимальные варианты всех заданий в плане выбранного аппарата.

Algorithm 1 Жадный алгоритм

for all $t \in T$ **do**
    for all $var \in Vars(t)$ **do**
        for all $v \in V$ **do**
            $time, pos \leftarrow MinPathTime(t, var, Plan(v))$
            if $time < minTime$ **then**
                $minTime \leftarrow time$
                $bestPos \leftarrow pos$
                $bestV \leftarrow v$
                $bestVar \leftarrow var$
            end if
        end for
    end for
    $Insert(t, bestVar, Plan(bestV), bestPos)$
    $OptimizeVars(Plan(bestV))$
end for

Функция $OptimizeVars$ в данном алгоритме устанавливает оптимальные варианты заданий в плане аппарата $bestV$ с помощью метода динамического программирования. Эта функция устроена следующим образом:

Algorithm 2 Оптимальный выбор вариантов

2.2.3 Генетический алгоритм

Хромосома

Мутации

Скрещивание

Отбор

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 аппаратов за приемлемое 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, реализующий генетический алгоритм для решения исходной задачи.
- HKSolver. Содержит класс, содержащий реализацию алгоритма, описанного в [6]

5 Тестирование

5.1 Целочисленное программирование

5.1.1 Бинарный поиск максимума с оптимизацией суммы

5.1.2 Оптимизация максимума

5.2 Жадный алгоритм

5.3 Генетический алгоритм

6 Заключение

Список литературы

[1] Агеев М. Д. Киселев Л. В. Матвиенко Ю. В. Автономные подводные роботы: системы и технологии. — Наука, 2005. — ISBN: 5020335266.

[2] Bektas Tolga. The multiple traveling salesman problem: an overview of formulations and solution procedures // Omega. — 2006. — Vol. 34, no. 3. — P. 209–219.

[3] Wikipedia. Genetic algorithm. — 2002. — URL: http://en.wikipedia.org/wiki/Genetic_algorithm (online; accessed: 2015-06-07).

[4] Cannell Christopher J, Gadre Aditya S, Stilwell Daniel J. Boundary tracking and rapid mapping of a thermal plume using an autonomous vehicle // OCEANS 2006 / IEEE. — 2006. — P. 1–6.

[5] Use of REMUS 6000 AUVs in the search for the Air France Flight 447 / Michael Purcell, Dave Gallo, Greg Packard et al. // OCEANS 2011 / IEEE. — 2011. — P. 1–7.

[6] Туфанов Игорь Евгеньевич и Щербатюк Александр Федорович. Разработка алгоритмов группового поведения АНПА в задаче обследования локальных неоднородностей морской среды // Управление большими системами. — 2012. — Vol. 36. — P. 262–284.

[7] Евгеньевич Туфанов Игорь, Федорович Щербатюк Александр. ОБ АЛГОРИТМАХ ВЫСОКОТОЧНОГО ИЗМЕРЕНИЯ ПА-

РАМЕТРОВ ВОДНОЙ СРЕДЫ, ОСНОВАННЫХ НА ИСПОЛЬЗОВАНИИ ГРУППЫ АНПА // Управление большими системами: сборник трудов. — 2013. — no. 3. — P. 254–270. — URL: http://EconPapers.repec.org/RePEc:scn:022092:14476916.

[8] Kivelevitch Elad H., Cohen Kelly, Kumar Manish. A Binary Programming Solution to the Min-Max Multiple-Depots, Multiple Traveling Salesman Problem // AIAA Infotech@Aerospace (I@A) Conference. — American Institute of Aeronautics and Astronautics, 2013. — 2015/06/06. — URL: http://dx.doi.org/10.2514/6.2013-4665.

[9] Király András, Abonyi János. A novel approach to solve multiple traveling salesmen problem by genetic algorithm // Computational Intelligence in Engineering. — Springer, 2010. — P. 141–151.

[10] Na Byungsoo. Heurisic approaches for no-depot k-traveling salesmen problem with a minmax objective : Ph. D. thesis / Byungsoo Na ; Texas A&M University. — 2007.

[11] Shmoys David B, Williamson David P. Analyzing the Held-Karp TSP bound: A monotonicity property with application // Information Processing Letters. — 1990. — Vol. 35, no. 6. — P. 281–285.

[12] Lawler Eugene L, Wood David E. Branch-and-bound methods: A survey // Operations research. — 1966. — Vol. 14, no. 4. — P. 699–719.

[13] Ralphs Ted K, Güzelsoy Menal. The SYMPHONY callable library for mixed integer programming // The next wave in computing, optimization, and decision technologies. — Springer, 2005. — P. 61–76.

[14] Hultberg Tim Helge. Flopc++ an algebraic modeling language embedded in c++ // Operations Research Proceedings 2006. — Springer, 2007. — P. 187–190.