Математические подходы к решению задачи синтеза новых конфигураций бортовой информационно-вычислительной системы

А.С. Пикалов, e-mail: snk.pik@yandex.ru

Воронежский государственный университет

Аннотация. *В настоящей работе предлагаются принципы построения перспективной бортовой информационно-вычислительной сети с интеллектуальным управлением. Рассмотрены основные математические походы к решению задачи синтеза новых конфигураций бортовой информационно-вычислительной системы Предложенные принципы позволяют значительно повысить надежность и отказоустойчивость комплекса бортового оборудования, что в свою очередь оказывает положительное влияние на безопасность полета в целом.*

Ключевые слова: *Архитектура распределённой информационной вычислительной сети, полностью оптическая сеть, динамическая реконфигурация, экспертная система, система поддержки принятия решения, нейроконтроллер, WDM-сеть, система на кристалле.*

Введение

Рассмотрим математические подходы к решению задачи синтеза новых конфигураций БИВС. При этом, под новыми понимаются конфигурации, отсутствующие изначально в СБЗИП, но структура и состав которых базируются на всех возможных неполных решениях. Для решения задачи синтеза используется информация из пространства архитектур СБЗИП. В силу того, что конфигурации БИВС в пространстве архитектур формализуются с помощью СФГ (1), в качестве математических методов синтеза новых конфигураций целесообразно использовать методы дискретной оптимизации и теории графов.

1. Разработка математической моделей

Для Задачи дискретной оптимизации имеют конечное множество допустимых решений, которые теоретически можно перебрать и выбрать наилучшее (дающее минимум или максимум целевой функции). Практически же зачастую это бывает неосуществимо даже для задач небольшой размерности. В методах неявного перебора делается

попытка так организовать перебор, используя свойства рассматриваемой задачи, чтобы отбросить часть допустимых решений. Наибольшее распространение среди схем неявного перебора получил метод ветвей и границ, в основе которого лежит идея последовательного разбиения множества допустимых решений. На каждом шаге метода элементы разбиения (подмножества) подвергаются анализу – содержит ли данное подмножество оптимальное решение или нет. Если рассматривается задача на минимум, то проверка осуществляется путем сравнения нижней оценки значения целевой функции на данном подмножестве с верхней оценкой функционала. В качестве оценки сверху используется значение целевой функции на некотором допустимом решении. Допустимое решение, дающее наименьшую верхнюю оценку, называют рекордом. Если оценка снизу целевой функции на данном подмножестве не меньше оценки сверху, то рассматриваемое подмножество не содержит решения лучше рекорда и может быть отброшено. Если значение целевой функции на очередном решении меньше рекордного, то происходит смена рекорда. Будем говорить, что подмножество решений просмотрено, если установлено, что оно не содержит решения лучше рекорда. Если просмотрены все элементы разбиения, алгоритм завершает работу, а текущий рекорд является оптимальным решением. В противном случае среди непросмотренных элементов разбиения выбирается множество, являющееся в определенном смысле перспективным. Оно подвергается разбиению (ветвлению). Новые подмножества анализируются по описанной выше схеме. Процесс продолжается до тех пор, пока не будут просмотрены все элементы разбиения.

Очевидно, что задачу синтеза новой конфигурации на основе имеющегося ограниченного набора можно разделить на два этапа:

Построение последовательности (очереди) конфигураций из имеющихся в СБЗИП в таком порядке, который бы обеспечил оптимизацию поиска решения (синтеза);

Модификация выбранной на данной итерации конфигурации, приводящая к синтезу «новой» конфигурации, являющейся решением возникшей проблемы (отказа).

Таким образом, первый этап синтеза аналогичен с математической точки зрения поиску решения задачи коммивояжера. Второй этап, в свою очередь, аналогичен решению задачи поиска минимального остовного дерева. В соответствие с этой особенностью, в качестве математического подхода на первом этапе синтеза используем метод ветвей и границ, а на втором этапе – метод Краскала. Рассмотрим эти методы подробней.

Метод ветвей и границ. Задан полный ориентированный граф $G = (V, E)$ с множеством вершин $V = \{1, \dots, n\}$ и множеством дуг E. Каждой дуге $(i, j) \in E$ приписана длина $C_{ij} \geq 0$. В общем случае задача коммивояжера формулируется на произвольном графе, поэтому длины некоторых (в частности, не существующих) ребер могут быть сколь угодно большими. Так, считаем, что $C_{ij} = +\infty$.

Требуется найти гамильтонов контур минимальной длины. Обозначим:

– $\{i_1, \dots, i_p\}$ – простой путь из i_1, в i_p ;

– $f(i_1, \dots, i_n)$ – длина контура $\{i_1, \dots, i_n, i_1\}$.

Будем считать, что $i_1 = 1$.

Подмножества допустимых решений зададим парой множеств (I, J), где

– $I = \{i_1, \dots, i_p\}$ – частичный маршрут (последовательность посещения первых p городов);

– $J = \{j_1, \dots, j_q\} \subset V \setminus I$ – совокупность запретов на переходы из последнего города i_p частичного маршрута I.

В случае $p = n - 1$ имеем $q = 0$ и пара (I, J) задает одноэлементное множество.

Для случая $p < n - 1$ определим функцию ветвления, которая разбивает множество контуров (I, J) на два подмножества. Первое подмножество включает в себя контуры, в которых коммивояжер из конечного пункта частичного маршрута переезжает в некоторый фиксированный город, а второе – все контуры, в которых этот переезд запрещен.

Для представления множеств разбиения выберем в $V' = V \setminus (I \cup J)$ некоторый элемент i. Если кроме элемента i в V' есть только один элемент k, то искомые множества $(I = \{i_1, \dots, i_p, i\}, J = \emptyset)$ и $(I = \{i_1, \dots, i_p, k\}, J = \emptyset)$. В противном случае первое множество – $(I = \{i_1, \dots, i_p, i\}, J = \emptyset)$, а второе – $(I = \{i_1, \dots, i_p\}, J = \{j_1, \dots, j_q, i\})$.

Опишем один из способов вычисления нижней границы $H(I, J)$ для подмножества (I, J). Будем считать, что $i_2 = 2, \dots, i_p = p < n$. Рассмотрим матрицу $|| C'_{ij} ||, i, j = 1, \dots, n$, в которой $|| C'_{ij} || = +\infty, i = 1, \dots, n, j \in I$ и $C'_{pj} = +\infty, j \in \{1, \dots, p-1\} \cup J$, а все остальные элементы $C'_{ij} = C_{ij}$. Эти изменения в матрице расстояний сделаны для того, чтобы запрещенным маршрутам соответствовали бесконечные длины дуг.

Введем величины

$\alpha_i = \min_{k=1,p+1,\dots,n} C'_{ij}, i = p, \dots, n,$

$$\beta_k = \min_{i=p,\dots,n} \{C'_{ik} - \alpha_i, k = 1, p+1, \dots, n\}$$

и определим нижнюю границу

$$H\ I,J\ =\ \sum_{i=1}^{p-1} c_{i,i+1} + \sum_{i=p}^{n} \alpha_i + \sum_{i=p+1}^{n+1} \beta_i \quad (1)$$

.

Algorithm 1: Dynamic Approach for TSP

Data: *s*: starting point; *N*: a subset of input cities; *dist*(): distance among the cities

Result: *Cost* : TSP result

Visited[N] = 0;

Cost = 0;

Procedure TSP(*N, s*)

  Visited[s] = 1;

  if |*N*| = 2 *and* $k \neq s$ **then**

    Cost(N, k) =*dist(s, k)*;

    Return Cost;

  else

    for $j \in N$ **do**

      for $i \in N$ *and visited[i]* = 0 **do**

        if $j \neq i$ *and* $j \neq s$ **then**

          Cost(N, j) = min (*TSP(N* − {*i*}, *j)* + *dist(j, i)*)

          Visited[j] = 1;

        end

      end

    end

  end

  Return *Cost*;

end

Рис. 1. Метод динамического программирования

В этом алгоритме мы берем подмножество требуемых вершин графа, которые необходимо охватить, расстояние между вершинами и начальная вершина в качестве входных данных. Каждая вершина идентифицируется уникальным идентификатором вершины, например {1,2,3,…,n}.

Изначально все вершины свободны (не охвачены), и охват начинается с вершины S. Предположим, что первоначальное расстояние до вершины равно 0. Затем значение расстояния TSP вычисляется на основе рекурсивной функции. Если количество вершин в подмножестве равно двум, то рекурсивная функция возвращает их расстояние в качестве базового случая.

С другой стороны, если количество вершин больше 2, то мы рассчитаем расстояние от текущей вершины до ближайшей вершины, а минимальное расстояние между оставшимися вершинами вычислим рекурсивно.

Наконец, алгоритм возвращает минимальное расстояние в качестве решения. Здесь мы используем динамический подход для расчета

функции стоимости Cost(). Используя рекурсивные вызовы, мы вычисляем функцию стоимости для каждого подмножества исходной задачи.

В методе Краскала весь единый список ребер упорядочивается по неубыванию весов ребра. Далее ребра перебираются от ребер с меньшим весом к большему, и очередное ребро добавляется к каркасу, если оно не образовывает цикла с ранее выбранными ребрами. В частности, первым всегда выбирается одно из ребер минимального веса в графе.

Для проверки того, что выбранные ребра не образовывает цикл, будем представлять граф, как объединение нескольких компонент связности. В самом начале, когда ни одно ребро графа не выбрано, каждая вершина является отдельной компонентой связности. По мере добавления новых ребер компоненты связности будут объединяться, пока не получится одна общая компонента связности. Пронумеруем все компоненты связности и для каждой вершины будем хранить номер ее компоненты связности, таким образом, в самом начале для каждой вершины номер ее компоненты связности будет равен номеру самой вершины, а в конце у всех вершин будут одинаковые номера компоненты связности, которой они принадлежал.

При рассмотрении очередного ребра посмотрим номера компонент связности, соответствующих концам этого ребра. Если эти номера совпадают, то ребро соединяет две вершины, уже лежащие в одной компоненте связности, поэтому добавление этого ребра образовывает цикл. Если же ребро соединяет две разные компоненты связности, например, с номерами a и b, то ребро добавляется к части основного дерева, а эти две компоненты связности объединяются вместе. Для этого можно, например, всем вершинам, которые раньше находились в компоненте b изменить номер компоненты на a.

Также, для поиска решения на первом этапе синтеза (решения задачи коммивояжера), в случае использования аппаратного ускорителя (нейросетевого сопроцессора) может применяться нейросетевой подход, заключающийся в решении оптимизационной задачи с помощью рекуррентной искусственной нейронной сети.

2. Разработка алгоритма динамического синтеза новых конфигураций

На базе разработанных математических подходах разработан алгоритм, реализующий динамический синтез новых конфигураций БИВС при возникновении нештатных ситуаций. Для повышения гибкости процесса динамической реконфигурации БИВС введем на прикладном уровне дополнительный функциональный элемент

(функциональную задачу в терминах ОСРВ JetOS) – ретранслятор оптических каналов (РОК). В этом случае СФГ произвольной конфигурации из пространства архитектур СБЗИП будет иметь следующий вид (на примере трех ВМ):

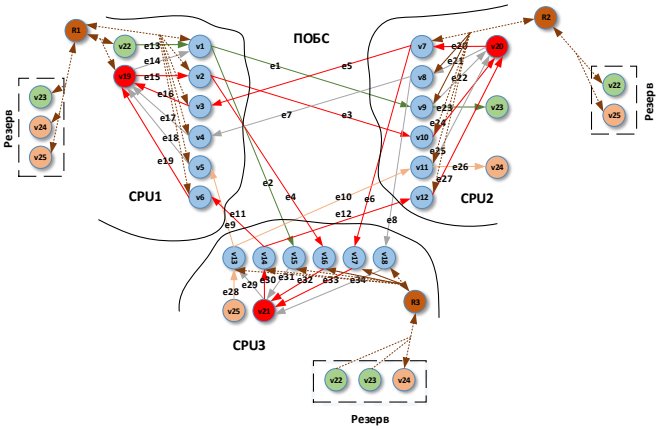

Рис. 2. СФГ конфигурации с РОК

Ра рис. 2 показан СФГ одной из возможных конфигураций БИВС, состоящей из 3-х ВМ (CPU), 36-ти функциональных задач (v), связанных между собой внутри ВМ посредством разделяемой памяти, а между ВМ посредством оптических каналов ПОБС. При этом, задачи v1-v18 отвечают за связь ВМ с ПОБС (через СОУ), задачи {v19,v20,v21} являются задачами диагностики БИВС, задачи {v22,v23,v24,v25} являются задачами общего назначения, а задачи {R1,R2,R3} – это задачи РОК. При этом, каждый ВМ имеет горячий резерв, состоящий из набора копий задач общего назначения. Как видно из рисунка, РОК являются связующим звеном между всеми элементами прикладного уровня БИВС.

Первый этап синтеза новой конфигурации иллюстрирует рис. 3.

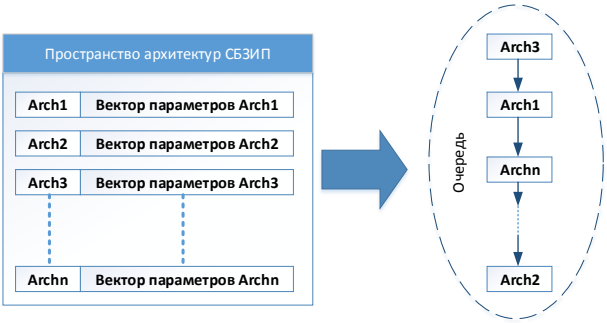

Рис. 3. Первый этап синтеза – упорядочение (создание очереди)

На данном этапе формируется полносвязный граф всех конфигураций, хранящихся в СБЗИП. При этом, вершинами изображаются конфигурации БИВС, а гранями их векторы параметров. Величина связи (грани) определяется как функционал вектора параметров, вида

$$F = \sqrt{k_1 K_a^2 + k_2 K_e^2 + k_3 K_t^2 + k_4 K_p^2 + k_5 K_\sigma^2 + k_6 N_{fch}^2}, \quad (2)$$

Коэффициенты k1 – k6 выбираются пользователем (первичная настройка) и позволяют скорректировать направление поиска (синтеза) новой конфигурации в ту или иную сторону приоритетности ее параметров.

Второй этап синтеза можно проиллюстрировать следующим образом:

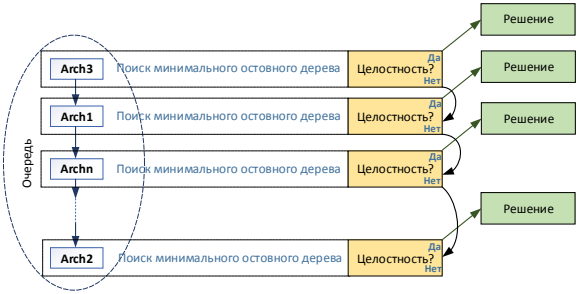

Рис. 4. Второй этап синтеза – поиск/модификация

Заключение

На данном этапе синтеза производится последовательный поиск минимального остовного дерева в графе каждой конфигурации методом Краскала в очередности, сформированной на первом этапе. При это, в процессе поиска участвуют задачи типа РОК, позволяющие найти/достроить недостающие связи в обрабатываемой конфигурации. Первое найденное решение, представляющее собой целостную (восстановленную за счет РОК) конфигурацию останавливает процесс синтеза и передает необходимые параметры в органы управления ЕИС, а также пилоту в виде рекомендаций.

Литература

1. Уколов И.С. и др. Проблемы интеграции комплекса управления ЛА на базе БЦВС. Вопросы кибернетики. Система и методы управления движущимися объектами. М., 1984.

2. Уколов И.С. и др. Проблемы интеграции комплекса управления ЛА на базе БЦВС. Вопросы кибернетики. Система и методы управления движущимися объектами. М., 1984с.