ОБЗОР МЕТОДОВ СТРУКТУРНОГО СИНТЕЗА ДЛЯ РЕШЕНИЯ КВАДРАТИЧНЫХ ЗАДАЧ О НАЗНАЧЕНИЯХ

Кравец О.Я.,
д.т.н., профессор кафедры АВС Воронежского государственного технического университета
spiconf@bk.ru
Сафронова А.П.,
магистрант кафедры АВС Воронежского государственного технического университета
apsafr@mail.ru

Аннотация: *Материалы XVIII Международной открытой научной конференции (Lorman, MS, USA, January 2013)/ Главный редактор, доктор технических наук, профессор, О.Я.Кравец. – Lorman, MS, USA: Science Book Publishing House, 2013.*

AN OVERVIEW OF METHODS FOR THE SYNTHESIS OF STRUCTURAL SOLUTIONS OF QUADRATIC ASSIGNMENT PROBLEM

Kravets O.Ya.,
Doctor of Technical Sciences, Professor of AVS of Voronezh State Technical University
Safronova A.P.,
Undergraduate of AVS of Voronezh State Technical University

Abstract: *Proceedings of the XVIII-th International Open Science Conference (Lorman, MS, USA, January 2013)/ Editor in Chief Dr. Sci., Prof. O.Ya. Kravets. - Lorman, MS, USA: Science Book Publishing House, 2013.*

Введение. Задачи структурного синтеза привлекают к себе огромное внимание специалистов различных областей. В настоящее время остается достаточно актуальным направление, целью которого является разработка универсальных методов структурного синтеза, применимых для различных технических объектов, независящих от отраслевой специфики.

Рассмотрим метод полного перебора и популярные в настоящее время методы структурного синтеза, применяемые для решения квадратичных задач о назначениях: методы локальной оптимизации, моделирование отжига, генетические алгоритмы. Все эти методы, кроме генетических алгоритмов, в каждый момент поиска рассматривают только одно решение, которое неоднократно улучшается во время работы алгоритма.

1. Метод полного перебора

Полный перебор является одним из основных методов решения задач, связанных с поиском оптимальных решений. Полный перебор – это метод решения математических задач, относящийся к методам поиска решений исчерпыванием всевозможных вариантов [1].

Метод полного перебора наиболее прост по своей сути и тривиален в программировании. Для поиска оптимального решения требуется последовательно вычислить значения целевой функции во всех возможных точках, запоминая максимальное из них.

Ключевыми терминами при использовании метода перебора являются счетность и конечность. В этом случае, все элементы множества потенциальных решений можно будет просто пронумеровать от 0 до некоторого N (вполне определенное число для конкретной задачи) и в цикле от нуля до N произвести подстановку вариантов решения в условие.

Трудоемкость данного метода с ростом числа переменных и расширением области граничных условий значительно возрастает. Поэтому для реальных задач он неприменим.

2. Методы локального поиска

Метод локального поиска – это большой класс алгоритмов, который включает в себя метод поиска с запретами, метод локальных улучшений, метод локального поиска с историей и др. Общая схема поиска заключается в следующем: пусть имеется некоторая проблема P, для которой F – возможные варианты решения. Примем $X \in F$ – текущий рассматриваемый вектор, состояние поиска. Определим вектор $N(X) \subset F$, который назовем «соседним» вектором для текущего вектора X.

«Соседними» будут состояния, которые отличаются на одну пару перестановок элементов. Выбор вектора $N(X)$ напрямую зависит от структуры F. Локальный поиск заключается в нахождении локального оптимального решения X' (среди «соседей» $N(X)$) на каждом шаге поиска, допуская, что конечное решение также окажется оптимальным [2].

Принятие локальных оптимальных решений на каждом этапе приводит к нахождению только локального экстремума, поэтому нельзя ограничиться только локальным поиском. Однако различные модификации данного метода способны выходить из локального экстремума. Также локальный поиск активно используется для улучшения решений, полученных, например, методом случайного поиска.

3. Метод локальных улучшений

Одним из вариантов локального поиска является метод локальных улучшений. Алгоритм ищет начальное решение, а затем запускается процедура локального поиска, которая пытается улучшить текущее решение. Основными элементами поиска для метода локальных улучшений являются векторы «соседи» и критерий выбора, который определяет порядок просмотра «соседей». После того, как определены структура соседства и порядок просмотра соседей, выбирают один из трех вариантов критерия, по которому будут проводить попытки улучшения текущего решения:

- метод начальных улучшений;
- метод лучших улучшений;
- метод Heider.

Метод начальных улучшений обновляет текущее решение, как только сформирован «сосед» с лучшим вариантом решения. Метод лучших улучшений просматривает все «соседние» решения и выбирает лучший вариант. Метод Heider просматривает «соседей» в определенном порядке. Решение обновляется, как только просмотрен «сосед» с лучшим вариантом решения.

Метод локальных улучшений повторяет шаги поиска для различных вариантов начального решения. Однако, в настоящее время нет четкого алгоритма формирования начального решения. Bruijs доказал, что нет никакого обоснования того, что, используя поиск из начальных решений, можно достичь глобального экстремума.

Сильной стороной метода локальных улучшений является то, что он быстро находит локальный экстремум.

Недостатками данного метода являются: большая вероятность того, что поиск скатится к локальному экстремуму, необходимость сложной настройки структуры «соседей» и критерия выбора «соседа», большое количество циклов поиска.

Метод локальных улучшений часто внедряют в гибридные алгоритмы, как отдельную процедуру.

4. Метод локального поиска с запоминанием

Другим примером локального поиска является метод локального поиска с запоминанием, в котором реализованы два условия:

- разрешается заходить в состояние с большим значением, чем лучшее на данном шаге поиска;
- поиск запоминает уже просмотренные состояния и больше к ним не возвращается.

В этом методе поиск исследует окрестности минимума и может выходить за его пределы. Такой алгоритм плохо работает при решении задач, в которых имеется плотная концентрация локальных минимумов и велико множество одинаковых значений вокруг глобального экстремума.

Сильной стороной метода поиска с запоминанием является его простота реализации, а слабой стороной данного метода является то, что поиск может обойти

глобальный экстремум в задачах с плотной концентрацией локальных экстремумов.

Метод локального поиска с запоминанием чаще всего используется совместно с другими методами в гибридных алгоритмах.

5. Метод поиска с запретами

На сегодняшний момент наиболее популярным из методов локального поиска является метод поиска с запретами (TS). Fred Glover [3,4] предложил использовать TS для выхода из локальных экстремумов. Glover ввел следующие основные понятия:

- шаг;
- список запретов;
- запрещенный шаг;
- критерий направления.

Шаг – операция, которая генерирует соседей π' для текущего варианта решения π. Список запретов – список шагов, которые нельзя применять к текущему решению. Данный список обновляется во время поиска. Шаги, занесенные в список запретов – запрещенные шаги. Критерий направления – это условие, которое отменяет запрет на шаг после очередного появления шага в списке соседей.

Процедура поиска выглядит следующим образом. Имеется начальное решение. Для этого решения алгоритм выбирает лучший вариант из «соседей», полученных незапрещенными шагами. Важно отметить, что лучший вариант решения из «соседей» необязательно будет лучше текущего варианта решения. Затем обновляется текущее решение, и процедура поиска повторяется. Поиск может сгенерировать одно и то же решение не один раз. Чтобы избежать этого явления, шаги, которые приводят к таким циклам, записываются в список запретов. Однако, запрещая некоторые шаги существует вероятность запрета шагов, которые могут привести поиск к глобальному экстремуму. Чтобы этого избежать, введен критерий направления. Алгоритм останавливается после определенного количества шагов поиска.

В алгоритмах TS заложены два главных недостатка – это поиск из случайно полученного решения, которое не всегда приводит к глобальному экстремуму, и список запретов, в который с большой долей вероятности может быть внесено решение, которое может привести поиск к глобальному экстремуму.

6. Генетический алгоритм

Отцом идеи генетического алгоритма (GA) считается американский ученый John Holland, физик по образованию, работающий одновременно в психологии и в организации вычислительных процессов. Первая его работа по GA появилась в 1962 году, но долго оставалась только идеей. Ему помог его ученик Edgar Codd, ставший широко известным благодаря своим работам по реляционным базам данных. Codd убедил фирму IBM продать Holland по низкой цене компьютер, и у того появилась база для численных экспериментов [5].

GA - это эволюционный метод поиска, используемый для решения задач оптимизации и моделирования путем последовательного подбора, комбинирования и вариации искомых параметров с использованием механизмов, напоминающих биологическую эволюцию [6].

В отличии от TS, в котором поиск производится для одного решения, в GA поиск ведется для совокупности решений. В GA генерируется жизненный цикл: воспроизведение потомства, преемственность и механизм естественного отбора, который сохраняет особи с лучшим критерием.

В начале поиска по GA генерируется совокупность начальных решений, которая называется начальной популяцией. Элемент популяции – особь. Далее алгоритм выбирает пары особей от текущей популяции и, используя правило кроссинговера, получает новое решение (потомок) отдельно для каждой пары. После этого решения с плохим значением критерия удаляются из текущей популяции. Данный процесс повторяется до достижения критерия остановки поиска: время или количество шагов поиска. В ходе поиска периодически применяют мутацию или иммиграцию к текущим популяциям для улучшения значения решения. Часто в GA используют инструменты локального поиска. Такие алгоритмы называют гибридными.

Tate и Smith разработали стандартный GA. В своем алгоритме Tate и Smith реализовали мутацию и скрещивание независимо друг от друга, в отличие от большинства реализаций GA, в которых мутацию используют как вспомогательную процедуру для отдельных особей в популяции. В алгоритме использовалась хромосомная мутация с парным обменом. Tate и Smith создали собственную процедуру скрещивания, которая состояла в следующем. Имеется двое родителей. Объекты, которые закреплены за одинаковыми пунктами у родителей, закрепляются за теми же пунктами и у потомка. В оставшихся пунктах случайным образом слева направо выбирают объект, закрепленный у одного из родителей, исключая при этом повторения объектов, и закрепляют его за соответствующим пунктом. Оставшиеся объекты случайным образом закрепляются за пустыми пунктами. После получения потомков и мутантов происходит отбор по лучшему значению критерия. Размер популяции остается постоянным на каждом шаге поиска.

Явным недостатком GA Tate и Smith является неполное использование идеи естественного отбора (например, не была реализована идея генной мутации). Кроме того, не была произведена настройка параметров GA: не были рассчитаны оптимальный объем популяций и количество хромосом для хромосомной мутации.

Ahuja, Orlin и Tivari разработали «жадный» GA [7]. Для задач с использованием «жадного» GA были получены варианты решения, которые считаются лучшими на сегодняшний день.

В «жадном» GA реализованы следующие идеи: генерирование начальной популяции случайным образом; новые схемы кроссинговера; схема иммиграции; периодическое использование локального поиска для определенных особей из популяции; использование идеи соревнований среди различных популяций и главная идея, которая заключается в том, чтобы сместить баланс от разнообразия к лучшим вариантам решения.

Ahuja, Orlin и Tivari реализовали в своем алгоритме новую схему кроссинговера - путевая схема кроссинговера. Схема имеет следующий вид: пусть имеются двое родителей – I_1 и I_2. При скрещивании этих двух особей получается потомок, которого назовем I_3. Получаем следующий набор хромосом для данных особей:

$$I_1 = a_1 - a_2 - ... - a_n,$$
$$I_2 = b_1 - b_2 - ... - b_n,$$
$$I_3 = c_1 - c_2 - ... - c_n.$$

В любой схеме кроссинговера, потомок наследовал гены, характерные для обоих родителей: если $a_k = b_k$ для некоторого k, то $c_k = a_k = b_k$. В двух путевых схемах кроссинговера Ahuja, Orlin и Tivari определили путь $p_1 - p_2 - ... - p_r$, где каждый элемент пути соответствует особи, элемент p_1 соответствует I_1, а элемент $p_k - I_2$, и для каждого $1 \le k \le r-1$, p_{k+1} получено из p_k с помощью простых преобразований – вставки или смещения. Отсюда и две путевых схемы кроссинговера: со вставкой или со смещением.

В путевой схеме кроссинговера со вставкой процесс скрещивания начинается с некоторой позиции хромосомы, которую определяют случайным образом, и просматриваются хромосомы, соответствующие I_1 и I_2, в цикле слева направо. Если объект уже был просмотрен, то поиск перемещается на следующую позицию, иначе происходит вставка соответствующего объекта из I_2 в I_1 или наоборот, дающая лучший вариант решения. При этом объекты, которые были перемещены, больше не участвуют в данном скрещивании. В результате получаются две дочерних особи: I_1' и I_2', из которых выбирается лучшая и получается I_3.

Ahuja, Orlin и Tivari пришли к выводу, что используемая «жадная» схема в GA выбора лучшего варианта из I_1, I_2, I_3, приводит к быстрой сходимости алгоритма к локальному экстремуму и не дает достаточного разнообразия решений. Авторы «жадного» GA предложили использовать следующий вариант замены: если I_3 лучше I_1 или I_2, то I_3 заменяет соответственно I_1 или I_2, если I_3 хуже I_1 и I_2, то I_3 заменяет худшего из I_1 и I_2.

Мутация необходима для увеличения разнообразия в популяции и заключается в случайном изменении отдельной особи. Ahuja, Orlin, и Tivari предло-

жили использовать иммиграцию для достижения этой цели. Суть иммиграции заключает в замене плохих особей популяции совершенно новыми. Для того, чтобы поиск не сходился к локальному экстремуму, в «жадном» GA реализованы различные схемы генерации иммигрантов, которые вводят совершенно новые особи.

Также Ahuja, Orlin и Tivari применили идею соревнований, которая заключается в запуске нескольких GA с различными начальными популяциями и остановки поиска до того, как он начнет сходиться. В ходе этих процедур получают «лучшую» популяцию, составленную из объединения двух конечных популяций в данной ветви соревнования. Затем запускают GA с «лучшей» популяцией, которая исполняет роль начальной популяции.

Ahuja, Orlin и Tivari предположили, что успех «жадного» GA зависит от правильного смещения баланса от разнообразия к лучшим вариантам решения. Поэтому авторы данного алгоритма и считают, что «жадность» улучшает качество поиска GA.

Недостатками метода «жадного» GA являются сложная настройка параметров и трудная балансировка между разнообразием и «жадностью».

Генетические алгоритмы представляют собой мощный и устойчивый подход для решения крупномасштабных комбинаторных проблем оптимизации. Главной причиной исследования GA в настоящее время является простота реализации параллельного поиска на вычислительных средствах. Главная особенность GA – это большое разнообразие решений на каждом шаге поиска, в отличии от других методов, где на очередном шаге поиска используется только одно решение.

7. Моделирование процесса отжига

Моделирование процесса отжига (SA) – один из первых метаэвристических алгоритмов. Метод основывается на имитации физического процесса, который происходит при кристаллизации вещества из жидкого состояния в твердое, в том числе при отжиге металлов. Предполагается, что атомы уже выстроились в кристаллическую решетку, но еще допустимы переходы отдельных атомов из одной ячейки в другую. Процесс протекает при постепенном понижении температуры. Переход атома из одной ячейки в другую происходит с некоторой вероятностью, причем вероятность уменьшается с понижением температуры. Устойчивая кристаллическая решетка соответствует минимуму энергии атомов, поэтому атом либо переходит в состояние с меньшим уровнем энергии, либо остается на месте.

SA основывается на модели, которую используют для моделирования энергетических уровней при охлаждении твердых тел. В SA закон Больцмана используется для определения вероятности принятия решения:

$$P = \begin{cases} 1, & \Delta E < 0, \\ e^{-\frac{\Delta E}{C_B \cdot t}}, & \Delta E \geq 0, \end{cases}$$

где ΔE – изменение энергии;

t – текущая температура;

C_B – постоянная Больцмана.

Burkard и Rendl доказали, что SA – метод, который может быть применен для решения любой комбинаторной задачи, обладающей структурой соседства [8].

Общая схема SA имеет вид: пусть S – набор решений с целевой функцией f. Примем функцию соседства N, которая определяет для любого $s \in S$ набор соседних решений $N(s) \subseteq S$. Каждое решение $s' \in N(s)$ может быть получено из s путем операции, называемой шагом. Принцип SA прост: начнем с любого случайного начального решения s. Используя s, выбираем соседние решения s' и вычисляем функцию ценности выбора:

$$\Delta f = f(s') - f(s).$$

Если функция ценности выбора улучшена $(\Delta f < 0)$, то заменяем текущее решение новым, т.е. выполняем шаг и используем полученное решение, как отправную точку для следующего шага. Если $\Delta f \geq 0$, то вероятность выполнения данного шага:

$$P(\Delta f) = e^{-\frac{\Delta f}{t}},$$

где t – текущее значение температуры.

Шаг выполняется, если полученная вероятность $e^{-\frac{\Delta f}{t}}$ окажется больше числа, которое случайным образом сгенерировано из интервала $[0,\ 1]$.

Постоянная Больцмана не требуется при решении комбинаторных задач оптимизации.

Процедура повторяется до тех пор, пока не выполнится условие окончания поиска, например, количество шагов достигло определенного значения.

Различные варианты SA отличаются друг от друга в основном по следующим параметрам: метод поиска соседних решений, функция охлаждения и критерий завершения поиска.

Предположим, что:

$$S = \{s \mid s = (s(1), s(2), ..., s(n))\},$$

где n – количество элементов набора решений.

Учитывая, что решение s принадлежит S, k вариант обмена для соседних решений определяется следующим образом:

$$N_k(s) = \{s' \mid s' \in S,\ d(s, s') \le k\},$$

где $d(s, s')$ – функция ценности выбора.

Если $k = 2$ получаем функцию соседства, для которой меняются местами два объекта.

Существуют две альтернативы поиска соседних решений: выбрать случайным образом вариант решения или исследовать всех соседей, которые могут быть получены из текущего решения заданной функцией соседства.

Функция охлаждения определяется следующими параметрами: начальным значением температуры, методом обновления температуры и методом тестирования равновесия.

Ход поиска в SA зависит от температуры t. Самая главная задача – это определение начальной температуры t_0. В случае выбора слишком высокой t_0 поиск замедлится, а при слишком низкой t_0 поиск скатится в локальный экстремум. Поэтому начальная температура должна лежать между этими значениями.

К примеру, можно выбрать начальную температуру, равной:

$$t_0 = \Delta f_{\max},$$

где $\Delta f_{\max}$ – максимальное значение функции ценности выбора между любыми двумя соседними решениями.

Однако, точное вычисление $\Delta f_{\max}$ является довольно затруднительной задачей. Поэтому используют различные методы приближенного вычисления. Конечное значение температуры t_f может быть связано с минимальным значением функции ценности выбора между двумя любыми соседними решениями.

Температура не остается постоянной, а изменяется в процессе поиска. Правильно сформированная функция обновления температуры может гарантировать формирование оптимального режима охлаждения.

Существует множество примеров функции обновления температуры. Hajek [9] предложил использовать следующую функцию обновления температуры:

$$t_k = \frac{const}{\log(k+2)},$$

где $k = 0,\ 1,\ \ldots$ – номер очередного шага поиска.

В этом случается поиск замедляется по сравнению с другими вариантами. Обычно из эвристических соображений процесс остывания ускоряют.

Чаще всего используют функцию обновления температуры, разработанную Kirkpatrick [10]:

$$t_{k+1} = \alpha \cdot t_k,$$

где $k = 0,\ 1,\ \ldots$ – номер очередного шага поиска;

$t_0 = const$ – начальная температура;

$\alpha < 1$ – коэффициент геометрического графика.

Также используют функцию обновления температуры, разработанную Lundy и Mees [11]:

$$t_{k+1} = \frac{t_k}{1 + \beta \cdot t_k},$$

где $k = 0,\ 1,\ \ldots$ – номер очередного шага поиска;

$t_0 = const$ – начальная температура;

$\beta \le t_0$ – коэффициент графика Lundy-Mees.

Нужно отметить, что в настоящее время процесс изменения температуры в SA носит скорее периодический характер, чем монотонный. Т.е. предлагают в SA периодически проводить процесс нагревания и процесс охлаждения.

Для определения условия необходимости уменьшения температуры применяют различные методы тестирования равновесия. Чаще всего температура уменьшается после определенного числа испытаний. Kirkpatrick предложил использовать более сложный критерий уменьшения температуры: отношение числа принятых и отклоненных соседей достигло некоторой константы.

В теории принято, что поиск заканчивает при температуре $t_f = 0$, но на практике используют другие критерии окончания поиска: функция ценности выбора не была уменьшена для определенного количества шагов; число принятых шагов стало меньше некоторой константы; выполнено определенное количество просмотров соседей.

Заключение

К методам, которые лучше себя проявляют в комбинации с другими методами, можно отнести метод локальных улучшений, который используется на ранних стадиях поиска, и метод локального поиска с запоминанием, который применяется для выхода из локального экстремума. В отдельности данные алгоритмы не приводят поиск к хорошим результатам.

Среди методов локального поиска наиболее популярен TS. Основной проблемой использования данного алгоритма является сложность в настройки поиска, которая требует глубокого исследования для каждой конкретной задачи.

Для решения задач с большими размерностями чаще всего используют GA, среди которых на данный момент лучше проявил себя «жадный» GA, разработанный Ahuja, Orlin и Tivari. Однако и у этого алгоритма имеются недостатки – это сложная настройка параметров поиска и трудная балансировка между разнообразием и «жадностью». Но, несмотря на эти недостатки, GA имеют большой потенциал - наличие большого разнообразия вариантов решения на каждом шаге поиска и возможность реализации параллельного поиска на вычислительных системах.

Метод SA, наверное, самый простой в реализации. Ценность данного метода – простота настройки поиска.

Список литературы

1. Электронная энциклопедия Wikipedia. – Электрон. дан. – Режим доступа: http://ru.wikipedia.org/wiki/Полный_перебор.
2. Cela E. The quadratic assignment problem: special cases and relatives / E. Cela. – Graz, Austria: Graz University of Technology, 1995. – 265 p.
3. Glover F. Tabu search - Part I / ORSA Journal on Computing. – 1989. – Vol. 1. – P. 190 – 206.
4. Glover F. Tabu search - Part II / ORSA Journal on Computing. – 1989. – Vol. 2. – P. 4 – 32.
5. Электронный сайт СПбГУ. – Электрон. дан. – Режим доступа: http://www.math.spbu.ru/user/jvr/DA_html/_lec_2_11.html.
6. Электронная энциклопедия Wikipedia. – Электрон. дан. – Режим доступа: http://en.wikipedia.org/wiki/genetic_algorithm.
7. Ahuja R.K., Orlin J.B., Tivari A. A greedy genetic algorithm for the quadratic assignment problem / Working paper Sloan School of Management: – 1995. – Vol. 2. – P. 13 – 17.
8. Burkard R., Rendl F. A thermodynamically motivated simulation procedure for combinatorial optimization problems / European Journal of Operational Research: – 1983. – Vol. 17. – P. 169 – 174.
9. Hajek B. Cooling schedules for optimal annealing / of Operations Research: – 1988. – Vol. 13. – P. 311 – 329.
10. Kirkpatrick S., Gelatt C.D., Vecchi M. Optimization by simulated annealing / Science: 1983. Vol. 220. P. 671-680.
11. Lundy M., Mees A. Convergence of an annealing algorithm /Mathematical Programming: – 1986. – Vol. 34. – P. 111 – 124.