3.2 Метаэвристики

А. А. Мельников

Эвристиками называют обычно достаточно простые общие правила решения задач. В качестве примеров можно привести жадные эвристики, эвристики округления дробных решений для задач с целочисленными переменными, специфичные для задач правила, которые позволяют строить относительно качественные решения или их фрагменты. Термин *метаэвристика* используется для общих подходов к решению оптимизационных задач. Эти подходы объединяет нацеленность на быстрое отыскание качественных решений. При этом априорные гарантии качества найденных решений обычно предложить невозможно, однако для многих задач таких как задачи комбинаторной оптимизации, задачи оптимизации черного ящика и других метаэвристические подходы оказываются одними из основных, когда речь заходит о примерах, возникающих на практике.

Метаэвристики можно классифицировать по числу рассматриваемых одновременно решений. В популяционных метаэвристиках, к которым относятся генетические алгоритмы, алгоритмы роя частиц, алгоритмы муравьиных колоний и другие, на каждой итерации рассматривается множество решений — популяция — внутри которой происходит обмен информацией и другие изменения, направленные на улучшение качества. Их дополнением является класс метаэвристик, работающих с единственным решением (single-solution based), такие как поиск с запретами, имитация отжига и другие многочисленные модификации локального поиска.

Ключевыми понятиями, участвующими в разработке метаэвристик, являются исследование пространства поиска (диверсификация) и использование информации о лучших найденных решениях для локализации поиска (интенсификация). Диверсификация подразумевает возможность поиска не зацикливаться в ограниченном регионе и способность покидать его для исследования других областей. Интенсификация в свою очередь предполагает более интенсивный поиск в области концентрации хороших решений.

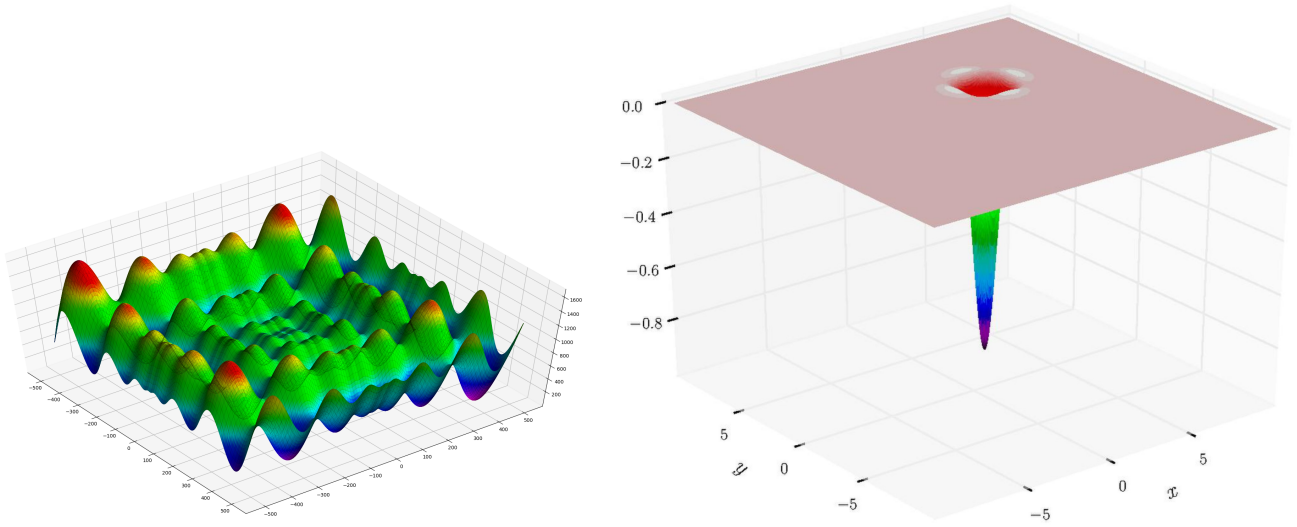
С точки зрения устройства, популяционные эвристики имеют более диверсификационные компоненты, а алгоритмы локального поиска — более интенсификационные, и баланс ингредиентов подбирается исходя из свойств решаемых задач. Одним из ключевых свойств является ландшафт оптимизируемой функции (см. рис. 1). С яркими примерами функций со специфическими свойствами ландшафта можно ознакомиться на Википедии. Устройства ландшафта можно исследовать, отслеживая такие характеристики как распределение локальных оптимумов, которое может быть относительно равномерным, одно- или многокластерным. Характер распределения может подсказать эффективные стратегии поиска.

1 Основные ингредиенты

В дизайне метаэвристики выделяют две основных части: представление решения и вид функции, направляющей поиск. Представление решения должно (на практике это не всегда требуется) обладать свойствами

- полноты — все допустимые решения должны быть представимы
- связности — любые два решения должны быть достижимы одно из другого
- эффективности — алгоритм должен быстро оперировать представлениями решений.

Рис. 1: Ландшафты функции: множество локальных оптимумов функции Розенброка (слева) и плато функции Изома (справа)



Примерами представлений или *кодировок* решений могут служить

- (0,1)–векторы для булевой задачи о рюкзаке, задачи о выполнимости и других задачах (0, 1)–программирования
- целочисленные векторы для обобщенной задачи о назначениях (номера рабочих, выполняющих каждое из заданий)
- числовые векторы для задач непрерывной оптимизации или задач настройки непрерывных гиперпараметров
- перестановки для задач маршрутизации и некоторых задач теории расписаний (в какой последовательности выполнять работы или посещать клиентов).

Естественным выбором для функции, управляющей поиском, является оптимизируемая функция. К примеру, в задаче маршрутизации необходимо минимизировать длину маршрутов, посещающих клиентов, и алгоритм, перебирая перестановки, кодирующие последовательность обхода клиентов, может восстанавливать соответствующий маршрут и ориентироваться на его длину.

Однако для многих задач такая функция не задана или недостаточно информативна. Например, задачи удовлетворения ограничений, в которых требуется подобрать значения переменных, для которого все ограничения были бы выполнены, не имеет целевой функции (или она принимает только два значения: 0, когда ограничения не удовлетворены, и 1, когда удовлетворены). В такой ситуации можно задать веса каждому из ограничений и максимизировать суммарный вес выполненных ограничений.

Другим примером задачи, в которой целевая функция не информативна, может служить задача упаковки в контейнеры, где необходимо разместить предметы с заданным весом в минимальное число одинаковых контейнеров таким образом, чтобы суммарный вес предметов в каждом контейнере не превосходил бы заданную вместимость. Для такой задачи естественной кодировкой была бы кодировка в виде целочисленных векторов:

каждому предмету сопоставляется номер контейнера, в котором он лежит. Сложность, связанная с таким выбором, состоит в том, что модификации решения, которые предпринимает алгоритм во время поиска — перекладывание предмета из одного контейнера в другой или обмен предметами между контейнерами — во многих ситуациях не меняют значение целевой функции: число использованных контейнеров не меняется. Алгоритм в такой ситуации сталкивается с большим количеством одинаковых решений, как на рис. 1 справа, и нуждается в дополнительных ориентирах. Такими ориентирами могут быть, к примеру, суррогатные целевые функции вида $\sum_{c \in C} \sqrt{w_c}$, где C — множество контейнеров, а w_c — доля заполненности контейнера c. Такая функция уменьшается, когда предмет перемещается из менее заполненного контейнера в более заполненный, и алгоритм может целенаправленно уплотнять контейнеры даже, если общее число использованных контейнеров при этом не меняется.

2 Метаэвристики, оперирующие одним решением

К таким методам относят различные модификации локального поиска. Они итеративно применяют к текущему решению процедуры генерации и замещения. На фазе генерации строится множество кандидатов на роль нового текущего решения. Такое множество строится путем локальных трансформаций имеющегося решения. Одно из сгенерированных решений замещает текущее, и начинается новая итерация. Процедуры замещения и генерации могут не использовать историю поиска, либо опираться на нее для интенсификации.

Упомянутые локальные трансформации связаны с понятием *окрестности*, то есть подмножества решений $N(s)$, считающихся соседними для текущего решения s. Типичными окрестностями является замена одного бита в (0, 1)–кодировке (flip) или перемещение единицы в ней с одного места на другое (swap). Для перестановочных кодировок соседними могут считаться перестановки, отличающиеся в двух произвольных позициях или в двух подряд идущих позициях. При этом, если кодировка имеет длину n, то мы в первом случае будем иметь n^2 соседних решений, а во втором n.

Помимо относительно компактных окрестностей, которые можно исследовать непосредственно, может оказаться оправданным использование окрестностей большого размера. Они позволяют провести существенные и направленные модификации текущего решения. При этом непосредственный просмотр всех соседних решений в такой окрестности невозможен. Вместо этого формулируется специальная оптимизационная задача, решение которой — точное или приближенное — помогает отыскать качественного представителя большой окрестности. Больше об этом можно найти в поиске по ключевым словам large neighborhoood search.

Общий вид метаэвристики рассматриваемого класса можно сформулировать в виде следующей схемы

1. Вход: начальное решение s_0
2. $t = 0$
3. Повторять до выполнения критерия остановки
   (a) Сгенерировать кандидатов $C(s_t)$ — часть окрестности текущего решения s_t
   (b) Выбрать решение s_{t+1} из $C(s_t)$ для замещения s_t.
4. Вернуть лучшее найденное решение.

Диверсификация поиска может производиться путем перезапуска алгоритма с нового решения, случайного прыжка к решению достаточно удаленному от текущего или временному переводу алгоритма в режим, близкий к случайным блужданиям. Интенсификация может быть реализована с помощью возврата к качественным решениям или переводу алгоритма в режим более тщательного просмотра окрестностей.

Разберем схему имитации отжига с точки зрения особенностей реализации поисковых инструментов. Алгоритм использует числовой параметр, называемый температурой, для принятия решений о переходе к решениям с ухудшением целевой функции.

1. Вход: график снижения температуры
2. Сгенерировать стартовое решение $s = s_0$
3. Задать начальную температуру $T = T_{max}$
4. Повторять до выполнения критерия остановки $T < T_{min}$
   (a) Повторять до установления равновесия при данной температуре
      i. Сгенерировать случайное соседнее решение s'
      ii. Вычислить улучшение целевой функции Δf
      iii. Если $\Delta f \geq 0$, положить $s = s'$. Иначе положить $s = s'$ с вероятностью $e^{-\frac{\Delta f}{T}}$
   (b) Изменить температуру
5. Вернуть лучшее найденное решение.

В качестве механизма диверсификации поиска выступает вероятностный критерий принятия ухудшающего решения: на начальном этапе поиска, когда температура высока, алгоритм достаточно часто переходит в новые решения с ухудшением целевой функции. Это позволяет алгоритму изучить допустимую область. Позднее температура снижается, ухудшающие решения принимаются реже, и алгоритм переходит в режим интенсификации, поскольку при более низкой температуре просматривается больше соседних решений для осуществления перехода.

3 Популяционные метаэвристики

Данный класс алгоритмов оперирует сразу несколькими решениями, множество которых называется популяцией. Общая схема организации таких алгоритмов также, как и в случае алгоритмов, оперирующих единственными решением, подразумевает обновление популяции последовательным применением процедур генерации и замещения. В отличие от алгоритмов с одним решением, операторы, генерирующие новые решения, могут опираться на информацию от нескольких решений. Для иллюстрации приведем и обсудим шаблонную схему эволюционного алгоритма

1. Сгенерировать начальную популяцию P_0
2. $t = 0$
3. Повторять до выполнения критерия остановки
   (a) $evaluate(P_t)$
   (b) $P'_t = selection(P_t)$
   (c) $P'_t = reproduction(P'_t)$; $evaluate(P'_t)$
   (d) $P_{t+1} = replace(P_t, P'_t)$
   (e) $t = t + 1$

4. Вернуть лучшее найденное решение.

В процедуре *selection* особям сопоставляется характеристика приспособленности, которая может быть выражена значением целевой функции или номером особи в упорядоченном по значению целевой функции списке особей. Далее происходит сам выбор, организованный в соответствии с некоторой стратегией. Например, особь может отбираться с вероятностью пропорциональной ее приспособленности. В турнирной селекции случайным образом выбирается k особей, наиболее приспособленная из которых включается в популяцию следующего поколения, и процедура повторяется требуемое число раз.

Процедура *reproduction* подразумевает применение операторов мутации и скрещивания. В качестве мутаций применяют локальные модификации решения такие как flip для (0, 1)–кодировок. Примером оператора скрещивания является процедура одноточечного скрещивания, которая для двух родительских (0, 1)–векторов производит два решения-потомка, имеющих начальную часть одного родителя и конечную часть второго.

Для диверсификации поиска таким образом служат процедуры мутации и скрещивания, а также общая идея метода оперировать одновременно целым набором решений. Интенсификация может быть реализована с помощью использования урезанных вариаций локального поиска на фазе *reproduction*, когда решения–потомки могут быть локально улучшены перед включением в популяцию следующего поколения.

Для дальнейшего знакомства с великим разнообразием метаэвристик рекомендуем к изучению учебники [1—3].

Список литературы

[1] J. Dréo и др. *Metaheuristics for Hard Optimization: Methods and Case Studies*. Springer Berlin Heidelberg, 2006.

[2] Fred W. Glover и Gary A. Kochenberger. *Handbook of Metaheuristics*. 2019.

[3] El-Ghazali Talbi. “Metaheuristics - From Design to Implementation”. В: 2009.