УДК 519.658

МЕТАЭВРИСТИЧЕСКИЕ АЛГОРИТМЫ ДЛЯ ЗАДАЧ КОМБИНАТОРНОЙ ОПТИМИЗАЦИИ (ОБЗОР)

© О. А. Щербина

Таврический национальный университет им. В.И. Вернадского
факультет математики и информатики
пр-т Вернадского, 4, г. Симферополь, 295007
e-mail: *oshcherbina@gmail.com*

Metaheuristic algorithms for combinatorial optimization problems (Review).

Shcherbina O. A.

Abstract. We survey metaheuristic algorithms that perform directed random searches of possible solutions of combinatorial optimization problems, optimal or near optimal, until a particular termination condition is met or after a predefined number of iterations. Metaheuristics combine basic heuristic methods in higher level frameworks aimed at efficiently and effectively exploring a search space. Metaheuristics fall in two categories: local search metaheuristics and evolutionary algorithms. In this paper, we describe the major solution methods: Local Search Metaheuristics (Simulated Annealing, Tabu Search, Greedy Randomized Adaptive Search Procedure (GRASP), Variable Neighborhood Search) and Evolutionary Algorithms (Genetic Algorithms, Ant Colonies Optimization).

Введение

Постановка проблемы в общем виде и ее связь с важными научными или практическими задачами

Анализ последних исследований и публикаций оптимизации показывает, что использование моделей и алгоритмов комбинаторной оптимизации (КО) позволяет решать многие практические задачи, поскольку дискретные оптимизационные модели адекватно отражают нелинейные зависимости, неделимость объектов, учитывают ограничения логического типа и всевозможные технологические, в том числе и имеющие качественный характер, требования. Согласно Papadimitriou и Steiglitz [39], задачей комбинаторной оптимизации (КО) $\mathscr{P} = (\mathscr{S}, f)$ называется задача оптимизации, в которой задано конечное множество объектов $\mathscr{S}$ и целевая функция $f : \mathscr{S} \to \mathbb{R}^+$, которая назначает положительное значение стоимости для каждого из объектов

$s \in \mathscr{S}$. Цель состоит в том, чтобы найти объект с минимальным значением стоимости. Объектами, как правило, являются целые числа, подмножества множества элементов, перестановки множества элементов или графовые структуры. Примером задачи КО может служить известная задача коммивояжера [32]. Другие примеры задач КО: задачи о назначениях, задачи составления расписаний, а также задачи планирования. К сожалению, большинство интересных задач КО являются NP-трудными и точное решение их в худшем случае может требовать построения дерева поиска решений экспоненциального размера. В связи с практической значимостью задач КО, для их решения разработан ряд алгоритмов, которые могут быть классифицированы как точные или приближенные алгоритмы. Точные алгоритмы гарантированно находят оптимальное решение для любой задачи КО конечного размера за ограниченное время (см. [37], [39]). В связи с этим чрезвычайно актуальны разработка и исследование приближенных, в том числе метаэвристических, алгоритмов для решения задач КО.

Метаэвристики являются мощным и чрезвычайно популярным классом оптимизационных методов. позволяющих находить решения для широкого круга задач из различных приложений. Сила метаэвристик состоит в их способности решения сложных задач без знания пространства поиска, именно поэтому эти методы дают возможность решать трудноразрешимые задачи оптимизации. Упрощенно можно рассматривать метаэвристики как алгоритмы, реализующие прямой случайный поиск возможных решений задачи, оптимальных или близких к оптимальным, пока не будет выполнено некое условие или достигнуто заданное число итераций.

Термин *метаэвристика*, впервые введенный в [23], происходит от композиции двух греческих слов («мета» + «эвристика»). «Мета» означает «за его пределами, в верхнем уровне». «Эвристика» происходит от глагола heuriskein[1]. Поначалу, как правило, для решения сложных задач комбинаторной оптимизации разрабатывались специализированные *эвристики*. Эвристика – это любая процедура, которая находит допустимое решение $\widetilde{x} \in X$. Конечно, хотелось бы, чтобы $\widetilde{x}$ совпадало с оптимальным решением x^* (если последнее решение единственно) или $f(\widetilde{x})$ было бы равно $f(x^*)$. Для большинства эвристик, однако, можно только надеяться (и для некоторых и доказать), что $f(\widetilde{x})$ является «близким» к $f(x^*)$. Более общие схемы решения задач КО, называемые *метаэвристиками*, были разработаны Фредом Гловером в 1986 году [23], [26]. Метаэвристики пытаются объединить основные эвристические методы в рамках алгоритмических схем более высокого уровня, направленных на эффективное изучение пространства поиска. Это обычно требует много меньше работы, чем

[1]heuriskein ($\epsilon\upsilon\rho\iota\sigma\kappa\epsilon\iota\nu$) означает «найти».

разработка специализированных эвристик «с нуля». Задача теперь состоит в адаптации общих (метаэвристических) схем решения к решению трудных задач КО. Кроме того, хорошая реализация метаэвристики может обеспечить нахождение за разумное время близкого к оптимальному решения.

Прежде чем термин «метаэвристики» получил широкое распространение, метаэвристики часто называли *современными эвристиками* [40]. Класс метаэвристических алгоритмов включает в себя — но не ограничивается — алгоритмы оптимизации муравьиной колонии (ant colony optimization (ACO)), эволюционные вычисления, включая генетические алгоритмы (ГА), итеративный локальный поиск, метод имитации отжига и алгоритм табу-поиска (или поиска с запретами).

В настоящее время существует достаточно много обзоров, библиографий и классификаций метаэвристических алгоритмов (см., например, Voß (1993) [42], Glover & Laguna (1997) [25], Osman & Laporte [38]).

К сожалению обзоров на русском языке, посвященных метаэвристическим подходам к решению задач КО, в настоящее время нет, хотя имеются публикации, посвященные отдельным метаэвристикам: [5, 6, 7, 8, 9, 12]. В основном имеется литература, посвященная генетическим алгоритмам: [1, 3, 4, 10] и эволюционному моделированию [2]. Единственным исключением, насколько известно автору, является книга [11].

Разумеется, в рамках данной обзорной статьи невозможно подробно описать все аспекты и направления метаэвристических подходов к решению задач КО, поэтому более полную информацию можно найти в следующих книгах и обзорах по метаэвристикам: [14, 26, 44].

Настоящая статья призвана заполнить указанный пробел и дать нашему читателю представление об основных направлениях метаэвристических подходов к решению задач КО.

1. Метаэвристические алгоритмы для задач комбинаторной оптимизации

Метаэвристики — это общие эвристики, позволяющие находить близкие к оптимальным решения различных задач оптимизации за приемлемое время.

Различные описания метаэвристик в литературе позволяют сформулировать некоторые фундаментальные свойства, которыми характеризуются метаэвристики:

- Метаэвристики — это стратегии, которые «управляют» процессом поиска решения.
- Цель метаэвристики состоит в эффективном исследовании пространства поиска для нахождения (почти) оптимальных решений.

- Метаэвристические алгоритмы варьируют от простых процедур локального поиска до сложных процессов обучения.
- Метаэвристические алгоритмы являются приближенными и, как правило, недетерминированными.
- Метаэвристические алгоритмы могут включать механизмы избегания попадания в ловушку в ограниченном области пространства поиска.
- Метаэвристики могут быть описаны на абстрактном уровне (т.е. они не предназначены для решения конкретных задач).
- Метаэвристики могут использовать предметно-ориентированных знания в виде эвристик, которые находятся под контролем стратегии верхнего уровня.
- Современные метаэвристики используют сохраненный в памяти опыт поиска решения для управления поиском.

Каждая метаэвристика имеет свои собственное поведение и характеристики. Однако все метаэвристики имеют ряд основных компонент и выполняют операции в пределах ограниченного числа категорий.

1. *Инициализация.* Метод нахождения начального решения.

2. *Окрестности.* Каждому решению x соответствует множество окрестностей и связанные с ними переходы: $\{N_1, N_2, \ldots, N_q\}$.

3. *Критерий выбора окрестности* определяется в случае наличия более одной окрестности. Этот критерий должны указать не только выбираемую окрестность, но и условия ее выбора. Альтернативы варьируют от «на каждой итерации» (например, генетические методы) до «при данных условиях».

4. *Отбор кандидатов.* Окрестности могут быть очень большими. Тогда обычно рассматривается только подмножество переходов на каждой итерации. Соответствующий список кандидатов $C(x) \subseteq N(x)$ может быть постоянным и обновляемым от итерации к итерации (например, табу-поиск), или же он может быть построен на каждой новой итерации (например, генетические методы). Во всех случаях критерий выбора определяет, каким образом могут быть выбраны решения для включения в список кандидатов.

5. *Критерий принятия.* Переходы оцениваются с помощью функции $g(x, y)$ зависящей от таких параметров двух решений, как значение целевой функции, штрафы за нарушение некоторых ограничений и т.п. Выбирается наилучшее решение по отношению к этому критерию $\widetilde{x} = argopt\{g(x, y); y \in C(x)\}$ (с учетом необходимости предотвращения зацикливания).

6. *Критерии остановки.* Метаэвристика может быть остановлена согласно различным критериям: время вычислений, число итераций, темпы улучшения решения

1. Инициализация: x^0
2. Выбор окрестностей $\mathscr{N} \in \{N_1, \ldots, N_q\}$
3. Выбор кандидата $\mathscr{C}(x) \subseteq \mathscr{N}(x)$
4. Оценка перехода/исследование окрестности $g(x, y), y \in \mathscr{C}(x)$
5. Реализация перехода $\widetilde{x} = argopt\{g(x, y)\}$
6. Оценка решения, обновить параметры поиска
7. Проверка критериев остановки: Stop или Goto 3 (продолжить локальный поиск)
   или Goto 2 (начать новый этап поиска)

Рис. 1. Общая метаэвристика.

1. Определить исходное решение $x^0 \in \mathscr{X}$; $k = 0$;
2. $k = k + 1$;
3. Найти $\widetilde{x} = argmin f(x) | x \in \mathscr{N}(x^k)$;
4. Если $f(\widetilde{x}) \geq f(x^k)$ Stop.
5. Иначе $x^{k+1} = m(\widetilde{x})$; Goto 2.

Рис. 2. Простая эвристика локального поиска.

и т.д. Может быть определен более чем один критерий для управления различными фазами поиска.

Используя эти определения, опишем метаэвристическую процедуру, показанную на Рис. 1, и используем ее для описания трех главных классов метаэвристик: генетических методов, методов имитации отжига и табу-поиска.

Метаэвристики включают две категории: метаэвристики локального поиска (МЛП[2]) и эволюционные алгоритмы (ЭА).

2. Методы локального поиска

2.1. **Общие замечания.** Алгоритм локального поиска начинает свою работу с начального решения. На каждом шаге поиска текущее решение заменяется другим, лучшим, решением, найденным в окрестности текущего решения (рис. 2).

[2]LSMs — local search metaheuristics.

МЛП обычно позволяет найти локальный оптимум. К основным методам МЛП относятся метод имитации отжига [30], табу-поиск [24], процедура жадного рандомизированного адаптивного поиска (GRASP[3]) [19], метод поиска чередующихся окрестностей VNS[4] [35].

2.2. **Метод имитации отжига.** Эта метаэвристика является рандомизированным методом локального поиска, позволяющим избежать плохих локальных оптимумов. Имитация отжига [31] исходит из аналогии с физическим процессом отжига, направленным на получение твердых тел с низкой энергией состояния. В физике конденсированного состояния отжиг является процессом, в котором твердое тело сначала расплавляют путем увеличения температуры, затем постепенно снижают температуру для восстановления твердого состояния с низкой энергией. Метод имитации отжига – это стохастический метод поиска, в котором на каждом шаге текущее решение заменяется другим, случайно выбранным из окрестности и улучшающим значение целевой функции решением. Метод имитации отжига использует управляющий параметр, именуемый температурой, для определения вероятности принятия решений, не улучшающих значение целевой функции. Температура постепенно снижается согласно графику охлаждения так, что отдельные не улучшающие значение целевой функции решения принимаются в конце поиска.

Тщательный отжиг с рядом температурных уровней, на которых температура достаточно долго сохраняется с целью достижения равновесия системы, приводит к более регулярным структурам, соответствующим твердым телам с низкой энергией. В отличие от большинства метаэвристик, для алгоритма имитации отжига доказана асимптотическая сходимость к глобальному оптимуму. Успех метода имитации отжига вызвал разработку детерминистских аналогов, эффективность которых близка к эффективности имитации отжига: Threshold Accepting [18], Record-to-record Travel [17], и алгоритм Great Deluge[5] [17].

2.3. **Табу-поиск.** Как и алгоритм имитации отжига, табу-поиск [25] является метаэвристикой, основанной на локальном поиске, где на каждой итерации выбирается лучшее решение в окрестности текущего решения в качестве нового текущего решения, даже если это приводит к увеличению стоимости решения.

[3]GRASP=Greedy Randomized Adaptive Search Procedure)

[4]Variable Neighborhood Search

[5]Алгоритм Великого Потопа

1. инициализация: выбрать
   a) начальное состояние (решение) $x = x_0$;
   b) начальную температуру $\tau = \tau_0$;
   c) функцию снижения температуры α;
2. окрестность и выбор кандидатов: нет (как правило); заменить;
3. выбрать число итераций L для приблизительного равновесия температуры τ;
4. изменение положения оценки/исследования окрестности: случайным образом выбрать $y \in \chi$;
5. изменение положения $\delta f := f(y) - f(x)$;

   если $\delta f \leq 0$ **то**
   $x := y$
   иначе
   если $g(x, y) = \exp(-\delta f/\tau) > random(0, 1)$ **то**
   $x := y$
6. оценки решения;
7. проверка выполнения критерия остановки
   a) **если** число итераций меньше L **то**
   Goto 4
   b) **если** сходимость не доказана **то**
   $\tau = \alpha(\tau)$; **Goto** 3;

Рис. 3. Общая процедура имитации отжига.

Метод табу-поиска, таким образом, может уйти от плохих локальных оптимумов. В кратковременной памяти, называемой списком табу, сохраняется недавно найденные решения (или атрибуты недавно найденных решений), чтобы избежать краткосрочного зацикливания. Поиск прекращается после определенного числа итераций или если после ряда последовательных итераций не было достигнуто каких-либо улучшений в наилучшем известном решении.

2.4. **Жадный рандомизированный адаптивный поиск GRASP.** Основная идея жадной рандомизированной адаптивной процедуры поиска (GRASP) [20], [41] состоит в использовании рандомизированной жадной эвристики в мультистарт-процедуре для генерирования различных решений. На каждом шаге жадной эвристики элементы, еще не включенные в текущее частичное решение, оцениваются с помощью эвристической функции, а лучшие элементы сохраняются в ограниченном

Вход: Пример задачи
Выход: суб-оптимальное решение

1. найти начальное решение случайным образом и инициализировать температуру T;
2. **пока** $(T > 0)$
   a) **пока** (достижимо термическое равновесие)
      (i) создать случайным образом окрестность состояния и оценить изменения в энергетическом уровне δE;
      (ii) если $\delta E < 0$ обновить текущее состояние на новое состояние;
      (iii) если $\delta E >= 0$ обновить текущее состояние на новое состояние с вероятностью $e^{\frac{-\delta E}{K_B T}}$;
   b) снижение температуры T в соответствии с расписанием отжига;
3. вывод решения, имеющего самую низкую энергию;

Рис. 4. Алгоритм имитации отжига.

1. инициализация: x_0;
2. выбор окрестности: локальный поиск, интенсификация, диверсификация, ...;
3. выбор кандидата $C(\mathrm{x}) \subseteq N(x)$;
4. изменение положения оценки/окрестности исследования: критерии табу, критерий аспирации.
5. изменение положения реализации;
6. обновление памяти и статуса табу;
7. проверка выполнения критерия остановки

если проверка не прошла, **то**
Goto 3 //*продолжение локального поиска* **or Goto** 2 //*изменение фазы поиска*

Рис. 5. Табу-поиск.

списке кандидатов. Один из элементов затем случайно выбирается из этого списка и включается в частичное решение. Когда процесс построения решения завершен, решение дополнительно улучшается с помощью локального поиска. Лучшее решение получается в конце вычислений после определенного количества перезапусков.

Вход: пример задачи
Выход: суб-оптимальное решение
1. инициализация:
   a) Создать начальное решение x и множество $x^* = \{x\}$;
   b) Инициализировать список табу $T = \emptyset$;
   c) Положить счётчики итераций $k = 0$ и $l = 0$;
2. **пока** $(N(x) \setminus T \neq \emptyset)$
   a) $k = k + 1, l = l + 1$;
   b) Выбрать x в качестве лучшего решения из множества $N(x) \setminus T$;
   c) Если $f(\tilde{x}) < f(x^*)$ тогда обновить $x^* = x$ и множество $l = 0$;
   d) Если $k = \bar{k}$ или $l = \bar{l}$ **Goto** 3;
3. вывод лучшего найденного решения x^*;

Рис. 6. Алгоритм табу-поиска.

Вход: пример задачи
Выход: суб-оптимальное решение
множество $x^* = \infty$;
пока (условие остановки не выполнено)
(a) найти случайное «жадное» решение x;
(b) найти локальный минимум $\tilde{x}$ из окрестности $N(x)$ решения x;
(c) **если** $f(\tilde{x}) < f(x^*)$ **то**
обновить множество $x^* = \tilde{x}$;
вывод лучшего найденного решения x^*;

Рис. 7. Общий поиск GRASP.

2.5. **Метод поиска чередующихся окрестностей.** Метод чередующихся окрестностей VNS[6] [28, 35] — это метаэвристика локального поиска, которая использует окрестности для ухода от плохих локальных оптимумов. Основная идея поиска с чередующимися окрестностями VNS [35] состоит в последовательном изучении набора предопределенных окрестностей для получения лучшего решения. Алгоритм VNS использует метод спуска для получения локального минимума. Затем он исследует

[6]Variable Neighborhood Search (VNS)

Вход: пример задачи
Выход: суб-оптимальное решение
1. инициализация: множество решений $S = \emptyset$;
2. **пока** (не построено решение)
   (a) с помощью «жадной» функции создать ограниченный список кандидатов;
   (b) случайно выбрать элемент s из списка кандидатов;
   (c) поместить s во множество решений, то есть $S = S \cup \{s\}$;
   (d) изменить «жадную» функцию с учётом обновленного S;
3. вывод лучшего решения x, соответствующего набору S;

Рис. 8. Алгоритм GRASP.

Вход: пример задачи, решение x, окрестность $N(x)$
Выход: локально-оптимальное решение $\tilde{x}$
1: **пока** (x не локально-оптимальное)
2: (a) найти $\tilde{x} \in N(x)$, где $f(\tilde{x}) < f(x)$;
3: (b) обновить $x = \tilde{x}$;
4: вывод локально-оптимального решения x;

Рис. 9. Фаза локального поиска алгоритма GRASP

случайно либо систематически множество окрестностей. Текущее решение заменяется новым лучшим решение. Поиск начинается с первой окрестности. Если решение, лучшее, чем текущее, там не будет найдено, алгоритм переходит к следующей окрестности, случайным образом генерирует новое решение, и пытается улучшить его. Когда в данной окрестности найден локальный оптимум, выбирается другая окрестность, которая используется на следующих итерациях. Таким образом, для данного множества окрестностей решение порождается случайным образом в первой окрестности текущего решения, из которого выполняется локальный спуск. Если полученный локальный оптимум не лучше текущего решения, то процедура повторяется для следующей окрестности. Поиск стартует вновь из первой окрестности, когда либо найдено решение, лучшее, чем текущее решение, либо все окрестности

Вход: пример задачи P, возможное решение $s_0 \in F$, и
окрестности $N_1, N_2, \ldots, N_p$
Выход: суб-оптимальное решение $s \in F$

1. инициализация: $s = s_0$, $Improve =$ **истина**
2. **пока** ($Improve ==$ **истина**)
   a) $Improve =$ **ложь**;
   b) $k = 1$;
   c) **пока** $k \leq p$
      (i) создать s' в случайной $N_k(s)$;
      (ii) применить локальный поиск для N_1 и использовать s' в качестве локального решения. **пусть** s'' будет локальным оптимумом;
      (iii) **если** $f(s'') < f(s)$ **то**
         пусть $s = s''$;
         $Impruve =$ **истина**;
         прерывание;
      иначе
         $k = k + 1$
3. вывод s в качестве субоптимального решения;

Рис. 10. Метод спуска переменных окрестностей.

были просмотрены. В известном методе локального спуска с чередующимися окрестностями[7], рассматривается наилучший сосед текущего решения вместо случайного выбора. Локальный спуск для этого соседа не выполняется. Этот сосед может стать новым текущим решением в случае улучшения для него значения целевой функции. Поиск затем возобновляется из первой окрестности. Иначе рассматривается следующая окрестность.

3. Эволюционные алгоритмы

3.1. **Общие замечания.** Эволюционные алгоритмы (ЭА) — это стохастические методы поиска, которые успешно применяются во многих реальных и сложных приложениях (эпистатические, мультимодальные, многоцелевые и очень ограниченные задачи). Успех этих алгоритмов в решении сложных задач оптимизации способствовал исследованиям в области, известной как эволюционные вычисления (ЭВ) [13]. ЭА — итеративный метод, которое применяет стохастические операторы к группе

[7]Variable Neighborhood Descent (VND)

```
Generate(P( 0));
t := 0;
while not Termination-Criterion(P(t)) do
Evaluate( P( t ));
P'(t) := Selection(P(t));
P'(t) := ApplyReproduction-Ops(P'( t));
P(t + 1) := Replace(P(t), P'(t));
t := t + 1;
endwhile
```

Рис. 11. Псевдокод ЭА.

индивидуумов (популяции) (см. алгоритм 2). Каждая особь в популяции является кодированной версией предполагаемого решения. Первоначально эта популяция генерируются случайным образом. Функция оценки ставит в соответствие значение пригодности для каждой особи, оценивая ее пригодность для рассматриваемой задачи.

ЭА включают генетические алгоритмы [13], эволюционные стратегии [13], генетическое программирование [13], метод оптимизации муравьиной колонии [15], Estimation of Distribution Algorithms [36], Scatter Search (Рассеянный поиск) [22]). ЭА используют случайно порожденную популяцию решений. Начальная популяция улучшается путем естественного эволюционного процесса. При каждой генерации процесса вся популяция (либо ее часть) заменяется вновь сгенерированными индивидуумами (обычно лучшими, чем предшествующие).

3.2. **Генетические алгоритмы.** Генетические алгоритмы относятся к классу эволюционных методов и имитируют процессы эволюции биологических организмов. В биологии природные популяции изучаются на протяжении многих поколений, оказывается, что они развиваются в соответствии с принципами естественного отбора и выживания наиболее приспособленных для воспроизводства «хорошо адаптированных» особей. Генетические алгоритмы имитируют этот процесс при решении задач оптимизации (Holland 1975 [29]; Goldberg 1989 [27]; Whitley 1994 [43]; Fogel 1994 [21]; Michalewicz 1992 [33]; Michalewicz & Fogel 2000 [34]). Согласно этой парадигме, популяция решений (обычно закодированных в виде битовых или целочисленных строк, называемых хромосомами) эволюционирует от одного поколения к следующему путем применения операторов, подобных тем, что существуют в природе (селекция, генетическое скрещивание и мутация). В процессе селекции только лучшие решения

1. Инициализация: порождение начальной популяции.
2. Выбор окрестности: выбор операторов crossover и mutation.
3. Выбор кандидата-родителя: использование оператора селекции к текущей популяции
4. Оценка шага/исследование окрестности: не производятся
5. Реализация шага: использование операторов crossover, mutation, hill climbing, выбора потомка и родителя для получения новой популяции
6. Если критерии остановки не выполняются, Goto 3 (продолжить эволюцию) или Goto 1.3 (изменить критерии эволюции)

Рис. 12. Общий генетический алгоритм.

могут быть взяты в качестве родителей для создании потомства. Процесс спаривания, известный как скрещивание, использует два выбранных родительских решения и комбинирует их наиболее желательные свойства для получения одного или более решений-потомков.

Оператор hill climbing (локальный поиск) меняет определение характеристик новых индивидуумов для улучшения их годности и разнообразия популяции. Процесс повторяется, пока не будет получено новое поколение потомков. Наконец каждый потомок меняется случайным образом с помощью оператора мутации. Начиная с некоторой начальной популяции (полученной случайным образом или с помощью эвристической процедуры), этот цикл повторяется для множества поколений и в конце будет найдено лучшее решение.

На рис. 12 показаны основные шаги общего генетического алгоритма.

3.3. **Метод оптимизации муравьиной колонии.** Эта метаэвристика инспирирована общением и механизмами взаимодействия реальных муравьев, которые позволяют им найти короткие пути из муравейника к источникам пищи. Средой, через которую осуществляется общение муравьев, является химическое соединение, известное как *феромон*, который оставляется на земле. В то время как изолированный муравей более или менее случайно блуждает, муравей, обнаруживший путь, помеченный феромоном, с некоторой вероятностью последует по нему и укрепит его своим собственным феромоном.

Таким образом, вероятность того, что в будущем другие муравьи будут двигаться по данному пути, растет с числом муравьев, ранее использовавших этот путь. Это приводит к возникновению кратчайших путей, так как феромон стремится аккумулироваться скорее на этих путях. В методе оптимизации муравьиной колонии

1: инициализация переменных $pheromone$;
2: **повторять**
3: **для** $k = 1, \ldots, m$
4: построить решение;
5: **для всех** переменных $pheromone$
6: построить решение, уменьшить переменную на некоторый процент {испарение};
7: **для всех** переменных $pheromone$, соответствующих хорошему решению
8: увеличить переменную { усиление};
9: **пока** не выполнится критерий остановки

Рис. 13. Алгоритм оптимизации муравьиной колонии.

(ACO) [16], множество искусственных муравьев строят решения на каждом цикле случайным и «жадным» способом. Каждый муравей выбирает следующий элемент для включения в свое частичное решение, основываясь на эвристическом оценивании этого элемента и количества феромона – его веса, связанного с этим элементом. Феромон представляет память системы и связан с наличием того элемента в хороших решениях, ранее построенных муравьями. Алгоритм оптимизации муравьиной колонии естественным образом был применен для решения задачи коммивояжера.

Заключение

В настоящей статье сделан краткий обзор метаэвристических алгоритмов, в том рассмотрены числе алгоритмы оптимизации муравьиной колонии, эволюционные алгоритмы, включая генетические алгоритмы, итеративный локальный поиск, метод имитации отжига и алгоритм табу-поиска (или поиска с запретами). Метаэвристики являются мощным и чрезвычайно популярным классом оптимизационных методов, позволяющих находить решения для широкого круга задач комбинаторной оптимизации из различных приложений.

Список литературы

1. *Алексеева Е.В., Кочетов Ю.А.* Генетический локальный поиск для задачи о р-медиане с предпочтениями клиентов // Дискретный анализ и исследование операций. Серия 2. – 2007. – Т. 14, No 1. – С. 3–31.
2. *Емельянов В.В., Курейчик В.В., Курейчик В.М.* Теория и практика эволюционного моделирования. – М: Физматлит, 2003. – 432 с.
3. *Гладков Л.А., Курейчик В.В., Курейчик В.М.* Генетические алгоритмы: Учебное пособие. – 2-е изд. – М: Физматлит, 2006. – 320 с.

4. *Гладков Л.А., Курейчик В.В, Курейчик В.М. и др.* Биоинспирированные методы в оптимизации: монография. – М: Физматлит, 2009. – 384 с.

5. *Гончаров Е.Н., Кочетов Ю.А.* Вероятностный поиск с запретами для дискретных задач безусловной оптимизации // Дискретный анализ и исследование операций. Серия 2. –2002. – Т.9, No 2. – С. 13–30.

6. *Кононов А.В., Кочетов А.В., Плясунов А.В.* Конкурентные модели размещения производства // Журнал вычислительной математики и математической физики. – 2009. – Т. 49, No 6. – С. 1037– 1054.

7. *Кононова П.А., Кочетов Ю.А.* Локальный поиск с чередующимися окрестностями для задачи Джонсона с пассивным буфером // Дискрет. анализ и исслед. операций. – 2012. – Т. 19, № 5. – С. 63–82.

8. *Кочетов Ю.А.* Вычислительные возможности локального поиска в комбинаторной оптимизации // Журнал вычислительной математики и математической физики. – 2008. – Т.48, No 5. – С. 747–764.

9. *Кочетов Ю., Младенович Н., Хансен П.* Локальный поиск с чередующимися окрестностями // Дискретный анализ и исследование операций. Серия 2. – 2003. – Т. 10, No 1. – С. 11–43.

10. *Кочетов Ю.А., Плясунов А.В.* Генетический локальный поиск для задачи о разбиении графа на доли ограниченной мощности // Журнал вычислительной математики и математической физики. – 2012. – Т. 52, № 1. – С. 164–176.

11. Пантелеев А.В. Метаэвристические алгоритмы поиска глобального экстремума. – М: МАИ-Принт, 2009. – 159 стр.

12. *Штовба С. Д.* Муравьиные алгоритмы // Exponenta Pro. Математика в приложениях. – 2004. – № 4.

13. *Bäck T., Fogel D.B., and Michalewicz Z., eds.* Handbook of Evolutionary Computation. –Oxford University Press, 1997.

14. *Blum C. and Roli A.* Metaheuristics in combinatorial optimization: Overview and conceptual comparison // ACM Computing Surveys. – 2003. – 35(3). – P. 268-308.

15. *Dorigo M.* Optimization, Learning and Natural Algorithms. PhD thesis, Dipartamento di Elettronica, Politecnico di Milano, 1992.

16. *Dorigo M. and Stützle T.* Ant Colony Optimization: Overview and Recent Advances // M. Gendreau and J.-Y. Potvin, editors, Handbook of Metaheuristics, volume 146 of International Series in Operations Research & Management Science, chapter 8, pages 227-263. – New York: Springer, 2010. Режим доступа http://code.ulb.ac.be/dbfiles/DorStu2010MetaHandBook.pdf.

17. *Dueck G.* New optimization heuristics: the great deluge algorithm and the record-to-record travel // Journal of Computational Physics. – 1993. – 104 – P. 86–92.

18. *Dueck G. and Scheuer T.* Threshold Accepting: a general purpose optimization algorithm // Journal of Computational Physics. – 1990. – 90 – P. 161–175.

19. *Feo T.A. and Resende M.G.C.* Greedy randomized adaptive search procedures // Journal of Global Optimization. – 1999. – 6. – P. 109-133.

20. *Festa P. and Resende M.G.C.* GRASP: An annotated bibliography // Essays and Surveys on Metaheuristics / C.C. Ribeiro and P. Hansen, eds., pp. 325–367. – Kluwer Academic Publishers, 2002.
21. *Fogel D.B.* Evolutionary Programming: An introduction and some current directions // Statistics and Computing. — 1994. – 4. – P. 113–130.
22. *García-López F., García-Torres M., Melián-Batista B., Moreno-Pérez J.A., Moreno-Vega J.M.* Solving feature subset selection problem by a Parallel Scatter Search // European Journal of Operational Research. – 2006. – 169(2). – P. 477-489.
23. *Glover F.* Future paths for integer programming and links to artificial intelligence // Computers & Operations Research. — 1986. — 131. — P. 533-549.
24. *Glover F.* Tabu Search, part I // ORSA, Journal of Computing. – 1989. – 1. – P. 190-206.
25. *Glover F. and Laguna M.* Tabu Search. – Norwell: Kluwer Academic Publishers, 1997.
26. *Glover F. and Kochenberger G., eds.* Handbook of Metaheuristics. – Norwell: Kluwer Academic Publishers, 2002.
27. *Goldberg D.E.* Genetic algorithms in search, optimization, and machine learning. – Reading: Addison-Wesley, 1989.
28. *Hansen P. and Mladenović N.* Variable Neighborhood Search // Chapter 6 in Handbook of Metaheuristics, F. Glover and G.A. Kochenberger, eds., Kluwer, 145–184, 2003.
29. *Holland J.H.* Adaptation in Natural and Artificial Systems. – The University of Michigan Press, 1975.
30. *Kirkpatrick S., Gelatt C.D., and Vecchi M.P.* Optimization by simulated annealing // Science. – 1983. – 220(4598). – P. 671-680.
31. *van Laarhoven P.J.M. and Aarts E.H.L.* Simulated Annealing: Theory and Applications. – Dordrecht: Springer, 1987.
32. *Lawler E., Lenstra J. K., Rinnooy Kan A. H. G., and Shmoys D.B.* The Travelling Salesman Problem. — New York: John Wiley & Sons, 1985.
33. *Michalewicz Z.* Genetic Algorithms + Data Structures = Evolution Programs. – Springer-Verlag, 1992.
34. *Michalewicz Z. & Vogel D.B.* How to Solve it: Modern Heuristics – Berlin: Springer, 2000.
35. *Mladenović N. and Hansen P.* Variable neighborhood search // Computers and Opereration Research. – 1997. –24. – P. 1097-1100.
36. *Mühlenbein H., Mahnig T., and Ochoa A.* Schemata, distributions and graphical models in evolutionary optimization // Journal of Heuristics. – 1999. – 5(2). – P. 215-247.
37. Nemhauser G.L. and Wolsey A.L. Integer and Combinatorial Optimization. – New York: John Wiley & Sons, 1988.
38. *Osman I.H., Laporte G.* Metaheuristics: a bibliography // Ann. Oper. Res. – 1996. – V. 63. – P. 513–628.
39. *Papadimitriou C.H. and Steiglitz K.* Combinatorial Optimization – Algorithms and Complexity. – New York: Dover Publications, 1982.

40. *Reeves C.R., ed.* Modern Heuristic Techniques for Combinatorial Problems. – Oxford: Blackwell Scientific Publishing, 1993.

41. *Resende M.G.C., and Ribeiro C.C.* Greedy Randomized Adaptive Search Procedures // Chapter 8 in Handbook of Metaheuristics, F. Glover and G.A. Kochenberger, eds., Kluwer, pp. 219–249, 2003.

42. *Voß S.* Tabu Search: Applications and Prospects // Network Optimization Problems / Du D.-Z. and Pardalos P., eds., pp. 333–353. – Singapor: World Scientific Publishing Co., 1993.

43. *Whitley L.D.* A Genetic Algorithm Tutorial // Statistics and Computing. – 1994. – 4. – P. 65–85.

44. *Yagiura M. and Ibaraki T.* On metaheuristic algorithms for combinatorial optimization problems // Systems and Computers in Japan. – 2001. – 32(3). – P. 33-55.

Статья поступила в редакцию 16.02.2014