Метаэвристики

Виктор Васильевич Лепин

Обозначения

$$\min_{s \in S} f(s)$$

f — целевая функция
S — множество допустимых решений
$s \in S$ — допустимое решение
$N \subseteq S \times S$ — окрестность на множестве S
$N(s) = \{s' \in S | N(s, s')\} \subsetneq S$ — окрестность решения s
$f^* = \min\{f(s') | s' \in N(s)\}$
$N^*(s) = \{s' \in N(s) | f(s') \leq f(s)\}$

$s \in S$ (строгий) локальный минимум
если $(f(s) < f(s'))f(s) \leq f(s') \forall s' \in N(s)$

Эвристики, основанные на локальном поиске

- Идеи локального поиска получили свое дальнейшее развитие в так называемых метаэвристиках, т. е. в общих схемах построения алгоритмов, которые могут быть применены практически к любой задаче дискретной оптимизации.

- Идеи локального поиска получили свое дальнейшее развитие в так называемых метаэвристиках, т. е. в общих схемах построения алгоритмов, которые могут быть применены практически к любой задаче дискретной оптимизации.
- Все метаэвристики являются итерационными процедурами и для многих из них установлена асимптотическая сходимость наилучшего найденного решения к глобальному оптимуму.

- Идеи локального поиска получили свое дальнейшее развитие в так называемых метаэвристиках, т. е. в общих схемах построения алгоритмов, которые могут быть применены практически к любой задаче дискретной оптимизации.
- Все метаэвристики являются итерационными процедурами и для многих из них установлена асимптотическая сходимость наилучшего найденного решения к глобальному оптимуму.
- К числу метаэвристик относятся алгоритмы имитации отжига, поиск с запретами, генетические алгоритмы, о которых пойдет речь в этом разделе, а также нейронные сети, муравьиные колонии, вероятностные жадные процедуры и другие.

Алгоритм имитации отжига

- Экзотическое название данного алгоритма связано с методами имитационного моделирования в статистической физике, основанными на технике Монте-Карло.

- Экзотическое название данного алгоритма связано с методами имитационного моделирования в статистической физике, основанными на технике Монте-Карло.
- Исследование кристаллической решетки и поведения атомов при медленном остывании тела привело к появлению на свет вероятностных алгоритмов, которые оказались чрезвычайно эффективными в комбинаторной оптимизации.

- Экзотическое название данного алгоритма связано с методами имитационного моделирования в статистической физике, основанными на технике Монте-Карло.
- Исследование кристаллической решетки и поведения атомов при медленном остывании тела привело к появлению на свет вероятностных алгоритмов, которые оказались чрезвычайно эффективными в комбинаторной оптимизации.
- Сегодня эти алгоритмы являются популярными как среди практиков благодаря своей простоте, гибкости и эффективности, так и среди теоретиков, поскольку удается аналитически исследовать их свойства и доказать асимптотическую сходимость.

- Алгоритм имитации отжига относится к классу пороговых алгоритмов локального поиска.

- Алгоритм имитации отжига относится к классу пороговых алгоритмов локального поиска.
- Пусть определена (возможно бесконечная) последовательность $t_0, t_1, \ldots,$ *порогов*.

- Алгоритм имитации отжига относится к классу пороговых алгоритмов локального поиска.
- Пусть определена (возможно бесконечная) последовательность $t_0, t_1, \ldots,$ *порогов*.
- На каждом шаге порогового алгоритма в окрестности текущего решения s_k выбирается некоторое решение p, и если разность по целевой функции между новым и текущим решением не превосходит заданного порога t_k, то новое решение p заменяет текущее.

- Алгоритм имитации отжига относится к классу пороговых алгоритмов локального поиска.
- Пусть определена (возможно бесконечная) последовательность $t_0, t_1, \ldots,$ *порогов*.
- На каждом шаге порогового алгоритма в окрестности текущего решения s_k выбирается некоторое решение p, и если разность по целевой функции между новым и текущим решением не превосходит заданного порога t_k, то новое решение p заменяет текущее.
- В противном случае выбирается новое соседнее решение.

- Алгоритм имитации отжига относится к классу пороговых алгоритмов локального поиска.
- Пусть определена (возможно бесконечная) последовательность $t_0, t_1, \ldots,$ *порогов*.
- На каждом шаге порогового алгоритма в окрестности текущего решения s_k выбирается некоторое решение p, и если разность по целевой функции между новым и текущим решением не превосходит заданного порога t_k, то новое решение p заменяет текущее.
- В противном случае выбирается новое соседнее решение.
- Общая схема пороговых алгоритмов может быть представлена следующим образом.

Пороговый алгоритм

1. Выбрать начальное решение $i_0 \in \mathrm{SOL}$ и положить $m^* = m(i_0)$; $k = 0$.

2. Пока не выполнен критерий остановки, делать следующее:

 2.1. Случайно выбрать $j \in \mathcal{N}(i_k)$.

 2.2. Если $m(j) - m(i_k) < t_k$ то $i_{k+1} := j$.

 2.3. Если $m^* > m(i_k)$, то $m^* := m(i_k)$.

 2.4. Положить $k := k + 1$.

Отметим, что если $t_k > 0$, то возможен переход к новому решению i_{k+1} с худшим значением целевой функции. В зависимости от способа задания пороговой последовательности $\{t_k\}$ различают три типа алгоритмов.

1. <u>Последовательное улучшение</u>: $t_k = 0$, $k = 0, 1, 2 \ldots$, — вариант классического локального спуска с монотонным улучшением по целевой функции.

2. <u>Пороговое улучшение</u>: пороговая последовательность монотонно убывает до 0, т. е. $t_k = c_k$, $k = 0, 1, 2 \ldots$, $c_k \geq c0$, $c_k \geq c_{k+1}$ и $\lim_{k\to\infty} c_k \to 0$ — вариант локального поиска, когда допускается ухудшение по целевой функции до некоторого заданного порога, и этот порог последовательно снижается до нуля.

3. Имитация отжига: $t_k \geq 0$, $k = 0, 1, 2 \ldots,$ — случайная величина с математическим ожиданием $E(t_k) = c_k \geq 0$ — вариант локального поиска, когда допускается произвольное ухудшение по целевой функции, но вероятность такого перехода обратно пропорциональна величине ухудшения, точнее для любого $j \in \mathcal{N}(i)$

$$P_{ij} = \begin{cases} 1, & \text{если } m(i) \leq m(i), \\ \exp(\frac{m(i)-m(j)}{c_k}), & \text{если } m(j) > m(i). \end{cases}$$

Последовательность $\{c_k\}$ оказывает существенное влияние на сходимость алгоритма. Поэтому ее выбирают так, чтобы $c_k \to 0$ при $k \to \infty$. Параметр c_k называют температурой. Поведение алгоритма "имитация отжига"определяется структурой окрестностей $\mathcal{N}$, начальным решением s_0, и значениями параметров r, t, l. Правило остановки представлено в алгоритме предикатом Frozen.

Имитация отжига

Input: Пример x.
Output: Решение s.

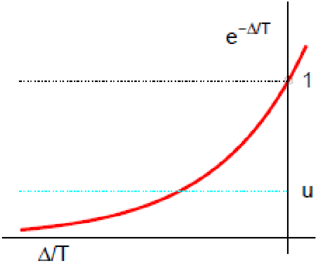
```
begin
  τ := t;
  s := начальное допустимое решение s_0;
  repeat
    for i = 1 to l do
    (* локальный поиск при температуре τ *);
    begin
      Выбрать не исследованное решение s' ∈ (s);
      if m(x, s') < m(x, s) then
        s := s'
      else;
      begin
        δ := m(x, s') − m(x, s);
        s := s' с вероятностью e^(−δ/τ)
      end
    end;
    (* изменение температуры *);
    τ := rτ
  until FROZEN;
  return s
```

Алгоритм имитации отжига (SA: S. Kirkpatrick, C. Gelatt, M. P. Vecchi, 1982)

Выбрать начальное решение $s \in S$, вычислить $f(s)$
Задать начальную температуру T
Пока не выполнен критерий остановки выполнить:

- выбрать решение $s' \in N(s)$ случайным образом; если $f(s') < f(s)$, то $s := s'$
- иначе с вероятностью $e^{\frac{f(s)-f(s')}{T}}$ положить $s := s'$;
- понизить температуру $T := T \cdot r$;

Выдать в качестве ответа s – локальный оптимум.



Поведение алгоритма имитации отжига

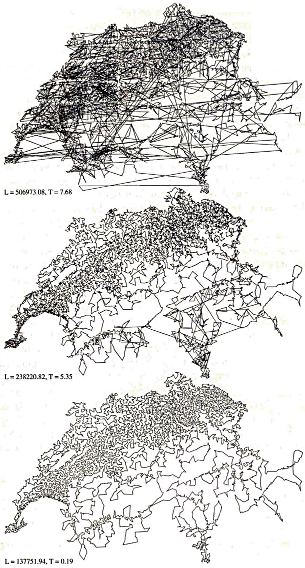


Fig. 1.2. The traveling salesman problem (13206 nodes of the Swiss road network): best known configurations (length: L) at the end of 3 temperature stages (T).

Поиск с запретами

- Основоположником алгоритма поиска с запретами (Tabu search) является Ф. Гловер, который предложил принципиально новую схему локального поиска.

- Основоположником алгоритма поиска с запретами (Tabu search) является Ф. Гловер, который предложил принципиально новую схему локального поиска.
- Она позволяет алгоритму не останавливаться в точке локального оптимума, как это предписано в стандартном алгоритме локального спуска, а путешествовать от одного оптимума к другому в надежде найти среди них глобальный оптимум.

- Основоположником алгоритма поиска с запретами (Tabu search) является Ф. Гловер, который предложил принципиально новую схему локального поиска.
- Она позволяет алгоритму не останавливаться в точке локального оптимума, как это предписано в стандартном алгоритме локального спуска, а путешествовать от одного оптимума к другому в надежде найти среди них глобальный оптимум.
- Основным механизмом, позволяющим алгоритму выбираться из локального оптимума, является список запретов $Tabu(i_k)$.

- Основоположником алгоритма поиска с запретами (Tabu search) является Ф. Гловер, который предложил принципиально новую схему локального поиска.
- Она позволяет алгоритму не останавливаться в точке локального оптимума, как это предписано в стандартном алгоритме локального спуска, а путешествовать от одного оптимума к другому в надежде найти среди них глобальный оптимум.
- Основным механизмом, позволяющим алгоритму выбираться из локального оптимума, является список запретов $Tabu(i_k)$.
- Он строится по предыстории поиска, т. е. по нескольким последним точкам $i_k, i_{k-1}, \ldots, i_{k-l+1}$, и запрещает исследовать часть окрестности $\mathcal{N}(i_k)$ текущего решения i_k.

- Основоположником алгоритма поиска с запретами (Tabu search) является Ф. Гловер, который предложил принципиально новую схему локального поиска.
- Она позволяет алгоритму не останавливаться в точке локального оптимума, как это предписано в стандартном алгоритме локального спуска, а путешествовать от одного оптимума к другому в надежде найти среди них глобальный оптимум.
- Основным механизмом, позволяющим алгоритму выбираться из локального оптимума, является список запретов $Tabu(i_k)$.
- Он строится по предыстории поиска, т. е. по нескольким последним точкам $i_k, i_{k-1}, \ldots, i_{k-l+1}$, и запрещает исследовать часть окрестности $\mathcal{N}(i_k)$ текущего решения i_k.
- Точнее на каждом шаге алгоритма очередная точка i_{k+1} является оптимальным решением подзадачи

$$m(i_{k+1}) = \min\{m(j) \mid j \in \mathcal{N}(i_k) \setminus Tabu_l(i_k)\}.$$

- Множество $Tabu_l(i_k) \subseteq \mathcal{N}(i_k)$ определяется по предшествующим решениям.

- Множество $Tabu_l(i_k) \subseteq \mathcal{N}(i_k)$ определяется по предшествующим решениям.
- Список запретов учитывает специфику задачи и, как правило, запрещает использование тех "фрагментов"решения (ребер графа, координат вектора, цвет вершины), которые менялись на последних l шагах алгоритма.

- Множество $Tabu_l(i_k) \subseteq \mathcal{N}(i_k)$ определяется по предшествующим решениям.
- Список запретов учитывает специфику задачи и, как правило, запрещает использование тех "фрагментов"решения (ребер графа, координат вектора, цвет вершины), которые менялись на последних l шагах алгоритма.
- Константа $l \geq 0$ определяет его память.

- Множество $Tabu_l(i_k) \subseteq \mathcal{N}(i_k)$ определяется по предшествующим решениям.
- Список запретов учитывает специфику задачи и, как правило, запрещает использование тех "фрагментов"решения (ребер графа, координат вектора, цвет вершины), которые менялись на последних l шагах алгоритма.
- Константа $l \geq 0$ определяет его память.
- При «короткой памяти» $(l = 0)$ получаем стандартный локальный спуск.

- Существует много вариантов реализации основной идеи поиска с запретами.

- Существует много вариантов реализации основной идеи поиска с запретами.
- Приведем один из них, для которого удается установить асимптотические свойства.

- Существует много вариантов реализации основной идеи поиска с запретами.
- Приведем один из них, для которого удается установить асимптотические свойства.
- Рассмотрим рандомизированную окрестность $\mathcal{N}_p(i) \subseteq \mathcal{N}(i)$, где каждый элемент окрестности $j \in \mathcal{N}(i)$ включается в множество $\mathcal{N}_p(i)$ с вероятностью p независимо от других элементов.

- Существует много вариантов реализации основной идеи поиска с запретами.
- Приведем один из них, для которого удается установить асимптотические свойства.
- Рассмотрим рандомизированную окрестность $\mathcal{N}_p(i) \subseteq \mathcal{N}(i)$, где каждый элемент окрестности $j \in \mathcal{N}(i)$ включается в множество $\mathcal{N}_p(i)$ с вероятностью p независимо от других элементов.
- С ненулевой вероятностью множество $\mathcal{N}_p(i)$ может совпадать с $\mathcal{N}(i)$, может оказаться пустым или содержать ровно один элемент.

- Существует много вариантов реализации основной идеи поиска с запретами.
- Приведем один из них, для которого удается установить асимптотические свойства.
- Рассмотрим рандомизированную окрестность $\mathcal{N}_p(i) \subseteq \mathcal{N}(i)$, где каждый элемент окрестности $j \in \mathcal{N}(i)$ включается в множество $\mathcal{N}_p(i)$ с вероятностью p независимо от других элементов.
- С ненулевой вероятностью множество $\mathcal{N}_p(i)$ может совпадать с $\mathcal{N}(i)$, может оказаться пустым или содержать ровно один элемент.
- Общая схема алгоритма поиска с запретами может быть представлена следующим образом.

Алгоритм поиска с запретами

1. Выбрать начальное решение $i_0 \in \mathrm{SOL}$ и положить $m^* = m(i_0)$; $k = 0$.

2. Пока не выполнен критерий остановки, делать следующее:

    2.1. Сформировать окрестность $\mathcal{N}_p(i_k)$.

    2.2. Если $\mathcal{N}_p(i_k) \neq \emptyset$, то $i_{k+1} := i_k$, иначе найти i_{k+1}, такой, что $m(i_{k+1}) = \min\{m(j) \mid j \in \mathcal{N}_p(i_k) \setminus Tabu_l(i_k)\}$.

    2.3. Если $m^* > m(i_{k+1})$, то $m^* := m(i_{k+1})$.

    2.4. Положить $k := k + 1$ и обновить список запретов $Tabu_l(i_k)$.

Параметры p и l являются управляющими для данного алгоритма и выбор их значений зависит от размерности задачи и мощности окрестности.

Алгоритм поиска с запретами (TS: F. Glover, 1986)

Выбрать начальное решение $s \in S$, вычислить $f(s)$
Пока не выполнен критерий остановки выполнить:

- выбрать наилучшее решение $s \in N(s) \setminus TabuList$ с $f(s') < f(s)$
- $s := s'$
- по решению s обновить $TabuList$

Выдать в качестве ответа s – локальный оптимум.

Генетические алгоритмы

- Идея генетических алгоритмов заимствована у живой природы и состоит в организации эволюционного процесса, конечной целью которого является получение оптимального решения в сложной комбинаторной задаче.

- Идея генетических алгоритмов заимствована у живой природы и состоит в организации эволюционного процесса, конечной целью которого является получение оптимального решения в сложной комбинаторной задаче.
- Разработчик генетических алгоритмов выступает в данном случае как "создатель который должен правильно установить законы эволюции, чтобы достичь желаемой цели как можно быстрее.

- Идея генетических алгоритмов заимствована у живой природы и состоит в организации эволюционного процесса, конечной целью которого является получение оптимального решения в сложной комбинаторной задаче.
- Разработчик генетических алгоритмов выступает в данном случае как "создатель который должен правильно установить законы эволюции, чтобы достичь желаемой цели как можно быстрее.
- Впервые эти нестандартные идеи были применены к решению оптимизационных задач в середине 70-х годов.

- Идея генетических алгоритмов заимствована у живой природы и состоит в организации эволюционного процесса, конечной целью которого является получение оптимального решения в сложной комбинаторной задаче.
- Разработчик генетических алгоритмов выступает в данном случае как "создатель который должен правильно установить законы эволюции, чтобы достичь желаемой цели как можно быстрее.
- Впервые эти нестандартные идеи были применены к решению оптимизационных задач в середине 70-х годов.
- Примерно через десять лет появились первые теоретические обоснования этого подхода.

- На сегодняшний день генетические алгоритмы доказали свою конкурентоспособность при решении многих NP-трудных задач и особенно в практических приложениях, где математические модели имеют сложную структуру и применение стандартных методов типа ветвей и границ, динамического или линейного программирования крайне затруднено.

- На сегодняшний день генетические алгоритмы доказали свою конкурентоспособность при решении многих NP-трудных задач и особенно в практических приложениях, где математические модели имеют сложную структуру и применение стандартных методов типа ветвей и границ, динамического или линейного программирования крайне затруднено.
- Общую схему генетических алгоритмов проще всего понять, рассматривая задачи безусловной оптимизации

$$max\{m(i) \mid i \in B^n\}, \quad B^n = \{0,1\}^n.$$

- Примерами служат задачи размещения, стандартизации, выполнимости и др.

- Примерами служат задачи размещения, стандартизации, выполнимости и др.
- Стандартный генетический алгоритм начинает свою работу с формирования начальной *популяции* $I_0 = \{i_1, i_2, \ldots, i_s\}$ — конечного набора допустимых решений задачи.

- Примерами служат задачи размещения, стандартизации, выполнимости и др.
- Стандартный генетический алгоритм начинает свою работу с формирования начальной *популяции* $I_0 = \{i_1, i_2, \dots, i_s\}$ — конечного набора допустимых решений задачи.
- Эти решения могут быть выбраны случайным образом или получены с помощью вероятностных жадных алгоритмов.

- Примерами служат задачи размещения, стандартизации, выполнимости и др.
- Стандартный генетический алгоритм начинает свою работу с формирования начальной *популяции* $I_0 = \{i_1, i_2, \ldots, i_s\}$ — конечного набора допустимых решений задачи.
- Эти решения могут быть выбраны случайным образом или получены с помощью вероятностных жадных алгоритмов.
- Как мы увидим ниже, выбор начальной популяции не имеет значения для сходимости процесса в асимптотике, однако формирование «хорошей» начальной популяции (например, из множества локальных оптимумов) может заметно сократить время достижения глобального оптимума.

- На каждом шаге эволюции с помощью вероятностного оператора селекции выбираются два решения, *родители* i_1, i_2.

- На каждом шаге эволюции с помощью вероятностного оператора селекции выбираются два решения, *родители* i_1, i_2.
- Оператор *скрещивания* по решениям i_1, i_2 строит новое решение i', которое затем подвергается небольшим случайным модификациям, которые принято называть *мутациями*.

- На каждом шаге эволюции с помощью вероятностного оператора селекции выбираются два решения, *родители* i_1, i_2.
- Оператор *скрещивания* по решениям i_1, i_2 строит новое решение i', которое затем подвергается небольшим случайным модификациям, которые принято называть *мутациями*.
- Затем решение добавляется в популяцию, а решение с наименьшим значением целевой функции удаляется из популяции.

- На каждом шаге эволюции с помощью вероятностного оператора селекции выбираются два решения, *родители* i_1, i_2.
- Оператор *скрещивания* по решениям i_1, i_2 строит новое решение i', которое затем подвергается небольшим случайным модификациям, которые принято называть *мутациями*.
- Затем решение добавляется в популяцию, а решение с наименьшим значением целевой функции удаляется из популяции.
- Общая схема такого алгоритма может быть записана следующим образом.

Генетический алгоритм

1. Выбрать начальную популяцию I_0 и положить $m^* = \max\{m(i) \mid i \in I_0\}$; $k := 0$.

2. Пока не выполнен критерий остановки, делать следующее:

    2.1. Выбрать родителей i_1, i_2. из популяции I_k.

    2.2. Построить i' по i_1, i_2.

    2.3. Модифицировать i'.

    2.4. Если $m^* < m(i')$, то $m^* := m(i')$.

    2.5. Обновить популяцию и положить $k := k + 1$.

- Остановимся подробнее на основных операторах этого алгоритма: селекции, скрещивании и мутации.

- Остановимся подробнее на основных операторах этого алгоритма: селекции, скрещивании и мутации.
- Среди операторов селекции наиболее распространенными являются два вероятностных оператора пропорциональной и турнирной селекции.

- Остановимся подробнее на основных операторах этого алгоритма: селекции, скрещивании и мутации.
- Среди операторов селекции наиболее распространенными являются два вероятностных оператора пропорциональной и турнирной селекции.
- При пропорциональной селекции вероятность на k-м шаге выбрать решение i в качестве одного из родителей задается формулой

$$P(i\text{– выбрано}) = \frac{m(i)}{\sum_{j \in I_k} m(j)}, \quad i \in I_k,$$

в предположении, что $m(i) > 0$ для всех $i \in \mathrm{SOL}$.

- Остановимся подробнее на основных операторах этого алгоритма: селекции, скрещивании и мутации.
- Среди операторов селекции наиболее распространенными являются два вероятностных оператора пропорциональной и турнирной селекции.
- При пропорциональной селекции вероятность на k-м шаге выбрать решение i в качестве одного из родителей задается формулой

$$P(i\text{– выбрано}) = \frac{m(i)}{\sum_{j \in I_k} m(j)}, \quad i \in I_k,$$

  в предположении, что $m(i) > 0$ для всех $i \in \mathrm{SOL}$.
- При турнирной селекции формируется случайное подмножество из элементов популяции и среди них выбирается один элемент с наибольшим значением целевой функции.

- Турнирная селекция имеет определенные преимущества перед пропорциональной, так как не теряет своей избирательности, когда в ходе эволюции все элементы популяции становятся примерно равными по значению целевой функции.

- Турнирная селекция имеет определенные преимущества перед пропорциональной, так как не теряет своей избирательности, когда в ходе эволюции все элементы популяции становятся примерно равными по значению целевой функции.
- Операторы селекции строятся таким образом, чтобы с ненулевой вероятностью любой элемент популяции мог бы быть выбран в качестве одного из родителей. Более того, допускается ситуация, когда оба родителя представлены одним и тем же элементом популяции.

- Как только два решения выбраны, к ним применяется вероятностный оператор скрещивания (*crossover*).

- Как только два решения выбраны, к ним применяется вероятностный оператор скрещивания (*crossover*).
- Существует много различных версий этого оператора, среди которых простейшим, по видимому, является однородный оператор.

- Как только два решения выбраны, к ним применяется вероятностный оператор скрещивания (*crossover*).
- Существует много различных версий этого оператора, среди которых простейшим, по видимому, является однородный оператор.
- По решениям i_1, i_2 он строит решение i', присваивая каждой координате этого вектора с вероятностью 0,5 соответствующее значение одного из родителей.

- Как только два решения выбраны, к ним применяется вероятностный оператор скрещивания (*crossover*).
- Существует много различных версий этого оператора, среди которых простейшим, по видимому, является однородный оператор.
- По решениям i_1, i_2 он строит решение i', присваивая каждой координате этого вектора с вероятностью 0,5 соответствующее значение одного из родителей.
- Если вектора i_1, i_2 совпадали, скажем, по первой координате, то вектор i' "унаследует"это значение.

- Как только два решения выбраны, к ним применяется вероятностный оператор скрещивания (*crossover*).
- Существует много различных версий этого оператора, среди которых простейшим, по видимому, является однородный оператор.
- По решениям i_1, i_2 он строит решение i', присваивая каждой координате этого вектора с вероятностью 0,5 соответствующее значение одного из родителей.
- Если вектора i_1, i_2 совпадали, скажем, по первой координате, то вектор i' "унаследует"это значение.
- Геометрически оператор скрещивания случайным образом выбирает в гиперкубе вершину i', которая принадлежит минимальной грани, содержащей вершины i_1, i_2.

- Как только два решения выбраны, к ним применяется вероятностный оператор скрещивания (*crossover*).
- Существует много различных версий этого оператора, среди которых простейшим, по видимому, является однородный оператор.
- По решениям i_1, i_2 он строит решение i', присваивая каждой координате этого вектора с вероятностью 0,5 соответствующее значение одного из родителей.
- Если вектора i_1, i_2 совпадали, скажем, по первой координате, то вектор i' "унаследует"это значение.
- Геометрически оператор скрещивания случайным образом выбирает в гиперкубе вершину i', которая принадлежит минимальной грани, содержащей вершины i_1, i_2.
- Можно сказать, что оператор скрещивания старается выбрать новое решение i' где-то между i_1, i_2, полагаясь на удачу.

- Более аккуратная процедура могла бы выглядеть таким образом.

- Более аккуратная процедура могла бы выглядеть таким образом.
- Новым решением i' является оптимальное решение исходной задачи на соответствующей грани гиперкуба.

- Более аккуратная процедура могла бы выглядеть таким образом.
- Новым решением i' является оптимальное решение исходной задачи на соответствующей грани гиперкуба.
- Конечно, если расстояние Хемминга между i_1, i_2 равно n, то задача оптимального скрещивания совпадает с исходной.

- Более аккуратная процедура могла бы выглядеть таким образом.
- Новым решением i' является оптимальное решение исходной задачи на соответствующей грани гиперкуба.
- Конечно, если расстояние Хемминга между i_1, i_2 равно n, то задача оптимального скрещивания совпадает с исходной.
- Тем не менее даже приближенное решение этой задачи вместо случайного выбора заметно улучшает работу генетического алгоритма.

- Более аккуратная процедура могла бы выглядеть таким образом.
- Новым решением i' является оптимальное решение исходной задачи на соответствующей грани гиперкуба.
- Конечно, если расстояние Хемминга между i_1, i_2 равно n, то задача оптимального скрещивания совпадает с исходной.
- Тем не менее даже приближенное решение этой задачи вместо случайного выбора заметно улучшает работу генетического алгоритма.
- По аналогии с однородным оператором скрещивания легко предложить и другие операторы, использующие не только два, но и произвольное число решений из популяции.

- Оператор мутации, применяемый к решению i' в п. 2.3. генетического алгоритма, с заданной вероятностью $p_m \in (0, 1)$ меняет значение каждой координаты на противоположное.

- Оператор мутации, применяемый к решению i' в п. 2.3. генетического алгоритма, с заданной вероятностью $p_m \in (0,1)$ меняет значение каждой координаты на противоположное.
- Например, вероятность того, что $i' = (0,0,0,0,0)$ в ходе мутации перейдет в $j' = (1,1,1,0,0)$, равна $p_m \times p_m \times p_m \times (1 - p_m) \times (1 - p_m) > 0$.

- Оператор мутации, применяемый к решению i' в п. 2.3. генетического алгоритма, с заданной вероятностью $p_m \in (0,1)$ меняет значение каждой координаты на противоположное.
- Например, вероятность того, что $i' = (0,0,0,0,0)$ в ходе мутации перейдет в $j' = (1,1,1,0,0)$, равна $p_m \times p_m \times p_m \times (1-p_m) \times (1-p_m) > 0$.
- Таким образом, с ненулевой вероятностью решение i' может перейти в любое другое решение.

- Оператор мутации, применяемый к решению i' в п. 2.3. генетического алгоритма, с заданной вероятностью $p_m \in (0, 1)$ меняет значение каждой координаты на противоположное.
- Например, вероятность того, что $i' = (0, 0, 0, 0, 0)$ в ходе мутации перейдет в $j' = (1, 1, 1, 0, 0)$, равна $p_m \times p_m \times p_m \times (1 - p_m) \times (1 - p_m) > 0$.
- Таким образом, с ненулевой вероятностью решение i' может перейти в любое другое решение.
- Отметим, что модификация решения i' может состоять не только в случайной мутации, но и в частичной перестройке решения алгоритмами локального поиска.

- Оператор мутации, применяемый к решению i' в п. 2.3. генетического алгоритма, с заданной вероятностью $p_m \in (0,1)$ меняет значение каждой координаты на противоположное.
- Например, вероятность того, что $i' = (0,0,0,0,0)$ в ходе мутации перейдет в $j' = (1,1,1,0,0)$, равна $p_m \times p_m \times p_m \times (1-p_m) \times (1-p_m) > 0$.
- Таким образом, с ненулевой вероятностью решение i' может перейти в любое другое решение.
- Отметим, что модификация решения i' может состоять не только в случайной мутации, но и в частичной перестройке решения алгоритмами локального поиска.
- Применение локального спуска позволяет генетическому алгоритму сосредоточиться только на локальных оптимумах.

- Оператор мутации, применяемый к решению i' в п. 2.3. генетического алгоритма, с заданной вероятностью $p_m \in (0, 1)$ меняет значение каждой координаты на противоположное.
- Например, вероятность того, что $i' = (0, 0, 0, 0, 0)$ в ходе мутации перейдет в $j' = (1, 1, 1, 0, 0)$, равна $p_m \times p_m \times p_m \times (1 - p_m) \times (1 - p_m) > 0$.
- Таким образом, с ненулевой вероятностью решение i' может перейти в любое другое решение.
- Отметим, что модификация решения i' может состоять не только в случайной мутации, но и в частичной перестройке решения алгоритмами локального поиска.
- Применение локального спуска позволяет генетическому алгоритму сосредоточиться только на локальных оптимумах.
- Множество локальных оптимумов может оказаться экспоненциально большим и на первый взгляд кажется, что такой вариант алгоритма не будет иметь больших преимуществ.

Гибридная схема генетического алгоритма

1. Выбрать начальную популяцию из k решений. Запомнить рекорд $f^* = \min_{i=1,\dots,k} f(s_i)$.
2. Пока не выполнен критерий остановки делать следующее:
   2.1. Выбрать "родителей" s_{i_1} s_{i_2} из популяции.
   2.2. Применить к s_{i_1} s_{i_2} оператор скрещивания и получить новое решение s'.
   2.3. Применить к s' оператор мутации и получить новое решение s''.
   2.4. Применить к s'' оператор локального улучшения и получить новое решение s'''.
   2.5. Если $f(s''') < f^*$, то сменить рекорд $f^* := f(s''')$.
   2.6. Добавить s''' к популяции и удалить из нее наихудшее решение.

Генетический алгоритм (GA, Holland, 1975, Goldberg, 1989)

Имитационный алгоритм без вспомогательной процедуры локального улучшения на шаге 2.4.

Поиск с чередующимися окрестностями (VNS: N. Mladenovic, P. Hansen, 1997)

Задана система разнотипных окрестностей $N_1, \ldots, N_k$.

Выбрать начальное решение $s \in S$, вычислить $f(s)$
Пока не выполнен критерий остановки повторять:

- $i := 1$;
- Пока $i \leq k$ выполнить:
  - выбрать решение $s \in N_i(s)$ случайным образом;
  - применить процедуру локального улучшения с окрестностью N к s',
  - s'' – полученное решение
  - если $f(s'') < f(s)$, то $s := s''$
  - иначе $i := i + 1$;

Выдать в качестве ответа s – локальный оптимум.

Вероятностный жадный алгоритм (GRASP: Feo, Resende, 1989)

Пока не выполнен критерий остановки выполнить:

- жадной стратегией покомпонентно построить решение s
- запустить процедуру локального поиска со стартовым решением s

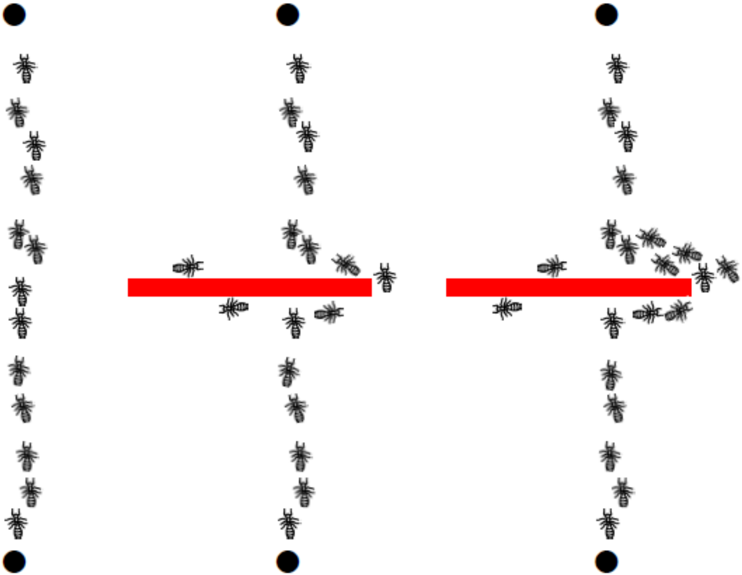
Выдать в качестве ответа локальный оптимум.

Алгоритм муравьиных колоний (ACO: M. Dorigo, V. Maniezzo, 1991)

1. Задать исходную статистическую информацию.
2. Пока не выполнен критерий остановки делать следующее:
    1. Построить популяцию решений рандомизированным алгоритмом
    2. Применить процедуру локального поиска к решениям из популяции
    3. Выбрать часть наилучших решений из популяции
    4. По выбранным решениям обновить статистическую информацию

В качестве ответа предъявить наилучшее найденное решение.

Поведение муравьиной колонии



- Экспериментальные исследования распределения локальных оптимумов свидетельствуют о высокой концентрации их в непосредственной близости от глобального оптимума.

- Экспериментальные исследования распределения локальных оптимумов свидетельствуют о высокой концентрации их в непосредственной близости от глобального оптимума.
- Это наблюдение известно как гипотеза о существовании «большой долины» для задач на минимум или «центрального горного массива» для задач на максимум.

Гипотеза

Гипотеза

В среднем локальные оптимумы расположены гораздо ближе к глобальному оптимуму, чем к случайно выбранной точке. Их распределение в области допустимых решений не является равномерным. Они концентрируются в районе глобального оптимума, занимая область небольшого диаметра.

- Эта гипотеза отчасти объясняет работоспособность генетических алгоритмов.

Гипотеза

Гипотеза

В среднем локальные оптимумы расположены гораздо ближе к глобальному оптимуму, чем к случайно выбранной точке. Их распределение в области допустимых решений не является равномерным. Они концентрируются в районе глобального оптимума, занимая область небольшого диаметра.

- Эта гипотеза отчасти объясняет работоспособность генетических алгоритмов.
- Если в популяции собираются локальные оптимумы, которые согласно гипотезе сконцентрированы в одном месте, и очередное решение i' выбирается где-то между двумя произвольными локальными оптимумами, то такой процесс имеет много шансов найти глобальный оптимум.

- Аналогичные рассуждения объясняют работоспособность и других локальных алгоритмов.

- Аналогичные рассуждения объясняют работоспособность и других локальных алгоритмов.
- В связи с этим проверка и теоретическое обоснование данной гипотезы представляет несомненный интерес.

Рекомендации при разработке метаэвристик

- Выбрать подходящую окрестность, чтобы быстро находить соседнее решение

Рекомендации при разработке метаэвристик

- Выбрать подходящую окрестность, чтобы быстро находить соседнее решение
- Изображать графически "посещаемые" решения при фиксированных параметрах алгоритма

Рекомендации при разработке метаэвристик

- Выбрать подходящую окрестность, чтобы быстро находить соседнее решение
- Изображать графически “посещаемые” решения при фиксированных параметрах алгоритма
- Проверять не стал ли метод похож на случайный поиск? Не установлены ли параметры алгоритма на максимум?