Локальный поиск. Метаэвристики

Виктор Васильевич Лепин

Обозначения

Формально под *оптимизационной проблемой* будем понимать четверку $\mathcal{P} = (I_{\mathcal{P}}, \mathrm{SOL}_{\mathcal{P}}, m_{\mathcal{P}}, \mathrm{goal}_{\mathcal{P}})$, где:

Формально под *оптимизационной проблемой* будем понимать четверку $\mathcal{P} = (I_{\mathcal{P}}, \mathrm{SOL}_{\mathcal{P}}, m_{\mathcal{P}}, \mathrm{goal}_{\mathcal{P}})$, где:

1) $I_{\mathcal{P}}$ — множество индивидуальных задач для $\mathcal{P}$;

Обозначения

Формально под *оптимизационной проблемой* будем понимать четверку $\mathcal{P} = (I_{\mathcal{P}}, \mathrm{SOL}_{\mathcal{P}}, m_{\mathcal{P}}, \mathrm{goal}_{\mathcal{P}})$, где:

1) $I_{\mathcal{P}}$ — множество индивидуальных задач для $\mathcal{P}$;

2) $\mathrm{SOL}_{\mathcal{P}}$ — отображение, определенное на множестве $I_{\mathcal{P}}$ и такое, что для каждой индивидуальной задачи $x \in I_{\mathcal{P}}$ множество образов $\mathrm{SOL}_{\mathcal{P}}(x)$ является множеством всех решений задачи x;

Обозначения

Формально под *оптимизационной проблемой* будем понимать четверку $\mathcal{P} = (I_{\mathcal{P}}, \mathrm{SOL}_{\mathcal{P}}, m_{\mathcal{P}}, \mathrm{goal}_{\mathcal{P}})$, где:

1) $I_{\mathcal{P}}$ — множество индивидуальных задач для $\mathcal{P}$;

2) $\mathrm{SOL}_{\mathcal{P}}$ — отображение, определенное на множестве $I_{\mathcal{P}}$ и такое, что для каждой индивидуальной задачи $x \in I_{\mathcal{P}}$ множество образов $\mathrm{SOL}_{\mathcal{P}}(x)$ является множеством всех решений задачи x;

3) $m_{\mathcal{P}} : \{(x, s) : x \in I_{\mathcal{P}} \wedge s \in \mathrm{SOL}_{\mathcal{P}}(x)\} \to \mathbb{N}$ — функция меры, называемая также целевой функцией. Для каждой пары (x, s) =(индивидуальная задача, ее решение) значение $m_{\mathcal{P}}(x, s)$ является натуральным числом, которое называется стоимостью решения s;

Обозначения

Формально под *оптимизационной проблемой* будем понимать четверку $\mathcal{P} = (I_{\mathcal{P}}, \mathrm{SOL}_{\mathcal{P}}, m_{\mathcal{P}}, \mathrm{goal}_{\mathcal{P}})$, где:

1) $I_{\mathcal{P}}$ — множество индивидуальных задач для $\mathcal{P}$;

2) $\mathrm{SOL}_{\mathcal{P}}$ — отображение, определенное на множестве $I_{\mathcal{P}}$ и такое, что для каждой индивидуальной задачи $x \in I_{\mathcal{P}}$ множество образов $\mathrm{SOL}_{\mathcal{P}}(x)$ является множеством всех решений задачи x;

3) $m_{\mathcal{P}} : \{(x, s) : x \in I_{\mathcal{P}} \wedge s \in \mathrm{SOL}_{\mathcal{P}}(x)\} \to \mathbb{N}$ — функция меры, называемая также целевой функцией. Для каждой пары (x, s) =(индивидуальная задача, ее решение) значение $m_{\mathcal{P}}(x, s)$ является натуральным числом, которое называется стоимостью решения s;

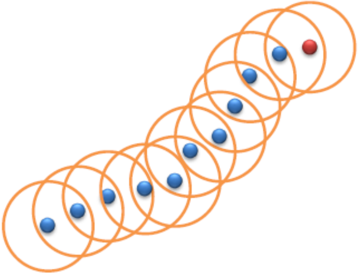
4) goal $\in \{\max, \min\}$ специфицирует, что $\mathcal{P}$ является максимизационной или минимизационной проблемой.

Локальный поиск

- Для любой задачи алгоритм локального поиска начинает свою работу от некоторого исходного решения (найденного каким-то другим алгоритмом или выбранного случайно) и представляет собой итеративный процесс.

Локальный поиск

- Для любой задачи алгоритм локального поиска начинает свою работу от некоторого исходного решения (найденного каким-то другим алгоритмом или выбранного случайно) и представляет собой итеративный процесс.
- На каждом шаге локального спуска происходит переход от текущего решения к соседнему решению с меньшим значением целевой функции до тех пор, пока не будет достигнут локальный оптимум.



- Соседнее решение не обязательно должно быть наилучшим в окрестности, а вот критерий оценки решения не должен меняться по ходу этого итеративного процесса.

- Соседнее решение не обязательно должно быть наилучшим в окрестности, а вот критерий оценки решения не должен меняться по ходу этого итеративного процесса.
- Таким образом, для любого решения s должно быть задано некоторое множество $\mathcal{N}(s)$ *соседних* решений, которое называется окрестностью s. Грубо говоря, окрестность $\mathcal{N}(s)$ состоит из решений, незначительно отличающихся от s.

Локальный поиск

- Соседнее решение не обязательно должно быть наилучшим в окрестности, а вот критерий оценки решения не должен меняться по ходу этого итеративного процесса.
- Таким образом, для любого решения s должно быть задано некоторое множество $\mathcal{N}(s)$ *соседних* решений, которое называется окрестностью s. Грубо говоря, окрестность $\mathcal{N}(s)$ состоит из решений, незначительно отличающихся от s.
- Семейство всех окрестностей называется структурой окрестностей.

Дадим формальные определения.

Определение

Структурой окрестностей для индивидуальной задачи x оптимизационной проблемы $\mathcal{P}$ называется отображение $\mathcal{N}$, которое каждому допустимому решению s задачи ставит в соответствие множество решений $\mathcal{N}(s)$. Структура окрестностей может быть достаточно сложной и отношение соседства не всегда симметрично, т. е. s может быть соседом s', но s' может не быть соседом s.

Локальный поиск

Определение

Решение $s \in \mathrm{SOL}_{\mathcal{P}}(x)$ называется *локально-минимальным (локально-максимальным)* по отношению к структуре окрестностей $\mathcal{N}$, если

$$m_{\mathcal{P}}(s) \leq m_{\mathcal{P}}(s') \;\; (m_{\mathcal{P}}(s) \geq m_{\mathcal{P}}(s')) \text{ для всех } s' \in \mathcal{N}(s).$$

Определение

Структура окрестностей $\mathcal{N}$ называется точной, если $\mathrm{SOL}^{\mathcal{N}}(x) \subseteq \mathrm{SOL}^{*}(x)$.

Для данного решения s поиск более лучшего решения в окрестности $\mathcal{N}(s)$ может быть осуществлен двумя способами:

- (i) *выбор наилучшего* решения во всей окрестности;

Для данного решения s поиск более лучшего решения в окрестности $\mathcal{N}(s)$ может быть осуществлен двумя способами:

- (i) *выбор наилучшего* решения во всей окрестности;
- (ii) *выбор первого лучшего*, когда окрестность каким-либо образом перебирается и этот перебор сразу же прекращается, как только будет найдено более лучшее решение.

Определение

Локальным поиском для оптимизационной проблемы $\mathcal{P}$ называется алгоритм, который для любой индивидуальной задачи $x \in I_{\mathcal{P}}$ просматривает подмножество $SOL_{\mathcal{P}(x)}$, до момента когда будет найдено локально-оптимальное решение относительно структуры окрестностей $\mathcal{N}$.

Пусть $\mathcal{P}$ дискретная оптимизационная проблема, тогда структуру окрестностей можно задать с помощью ориентированного графа.

Определение

Графом соседства (окрестностей) для индивидуальной задачи $x \in I_{\mathcal{P}}$ называется взвешенный ориентированный граф $G_{\mathcal{N}}(x) = (V(x), A)$ с множеством вершин $V(x) = SOL_{\mathcal{P}(x)}$ и множеством дуг $A = \{(s_1, s_2) \mid s_2 \in \mathcal{N}(s_1)\}$. Веса в этом графе приписаны вершинам и равны соответствующим значениям целевой функции.

- Следовательно, окрестность решения s совпадает с замкнутым окружением соответствующей вершины, т. е. $\mathcal{N}(s) = \{s\} \cup \{s' \in V(x) \mid (s, s') \in A\}$.

- Следовательно, окрестность решения s совпадает с замкнутым окружением соответствующей вершины, т. е. $\mathcal{N}(s) = \{s\} \cup \{s' \in V(x) \mid (s, s') \in A\}$.
- Граф окрестностей $G_{\mathcal{N}}(x)$ (neighborhood graph) иногда называют еще ландшафтом целевой функции или просто ландшафтом (landscape, fitness landscape).

- При определении структуры окрестностей $\mathcal{N}$ важно следить за тем, чтобы получающийся граф $G_{\mathcal{N}}(x)$ был строго связен, т. е. для каждой пары вершин s и s' существовал путь из s в s'.

- При определении структуры окрестностей $\mathcal{N}$ важно следить за тем, чтобы получающийся граф $G_{\mathcal{N}}(x)$ был строго связен, т. е. для каждой пары вершин s и s' существовал путь из s в s'.
- Это свойство является важным при анализе асимптотического поведения алгоритмов, например, вероятностных метаэвристик.

- При определении структуры окрестностей $\mathcal{N}$ важно следить за тем, чтобы получающийся граф $G_{\mathcal{N}}(x)$ был строго связен, т. е. для каждой пары вершин s и s' существовал путь из s в s'.
- Это свойство является важным при анализе асимптотического поведения алгоритмов, например, вероятностных метаэвристик.
- Если же это свойство не выполняется, то стараются получить хотя бы свойство *вполне связности*, когда из любой вершины существует путь в вершину $s^* \in \mathrm{SOL}^*(x)$ с минимальным значением целевой функции.

- При определении структуры окрестностей $\mathcal{N}$ важно следить за тем, чтобы получающийся граф $G_{\mathcal{N}}(x)$ был строго связен, т. е. для каждой пары вершин s и s' существовал путь из s в s'.
- Это свойство является важным при анализе асимптотического поведения алгоритмов, например, вероятностных метаэвристик.
- Если же это свойство не выполняется, то стараются получить хотя бы свойство *вполне связности*, когда из любой вершины существует путь в вершину $s^* \in \mathrm{SOL}^*(x)$ с минимальным значением целевой функции.
- Если же и этого свойства нет, то мы теряем уверенность в достижении глобального оптимума локальными методами и должны либо ограничиться локальными оптимумами, либо переопределить функцию окрестности.

- Переходу от одного текущего решения к другому в графе соседства соответствует дуга.

- Переходу от одного текущего решения к другому в графе соседства соответствует дуга.
- Поэтому локальному поиску для задачи x соответствует путь в графе соседства, который заканчивается в локально-оптимальной вершине $s_r \in V(x)$ со свойством $m_{\mathcal{P}}(x, s_r) \leq m_{\mathcal{P}}(x, s)$ для любого s такого, что $(s_r, s) \in A$.

- Переходу от одного текущего решения к другому в графе соседства соответствует дуга.
- Поэтому локальному поиску для задачи x соответствует путь в графе соседства, который заканчивается в локально-оптимальной вершине $s_r \in V(x)$ со свойством $m_{\mathcal{P}}(x, s_r) \leq m_{\mathcal{P}}(x, s)$ для любого s такого, что $(s_r, s) \in A$.
- В худшем случае локальный поиск может исследовать экспоненциальное число решений, прежде чем достигнет локального оптимума.

- Переходу от одного текущего решения к другому в графе соседства соответствует дуга.
- Поэтому локальному поиску для задачи x соответствует путь в графе соседства, который заканчивается в локально-оптимальной вершине $s_r \in V(x)$ со свойством $m_{\mathcal{P}}(x, s_r) \leq m_{\mathcal{P}}(x, s)$ для любого s такого, что $(s_r, s) \in A$.
- В худшем случае локальный поиск может исследовать экспоненциальное число решений, прежде чем достигнет локального оптимума.
- По этой причине часто в эвристике локального поиска используется *правило остановки.*

Все алгоритмы локального поиска основаны на следующей общей схеме.

<u>Схема локального поиска</u>

Input: Индивидуальная задача x;
Output: Решение s;
begin
  $s :=$ исходное допустимое решение s_0;
  (* $\mathcal{N}$ — структура окрестностей *)
  repeat
    Выбрать еще не рассмотренное решение $s' \in \mathcal{N}(s)$;
    if $m(x, s') < m(x, s)$ **then**
      $s := s'$;
  until все решения из $\mathcal{N}$ не будут рассмотрены;
  return s
end.

Отметим, что для данной индивидуальной задачи поведение этого алгоритма зависит от следующих факторов:

1. Структуры окрестностей $\mathcal{N}$. Размер окрестности любого решения должен выбираться на основе компромисса между целью получения хорошего улучшения при каждом переходе к новому текущему решению и целью ограничения времени просмотра одной окрестности. Обычно, для любого решения s, окрестность $\mathcal{N}(s)$ порождается с помощью некоторой операции локального изменения s.

Отметим, что для данной индивидуальной задачи поведение этого алгоритма зависит от следующих факторов:

1. Структуры окрестностей $\mathcal{N}$. Размер окрестности любого решения должен выбираться на основе компромисса между целью получения хорошего улучшения при каждом переходе к новому текущему решению и целью ограничения времени просмотра одной окрестности. Обычно, для любого решения s, окрестность $\mathcal{N}(s)$ порождается с помощью некоторой операции локального изменения s.
2. Начального решения s_0. Его можно находить с помощью алгоритма (например, конструктивной эвристикой), который выдает хорошее допустимое решение, или с помощью процедуры случайной генерации.

Отметим, что для данной индивидуальной задачи поведение этого алгоритма зависит от следующих факторов:

1. Структуры окрестностей $\mathcal{N}$. Размер окрестности любого решения должен выбираться на основе компромисса между целью получения хорошего улучшения при каждом переходе к новому текущему решению и целью ограничения времени просмотра одной окрестности. Обычно, для любого решения s, окрестность $\mathcal{N}(s)$ порождается с помощью некоторой операции локального изменения s.
2. Начального решения s_0. Его можно находить с помощью алгоритма (например, конструктивной эвристикой), который выдает хорошее допустимое решение, или с помощью процедуры случайной генерации.
3. Стратегии выбора новых решений. Например, все решения из $\mathcal{N}(s)$ просматриваются, выбирается лучшее из них и сравнивается с s. Это означает, что если s не локально-оптимально, то осуществляется переход к наилучшему соседу. Или, наоборот, осуществляется переход к первому лучшему решению, найденному в окрестности.

Примеры, которые указывают на важность локального поиска при построении оптимизационных алгоритмов и достаточно общий характер этого подхода.

1) *Линейное программирование.*

- Геометрически алгоритм симплекс метода можно интерпретировать как движение по вершинам многогранника допустимой области.

Примеры, которые указывают на важность локального поиска при построении оптимизационных алгоритмов и достаточно общий характер этого подхода.

1) *Линейное программирование.*

- Геометрически алгоритм симплекс метода можно интерпретировать как движение по вершинам многогранника допустимой области.
- Вершина не является оптимальной, если и только если существует смежная с ней вершина с меньшим значением целевой функции.

Примеры, которые указывают на важность локального поиска при построении оптимизационных алгоритмов и достаточно общий характер этого подхода.

1) *Линейное программирование.*

- Геометрически алгоритм симплекс метода можно интерпретировать как движение по вершинам многогранника допустимой области.
- Вершина не является оптимальной, если и только если существует смежная с ней вершина с меньшим значением целевой функции.
- Алгебраически, предполагая невырожденность задачи, базисное допустимое решение не является оптимальным, если и только если оно может быть улучшено локальным изменением базиса, т. е. заменой одной базисной переменной на небазисную.

Примеры, которые указывают на важность локального поиска при построении оптимизационных алгоритмов и достаточно общий характер этого подхода.

1) *Линейное программирование.*

- Геометрически алгоритм симплекс метода можно интерпретировать как движение по вершинам многогранника допустимой области.
- Вершина не является оптимальной, если и только если существует смежная с ней вершина с меньшим значением целевой функции.
- Алгебраически, предполагая невырожденность задачи, базисное допустимое решение не является оптимальным, если и только если оно может быть улучшено локальным изменением базиса, т. е. заменой одной базисной переменной на небазисную.
- Получающаяся таким образом окрестность является точной и имеет полиномиальную мощность.

2) *Минимальное остовное дерево.*

- Остовное дерево не является оптимальным, если и только если локальной перестройкой, добавляя одно ребро и удаляя из образовавшегося цикла другое ребро, можно получить новое остовное дерево с меньшим суммарным весом.

2) *Минимальное остовное дерево.*

- Остовное дерево не является оптимальным, если и только если локальной перестройкой, добавляя одно ребро и удаляя из образовавшегося цикла другое ребро, можно получить новое остовное дерево с меньшим суммарным весом.
- Операция локальной перестройки задает отношение соседства на множестве остовных деревьев.

2) *Минимальное остовное дерево.*

- Остовное дерево не является оптимальным, если и только если локальной перестройкой, добавляя одно ребро и удаляя из образовавшегося цикла другое ребро, можно получить новое остовное дерево с меньшим суммарным весом.
- Операция локальной перестройки задает отношение соседства на множестве остовных деревьев.
- Окрестность любого дерева имеет полиномиальную мощность, а функция окрестности является точной.

3) *Максимальное паросочетание.*

- Паросочетание не является максимальным, если и только если существует увеличивающий путь.

3) *Максимальное паросочетание.*

- Паросочетание не является максимальным, если и только если существует увеличивающий путь.
- Два паросочетания называют соседними, если их симметрическая разность образует путь.

3) *Максимальное паросочетание.*

- Паросочетание не является максимальным, если и только если существует увеличивающий путь.
- Два паросочетания называют соседними, если их симметрическая разность образует путь.
- Определенная таким образом окрестность является точной и имеет полиномиальную мощность.

3) *Максимальное паросочетание.*

- Паросочетание не является максимальным, если и только если существует увеличивающий путь.
- Два паросочетания называют соседними, если их симметрическая разность образует путь.
- Определенная таким образом окрестность является точной и имеет полиномиальную мощность.
- Аналогичные утверждения справедливы для взвешенных паросочетаний, совершенных паросочетаний минимального веса, задач о максимальном потоке и потоке минимальной стоимости.

- На каждом шаге локального спуска структура окрестностей $\mathcal{N}$ задает множество возможных направлений движения.

- На каждом шаге локального спуска структура окрестностей $\mathcal{N}$ задает множество возможных направлений движения.
- Очень часто это множество состоит из нескольких элементов и имеется определенная свобода в выборе следующего решения.

- На каждом шаге локального спуска структура окрестностей $\mathcal{N}$ задает множество возможных направлений движения.
- Очень часто это множество состоит из нескольких элементов и имеется определенная свобода в выборе следующего решения.
- Правило выбора может оказать существенное влияние на трудоемкость алгоритма и результат его работы.

- На каждом шаге локального спуска структура окрестностей $\mathcal{N}$ задает множество возможных направлений движения.
- Очень часто это множество состоит из нескольких элементов и имеется определенная свобода в выборе следующего решения.
- Правило выбора может оказать существенное влияние на трудоемкость алгоритма и результат его работы.
- Например, в задаче о максимальном потоке алгоритм Форда-Фалкерсона (который тоже можно рассматривать как вариант локального спуска) имеет полиномиальную временную сложность при выборе кратчайшего пути для увеличения потока и экспоненциальную временную сложность без гарантии получить глобальный оптимум при произвольном выборе пути.

- На каждом шаге локального спуска структура окрестностей $\mathcal{N}$ задает множество возможных направлений движения.
- Очень часто это множество состоит из нескольких элементов и имеется определенная свобода в выборе следующего решения.
- Правило выбора может оказать существенное влияние на трудоемкость алгоритма и результат его работы.
- Например, в задаче о максимальном потоке алгоритм Форда-Фалкерсона (который тоже можно рассматривать как вариант локального спуска) имеет полиномиальную временную сложность при выборе кратчайшего пути для увеличения потока и экспоненциальную временную сложность без гарантии получить глобальный оптимум при произвольном выборе пути.
- Таким образом, при разработке алгоритмов локального поиска важно не только правильно определить окрестность, но и верно задать правило выбора направления спуска.

- Интуитивно кажется, что в окрестности надо брать элемент с наименьшим значением целевой функции.

- Интуитивно кажется, что в окрестности надо брать элемент с наименьшим значением целевой функции.
- Однако, как мы увидим ниже, разумным оказывается не только такой выбор, но и движение в «абсурдном» направлении, когда несколько шагов с ухудшением могут привести (и часто приводят) к лучшему локальному оптимуму.

- Интуитивно кажется, что в окрестности надо брать элемент с наименьшим значением целевой функции.
- Однако, как мы увидим ниже, разумным оказывается не только такой выбор, но и движение в «абсурдном» направлении, когда несколько шагов с ухудшением могут привести (и часто приводят) к лучшему локальному оптимуму.
- При выборе окрестности хочется иметь множество $\mathcal{N}(s)$ как можно меньшей мощности, чтобы сократить трудоемкость одного шага.

- Интуитивно кажется, что в окрестности надо брать элемент с наименьшим значением целевой функции.
- Однако, как мы увидим ниже, разумным оказывается не только такой выбор, но и движение в «абсурдном» направлении, когда несколько шагов с ухудшением могут привести (и часто приводят) к лучшему локальному оптимуму.
- При выборе окрестности хочется иметь множество $\mathcal{N}(s)$ как можно меньшей мощности, чтобы сократить трудоемкость одного шага.
- С другой стороны, более широкая окрестность, вообще говоря, приводит к лучшему локальному оптимуму.

- Интуитивно кажется, что в окрестности надо брать элемент с наименьшим значением целевой функции.
- Однако, как мы увидим ниже, разумным оказывается не только такой выбор, но и движение в «абсурдном» направлении, когда несколько шагов с ухудшением могут привести (и часто приводят) к лучшему локальному оптимуму.
- При выборе окрестности хочется иметь множество $\mathcal{N}(s)$ как можно меньшей мощности, чтобы сократить трудоемкость одного шага.
- С другой стороны, более широкая окрестность, вообще говоря, приводит к лучшему локальному оптимуму.
- Поэтому при создании алгоритмов каждый раз приходится искать оптимальный баланс между этими противоречивыми факторами.

- Интуитивно кажется, что в окрестности надо брать элемент с наименьшим значением целевой функции.
- Однако, как мы увидим ниже, разумным оказывается не только такой выбор, но и движение в «абсурдном» направлении, когда несколько шагов с ухудшением могут привести (и часто приводят) к лучшему локальному оптимуму.
- При выборе окрестности хочется иметь множество $\mathcal{N}(s)$ как можно меньшей мощности, чтобы сократить трудоемкость одного шага.
- С другой стороны, более широкая окрестность, вообще говоря, приводит к лучшему локальному оптимуму.
- Поэтому при создании алгоритмов каждый раз приходится искать оптимальный баланс между этими противоречивыми факторами.
- Ясных принципов разрешения этого противоречия на сегодняшний день не известно, и для каждой задачи этот вопрос решается индивидуально.

Локальный поиск для задачи о максимальном разрезе

Рассмотрим задачу о максимальном разрезе и построим алгоритм локального поиска, который находит решение этой задачи, отличающееся от оптимального не более чем в два раза.

Максимальный Разрез

Индивидуальная задача: Граф $G = (V, E)$.

Решение: Разбиение V на два множества A и B.

Критерий: Мощность разреза, т.е. число ребер, имеющих одну концевую вершину в A, а другую в B.

- Прежде всего мы должны определить процедуру получения исходного допустимого решения и описать структуру окрестностей.

- Прежде всего мы должны определить процедуру получения исходного допустимого решения и описать структуру окрестностей.
- Простой выбор в качестве исходного допустимого решения пары (A, B), где $A = \emptyset$ и $B = V$, решает первую задачу тривиально.

- Прежде всего мы должны определить процедуру получения исходного допустимого решения и описать структуру окрестностей.
- Простой выбор в качестве исходного допустимого решения пары (A, B), где $A = \emptyset$ и $B = V$, решает первую задачу тривиально.
- Структуру окрестностей $\mathcal{N}$ для этой задачи определим так: окрестность решения (A, B) состоит из всех таких разбиений (A_i, B_i), $i = 1, \ldots, |V|$, что

- Прежде всего мы должны определить процедуру получения исходного допустимого решения и описать структуру окрестностей.
- Простой выбор в качестве исходного допустимого решения пары (A, B), где $A = \emptyset$ и $B = V$, решает первую задачу тривиально.
- Структуру окрестностей $\mathcal{N}$ для этой задачи определим так: окрестность решения (A, B) состоит из всех таких разбиений (A_i, B_i), $i = 1, \ldots, |V|$, что
  1. если вершина $v_i \in A$, то $A_i = A - \{v_i\}$ и $B_i = B \cup \{v_i\}$;

- Прежде всего мы должны определить процедуру получения исходного допустимого решения и описать структуру окрестностей.
- Простой выбор в качестве исходного допустимого решения пары (A, B), где $A = \emptyset$ и $B = V$, решает первую задачу тривиально.
- Структуру окрестностей $\mathcal{N}$ для этой задачи определим так: окрестность решения (A, B) состоит из всех таких разбиений (A_i, B_i), $i = 1, \ldots, |V|$, что
    1. если вершина $v_i \in A$, то $A_i = A - \{v_i\}$ и $B_i = B \cup \{v_i\}$;
    2. если вершина $v_i \notin A$, то $A_i = A \cup \{v_i\}$ и $B_i = B - \{v_i\}$.

- Прежде всего мы должны определить процедуру получения исходного допустимого решения и описать структуру окрестностей.
- Простой выбор в качестве исходного допустимого решения пары (A, B), где $A = \emptyset$ и $B = V$, решает первую задачу тривиально.
- Структуру окрестностей $\mathcal{N}$ для этой задачи определим так: окрестность решения (A, B) состоит из всех таких разбиений (A_i, B_i), $i = 1, \ldots, |V|$, что
  1. если вершина $v_i \in A$, то $A_i = A - \{v_i\}$ и $B_i = B \cup \{v_i\}$;
  2. если вершина $v_i \notin A$, то $A_i = A \cup \{v_i\}$ и $B_i = B - \{v_i\}$.

- Прежде всего мы должны определить процедуру получения исходного допустимого решения и описать структуру окрестностей.
- Простой выбор в качестве исходного допустимого решения пары (A, B), где $A = \emptyset$ и $B = V$, решает первую задачу тривиально.
- Структуру окрестностей $\mathcal{N}$ для этой задачи определим так: окрестность решения (A, B) состоит из всех таких разбиений (A_i, B_i), $i = 1, \dots, |V|$, что
  1. если вершина $v_i \in A$, то $A_i = A - \{v_i\}$ и $B_i = B \cup \{v_i\}$;
  2. если вершина $v_i \notin A$, то $A_i = A \cup \{v_i\}$ и $B_i = B - \{v_i\}$.

Приведенная структура окрестностей имеет следующее важное свойство: стоимость каждого локально-оптимального решения больше половины стоимости оптимального решения.

Theorem

Пусть $x =' G = (V, E)'$ *индивидуальная задача проблемы Максимальный Разрез и пусть* (A, B) *локально-оптимальное решение этой задачи относительно структуры окрестностей* $\mathcal{N}$, *имеющее стоимость* $m_N(G)$. *Тогда*

$$m^*(G)/m_N(G) \leq 2.$$

Theorem

Пусть $x =' G = (V, E)'$ *индивидуальная задача проблемы Максимальный Разрез и пусть* (A, B) *локально-оптимальное решение этой задачи относительно структуры окрестностей* $\mathcal{N}$, *имеющее стоимость* $m_N(G)$. *Тогда*

$$m^*(G)/m_N(G) \leq 2.$$

Доказательство.

- Пусть $q = |E|$ число ребер графа G.

Theorem

Пусть $x =' G = (V, E)'$ *индивидуальная задача проблемы Максимальный Разрез и пусть* (A, B) *локально-оптимальное решение этой задачи относительно структуры окрестностей* $\mathcal{N}$, *имеющее стоимость* $m_N(G)$. *Тогда*

$$m^*(G)/m_N(G) \leq 2.$$

Доказательство.

- Пусть $q = |E|$ число ребер графа G.
- Поскольку $m^*(G) \leq q$, то достаточно показать, что $m_N(G) \geq q/2$.

Theorem

Пусть $x =' G = (V, E)'$ *индивидуальная задача проблемы Максимальный Разрез и пусть* (A, B) *локально-оптимальное решение этой задачи относительно структуры окрестностей* $\mathcal{N}$*, имеющее стоимость* $m_N(G)$*. Тогда*

$$m^*(G)/m_N(G) \leq 2.$$

Доказательство.

- Пусть $q = |E|$ число ребер графа G.
- Поскольку $m^*(G) \leq q$, то достаточно показать, что $m_N(G) \geq q/2$.
- Пусть q_A — число всех ребер графа, имеющих концевые вершины в множестве A, и q_B — число всех ребер графа, имеющих концевые вершины в множестве B.

Theorem

Пусть $x =' G = (V, E)'$ *индивидуальная задача проблемы Максимальный Разрез и пусть* (A, B) *локально-оптимальное решение этой задачи относительно структуры окрестностей* $\mathcal{N}$, *имеющее стоимость* $m_N(G)$. *Тогда*

$$m^*(G)/m_N(G) \leq 2.$$

Доказательство.

- Пусть $q = |E|$ число ребер графа G.
- Поскольку $m^*(G) \leq q$, то достаточно показать, что $m_N(G) \geq q/2$.
- Пусть q_A — число всех ребер графа, имеющих концевые вершины в множестве A, и q_B — число всех ребер графа, имеющих концевые вершины в множестве B.
- Тогда

$$q = q_A + q_B + m_N(G). \tag{1}$$

- Для любой вершины $v \in V$ множество смежных с ней вершин можно разбить на два множества:

$$U_A(v) = \{u \mid u \in A \land (v, u) \in E\}$$

и

$$U_B(v) = \{u \mid u \in B \land (v, u) \in E\}.$$

- Для любой вершины $v \in V$ множество смежных с ней вершин можно разбить на два множества:

$$U_A(v) = \{u \mid u \in A \wedge (v, u) \in E\}$$

и

$$U_B(v) = \{u \mid u \in B \wedge (v, u) \in E\}.$$

- Если (A, B) — локальный оптимум, то любое разбиение (A_i, B_i) из его окрестности имеет стоимость не большую, чем $m_N(G)$.

- Для любой вершины $v \in V$ множество смежных с ней вершин можно разбить на два множества:

$$U_A(v) = \{u \mid u \in A \wedge (v, u) \in E\}$$

и

$$U_B(v) = \{u \mid u \in B \wedge (v, u) \in E\}.$$

- Если (A, B) — локальный оптимум, то любое разбиение (A_i, B_i) из его окрестности имеет стоимость не большую, чем $m_N(G)$.
- Поэтому для каждой вершины $v \in A$ верно неравенство

$$|U_A(v)| - |U_B(v)| \leq 0$$

и для каждой вершины $w \in B$ верно неравенство

$$|U_B(w)| - |U_A(w)| \leq 0.$$

- Для любой вершины $v \in V$ множество смежных с ней вершин можно разбить на два множества:

$$U_A(v) = \{u \mid u \in A \wedge (v, u) \in E\}$$

и

$$U_B(v) = \{u \mid u \in B \wedge (v, u) \in E\}.$$

- Если (A, B) — локальный оптимум, то любое разбиение (A_i, B_i) из его окрестности имеет стоимость не большую, чем $m_N(G)$.
- Поэтому для каждой вершины $v \in A$ верно неравенство

$$|U_A(v)| - |U_B(v)| \leq 0$$

и для каждой вершины $w \in B$ верно неравенство

$$|U_B(w)| - |U_A(w)| \leq 0.$$

- Просуммировав такие неравенства соответственно по вершинам из A и B, получим, что

$$\sum_{v \in A} (|U_A(v)| - |U_B(v)|) = 2q_A - m_N(G) \leq 0$$

- Следовательно, $q_A + q_B - m_N(G) \leq 0$. Из этого неравенства, учитывая (1), получаем, что $m_N(G) \geq q/2$, и, следовательно, теорема доказана.

Для рассмотренной проблемы, так же как и для любой другой, можно по-разному определить структуру окрестностей. Основные факторы, на которые следует обращать внимание при выборе структуры окрестностей, следующие:

- Следовательно, $q_A + q_B - m_N(G) \leq 0$. Из этого неравенства, учитывая (1), получаем, что $m_N(G) \geq q/2$, и, следовательно, теорема доказана.

Для рассмотренной проблемы, так же как и для любой другой, можно по-разному определить структуру окрестностей. Основные факторы, на которые следует обращать внимание при выборе структуры окрестностей, следующие:

- качество получаемого решения (т. е., как близка стоимость локального оптимума к стоимости глобального оптимума);

- Следовательно, $q_A + q_B - m_N(G) \leq 0$. Из этого неравенства, учитывая (1), получаем, что $m_N(G) \geq q/2$, и, следовательно, теорема доказана.

Для рассмотренной проблемы, так же как и для любой другой, можно по-разному определить структуру окрестностей. Основные факторы, на которые следует обращать внимание при выборе структуры окрестностей, следующие:

- качество получаемого решения (т. е., как близка стоимость локального оптимума к стоимости глобального оптимума);
- порядок, в каком будет исследоваться окрестность;

- Следовательно, $q_A + q_B - m_N(G) \leq 0$. Из этого неравенства, учитывая (1), получаем, что $m_N(G) \geq q/2$, и, следовательно, теорема доказана.

Для рассмотренной проблемы, так же как и для любой другой, можно по-разному определить структуру окрестностей. Основные факторы, на которые следует обращать внимание при выборе структуры окрестностей, следующие:

- качество получаемого решения (т. е., как близка стоимость локального оптимума к стоимости глобального оптимума);
- порядок, в каком будет исследоваться окрестность;
- сложность проверки условия, что окрестность не содержит лучших решений;

- Следовательно, $q_A + q_B - m_N(G) \leq 0$. Из этого неравенства, учитывая (1), получаем, что $m_N(G) \geq q/2$, и, следовательно, теорема доказана.

Для рассмотренной проблемы, так же как и для любой другой, можно по-разному определить структуру окрестностей. Основные факторы, на которые следует обращать внимание при выборе структуры окрестностей, следующие:

- качество получаемого решения (т. е., как близка стоимость локального оптимума к стоимости глобального оптимума);
- порядок, в каком будет исследоваться окрестность;
- сложность проверки условия, что окрестность не содержит лучших решений;
- число решений, генерируемых до момента, когда будет найдено локально-оптимальное решение.

Учитывать эти факторы следует на основе компромисса. Например, если структура окрестностей является "большой то вероятно, что стоимость найденного решения близка к оптимальной стоимости. Однако в этом случае задача проверки условия, что более предпочтительного по стоимости решения в окрестности не существует, становится сложнее.

Сложность локального поиска

Анализ вычислительной сложности локального поиска в последние годы интенсивно ведется в двух направлениях: эмпирическом и теоретическом. Как ни странно, но эти направления дают существенно разные оценки возможностям локального поиска.

Эмпирические результаты.

- Для многих NP-трудных задач локальный поиск позволяет находить приближенные решения, близкие по целевой функции к глобальному оптимуму.

Анализ вычислительной сложности локального поиска в последние годы интенсивно ведется в двух направлениях: эмпирическом и теоретическом. Как ни странно, но эти направления дают существенно разные оценки возможностям локального поиска.

Эмпирические результаты.

- Для многих NP-трудных задач локальный поиск позволяет находить приближенные решения, близкие по целевой функции к глобальному оптимуму.
- Трудоемкость алгоритмов часто оказывается полиномиальной, причем степень полинома достаточно мала.

Анализ вычислительной сложности локального поиска в последние годы интенсивно ведется в двух направлениях: эмпирическом и теоретическом. Как ни странно, но эти направления дают существенно разные оценки возможностям локального поиска.

Эмпирические результаты.

- Для многих NP-трудных задач локальный поиск позволяет находить приближенные решения, близкие по целевой функции к глобальному оптимуму.
- Трудоемкость алгоритмов часто оказывается полиномиальной, причем степень полинома достаточно мала.
- Так для задачи о разбиении множества вершин графа на две равные части разработаны алгоритмы локального поиска со средней трудоемкостью $O(n \log n)$, n — число вершин, которые дают всего несколько процентов погрешности.

- Для задачи коммивояжера алгоритмы локального поиска являются наилучшими с практической точки зрения.

- Для задачи коммивояжера алгоритмы локального поиска являются наилучшими с практической точки зрения.
- Один из таких алгоритмов с окрестностью Лина-Кернигхана в среднем имеет погрешность около 2%, и максимальная размерность решаемых задач достигает 1 000 000 городов.

- Для задачи коммивояжера алгоритмы локального поиска являются наилучшими с практической точки зрения.
- Один из таких алгоритмов с окрестностью Лина-Кернигхана в среднем имеет погрешность около 2%, и максимальная размерность решаемых задач достигает 1 000 000 городов.
- На случайно сгенерированных задачах такой колоссальной размерности итерационная процедура Джонсона позволяет находить решения с отклонением около 0,5% за несколько минут на современных компьютерах.

- Для задачи коммивояжера алгоритмы локального поиска являются наилучшими с практической точки зрения.
- Один из таких алгоритмов с окрестностью Лина-Кернигхана в среднем имеет погрешность около 2%, и максимальная размерность решаемых задач достигает 1 000 000 городов.
- На случайно сгенерированных задачах такой колоссальной размерности итерационная процедура Джонсона позволяет находить решения с отклонением около 0,5% за несколько минут на современных компьютерах.
- Для реальных задач с числом городов до 100 000 существуют алгоритмы с аналогичными характеристиками.

- Для задач теории расписаний, размещения, покрытия, раскраски и многих других NP-трудных задач алгоритмы локального поиска показывают превосходные результаты.

- Для задач теории расписаний, размещения, покрытия, раскраски и многих других NP-трудных задач алгоритмы локального поиска показывают превосходные результаты.
- Более того, их гибкость при изменении математической модели, простота реализации и наглядность превращают локальный поиск в мощное средство для решения практических задач.

Теоретические результаты.

- Исследование локального поиска с точки зрения гарантированных оценок качества показывают границы его возможностей.

Теоретические результаты.

- Исследование локального поиска с точки зрения гарантированных оценок качества показывают границы его возможностей.
- Построены трудные для локального поиска примеры, из которых следует, что

Теоретические результаты.

- Исследование локального поиска с точки зрения гарантированных оценок качества показывают границы его возможностей.
- Построены трудные для локального поиска примеры, из которых следует, что
  1. минимальная точная окрестность может иметь экспоненциальную мощность;

Теоретические результаты.

- Исследование локального поиска с точки зрения гарантированных оценок качества показывают границы его возможностей.
- Построены трудные для локального поиска примеры, из которых следует, что
  1. минимальная точная окрестность может иметь экспоненциальную мощность;
  2. число шагов для достижения локального оптимума может оказаться экспоненциальным;

Теоретические результаты.

- Исследование локального поиска с точки зрения гарантированных оценок качества показывают границы его возможностей.
- Построены трудные для локального поиска примеры, из которых следует, что
  1. минимальная точная окрестность может иметь экспоненциальную мощность;
  2. число шагов для достижения локального оптимума может оказаться экспоненциальным;
  3. значение локального оптимума может сколь угодно сильно отличаться от глобального оптимума.

Окрестности, основанные на структурной близости решений

- Выбор окрестности играет важную роль при построении алгоритмов локального поиска.

- Выбор окрестности играет важную роль при построении алгоритмов локального поиска.
- От него существенно зависит трудоемкость одного шага алгоритма, общее число шагов и в конечном счете качество получаемого локального оптимума.

- Выбор окрестности играет важную роль при построении алгоритмов локального поиска.
- От него существенно зависит трудоемкость одного шага алгоритма, общее число шагов и в конечном счете качество получаемого локального оптимума.
- На сегодняшний день нет и, возможно никогда не будет, единого правила выбора окрестности.

- Выбор окрестности играет важную роль при построении алгоритмов локального поиска.
- От него существенно зависит трудоемкость одного шага алгоритма, общее число шагов и в конечном счете качество получаемого локального оптимума.
- На сегодняшний день нет и, возможно никогда не будет, единого правила выбора окрестности.
- Для каждой задачи структуру окрестностей $\mathcal{N}$ приходится определять заново, учитывая специфику данной задачи.

- Выбор окрестности играет важную роль при построении алгоритмов локального поиска.
- От него существенно зависит трудоемкость одного шага алгоритма, общее число шагов и в конечном счете качество получаемого локального оптимума.
- На сегодняшний день нет и, возможно никогда не будет, единого правила выбора окрестности.
- Для каждой задачи структуру окрестностей $\mathcal{N}$ приходится определять заново, учитывая специфику данной задачи.
- Более того, для каждой задачи можно предложить несколько структур окрестностей с разными по мощности множествами $\mathcal{N}(s)$ и, как следствие, разными множествами локальных оптимумов.

- Выбор окрестности играет важную роль при построении алгоритмов локального поиска.
- От него существенно зависит трудоемкость одного шага алгоритма, общее число шагов и в конечном счете качество получаемого локального оптимума.
- На сегодняшний день нет и, возможно никогда не будет, единого правила выбора окрестности.
- Для каждой задачи структуру окрестностей $\mathcal{N}$ приходится определять заново, учитывая специфику данной задачи.
- Более того, для каждой задачи можно предложить несколько структур окрестностей с разными по мощности множествами $\mathcal{N}(s)$ и, как следствие, разными множествами локальных оптимумов.
- Ниже будут приведены три примера выбора окрестностей для задачи коммивояжера, которые иллюстрируют возможные пути построения окрестностей и их свойства.

- Напомним, что задача коммивояжера состоит в нахождении минимального по длине гамильтонова цикла в полном взвешенном ориентированном (или неориентированном) графе с n вершинами.

- Напомним, что задача коммивояжера состоит в нахождении минимального по длине гамильтонова цикла в полном взвешенном ориентированном (или неориентированном) графе с n вершинами.
- Для удобства ниже будут рассматриваться только неориентированные графы с симметричной целочисленной неотрицательной матрицей расстояний $w_{ij} = w_{ji}, 1 \leq i, j \leq n$.

- Напомним, что задача коммивояжера состоит в нахождении минимального по длине гамильтонова цикла в полном взвешенном ориентированном (или неориентированном) графе с n вершинами.
- Для удобства ниже будут рассматриваться только неориентированные графы с симметричной целочисленной неотрицательной матрицей расстояний $w_{ij} = w_{ji}, 1 \le i, j \le n$.
- Каждое допустимое решение задачи коммивояжера или тур в графах будем представлять в виде перестановки $\pi = \{i_1, \dots i_n\}$. Для $\pi \in \mathrm{SOL}$ определим значение функции окрестности $N(\pi)$ как множество всех перестановок, отличающихся от π только в двух позициях (*city-swap*).

- Напомним, что задача коммивояжера состоит в нахождении минимального по длине гамильтонова цикла в полном взвешенном ориентированном (или неориентированном) графе с n вершинами.
- Для удобства ниже будут рассматриваться только неориентированные графы с симметричной целочисленной неотрицательной матрицей расстояний $w_{ij} = w_{ji}, 1 \leq i, j \leq n$.
- Каждое допустимое решение задачи коммивояжера или тур в графах будем представлять в виде перестановки $\pi = \{i_1, \ldots i_n\}$. Для $\pi \in \mathrm{SOL}$ определим значение функции окрестности $N(\pi)$ как множество всех перестановок, отличающихся от π только в двух позициях (*city-swap*).
- Множество $N(\pi)$ содержит ровно $n(n-1)/2$ элементов, и вычислительная сложность одного шага локального поиска с учетом вычисления целевой функции не превосходит $O(n^2)$ операций.

- Обозначим через $W(\pi)$ длину гамильтонова цикла и определим разностный оператор Δ^2 для $W(\pi)$ следующим образом:

$$\Delta^2 W(\pi) = \frac{1}{|\mathcal{N}(\pi)|} \sum_{\pi' \in \mathcal{N}(\pi)} W(\pi') - W(\pi), \quad \pi \in \mathrm{SOL}.$$

Theorem

Функция $\widetilde{W} = W - W_{av}$ *удовлетворяет уравнению*

$$\Delta^2 \widetilde{W} = -\frac{4}{n} \widetilde{W}.$$

- Обозначим через $W(\pi)$ длину гамильтонова цикла и определим разностный оператор Δ^2 для $W(\pi)$ следующим образом:

$$\Delta^2 W(\pi) = \frac{1}{|\mathcal{N}(\pi)|} \sum_{\pi' \in \mathcal{N}(\pi)} W(\pi') - W(\pi), \quad \pi \in \mathrm{SOL}.$$

- Оператор $\Delta^2 W(\pi)$ задает среднее отклонение целевой функции $W(\pi)$ в окрестности данной перестановки π.

Theorem

Функция $\widetilde{W} = W - W_{av}$ *удовлетворяет уравнению*

$$\Delta^2 \widetilde{W} = -\frac{4}{n} \widetilde{W}.$$

- Обозначим через $W(\pi)$ длину гамильтонова цикла и определим разностный оператор Δ^2 для $W(\pi)$ следующим образом:

$$\Delta^2 W(\pi) = \frac{1}{|\mathcal{N}(\pi)|} \sum_{\pi' \in \mathcal{N}(\pi)} W(\pi') - W(\pi), \quad \pi \in \mathrm{SOL}.$$

- Оператор $\Delta^2 W(\pi)$ задает среднее отклонение целевой функции $W(\pi)$ в окрестности данной перестановки π.
- Для любого оптимума $\pi \in \mathrm{SOL}^{\mathcal{N}}$ справедливо неравенство $\Delta^2 W(\pi) \geq 0$. Пусть W_{av} — средняя длина тура на множестве всех допустимых решений SOL.

Theorem

Функция $\widetilde{W} = W - W_{av}$ *удовлетворяет уравнению*

$$\Delta^2 \widetilde{W} = -\frac{4}{n} \widetilde{W}.$$

- Обозначим через $W(\pi)$ длину гамильтонова цикла и определим разностный оператор Δ^2 для $W(\pi)$ следующим образом:

$$\Delta^2 W(\pi) = \frac{1}{|\mathcal{N}(\pi)|} \sum_{\pi' \in \mathcal{N}(\pi)} W(\pi') - W(\pi), \quad \pi \in \mathrm{SOL}.$$

- Оператор $\Delta^2 W(\pi)$ задает среднее отклонение целевой функции $W(\pi)$ в окрестности данной перестановки π.
- Для любого оптимума $\pi \in \mathrm{SOL}^{\mathcal{N}}$ справедливо неравенство $\Delta^2 W(\pi) \geq 0$. Пусть W_{av} — средняя длина тура на множестве всех допустимых решений SOL.
- Тогда справедливы следующие утверждения.

Theorem

Функция $\widetilde{W} = W - W_{av}$ *удовлетворяет уравнению*

$$\Delta^2 \widetilde{W} = -\frac{4}{n} \widetilde{W}.$$

Corollary

Любой локальный минимум $\pi \in SOL^{\mathcal{N}}$ *имеет длину* $W(\pi) \leq W_{av}$.

Corollary

Алгоритм локального поиска, начиная с произвольной перестановки, достигает локального оптимума за $O(nW)$ *шагов, если длина максимального тура превосходит среднее значение* W_{av} *не более чем в* 2^W *раз.*

Локальный поиск фиксированной глубины

- В эвристиках локального поиска фиксированной глубины используются окрестности, определяемые с помощью последовательностей ограниченной длинны операций локального обмена.

- В эвристиках локального поиска фиксированной глубины используются окрестности, определяемые с помощью последовательностей ограниченной длинны операций локального обмена.
- Пусть, при фиксированным целом $k > 0$, решается этим методом индивидуальная задача $x \in I_{\mathcal{P}}$ проблемы $\mathcal{P}$.

- В эвристиках локального поиска фиксированной глубины используются окрестности, определяемые с помощью последовательностей ограниченной длинны операций локального обмена.
- Пусть, при фиксированным целом $k > 0$, решается этим методом индивидуальная задача $x \in I_{\mathcal{P}}$ проблемы $\mathcal{P}$.
- Тогда говорят, что решение y находится в *k-обменной окрестности* $\mathcal{N}^k(s)$, если из решения s можно получить решение y, применив не более чем k операций локального обмена.

- В эвристиках локального поиска фиксированной глубины используются окрестности, определяемые с помощью последовательностей ограниченной длинны операций локального обмена.
- Пусть, при фиксированным целом $k > 0$, решается этим методом индивидуальная задача $x \in I_{\mathcal{P}}$ проблемы $\mathcal{P}$.
- Тогда говорят, что решение y находится в *k-обменной окрестности* $\mathcal{N}^k(s)$, если из решения s можно получить решение y, применив не более чем k операций локального обмена.
- Эвристики, основанные на k-обменных окрестностях, часто называют *k-оптимальными* (k-opt) эвристиками.

- Рассмотрим 2-оптимальную эвристику для задачи о коммивояжере.

- Рассмотрим 2-оптимальную эвристику для задачи о коммивояжере.
- Этот метод основан на следующей 2-обменной окрестности: для любого тура τ 2-обменная окрестность $\mathcal{N}(\tau)$ это множество всех туров τ', которые могут быть получены из τ после удаления двух ребер (x, y) и (v, z) и добавления двух новых ребер (x, v) и (z, y).

- Рассмотрим 2-оптимальную эвристику для задачи о коммивояжере.
- Этот метод основан на следующей 2-обменной окрестности: для любого тура τ 2-обменная окрестность $\mathcal{N}(\tau)$ это множество всех туров τ', которые могут быть получены из τ после удаления двух ребер (x, y) и (v, z) и добавления двух новых ребер (x, v) и (z, y).
- Тот же процесс построения нового тура τ' из тура τ можно описать так: выделяется некоторая цепь, которая в новом туре проходится в противоположном направлении.

- Рассмотрим 2-оптимальную эвристику для задачи о коммивояжере.
- Этот метод основан на следующей 2-обменной окрестности: для любого тура τ 2-обменная окрестность $\mathcal{N}(\tau)$ это множество всех туров τ', которые могут быть получены из τ после удаления двух ребер (x, y) и (v, z) и добавления двух новых ребер (x, v) и (z, y).
- Тот же процесс построения нового тура τ' из тура τ можно описать так: выделяется некоторая цепь, которая в новом туре проходится в противоположном направлении.
- Такая окрестность содержит $n(n-3)/2$ элементов, что несколько меньше, чем в окрестности city-swap.

- Алгоритм 2-opt реализует 2-оптимальную эвристику для задачи о коммивояжере и является примером локального поиска с 2-обменной окрестностью.

- Алгоритм 2-opt реализует 2-оптимальную эвристику для задачи о коммивояжере и является примером локального поиска с 2-обменной окрестностью.
- В алгоритме используется функция l, определенная на множестве туров так:

$$l(\tau) = \sum_{i=1}^{n-1} D(\pi(i), \pi(i+1)) + D(\pi(n), \pi(1)),$$

где $\tau = \langle c_{\pi(1)}, \ldots, c_{\pi(n)} \rangle$.

Алгоритм 2-opt

Input: Множество городов $C = \{c_1, \ldots, c_n\}$, $n \times n$ матрица D расстояний.

Output: Перестановка $T = (c_{\pi_1}, \ldots, c_{\pi_n})$ городов.

begin

$\tau :=$ исходный тур τ_0;

Пусть $Q = \{(i,j) \mid i,j \in \{1,\ldots,n\}$ и $i \neq j\}$;

$N := Q$;

repeat

Пусть $\tau = (c_{i_1}, \ldots, c_{i_n})$;

Выбрать пару индексов $(p,q) \in N$;

$N := N - \{(p,q)\}$;

$\tau' := (c_{i_1}, \ldots, c_{i_{p-1}}, c_{i_q}, c_{i_{q-1}}, \ldots, c_{i_{p+1}}, c_{i_p}, c_{i_{q+1}}, \ldots, c_{i_n})$;

if $l(\tau') < l(\tau)$ **then**

begin

$\tau := \tau'$;

$N := Q$

end

until $N = \emptyset$;

return τ

end.

- Алгоритм 2-opt можно модифицировать в алгоритм локального поиска с большими окрестностями.

- Алгоритм 2-opt можно модифицировать в алгоритм локального поиска с большими окрестностями.
- Например, 3-opt эвристика основывается на 3-обменных окрестностях.

- Алгоритм 2-opt можно модифицировать в алгоритм локального поиска с большими окрестностями.
- Например, 3-opt эвристика основывается на 3-обменных окрестностях.
- Для тура τ его 3-обменная окрестность $\mathcal{N}(\tau)$ это множество всех туров τ', которые могут быть получены из τ после замены не более чем трех ребер.

- Алгоритм 2-opt можно модифицировать в алгоритм локального поиска с большими окрестностями.
- Например, 3-opt эвристика основывается на 3-обменных окрестностях.
- Для тура τ его 3-обменная окрестность $\mathcal{N}(\tau)$ это множество всех туров τ', которые могут быть получены из τ после замены не более чем трех ребер.
- Алгоритм 3-opt эвристика имеет более лучшее приближение, но хуже по трудоемкости.

- Алгоритм 2-opt можно модифицировать в алгоритм локального поиска с большими окрестностями.
- Например, 3-opt эвристика основывается на 3-обменных окрестностях.
- Для тура τ его 3-обменная окрестность $\mathcal{N}(\tau)$ это множество всех туров τ', которые могут быть получены из τ после замены не более чем трех ребер.
- Алгоритм 3-opt эвристика имеет более лучшее приближение, но хуже по трудоемкости.
- Отметим, что для задачи коммивояжера с n городами k-обменная окрестность имеет размер $\binom{n}{k} = \Theta(n^k)$.

- Алгоритм 2-opt можно модифицировать в алгоритм локального поиска с большими окрестностями.
- Например, 3-opt эвристика основывается на 3-обменных окрестностях.
- Для тура τ его 3-обменная окрестность $\mathcal{N}(\tau)$ это множество всех туров τ', которые могут быть получены из τ после замены не более чем трех ребер.
- Алгоритм 3-opt эвристика имеет более лучшее приближение, но хуже по трудоемкости.
- Отметим, что для задачи коммивояжера с n городами k-обменная окрестность имеет размер $\binom{n}{k} = \Theta(n^k)$.
- Следовательно, эвристике требуется выполнить $O(n^k)$ шагов для того, чтобы удостовериться, что текущее решение является локально оптимальным.

- На практике эвристики локального поиска фиксированной глубины для задачи коммивояжера весьма эффективны.

Theorem

Для любых $k \geq 2$, $n \geq 2k+8$ и $\alpha > 1/n$ существует пример x задачи коммивояжера с n городами $\{c_1, \ldots, c_n\}$, такой, что эвристика k-opt для входа x с начальным туром $\tau_0 = (c_1, \ldots, c_n)$ находит тур стоимость, которого $m_k(x)$ удовлетворяет неравенству $m_k(x) > \alpha m^*(x)$.

- На практике эвристики локального поиска фиксированной глубины для задачи коммивояжера весьма эффективны.
- Однако можно построить примеры, для которых эти эвристики находят решения со стоимостью далекой от оптимальной.

Theorem

Для любых $k \geq 2$, $n \geq 2k + 8$ *и* $\alpha > 1/n$ *существует пример* x *задачи коммивояжера с* n *городами* $\{c_1, \ldots, c_n\}$, *такой, что эвристика* k-*opt для входа* x *с начальным туром* $\tau_0 = (c_1, \ldots, c_n)$ *находит тур стоимость, которого* $m_k(x)$ *удовлетворяет неравенству* $m_k(x) > \alpha m^*(x)$.

- На практике эвристики локального поиска фиксированной глубины для задачи коммивояжера весьма эффективны.
- Однако можно построить примеры, для которых эти эвристики находят решения со стоимостью далекой от оптимальной.
- Как уже говорилось, результат, полученный на выходе эвристики, сильно зависит от выбора начального тура τ_0.

Theorem

Для любых $k \geq 2$, $n \geq 2k + 8$ и $\alpha > 1/n$ существует пример x задачи коммивояжера с n городами $\{c_1, \ldots, c_n\}$, такой, что эвристика k-opt для входа x с начальным туром $\tau_0 = (c_1, \ldots, c_n)$ находит тур стоимость, которого $m_k(x)$ удовлетворяет неравенству $m_k(x) > \alpha m^*(x)$.

- На практике эвристики локального поиска фиксированной глубины для задачи коммивояжера весьма эффективны.
- Однако можно построить примеры, для которых эти эвристики находят решения со стоимостью далекой от оптимальной.
- Как уже говорилось, результат, полученный на выходе эвристики, сильно зависит от выбора начального тура τ_0.
- Следующая теорема показывает, что существуют такие начальные туры, при использовании которых эвристика дает решение хуже оптимального в произвольное число раз.

Theorem

Для любых $k \geq 2$, $n \geq 2k + 8$ и $\alpha > 1/n$ существует пример x задачи коммивояжера с n городами $\{c_1, \ldots, c_n\}$, такой, что эвристика k-opt для входа x с начальным туром $\tau_0 = (c_1, \ldots, c_n)$ находит тур стоимость, которого $m_k(x)$ удовлетворяет неравенству $m_k(x) > \alpha m^*(x)$.

Локальный поиск для задачи о разбиении графа

- В задаче о разбиении графа задан взвешенный неориентированный граф $G = (V, E)$ с четным числом вершин и весами ребер $w(e),\ e \in E$.

- В задаче о разбиении графа задан взвешенный неориентированный граф $G = (V, E)$ с четным числом вершин и весами ребер $w(e)$, $e \in E$.
- Далее под *разбиением множества вершин* V всегда будем понимать разбиение на два множества (A, B) таких, что $|A| = |B| = |V|/2$.

- В задаче о разбиении графа задан взвешенный неориентированный граф $G = (V, E)$ с четным числом вершин и весами ребер $w(e)$, $e \in E$.
- Далее под *разбиением множества вершин* V всегда будем понимать разбиение на два множества (A, B) таких, что $|A| = |B| = |V|/2$.
- Задача о разбиении графа заключается в нахождении разбиения (A, B) множества вершин V с минимальной стоимостью $c(A, B)$, которая по определению равна сумме весов всех ребер между A и B.

- В задаче о разбиении графа задан взвешенный неориентированный граф $G = (V, E)$ с четным числом вершин и весами ребер $w(e)$, $e \in E$.
- Далее под *разбиением множества вершин* V всегда будем понимать разбиение на два множества (A, B) таких, что $|A| = |B| = |V|/2$.
- Задача о разбиении графа заключается в нахождении разбиения (A, B) множества вершин V с минимальной стоимостью $c(A, B)$, которая по определению равна сумме весов всех ребер между A и B.
- Самой известной эвристикой для задачи о разбиении графа является алгоритм локального поиска Кернигана—Лина.

- В задаче о разбиении графа задан взвешенный неориентированный граф $G = (V, E)$ с четным числом вершин и весами ребер $w(e)$, $e \in E$.
- Далее под *разбиением множества вершин* V всегда будем понимать разбиение на два множества (A, B) таких, что $|A| = |B| = |V|/2$.
- Задача о разбиении графа заключается в нахождении разбиения (A, B) множества вершин V с минимальной стоимостью $c(A, B)$, которая по определению равна сумме весов всех ребер между A и B.
- Самой известной эвристикой для задачи о разбиении графа является алгоритм локального поиска Кернигана—Лина.
- Этот алгоритм начинает свою работу со случайного разбиения множества вершин V.

- В задаче о разбиении графа задан взвешенный неориентированный граф $G = (V, E)$ с четным числом вершин и весами ребер $w(e)$, $e \in E$.
- Далее под *разбиением множества вершин* V всегда будем понимать разбиение на два множества (A, B) таких, что $|A| = |B| = |V|/2$.
- Задача о разбиении графа заключается в нахождении разбиения (A, B) множества вершин V с минимальной стоимостью $c(A, B)$, которая по определению равна сумме весов всех ребер между A и B.
- Самой известной эвристикой для задачи о разбиении графа является алгоритм локального поиска Кернигана—Лина.
- Этот алгоритм начинает свою работу со случайного разбиения множества вершин V.
- Начиная с текущего разбиения (A_0, B_0), жадным способом строится последовательность разбиений $(A_1, B_1), \ldots, (A_l, B_l)$.

- Каждое разбиение (A_k, B_k), $1 \leq k \leq l$, в этой последовательности получается из предыдущего — (A_{k-1}, B_{k-1}) обменом одной вершины из A_{k-1} на одну вершину из B_{k-1}.

- Каждое разбиение (A_k, B_k), $1 \leq k \leq l$, в этой последовательности получается из предыдущего — (A_{k-1}, B_{k-1}) обменом одной вершины из A_{k-1} на одну вершину из B_{k-1}.
- Стоимость каждого разбиения из такой последовательности меньше, чем стоимость текущего разбиения. Локальный поиск осуществляется среди разбиений этой последовательности.

- Каждое разбиение (A_k, B_k), $1 \le k \le l$, в этой последовательности получается из предыдущего — (A_{k-1}, B_{k-1}) обменом одной вершины из A_{k-1} на одну вершину из B_{k-1}.
- Стоимость каждого разбиения из такой последовательности меньше, чем стоимость текущего разбиения. Локальный поиск осуществляется среди разбиений этой последовательности.
- При этом выбирается разбиение с минимальной стоимостью и им заменяется текущее разбиение.

- Каждое разбиение (A_k, B_k), $1 \leq k \leq l$, в этой последовательности получается из предыдущего — (A_{k-1}, B_{k-1}) обменом одной вершины из A_{k-1} на одну вершину из B_{k-1}.
- Стоимость каждого разбиения из такой последовательности меньше, чем стоимость текущего разбиения. Локальный поиск осуществляется среди разбиений этой последовательности.
- При этом выбирается разбиение с минимальной стоимостью и им заменяется текущее разбиение.
- Алгоритм останавливается, когда для текущего разбиения последовательность оказывается пустой.

Алгоритм Локальный поиск для ЗРГ

ВХОД: реберно-взвешенный граф $G = (V, E)$, $|V| = n$, разбиение A, B множества вершин V.

ВЫХОД: Локально-оптимальное разбиение A, B множества вершин V.

1. Положить $A_0 = A$ и $B_0 = B$, вычислить стоимость $c(A_0, B_0)$. Положить $i = 0$, $g_i = 0$, и $G(i) = 0$, где g_i — приращение при транспозиции; $G(i)$ — суммарный прирост при нескольких последовательных транспозициях.

Алгоритм Локальный поиск для ЗРГ

ВХОД: реберно-взвешенный граф $G = (V, E)$, $|V| = n$, разбиение A, B множества вершин V.
ВЫХОД: Локально-оптимальное разбиение A, B множества вершин V.

1. Положить $A_0 = A$ и $B_0 = B$, вычислить стоимость $c(A_0, B_0)$. Положить $i = 0$, $g_i = 0$, и $G(i) = 0$, где g_i — приращение при транспозиции; $G(i)$ — суммарный прирост при нескольких последовательных транспозициях.
2. Положить $i = 1$. Выбрать такую пару вершин $a_1 \in A_0$ и $b_1 \in B_0$, что при их транспозиции мы получаем разбиение A_1, B_1 с положительным приростом стоимости g_1, т.е. $g_1 = c(A_0, B_0) - c(A_1, B_1) > 0$. Если таких пар не существует, то перейти на 7, иначе положить $G(1) = g_1$.

Алгоритм Локальный поиск для ЗРГ

ВХОД: реберно-взвешенный граф $G = (V, E)$, $|V| = n$, разбиение A, B множества вершин V.
ВЫХОД: Локально-оптимальное разбиение A, B множества вершин V.

1. Положить $A_0 = A$ и $B_0 = B$, вычислить стоимость $c(A_0, B_0)$. Положить $i = 0$, $g_i = 0$, и $G(i) = 0$, где g_i — приращение при транспозиции; $G(i)$ — суммарный прирост при нескольких последовательных транспозициях.
2. Положить $i = 1$. Выбрать такую пару вершин $a_1 \in A_0$ и $b_1 \in B_0$, что при их транспозиции мы получаем разбиение A_1, B_1 с положительным приростом стоимости g_1, т.е. $g_1 = c(A_0, B_0) - c(A_1, B_1) > 0$. Если таких пар не существует, то перейти на 7, иначе положить $G(1) = g_1$.
3. Положить $i = i + 1$. Для каждой пары вершин, не входящих в уже выбранные пары вершин, оценить прирост при их транспозиции. Выбрать пару вершин a_i, b_i, где $a_i \in A_{i-1}$ и $b_i \in B_{i-1}$, такую, что при их транспозиции мы получаем разбиение A_i, B_i с максимальным приростом стоимости — $g_i = c(A_{i-1}, B_{i-1}) - c(A_i, B_i)$.

4. Вычислить суммарный прирост: $G(i) = \sum_{k=1}^{i} g_k$. Если $i < n/2$ и $G(i) > 0$, то перейти на 3.

4. Вычислить суммарный прирост: $G(i) = \sum_{k=1}^{i} g_k$. Если $i < n/2$ и $G(i) > 0$, то перейти на 3.
5. Выбрать k, такое, что $0 \leq k \leq i$ и $G(k)$ максимально.

4. Вычислить суммарный прирост: $G(i) = \sum_{k=1}^{i} g_k$. Если $i < n/2$ и $G(i) > 0$, то перейти на 3.

5. Выбрать k, такое, что $0 \le k \le i$ и $G(k)$ максимально.

6. Если $k > 0$, то положить $A_0 = A_k$, $B_0 = B_k$ и перейти на 2.

4. Вычислить суммарный прирост: $G(i) = \sum_{k=1}^{i} g_k$. Если $i < n/2$ и $G(i) > 0$, то перейти на 3.
5. Выбрать k, такое, что $0 \le k \le i$ и $G(k)$ максимально.
6. Если $k > 0$, то положить $A_0 = A_k$, $B_0 = B_k$ и перейти на 2.
7. В решаемой задаче достигнут локальный оптимум. Положить $A = A_0$ и $B = B_0$. Выйти с решением A, B стоимости $c(A, B)$.

- *Структура окрестностей Кернигана–Лина* — это структура окрестностей, соответствующая этому алгоритму локального поиска.

- *Структура окрестностей Кернигана–Лина* — это структура окрестностей, соответствующая этому алгоритму локального поиска.
- Она определяется следующим образом. Разбиение (A', B') будем называть *s-разбиением для разбиения* (A, B), если (A', B') можно получить из (A, B) обменом одного элемента A на один элемент B.

- *Структура окрестностей Кернигана–Лина* — это структура окрестностей, соответствующая этому алгоритму локального поиска.
- Она определяется следующим образом. Разбиение (A', B') будем называть *s-разбиением для разбиения* (A, B), если (A', B') можно получить из (A, B) обменом одного элемента A на один элемент B.
- Назовем (A', B') *жадным s-разбиением*, если $c(A, B) - c(A', B')$ достигает максимума среди всех s-разбиений для разбиения (A, B).

- *Структура окрестностей Кернигана–Лина* — это структура окрестностей, соответствующая этому алгоритму локального поиска.
- Она определяется следующим образом. Разбиение (A', B') будем называть *s-разбиением для разбиения* (A, B), если (A', B') можно получить из (A, B) обменом одного элемента A на один элемент B.
- Назовем (A', B') *жадным s-разбиением*, если $c(A, B) - c(A', B')$ достигает максимума среди всех s-разбиений для разбиения (A, B).
- Если, кроме того, (A', B') является лексикографически наименьшим среди всех жадных s-разбиений, то будем говорить, что (A', B') — лексикографически жадное s-разбиение для (A, B).

- *Структура окрестностей Кернигана–Лина* — это структура окрестностей, соответствующая этому алгоритму локального поиска.
- Она определяется следующим образом. Разбиение (A', B') будем называть *s-разбиением для разбиения* (A, B), если (A', B') можно получить из (A, B) обменом одного элемента A на один элемент B.
- Назовем (A', B') *жадным s-разбиением*, если $c(A, B) - c(A', B')$ достигает максимума среди всех s-разбиений для разбиения (A, B).
- Если, кроме того, (A', B') является лексикографически наименьшим среди всех жадных s-разбиений, то будем говорить, что (A', B') — лексикографически жадное s-разбиение для (A, B).
- Пусть (A_i, B_i) — последовательность разбиений, каждое из которых, кроме начального (A_0, B_0), является s-разбиением для предшествующего ему.

- Эту последовательность будем называть монотонной, если разности $A_i - A_0$ и $B_i - B_0$ монотонно возрастают (т.е. ни одна вершина не переносится обратно в свое первоначальное множество).

- Эту последовательность будем называть монотонной, если разности $A_i - A_0$ и $B_i - B_0$ монотонно возрастают (т.е. ни одна вершина не переносится обратно в свое первоначальное множество).
- Наконец, мы будем говорить, что разбиение (A', B') принадлежит окрестности разбиения (A, B), если оно появляется в максимальной монотонной последовательности с начальным разбиением (A, B) лексикографически жадных s-разбиений, однозначно выбираемых эвристикой Кернигана–Лина.

- Эту последовательность будем называть монотонной, если разности $A_i - A_0$ и $B_i - B_0$ монотонно возрастают (т.е. ни одна вершина не переносится обратно в свое первоначальное множество).
- Наконец, мы будем говорить, что разбиение (A', B') принадлежит окрестности разбиения (A, B), если оно появляется в максимальной монотонной последовательности с начальным разбиением (A, B) лексикографически жадных s-разбиений, однозначно выбираемых эвристикой Кернигана–Лина.
- Отметим, что такая последовательность состоит из $|V|/2 + 1$ разбиений, причем последнее равно (B, A).

- Эту последовательность будем называть монотонной, если разности $A_i - A_0$ и $B_i - B_0$ монотонно возрастают (т.е. ни одна вершина не переносится обратно в свое первоначальное множество).
- Наконец, мы будем говорить, что разбиение (A', B') принадлежит окрестности разбиения (A, B), если оно появляется в максимальной монотонной последовательности с начальным разбиением (A, B) лексикографически жадных s-разбиений, однозначно выбираемых эвристикой Кернигана–Лина.
- Отметим, что такая последовательность состоит из $|V|/2 + 1$ разбиений, причем последнее равно (B, A).
- Таким образом, каждое разбиение имеет окрестность, состоящую из $|V|/2$ разбиений.

Локальный поиск переменной глубины

- Обменный алгоритм фиксированной глубины, такой, как k-opt, при переходе от текущего решения к следующему, выполняет последовательно серию локальных обменов.

- Обменный алгоритм фиксированной глубины, такой как k-opt, при переходе от текущего решения к следующему, выполняет последовательно серию локальных обменов.
- Длина этой последовательности ограничена (не более k).

- Обменный алгоритм фиксированной глубины, такой, как k-opt, при переходе от текущего решения к следующему, выполняет последовательно серию локальных обменов.
- Длина этой последовательности ограничена (не более k).
- При локальном поиске переменной глубины длина такой последовательности априори не ограничивается.

- Обменный алгоритм фиксированной глубины, такой как k-opt, при переходе от текущего решения к следующему, выполняет последовательно серию локальных обменов.
- Длина этой последовательности ограничена (не более k).
- При локальном поиске переменной глубины длина такой последовательности априори не ограничивается.
- В обменных алгоритмах переменной глубины сначала к текущему решению s применяется последовательность из t обменов (t зависит от s и специфики эвристики), в результате получается последовательность решений $s_1, \dots, s_t$, где $s_i \in \mathcal{N}^i(s)$. Затем в качестве нового текущего выбирается лучшее среди этих решений.

Локальный поиск переменной глубины

Input: Пример x. *Output:* Решение s.

begin

$s :=$ начальное допустимое решение s_0;

repeat

$X := \varepsilon$ (пустую последовательность);

$s' := s$;

(* начало процедуры выбора следующего текущего решения *);

Пусть C — множество обменных операций, возможных на s';

while правило остановки не выполняется **do**

begin

Выбрать $\tau \in C$, такое, что $\mu(x) = \max_{c \in C} \mu(c)$;

Добавить x в конец X;

$C := C - \{\tau\}$;

end;

Выбрать префиксную подпоследовательность X' из X;

Применить X' к s, чтобы получить новое решение s';

(* окончание процедуры выбора следующего текущего решения *);

if $m(x, s') < m(x, s)$ **then** $s := s'$

until $s \neq s'$;

return s

end.

Для такого выбора лучшего решения используется мера μ прироста, которая вычисляется для любой последовательности обменов, начинающейся от текущего решения.
Чтобы этот метод был эффективным, должны выполняться несколько условий:

1. Функция μ должна быть пригодной для вычисления изменения стоимости (приращения), которое получается после применения любой последовательности обменов. Такая функция должна быть аддитивной на последовательности, может принимать отрицательные значения, и должна быть такой, что улучшение ее значения указывает вероятное направление к оптимальному решению.

Для такого выбора лучшего решения используется мера μ прироста, которая вычисляется для любой последовательности обменов, начинающейся от текущего решения.
Чтобы этот метод был эффективным, должны выполняться несколько условий:

1. Функция μ должна быть пригодной для вычисления изменения стоимости (приращения), которое получается после применения любой последовательности обменов. Такая функция должна быть аддитивной на последовательности, может принимать отрицательные значения, и должна быть такой, что улучшение ее значения указывает вероятное направление к оптимальному решению.
2. Операция замены должна быть такой, что при последовательности любой длины, полученной из текущего решения, она дает новое допустимое решение.

Для такого выбора лучшего решения используется мера μ прироста, которая вычисляется для любой последовательности обменов, начинающейся от текущего решения.
Чтобы этот метод был эффективным, должны выполняться несколько условий:

1. Функция μ должна быть пригодной для вычисления изменения стоимости (приращения), которое получается после применения любой последовательности обменов. Такая функция должна быть аддитивной на последовательности, может принимать отрицательные значения, и должна быть такой, что улучшение ее значения указывает вероятное направление к оптимальному решению.
2. Операция замены должна быть такой, что при последовательности любой длины, полученной из текущего решения, она дает новое допустимое решение.
3. Стоп правило должно исключать генерацию слишком длинных бесполезных и слишком коротких не качественных последовательностей.

- На основе окрестности 2-opt Линoм и Керниганом предложена очень эффективная эвристика переменной глубины для задачи коммивояжера.

- На основе окрестности 2-opt Линoм и Керниганом предложена очень эффективная эвристика переменной глубины для задачи коммивояжера.
- Она позволяет заменять произвольное число ребер и переходит от одного тура к другому, используя принципы жадных алгоритмов.

- На основе окрестности 2-opt Линoм и Керниганом предложена очень эффективная эвристика переменной глубины для задачи коммивояжера.
- Она позволяет заменять произвольное число ребер и переходит от одного тура к другому, используя принципы жадных алгоритмов.
- Основная идея эвристики заключается в следующем.

- На основе окрестности 2-opt Линoм и Керниганом предложена очень эффективная эвристика переменной глубины для задачи коммивояжера.
- Она позволяет заменять произвольное число ребер и переходит от одного тура к другому, используя принципы жадных алгоритмов.
- Основная идея эвристики заключается в следующем.
- Удалим из гамильтонового цикла произвольное ребро, скажем (a, b).

- На основе окрестности 2-opt Линoм и Керниганом предложена очень эффективная эвристика переменной глубины для задачи коммивояжера.
- Она позволяет заменять произвольное число ребер и переходит от одного тура к другому, используя принципы жадных алгоритмов.
- Основная идея эвристики заключается в следующем.
- Удалим из гамильтонового цикла произвольное ребро, скажем (a, b).
- В полученном пути один конец (вершину a) будем считать фиксированной, а другой конец будем менять, перестраивая гамильтонов путь.

- На основе окрестности 2-opt Линон и Керниганом предложена очень эффективная эвристика переменной глубины для задачи коммивояжера.
- Она позволяет заменять произвольное число ребер и переходит от одного тура к другому, используя принципы жадных алгоритмов.
- Основная идея эвристики заключается в следующем.
- Удалим из гамильтонового цикла произвольное ребро, скажем (a, b).
- В полученном пути один конец (вершину a) будем считать фиксированной, а другой конец будем менять, перестраивая гамильтонов путь.
- Добавим ребро из вершины b, например (b, c), и разорвем образовавшийся единственный цикл так, чтобы снова получить гамильтонов путь.

- На основе окрестности 2-opt Линoм и Керниганом предложена очень эффективная эвристика переменной глубины для задачи коммивояжера.
- Она позволяет заменять произвольное число ребер и переходит от одного тура к другому, используя принципы жадных алгоритмов.
- Основная идея эвристики заключается в следующем.
- Удалим из гамильтонового цикла произвольное ребро, скажем (a, b).
- В полученном пути один конец (вершину a) будем считать фиксированной, а другой конец будем менять, перестраивая гамильтонов путь.
- Добавим ребро из вершины b, например (b, c), и разорвем образовавшийся единственный цикл так, чтобы снова получить гамильтонов путь.
- Для этого придется удалить ребро, инцидентное вершине c. Обозначим его (c, d).

- На основе окрестности 2-opt Линомом и Керниганом предложена очень эффективная эвристика переменной глубины для задачи коммивояжера.
- Она позволяет заменять произвольное число ребер и переходит от одного тура к другому, используя принципы жадных алгоритмов.
- Основная идея эвристики заключается в следующем.
- Удалим из гамильтонового цикла произвольное ребро, скажем (a, b).
- В полученном пути один конец (вершину a) будем считать фиксированной, а другой конец будем менять, перестраивая гамильтонов путь.
- Добавим ребро из вершины b, например (b, c), и разорвем образовавшийся единственный цикл так, чтобы снова получить гамильтонов путь.
- Для этого придется удалить ребро, инцидентное вершине c. Обозначим его (c, d).
- Новый гамильтонов путь имеет концевые вершины a и d.

- Эту процедуру будем называть ротацией. Для получения нового гамильтонова цикла достаточно добавить ребро (a, d).

- Эту процедуру будем называть ротацией. Для получения нового гамильтонова цикла достаточно добавить ребро (a, d).
- Согласно алгоритму Лина–Кернигана переход от одного тура к другому состоит из удаления некоторого ребра, выполнения серии последовательных ротаций и, наконец, замыкания концевых вершин полученного гамильтонова пути.

- Эту процедуру будем называть ротацией. Для получения нового гамильтонова цикла достаточно добавить ребро (a, d).
- Согласно алгоритму Лина–Кернигана переход от одного тура к другому состоит из удаления некоторого ребра, выполнения серии последовательных ротаций и, наконец, замыкания концевых вершин полученного гамильтонова пути.
- Существуют различные варианты этой основной схемы, которые отличаются правилами выбора ротаций и ограничениями на множества удаляемых и добавляемых ребер.

- Эту процедуру будем называть ротацией. Для получения нового гамильтонова цикла достаточно добавить ребро (a, d).
- Согласно алгоритму Лина–Кернигана переход от одного тура к другому состоит из удаления некоторого ребра, выполнения серии последовательных ротаций и, наконец, замыкания концевых вершин полученного гамильтонова пути.
- Существуют различные варианты этой основной схемы, которые отличаются правилами выбора ротаций и ограничениями на множества удаляемых и добавляемых ребер.
- В алгоритме Лина-Кернигхана ротации выбираются так, чтобы минимизировать разность $w_{bc} - w_{cd}$.

- Эту процедуру будем называть ротацией. Для получения нового гамильтонова цикла достаточно добавить ребро (a, d).
- Согласно алгоритму Лина–Кернигана переход от одного тура к другому состоит из удаления некоторого ребра, выполнения серии последовательных ротаций и, наконец, замыкания концевых вершин полученного гамильтонова пути.
- Существуют различные варианты этой основной схемы, которые отличаются правилами выбора ротаций и ограничениями на множества удаляемых и добавляемых ребер.
- В алгоритме Лина-Кернигхана ротации выбираются так, чтобы минимизировать разность $w_{bc} - w_{cd}$.
- При этом мощность множества удаляемых и добавляемых ребер в серии ротаций не превысит n^2 и трудоемкость одного шага (перехода от одного тура к другому) останется полиномиальной.

- Эту процедуру будем называть ротацией. Для получения нового гамильтонова цикла достаточно добавить ребро (a, d).
- Согласно алгоритму Лина–Кернигана переход от одного тура к другому состоит из удаления некоторого ребра, выполнения серии последовательных ротаций и, наконец, замыкания концевых вершин полученного гамильтонова пути.
- Существуют различные варианты этой основной схемы, которые отличаются правилами выбора ротаций и ограничениями на множества удаляемых и добавляемых ребер.
- В алгоритме Лина-Кернигхана ротации выбираются так, чтобы минимизировать разность $w_{bc} - w_{cd}$.
- При этом мощность множества удаляемых и добавляемых ребер в серии ротаций не превысит n^2 и трудоемкость одного шага (перехода от одного тура к другому) останется полиномиальной.
- Общее число шагов алгоритма, по-видимому, не может быть ограничено полиномом.