Дискретная оптимизация

МФТИ, осень 2015

Александр Дайняк

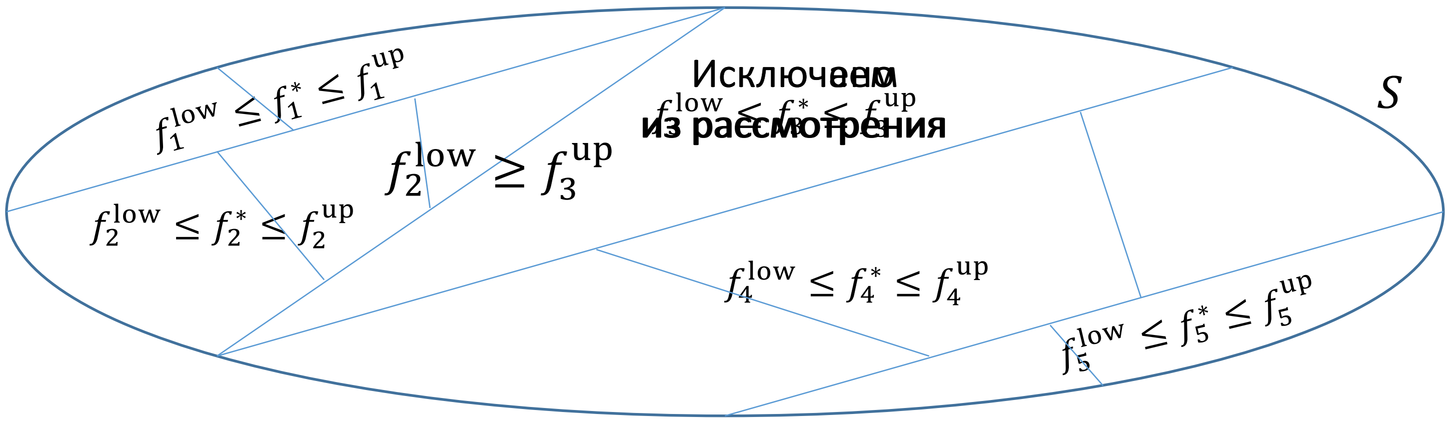
www.dainiak.com

Метаэвристики

- Метод ветвлений и ограничений (ветвей и границ, branch and bound)
- Алгоритмы «по мотивам биологических явлений»:
  - Генетические алгоритмы (genetic algorithms)
  - Алгоритмы «муравьиных колоний» (ant colony algorithms)

Метод ветвей и границ

- Максимизируем функцию f на множестве S, просматривать всё S накладно.
- Разбиваем множество S на части, ищем в каждой части верхнюю и нижнюю оценку для f_i^* — максимальное значение f на i-й подобласти.
- Если оказывается, что $\exists i, j \,:\, f_i^{\text{up}} \leq f_j^{\text{low}}$, то больше не рассматриваем j-ю область.
- Рекурсивно производим ту же процедуру на подобластях.



Метод ветвей и границ

- Метод ветвей и границ эффективен, если
  - Хорошие оценки минимума функции f на подобластях легко вычислить (например, локальным поиском или другими эвристиками)
  - Можно эффективно производить разбиение множества S на области примерно равного размера (в этом случае метод можно эффективно распараллелить)

Пример применения метода в задаче коммивояжёра

- S — множество всех ГЦ в графе G
- Разбиение множества S на подмножества:
  - $S = S_e \sqcup S_{\bar{e}}$, где $S_e = \{h \in S \mid e \in H\}$
  - Фиксируем $\mathrm{E}' \subset E(G)$ и раскладываем $S = \bigsqcup_{A \in E'} S_A$, где $S_A = \{h \in S \mid A \subseteq H\}$
- Нижние оценки веса минимального ГЦ:
  - Вес минимального остовного дерева
  - …
- Верхняя оценка веса минимального ГЦ — вес любого ГЦ.

Генетические (эволюционные) алгоритмы

- Идея — естественный отбор в ходе эволюции:
  - Имеется популяция особей, обитающих во враждебной среде
  - Особи скрещиваются, передавая потомкам часть своих генов
  - Наиболее приспособленные потомки выживают

Генетические (эволюционные) алгоритмы

- Минимизируем функцию f на множестве S
- Формализация генетического алгоритма:
  - Задаёмся функциями скрещивания C и мутации M

$$C: S \times S \to S, \qquad M: S \to S$$

1. Выбираем «начальную популяцию» $A \subseteq S,\ |A| = m$
2. Строим множество «потомков»: $D = \{C(s', s'') \mid s', s'' \in A\}$
   и множество «мутантов» $J = \{M(s) \mid s \in A\}$
3. Сортируем множество $A \cup D \cup J$ по возрастанию значений функции f и берём первые m элементов. Они составляют новую популяцию A.
   Переходим к шагу 2.

Генетические (эволюционные) алгоритмы

$$A_{\text{new}} = \text{best}_m(A_{\text{old}} \cup \{C(s', s'') \mid s', s'' \in A_{\text{old}}\} \cup \{M(s) \mid s \in A_{\text{old}}\})$$

- Когда можно останавливаться:
  - Значение функции f достаточно мало:
  $$\min_{s \in A_{\text{new}}} f(s) < \gamma$$
  - Новая популяция ненамного лучше старой:
  $$\min_{s \in A_{\text{new}}} f(s) \geq 0.99 \cdot \min_{s \in A_{\text{old}}} f(s)$$

Генетические (эволюционные) алгоритмы

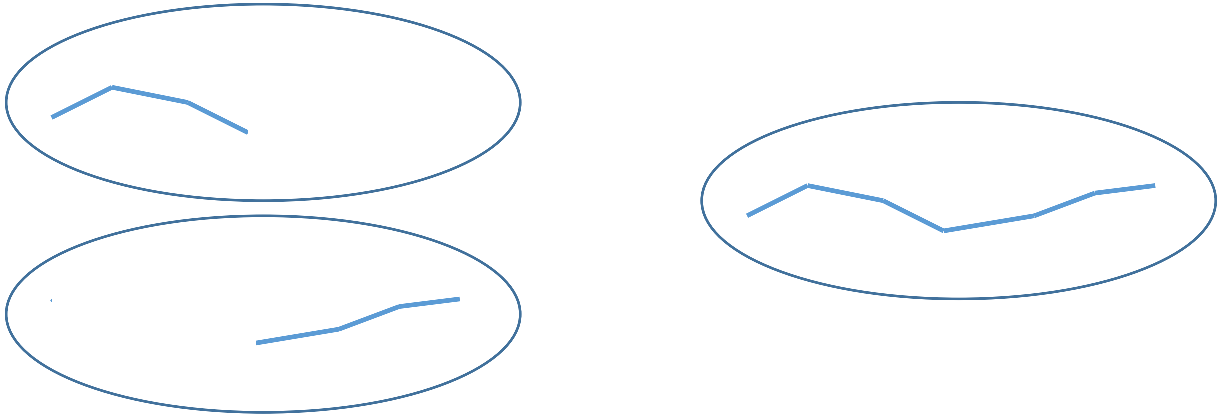
$$A_{\text{new}} = \text{best}_m(A_{\text{old}} \cup \{C(s', s'') \mid s', s'' \in A_{\text{old}}\} \cup \{M(s) \mid s \in A_{\text{old}}\})$$

- Выбор функции мутации:
  - Берём окрестностную функцию N из локального поиска и полагаем $M(s) = \text{random}\big(N(s)\big)$ или $M(s) = \text{best}\big(N(s)\big)$
- Выбор функции скрещивания:
  - Если $S \subset \mathbb{R}^n$, то можно взять $C(s', s'') = \frac{1}{2}(s' + s'')$ или $C\big((s_1', \dots, s_n'), (s_1'', \dots, s_n'')\big) = (s_1', \dots s_i', s_{i+1}'', \dots, s_n'')$
  - Обычно диктуется спецификой задачи: что такое «хорошие гены» особи
  - Функция скрещивания может быть от ≥ 3 переменных

Генетические алгоритмы

- Пример применения ГА в задаче поиска кратчайшего пути:
  - Мутация:

  - Скрещивание:



Генетические (эволюционные) алгоритмы

- Пример применения ГА в задаче коммивояжёра:
  - Функцию мутации строим по k-окрестности (удаление/добавление k рёбер)
  - Скрещивание: часть графа обходим по первому ГЦ, а оставшиеся вершины обходим в том порядке, в каком они лежат на втором ГЦ

Генетические (эволюционные) алгоритмы

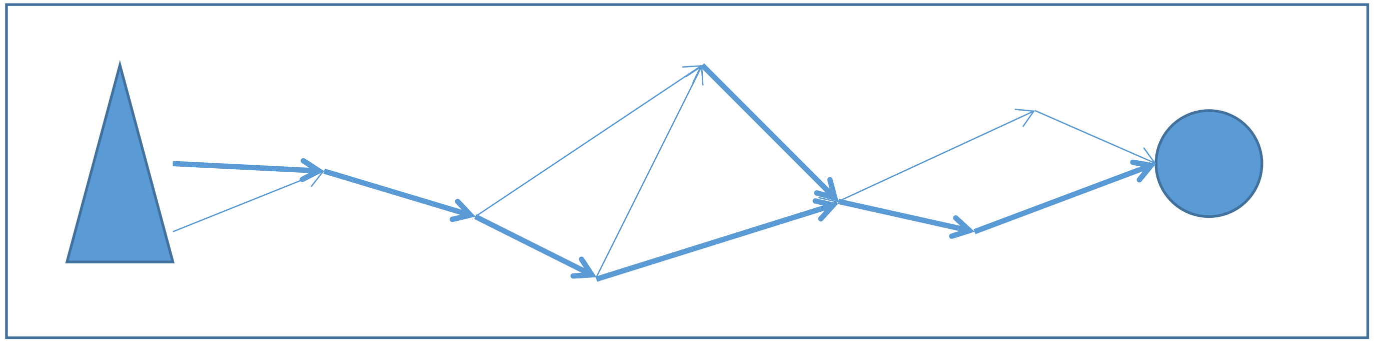
- Плюсы:
  - Простота и естественность подхода
  - Наличие большого числа управляемых параметров
  - Эффективность при удачной реализации
  - Возможность распараллеливания вычислений значений функций скрещивания и мутации
- Минусы:
  - Сложность формального анализа (как следствие, отсутствие гарантии результата)
  - Необходимость подбора параметров

Алгоритмы муравьиных колоний

- Общая идея:
  Колонии насекомых в природе при поиске пищи ориентируются по запахам.
  Насекомое, найдя «клад», возвращается в колонию, оставляя по пути запаховую метку.
  Отправляющиеся из колонии собратья с большей вероятностью идут туда, где «запах успеха» сильнее.

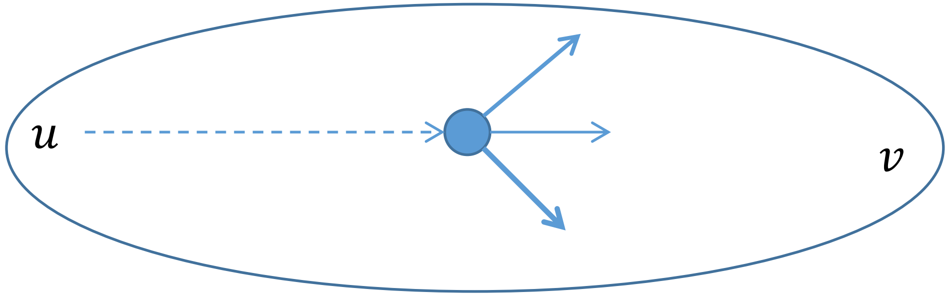
Алгоритмы муравьиных колоний

- Наилучшим образом алгоритмы м.к. подходят для задач, которые могут быть сведены к оптимальному обходу графов:



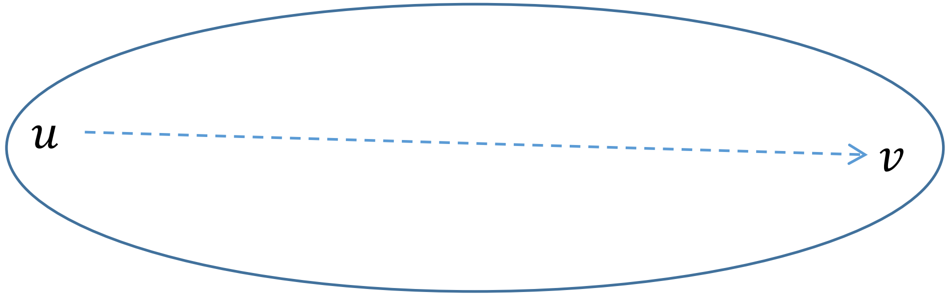
Алгоритмы муравьиных колоний

- «Муравьиный» алгоритм поиска кратчайшего пути в графе:
  - Дан граф G с весами на рёбрах, и пара вершин $u, v \in V(G)$
  - На каждом ребре e графа храним «запах» $\mathrm{pher}(e) \geq 0$
  - Из вершины u по графу отправляется агент, который, попав в очередную вершину, идёт в её соседа с вероятностью, пропорциональной $\mathrm{pher}(e)$ и обратно пропорциональной $w(e)$



Алгоритмы муравьиных колоний

- «Муравьиный» алгоритм поиска кратчайшего пути в графе:
  - Дан граф G с весами на рёбрах, и пара вершин $u, v \in V(G)$
  - На каждом ребре e графа храним «запах» $\mathrm{pher}(e) \geq 0$
  - Дойдя до v, агент увеличивает запах на рёбрах пройденного пути на величину, обратно пропорциональную длине пути



Алгоритмы муравьиных колоний

- Плюсы:
  - Одновременно можно запускать несколько агентов, что позволяет распараллелить поиск решения
  - Поведения агентов можно изменять в широких пределах
- Минусы:
  - Агенты обращаются к общей памяти (массив «запахов»)
  - Сложность формального анализа (как следствие, отсутствие гарантии результата)
  - Необходимость подбора параметров