discropt

20.04.2020

1 Эвристическое решение некоторых задач

Локальный поиск (обычный) двигается по маленьким шагам и обычно его перезапускаем и получаем разные качественно решения. Есть модификации, например когда застреваем в локальном оптимуме, идем в горку.

Вернемся к обычному локальному поиску. Мы берем маленькие окрестности для меньших вычислений. Поэтому, если мы не можем просмотреть всю окрестность, то мы решаем менее точную задачу, пытаясь искать вместо лучшей точки точку в ее окрестности. Если мы найдем такую точку, то запустим Кернигана-Лина или др. алгоритм. Т.е. теперь решаем приближенную задачу поиска.

Large neighbourhood search - парадигма, в которой мы находимся, как только начинаем использовать локальный поиск, но задачу перехода к новой точке решаем эвристически или умным перебором (отсечение, эвристики на продуктивность). Имеет достоинства локального поиска, вдобавок можно в любой момент остановить локальный поиск и взять промежуточный результат, т.к. он будет корректный, хоть и не точный. Алгоритм ближайшего соседа не умеет так, т.к. никакого гамильт. цикла не будет если он не сработает до конца.

Этот подход также применим к задаче коммивояжера.

Рюкзак

Попробуем применить такой подход к задаче о рюкзаке. Отсортируем по цена/вес:

$$\frac{p_1}{w_1} > \frac{p_2}{w_2} > ... > \frac{p_n}{w_n} \quad (1)$$

Тогда по этой стратегии сначало выбрали бы 1 предмет, а выкинули бы последний. Есть еще две стратегии:

$$p_1 > p_2 > ... > p_n \quad (2)$$

и

$$w_1 < w_2 < ... < w_n \quad (3)$$

Теперь используя каждую из этих стратегий можно решить задачу заполнения остатка рюкзака. Если это небольшое место, то псевдополиномиальный динамический алгоритм решит его. То есть эвристически выбираем что выложить, а задача наполнения рюкзака уже решается точно.

Задача о покрытии В терминах матрицы - выбираем как можно меньшн строк, чтобы в каждом столбце была хотя бы одна 1. Нужна стратегия удаления строк. Можно, например, по количеству покрываемых столбцов (удалить те, которые покрывают мало столбцов). Далее нужно покрытие для потерянных столбцов.

2 Генетические(эволюционные) алгоритмы

Идея - естественный отбор в ходе эволюции

- имеется популяция, обитающая во враждебной среде
- особи скрещиваются и передают часть генов потомкам
- выживают приспособленные

Формальная постановка задачи Пусть дана функция f на множестве S. Тогда мы должны минимзировать ее.
Функция скрещивания - $C : S \times S \to S$
Функция мутации - $M : S \to S$

1. выбор начальной популяции $A \subseteq S, |A| = m$
2. построение множества потомков: $D = \{C(s'.s'')|s', s'' \in A\}$ и множества мутантов $J = \{M(s)|s \in A\}$
3. Сортируем множество $A \cup D \cup J$ по возрастанию f и берем первые m элементов. Это состав новой популяции A. А Возвращаемся к шагу 2. $\mathrm{A}_{new} = (sorted(A \cup D \cup J))$ [:m]

Условия остановки:

- если f достаточно мала:
  $\min\limits_{s \in A_{new}} f(s) < \gamma$
- если новая популяция несильно лучше старой:
  $\min\limits_{s \in A_{new}} f(s) > 0.99 * \min\limits_{s \in A_{old}} f(s)$

Выбор мутации:

- Берем окрестностную функцию N из локального поиска и полагаем $M(s) = random(N(s))$ или $M(s) = best(N(s))$

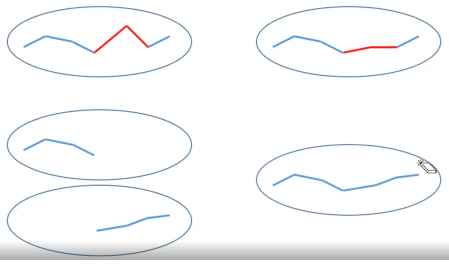
Выбор функции скрещивания:

- Если $S \subset R^n, C(s', s'') = \frac{1}{2}(s' + s'')$ или $\mathrm{C}((\mathrm{s}'_1, \ldots, s'_n), ((s''_1, \ldots, s''_n)) = (s'_1, \ldots, s'_i, s''_{i+1}, \ldots, s''_n)$
- Обычно дикуется спецфикой задачи: что такое хорошие гены

- Функция скрещивания может быть от 3х 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