Оглавление

0.1 Приближённые алгоритмы

0.1.1 Муравьиная колония

Продемонстрируем на задаче коммивояжёра:
На входе n городов
Определяем m – количество муравьёв
Каждый муравей будет строить свой маршрут, и будет оставлять в каждом городе какое-то количество ферромона. Остальные муравьи будут при выборе маршрута это учитывать
Есть несколько способ выбрать начальные города:

- "Рассадить" всех по разным городам
- "Посадить" всех в один город
- "Рассадить" группами

Так что, считаем что изначально каждый муравей находится в каком-то городе
$d(i,j)$ – длина ребра i,j
N_i^k – в i-м городе, k – номер муравья – непосещённые города

$$\eta(i,j) \coloneqq \frac{1}{d(i,j)}$$

$\tau(i,j)$ – количество ферромона на дуге i,j
$p(i,j)$ – вероятность перехода из i в j
Когда алгоритм закончится, получим m маршрутов, и выберем из них кратчайший
Естетственно, этот процесс повторяется несколько раз (возможно, с разными "рассадками" муравьёв)
Запишем формулу для вероятности:

$$p(i,j) = \frac{[\tau(i,j)]^\alpha \cdot [\eta(i,j)]^\beta}{\sum_{l \in N_i^k} [\tau(i,l)]^\alpha \cdot [\eta(i,l)]^\beta}$$

После этого, применяем схему Уолкера и выбираем город, куда дальше идти

- α, β – некоторые параметры, которые вводятся заранее (чаще всего, определяются экспериментально)
- $\tau(i,j)$ – количество ферромона
- $\eta(i,j)$ – функция "качества" ребра (зависит от его длины)
- В значенателе – все непосещённые варианты

> **Примечание.** На первом шаге, ферромона везде одинаково, а значит, вероятность определяется только длиной дуги

Как меняется ферромон?.

$$\tau(i,j) = \rho \cdot \tau(i,j) + \sum_{k=1}^{m} \triangle \tau(i,j)$$

$$\triangle \tau(i,j) = \begin{cases} 1/L_k, & (i,j) \in P_{L_k} \\ 0 \end{cases}$$

- ρ – некий понижающий коэффициент, определяется заранее, $\rho \in (0,1]$
- L_k – длина маршрута, уже пройденного k-м муравьём
- P_{L_k} – путь, уже пройденный k-м муравьём

Примечание. Ферромон со временем "испаряется":
Если k-й муравей не посетил ребро i,j, то $\tau(i,j) = \rho \cdot \tau(i,j)$

Пять пунктов, которые нужно придумать, чтобы применить муравьиный алгоритм:

1. $\eta(i,j)$ – функция "качества"
2. Определение ферромона
3. Правило обновления ферромона
4. Функция, определяющая качество всего решения (фитнес-функция)
   У нас это была длина маршрута
5. Как муравьи строят решение и определяют, что решение построено?

0.1.2 Локальный поиск

x_0 – начальное решение
$x \in U(x_0)$ – окрестность решения $f(x)$ – целевая функция

- Если $f(x) < f(x_0)$, то $x_0 \coloneqq x$
  Переходим в начало

0.1.3 Поиск с запретами

Возьмём самый простой вариант окрестности для задачи коммивояжёра:
Берём два случайных города из маршрута и меняем их местами
В таком случае можно зациклиться – поменять их обратно
Вводим список запретов. Его длина задаётся заранее (не очень много)
Кждое принятое решение записывается в этот список
Если место кончилось – выталкивается первая запись
Если на следующем шаге мы выбрали действие, которое есть в списке, то его не делаем (выбираем другое)

0.1.4 Имитация отжига

S_i – состояние (само решение)
T_i – температура
E – энергия
Нужно определить S_0 – начльное решение и T_{max}, T_{min}
f – функция изменения состояния

Алгоритм.

1. Вычислить $S_{new} = f(S_{i-1})$
2. - Если S_{new} лучше, то переходим в него
   - Если хуже, то переходим с вероятностью $P(\triangle E, T)$
3. Уменьшаем T

```
i := 1
T_0 := T_max
S_best := S_0
while T_i > T_min do
    S_new := f(S_{i-1})    // метод перехода S_{i+1} → S_new
    △ E = E(S_new) − E(S_{i-1})
    if p(△ E) ≥ random(0, 1) then
        S_i := S_new
    else
        S_i := S_{i-1}
    end
    if E(S_i) < E(S_best) then
        S_best := S_i
        ++i
    end
    T_{i+1} := f(T_i)
end
```

0.1.5 Иммунный алгоритм

1. Начальная популяция
2. Клонирование
3. Мутации
4. Афинность

Количество решений определяется качеством решения
Для афинности есть формула:

$$A(k) = \frac{LB}{1 + M(k) - LB}$$

- LB – Lower Bound – нижняя граница решения
- $M(k)$ – качество решения

Из клонов выбираем лучшего. Если он лучше отца, помещаем его в популяцию