巡回セールスマン問題の解法と理論

目次

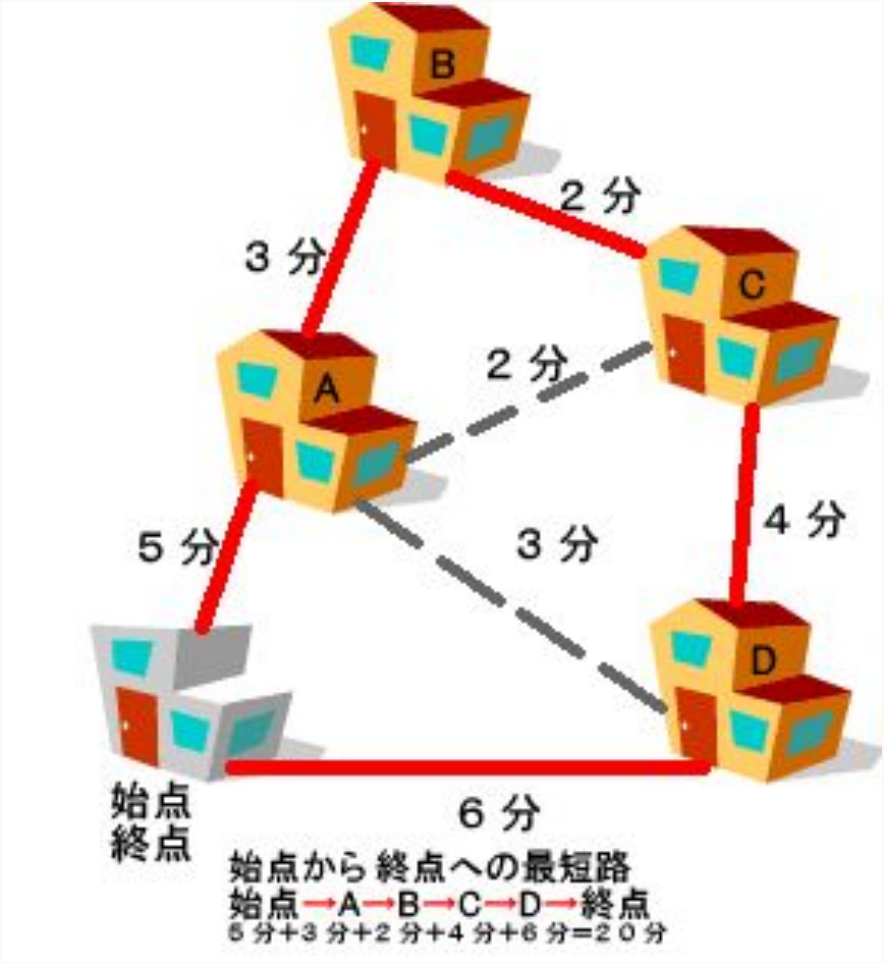
- 巡回セールスマン問題（TSP）への挑戦
- ヘルドカープのアルゴリズム
- 焼きなまし法
- 厳密解法の紹介

巡回セールスマン問題とは？

・複数の都市が与えられたとき、全ての都市を一度ずつ訪問して出発点に戻ってくる最短の経路を求める問題

・計算量が爆発的に増加してしまうことが知られている

・しかし解決できれば多様な分野で応用可能



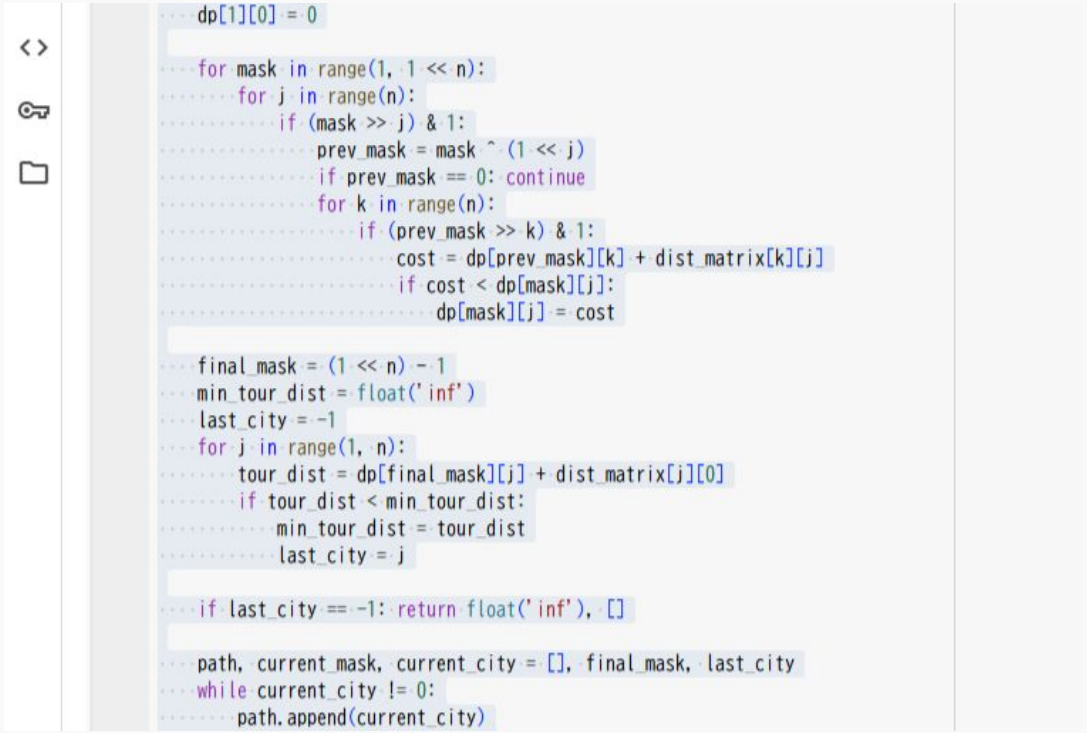
参考文献:
https://www.bunkyo.ac.jp/~nemoto/lecture/seminar2/2000/kimura/ronbun2.html

ヘルドカープのアルゴリズム

・動的計画法に基づく厳密解法
・C(S, j) :=「出発都市（例：都市 0）から、都市の部分集合 S をすべて経由し、最後に都市 j ∈ S に到達する 最短経路長」
・この部分問題は、以下の漸化式によって再帰的に解くことができる。C(S, j) = min k∈S\{j} {C(S \ {j}, k) + dkj}
ここで、dkj は都市 k から都市 j への距離を表す。この計算を、集合 S のサイズを 1 から N まで大きくしな がら進めることで、最終的な最適解を構築する。
・計算量は O(N² 2ⁿ) であり、全探索の O(N!) よりは遥かに 効率的だが、それでも N が 20 程度を超えると現実的な時間内での計算は困難となる。

一部コードの解説

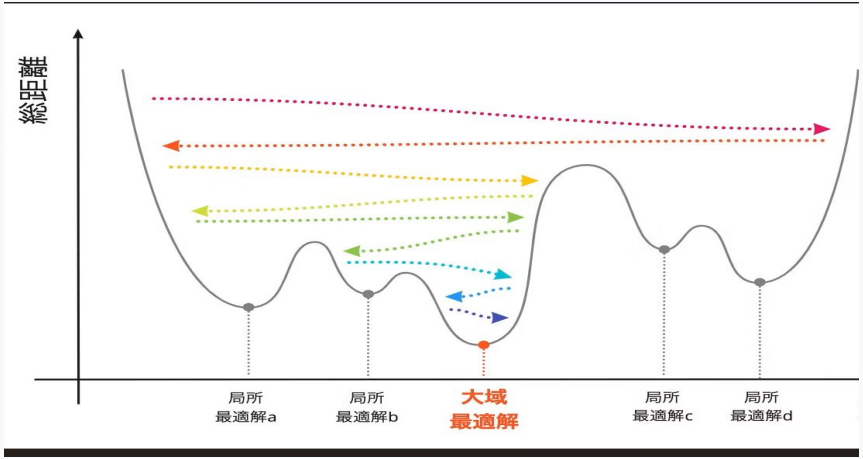
・アルゴリズムに関しては先ほど説明したとおりだが、実装方法について解説する。
・n個の頂点を回る必要がある場合、その頂点をすでに巡回したかのフラグ（つまり冪集合）
と最後に訪問した都市の情報が必要になる。
・冪集合ではi番目のbitが0ならばi番目の頂点は未訪問、1ならば訪問済みとすればよい
・スタートを頂点0とし、そのためdp[1][0] = 0としている。
・最後の頂点と出発点までをつなぎmin j (dp[2**n - 1][j] + 頂点jから0までの距離) が最短経路となる。



```
    dp[1][0] = 0

    for mask in range(1, 1 << n):
        for j in range(n):
            if (mask >> j) & 1:
                prev_mask = mask ^ (1 << j)
                if prev_mask == 0: continue
                for k in range(n):
                    if (prev_mask >> k) & 1:
                        cost = dp[prev_mask][k] + dist_matrix[k][j]
                        if cost < dp[mask][j]:
                            dp[mask][j] = cost

    final_mask = (1 << n) - 1
    min_tour_dist = float('inf')
    last_city = -1
    for j in range(1, n):
        tour_dist = dp[final_mask][j] + dist_matrix[j][0]
        if tour_dist < min_tour_dist:
            min_tour_dist = tour_dist
            last_city = j

    if last_city == -1: return float('inf'), []

    path, current_mask, current_city = [], final_mask, last_city
    while current_city != 0:
        path.append(current_city)
```

焼きなまし法の紹介

- 常に良い解を選ぶのではなく、「温度」パラメータに応じて
  確率的に悪い解も許容する
- 一般に離散的に状態空間を持つ組み合わせ最適化問題に対してのアプローチ
- 最初に温度を設定し、徐々に冷やしていく
- **探索初期（高温）**：解が悪化する移動も許容し、広域的に探索する
- **探索終盤（低温）**：次第に改善する移動のみを選択し、一つの解に収束させる
- この方法はマルコフ過程によってモデリングでき大域的最適解が保証できる！（一般にヒューリスティック解法は解の保証がないものが多い）
- 右の画像では温度と近傍の範囲が依存しているように見えるが、一般的には温度に依存しない



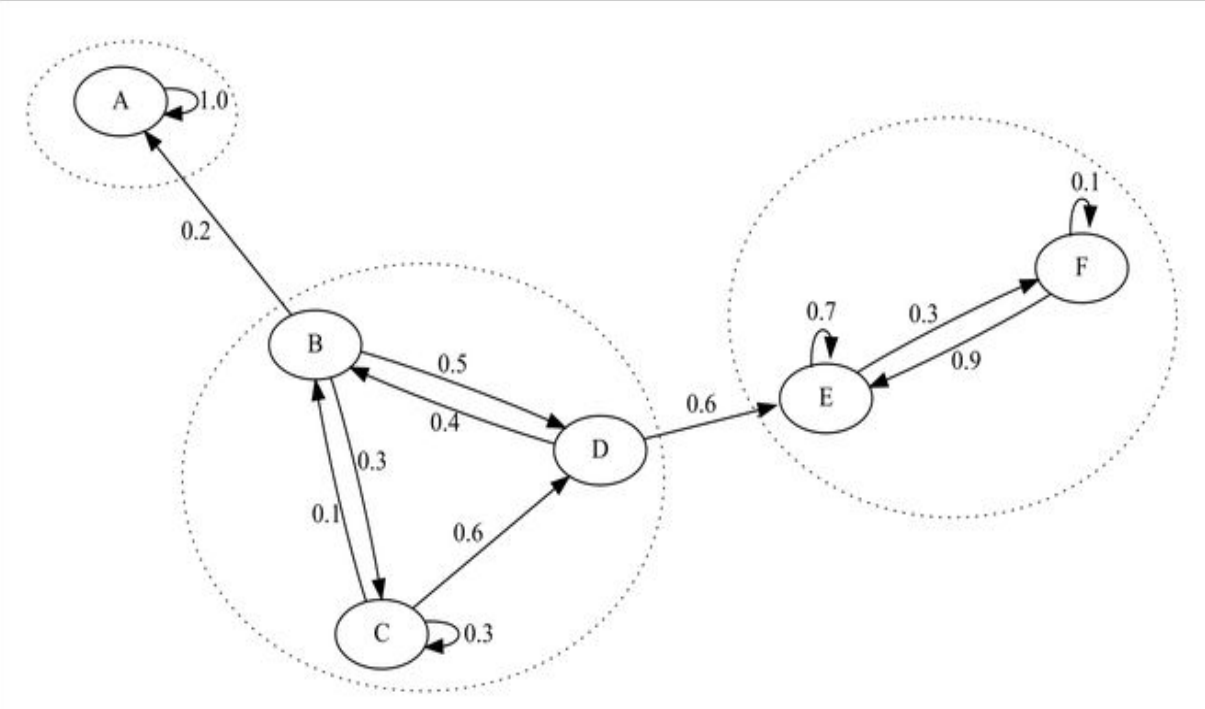
参考文献：https://qiita.com/take314/items/7eae18045e989d7eaf52

※準備（マルコフ連鎖とは）

・マルコフ過程のうち、状態が離散的（有限的）なものをマルコフ連鎖と呼ぶ。

・オートマトンと非常に似た構造を持っている！

・違いは遷移が確立によって起こるか、外部入力される文字によって起こるか



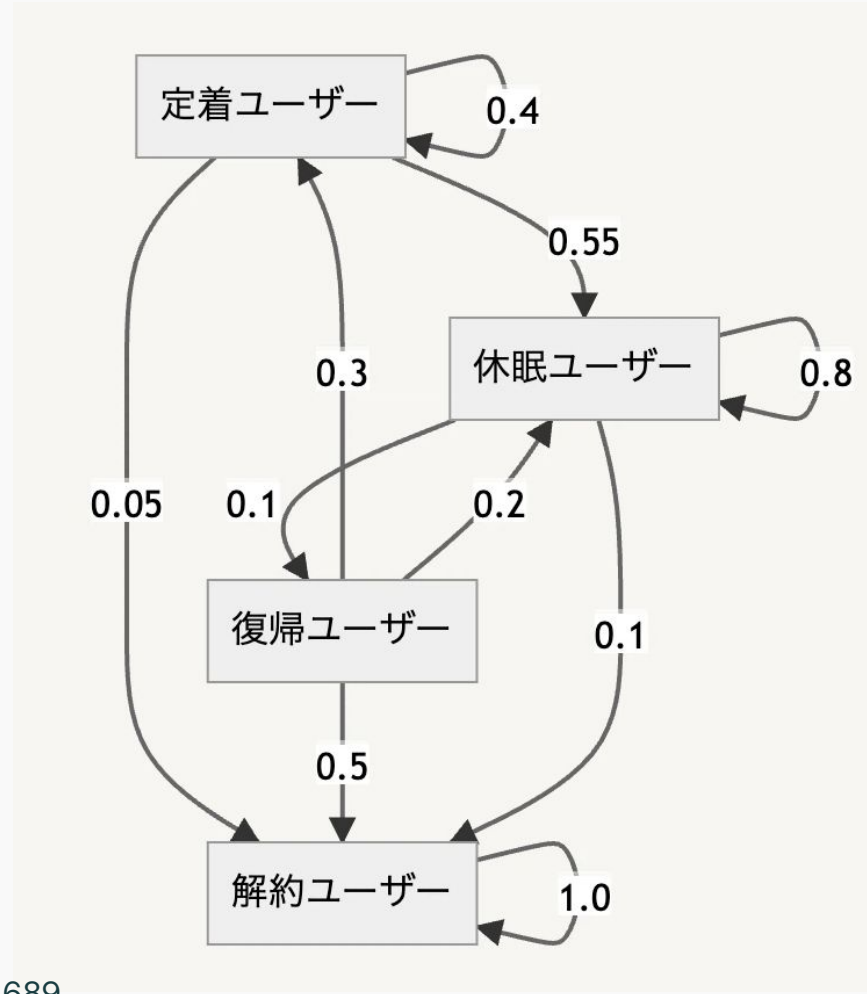
参考文献
:https://omedstu.jimdofree.com/2018/05/04/%E3%83%9E%E3%83%AB%E3%82%B3%E3%83%95%E9%80%A3%E9%8E%96-markov-chain/

マルコフ連鎖の分類

・斉時性：状態の推移確率が時間に拠らず一定
（焼きなまし法では実際の推移する確率でないことに注意）

・既約性：マルコフ連鎖の状態空間の任意の二状態が互いに到達可能。たとえば吸収状態集合などがあるとダメ（右の図では解約ユーザーの状態になってしまうと脱出できない）

・状態が互いに到達可能かは同値関係を満たす。そして状態集合はこの同値関係を使って直和分解できる。



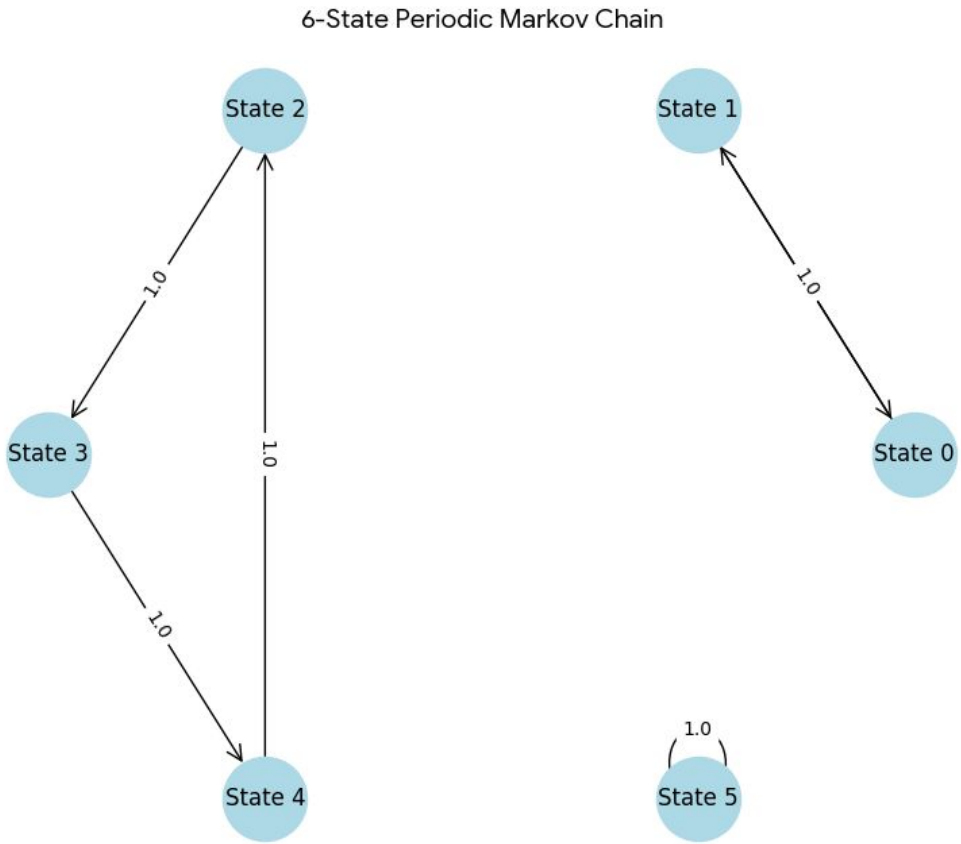
参考文献： https://note.com/dmaruyama/n/n1e144b6b8689

マルコフ連鎖の分類

・非周期性 : n回のステップで頂点iからjに到達できる確率をPij**(n)とする。
（到達不可なら0）
・状態iについて、
Ai = {n ∈ 自然数 | Pii**(n) > 0}とする。
任意のiについて、Aiの要素の最大公約数が1ならばマルコフ連鎖は非周期性を満たす

・つまり、元に戻ってくるステップ数がある周期に縛られない

・右は周期的なマルコフ連鎖の例



使用する定理

・命題 1.7.1. 次の条件 (1)–(3) は同値である.
(1) マルコフ連鎖はエルゴード的である.
(2) マルコフ連鎖は斉時性を満たし、既約で，すべての状態は非周期的である.
(3) マルコフ連鎖は斉時性を満たし、既約で，非周期的な状態が存在する.

・定理 1.7.2. マルコフ連鎖がエルゴード的であるとき，定常分布 $\pi = \{\pi i\}i \in S$ が一意に存 在し，次を満たす:
ある定数 $C > 0$ と $\gamma \in (0, 1)$, $n0 \in N$ が存在し， $|p(n) ij - \pi j| \leq C\gamma n$, $i, j \in S, n \geq n0$.
(1.7.2) 従って，このとき特に， $\lim n\to\infty p(n) ij = \pi j$, $i, j \in S$ が成り立つ.

・つまりマルコフ連鎖が有限、既約かつ斉時性、非周期性を満たすならば定常分布は唯一に定まる！（後で出てくる）

・エルゴード性は定常分布の一意性を示している。これは定常分布の存在条件ではない

引用 : MCS.T312「マルコフ解析」講義資料　マルコフ過程入門

焼きなまし法の定式化

・焼きなまし法のアルゴリズムがどのように定式化できるか見ていこう。

・焼きなまし法を以下のようにモデル化する。

・ある温度Tの時、Pij = Gij * Aij(T)とする。（Pijはiからjへの遷移確率）
・Gijはiからjへの遷移確率
・Aij(T) = min(1, exp((Ei – Ej)/ T))とする。

・ここでEi, Ejは状態i, j での目的関数値（TSPならば経路の長さに相当）
・つまりGijで遷移先の頂点をjと決め、Aij(T)で実際に遷移するかとどまるかを選択している

・Ei <= Ejなら温度に関係なく遷移するが、Ei > Ejの時温度が下がるほど受理確率が低くなるというのは焼きなまし法のアイデアをうまく反映できている。

ギブス分布と詳細つり合い条件

・ここでギブス分布というものを考える。
・状態sの温度Tの時、以下のようにπ(s)を定義する。

$$\pi(s) = \frac{1}{Z(T)} exp\left(-\frac{E(s)}{T}\right)$$

$$Z(T) = \sum_{x \in \Omega} exp\left(-\frac{E(x)}{T}\right)$$

※E(s) : 状態sでのコスト（TSPなら総移動距離）、Ωは状態集合全体

・また定常状態が存在する十分条件についても述べておく。

・任意の二状態i, jに対し、π(i)Pij = π(j)Pjiを満たす

・つまり任意の二状態間の流量が釣り合っている

・次でギブス分布が定常状態の分布となる証明をする。

証明(1)

$$\pi(i)\,P_{ij} = \pi(j)\,P_{ji}\,,\quad Pij = Gij\,Aij(T)\,,$$

一般的な探索手法では$G_{ij} = G_{ji}$であるため,

$$\pi(i)\,P_{ij} = \pi(j)\,P_{ji} \Rightarrow \frac{A_{ij}}{A_{ji}} = \frac{\pi(j)}{\pi(i)},\ \pi(i)\ にギブス分布の式を代入すると$$

$$\frac{\pi(j)}{\pi(i)} = \frac{exp\left(-\frac{E(j)}{T}\right)}{exp\left(\frac{-E(i)}{T}\right)} = exp\left(-\frac{E(j)\ -\ E(i)}{T}\right)$$

ここで遷移の受理確率 Aij(T) = min(1, exp((Ei – Ej)/ T))を思い出してみよう

証明(2)

Case 1: $E(j) \leq E(i)$ (*解*が改善する場合)

$A_{ij} = 1, A_{ji} = exp\left(-\frac{E(j) - E(i)}{T}\right)$ より

$$\frac{A_{ij}}{A_{ji}} = exp\left(\frac{E(j) - E(i)}{T}\right)$$

Case 2: $E(j) > E(i)$ (*解*が 化する場合)

$A_{ji} = 1, Aij = exp\left(-\frac{E(j) - E(i)}{T}\right)$ より $\frac{A_{ij}}{A_{ji}} = exp\left(\frac{E(j) - E(i)}{T}\right)$

どちらのケースでも一致している。

→つまりギブス分布は定常分布の候補の一つであることが示された！

証明の成果

・焼きなまし法が既約、非周期性、斉時性、有限性を満たすことは次から明らか

・理論的な解説は**温度Tを固定**して行う 。遷移確率は状態と固定された温度のみに依存し、時間ステップにはよらないため斉時的である 。

・焼きなまし法は、**組合せ最適化問題** を主対象とする 。これらの問題では、解の組み合わせ、すなわち状態空間は有限である 。

・近傍探索は、どんな解からでも他の全ての解へ到達できるよう設計される。受理確率はゼロにならないため、生成された候補への遷移は常に可能であり、既約性が保証される。

・生成された候補への遷移が受理されず、**現在の状態に留まる** 確率は常に正である。ある状態に1ステップで留まることができれば、その状態の周期は 1となり、非周期性が満たされる。

・エルゴード性を満たすことより、ギブス分布 = 定常分布 が示された！ (参照スライド9)

温度Tが0に漸近すると...

$$\lim_{x \to +0} e^{\frac{a}{x}} = \begin{pmatrix} 1(\,if\ \ a = 0) \\ 0(\,if\ \ a < 0) \end{pmatrix}$$

右上の式を利用すると,状態sにしてπ$(s) = \frac{1}{Z(T)} exp\left(-\frac{E(s)}{T}\right)$とする

$\lim_{T \to +0} P_{T(s)} = \lim_{T \to +0} \pi(s)$, s'を状態集合の中の最適解の要素だとする。

$$\lim_{T \to +0} \pi(s) = \lim_{T \to +0} \frac{exp\left(\frac{E(s') - E(s)}{T}\right)}{\sum_{x \in \Omega} exp\left(\frac{E(s') - E(x)}{T}\right)}, \ \{x'\}\text{を最適状態の集合とする}$$

$$= \lim_{T \to +0} \left(\frac{1}{\sum_{x \in \Omega} exp\left(\frac{E(s') - E(x)}{T}\right)} (\forall s \in \{x'\}) + \frac{exp\left(\frac{E(s') - E(s)}{T}\right)}{\sum_{x \in \Omega} exp\left(\frac{E(s') - E(x)}{T}\right)} (\forall s \notin \{x'\}) \right)$$

$$= \begin{pmatrix} \frac{1}{|\{x'\}|} (\forall s \in \{x'\}) \\ 0 \ \ (\forall s \notin \{x'\}) \end{pmatrix}$$

※|{x'}| は最適状態の数

温度Tが0に漸近すると…

・先ほどの証明から、定常分布がギブス分布で与えられるとき、温度を0に近づけた極限における定常分布は真の最適解だけからなる一様分布になることが示された。

・これらの証明から何が言えるのか

・温度Tを十分にゆっくり下げることで常に状態が平衡状態を保ちながら最終的に最適解へ到達することができる
(理論上最適解を出せるという面では少しA*アルゴリズムと似ている)。

・k回目のアニーリングの温度をTkとする。Tk = T1/(logk)と設定すれば、焼きなまし法は最適状態に到達できることが確率的に保証されているが今回は触れない。
（参考:https://www.jstage.jst.go.jp/article/jjsai/9/3/9_365/_pdf）

実際の応用

・しかし一般に十分ゆっくり冷却すると時間がかかりすぎるため、基本的には指数冷却が用いられる。（以下）

$$T_{k+1} = \gamma T_k (0 \leq \gamma < 1)$$

・焼きなまし法は汎用性を持つが、それゆえパラメータを問題に対してチューニング亜上う必要があり、主に6つのパラメータを設定する必要がある。

- 最高温度（初期温度）
- 最低温度（終了温度）
- 冷却周期（1つの温度での繰り返し回数）
- 冷却回数
- 冷却率(γ)
- 総探索回数

引用: 輪湖 純也 2004

EBSの実装

・先ほど挙げたようにパラメータを問題ごとにチューニングする必要があるのだが、それを自動化したいというのが、小西ら(1994)によって手提唱された方法によってパラメーターを定めることが可能である。そのパラメータで固定化した際に温度を適応的に設定する手法である。

・温度を更新する前に多数の近傍で2-opt操作を実施しエネルギーの経験的分布を作成

・改善側のエネルギーの総量(E+)と悪化側のエネルギーの総量(E-)を求める。

・二部探索でE+ = E-となる平衡温度を二部探索で見つけ出し次の温度とする。

・以上の流れを繰り返す

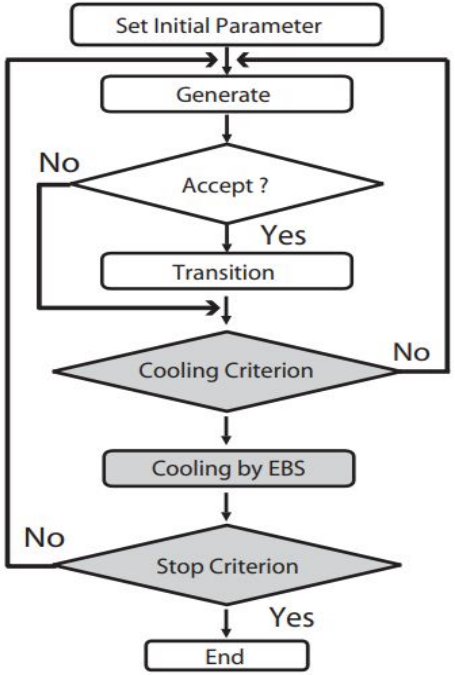


Fig. 2　SA/AT(EBS) のアルゴ リズム

出典 : 輪湖 純也　2004

EBS

・右の図より実験的にEBSのアルゴリズムが有効であることは示されている。

・しかし実際にローカルで動かそうとすると収束までが遅すぎるという欠点が挙げられる。（スパコンなどを使用できるのであれば別だが）

・少なくとも一般的なゲーミングPCでは経験的分布の作成がボトルネックになってしまい、nが大きくなると2,3分では終了しない

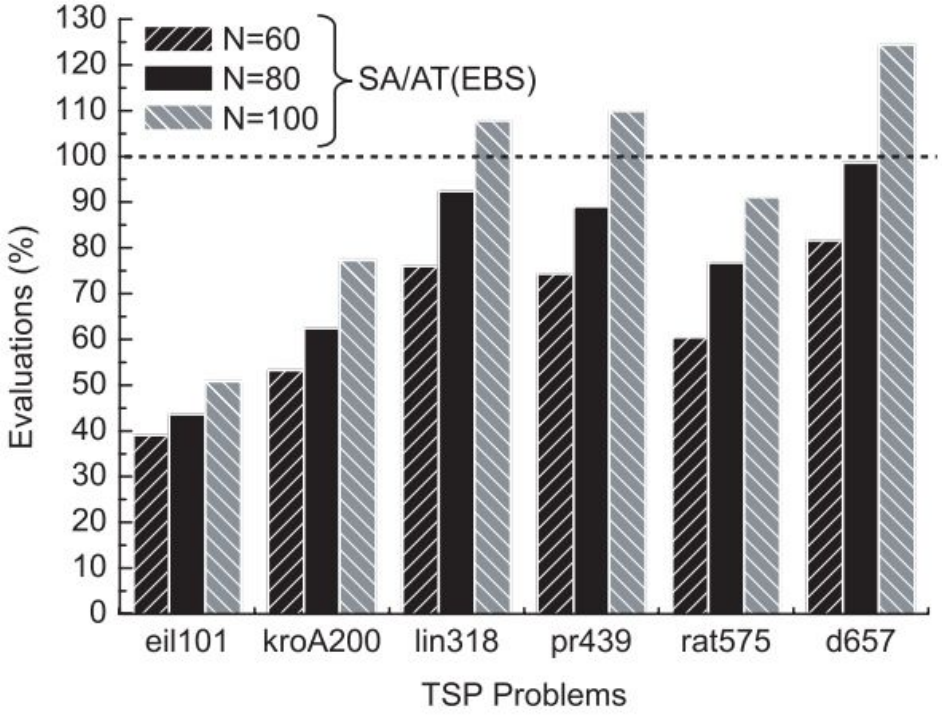


Fig. 4　SA/AT(EBS) の評価計算回数

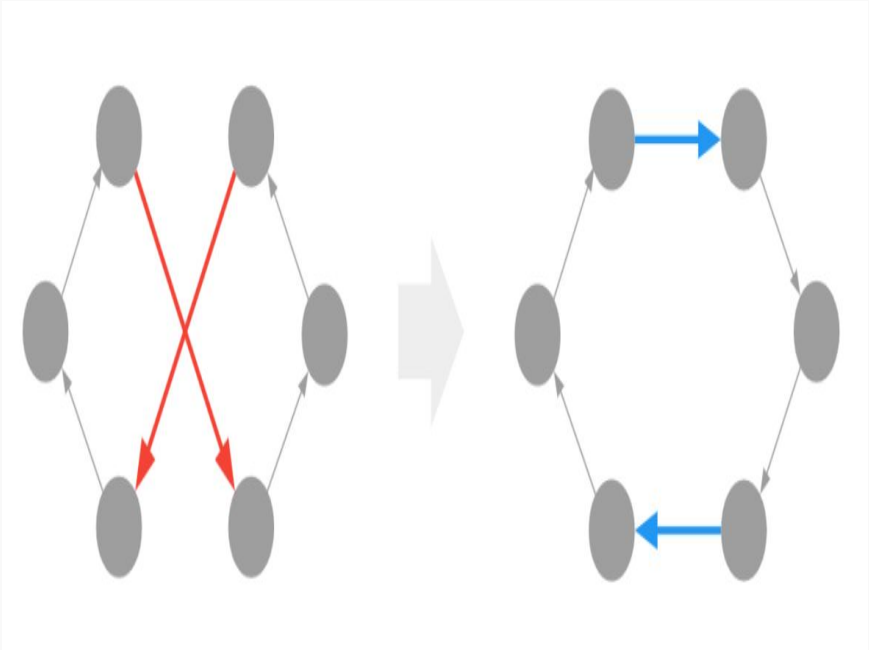
出典 : 輪湖 純也　2004

2-opt

・右図のようにパス上でi番目の頂点とj番目の頂点について考え、d(i,j)を頂点間の距離とする。

・この時、解が科改良方向に進むには以下の条件を満たしている必要がある。

$$d(i,j) + d(i+1,j+1) < d(i,i+1) + d(j,j+1)$$



引用 :
https://future-architect.github.io/articles/20211201a/#%E7%84%BC%E3%81%8D%E3%81%AA%E3%81%BE%E3%81%97%E6%B3%95-SA-Simulated-Annealing

2-opt(2)

・実際の一部コードは右のようになるがこのアルゴリズムは高速化を図ることができる。

・少し条件を緩和して削除する辺をdijとするときに、d(i,j) < d(i,i+1)またはd(j,j+1) > d(i+1, j+1)を満たすものをピックアップすることを考える。(緩和前後の条件は同値ではないが解に多様性が生まれる)

・この場合、計算量が頂点数n, ソート計算量lognで事前計算量がn**2lognとなる。

```
# 1. 異なる2つのインデックスをランダムに選ぶ
i, j = sorted(random.sample(range(N), 2))

# 2. 辺を特定
p1, p2 = current_tour[i], current_tour[(i + 1) % N]
p3, p4 = current_tour[j], current_tour[(j + 1) % N]

# 3. 差分計算: 辺(p1,p2)と(p3,p4)を辺(p1,p3)と(p2,p4)に繋ぎ変える
delta_e = (dist_matrix[p1][p3] + dist_matrix[p2][p4]) - (dist_matrix[p1][p2] + dist_matrix[p3][p4])

if delta_e < 0 or (temp > 0 and random.random() < math.exp(-delta_e / temp)):
    # 4. 経路更新: i+1からjまでを逆順にする
    current_tour[i+1:j+1] = reversed(current_tour[i+1:j+1])
    current_score += delta_e

    if current_score < best_score:
        best_tour = list(current_tour)
```

2-opt-(3)

・例えば実行時間制限ぎりぎりまで焼なましを実行し続けると仮定すると、高速化前と後で計算量は以下のように表せる。

高速化前 $O(n^2)$ x イテレーション回数

高速化後 $O(n^2 \log(n)) + o(1)$ x イテレーション回数

・高速化した方がより、多くのイテレーションを実行できることがわかるだろう。

厳密解法の紹介

・TSP問題は実は厳密解法も研究されており、現在のソルバーでは主に分枝限定法と切除平面法という二つのアルゴリズムを使用した分枝カット法について少し触れようと思う。

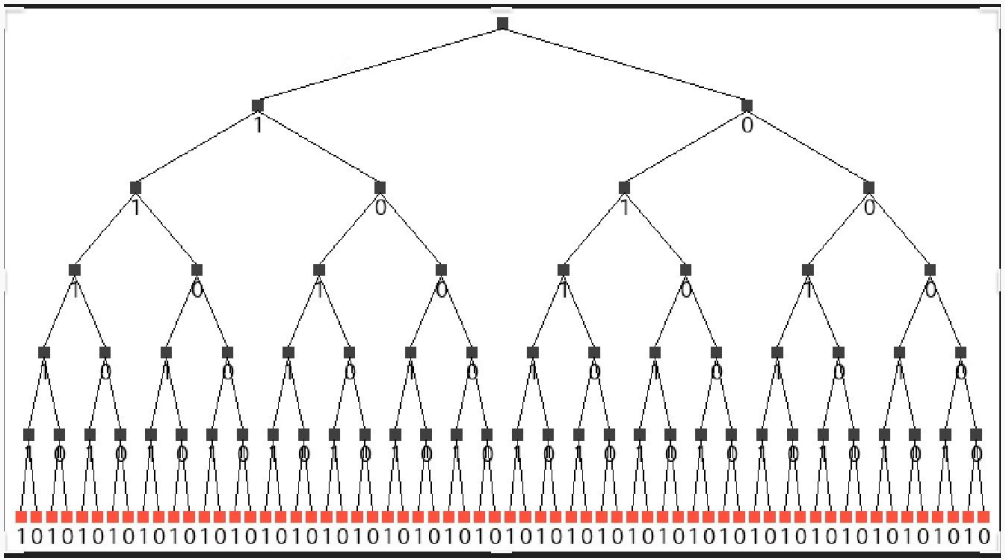
・しかし一般的にTSPはクラスNP（一般に最適解の計算量が入力長の指数倍になってしまうような問題、定義通り述べるならば非決定性チューリングマシンで処理可能な問題）に分類される問題であるため、現状では最適解を多項式時間で求めるのは不可能であることが知られている。

分枝限定法

・分枝限定法とは組み合わせ最適化問題のMLP(mixed linear programing)のソルバーで主に使用される方法である。

・一般的に主問題に対して部分問題の緩和問題を考え、緩和問題の最適解が、現在の最適解以下のであればそれ以降の枝の探索をカットするというものである。

・右図はナップザック問題に分子限定法を適用したときの図である。



引用 : https://qiita.com/SaitoTsutomu/items/7d257a855433e6917faf

ナップザック問題の定式化

ナップザック問題は右上、その緩和問題は右下のように定式化できる。

$$max \sum_i v_i x_i$$

$$s.t. \sum_i c_i x_i ,$$

$$\forall i, \quad x_i = \{0, 1\}$$

・ここで線形（ラグランジュ）緩和問題は変数のとる値域が0,1の離散値から[0,1]へと緩和されていることに注目してほしい。

・右下の緩和問題は元の問題の実行可能解をすべて含んでいることがわかる。このことからラグランジェ緩和問題の最適解は元の問題の最適解以上なることがわかる。

・つまり部分問題の緩和問題の最適解が暫定解の最適値以下ならば、それ以降の探索をしても無意味である。

・つまり探索をカットすることができる。

$$max \sum_i v_i x_i$$

$$s.t. \sum_i c_i x_i ,$$

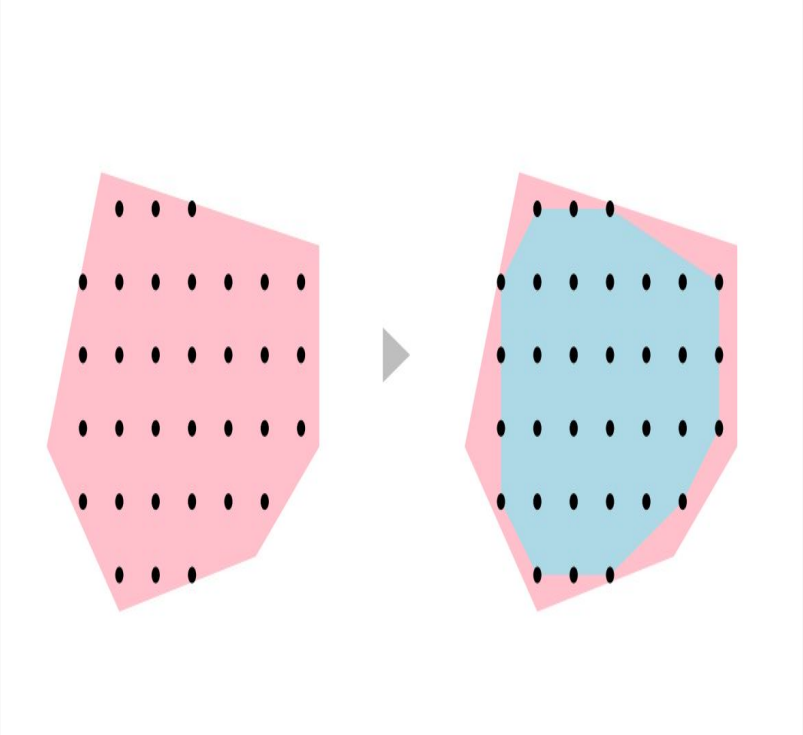
$$\forall i, \quad x_i = [0, 1]$$

切除平面法

・右図のようにMLPに対し、現在の解の存在範囲に妥当不等式を追加する。

・これにより緩和問題の解の実行可能解集合を小さくすることができ、より良い上界を与えることができる。

・妥当不等式をどのように生成するかについては今回は触れないが現在も研究が活発に行われている分野である。



引用　: http://dopal.cs.uec.ac.jp/okamotoy/lect/2022/ip/lect09.pdf

TSPでの適用

・巡回セールスマン問題ではLPとして定式化する際に、部分巡回路除去制約というものを考えなければならない。

・しかしその制約が多すぎるため、まず次数条件のみを制約に含んだ問題の線形緩和問題を考える。

・その後違反している部分巡回路を作ってしまっている場合、その巡回路を制約に追加する。（この問題は最大流最小カット定理を用いると効率的に解くことが可能）

・これを巡回路を含まなくなるまで行い、その後分枝限定法を適用する。

・これらを合わせて分枝カット法と呼称されたりする。

参考文献

・https://www.bunkyo.ac.jp/~nemoto/lecture/seminar2/2000/kimura/ronbun2.html

・https://qiita.com/take314/items/7eae18045e989d7eaf52

・https://omedstu.jimdofree.com/2018/05/04/%E3%83%9E%E3%83%AB%E3%82%B3%E3%83%95%E9%80%A3%E9%8E%96-markov-chain/

・https://note.com/dmaruyama/n/n1e144b6b8689

・https://s3.ap-northeast-1.amazonaws.com/wraptas-prod/pg-nakano-lab/aa48abe4-23a1-4b60-a438-43ce30f661ff.pdf

・https://elsur.jpn.org/diary/2023/04/16/8468/

・https://yamakuramun.info/2021/04/18/380/

・https://www.jstage.jst.go.jp/article/jjsai/9/3/9_365/_pdf

参考文献

- https://future-architect.github.io/articles/20211201a/#%E7%84%BC%E3%81%8D%E3%81%AA%E3%81%BE%E3%81%97%E6%B3%95-SA-Simulated-Annealing
- https://qiita.com/thun-c/items/ecd438fde4d237b1f7bc
- http://dopal.cs.uec.ac.jp/okamotoy/lect/2022/ip/lect10.pdf
- http://dopal.cs.uec.ac.jp/okamotoy/lect/2022/ip/lect09.pdf
- https://qiita.com/SaitoTsutomu/items/7d257a855433e6917faf
- https://www.library.osaka-u.ac.jp/doc/2014_Quotation.pdf
- http://mikilab.doshisha.ac.jp/dia/monthly/monthly04/20040927/wako.pdf
- https://is.doshisha.ac.jp/papers/pdf/06/20060301-wako.pdf

このスライドのQRコード

右のQRコードを読み込めばスライドをダウンロードできます（興味があれば）