The 33rd Annual Conference of the Japanese Society for Artificial Intelligence, 2019

クラスタリング手法を用いた TSP の解法

An Algorithm for solving the Traveling Salesman Problem using Clustering Method

内田　純平[*1]　　穴田　一[*1]
Jumpei Uchida　　Hajime Anada

[*1] 東京都市大学
Tokyo City University#1

Many economic and industrial problems lead to combinatorial optimization problems. Of these combinatorial optimization problems, the traveling salesman problem (TSP) is one of the most important problem in the field of technology and science. Therefore, we construct a new algorithm for the TSP using a new clustering method. We confirmed the effectiveness of our algorithm using several benchmark problems taken from the TSPLIB, which is a library of traveling salesman problem.

1. はじめに

工業や経済の問題の多くは, 最も効率が良い組み合わせを求める組み合わせ最適化問題に帰着することができる. その中に, 与えられた全ての都市を巡る最短経路を求める巡回セールスマン問題(Traveling Salesman Problem, TSP)がある.

本研究では, TSP に適した新しいクラスタリングのアルゴリズムである Clustering in clusters(以下 CIC)を提案し, CIC によって構成したクラスタに対して NN 法と 2-opt 法, Or-opt 法を用いることにより, 経路生成を行うアルゴリズムを構築した. そして, TSPLIB に掲載されているベンチマーク問題を用いて提案手法の評価実験を行い, その有効性を確認した.

2. 既存研究

2.1 Or-opt 法

Or-opt 法は巡回経路上の連続した n 個のノードを切り出し, それを他の位置に挿入する改善手法である.

Or-opt 法のアルゴリズムは以下の通りである

① 都市の選択
経路上の他の位置に挿入する連続した n 個のノードを選択する.

② 都市の挿入
①で選択されたノードを他のノード間に挿入し, つなぎ替えた時の総経路長が,元の総経路長より短くなる時にそのノードの挿入を行う.

①, ②を改善が見られなくなるまで繰り返す.

2.2 二次元凸包

二次元の点の集合があたえられたとき, その各点にまっすぐ釘を打ち付け, 周りに輪ゴムをかけ手を放すと, ゴムは多角形になる. このゴムに囲われた領域がこの点集合の二次元凸包である. また, 集合に属するすべての点は凸包の内側に存在する.

二次元凸包を求めるアルゴリズムは以下の通りである.

① 初期点の決定
二次元座標上の x 軸と, 原点から各点へのベクトルのなす角が最も小さい点を始点とし, 原点から始点へのベクトルを基準ベクトルとする.

② 次点の決定
始点から他の点へのベクトルと基準ベクトルのなす角を求め, なす角が最も小さい点を次の始点とし, 1 つ前の始点からこの始点へのベクトルを基準ベクトルとする.

②における次点が①で決定した初期点になるまで繰り返す.

連絡先:内田純平, 東京都市大学, 〒158-8557
東京都世田谷区玉堤 1-28-1

3. 提案手法

3.1 Clustering in Clusters (CIC)

提案手法である CIC とは, クラスタリングによって生成されたクラスタの中で, クラスタリングをするアルゴリズムである. CIC のアルゴリズムは以下の通りである.

I. クラスタリング i

① 初期設定
与えられた問題を第一象限に移動し, 二次元凸包を用いて, 与えられた問題の凸包を作る. 原点に最も近い凸包に含まれるノードを第 1 初期点(以下点 a)とし, a から最も遠い凸包に含まれるノードを第 2 初期点(以下点 b)とする。また, 与えられた問題の y 座標の最大点(以下 Max(y))を求め, 座標(0,Max(y))に最も近い凸包に含まれるノードを第 3 初期点(以下点 c)とし, c から最も遠い凸包に含まれるノードを第 4 初期点(以下点 d)とする.

なお, a, b, c, d 四ヶ所の始点から独立に次の②, ③, ④を行うものとする.

② 初期距離の決定
始点から最も近いノードを選び次の始点とする. また, ノード間の距離を基準距離 D とする. なお, 距離の計算にはユークリッド距離を用いる.

③ クラスタリング
始点から基準距離 D 以下の距離を持つノード全てと始点でクラスタを生成する. なお, a, b, c, d での 1 個目のクラスタ生成が終了後, それぞれ基準距離 D の範囲内にノードが存在しないときは全クラスタの重

心を求め, 最も近い重心のクラスタに入れるという手続きを加える.

④ 始点と基準距離の更新

クラスタリングされていないノードの中で最も始点に近いノードを次の始点とし, それらのノード間の距離 L とする. そして, 基準距離の更新を次式で定義する.

$$D = \frac{D+L}{2} \quad (1)$$

全ノードについてクラスタリングが終わるまで③と④を繰り返す.

II. クラスタリング ii

① 初期設定

各クラスタ内の最大ノード数 MN を次式で定義する.

$$MN = [AN \times 0.05] + 1 \quad (2)$$

ここで, AN は全ノード数を表す. また, クラスタリング i で生成されたクラスタをクラスタ内のノード数の降順に並び替え, この順番で②から行う.

② 始点の決定

クラスタの重心に最も近いノードを始点とする. また, 重心から始点までの距離を$d_{1,2}$とする.

③ 初期基準距離の決定

始点から最も近いノードを次の始点とし, その間の距離を$d_{2,3}$として$d_{1,2}$との平均を基準距離SD_1とする.

④ 初期クラスタリング

始点から基準距離SD_1以下の距離を持つノードを選択し, 選択されたノードと始点でクラスタを生成する. 基準距離SD_1以下の距離を持つノードが存在しないときは, 始点だけでクラスタを生成し, その始点から最も近いクラスタリングされていないノードを次の始点とする. そして新しい始点での距離を$d_{3,4}$として$d_{1,2}$, $d_{2,3}$との平均を基準距離SD_2とする.

⑤ クラスタリング

始点から基準距離$SD_i (i \geq 2)$以下の距離を持つノードを選択し, 選択されたノードと始点でクラスタを生成する. 基準距離SD_i以下の距離を持つノードが存在しないとき, 始点だけでクラスタを生成し, その始点から最も近いクラスタリングされていないノードを次の始点とする. この時, 元の始点を i , 次の始点を i+1 としたノード間の距離を$d_{i,i+1}$とする.

⑥ 基準距離 SD の更新

初期クラスタリング後の基準距離には, 次式で定義される更新式で随時更新された基準距離SD_iを使用する.

$$SD_i = \frac{\sum_{k=1}^{i+2} d_{k,k+1}}{i+1} \quad (3)$$

全ノードに対して ⑤, ⑥を繰り返す.

III. クラスタリング iii

① 初期設定

クラスタリング ii で生成されたクラスタをクラスタ内のノード数が 2 以上のクラスタのリスト(以下 clst2)とノード数が 1 つのクラスタのリスト(以下 clst1)に分ける.

② 並び替え

clst2 内のクラスタをノード数の昇順に並び替える.

③ 基準距離の設定

clst2 内のクラスタを昇順に1つ選び, 選ばれたクラスタ内でのノード間の最短距離を求める. 求められたノード間の最短距離をそのクラスタの基準距離とする. これを clst2 内のクラスタ全てに行う.

④ クラスタリング

clst2 内のクラスタを昇順に 1 つ選び, 選ばれたクラスタ内のノードを無作為に 1 つ選ぶ, そのノードと他のクラスタのノードとの距離が基準距離以下のとき, そのノードを選ばれたクラスタに入れる. クラスタのノード数が最大ノード数に達するか, クラスタ内の全てのノードから探索を終えたら, まだ選ばれていない次のノードについて同様の手続きを行う.

clst2 内の全てのクラスタに対して③, ④を繰り返す.

IV. クラスタリング iv

① 初期設定

クラスタリング iii によってクラスタリングされた clst1 と clst2 を用いる.

② 基準距離の設定

clst2 内の各クラスタの重心からクラスタ内の各ノードまでの距離を求め, 平均値の 1.5 倍を基準距離とする.

③ クラスタの分解

clst2 内のクラスタで, クラスタの重心とクラスタ内の各ノードとの距離の平均が②で決められた基準距離以上であるクラスタについて, そのクラスタ内で, クラスタの重心に一番近いクラスタ内のノードから一番離れているクラスタ内のノードをそのクラスタから外し, clst1 に入れる.

④ クラスタリング

②を行い基準距離の再設定をする. clst1 のノードにおいて, ノードを二つ選びノード間の距離が基準距離以下であれば 1 つのクラスタとして clst2 に入れる. これを clst1 内の全ノードに行う.

V. クラスタリング v

① 初期設定

クラスタリング iv によってクラスタリングされた clst1 と clst2 を用いる.

② 基準距離の設定

clst2 内のクラスタを 1 つ選ぶ. また, 選ばれたクラスタ内において, そのクラスタの重心に最も近いそのクラスタのノードを選び, 選ばれたノードと最も離れているクラスタ内のノードとの距離を選ばれたクラスタの基準距離とする. また, clst2 内のクラスタ全てに行う.

③ 候補の作成

clst2 内のクラスタを 1 つ選び, c2 とする. また, clst1 のノードを順に 1 つ選び, c1 とする. c2 内の重心を始点とし,重心と c1 の距離が c2 の基準距離の 1.5 倍以下であるならば c1 をリスト stock(以下 stock)に保存す

る. 1 つの c2 が選ぶ c1 の数の上限 SN を次式で定義する.

$$SN = MN + 1 - NN \quad (4)$$

ここで, MN は最大ノード数(式(2)), NN は c2 内のノード数を表か, clst1 内の全ノードとの比較が終了するか, c2 の選んだ c1 の数が SN に達したとき, clst2 内のまだ選ばれていない次のクラスタを c2 とする. これを clst2 内で選ばれていないクラスタがなくなるまで行う.

④ クラスタリング

stock 内の ノードを 1 つ選ぶ. 選ばれたクラスタの重心と最も近い重心を持つ clst2 内のクラスタに選ばれたノードを入れる. なお,この時のクラスタの最大ノード数は(2)式の最大ノード数に 1 を加えたものであり, これを満たさない場合は, そのクラスタを除いた, 他の clst2 内のクラスタで探す. このクラスタリングを stock 内のクラスタ全てに行う.

④が終了したとき, ②を行い各クラスタの基準距離を測定し, 各クラスタの基準距離の平均に 1.5 倍したものを次に使う基準距離とする. そして, clst1 のノードを 2 つ選び, それらのノード間の距離が基準距離以下であれば 1 つのクラスタとして clst2 に入れる. その後, ②, ③,④を行い終了.

3.2 解の構築方法

解構築に使用した手法は, 与えられた問題を CIC によってクラスタリングした後の各クラスタの重心で疑似的な最短経路を生成すると, 与えられた問題で生成した経路の厳密解と疑似経路が似通う事を利用したアルゴリズムである.

本手法のアルゴリズムは以下の通りである.

① クラスタリング

CIC(3.1)を用いてクラスタリングを行う.

② 疑似経路作成

生成されたクラスタの重心をノードとみて, 重心を結ぶ経路を NN 法と, 2opt 法, n=1 の Or-opt 法(2.1)の順で用いて最短経路になるように作成する.

③ 経路生成準備

クラスタを無作為に 1 つ選ぶ, 選ばれたクラスタの重心から最も近いノードを, 疑似経路上で次の経路の重心を持つクラスタ内のノードから選ぶ. これを②で生成された疑似経路をもとに一周するまで繰り返す.

④ 経路生成

③においてランダムに選ばれたクラスタをr_nとすると, ②の疑似経路におけるr_{n-1}の時に③で選ばれたr_nのノードを始点, r_nの時に選んだr_{n+1}のノードを終点とする. そして, 始点のノードからr_nの始点以外のノードを通り終点までの経路を NN 法, 2opt 法, n=1 の Or-opt 法(2.1)の順で用いて最短経路になるように作成し, 随時繋げていく.

生成された経路に対して 2opt 法, n=1 の Or-opt 法(2.1)を最後に行い終了する.

4. 結果

提案手法の有効性を確認するため, TSPLIB に掲載されている TSP のベンチマーク問題である eil51, kroa100, kroc100, kroa150 を用いて評価実験を行った.

表 1:eil51 の 100 試行の結果

eil51(最適解 426)	提案
平均誤差率(%)	2.6126
誤差率の標準偏差	0.6207
平均時間(秒)	1.28

表 2:kroa100 の 100 試行の結果

kroa100(最適解 21282)	提案
平均誤差率(%)	1.8286
誤差率の標準偏差	0.7378
平均時間(秒)	5.36

表 3:kroc100 の 100 試行の結果

kroc100(最適解 20749)	提案
平均誤差率(%)	1.4653
誤差率の標準偏差	0.6018
平均時間(秒)	4.10

表 4:kroa150 の 100 試行の結果

kroa150(最適解 26524)	提案
平均誤差率(%)	2.0923
誤差率の標準偏差	0.4178
平均時間(秒)	19.06

実験の結果, まだまだ十分な性能とは言えない.

5. 今後の課題

今後の課題として, クラスタリングの精度を上げることが挙げられる.提案アルゴリズムの性質上, 各クラスタの重心の位置への依存性が高く, 重心による疑似経路が厳密解と大きく離れていると, 厳密解と大きく離れてしまう欠点がある. また, (2)式より明らかだが, クラスタ内の最大ノード数は全ノード数に大きく依存しているので, 都市数の大きな問題をクラスタリングするとクラスタ 1 つ当たりのノード数が大きくなってしまい,厳密解に似通った疑似経路の作成が難しくなるという欠点もある. そこで, クラスタ内のノード数を小さくしたクラスタリングを行い, 出来たクラスタ以下を新しいノードとみてクラスタリングを行うことを一定のクラスタ数になるまで再帰的に行い, 解の構築(3.2)を生成されたクラスタの重心に行うことを, クラスタリングした回数階層的に繰り返して元の都市数に戻すことで, 現在の提案手法よりも最適解に到達しやすく, より大きな都市数の問題を解けるアルゴリズムにしたいと考えている.