SRA の技術情報誌
GSLetterNeo
Vol.166　2022 年 5 月

最適化入門
inspyred ではじめる群知能

熊澤 努

株式会社 SRA 技術本部 先端技術研究室

株式会社 SRA ホールディングス 先端技術研究所

はじめに

GSLetterNeo Vol. 156[1]と 158[2]で最適化問題、ならびに Python と Google OR Tools を使った解法を紹介しました。今回はその続きとして、最適化問題を解くための無償の Python パッケージ inspyred[3]を紹介します。inspyred は進化計算という問題解決技術を実装したパッケージです。進化計算の詳細には立ち入らず、具体的な問題を inspyred を使って解くプログラムを作ってみます。

[1] https://www.sra.co.jp/Portals/0/files/gsletter/pdf/GSLetterNeoVol156.pdf（2022 年 5 月 12 日閲覧）

[2] https://www.sra.co.jp/Portals/0/files/gsletter/pdf/GSLetterNeoVol158.pdf（2022 年 5 月 12 日閲覧）

[3] https://aarongarrett.github.io/inspyred/（2022 年 5 月 12 日閲覧）

日本のすべての都道府県庁所在地を訪れる

次の出張を計画する問題を考えてみましょう。

あなたはある会社の営業担当社員として、お客様を訪問するための出張計画を立てています。今回の訪問先は全国47都道府県庁所在地の近くにあり、長期の出張が必要そうです。ただ、予算や期間にも限りがあるため、できる限り少ない移動距離ですべての都道府県庁所在地を訪問したいと考えています。どの順番で訪問するのが望ましいでしょうか。

問題を少し扱いやすくするため、次のような場合を考えることにします。

- ✓ 最初に訪問する場所は任意に決めてよい。
- ✓ 各都道府県庁所在地には一度だけ訪問するものとする。
- ✓ 各都道府県庁所在地の間は最短距離で移動できると考えてよい。交通機関の制約や宿泊地などの状況は考慮しない。

今回は出張の間の総移動距離の最小化を考えることにします。まず、移動距離を算出するために、日本の都道府県庁所在地同士の距離を知る必要があります。今回は、国土交通省国土地理院が公開している「都道府県庁間の距離」データ[4]を使用します。

複数の都市を 1 回ずつ訪問する問題は、巡回セールスマン問題としてよく知られています。現在のところ、巡回セールスマン問題をコンピュータで解くための効率的な解法は知られていません。考えられる訪問順序を全て列挙して一つずつ総移動距離を計算した後、最も総移動距離の少ないルートを選べば答えは得られます。しかし、訪問順序には約2.6×10^{59}通りの可能性があり、すべての組合せを列挙することはできそうにありません（一つのルートにつき1マイクロ秒程度の計算時間としても約8.2×10^{45}年必要！）。沖縄の次に北海道を訪れる、といった見込みのなさそうな場合を含むルートを除去すれば良いのでは、とも考えられますが、各ルートについて見込みがあるかどうかを判定する処理が追加で必要になってしまい、大きな効果は見込めません。

群知能を使った解法

出張計画の問題を現実的な時間で解く方法には、進化計算や群知能を使う解法があります。これらの方法は、生物をはじめとする集団の挙動が生み出す性質をヒントに作られた問題解決法の総称で、数多くのアルゴリズムが作られています。この記事では、Python の無償ライブラリである inspyred を使って出張計画問題を解くことにします。筆者は Python 3.6、inpyred 1.0.1 を使用しました。

[4] https://www.gsi.go.jp/KOKUJYOHO/kenchokan.html （2022 年 5 月 12 日閲覧）

inspyred のインストールには、pip コマンドを使います。

```
> pip install inspyred
```

inspyred は多くの進化計算アルゴリズムを提供していますが、その中からアリコロニー最適化法を使ってみましょう。アリコロニー最適化法は、働きアリが餌を求めて行列を作る過程に注目した最適化法の一種です[5]。アリコロニー最適化法にもたくさんの種類がありますが、inpyred 1.0.1 で利用できるのは Ant Colony System (ACS)という方法です。

出張計画問題を解く前に準備が必要です。アリコロニー最適化法を使うためには、国土交通省国土地理院の「都道府県庁間の距離」データを読み込んで、Python プログラムで都市間の距離の表を作っておく必要があります。例えば、千葉県、東京都、神奈川県に限定した場合、以下のような表を作ることになります。なお、距離データは、2022 年 5 月 12 日現在のものです。

	千葉	東京都	神奈川
千葉	0	40.2km	47.0km
東京	40.2km	0	27.2km
神奈川	47.0km	27.2km	0

筆者は openpyxl[6]で Excel ファイルデータを読み込み、表を作成するようにプログラムを作りました。プログラムが長くなってしまうため、以降ではデータを読み込んで表を作る処理は省略します。

[5] アリコロニー最適化法は以下の文献で詳しく説明されています。

M. Dorigo and T. Stützle. *Ant Colony Optimization*, Bradford Company, MIT Press, 2004.

大谷紀子. 進化計算アルゴリズム入門 生物の行動科学から導く最適解、オーム社、2018.

[6] https://openpyxl.readthedocs.io/en/stable/ （2022 年 5 月 12 日閲覧）

出張計画問題を解くプログラムを以下に掲げます[7]。

tsp_acs.py

```
import inspyred
import random
import time

# 最良の出張計画をアリコロニー最適化法で作る
def tsp_acs(cities, distance_table):

    # (1) 出張計画問題の生成
    tsp = inspyred.benchmarks.TSP(distance_table)

    # (2) アリコロニー最適化法 (Ant Colony System: ACS)のソルバーの生成
    solver = inspyred.swarm.ACS(random.Random(time.time()), tsp.components)
    solver.terminator = inspyred.ec.terminators.generation_termination

    start_time = time.time()

    # (3) ACSの実行
    solver.evolve(
        generator=tsp.constructor,
        evaluator=tsp.evaluator,
        bounder=tsp.bounder,
        maximize=problem.maximize,
        max_generations=10)

    elapsed_time = time.time() - start_time

    best = max(solver.archive)

    # (4) 最良解の出力
    print('最良ルート:')
    print([cities[c.element[0]] for c in best.candidate] +
[cities[best.candidate[-1].element[1]]])
    print(f'移動距離: {1 / best.fitness} km')
    print(f'実行時間: {elapsed_time} sec')


...(「都道府県庁間の距離」データの読み込み、表の作成など)...
# 出張計画を作成する
tsp_acs(japan, distance_table(japan))
```

プログラム中のコメントにつけた番号順に説明します。

(1) inspyred で取り扱える出張計画問題を生成します。変数 `distance_table` は上で説明した都道府県庁所在地間の距離の表です。表は二次元のリストです。二次元リストの添え字を都道府県庁所在地に対応させるため、変数 `cities` に都道府県庁所在地名の

[7] 掲載したプログラムは、以下の inspyred の公式ドキュメントをもとに作成しました。
https://aarongarrett.github.io/inspyred/examples.html#the-traveling-salesman-problem
（2022 年 5 月 12 日閲覧）

リストを格納しておいて対応表とします。inspyred が提供する TSP は巡回セールスマン問題のことで、距離の表から解くべき問題を作成します。このように、inspyred には有名な問題群を簡単に扱える仕組みが用意されているので、ユーザは問題を記述したプログラムを書く必要はありません。ユーザが独自に問題を定義することも可能です。

(2) アリコロニー最適化法 ACS を使って問題を解くためのソルバーを生成します。ACS は乱数を使った乱択アルゴリズムの一種なので、ソルバーの生成に際して第 1 引数で擬似乱数生成器 `random.Random` オブジェクトを渡しています。第 2 引数は問題に依存した情報で、ACS が使用します。`solver.terminator` は ACS の終了条件です。ここでは、一定回数だけ最適な出張計画の生成を試みたら終了することにしています。

(3) ACS を実行して、最適な出張計画を生成します。`evolve` メソッドの引数の内、最初 4 つは出張計画問題の情報です。最後の `max_generations` は最適計画の生成試行回数で、10 回としています。ACS には他にも設定値がありますが、ここではすべてディフォルト値にしました。

(4) ACS が求めた計画から総移動距離が最小の計画を最良解として出力します。加えて、最良解での総移動距離と、最良解を求めるまでの経過時間も出力します。

出張計画問題を解く

筆者の手元の PC でプログラムを実行すると、以下のような結果になりました。100 秒ほどで無事に解が求められました。今回の出張は、奈良県からスタートして、次に兵庫県、そして最後に沖縄県を訪問して終了するというルートがよさそうです。

```
最良ルート:
['奈良', '兵庫', '大阪', '京都', '滋賀', '三重', '愛知', '岐阜', '福井', '石川', '富山', '長野', '群馬', '栃木', '茨城', '神奈川', '東京', '埼玉', '千葉', '山梨', '静岡', '和歌山', '徳島', '香川', '島根', '鳥取', '岡山', '高知', '愛媛', '広島', '山口', '大分', '熊本', '佐賀', '福岡', '長崎', '宮城', '山形', '福島', '新潟', '秋田', '青森', '岩手', '北海道', '宮崎', '鹿児島', '沖縄']
移動距離: 8558.6 km
実行時間: 107.27230906486511 sec
```

このルートをよく見ると、北海道から宮崎県までの移動が含まれており、長距離の移動が必要であるように思えます。これより望ましいルートはないのでしょうか。プログラムをもう一度実行してみましょう。

```
最良ルート:
['香川', '岡山', '徳島', '鳥取', '島根', '静岡', '山梨', '群馬', '栃木', '千葉', '東京', '埼玉', '神奈川', '茨城', '福島', '山形', '秋田', '岩手', '青森', '北海道', '宮城', '新潟', '長野', '富山', '石川', '福井', '岐阜', '愛知', '滋賀', '京都', '大阪', '奈良', '兵庫', '和歌山', '三重', '長崎', '熊本', '佐賀', '福岡', '愛媛', '山口', '広島', '大分', '宮崎', '鹿児島', '沖縄', '高知']
移動距離: 7341.299999999999 km
実行時間: 88.51622819900513 sec
```

結果が変わりました。総移動距離が約 7,300km のルートがありました。このように、ACS は実行速度が速い代わりに、必ずしも最適な計画を算出するとは限りません。今度の出張では 2 回目のルートが予算内に収まるので、2 回目に算出した計画を採用することにしましょう。

おわりに

既に述べたように、inspyred には他にも様々な進化計算技術を実装しています。たくさんある進化計算技術は個々に特徴があり、今回使ったアリコロニー最適化法とは異なる実行時間で結果が得られることもあると思います。実行時間や解の望ましさが解法選定の基準になるので、解きたい問題に適合した解法を見つけてみてください。

GSLetterNeo Vol.166
2022 年 5 月 20 日発行
発行者　株式会社 SRA 技術本部 先端技術研究室

編集者　熊澤努　方 学芳
バックナンバー https://www.sra.co.jp/public/sra/gsletter/
お問い合わせ　gsneo@sra.co.jp

株式会社SRA
〒171-8513 東京都豊島区南池袋 2-32-8

夢を。Yawaraka Innovation
やわらかいのべーしょん