

人工知能 中間レポート 課題 10

03-230450 平田 佑亮

2024 年 1 月 14 日

1 問 1

LLM にメタヒューリスティックの例を 5 個上げさせ、その詳細について説明させた。まず、打ち込んだプロンプトが、以下である。

群知能のメタヒューリスティックスや、組み合わせ探索を行うメタヒューリスティックの例を 5 つ上げ、それぞれについての説明を詳しく行ってください。コウモリ探索や Ant Colony Optimization のような細かい手法を挙げることを期待します。

それに対する LLM の答えは、以下である。

了解しました。コウモリ探索とアリコロニー最適化を除外して、他のメタヒューリスティックスの例について説明します。

1. **粒子群最適化 (Particle Swarm Optimization, PSO) **: **原理**: ** 鳥や魚の群れが協力して飛行や泳ぎを行う原理に基づく最適化手法。- **仕組み**: ** 解候補を粒子として表現し、これらの粒子が解空間を動き回りながら、個々の粒子の過去の最良解とグローバルな最良解を参照して更新される。集団の協力により最適解を発見する。

2. **人工蜂コロニー最適化 (Artificial Bee Colony, ABC) **: **原理**: ** 蜜蜂の巣での餌の探索行動を模倣した最適化手法。- **仕組み**: ** 解候補を蜜蜂として表現し、個々の蜜蜂が解の評価を行い、良い解を見つけると他の蜜蜂にその情報を伝える。良い解が見

つかると局所的な探索が進み、集団全体が最適解に収束する。

3. **遺伝的アルゴリズム (Genetic Algorithm, GA) **: **原理**: ** 生物学的進化の原理を模倣し、遺伝子の操作によって最適解を進化させる手法。- **仕組み**: ** 解候補を遺伝子として表現し、個体間で交叉や突然変異を行いながら新しい解の生成と進化を促進する。適応度に基づいて個体を選択し、最終的に最適解を得る。

4. **模擬アニーリング (Simulated Annealing, SA) **: **原理**: ** 冷却過程を模倣して局所的な最適解に囚われず、探索空間全体を探索する手法。- **仕組み**: ** 初期温度を設定し、一定のスケジュールに基づいて温度を下げながら探索を進める。一定の確率で悪い解を受け入れることで、局所最適解を回避し、最終的には良い解に収束する。

5. **巡回セールスマン問題アルゴリズム (TSP Algorithm) **: **原理**: ** 巡回セールスマン問題において最短経路を見つける手法。- **仕組み**: ** 探索空間内の異なる経路を生成し、それらを評価して改良する。局所的な探索や交叉、突然変異などの操作により、最短経路を見つけ出す。

2 問 2

問 1 での各手法に関して、重要な特徴や、探索性能に影響与える要素について LLM に

解説させた。

まず、打ち込んだプロンプトが、以下である。

それぞれの方法における、重要な特徴や探索性能に影響を与える要素について詳しく解説してください

それに対する LLM の答えは、以下である。

それぞれのメタヒューリスティック手法には、特有の特徴や探索性能に影響を与える要素が存在します。以下に、それぞれの手法における重要な要素について詳しく解説します。

それぞれのメタヒューリスティクス手法には重要な特徴や探索性能に影響を与える要素があります。以下に各手法に対する特徴や重要な要素について解説します。

1. ****粒子群最適化 (PSO) : ** - **特徴: ****
- パラメータ: 粒子の速度、位置、および各粒子の最良位置やグローバルな最良位置が重要。- グローバルベスト (GBest) とパーソナルベスト (PBest) のバランスが重要であり、適切な調整が性能に大きな影響を与える。- ****影響要素: **** - 速度と位置の更新における適切な加速度係数や重み係数の選択。- 収束速度と探索範囲のトレードオフを調整するパラメータ。

2. ****人工蜂コロニー最適化 (ABC) : ** - **特徴: **** - 蜜蜂が解空間内でランダムに探索し、良い解が見つかったら他の蜜蜂に情報を伝える。- 局所探索と大域探索のバランスを保つ。- ****影響要素: **** - パラメータの設定が重要で、探索空間に対する探索の広がりや局所探索の強さを調整する。

3. ****遺伝的アルゴリズム (GA) : ** - **特徴: **** - 遺伝子操作 (交叉、突然変異) により新しい解を生成し、適応度に基づいて個体を選択する。- 適応度関数が解の評価に影響

する。- ****影響要素: **** - 交叉率と突然変異率の適切な設定。- 選択手法 (トーナメント選択、ルーレット選択など) の選定。

4. ****模擬アニーリング (SA) : ** - **特徴: **** - 温度を下げることで受け入れる悪い解の確率が減少し、探索が局所的な最適解に収束する。- 局所探索と大域探索のバランスを制御する。- ****影響要素: **** - 冷却スケジュールの設定が重要であり、探索の進行によって温度がどのように下がるかが影響を与える。

5. ****巡回セールスマン問題アルゴリズム (TSP Algorithm) : ** - **特徴: **** - 解の表現が順列であり、操作が順列の交換に基づく。- 局所最適解に陥りやすい特性がある。- ****影響要素: **** - 局所探索手法の選定や、局所探索をどれだけ利用するかとのバランス。- 交叉操作の設計において、どれだけ局所的な構造を保持するかが重要。

これらの手法は問題によって異なる性能を発揮するため、適切なパラメータ設定や手法の選択が重要です。問題の特性や求める解の性質に応じて適切な手法とパラメータを選定することが、良好な探索性能を実現する鍵となります。

3 問 3

巡回トーナメント問題を、複数の手法のハイブリッドで解くように LLM に指示した。

まず、打ち込んだプロンプトが、以下である。

いずれかの手法の改良版、もしくは複数の手法のハイブリッド版を用いて、以下の制約を持つ巡回トーナメント問題を解く手法を提案し、更にそれを実行する python コードを書いてくださいハード制約:

各チームは、他のチームとそれぞれ 2 回

(ホームとアウェイ) 試合をする。各チームは、各時間枠に 1 試合だけ行う。3 回連続で、ホームまたはアウェイでの試合はできない。繰り返しなし: 同じ対戦相手と 2 回連続で対戦できない。ソフト制約:

全チームが移動する合計距離を最小限に抑える。

それに対する LLM の答えは、以下である。

制約の複雑性や問題の性質により、最適な手法やハイブリッド手法は異なりますが、ここでは遺伝的アルゴリズム (GA) と模擬アニーリング (SA) のハイブリッド手法を使用して、制約付き巡回トーナメント問題を解く方法を提案します。

以下に、Python コードの簡単な実装例を示します。ただし、問題の具体的な形式によっては、より詳細な調整が必要です。また、このコードは単なる例であり、問題によっては適切なパラメータの調整や探索戦略の変更が必要です。

```
“python import random import numpy
as np from copy import deepcopy
```

```
def calculate_distance(solution) :
各チームの移動距離を計算する関数distance =
0foriinrange(len(solution)) :
distance+= abs(solution[i]-solution[(i+
1)returndistance
```

```
def is_valid(solution) :
ハード制約を満たしているかを確認する関数foriinrange(len(solution)):
if(solution[i] == solution[(i +
1)(abs(solution[i] - solution[(i +
1)(abs(solution[i] - solution[(i +
2)returnFalsere turnTrue
```

```
def genetic_algorithm(population_size,generations)ランダムに 2 つの位置を選
遺伝的アルゴリズムpopulation = 択して入れ替え idx1, idx2 =
[list(range(1,11))forinrange(population_size)]best_solution = random.sample(range(len(new_solution)),2)new_solution[idx
None new_solution[idx2], new_solution[idx1]
```

```
for gen in range(generations): popula-
tion = sorted(population, key=lambda x:
calculate_distance(x))
```

```
一定割合の上位個体を選択
selected_parents = population[:
int(0.2 * population_size)]
```

```
選択した個体から新しい個体を生成
children = [] for inrange(population_size -
len(selected_parents)) :
parent1,parent2 =
random.sample(selected_parents,2)crossover_point =
random.randint(1,len(parent1) -
1)child = parent1[: crossover_point] +
[xforxinparent2ifxnotinparent1[:
crossover_point]]children.append(child)
```

```
突然変異を適用 for child in chil-
dren: if random.random() < 0.1: idx1,
idx2 = random.sample(range(len(child)),
2) child[idx1], child[idx2] = child[idx2],
child[idx1]
```

```
合体して新しい世代を作成 population =
selected_parents + children
```

```
ハード制約を満たす解が得られ
たら終了 for solution in population:
if is_valid(solution) : best_solution =
deepcopy(solution)break
return best_solution
```

```
def simulated_annealing(solution,temperature,cooling_rate,iterations)
模擬アニーリングforiinrange(1,iterations):current_solution =
deepcopy(solution)best_solution =
deepcopy(solution)
for inrange(iterations) : new_solution =
deepcopy(current_solution)
```

```
ランダムに 2 つの位置を選
択して入れ替え idx1, idx2 =
random.sample(range(len(new_solution)),2)new_solution[idx
new_solution[idx2], new_solution[idx1]
```

評価値が改善された場合、または一定確率で新しい解を受け入れる

```
if calculate_distance(new_solution) <
```

```
calculate_distance(current_solution) or random.random() <
```

```
np.exp((calculate_distance(current_solution) -  
calculate_distance(new_solution)) / temperature) :
```

```
current_solution = deepcopy(new_solution)
```

```
    最良解を更新    if  
calculate_distance(current_solution) <
```

```
calculate_distance(best_solution) :
```

```
best_solution = deepcopy(current_solution)
```

```
    温度を下げる    temperature *=  
cooling_rate
```

```
    return best_solution
```

```
def hybrid_algorithm(population_size, generations, sa_temperature, sa_cooling_rate, sa_iterations) :
```

遺伝的アルゴリズムと模擬アニーリングのハイブリッド手法

```
ga_solution = genetic_algorithm(population_size, generations)
```

```
sa_solution = simulated_annealing(ga_solution, sa_temperature, sa_cooling_rate, sa_iterations)
```

ハイブリッドアルゴリズムの実

```
行    population_size = 50    generations =
```

```
100    sa_temperature = 100    sa_cooling_rate =
```

```
0.95    sa_iterations = 1000
```

```
    best_solution =
```

```
hybrid_algorithm(population_size, generations, sa_temperature, sa_cooling_rate, sa_iterations)
```

```
    print("最適なスケジュール:",
```

```
best_solution)    print("移動距離" :
```

```
", calculate_distance(best_solution))    print("ハード制約を満たしているか :
```

```
", is_valid(best_solution))
```

このコードでは、遺伝的アルゴリズムと模擬アニーリングを組み合わせ、ハード制約付き巡回トーナメント問題を解く手法を提案しています。パラメータや手法の選択は問題によって調整が必要です。

4 問4

問3でLLMが出力したコードを実際に行した。

まず、文法エラーが見られた。これは、

genetic_algorithmという関数において最適解が見つからなかった。

参考文献

[1] <https://www.codingwith-thomas.com/blog/c-genetic-algorithm>