修 士 論 文

サービスの利用頻度と価値に応じた利用順探索

令和２年度 修了
三重大学大学院　工学研究科
博士前期課程　情報工学専攻
コンピュータソフトウェア研究室

熊谷　朋彦

目 次

第１章　はじめに

定期的に同じ種類の店舗を利用することはよくある。例えば多くの人が、買い物をするために定期的にスーパーやコンビニに行く。また、趣味のために定期的に行く同じ種類の店舗（例えばカラオケなど）があるという事も多いだろう。以上のような定期的に訪れる同じ種類の店舗は多い。

しかし、そのようなサービスへ効率良く行くのは簡単では無い。いくら定期的に利用するからとはいえ、全てのケースで同じ周期で行くのが効率が良いとは限らない。例えば、飲食店などでは味、値段、行きやすさなどの要素から価値を考えることができる。そういったサービスの価値を考えなければならない。

また、サービス単体の価値ばかり考えてもいけない。飲食店であれば、たとえ商品の価値が高くとも、同じ食べ物ばかり行っては飽きるだろう。他の例として、食品を買うことを考える。同じものであれば、安い方が一般的に良いだろうが、特に生鮮食品などでは買い置きは困難である。このようなことから、サービスを訪れる間隔を適切にする必要がある。

以上のことから、上記のような適切な利用間隔と価値という２つの要素を、適切に重み付けした経路探索は重要であり、本研究ではその定式化、解法の提案を行う。本研究ではその中でも特に長期の移動に注目し、長期の移動に利用できる経路探索を提案する。

第2章　関連する問題

以下の問題での「点」は、本研究でのサービスのことを指す。

2.1　最短経路問題 [1]

グラフ、始点、終点が与えられ、同じ頂点を複数回通らない最短経路を求める問題である。辺のコストの総和の最小化という意味では本研究に似ているが、点の価値が無く、利用間隔の考慮が無いという違いが有る。以下ではこの問題を具体的な例で説明する。

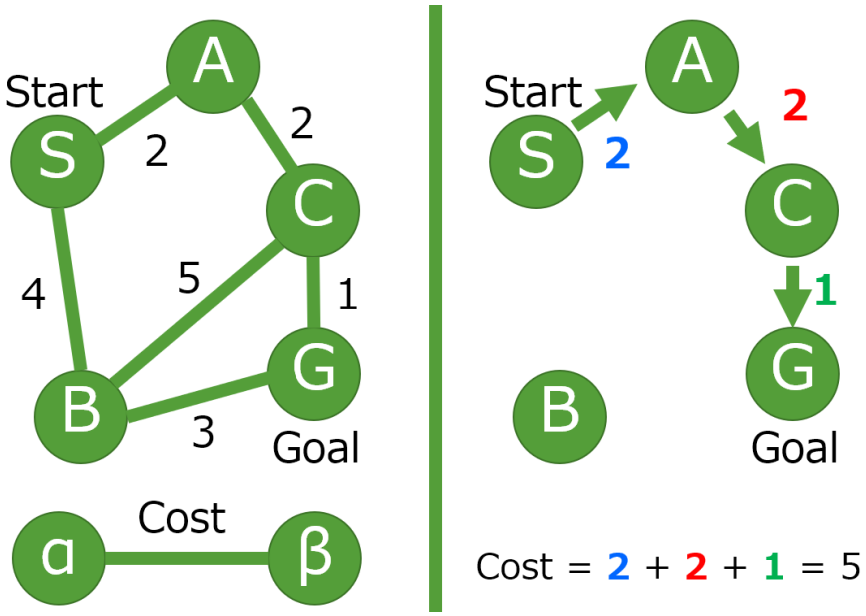

図 2.1: 最短経路問題の問題例

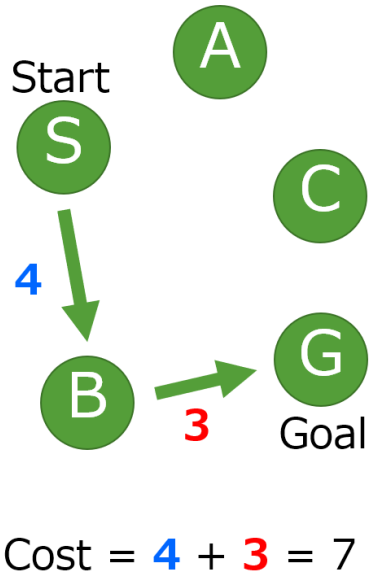

図 2.2: 最短経路問題の問題例

図 2.1 の左の図ではこの問題の入力を示している。入力は無向グラフで、S は Start、G は Goal を表し、各ノード間の重みはコストを示している。例えば、A・C 間を移動するにはコスト 2 が必要である。図 2.1 の右図は S、A、C、G を順に通る経路である。この経路の評価値は 2＋2＋ 1 ＝5 である。図 2.2 では S、B、G を順に通る経路である。この経路の評価値は 7 である。
同様に計算すると、S、B、C、G を順に通る経路の評価値は 10 で、S、A、C、B、G を順に通る経路の評価値は 12 である。同じ頂点を 2 回以上通らず、Start から始まり Goal で終わる経路は以上であり、最小値は 5 である。よって、この入力での解は 5 となる。

2.2 最長単純道 (longest path)[1]

グラフ、始点、終点が与えられ、同じ頂点を複数回通らない最長経路を求める問題である。辺の長さを価値と見るとすると、本研究に似ている問題として見ることが出来る。ただし、価値としてみる対象を点と辺としている違いや、利用間隔の考慮が有無などの違いが有る。

2.3 巡回セールスマン問題 [1]

与えられた点全てを巡回する最短経路を求める問題。本研究では巡回セールスマン問題の挿入法に似た解法を提案する。

2.4 最適観光経路問題 [2]

この問題は、観光経路の評価値を計算する問題である。

第3章　提案手法

本研究では、最短経路問題に点の価値と重み付けを導入した定式化を考える。重み付けの詳細は以下の通りである。

重み付け関数 w を用いて利用間隔と価値のバランスを考える。a を一定時間内に利用した同じ種類のサービス数とし、w を単調減少関数（例えば 2^{-a} など）とする。そして、$\{v \times w(a)\}$ を価値として見る (v は頻度を考慮しない価値である)。すると、サービスごとに固定の利用間隔を決めるものに比べ、価値の高さによって利用間隔が変わる。そして、w により利用間隔が狭くなりすぎる事や、広くなりすぎる事を防ぐことが出来る。

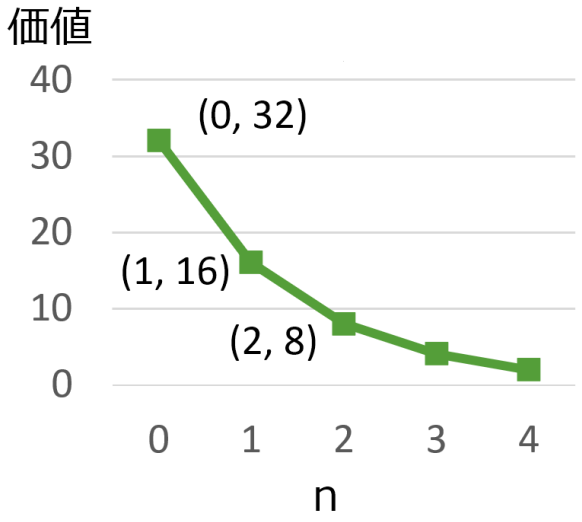

図 3.1: べき乗

上図では頻度考慮なし価値 $= 32$、$w(a) = 2^{-a}$ の時のグラフを示している。頻度考慮なし価値が同じ 32 でも a が変わる、つまり一定時間内に利用した同じ種類のサービス数が変わると、価値が $16, 8$ などになる。

3.1 重み関数の例

重み関数を実際の例で応用しやすいのように、複雑な例を示す。
$c_{t,j}$: 時刻 t のサービスグループ g_jの係数

$$c_{t,j} = \sum_{t'=t-t''}^{t} a'_t \times b_{td} \ (td = t - t') \tag{3.1}$$

a'_t : 時刻 t'に利用したサービスの影響力
ただし、時刻 t'にサービスを利用していない場合 0

$b_{td} = 1 - \ td/(t - t'')$
基本的に $w(0) = 1, w$ は減少関数
各グループ $g_1, ..., g_n$の重み優先度が $p_1, ..., p_n$の時、
価値 = 頻度考慮なし価値 $\times\, w(s, i)$

$$w'(s,i) = w^{+}(s,i) \quad \left(\text{ または } w^{\times}(s,i) \right) \tag{3.2}$$

$$w^{+}(s,i) = \frac{\sum_{j=1}^{n} p_j}{\sum_{j=1}^{n} p_i c_{t,i}} = \frac{p_1 + ... + p_n}{p_1 c_{t,1} + ... + p_n c_{t,n}} \tag{3.3}$$

$$\left\{ \ w^{\times}(s,i) = \frac{\prod_{j=1}^{n} p_j}{\prod_{j=1}^{n} (c_{t,j})_j^p} = \frac{p_1 \times ... \times p_n}{(c_{t,1})^{p_1} \times ... \times (c_{t,n})^{p_n}} \ \right\} \tag{3.4}$$

$$※ \prod_{k=a}^{b} c_k = c_a \times ... \times c_b \tag{3.5}$$

3.2 入力

V	頂点集合
s, d	始点, 終点
$v : V \to \mathbb{R}_+$	価値関数
$w' : V^+ \times I \to \mathbb{R}_+$	重み関数
$c : V \times V \to \mathbb{R}$	コスト関数
$k : I \times I$	種類関数
$t : V \times V \to \mathbb{R}$	時間関数

3.3 変数

r : 経路を表す V の要素の列
r_i : i 番目に利用するサービス

3.4 入力の詳細

b : $route$ の大きさ
src : 出発点
dst : 目的地
V : サービスと出発点、目的地の和集合s
$v(r_i)$: サービス r_iの価値
$w'(r, i)$: 重み関数
$c(r_{i-1}, r_i)$: r_{i-1}から r_iへ行くために必要なコスト
$c(r_i)$: サービス r_iを利用するために必要なコスト

3.4.1 目的関数

$$Score(r) = \sum_{i=2}^{b-1} v(r_i) \times w'(r, i) - \sum_{i=2}^{b} c(r_{i-1}, r_i) - \sum_{i=1}^{b} c'(r_i) \quad (3.6)$$

$$\to \text{最大化} \quad (3.7)$$

3.4.2 制約

$b \geq 2$
$r_1 = src$
$r_b = dst$
$\forall i, j \in [1, b](i \neq j) \rightarrow (r_i \neq r_j)$

3.4.3 具体例

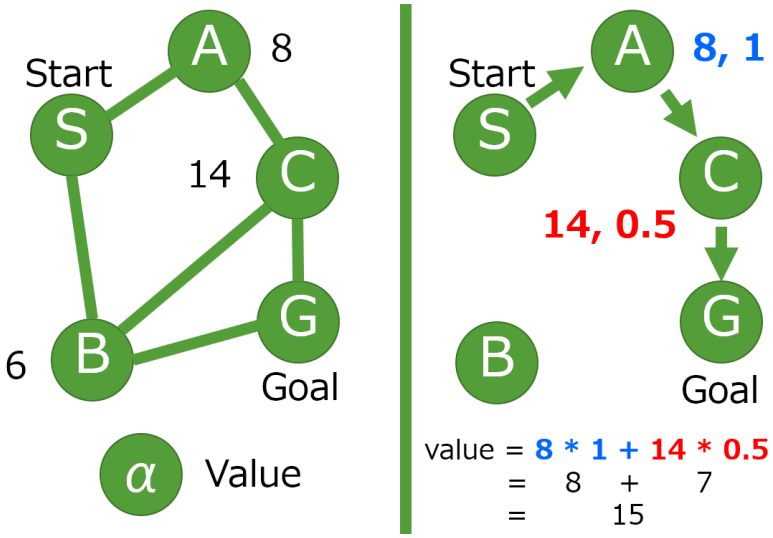

図 3.2: 価値

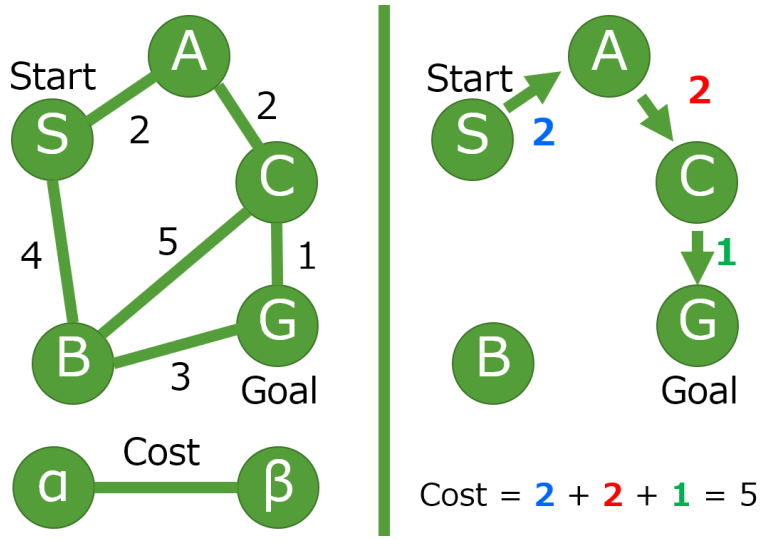

図 3.3: 経路コスト

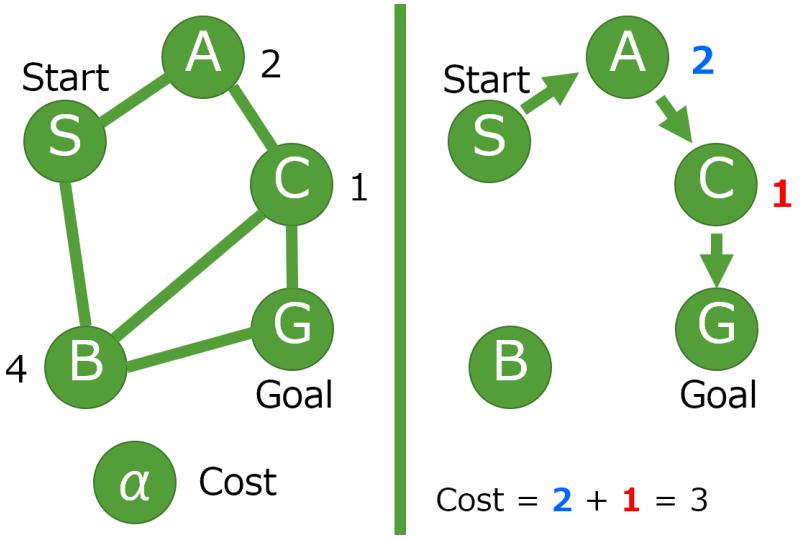

図 3.4: サービスコスト

図 3.2, 3.3, 3.4 の左図は入力を表している。例えば、図 3.2 の左図はそれぞれのサービスの価値を表している。S, A, C, G の価値の総和、経路コストの総和、サービスコストの総和はそれぞれ 15, 5, 3 であり、この経路の評価値は $15 - 5 - 3 = 7$ となる。

3.5 厳密解法

c をサービスの総数とする。求める経路の総数は

$$\sum_{i=0}^{c}({}_c\mathrm{P}_i) \tag{3.8}$$

である。これを全て求めるのは大きな c に対しては現実的な時間で解けない。また、重み関数 w と同じ点に二度訪れられないという制約から、多項式時間で解けるアルゴリズムに適用するのは困難だと考えられる。

ただし、利用するサービスの数の最大数を m 個とすると、総数は

$$\sum_{i=0}^{m}({}_{c+1}\mathrm{P}_i) \tag{3.9}$$

となり、利用するサービスの最大数が大きくない時、全探索でも実用的な時間で探索することが出来る。

次ページに厳密解法の擬似コードの例を示す。以下のコード内の recent は次の地点から見て一定の長さ certain_len（時間、距離など）以内に訪れたサービスの、次の地点からの長さの列を示す。定式化すると以下のようになる。

$Length(route, d, e)$：経路 $route$ の中で、d から e に行くためにかかるコスト。ただし、d より先に e に訪れる場合、無限大になる。
$recent = \{Length(route_i, route_j) \quad | \quad i < j \wedge Length(route_i, route_j) \leq certain\_len\}$ ただし、$route_j = next$

NextRecent(cur, next,recent)) は上記に述べた recent を次の recent に更新する関数である。

```
class Route:
    Route(route, v):
        this.route = route
        this.v = v

#外からはFindRoute(  cur, dst, Node, Array()  )のように呼び出す
FindRoute(cur, dst, notUsedNodes, recent):
    res = Route(Array(), 0)
    for next in notVisited:
        route = FindRoute(  next, dst, notUsedNodes - cur,
                            NextRecent(cur, next, recent)    )
        res.v += route[0].v * Weight(recent.size())
                 - Cost(route[0], cur)
        if (route.v > res.v):
             res = route
             res.route += Array(cur)


    res2 = Route(cur, -Cost(cur, dst) + M) #Mは十分に大きな数で、
目的地にたどり着いたかどうか。route.v - Mで評価値が得られる
    if (res.v > res2.v): return res
    else: return res2
```

図 3.5: 厳密解を求める擬似コード

3.6　近似解法

厳密解法ではサービス数が大きくなると現実的な時間で問題を解けなくなるため、近似解法を提案する。

以下の近似解法での初期経路は、出発地と目的地を結んだだけのサービスを含まない経路とし、この初期経路を simple と呼ぶ。

3.6.1　近似解法：挿入法

巡回セールスマン問題の挿入法に似た解法を提案する。以下にアルゴリズムを示す。

1. 初期経路は前述の通り、simple(出発地と目的地を結んだだけのサービスを含まない経路) とする。
2. 現在の経路のすべての辺に対し、各サービスそれぞれを挿入する。評価値の最も高いものを best とする。
3. best が、挿入を行う前より評価値が高ければ、挿入して 2 に戻る。そうでなければ、終了する。

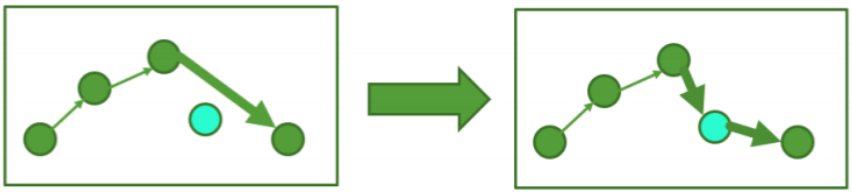

図 3.6: 挿入の例

3.6.2　近似解法：焼きなまし法 [3]

焼きなまし法とは近似解法アルゴリズムの一つで、局所解に陥りづらくするために、現在の状態より悪い状態にも一定の確率で遷移させるアルゴリズムである。焼きなまし法のアルゴリズムは以下の通りである。

焼きなまし法

1. あらかじめ、計算する時間を T を決める。
2. 以下のスコープ内のコードを時間 T 経過するまで繰り返す。
   (a) 一定のルールを定め、そのルールに基づき、次の状態（近傍）を選ぶ。
   (b) 次の状態のスコアが、今の状態のスコアより良くなった場合は、常に、状態を次の状態にする（遷移）。
   (c) 確率は以下のように決める。
      i. 序盤 (時刻 t=0 に近いとき) ほど高確率で、終盤 (t=T に近いとき) ほど低確率。
      ii. スコアが悪くなった量が大きければ大きいほど、低確率。
3. 最後に、今まで一番スコアのよかった状態を選ぶ。

3.6.3　近似解法：焼きなまし法 [3] の本研究での適用

この研究では、利用順を状態とする。初期状態は挿入法と同じく、simple とする。そして次の状態を「挿入」または「削除」を 1 回行った経路とする。「挿入」、「削除」の詳細は以下の通りである。

- 挿入：現在の経路に無いランダムなサービスを選び、現在の経路の辺に挿入する。挿入する場所は最も挿入後の経路が短くなるようにする。
- 削除：現在の経路のランダムなサービスを削除する。

挿入、削除の例は次ページの図で示す。

挿入、削除の例

以下の図で挿入、削除の例を示す。ただし、点はサービスを表す。

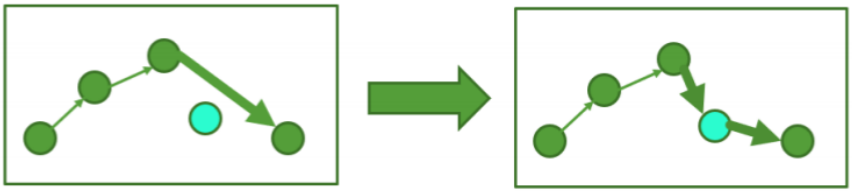

図 3.7: 挿入の例

図 3.8: 削除の例

第4章　実装

解法の評価と解法が制約を満たしているか検証するために、Visual Studio が提供している C#の Windows フォームアプリケーションでプログラムを作成した。

コンソールアプリケーションでなく、ビジュアライズできるフレームワークを利用した理由は 2 つある。 1 つ目は制約を満たしているか目視で確認するためである。もう一つは提示された経路を少し変更した経路の評価値を測るなどの行為が容易に行えるプログラムを書けるためである。次ページに実行画面の説明をし、プログラムの実行画面は 17 ページ以降に示す。

実行画面の説明

以下に実行画面の説明を示す。

- 点は出発点、目的地、またはサービスである
- 点の大きさ：価値を示す（ただし、出発地と目的地は価値を持たないので、常に同じ大きさ）
- 「num」：Node(出発地と目的地とサービスの和集合) の要素数
- 「Score」　：評価値
- best のボタンは num が 8 以下のものしか有効になっていないが、これは、それ以上のデータで厳密解を求めようとすると、膨大な時間がかかってしまうため
- 点の色の意味は次のとおりである。
  - 茶：経路に含まれないサービス
  - 黒：経路に含まれるサービス
  - 赤：出発地
  - 青：目的地

4.1　実行画面の例

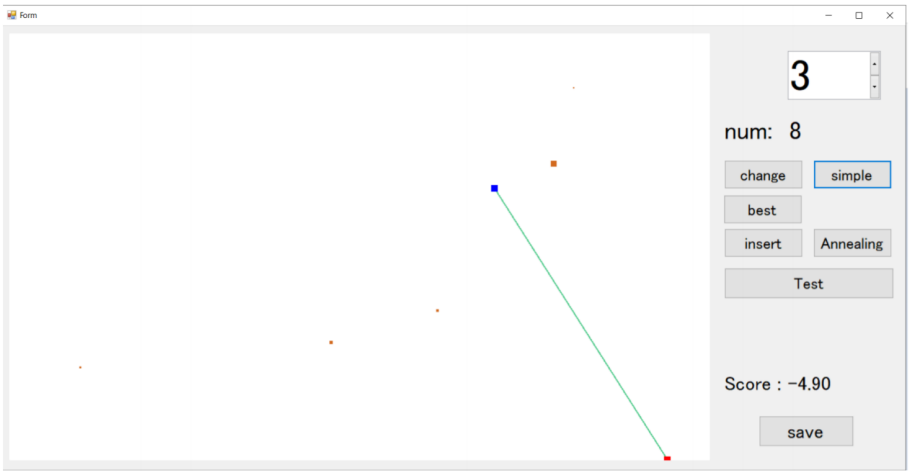

図 4.1: プログラム実行画面 1

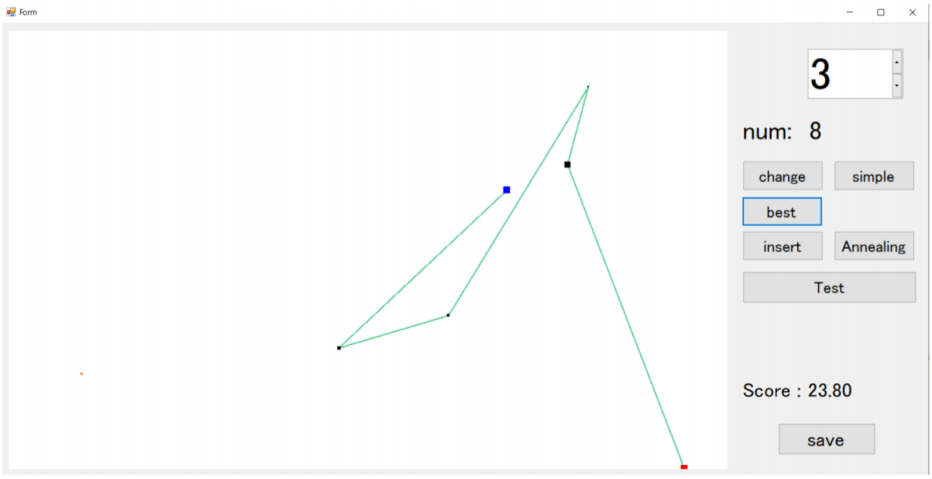

図 4.2: プログラム実行画面 2

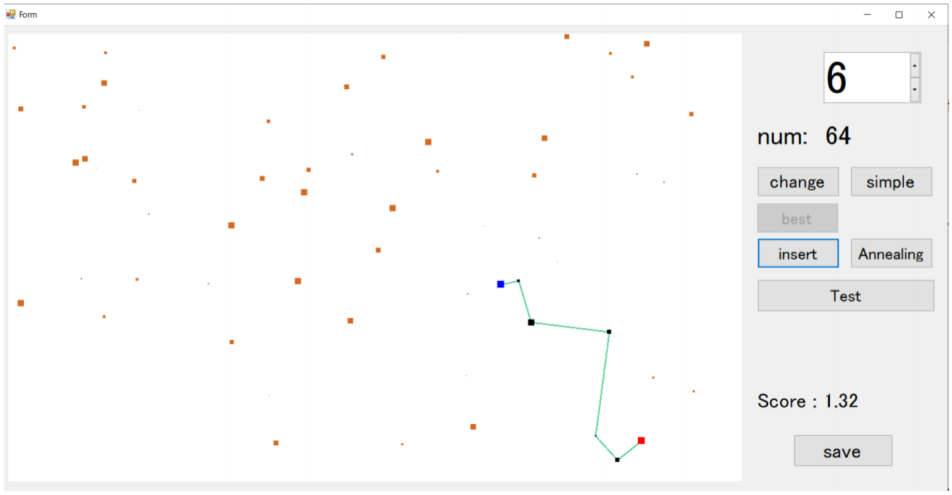

図 4.3: プログラム実行画面 3

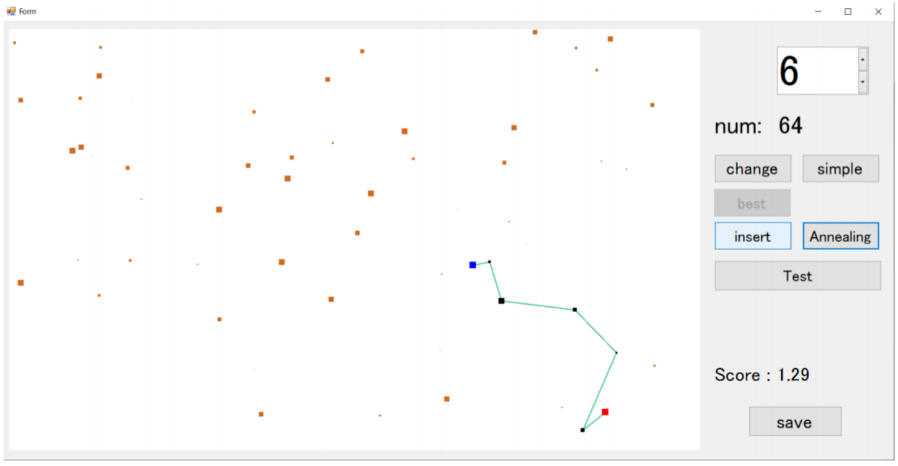

図 4.4: プログラム実行画面 4

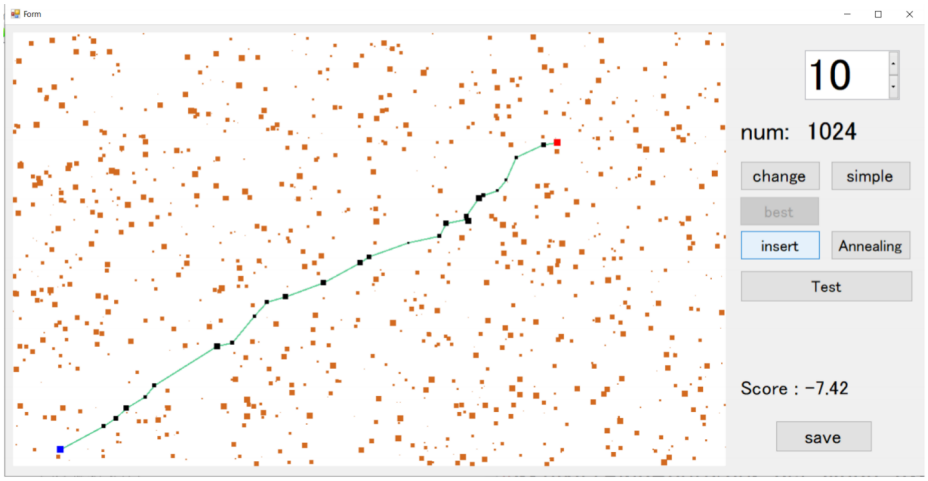

図 4.5: プログラム実行画面 5

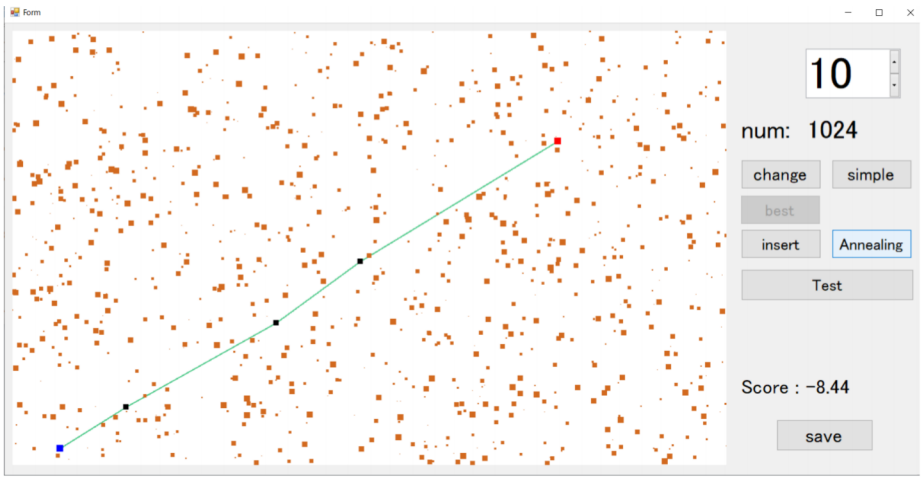

図 4.6: プログラム実行画面 6

第5章　解法の評価

5.1　解法の評価方法

挿入法の平均実行時間と各解法の比較をして評価する。挿入法の平均実行時間を測定するのは、焼きなまし法と違い、時間制約がなく、どの程度の速さで解けるのか検証する必要があるためである。

各解法の比較は、評価値の増加量の比較で行う。各解法の評価値の増加量を計算し、解法同士の比較を行うことにより解法の評価を行う。評価値の増加量とは (各解法で求めた経路の評価値 - simple の評価値) である。増加量で比較している理由は評価値が負の値になる場合があるためである。また、このデータの中の増加量の平均については正の値であることを確認している。

5.1.1　解法の評価方法の詳細

出発点、到着点、サービスの位置、サービスの価値は一様乱数により決定している。全てのサービス数に対して焼きなまし法の実行時間は 1 秒である。$w'(s,i) = 2^{-}a$、a は一定時間内に利用した同じ種類のサービス数である。実行環境の CPU は Intel(R) Core(TM) i7-8700 CPU @ 3.20GHz (passmark: 15137) であり、VRAM は 8GB である。また、各サービス数ごとに 2000 個のデータを用意し、実験したデータを下に示す。

以下のデータを下の表に示す。

- 挿入法の平均実行時間
- 近似解法と厳密解法の比較 (厳密解法の評価値の増加量を 100%として、近似解法の増加量を比較)
- 近似解法の比較 (挿入法の増加量を 100%として、焼きなまし法の増加量を比較)

5.2 実験結果

実験結果を以下に示す。ただし、実験結果で示す数値は全て小数点以下は四捨五入した結果を示す。

表 5.1: 挿入法の平均実行時間

サービス数	実行時間
8	0ms
16	0ms
32	0ms
64	2ms
128	9ms
256	32ms
512	109ms
1024	371ms

表 5.2: 近似解法の増加量の平均/厳密解法の増加量の平均

サービス数	挿入法	焼きなまし法
8	99%	97%

表 5.3: 焼きなまし法の増加量の平均/挿入法の増加量の平均

サービス数	焼きなまし法
8	98%
16	109%
32	132%
64	113%
128	110%
256	76%
512	51%
1024	38%

5.3 考察

5.3.1 挿入法の平均実行時間

挿入法の平均実行時間を見ると、サービス数が 1024 以下であれば実用的な時間で探索できることが分かる。

5.3.2 厳密解法との比較

サービス数が少ない時は非常に高い精度が出ている。

5.3.3 近似解法の比較

サービス数が 8〜32 の時、データ数が少ない時の方が、焼きなまし法の増加量が低くなっている。これは、焼きなまし法の挿入が原因である。焼きなまし法の挿入は

- 現在の経路に無いランダムなサービスを選び、現在の経路の辺に挿入する。挿入する場所は最も挿入後の経路が短くなるようにする。

であり、挿入場所が制限されている。そのため、焼きなまし法で探索できない、または探索しづらい経路がある。そのため、データ数が少ない時の方が、焼きなまし法の増加量が低くなっている。

サービス数が 32 以上の時、サービス数が少ないものでは焼きなまし法が有効で、データ数が大きくなると挿入法の方が評価値が高い。これはサービスが多くなると、焼きなまし法が探索する経路数に比べ、総経路数が非常に多くなるためだと考えられる。

第6章　結論

サービスの利用頻度と価値を考慮するために、定式化と解法を作成した。

本研究では、w を単調減少関数としたが、増加関数にすると、連続で行きたいが、単発では行きたくないようなサービスの経路探索などの応用も考えられる。

謝辞

日頃からご指導して頂いた山田俊行講師, 河内亮周教授, 森本尚之講師, 多くの助言や意見をくださったコンピュータソフトウェア研究室の皆様, 研究活動における様々な場面でお世話になりました落合美子事務員に感謝いたします.

参考文献

[1] Michael R. Garey / David S. Johnson COMPUTER AND INTRACTABILITY A Guide to the Theory of NP-Complete p211 - p214, March 1991

[2] 松田善臣, et al. ”最適観光経路問題とその解法.” 電気学会論文誌 C (電子・情報・システム部門誌) 124.7 (2004): 1507-1514.

[3] S．Kirkpatrick，C．D．GelattJr．and M．P．Vecchi：Optimizationby Simulated Annealing， Science，Vol．220 pp，671 − 680（1983）