曲芸飛行 - Aerobatics

JOI 2021 春合宿 Day1 解説

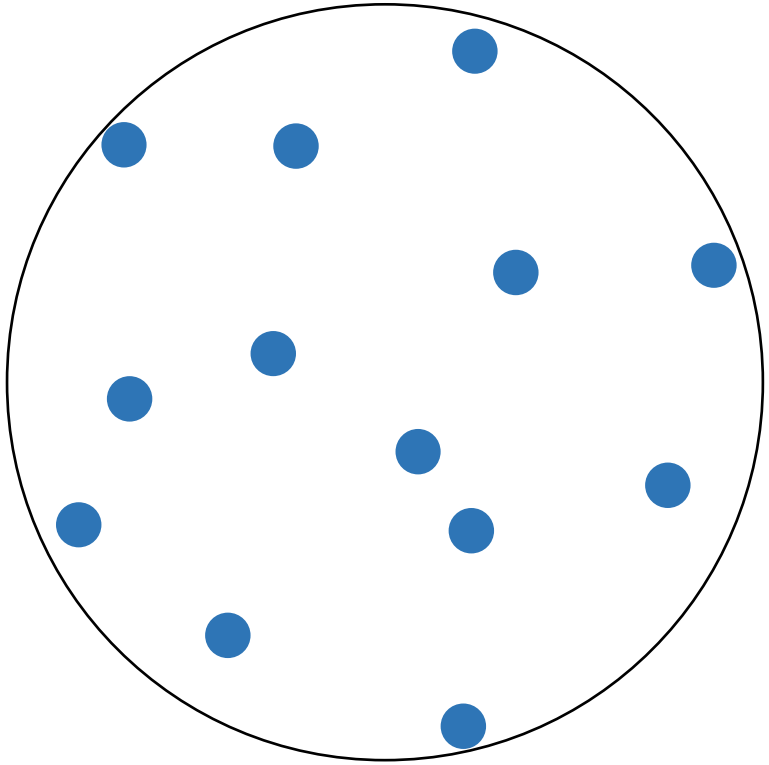
解説担当 ー 米田 寛峻（よねだ ひろたか）

2021. 03. 20

square1001

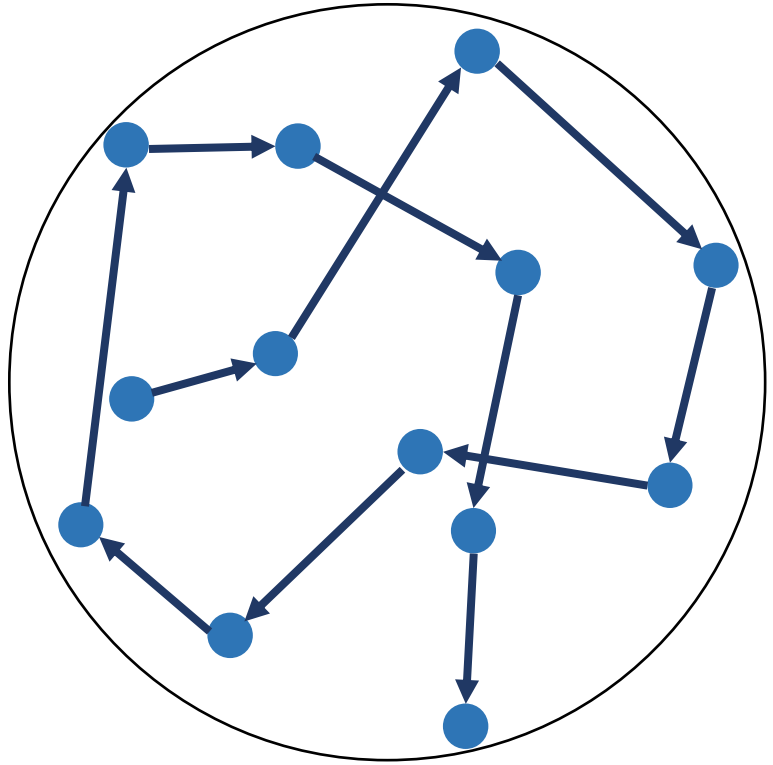
問題概要（1）

A



平面上に N 個の
チェックポイントがある

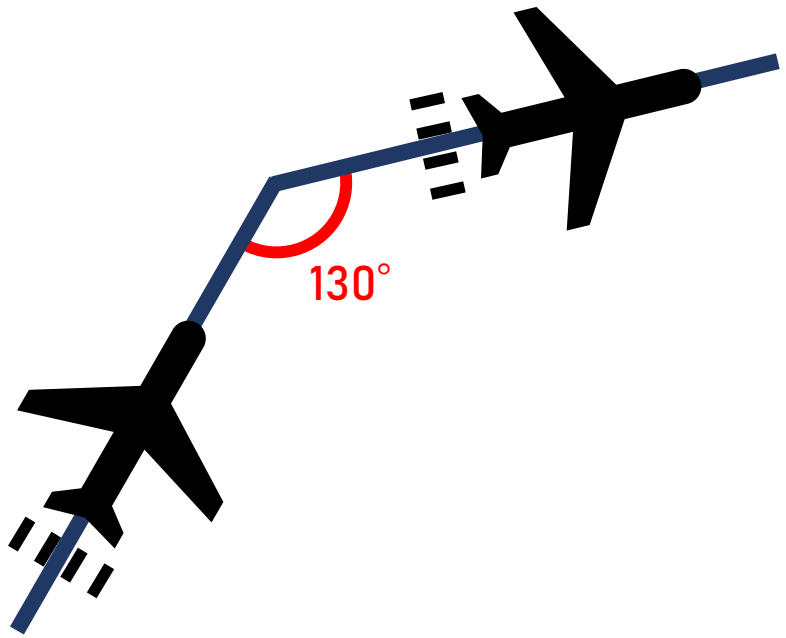
B



これらを 1 回ずつ通る
経路を見つける

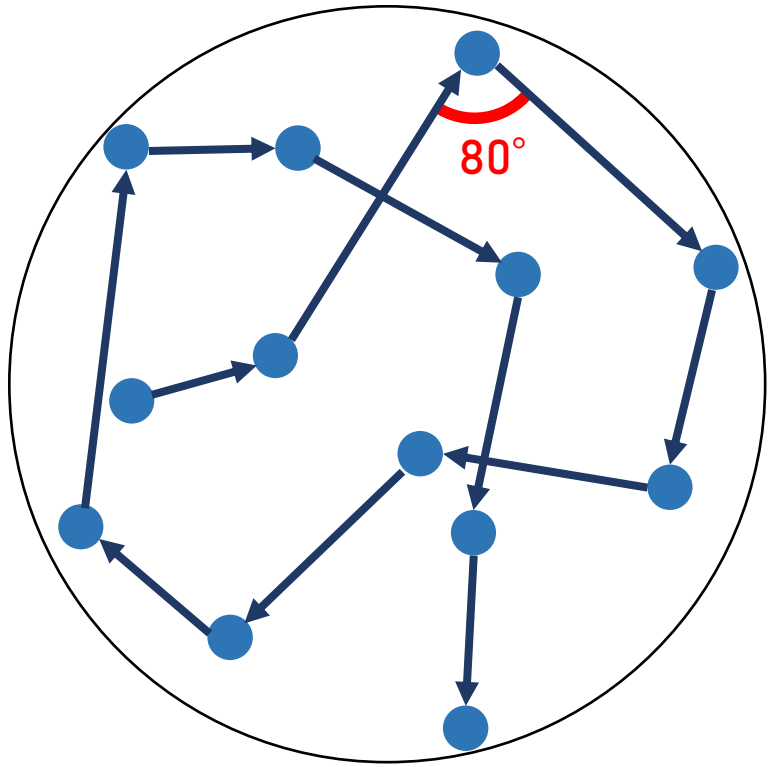
問題概要（2）

C



飛行機はあまり方向転換したくない
すなわち折れ線の角度を大きくしたい

D



経路の最小の角を可能な限り大きくし
曲芸飛行の成功率を上げよう！

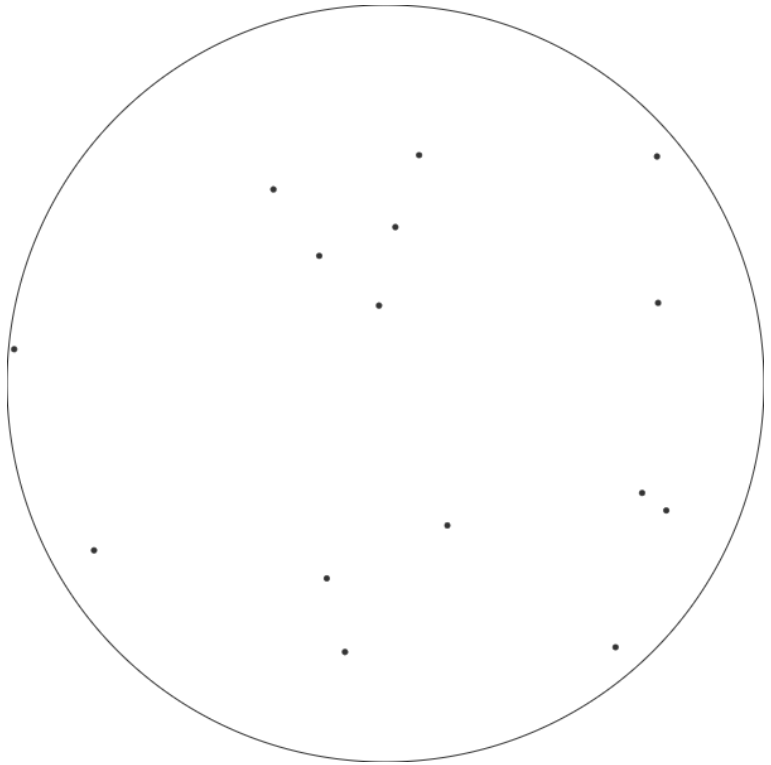
問題概要（3）

本課題は 6 個の入力データからなる
→ それぞれのデータに対して手元で実行し、出力結果を提出

入力データ	N の値	満点のスコア	半分のスコア
01.txt	15	100.000 度	68.867 度
02.txt	200	143.000 度	110.954 度
03.txt	200	134.000 度	102.338 度
04.txt	1000	156.000 度	123.182 度
05.txt	1000	150.000 度	117.569 度
06.txt	1000	153.000 度	120.382 度

入力データ１・２

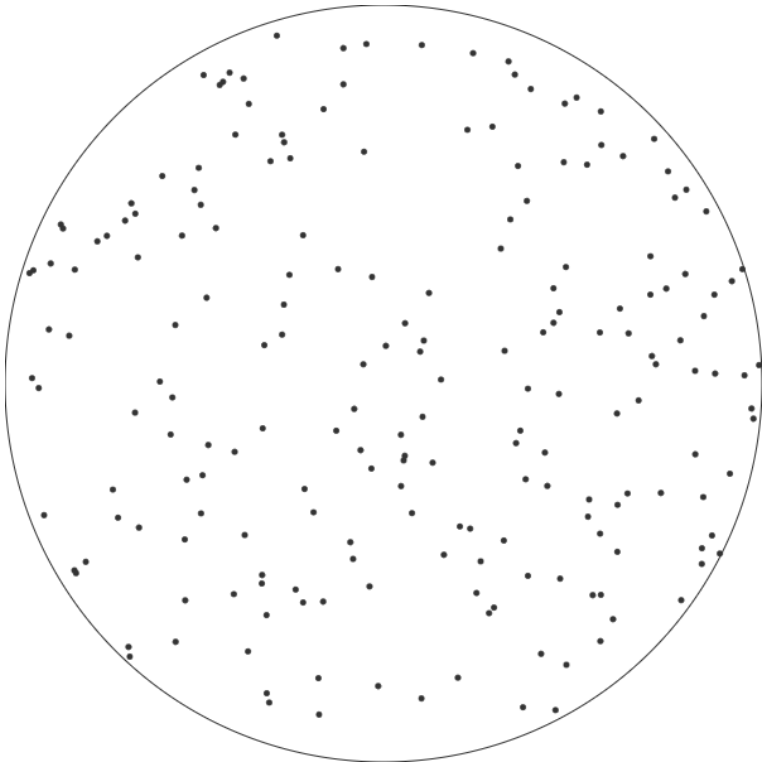
1



N = 15 ／ Z_0 = 100°

（ランダムな感じの点の分布）

2

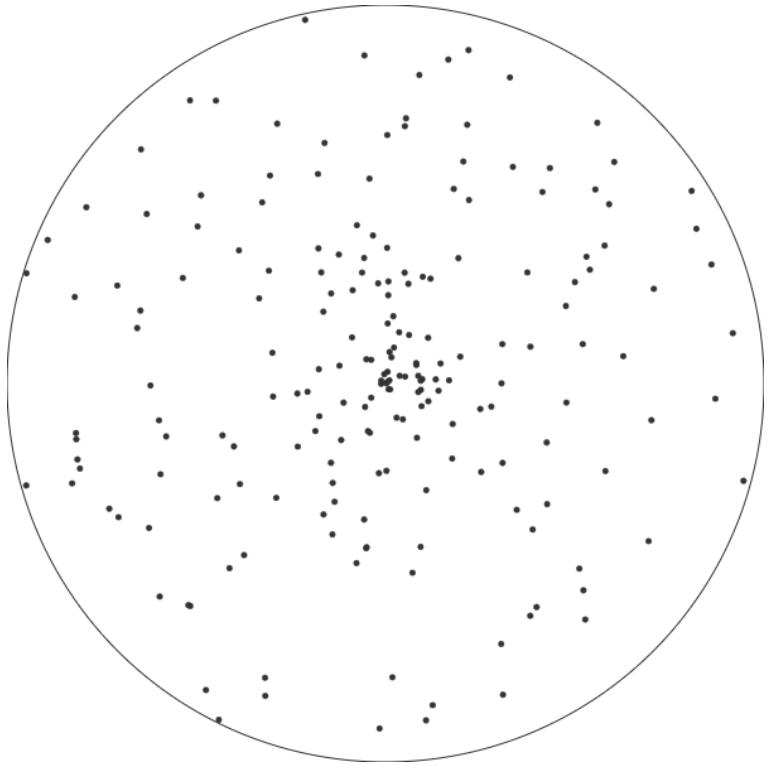


N = 200 ／ Z_0 = 143°

（ランダムな感じの点の分布）

入力データ３・４

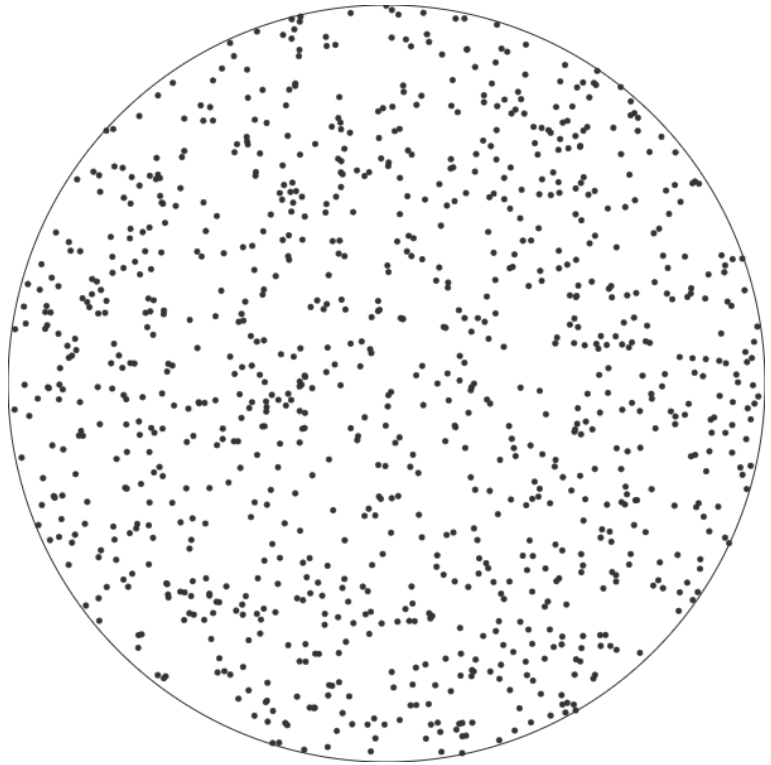
3



N = 200 ／ Z_0 = 134°

（中心のほうが点の密度が高い分布）

4

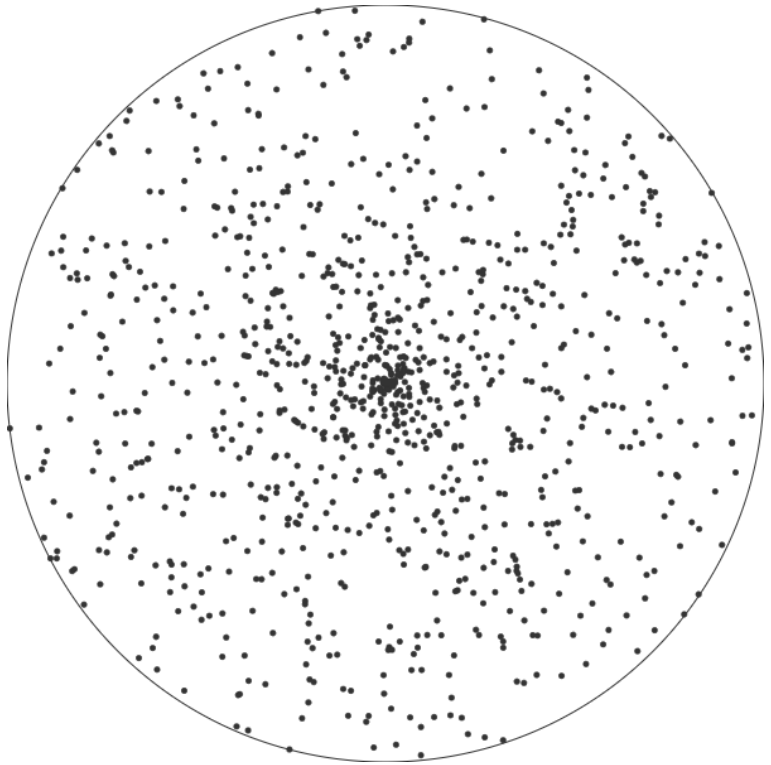


N = 1000 ／ Z_0 = 156°

（ランダムな感じの点の分布）

入力データ 5・6

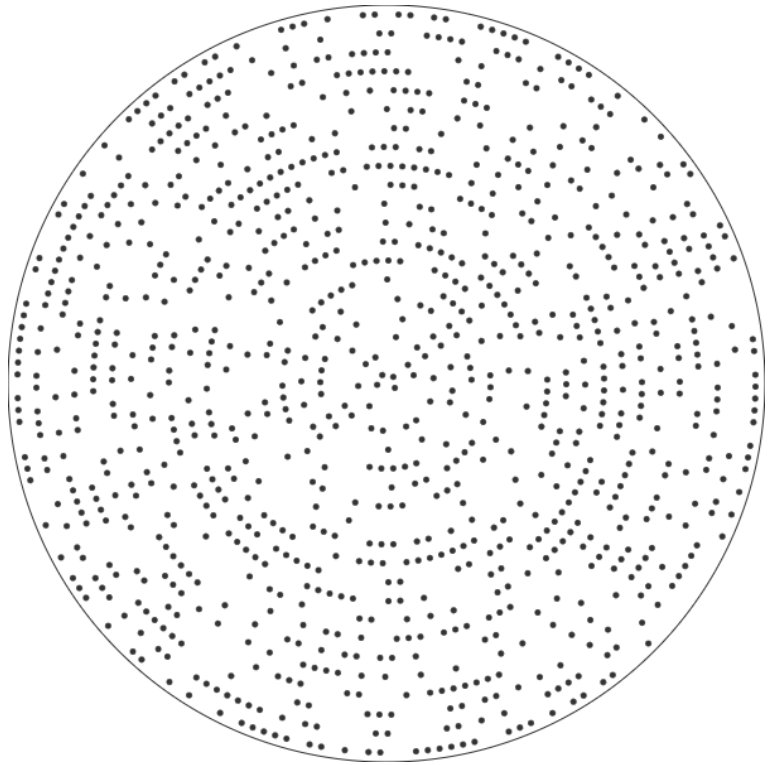
5



N = 1000 ／ Z_0 = 150°

（中心のほうが点の密度が高い分布）

6



N = 1000 ／ Z_0 = 153°

（特殊なタイプのデータ）

Output Only 形式について

課題の形式

- 入力データが公開されており
  これに対する出力を提出しなければならない
- より良い答えを出せば高得点を取れる
- 方針によって大きく点数が変わることが多い
- 入力データの性質をつかむことが必要な場合も

過去の出題

JOI 春合宿における出題

- 2020 年「Legendary Dango Maker」
- 2018 年「Road Service」

IOI における出題

- 2019 年「Broken Line」
- 2017 年「Nowruz」

Output Only 形式について

課題の形式

- 入力データが公開されており
  これに対する出力を提出しなければならない
- より良い答えを出せば、高得点を取れる
- 方針によって大きく点数が変わることが多い
- 入力データの性質をつかむことが必要な場合も

過去の出題

JOI 春合宿における出題

- 2020 年「Legendary Dango Maker」
- 2018 年「Road Service」

JOI 本選における出題

- 2019 年「Broken Line」
- 2017 年「Nowruz」

出題数はあまり多くないが
差が付きやすい問題になりうる

入力データ 1 に関して

入力データ 1 は「N = 15」と小さいので…

入力データ 1 に関して

入力データ 1 は「N = 15」と小さいので…

ビット DP

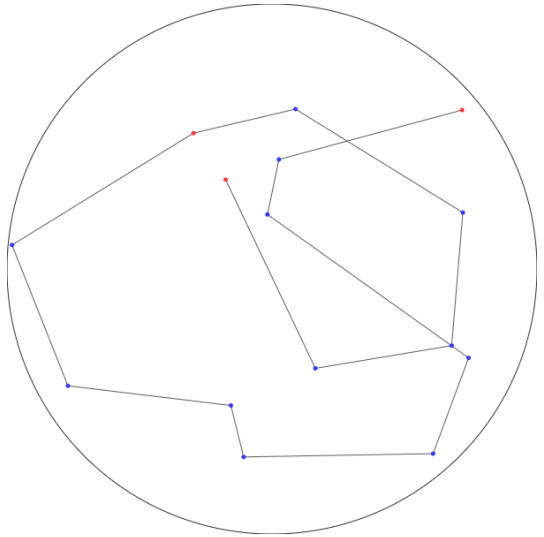
が使えます！

入力データ 1 に関して

ビット DP で解いてみる

- dp[既に訪れた場所の集合][今いる場所][直前にいた場所]
  ＝（作った最小の角の最大値）を記録
- すると、計算時間 $O(2^N \times N^3)$・メモリ $O(2^N \times N^2)$ で解ける
- したがって、$N = 15$ のケースだと
  数秒で最適解のうちひとつが求まる

Tip：配布ライブラリも利用できる！



入力データ 1 の最適解（約 100.008°）

10 点獲得

方針 1 - ランダムに経路を生成

最も簡単な方針：ランダムに経路を生成する

解法 A

チェックポイント 1 → 2 → 3 → … → N の順番に通る

解法 B

チェックポイントを通る順番をランダム生成
これを 1,000,000 回行い、スコアが最大のものに決める

方針 1 - ランダムに経路を生成

実験結果

入力データ	解法 A	解法 B
01.txt	1.540 度	80.517 度
02.txt	0.286 度	7.356 度
03.txt	0.139 度	5.354 度
04.txt	0.010 度	1.599 度
05.txt	0.048 度	1.069 度
06.txt	0.042 度	1.334 度
得点	0 点	7 点

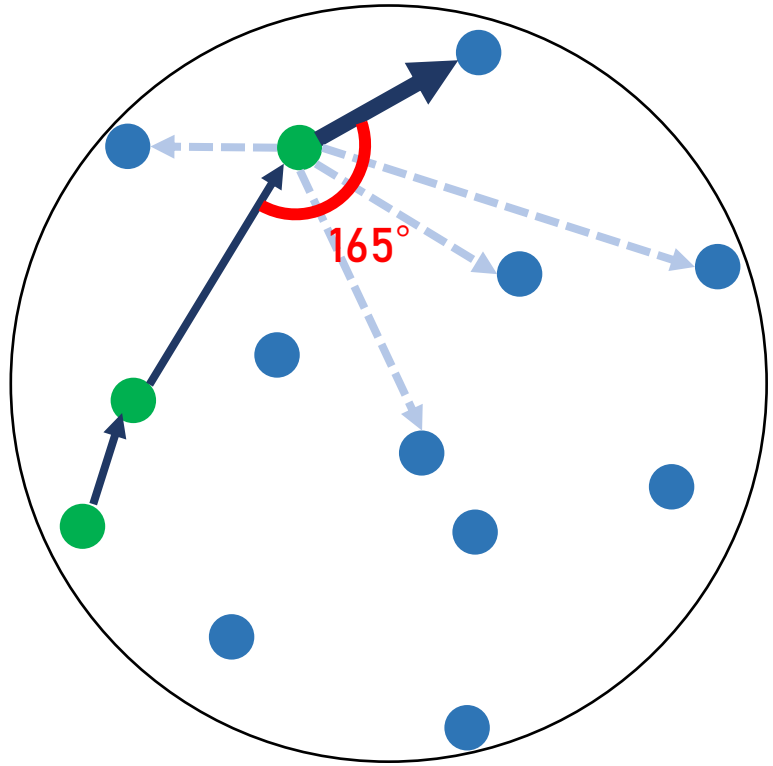
※ 注意：解法 B については、ひとつの実験結果にすぎません。

7 点獲得

方針 2 – 貪欲法

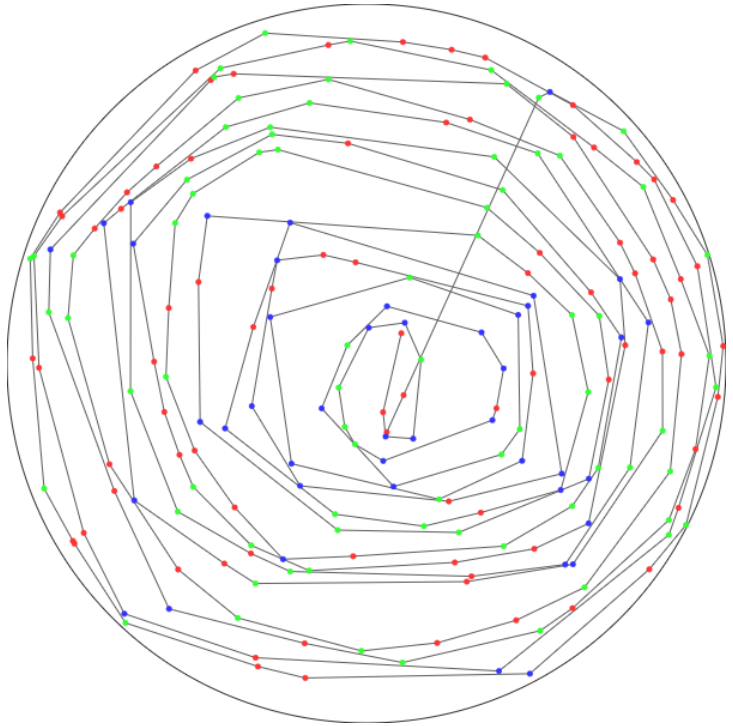
貪欲法を使って解く

- 最初に通る 2 つのチェックポイントを決める
- 3 番目以降のチェックポイントを決めるときに
  残った場所の中で「角が最大となるもの」を選ぶ

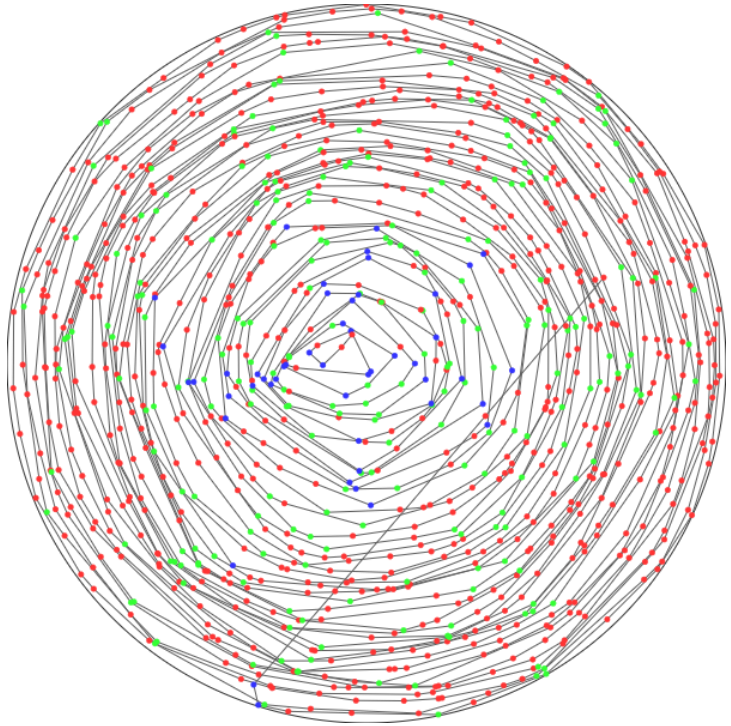


方針 2 – 貪欲法

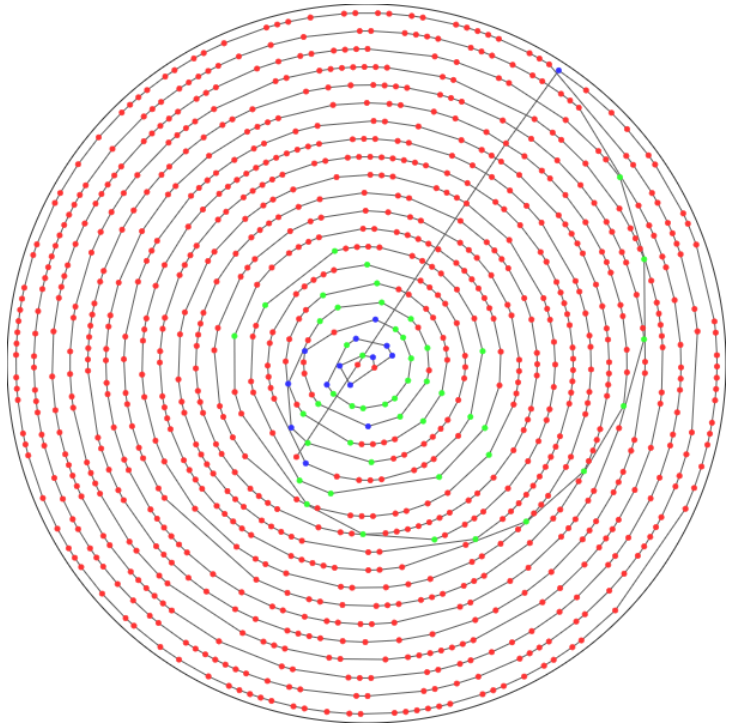
得られる答えは、らせん状になる



入力データ 2 の貪欲解



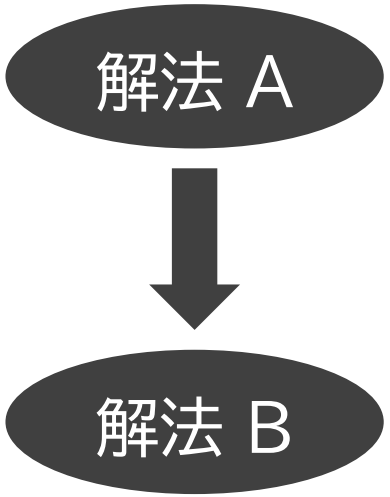
入力データ 4 の貪欲解



入力データ 6 の貪欲解

方針 2 – 貪欲法

貪欲法を使って解く



X 座標が最大の頂点をスタート地点
2 番目の地点はここから最も上に近い方向に進むように選び
3 番目以降は貪欲法で答えを求める

スタート地点・2 番目の地点を全探索し
3 番目以降は貪欲法で答えを求める

※ 解法 B の計算時間は $O(N^4)$ なので
入力データ 4, 5, 6 では実行に 20 分くらいかかる

方針 2 – 貪欲法

実験結果

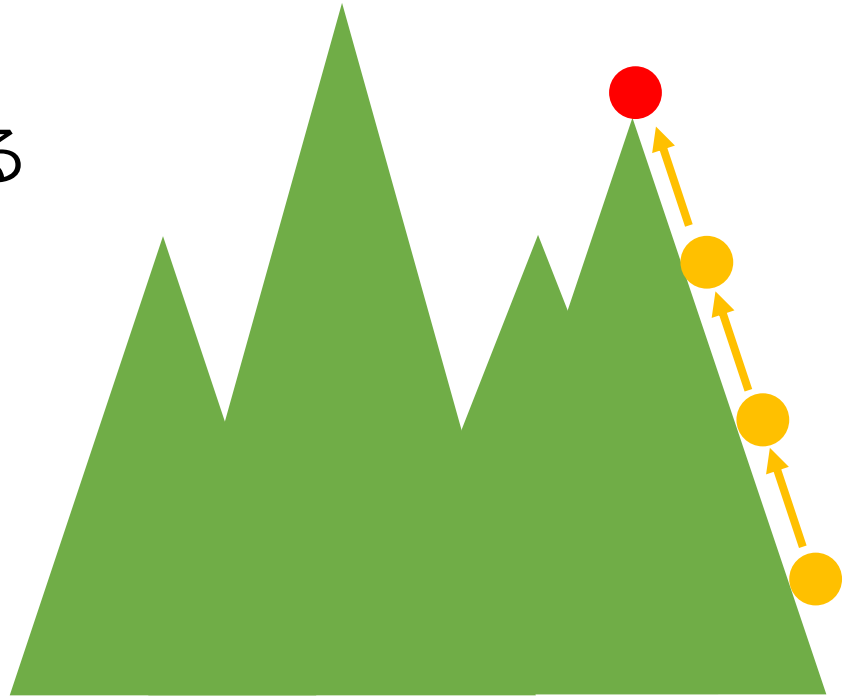
入力データ	解法 A	解法 B
01.txt	67.638 度	91.822 度
02.txt	1.021 度	91.327 度
03.txt	47.791 度	58.807 度
04.txt	40.454 度	79.341 度
05.txt	33.773 度	78.051 度
06.txt	83.346 度	84.000 度
得点	14 点	28 点

28 点獲得

方針 3 – 山登り法 ①

山登り法とは？

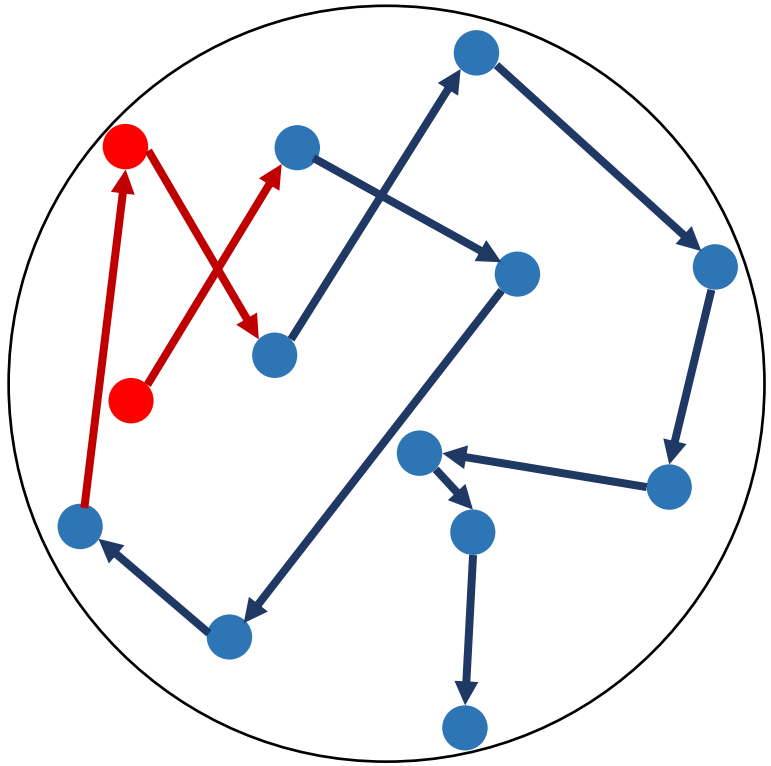
- スコアが改善できるところを見つけて、これを改善する
- これを「改善できなくなるまで」繰り返す
  上手くやれば、かなり良い解が得られることが多い！



山登り法のイメージ
局所的改善を繰り返して、頂上に到達する

方針 3 – 山登り法 ①

山登り法のイメージ

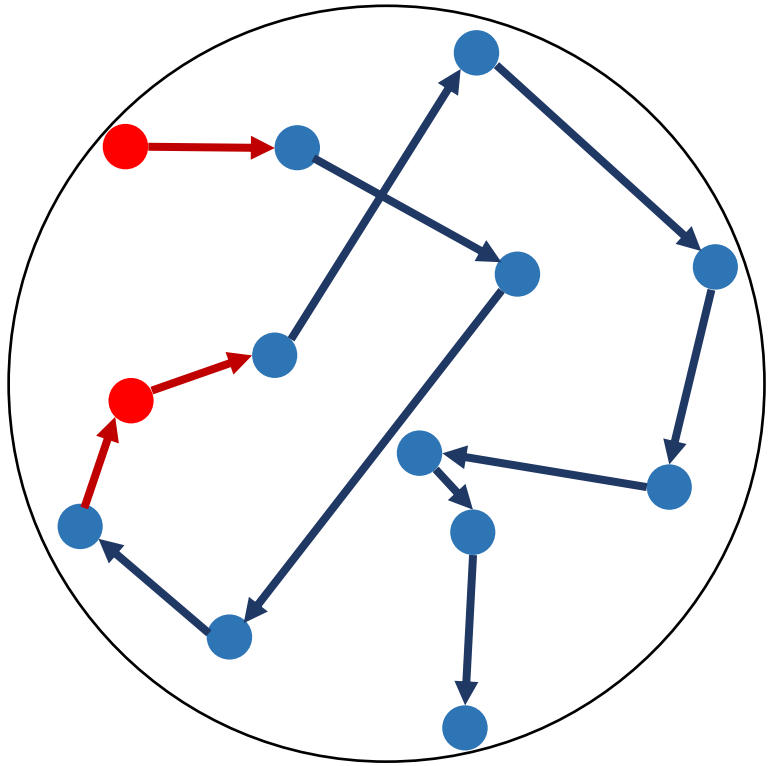


Step 1 - 改善できるところを見つける

- 2 つの赤い頂点の順番を交換できそう！

方針 3 – 山登り法 ①

山登り法のイメージ

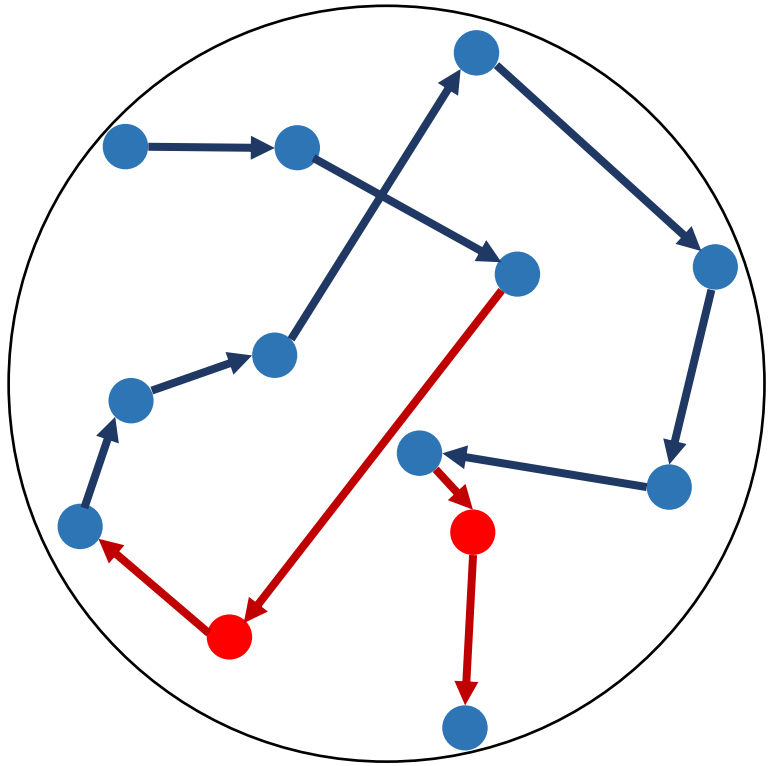


Step 1 - 改善できるところを見つける

- 2 つの赤い頂点の順番を交換できそう！
- スコアが改善するので、順番を交換する

方針 3 – 山登り法 ①

山登り法のイメージ



Step 1 - 改善できるところを見つける

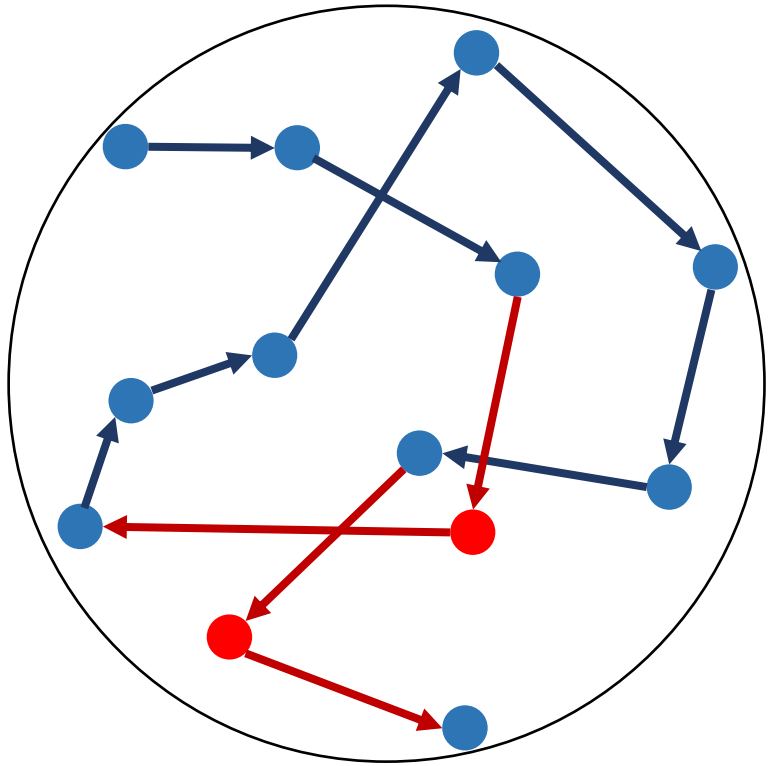
- 2 つの赤い頂点の順番を交換できそう！
- スコアが改善するので、順番を交換する

Step 2 - 改善できるところを見つける

- 2 つの赤い頂点の順番を交換できそう！

方針 3 – 山登り法 ①

山登り法のイメージ



Step 1 - 改善できるところを見つける

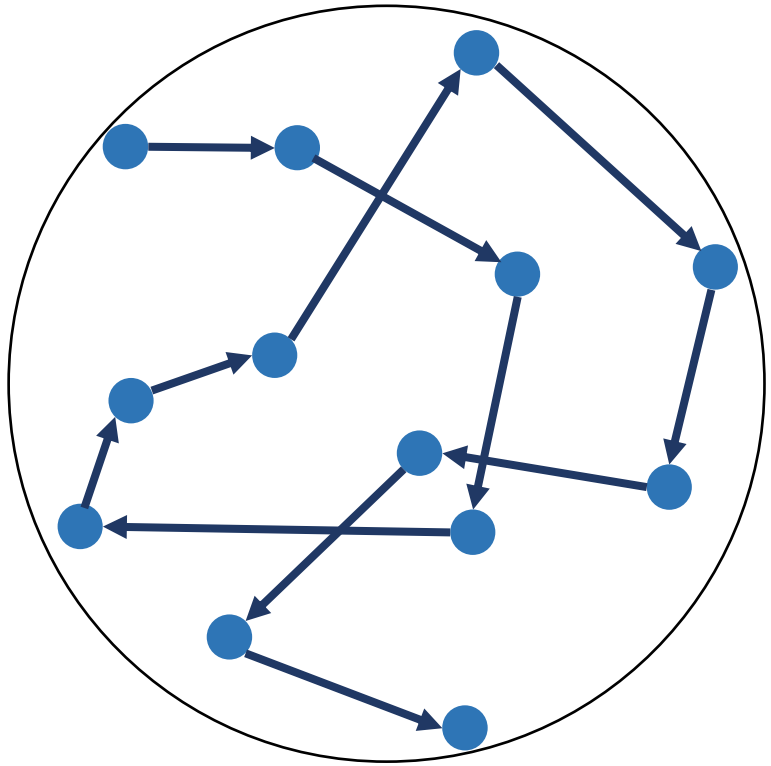
- 2 つの赤い頂点の順番を交換できそう！
- スコアが改善するので、順番を交換する

Step 2 - 改善できるところを見つける

- 2 つの赤い頂点の順番を交換できそう！
- スコアが改善するので、順番を交換する

方針 3 – 山登り法 ①

山登り法のイメージ



Step 1 - 改善できるところを見つける

- 2 つの赤い頂点の順番を交換できそう！
- スコアが改善するので、順番を交換する

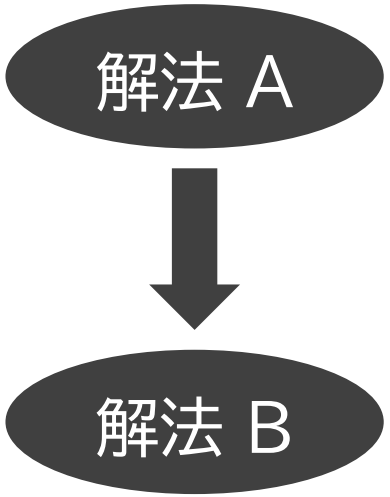
Step 2 - 改善できるところを見つける

- 2 つの赤い頂点の順番を交換できそう！
- スコアが改善するので、順番を交換する

Step 3 – これ以上スコアを改善できないので
探索を終了する

方針 3 – 山登り法 ①

山登り法を使って解く



最もスコアが改善できる「2 つの順序の交換」を適用し続ける
改善する方法がひとつもなくなれば、探索を打ち切る

2 つの場所をランダムに選び、スコアが改善すれば交換する
ただし、解の自由度を上げるため、スコアが同じでも交換を適用
$5N^2$ 回以上スコアが更新されなければ、探索を打ち切る

※ ただし初期解は 1 → 2 → … → N の順番とする

方針 3 – 山登り法 ①

実験結果

入力データ	解法 A	解法 B
01.txt	56.609 度	70.803 度
02.txt	57.037 度	102.910 度
03.txt	68.013 度	101.877 度
04.txt	64.315 度	118.428 度
05.txt	66.832 度	112.167 度
06.txt	71.712 度	111.941 度
得点	19 点	45 点

※ 注意：解法 B については、ひとつの実験結果にすぎません。

45 点獲得

方針 4 – 山登り法 ②

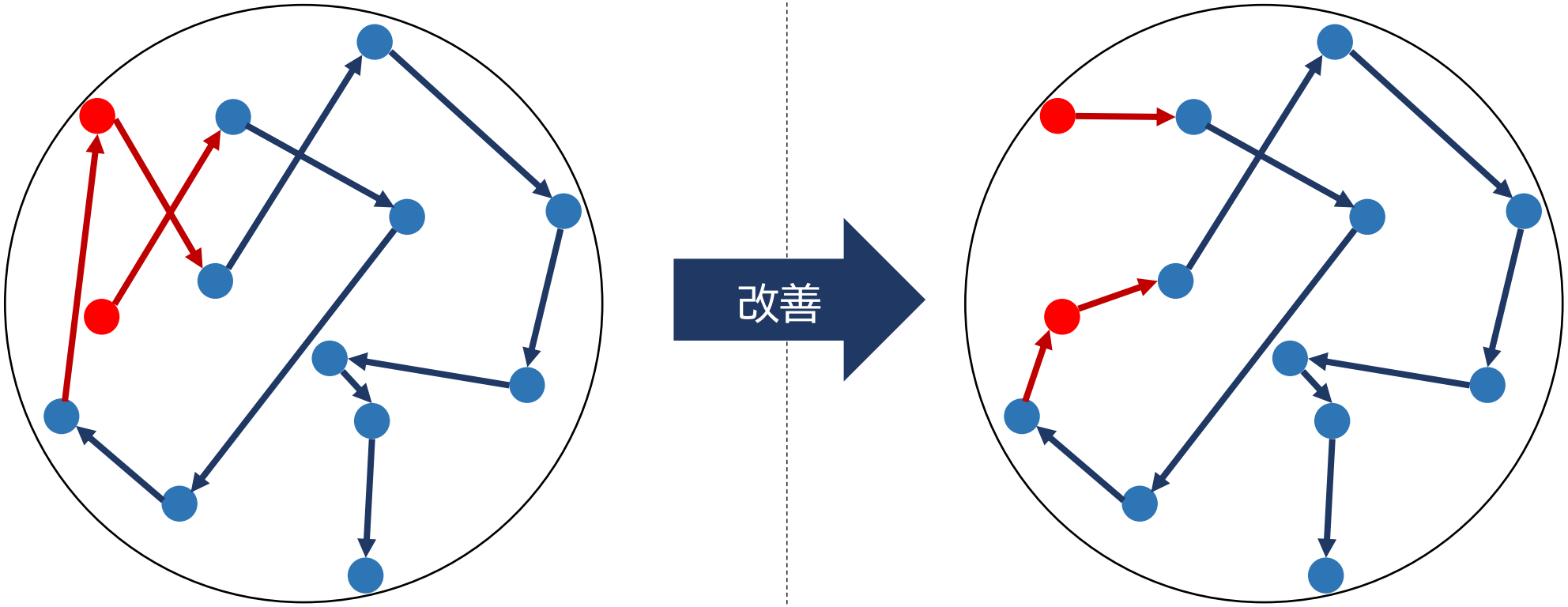
「改善」の方法を変えるということ

- 元々は 1 回の変化を「2 つの場所の訪れる順番を入れ替える」としていたが…
- これを上手く変えれば、”局所的最適解” にハマりにくくなるのではないか？

方針 4 – 山登り法 ②

元々の「1 回の変化」は

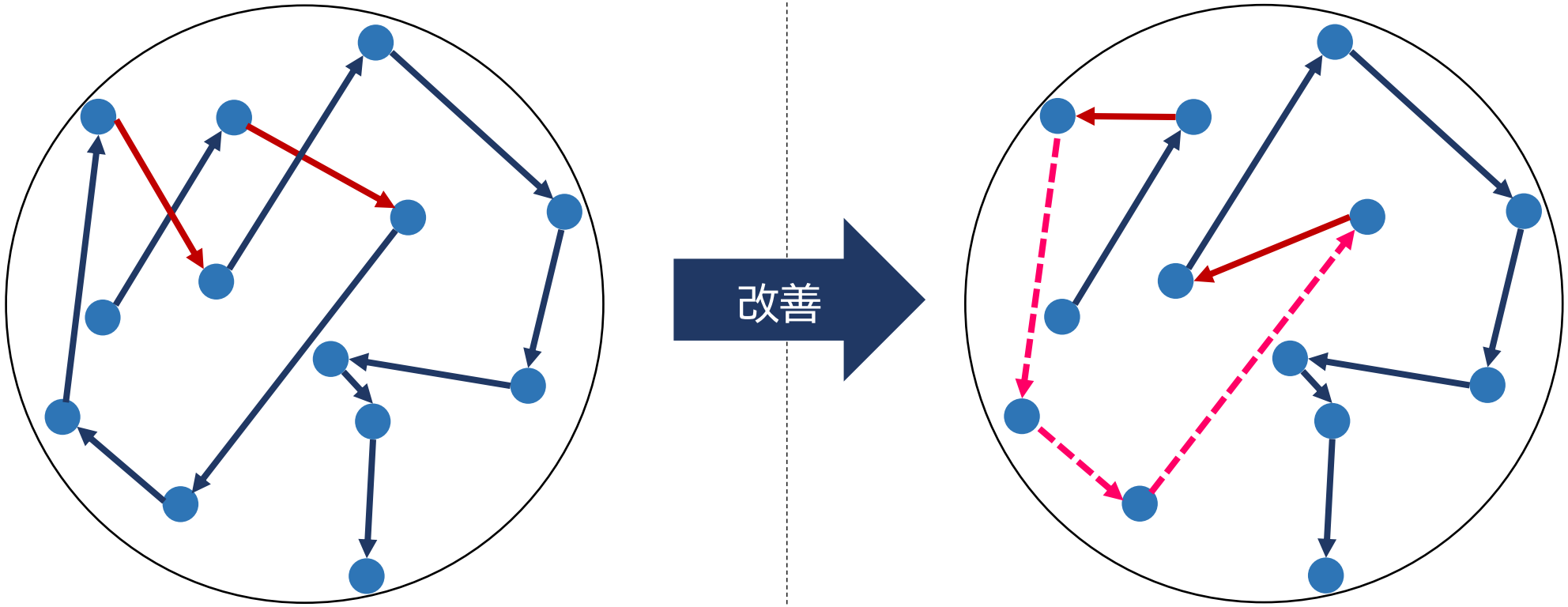
- ☀ 2 つの頂点を選び、これを訪れる順番を入れ替えること



方針 4 – 山登り法 ②

これをもっと良いものにしよう！（Part 1）

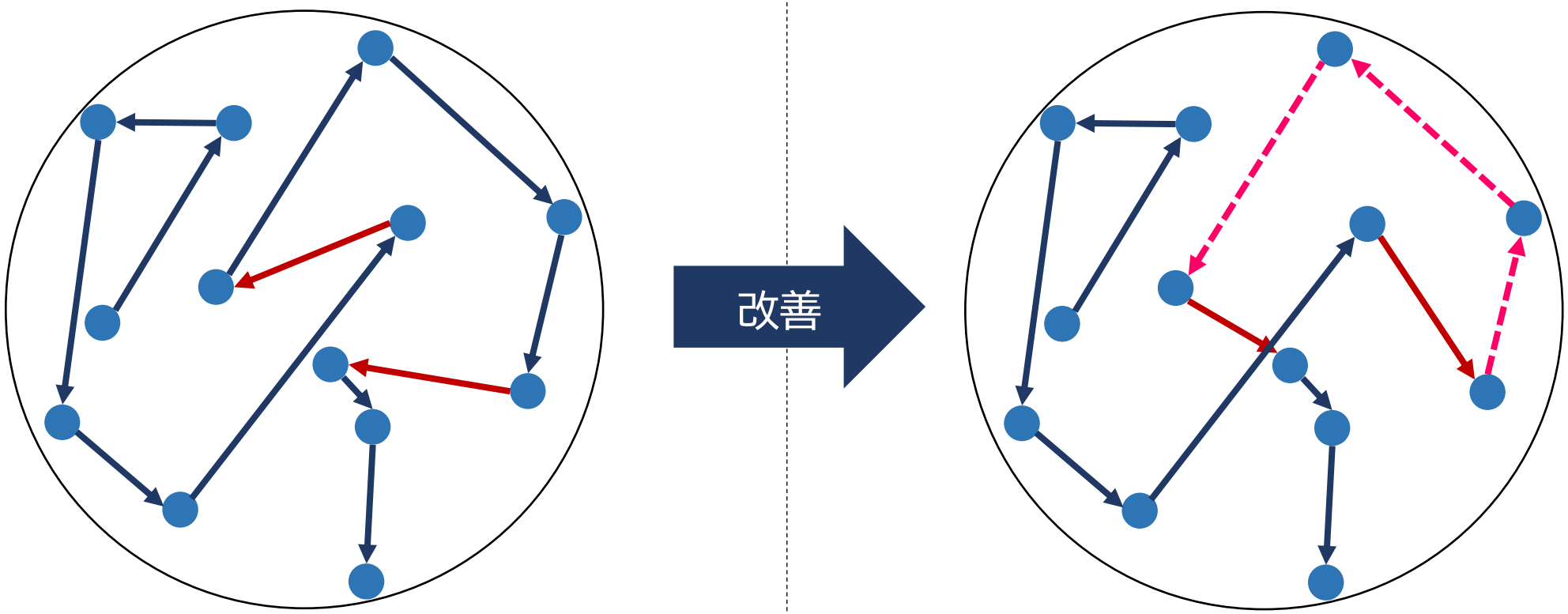
☀ 2 つの辺を選び、これを訪れる順番を入れ替えること



方針 4 – 山登り法 ②

これをもっと良いものにしよう！

☀ 2 つの辺を選び、これを訪れる順番を入れ替えること



方針 4 – 山登り法 ②

1 回の変化で「2 つの頂点を swap する」

VS.

1 回の変化で「2 つの辺を swap する」

方針 4 – 山登り法 ②

1 回の変化で「2 つの頂点を swap する」

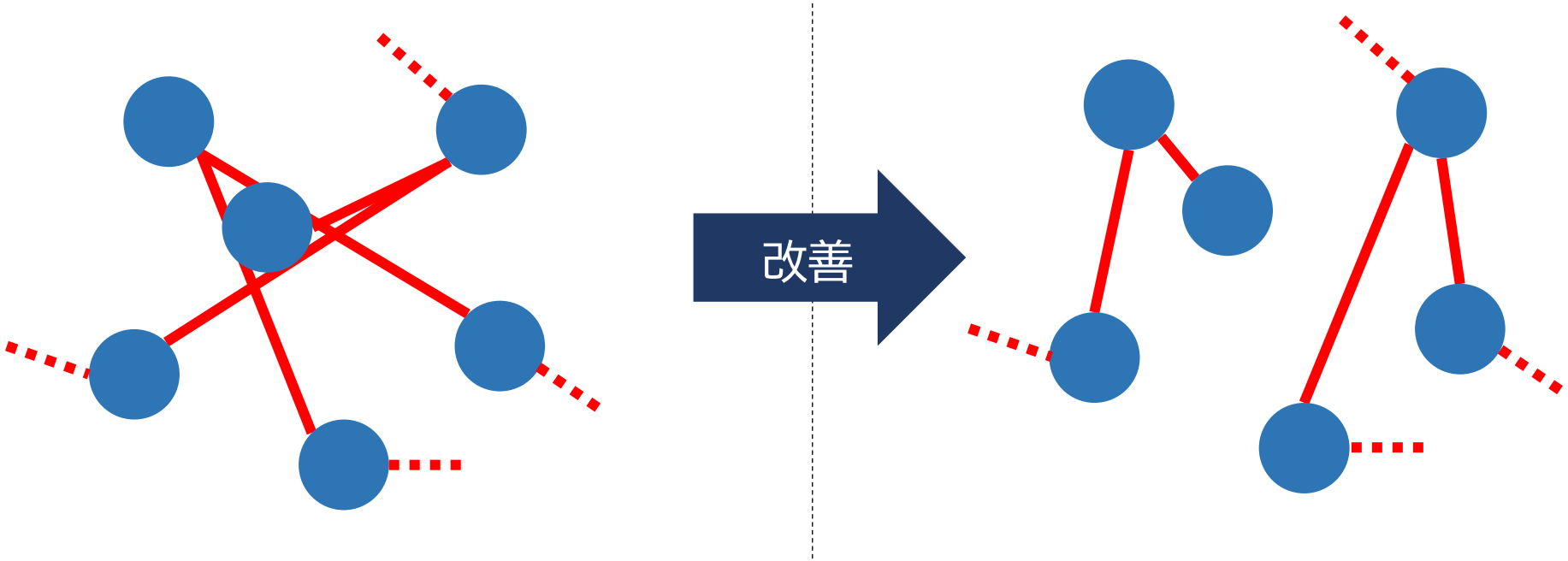
VS.

1 回の変化で「2 つの辺を swap する」

方針 4 – 山登り法 ②

2 つの頂点を swap する場合

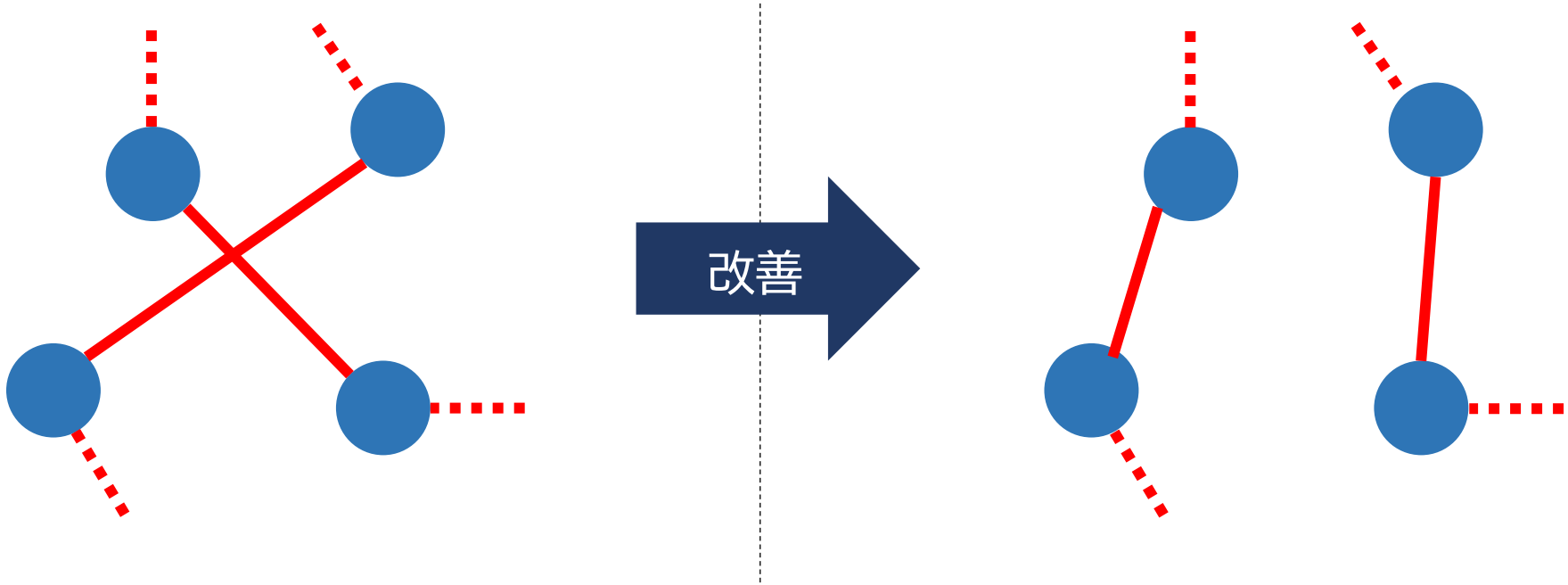
- 変化する角は最大 6 個
- すなわち、1 回の変化における “制約” が大きく、解を動かしにくい…



方針 4 – 山登り法 ②

2 つの辺を swap する場合

- 変化する角は最大 4 個
- すなわち、1 回の変化における “制約” が大きく、解を動かしやすい！



方針 4 – 山登り法 ②

2 つの頂点を swap する場合

- 変化する角は最大 6 個
- すなわち、1 回の変化における “制約” が大きく、解を動かしにくい…

2 つの辺を swap する場合

- 変化する角は最大 4 個
- すなわち、1 回の変化における “制約” が小さく、解を動かしやすい！

このような探索は「2-opt」とも呼ばれる
巡回セールスマン問題にも使われる

方針 4 – 山登り法 ②

2 つの頂点を swap する場合の実装

- チェックポイントを通る順番を $p_1 \to p_2 \to \cdots \to p_N$ とする
- 1 回の変化で、ランダムに x と y を選んで p_x と p_y を交換

2 つの辺を swap する場合の実装（reverse 操作）

- チェックポイントを通る順番を $p_1 \to p_2 \to \cdots \to p_N$ とする
- 1 回の変化で、ランダムに x と y を選んで $p_x, p_{x+1}, \ldots, p_y$ を逆順にする

方針 4 – 山登り法 ②

実験結果

入力データ	swap 操作	reverse 操作
01.txt	70.803 度	100.008 度
02.txt	102.910 度	115.290 度
03.txt	101.877 度	106.645 度
04.txt	118.428 度	139.830 度
05.txt	112.167 度	130.771 度
06.txt	111.941 度	137.443 度
得点	45 点	69 点

※ 注意： これらはひとつの実験結果にすぎません。

69 点獲得

方針 5 – 評価関数の工夫

評価値 にも一工夫できないか？

今までの評価値

角度の最小値を評価値として
これが大きくなる改善を適用していた

新しい評価値

角度を小さい順にソートした最初 K ＝ 5 個を評価値として
これが大きくなる改善を適用してみよう

※「角度の合計値」を評価値とする場合など、スコアが悪化する評価値もあります。

方針 5 – 評価関数の工夫

実験結果 （初期解を貪欲法により得られた解とする・K = 5 とする）

入力データ	最小値を評価	小さい K 個を評価
01.txt	91.822 度	97.812 度
02.txt	119.025 度	131.056 度
03.txt	104.667 度	122.552 度
04.txt	140.513 度	140.897 度
05.txt	132.003 度	140.442 度
06.txt	137.875 度	142.881 度
得点	68 点	80 点

※ 注意： これらはひとつの実験結果にすぎません。

80 点獲得

現状の課題

より良い答えを出すための課題

- 1 回の reverse 操作に O(N) かかってしまう
- 評価値を求めるのに O(N) かかってしまう
- 初期解を貪欲法で作っても、この結果がほとんど保存されない

現状の課題

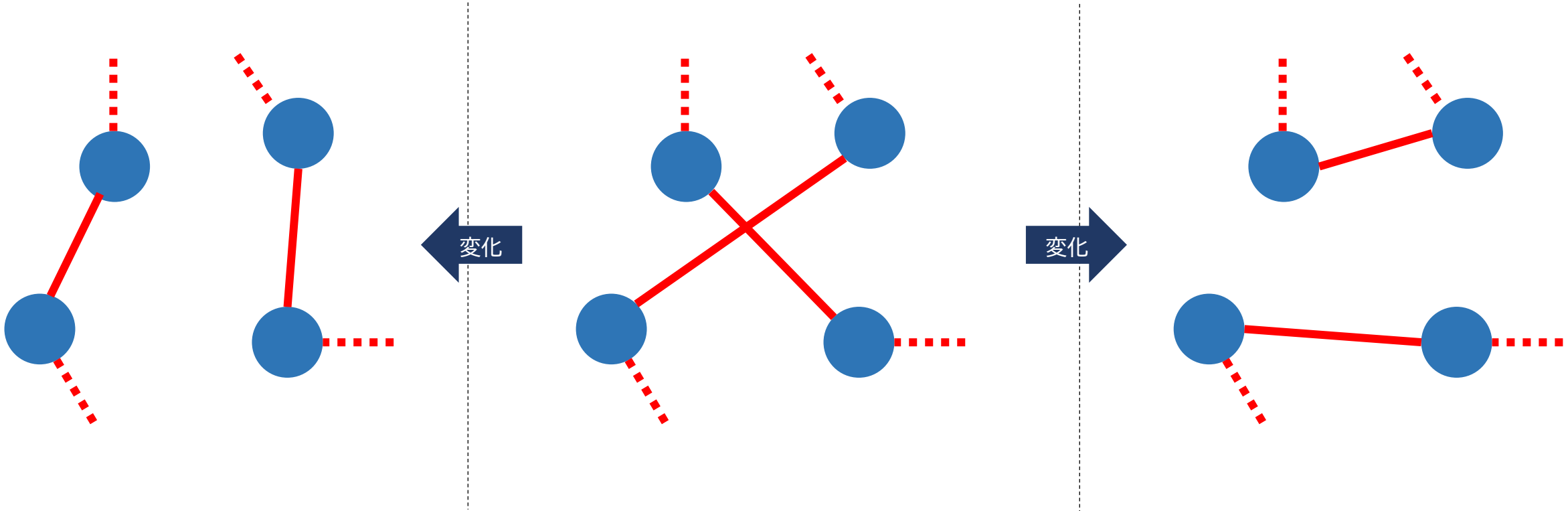
より良い答えを出すための課題

- 1 回の reverse 操作に O(N) かかってしまう
- 評価値を求めるのに O(N) かかってしまう
- 初期解を貪欲法で作っても、この結果がほとんど保存されない

現状の課題

なぜ reverse 操作には O(N) かかるのか？

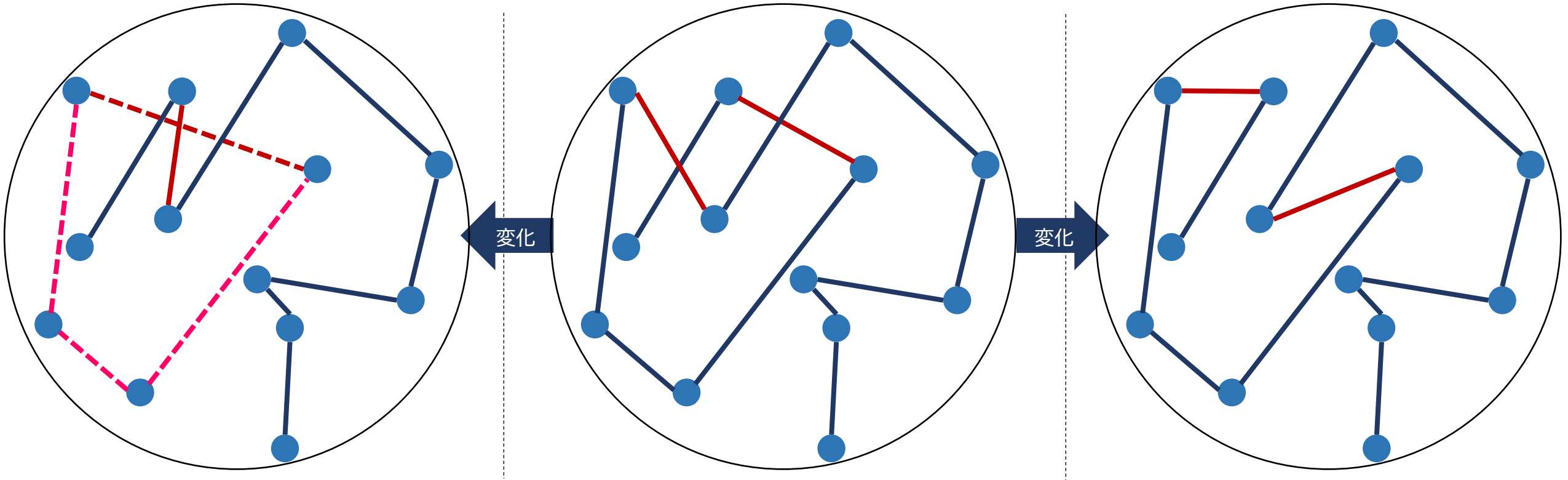
- 辺の交換の仕方には、以下の 2 通りがある



現状の課題

なぜ reverse 操作には O(N) かかるのか？

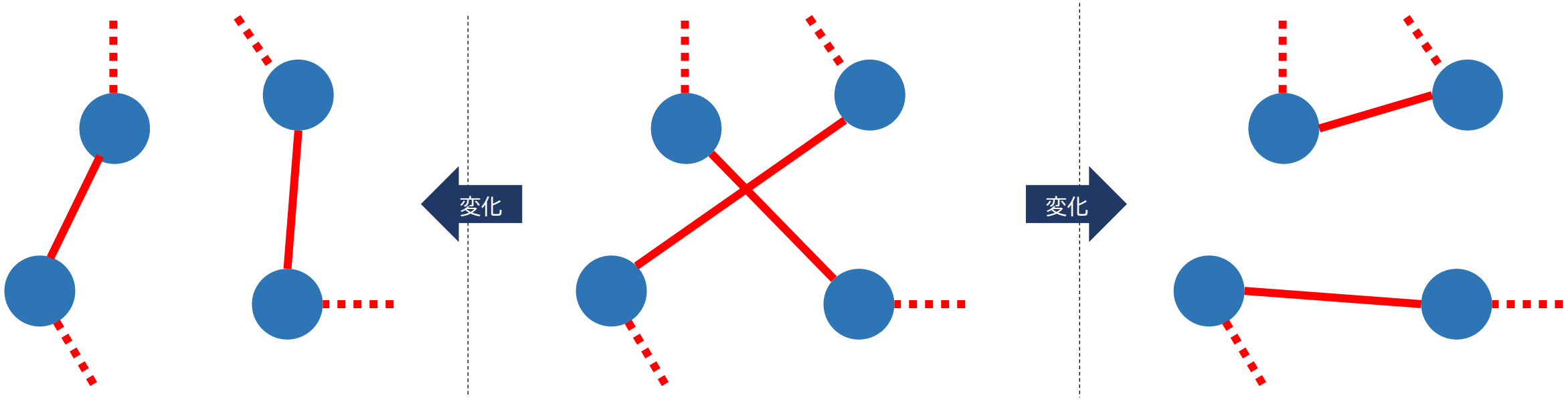
- このうち片方のやり方で サイクル ができてしまう



現状の課題

なぜ reverse 操作には O(N) かかるのか？

- しかし、どちらの変化で サイクル が作られ、どちらの変化が正しいのか判定するだけでも O(N) かかってしまう



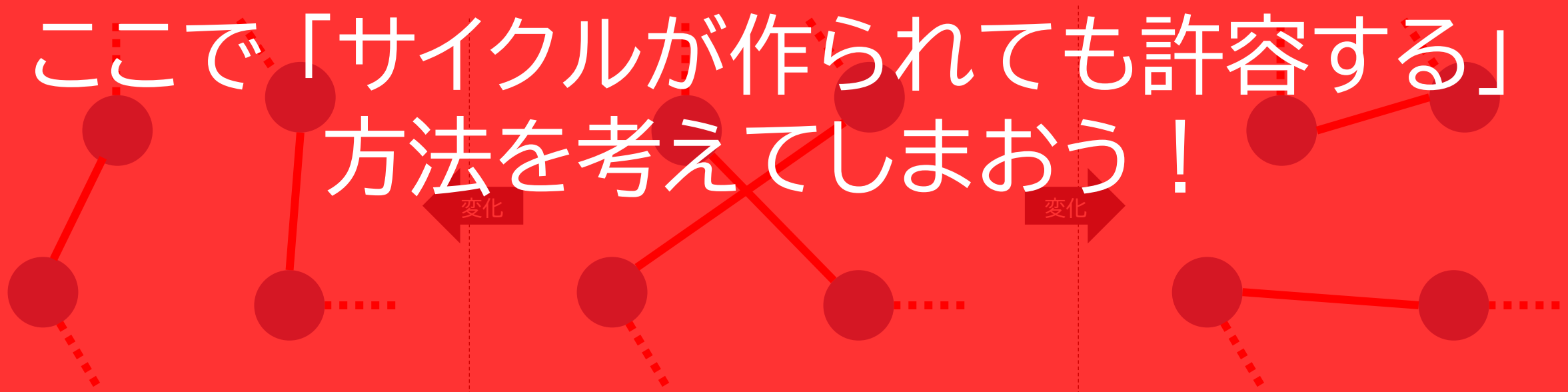
※ 平衡二分探索木などを用いると O(log N) でもできますが、N=200 や 1000 では大きな高速化は見込めません。

現状の課題

なぜ reverse 操作には O(N) かかるのか？

- しかし、どちらの変化で サイクル が作られ、どちらの変化が正しいのか
  判定するだけでも O(N) かかってしまう

ここで「サイクルが作られても許容する」方法を考えてしまおう！



※ 平衡二分探索木などを用いると O(log N) でもできますが、N=200 や 1000 では大きな高速化は見込めません。

現状の課題

新しいアルゴリズム

✈ 2 つの辺を入れ替える変化で、途中でサイクルが作られるのも許容する

最終的に得られる解にサイクルが含まれている可能性もあるが…

☀ ほとんどの場合で、サイクルの個数は 7 個くらい以下になる
（完全ランダムな場合でも、個数の期待値は $\log_e N$ 個程度）

☀ 最小の角度を変えずに、2 辺の入れ替えで「1 個ずつサイクルを減らしていく」

→ かなりの確率で“復元”に成功する！

現状の課題

新しいアルゴリズム

- 2 つの辺を入れ替える変化で、途中でサイクルが作られるのも許容する

最終的に得られる解にサイクルが含まれている可能性もあるが…

- ほとんどの場合、サイクルの個数は $\log_e N$ 個程度以下に収まる
  （完全ランダムな場合でも、個数の期待値は $\log_e N$ 個程度）

- 最小の角度を変えずに、2 辺の入れ替えで「1 個ずつサイクルを減らしていく」

→ かなりの確率で“復元”に成功する！

そのアイデアを使うと…

1 回の変化を定数時間で実現できる！

より良いスコアを求めて

より良い答えを出すための課題

1 回の reverse 操作に O(N) かかってしまう

評価値を求めるのに O(N) かかってしまう

初期解を貪欲法で作っても、この結果がほとんど保存されない

より良いスコアを求めて

現状の評価関数だと…

- 初期解を貪欲で生成すると、外周に近い頂点の角は 150° 以上のものも多い

しかし、最小値「だけ」を保存すると、150° の角なんて評価に関係ないので…

元々良かった角が崩壊する

ということが起こってしまいます…

より良いスコアを求めて

評価関数に取り込みたい値

- 入力データごとに基準値 θ を決め打ちする
- 評価値には「基準値 θ 以上の角の個数」を入れてみたらどうか…？
- 全ての角が基準値 θ 以上になったらゲームクリア

今までの「最小の角度」みたいな評価値も残した方がよさそうだけど…

より良いスコアを求めて

新たな評価関数

基準値を θ ／ 各頂点の角度を $A_1, A_2, \ldots, A_N$ とする

$$f(x) = \begin{cases} x - \theta & (x < \theta \text{ の場合}) \\ complete\_score & (x \geq \theta \text{ の場合}) \end{cases}$$ として

評価値を $f(A_1) + f(A_2) + \cdots + f(A_N)$ としてみよう

※ $complete\_score$ は 30～100 くらいに設定される定数

より良いスコアを求めて

新たな評価関数

基準値を θ ／ 各頂点の角度を $A_1, A_2, \ldots, A_N$ とする

$$f(x) = \begin{cases} x - \theta & (x < \theta \text{ の場合}) \\ complete\_score - x & (\text{それ以外の場合}) \end{cases}$$ として

評価値を $f(A_1) + f(A_2) + \cdots + f(A_N)$ としてみよう

※ $complete\_score$ は 30～100 くらいに設定される定数

変わる角は最大で 4 個なので
評価値を定数時間で更新できる！

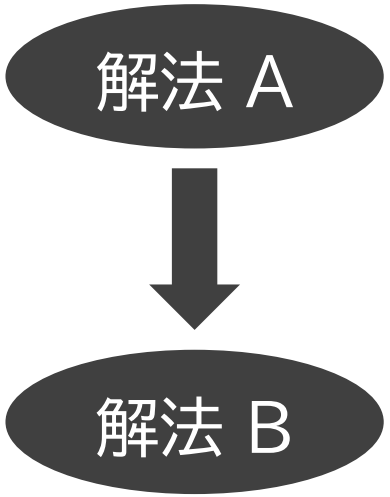
より良いスコアを求めて

ここまでで手に入れた改善手法

- 1 回の reverse 操作は定数時間で実現できた
- 評価値も工夫すれば定数時間で求められる
- 初期解の良い部分は残した状態で解の改善がなされる

より良いスコアを求めて

新たな評価関数を使って解いてみよう！



評価関数が上がるか同じな場合、交換を適用する

評価関数が上がるか同じな場合、交換を適用する
評価関数がほんの少し落ちても、確率的に交換を適用する
（焼きなまし法のようなアイデア）

※ ただし初期解は貪欲法で作るものとする

より良いスコアを求めて

解法 A の実験結果 （目安として 1 秒のループ回数は 400 万回・ループ回数制限は 4 億回）

入力データ	Z_0	θ	*complete_score*	スコア	ループ回数
01.txt	100°	100°	30	100.008 度	994
02.txt	143°	140°	50	140.135 度	24,921,224
03.txt	134°	132°	50	132.005 度	117,829,505
04.txt	156°	152°	50	152.004 度	218,769,334
05.txt	150°	147°	50	147.020 度	320,099,974
06.txt	153°	147°	100	147.007 度	369,905,056
得点				93 点	

※ 注意： これらはひとつの実験結果にすぎません。

93 点獲得

より良いスコアを求めて

解法 B の実験結果 （目安として 1 秒のループ回数は 400 万回・ループ回数制限は 4 億回）

入力データ	Z_0	θ	*complete_score*	スコア	ループ回数
01.txt	100°	100°	60	100.008 度	14,922
02.txt	143°	142°	80	142.014 度	370,739,181
03.txt	134°	134°	35	134.063 度	25,785,435
04.txt	156°	153°	60	153.003 度	325,223,062
05.txt	150°	147°	40	147.008 度	133,813,958
06.txt	153°	148°	100	148.000 度	110,789,454
得点				95 点	

※ 注意： これらはひとつの実験結果にすぎません。

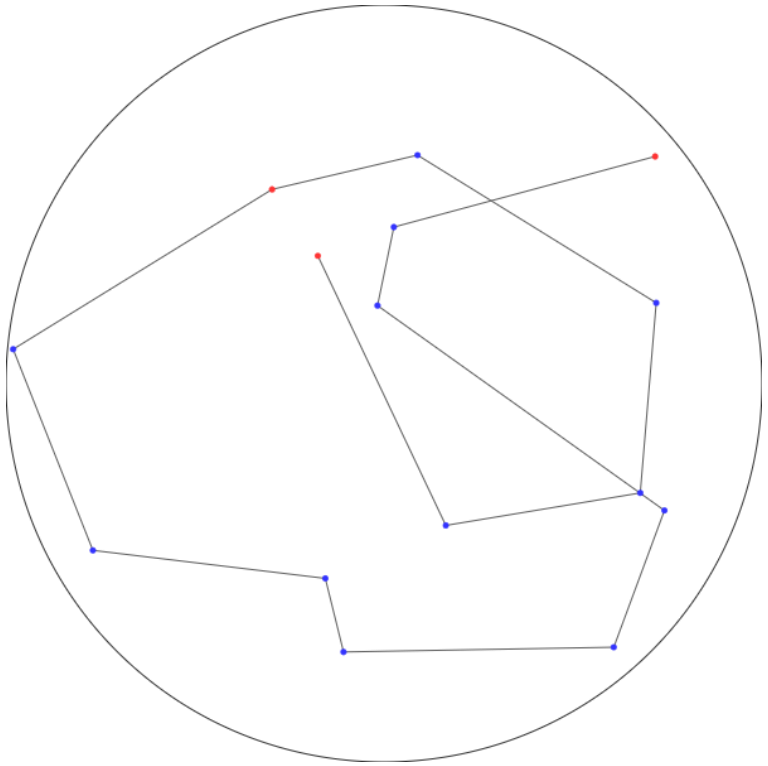
95 点獲得

ギャラリー

現在得られている最も良い解を
ビジュアライズしたものを紹介します

入力データ 1（N = 15）

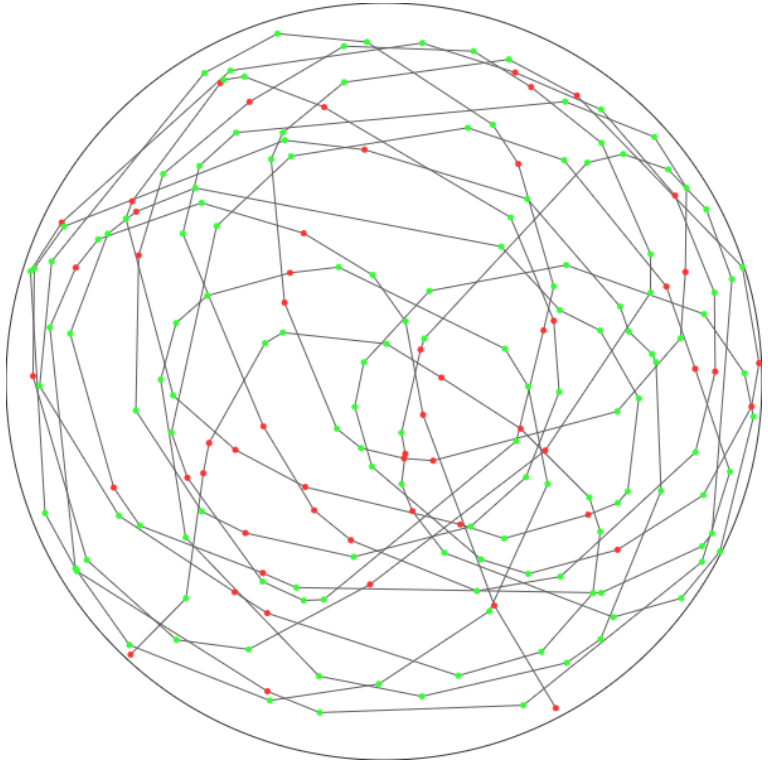
01.txt



Z = 100.008°（最適解）

入力データ 2（N = 200）

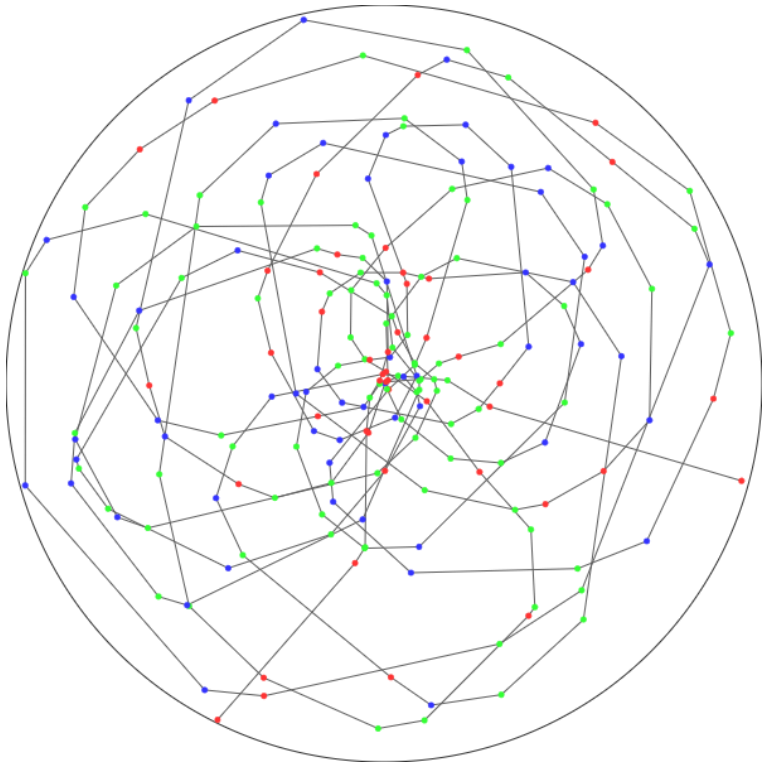
02.txt



Z = 143.048°

入力データ 3（N = 200）

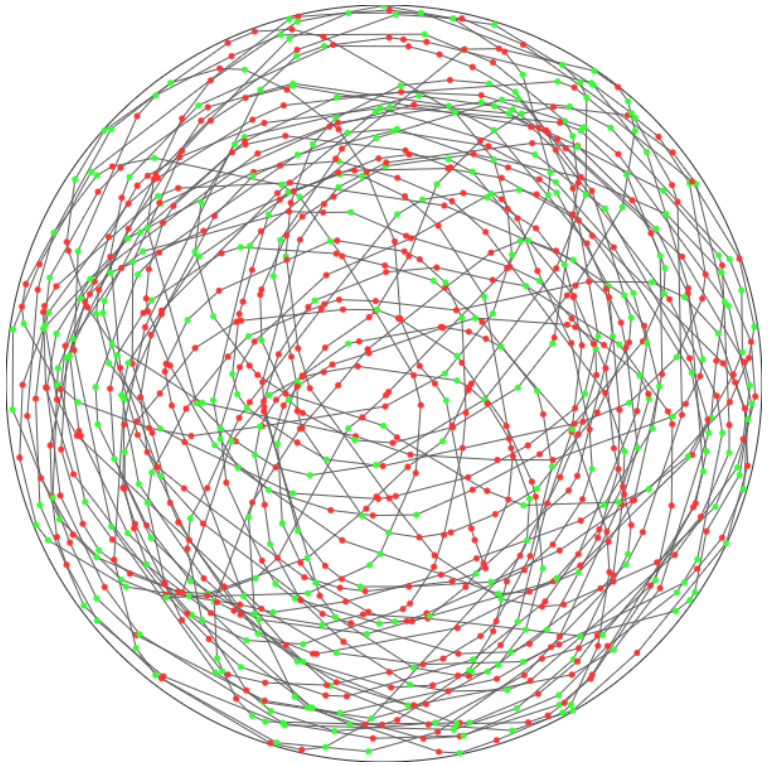
03.txt



Z = 134.554°（最適解）

入力データ４（N = 1000）

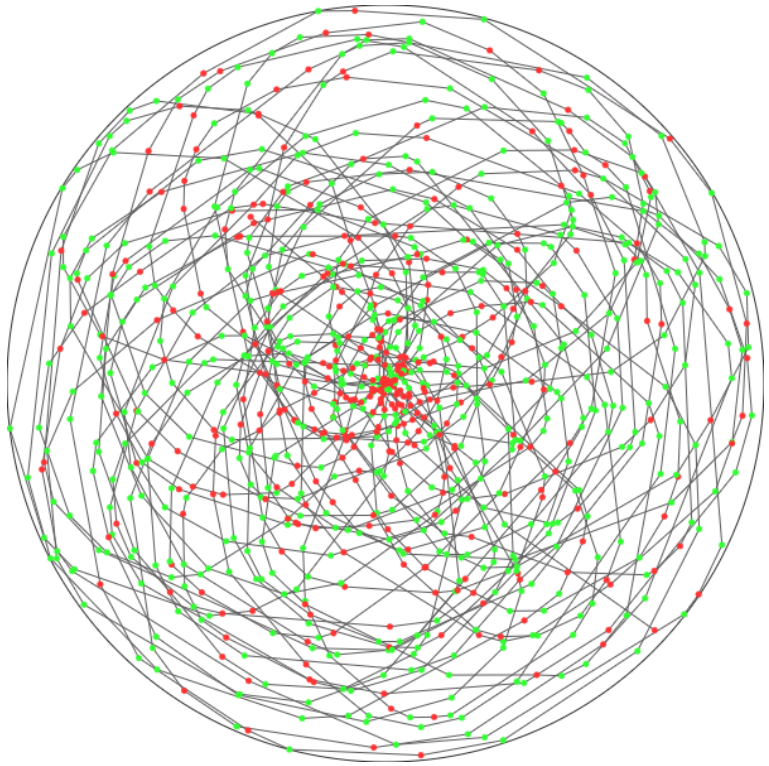
04.txt



Z = 156.011°

入力データ 5（N = 1000）

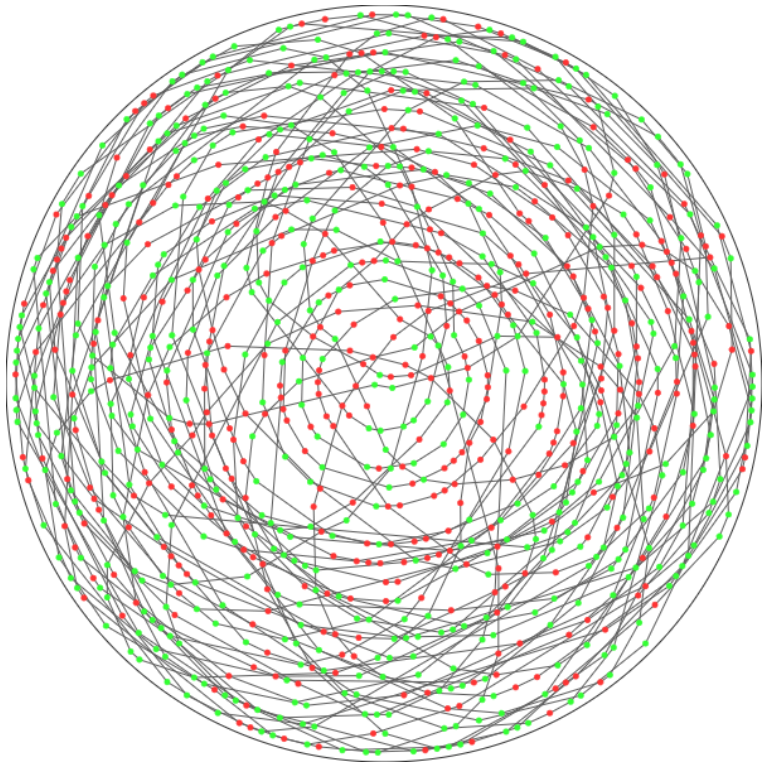
05.txt



Z = 150.007°

入力データ 6（N = 1000）

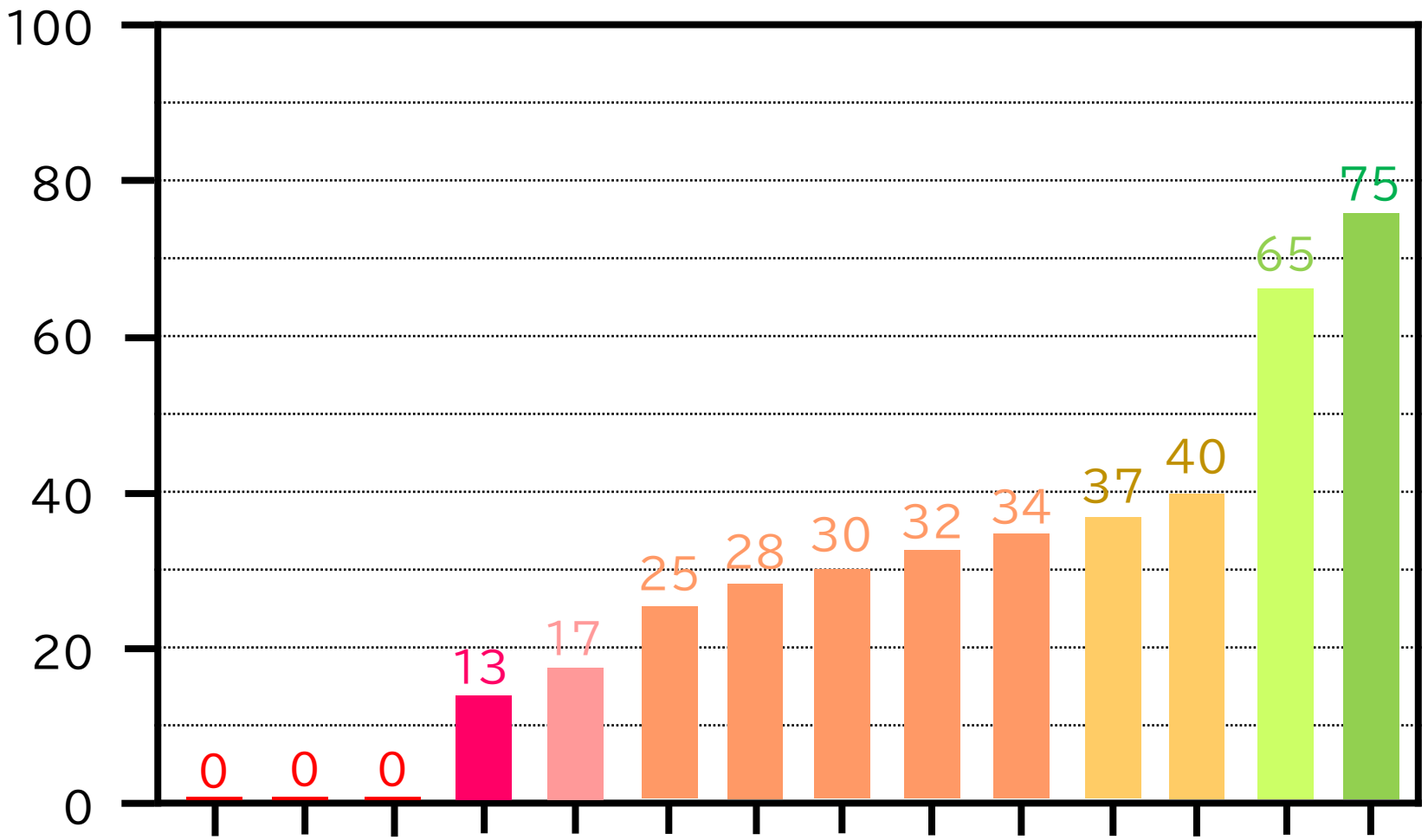
06.txt



Z = 153.022°

得点分布

得点分布



＜統計データ＞

最高点	75 点
最低点	0 点
平均点	28.3 点
標準偏差	22.5 点