このアプリについて SATSP

コンピュータ部 6年 宮岡

本アプリを起動して頂きありがとうございます。ここでは、本アプリが解こうとしている問題や内部の仕組みについて**ゆっくり、まったり、詳しく**説明しています。このpdfは下のQRコードからもご覧になれます。どうぞ、よろしくお願いいたします。

目次

1 SAについて

SA・Simmulated Annealing・擬似アニーリング法とは、数学の問題を解く1つの計算技法です。S. Kirkpatrick、C. D. Gelatt、 M. P. Vecchiらが1983年に考案しました。本アプリはこの技法を使用しています。

1.1 最小値を探せ

早速ですが問題です。以下の関数の**最小値**はいくつでしょう。またそれはxがいくつの時でしょう。

$$f(x) = (x - 3)^2$$

この関数の最小値は0で、 $x = 3$ の時ですね。

では、以下の関数の最小値はいくつでしょう。またそれはxがいくつでyがいくつの時でしょう？？

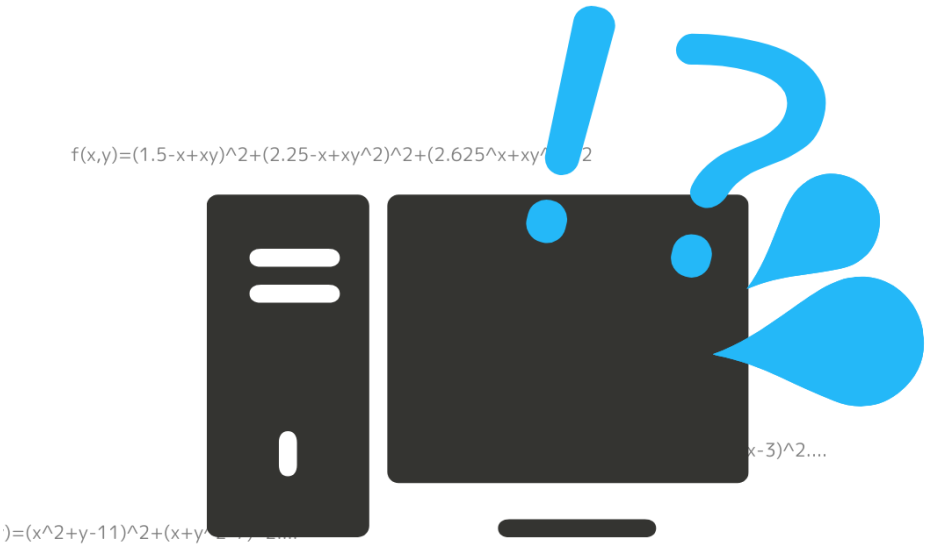
$$f(x, y) = (x^2 + y - 11)^2 + (x + y^2 - 7)^2$$

※ Himmelblau関数と呼ばれています

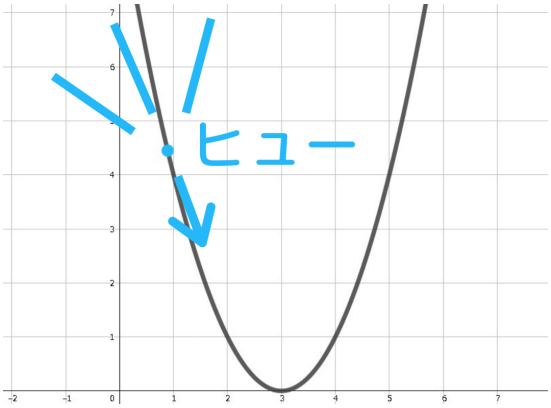
1.2 分からんわ！

答えを出すには相当の努力が要りそうですので、考えないことにしましょう。さて、このような関数の最小値を出すというのは日常生活ではあまりやらないと思いますが、**この世界には関数の最小値を日々求め続けているものがいます**。それは誰でしょう。

人工知能です。人工知能は、自分の知性を上げるためにいくつもの複雑な関数の最小値を求める必要があります。関数の最小値は方程式を立てれば解けるかもしれませんが、彼らが対象としている関数**はとても複雑で、方程式も複雑**なものになっています。いくらコンピューターの計算が早かろうと、複雑な方程式を解くのはやはり時間がかかってしまいます。**いかに手抜きして関数の最小値を求めるか**。これが課題になっています。

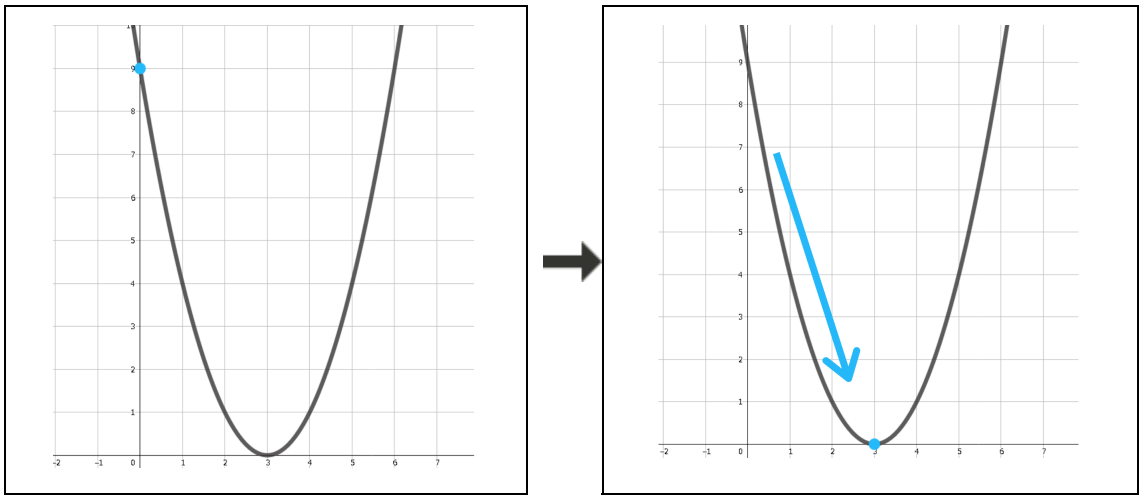


先ほどのHimmelblau関数のような難しい関数の最小値を求めるために、コンピューターはちょっと面白い方法を使います。それは、**グラフに玉を落とす**というものです。

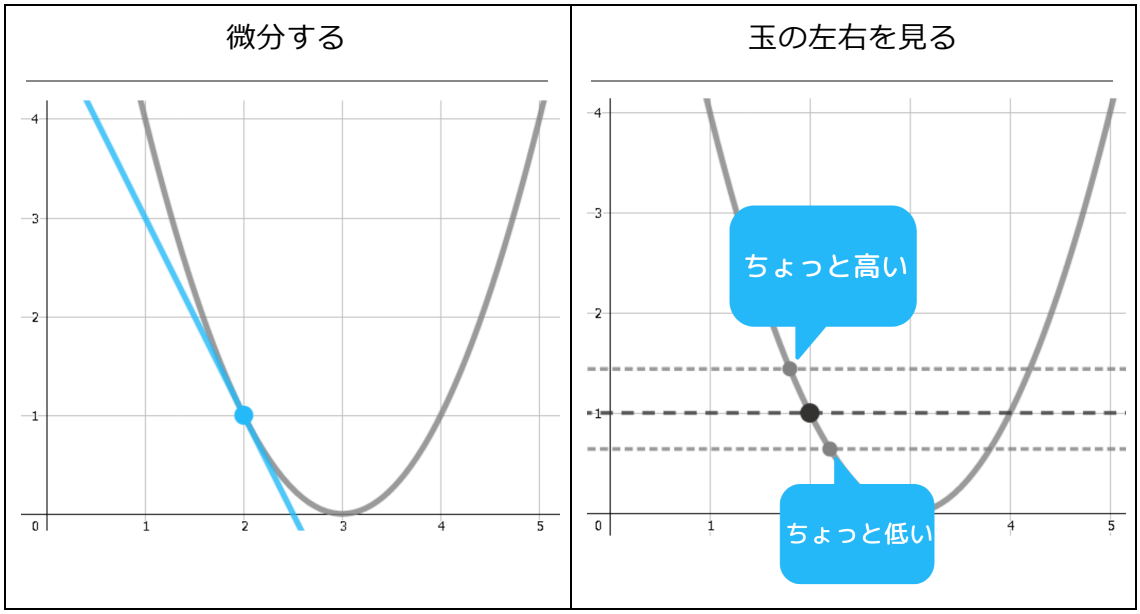


玉をグラフに落とすと玉は坂を下っていき、**最終的に谷底、つまり最小値にたどり着ける**のです。

玉を $x = 0$ に落とします。すると、そこから右の方向に転がり $x = 3$ で停止します。



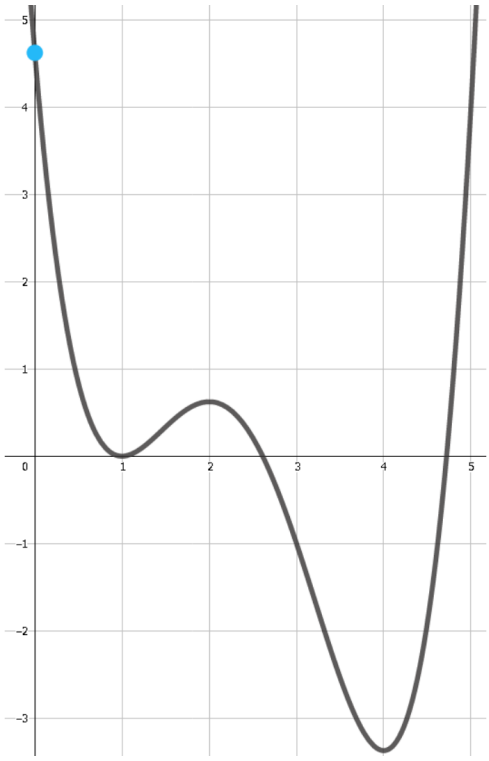
玉を坂の傾きに従って落とすプログラムは至って簡単に作れます。与えられた関数を**微分したり、玉の左右を見たり**すればどちらの向きに傾いているかが分かります。



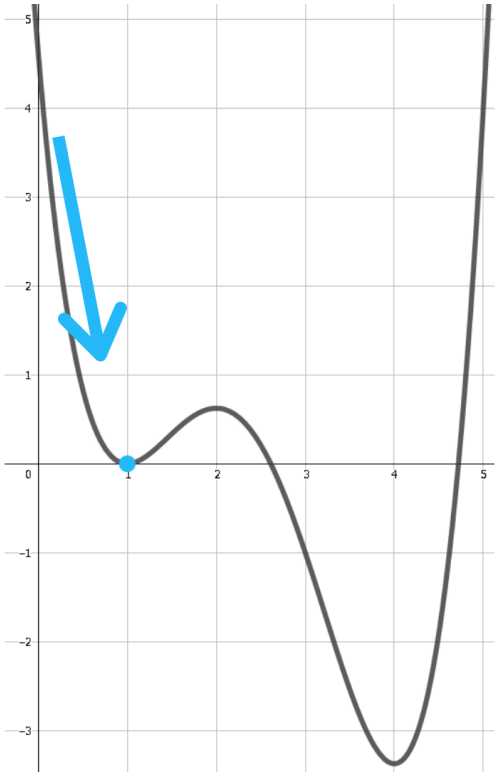
1.3 坂下りの問題点

さて、坂の通りに下ればより少ない値に行くことはできます。しかし、行きついた先は本当にその関数の最小値と言ってもいいのでしょうか。

以下の例を見てください。このような関数があり、x=0に玉が落とされたとします。



転がって、転がって、転がった先はこの場所となりました。



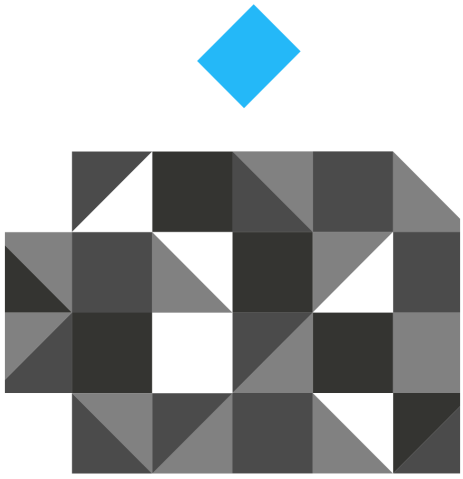
さて、この場所は本当に最小値でしょうか。違いますね。本当の最小値はその一つ右の谷です。**ただ単に坂を下るだけでは本当の最小値に行けるとは限らない**んですね。困ったものです。どうすればより小さい最小値に行けるのでしょうか。

1.4 自然に学ぶSA

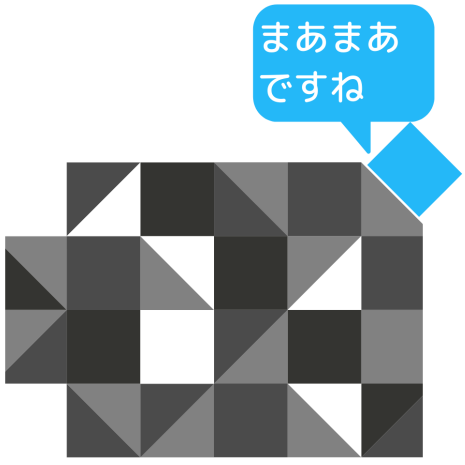
この問題を解決する一つの計算技法が、SA・Simmulated Annealing・疑似アニーリングと呼ばれるものです。これは**ミクロな世界**で起きていることを真似たちょっと面白い計算技法です。

---

1. ミクロな世界では、物質を作る小さい粒が動き回っています。そして、彼らは小さい穴などの**居心地のいい場所**を探しています。



2. 一つの小さい粒がまあまあ居心地のいい場所を見つけたとしましょう。



3. この時、基本的に小さい粒はそこに留まることになりますが、ごくたまに、**あえてその場からもう一度脱出する**ことが起きます。

4. さっきよりも居心地のよさそうな場所にたどり着きました。

小さい粒はある確率で居心地のいい場所から脱出しようとしますが、この確率は、**今の居心地のよさ**と、**今の温度**によって変わってきます。今の居心地がよければよいほど、確率は下がります。また、今の温度が低ければ低いほど確率は下がります。実際には、以下のような式を使って計算することができるそうです。

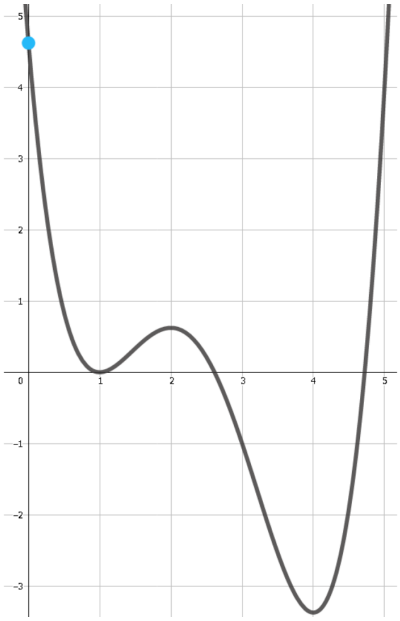
$$exp(-\frac{\Delta E}{k_B T})$$

熱した液体の金属などを**ゆっくり冷やす**と、小さな粒は居心地のいい場所を見つける時間を長く持つことができ、**小さな粒がきれいに並ぶ**ため、綺麗な金属ができます。このような金属の加工方法をアニーリングといいます。

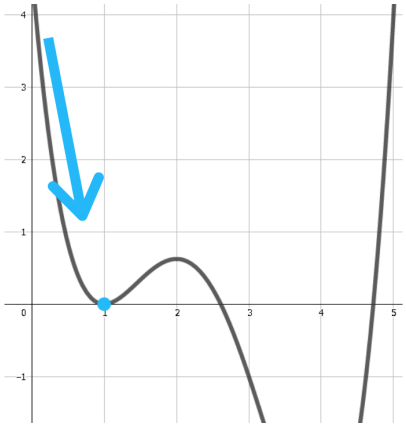
このミクロの世界で起きていることを、最小値を求める計算に応用してしまおうというのがSAです。**ミクロな世界での場所を関数のグラフに、ミクロな世界での小さい粒を関数の点、居心地のいい場所を最小値**に置き換えて、以下のような処理を行います。

---

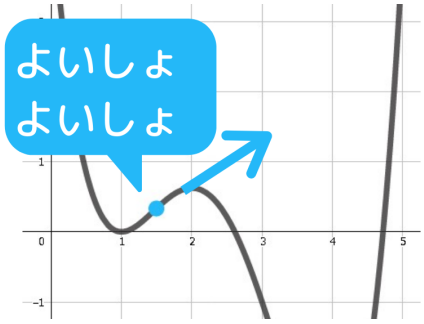
1. x=0に玉を落とします。グラフ上で、玉が動き回っています。玉は、より小さい場所をさがします。



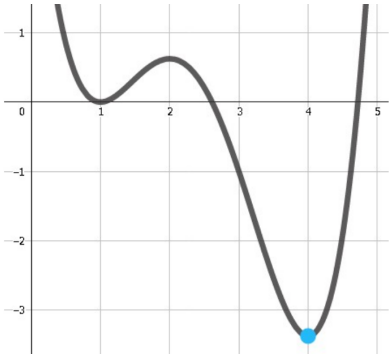
2. 玉が「最小値」を見つけたとしましょう。ここは確かに小さそうです。



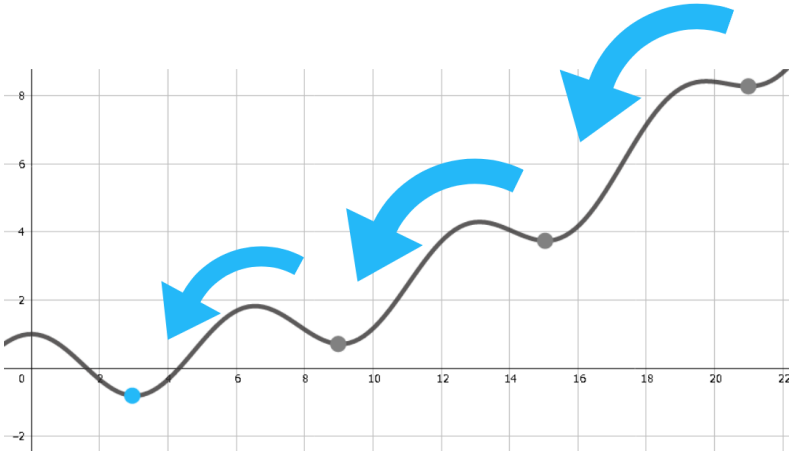
3. 一定の確率で、より大きい場所へと動きます。



4. 玉がさっきよりも小さい場所にたどり着きました。



これを**温度を徐々に下げながら**繰り返すと、凹凸が大量に存在する複雑な関数に対しても玉がより小さい場所を見つけられる可能性が高くなります。



先ほどの坂下りでは玉は本当の最小値に行くことはできませんでしたが、このSAでは本当の最小値に行くことができました。SAの凄いところはここにあります。**SAは単なる坂下りよりも、より小さい最小値を見つけられる可能性が高い**のです。

1.5 アルゴリズム

SAには、以下のような値が必要となります。

$f(x)$	最小値を見つける対象の関数
x	今の引数(玉の位置)
x_0	最初の引数
x^*	今分かっている限りの最小値をとる引数(**最良解**と呼びます)
T	今の温度
T_0	最初の温度

以下のような処理を行います。

---

1) x <– x_0 引数(玉の位置)を初期値にセットします。
2) x^* <– x 現時点での最良解は x です。
3) T <– T_0 温度を初期値にセットします。
4) T が0になるまで、もしくは十分な回数だけ以下の処理を繰り返します。
   a) 十分に小さい値 Δx を作ります。半々の確率で正負にします。
   b) x' <– $x + \Delta x$
      引数を少しずらします。玉を左右に動かす動作を再現しています。
   c) もし $f(x') \leq f(x)$ 引数(玉)がより低い場所に来たら
      i) x <– x'
         無条件で今の引数をその場所へ更新します。
   d) もし $f(x') > f(x)$ 引数(玉)がより高い場所に来たら
      i) ΔE <– $f(x') - f(x)$
         どのくらい高い場所に来たかを求めます。
      ii) $exp(-\frac{\Delta E}{T})$ の確率で x <– x'
         これがSA法の最もキーとなる処理です。「一定確率で居心地のいい場所から脱出する」動作を再現しています。
   e) もし $f(x') < f(x^*)$ これまでに見たこともないほど小さい値が出たら
      i) x^* <– x'
         最良解の記録を更新します。
   f) T の値を少し下げ、冷却します。

「十分な回数」「十分に小さい値 Δx」「 T の値を少し下げる」など曖昧な表現が多いですが、ここら辺の処理の詳細は実際にSAを使用する環境によって大きく変わります。

2 TSPについて

TSP・Traveling Salesman Problem・巡回セールスマン問題とは、**全ての点を一回ずつ通る閉じた経路**(全体として一つの大きな輪を描くような経路)の中で、**最も短いもの**を求めるという問題です。本アプリはこの問題を解こうとしています。

サンタさんは一日のうちに子供たちの家を回らなければいけません。この時、どう回れば移動距離が少なくなるか考えるのもTSPです。

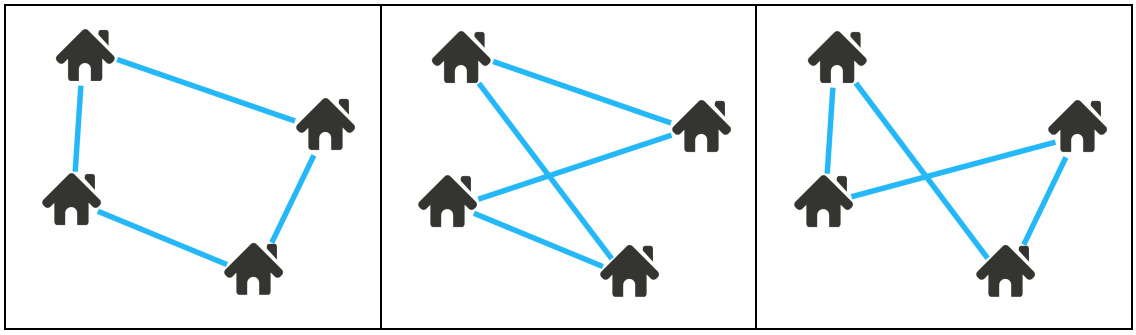
巡回セールスマン問題という名前は、「全ての家庭に訪問する必要のあるセールスマンが、どのような経路で訪問すれば効率よく回れるか」という問題例に由来しています。

2.1 超難問

この問題はとても難しいです。正しい最短経路を求めるには、全ての経路を見ていく必要がありますが、その経路が大量に存在するのです。それでは、具体的に経路が何通りあるのか考えてみましょう。

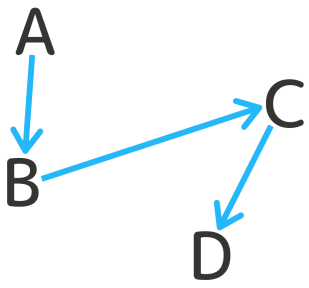
2.2 点の数が4個の場合

例えば点の数が4個あったとします。すると、直感的に3通りの経路が考えられると思います。

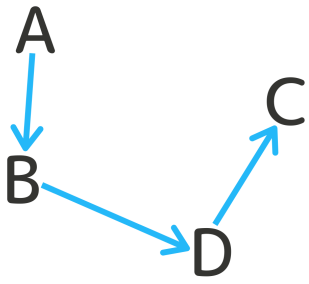


この3という数字を直観ではなく計算で求められないでしょうか。

各点に訪問する順番を考えます。各点をA、B、C、Dと置くと、経路の一つの例として、A→B→C→Dが考えられます。



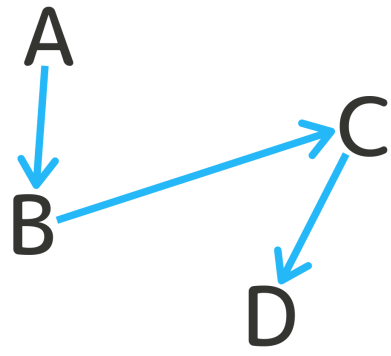
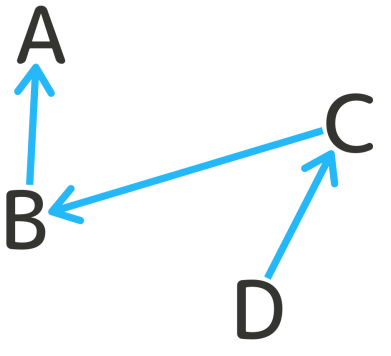
他にはA→B→D→Cも考えられます。



全てで訪問する順番が何通りあるか考えると、$4 \times 3 \times 2 \times 1 = 4! =$ 24通りです。

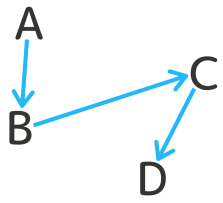
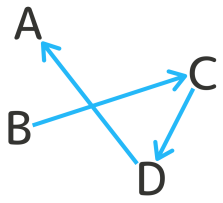
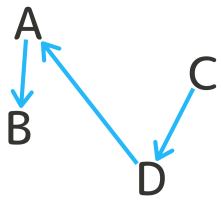
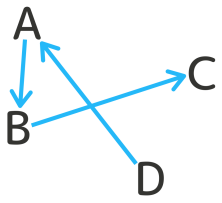
しかしこの訪問する順番には、閉じた経路として**重複**しているものがいくつかあります。

例えば、A→B→C→DとD→C→B→Aは訪問する**順番が逆なだけ**で経路は同じとみていいです。

A→B→C→D	D→C→B→A
	

2重の重複が存在することが分かりました。

また、A→B→C→D、B→C→D→A、C→D→A→B、D→A→B→Cは**始点が違うだけ**で経路の形は皆同じですね。

A→B→C→D	B→C→D→A	C→D→A→B	D→A→B→C
			

4重の重複が見つかりました。

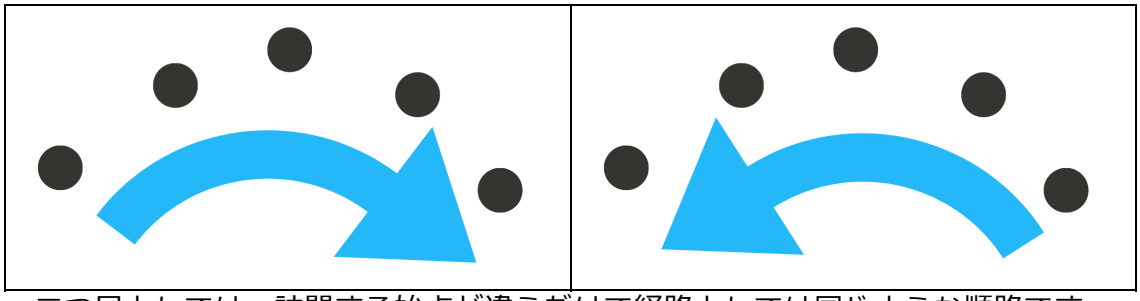
まとめると、24通りの中に2重の重複と4重の重複がありました。よって、4つの点を訪問する閉じた経路は全部で $24 \div 2 \div 4 =$ **3通り**と求まりました。

2.3 点の数がN個

それでは点の数が N 個あったら閉じた経路は何通りあるか考えてみましょう。
まず、各点に訪問する順番を考えます。先程の例の点が4個の時は
$4 \times 3 \times 2 \times 1 = 4!$ で求めました。よって、今回は $N!$ で求められますね。

次は、重複が何個あるか考えてみます。先程の例では重複は2種類ありました。ですので、ここでも重複は2種類あるはずです。

一つ目としては、訪問する順序が逆になっているような順路です。これは2重あります。



二つ目としては、訪問する始点が違うだけで経路としては同じような順路です。点の数が N 個あるのなら始点となり得る点も N 個存在しますので、ここでは N 重となります。

まとめると、 $N!$ 通りの順路の中には**2重の重複とN重の重複**が存在していることになります。
よって、N個の点を回るような閉じた経路は次のようにして求めます。

$$N! \div N \div 2 = (N-1)(N-2)(N-3) \ldots 3$$

2.3 爆弾

この式に実際に値を入れて計算してみます。例えば、点の数が10個あるとして計算します。先ほどの式の N を10とすると、

$10! \div 10 \div 2 = 9 \times 8 \times 7 \times 6 \times 5 \times 4 \times 3 =$ 181,440通りになりました。なんと多いことでしょう。
それでは点の数を20にすると、 $20! \div 20 \div 2 =$ 60,822,550,204,416,000通り。
なんと千兆を超え京まで来てしまいました。

点が1個増えるだけで経路の数が**爆発**するように増加します。例えば点が100個ある所に点が1つだけ加わって101個になったとします。すると、場合の数は100倍になります。こんな爆弾のような数をそのままの形で扱うのは少し限界がありそうです。

2.4 桁数で考える

閉じた経路の場合の数は爆弾です。そのままの形で見ても数が大きすぎてよく分かりません。なので、その代わりに**その数が何桁あるのか**で見てみます。例えば「1,000通り」あった場合は「4桁」と表示し、「10,000通り」あった場合は「5桁」と表示します。

$N! \div N \div 2$が何桁あるのかを計算します。桁数をDとすると、Dは以下の式で求められます。

$$D = floor(1 + log_{10} \frac{N!}{2N})$$

普通の電卓では$N!$などの巨大数は扱えないので、工夫して以下のように変換します。

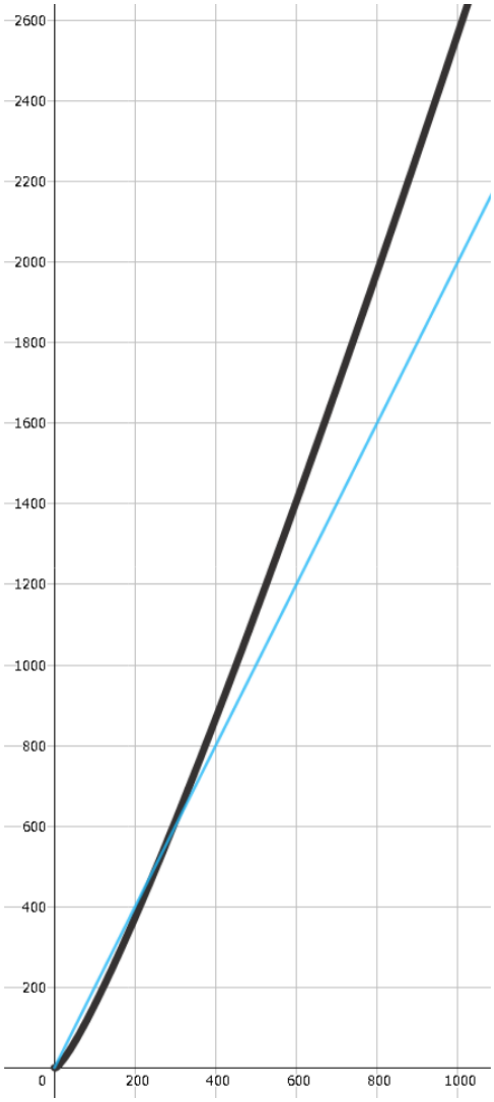
$$D = floor(1 + \sum_{k=3}^{N-1} log_{10} k)$$

点が200個ある時の閉じた経路の場合の数の桁数は、電卓を用いて計算すると373桁と求められました。

　この式をもう少しいじって、グラフに表せるようにしましょう。この式を積分で近似します。

$$D \approx \int_{3}^{N} log_{10} x dx = \frac{1}{log10} \{(NlogN - N) - (3log3 - 3)\}$$

これを黒いグラフで表示しました。また、$D = 2N$を青いグラフで表示しました。



　点の数が数十～数百個の時の考えられる閉じた経路の数は、だいたい点の数×2くらいだと考えていいでしょう。

2.5 宇宙を超える？スパコンで5000兆年！？

点がたくさんある時の閉じた経路の場合の数を、桁数を求めることによって見やすく求めることができました。点が200個ある時の閉じた経路の場合の数の桁数は、具体的には373桁と求められました。この373桁がどのくらい大きな数字なのか考えていきましょう。

一年の秒数は8桁
世界の人口は10桁
地球の重さ(キログラム)は24桁
500mlの水の水分子の個数は26桁
地球から見える宇宙全体にある原子の個数は82桁

373桁という数字は、**宇宙にある原子の数よりも圧倒的に多い**ことが分かります。

　こんなに多くの経路があるとスパコンでも計算するのは大変です。では、具体的にスパコンで計算するのには何年ほどかかるのか計算してみましょう。スパコンでは1秒に1京通りの経路を考えることができるとします。これは17桁です。また、1年は約 3.2×10^7 秒です。これは8桁です。
373桁の数をこれら2つの数で割ります。

$$10^{372} \div 10^{16} \div (3.2 \times 10^7) = 3.1 \times 10^{348}$$

計算にかかる年数もこれまた巨大数で、**349桁**となりました。宇宙の今の年齢は138億歳、たったの11桁です。計算には永遠といえるほどの時間が必要になるのです。
　TSPは、宇宙よりも遥かに大きな問題なんです。

※地球から見える宇宙全体にある原子の個数は、以下のようにして計算しました。
アボガドロ定数を 6×10^{23} とし、太陽は地球の質量(6×10^{24} kg)の 3×10^6 倍で水素原子から構成され、同様の恒星が銀河内に 10^{11} 個存在し、同様の銀河が宇宙内に 2×10^{12} 個存在する。

$$6 \times 10^{23} \times 10^3 \times 6 \times 10^{24} \times 3 \times 10^6 \times 10^{11} \times 2 \times 10^{12} = 2 \times 10^{81}$$

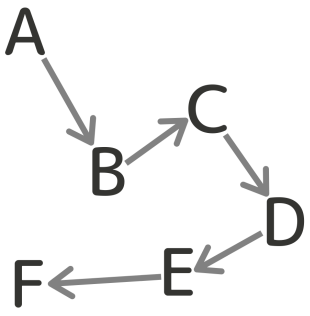
3 SA + TSP = SATSP

本アプリはそんな難解なTSPをSAを使って短時間で求めようというものです。

3.1 引数は順路、関数は距離

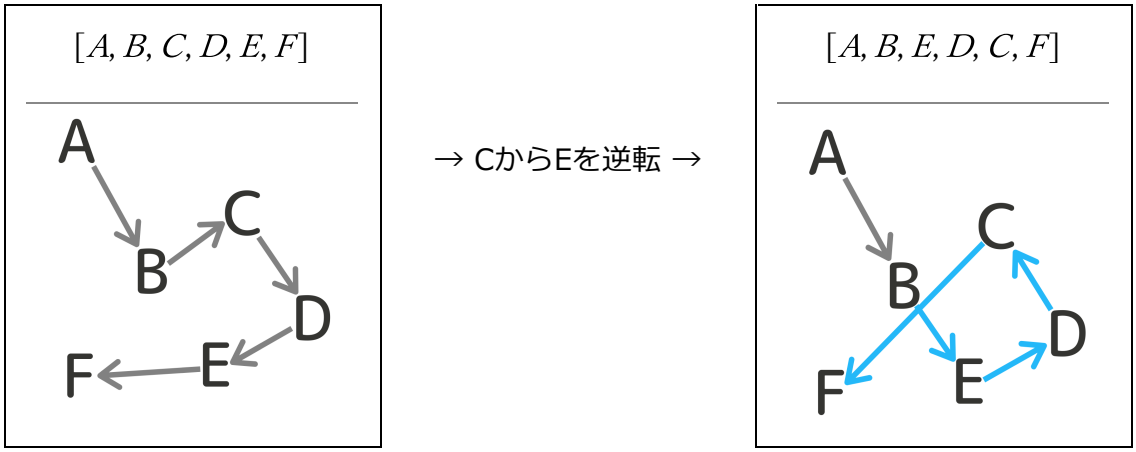
さて、SAを使うためには最小値を求める対象の**関数**とその関数に代入する**引数**が必要で、引数は少しだけ**ずらせる**ようになっている必要があります。本アプリでは、引数と関数を以下のように決めています。

引数は**順路を表す数列**とします。例えば以下のような順路があるとしましょう。

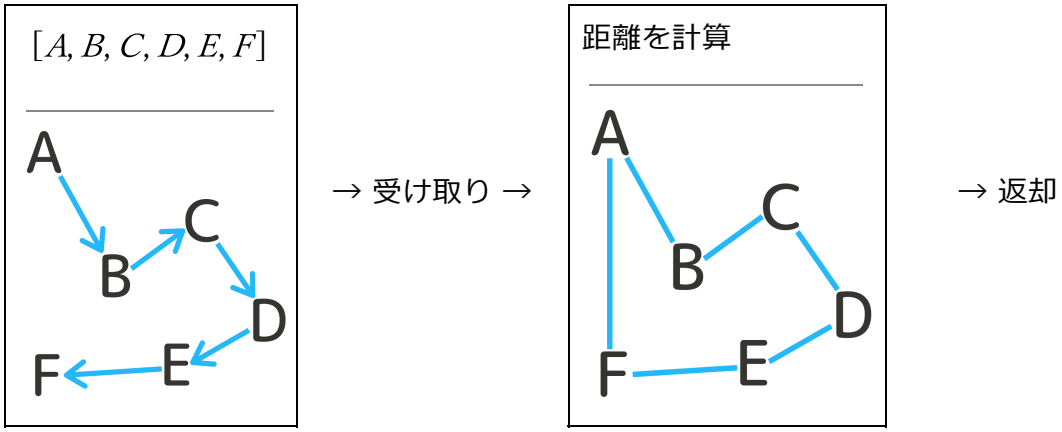


A→B→C→D→E→Fという順路になっています。これを数列で $[A, B, C, D, E, F]$ と表すことにします。

また、引数は**数列の一部を逆転させる**ことでずらします。例えば $[A, B, C, D, E, F]$ から適当にCとEを選び出し、CからEを逆転させ $[A, B, E, D, C, F]$ とします。訪問する順序を入れ替えることになり、順路が少しだけ変わります。



関数は順路を表す数列を受け取り、その順路が表す経路の**総距離**を返すようにします。例えば $[A, B, C, D, E, F]$ を受け取ったとします。それをA、B、C、D、E、Fの順に通る閉じた経路と解釈し、AB、BC、CD、DE、EF、FAの距離を合計して返却します。



この関数をSAに当てはめれば、関数の最小値をとるような引数、つまり、**総距離が最短になるような順路を計算してくれる**わけです。

3.2 アルゴリズム

本アプリでは、来場客の皆様に**処理時間**と**初期温度**を設定して頂いています。これら2つの数値は処理の中で使います。
以下のような値を使います。

t	残りの処理時間
t_0	来場客の皆様に設定して頂いた全体の処理時間
$f(x)$	順路が表す閉じた経路の距離を返す関数
x	今の順路(形式は数列です)
x^*	今分かっている限りの最短経路をとる順路(以下**最良順路**)
T	今の温度
T_0	来場客の皆様に設定して頂いた初期温度

以下のような処理を行います。

1) x <– x_0 順路を初期値にセットします。
2) x^* <– x 現時点での最良順路は x です。
3) t <– t_0 残りの処理時間を初期値(全体の処理時間)にセットします。
4) t が0になるまでのしばらくの間、以下の処理を繰り返します。
    a) x の一部を逆転させ新しい経路 x' を作ります。
    b) もし $f(x') \leq f(x)$ 新しい経路の方が今の経路より短いなら
        i) x <– x'
        無条件で今の経路をその新しい経路へ更新します。
    c) もし $f(x') > f(x)$ 新しい経路の方が今の経路より長いなら
        i) ΔE <– $f(x')$ – $f(x)$
        どのくらい長くなったかを求めます。
        ii) T <– $(t \div t_0)^6 T_0$
        今の温度を求めています。 T_0 から始まり、残りの処理時間が少なくなるにつれて徐々に温度が下がっていきます。残りの処理時間が0になる瞬間に温度は0になります。
        iii) $exp(-\frac{\Delta E}{T})$ の確率で x <– x'
        SA法の最もキーとなる処理です。「一定確率で居心地のいい場所から脱出する」動作を再現しています。
    d) もし $f(x') < f(x^*)$ これまでに見たこともないほど短い経路が出たら
        i) x^* <– x'
        最良経路の記録を更新します。
    e) t から、この繰り返し処理1周にかかった時間を引きます。

以下のQRコードより、この処理の実装コード(GitHub)にアクセスできます。本アプリはこのコードで動作しています。

さいごに

最後までお読みいただきありがとうございました。ささやかではございますが次のページに特典を用意致しました。是非ご覧ください。

参考文献

1. Optimization by Simulated Annealing, S. Kirkpatrick; C. D. Gelatt; M. P. Vecchi, Science, New Series, Vol. 220, No. 4598. (May 13, 1983), pp. 671-680., https://pdfs.semanticscholar.org/e893/4a942f06ee91940ab57732953ec6a24b3f00.pdf, 2019.10.27アクセス
2. Some New Test Functions for Global Optimization and Performance of Repulsive Particle Swarm Method, Sudhanshu K. Mishra, (August 23, 2006), https://papers.ssrn.com/sol3/papers.cfm?abstract_id=926132, 2019.8.17アクセス
3. 結晶成長の物理I-融液からの理想的結晶成長-, 斎藤幸夫, https://www.jstage.jst.go.jp/article/materia/49/7/49_327/_pdf, 2019.8.15アクセス
4. 鉄の特性を引き出す熱処理の科学, 日本製鉄, https://www.nipponsteel.com/company/publications/quarterly-nssmc/pdf/2017_19_14_17.pdf, 2019.8.16アクセス
5. 回復、再結晶および粒成長, 名古屋大学工学部物理工学科, http://www.numse.nagoya-u.ac.jp/P6/kobashi/img/file25.pdf, 2019.8.16アクセス
6. 鋼の熱処理基礎知識, 島根大学, http://www.ecs.shimane-u.ac.jp/~shutingli/MDE14.pdf, 2019.10.27アクセス
7. 宇宙に星はいくつあるの？-名古屋大学「宇宙100の謎」プロジェクト, http://www.a.phys.nagoya-u.ac.jp/100nazo/nazo/2-47.html, 2019/10/27アクセス
8. ＮＡＳＡ、銀河の数を２兆個と発表　従来推定の１０倍に-CNN News, https://www.cnn.co.jp/fringe/35090534.html, 2019/10/27アクセス

特典：TSPの閉じた経路の数を厳密に計算する

コンピューターは階乗などの、巨大数を扱う計算は基本的にできませんが、一部のソフトウェアでは巨大数をサポートしていたりします。今回は、GoogleColaboratoryというPython実行環境にてTSPの閉じた経路の数を厳密に計算してみました。

N個の点がある時、閉じた経路の数は$N! \div N \div 2$通りありますが、**2019個**の点がある場合を計算してみました。桁数は5,795桁です。計算結果は次のページにあります。

473345975395530667051044477598035040353832690632326471596176913406337128656709235591292903924358396260539731
641357765535041659163091482018517537543019162094793562877798571596971534917463181330646540375205795503190870 7
349697367153821590205698833403747172211790016559523114279387681683509313011222598771219037054111060170417236 7
942641374499225611228641901434326336669587172260438000298636971912133992321905604693736566639748046074357167 4
667180853769962429384946733241955441111087633316577510753407965365358001601805283584705592032750282113962696 7
728815271404930972358285546724135417034064925093524220687562281124156744377049848949126364216806776568829944
521233924618618483190693972166082087152915199017035727463062666823836077433923596698946790450470293047342713 4
785447858196665113677612769097469932392366938387449837739274344019654161781125317504746613241562650147062704 2
401424467017620029624391927209216343408744370027685479206458438176521536911229046429087191741705928974898304 0
967915731648258692360344180526618086090092485648317966673409388735183555836725355076942691766188076526225568 1
066604920782096509385010000622221980960133682891101009948822130379936493045991222816778361552884026175461897 7
389575192327701861523452027131768268947000480687927258967321256667005043987494570184626595980427364228610720 7
579403262552868585616604328274573879895980611943371057016933097609156752312482436369364333793016862117200771 1
676504049039908861230346787001232551140885645669370506519503311360772936302778907717514068746790264728729568 3
857756531288291267637321095835060467147814416510361055839895083522297223736559907665244451254595972449556810 4
039067116741546555711753389982627302976076699537017702897190421642578675837217343503727124204180321638658154 5
321230463337683921799593978981735792608559122307072691020447421483603746387997352707273762463111517663633871 4
233435325912051936965175724928963649749177693419084290453143141081529208553889836998752178965264175995933067 8
341433704062011249169244430501765301528129024235376390336122969872653114436602839225093586648997789395869194 5
809188088563150437359226798922671958243307393496028550802469293813009882697665173743525077859594825206386226 2
810418824939604192153279526332437834837536414587020984566228181570333991181865958220588495893019500383757518 9
324318122989375505218099178384136570664933965995208974174609432686055717256604661492190214303027913895246575 6
859552641302491153501863391016532751563581471143344915652672817730476027011078344966614993111347179518800110 9
451250113029205424530180519434726663347931078374590254190263820473958688486923634583556169643966633835431232 5
037956090025617701825181134288458721842543098373315908860666663559110942625610267566101825023553373492060695 0
142300385783868802562847355419349434979950625190231477012612760334128597657587918042293306942612829121578066 2
386217839658628681908875246729939569120618293661633994272267272470304516651313711013213266835609404187283435 9
728165175522179185954241591792847259625530033249427469054254731063173109410129490338077022370879149437544163 9
679529356446480099653214424751961608396566788589172218896705180932661353624799641313443065621830116349915652 2
690777333365209777507929850393027576769424063394921339797377597890697291070870353551444318513874292887773900 6
569647529337971493683367532144395925388137623617852161499929039035341953234706524477857626676121254435435674 1
573254733231798313097946041274148789825094336929105782974255691643847096527475077264880130793952734011701912 4
832246197144777207277164226318768310771040670081636118204899425680520201271767892604379287602785898152134461 8
845235048422440103250966572935317180885057406665532616227634541340700525208888759269800262955938971924098739 6
242908715897530282426203975879361612892905034010461184241903757616184947002576250077641092324590098171760507 2
259787601787590879330664028851073751190965011656218036818393935368714350498961323337857826592306379773156495 4
117167189757029088519040935874382560592942971910298927101021537418771229112214574169453540218942529184982757 3
298019456737755252318675273669823235452197406161485771080262260510437144916388402447962758473264340030546454 8
397663436664360496256931031988027942512463342440497606309589341980238166217903507669587208351676485338145536 0
839930762692240398010615789080481642817360076361370205232241900556519011223535893860679040701820730580794712 0
891575729074875866657469195572582574477921380198631317213246362576364352628740351529250455637567652041349489 1
837054839151109213071827780807217277173098908241212088969111302707513275963472105500818082488637497240920077 9
693921524231561793879795782414988726502057731654628931820518585846661747881247956928397289561360177448884332 0
510936977612819817911367655109327957491728278376034338476532694829282317615912804690661033351279052594581809 2
075375654944359940376339589560211347508715832833677613057561077441395974536266493982458090498726036301318346 0
387994374843937904697685468121668118949905510579671454511461020436021243416903789617295881412043397082746839 7
584952570119532613757965894195196553510518181906841198481456030262833701072541110646990270474045706286773544 1
310633102054269112990347128635244012846330661024348358292654255331128253982614167452291874021144800031644275 6
055031704860571104087879542344794988122400124632729524581171200000000000000000000000000000000000000000000000 0
000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000 0
000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000 0
000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000 0
000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000 0
000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000 0
000000000000000000