流域界データの自動的再構成手法

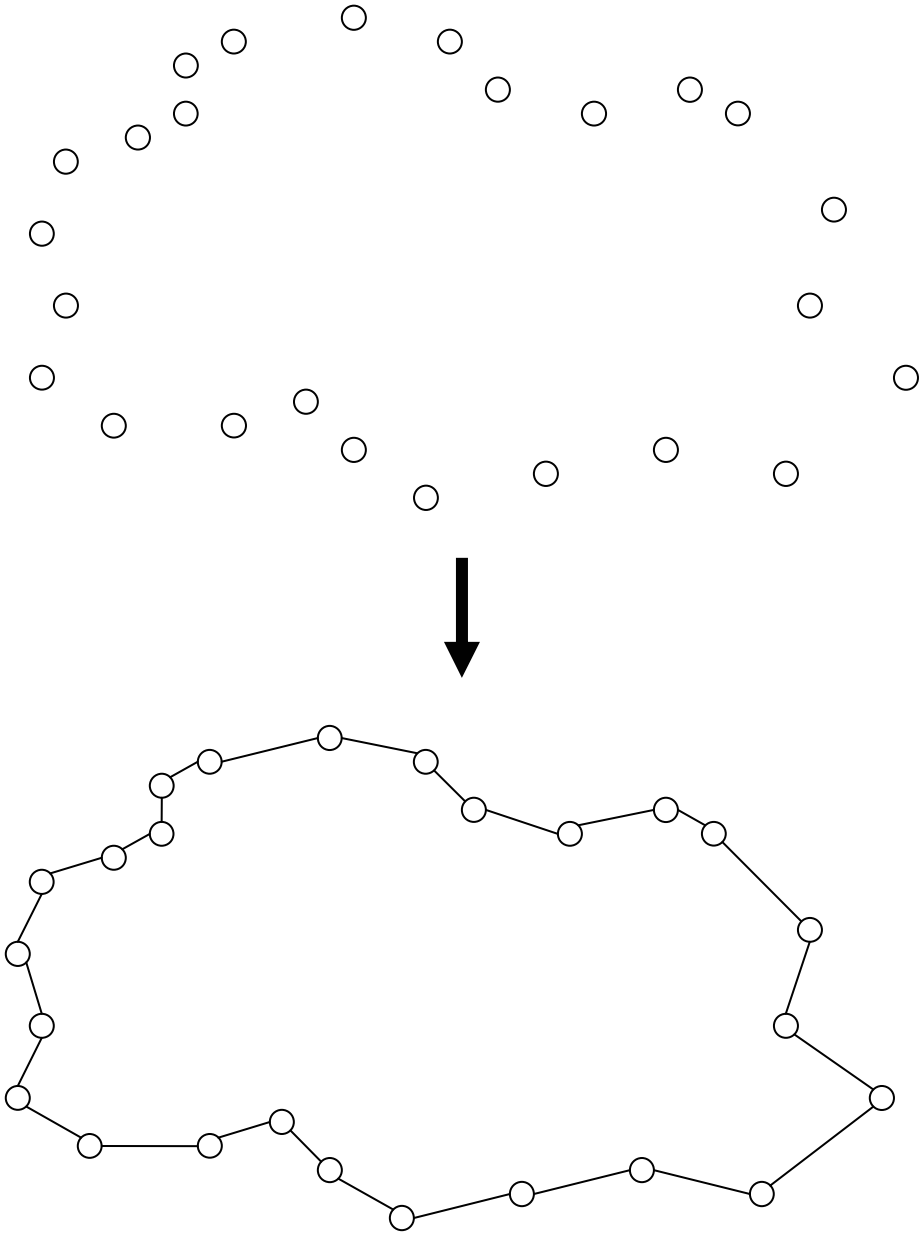
計算機科学講座　中村　吉郎(s98464-H)

指導教官　田中　章司郎　教授

目次

１．はじめに

国土地理院の流域界のデータをプログラムでできる限り正確に再構成を行いたい。この場合、どのような手法を用いたものが最も元のデータに近づけるのだろうか。この研究では実際にいくつかの手法を用いてある流域界のデータの再構成を行い、その結果について考察を行う。用いた手法は次で説明する。



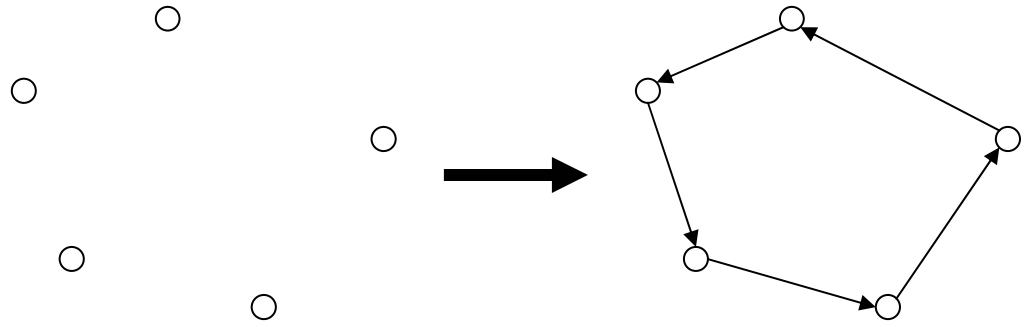
このように存在する座標データを正確にまとめあげたい。

２．再構成手法の紹介

ここでは、上で述べたように研究で用いた流域界データの再構成手法について簡単に紹介する。大きく分類すると、以下のように３つに分けられる。

１）貪欲法

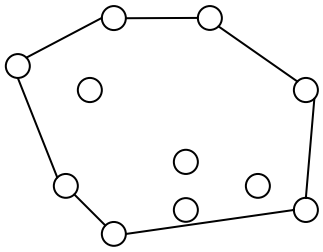
まずは最も簡単な方法としてあげられるものが、最短距離どうしをつないで１周するまで続けてデータの再構成を行うというものである。これを貪欲法(Greedy search)と呼んでいる。



始点を１つとり、ただ最短のものどうしをつなぐ

２）凸包アルゴリズム

凸包アルゴリズムとは、その名のとおり凸包に関係するアルゴリズムで凸包の頂点を求めるアルゴリズムのことを指している。凸包についてはあとで説明する。



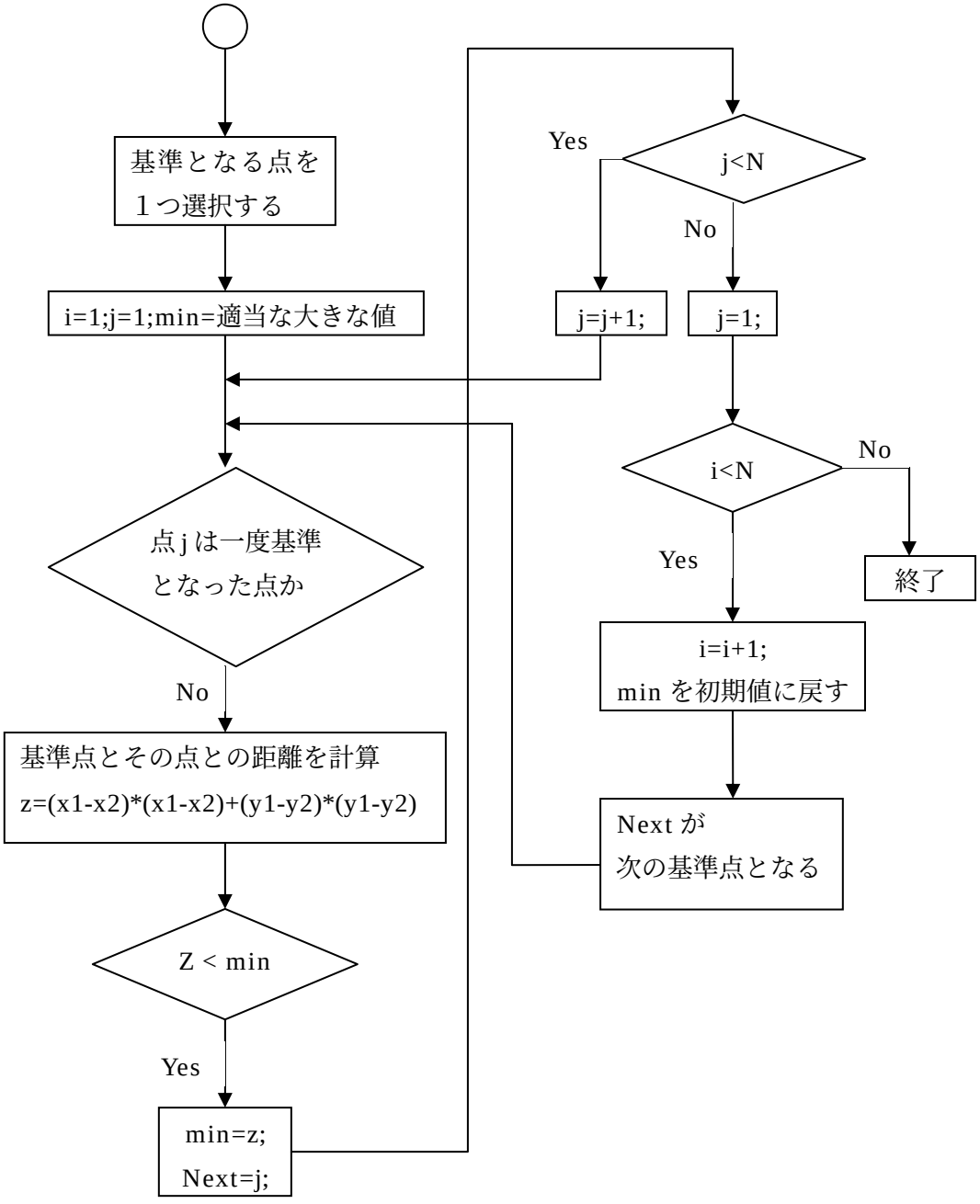
凸包

３）巡回セールスマン問題

N ある都市全てを通過してもとの都市に戻るとき、その総距離が最短となる巡回路を求める問題のことである。

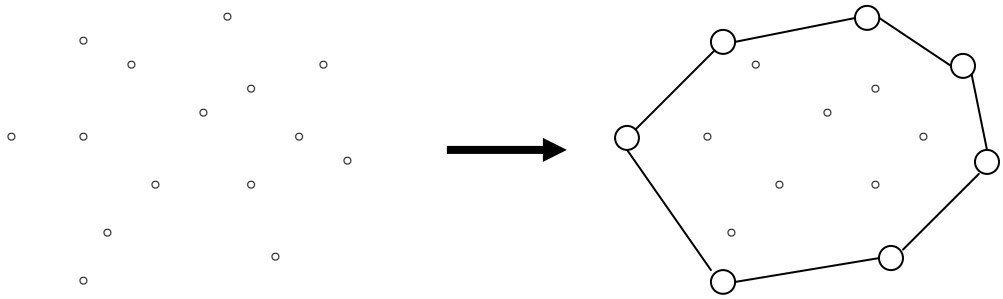
３．貪欲法

貪欲法のアルゴリズムは以下に示すとおりである。

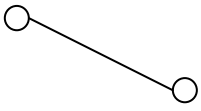


４．凸包アルゴリズム

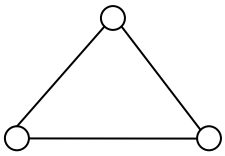
凸包とは、与えられた点列を全て含んだ最小の凸多角形のことをいう。凸包アルゴリズムとは、この凸包の頂点を見つけ出すためのアルゴリズムのことをいう。つまりここでは与えられたデータを凸包にすることでデータの再構成を行おうということである。注意しておくべきことは、凸包は凸多角形であるという性質上、必ず点が３つ以上必要であるということである。



左の図の凸包が右の図である。
図のように全ての点列を含んでいる最小の凸多角形が凸包である。

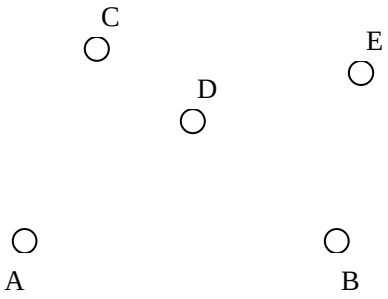


点が２つ以下では
凸包は作成できない

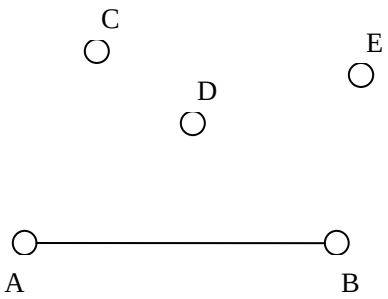


三角形は最小の凸包である

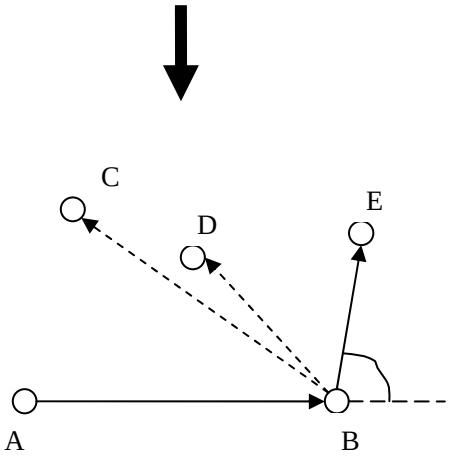
凸包アルゴリズムには包装法とグラハム法があるがここでは包装法でおこなう。包装法とは、確実に凸包の頂点となる点を１つとり、そこからもっとも角度が小さくて、その中で最も遠いものを次の点とする。という方法を繰り返し、始点自身にかえってくるまで同じ処理を繰り返すというものである。角度を測る方法としては、内積を用いる。内積で cos の値を求める。cos は１８０度以下の場合、角度が大きくなるほど cos の値は小さくなるので、それを利用する。つまり、cos の値が最も大きいものが角度の一番小さいものであり、凸包の次の頂点である。



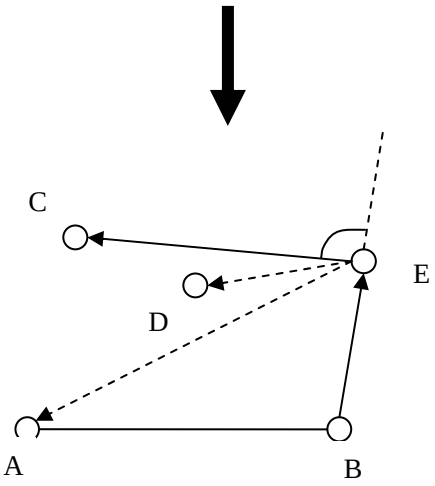
では実際に凸包アルゴリズムを使って凸包を作成してみる。左の図は全部で５つの点が存在している。これで凸包を作成する。



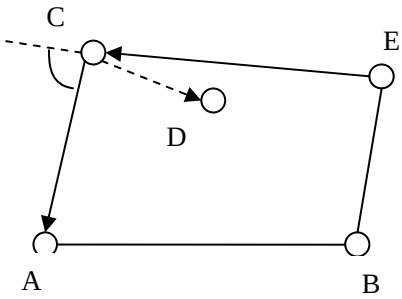
まず、確実に凸包の頂点になる点を１つとる。ここでは点 A をとる。そして、凸包の次の頂点として点 B をとる。(理由は後で説明する)



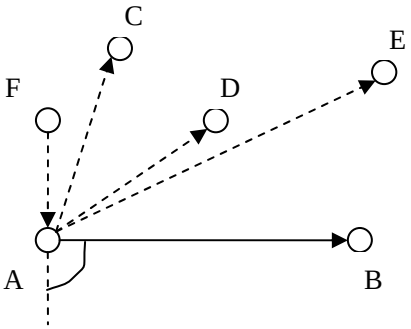
ここでは点 B の次に凸包の頂点となる点を求める。候補となるのはまだ選択されていない点 C、点 D、点 E である。ここではまず直前の点 A と点 B を直線でつなぎ、点 B から点 CDE にも線を延ばし、それらの角度を測り、角度が最小のものを凸包の次の頂点として選択する。ここでは点 E が該当する。



次の点を求める。さっきと同様、角度を計測して最小のものを次の頂点とする。ただ今回はいまだに選択されていない点だけではなく始点である点 A も次の凸包の頂点候補として選択する。その結果、点 C が頂点になる。

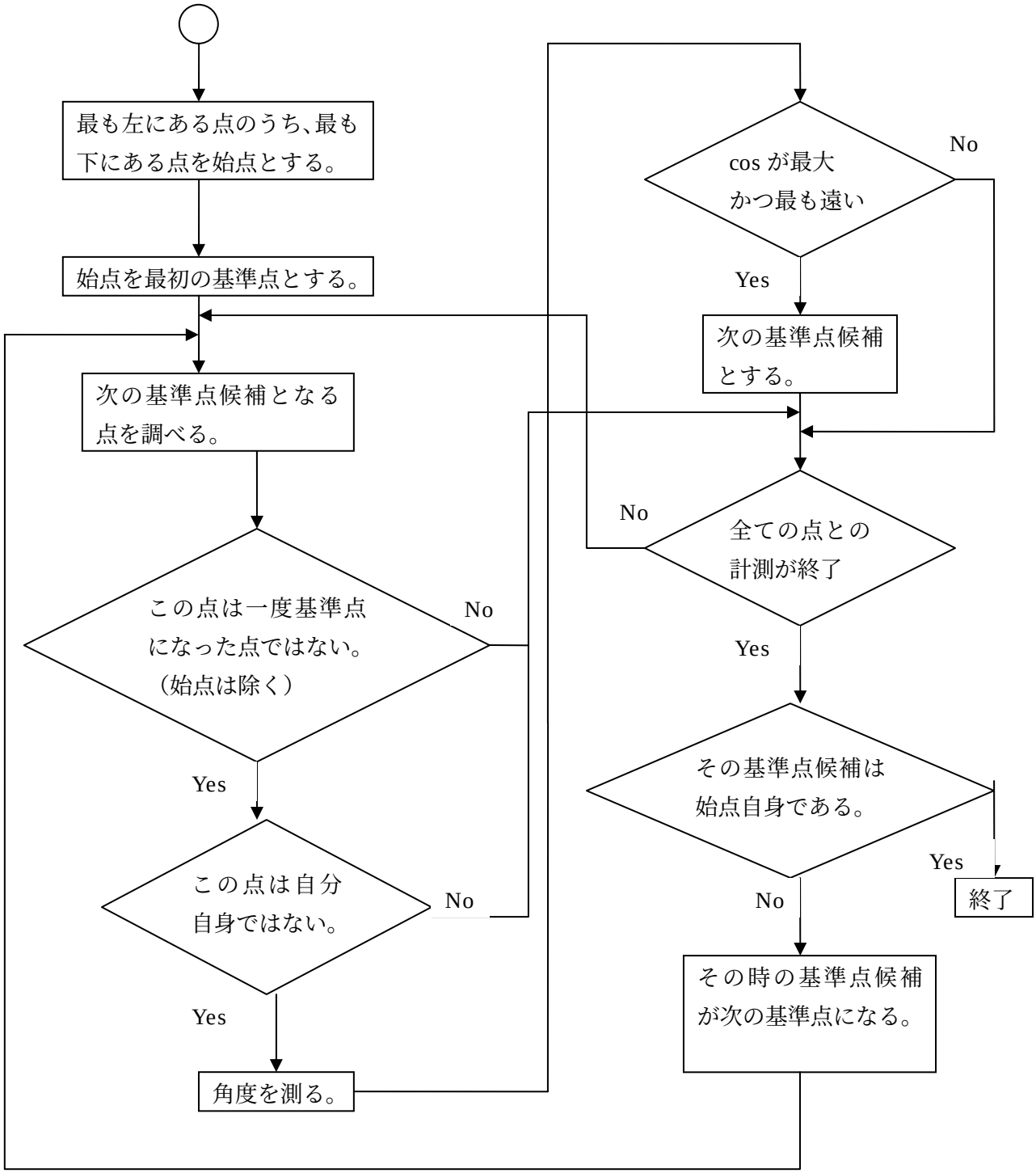


次の頂点を求める。残った頂点候補は点 AD の 2 つだが同様の操作を行うと始点の点 A が該当する。これで 1 周したことになるので凸包の完成である。残った点 D は、凸包に含まれる点であり、頂点でないことがわかる。



なお最初に直前の点がないのにもかかわらず点 A から点 B へと移動したのは、仮に直前の点として点 F を設定しておくことにより、次の点を点 B と判別することができる。

凸包アルゴリズム（包装法）のアルゴリズムは以下のとおりである。

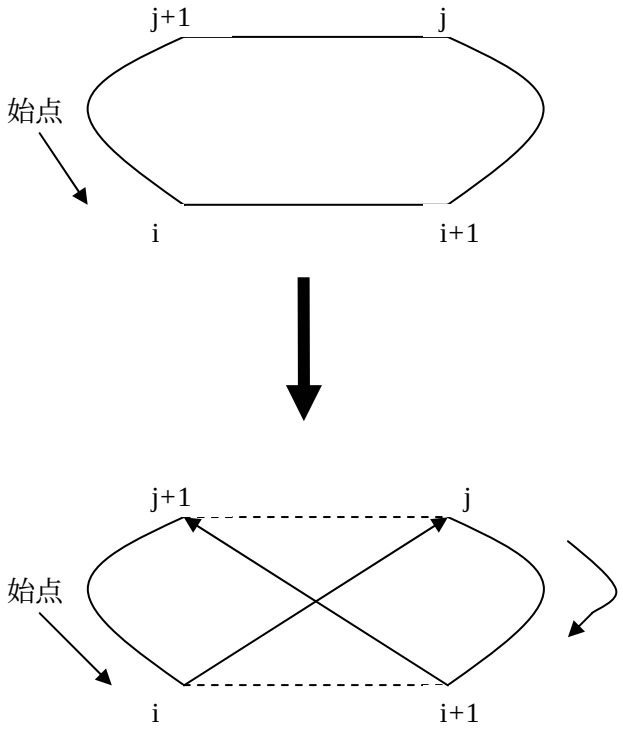


５．巡回セールスマン問題

先に述べたとおり、巡回セールスマン問題とは、N 個の都市全てを通過して元の都市に戻るとき、その総距離が最短となる巡回路を求める問題のことをいう。

この問題を解くための手法は何通りか考え出されてはいるが、今回はまず貪欲法のときに作成したデータをもとに近傍処理を行い、それを用いて解を求めた。

近傍処理は２近傍を用いた。２近傍とは任意の２箇所を組み替えたものである。

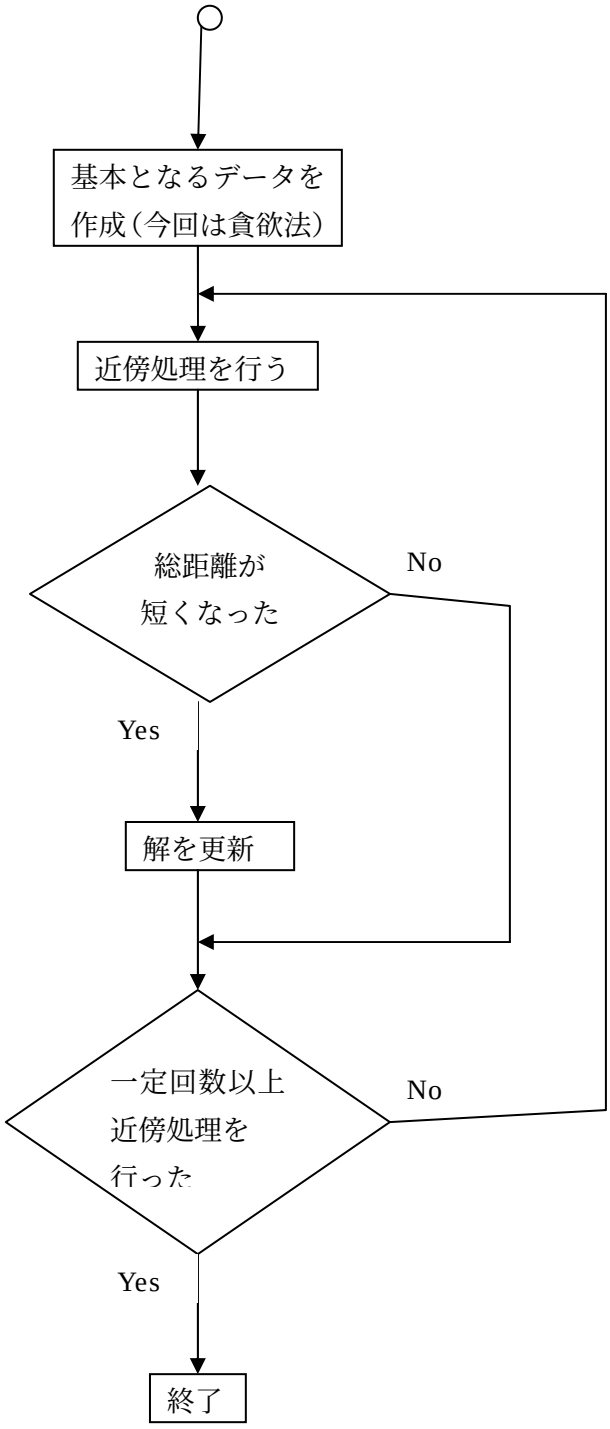


左の図は始点から出発すると i,i+1,j,j+1 番目の点を通過し、一周してまた始点に戻ってくる。

また、i と j は任意のものでいいが、隣接していないものであること。

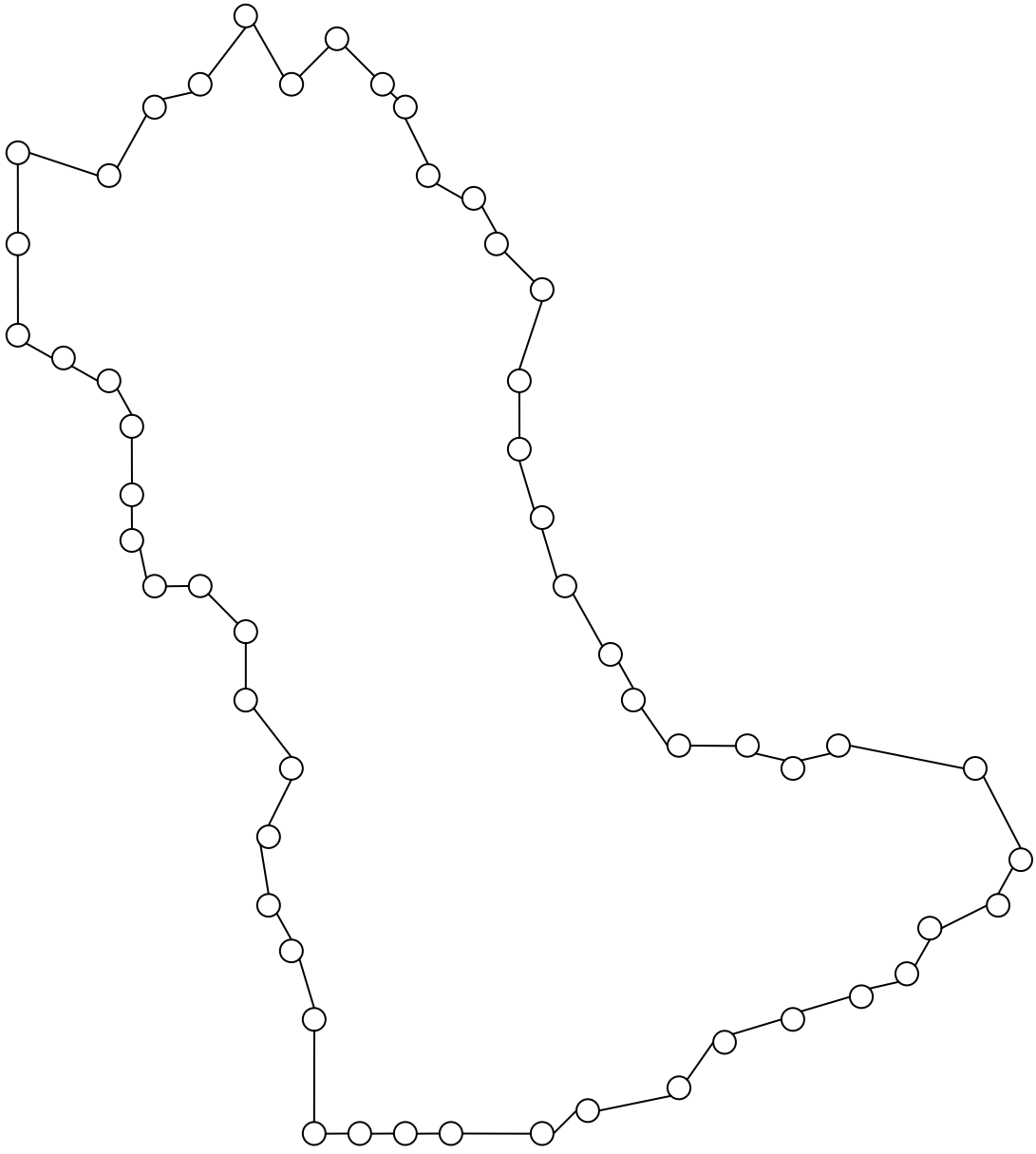
では、この図に近傍処理を行ってみることにする。この場合、i 番目と j 番目、i+1 番目と j+1 番目をつなぎかえればよい。

巡回セールスマン問題のアルゴリズムは以下のようになる。



６．結果

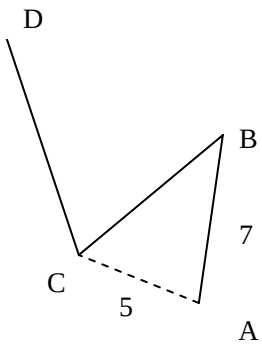
下の図は今回３つのアルゴリズムを試したデータである。



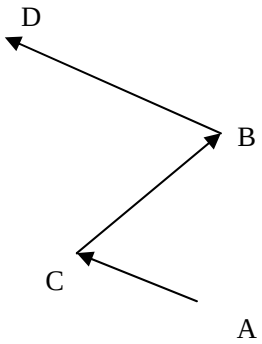
まず貪欲法から結果を述べる。前にのせたフローチャートのようなアルゴリズムでプログラムを組んで流域界のデータの入れ替えをおこなった。

結論をいうと大部分は順番通りになったが一部順番通りにならない部分がでてきてしまった。このような場合の解決方法だが、このプログラムでは最短距離同士をつなぐだけという単純なアルゴリズムをそのまま実行させただけのものであるので、コンピューターにこの結果がエラーであるかどうかを認識させることができない。仮にプログラムにエラーであることを認識させることができたとしても、それをもとどおりの正しい場所に修正することができない。なぜなら無理にデータを修正してしまうと、最短距離となった点が次に進むべき場所であるというこのアルゴリズム自体を否定してしまうことになってしまう。そのため、このアルゴリズムはまるで必要のないものとなってしまう。よって、貪欲法では完全に再構成することは不可能である。

例：

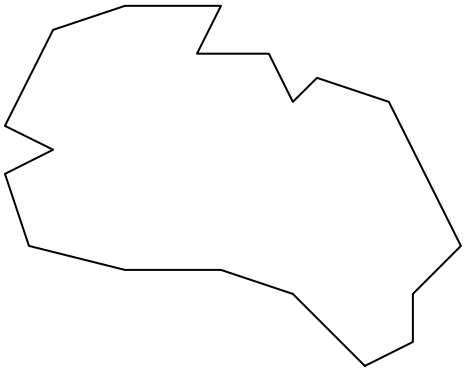


ABCD の順に並んでいるとする
AB の距離　7
AC の距離　5　とすると

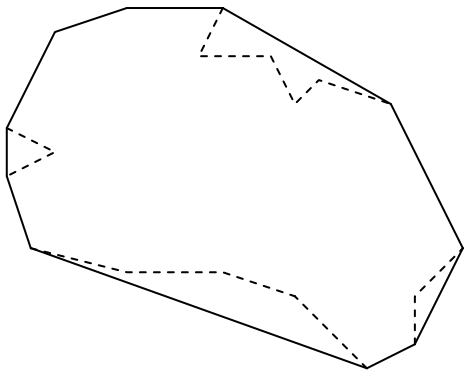


結果として、左の図のように誤った順序で構成されてしまうことがわかる。

凸包アルゴリズムで求めた結果は、予想以上に元のデータと一致しなかった。おおまかな形は似てはいるものの、かすかに内側にへこんでいるようなケースが多く、当然その場合は凸包の頂点にはならない。よって、一応すべてのデータを用いている貪欲法より元のデータと一致しなくなってしまう。

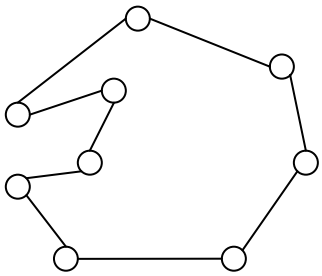


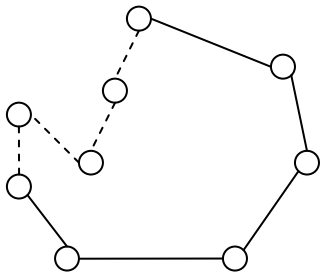
例として左のような座標のデータがあったとする。この図から凸包を作成すると下の図のようになる。



このように内側にある座標は全て凸包の頂点にはならないため、座標の損失が大きく正確な再構成はできない。

巡回セールスマン問題では、結果として貪欲法より総距離自体は短くなったが、貪欲法の時と同じで距離の短いものが正しい道順であるとは限らない。よって、有効であるとはいえない。





左の図が正しい道順であるとする。右の図が巡回セールスマン問題ででた答えとする。右の図は、総距離自体は左の図より短いが、結局正解から外れてしまっている。

全体的なまとめとして、これらの手法ではある程度までならば似せることはできるが、それ以上元のデータと一致させることはできないことがわかった。自然界のデータは今回程度のアルゴリズムでは再現できない。これ以上先に進むためにはもっと高度な計算が必要となるだろう。

引用・参考文献

1. http://www.al.cs.kobe-u.ac.jp/~sakaki/research.html
2. http://www.ss.u-tokai.ac.jp/~kikyo/cgeo/