解説

—近似アルゴリズムに関する最近の話題—

巡回セールスマン問題の近似アルゴリズム：天才アローラによる20年ぶりの急進展

玉木久夫／明治大学

まえがき

巡回セールスマン問題(Traveling Salesman Problem，以下TSPと略します)は，組合せ最適化問題の中でも一番よく知られている問題ではないでしょうか？「n個の都市があり，都市の対ごとの距離が表として与えられているとき，すべての都市をちょうど一度ずつ訪れて最初の都市に戻る巡回路で長さが一番短いものを求む」という問題です．距離の与え方は任意でもよいのですが，今日の話では三角不等式を仮定しておきます．つまり，都市Aから都市Bへ行くのに，第三の都Cを経由して行く道の方が直接の道よりも短くなることはないということです．

まさか，現実のセールスマンがこの通りの問題の解を必要とする場面はまずないでしょうし，このままの形に定式化できる実用的な問題はそれほど多くないのですが(プリント基盤の穴開けの順番を決める問題はその数少ないうちの1つです)研究の対象として人気のある問題です．さまざまな現実的問題がこの問題を複雑化した変種として定式化できるため，組合せ最適化問題のモデルケースと見なされるのでしょう[5),6)]．この問題に対する理論的な研究に最近大きな進展があったのでそれを解説します．計算量や近似可能性の理論的側面から見た解説になりますので，TSPについての研究の全貌を知りたい方は古典的なテキスト[5)]を読まれるようお薦めします．また，TSPに関連した実用上の問題については本号の加藤先生の解説を参照してください．

この解説ではアルゴリズムの理論についての基礎知識はまったく仮定しないため少し前置きが長くなります．NP完全性や近似アルゴリズムの概念になじみのあるかたは，次の節は飛ばしていただいてよいと思います．

難しい問題と近似アルゴリズム

TSPを解く一番素朴なやり方は，n個の都市の回り方をすべて調べてその中で一番よいものを選ぶというものでしょう．始点の都市の選び方がn通り，次に訪れる都市が$n-1$通り，...ということで回り方の数は$n!$あります(**図-1**参照)．これは2^nのような指数関数よりもさらに速く大きくなる数で，nが20ぐらいでも全数検査は絶望的です．もっとうまい方法を考える必要があります．

ちょっと脱線してTSPとよく似た**最小全域木問題**という問題を考えてみましょう．この問題では，TSPと同様にn個の都市と各都市間の距離が与えられています．これらの都市の間に道路を引いて，どの都市からどの都市へも行けるようにしたいのですが，道路の総長を最小にするにはどうしたらよいかというのが問題です．**図-2**(a)のように道路をつないでできる輪があるような場合は1本道路を除いて総長を短くできますから，この問題の答えは必ず**図-2**(b)のような輪のない形になります．グラフ理論の言葉では，このように輪を持たずにすべての点をつなぐような構造を**木**と呼びます．点と点を直接つなぐ道路は**辺**と呼びます．木の形につなぐとしてもいろいろなつなぎ方があります．**図-2**(c)のように1列につなぐ場合(これも木です)だけ考えても$n!$通りありますから，すべてのつなぎ方の数はずっと大きくなることは容易に想像できるでしょう．したがって，しらみつぶし法ではTSPの場合よりもさらに長い時間がかかってしまいそうです．

ところが実際は最小全域木問題はずっと速く解けるのです．脱線ついでにその考え方を簡単に見てみましょう．まず，2都市の対でその間の距離が最小であるようなものに注目し，これを都市A，Bとしましょう．距離が最小になる対が複数あってもかまわないのですが，ここでは簡単のためにそのような同着1位はないものと仮定します．そうすると，「最小全域

木のなかでは必ずこの2都市A，Bが直接辺で結ばれている」ということが証明できます．証明は省きますが，それほど難しくないので興味のある方は考えてみてください．この定理を使えば，まず1本必ず引かなければいけない辺が求まります．次のステップでは，結んだ都市AとBを合わせて1つの都市のように考えます．この複合都市と第三の都市Cとの距離は，ごく自然にAとBの近いほうからCまでの距離として定義します．そうすると，最初のステップと同様にしてまた1本必ず引かなければならない辺が求まります．このようなステップを$n-1$回繰り返すと，全部の都市がつながって最小全域木ができあがります．1つのステップでは距離が最小になる対を選ぶのが主な仕事で，対の数はn^2以下ですから素朴にプログラムを作っても，全体でたかだかn^3に比例する計算時間があれば十分ということになります．このことを，「最小全域木は$O(n^3)$時間で求まる」と表現します．$O(n^3)$というのは，$5n^3$以内とか，$100n^3$以内とか，とにかくnに依存しない定数cがあってcn^3以内であるような量を表す表現です．定数が何であるかを問題にしないので，時間の単位は時間でも秒でもマシンサイクルでも何でもよいのです．さて，この$O(n^3)$時間というのはかなり荒っぽい話で 工夫すればもっと速く解くことができるのですが，それはアルゴリズムの教科書に譲ることにしましょう．

$O(n^3)$時間というのは，しらみつぶし法の指数関数的な時間と比べると圧倒的な改善です．たとえば，もし10都市の問題が1ミリ秒で解けたとすると，1000都市の問題は100^3ミリ秒=1000秒以内で解けることになります．

それでは，TSPでも同じような工夫によりたとえば$O(n^3)$時間で解くことができないのでしょうか？　いや，一歩も二歩も譲って，

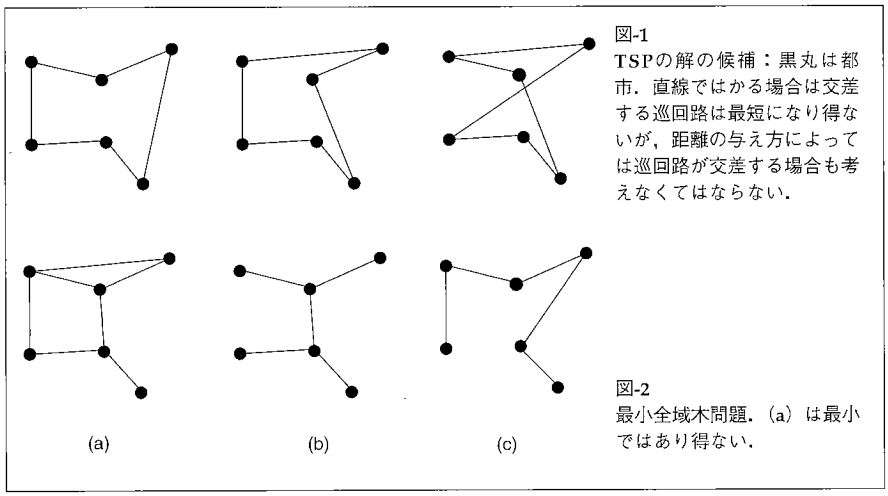

図-1
TSPの解の候補：黒丸は都市．直線ではかる場合は交差する巡回路は最短になり得ないが，距離の与え方によっては巡回路が交差する場合も考えなくてはならない．

図-2
最小全域木問題．(a) は最小ではあり得ない．

$O(n^5)$時間でも，$O(n^{10})$時間でも，あるいは$O(n^{100})$時間でもよい，とにかくcを何か定数として$O(n^c)$時間で解く方法はないのでしょうか？　この問いは「TSPは多項式時間で解けるか？」と表現されここ30年程計算機科学の理論家たちを悩ませてきた大問題です．

「多項式時間で解けるか」というのは随分遠慮した質問です．たとえ答えがYESであったとしても，その多項式というのがたとえばn^{100}であったら実用上お話しにならない計算時間といわねばなりません(たとえば10^{100}がどれくらい大きな数か考えてみてください．)．この遠慮した質問に対してさえ答えがどうもNOではないかと思われている問題がたくさんあり，今考えているTSPもその1つです．これらの問題はNP完全問題と呼ばれ，

> そのうちの1つでも多項式時間で解けるならば，他のNP完全問題のどれもが多項式時間で解けてしまう．

という性質を持っています．つまり，すべてのNP完全問題が多項式時間で解けるか，どれも多項式時間では解けないか，2つに1つです．さまざまな分野から生じたNP完全問題の多様性にもかかわらず多項式時間の解法は一様に難しく，多くの天才達の努力もむなしくまだ発見されていません．そのことから，どうもどのNP完全問題も多項式時間では解けないだろうというのが支配的な見方です．

以上をまとめると，TSPは$O(n^3)$時間はおろか$O(n^{100})$時間でも解けそうもないということになります．それでは，たとえば100都市のTSPを解くことはあきらめなければならないのでしょうか？答えは「問題による」です．TSPを解くというのはπの計算と同様に世界記録を争うオリンピック的あるいはギネスブック的な競技になっていて，TSPLIBと呼ばれる標準的な問題集があります．この問題集の中にある今まで解けなかった問題例を解くと記録に名を残すわけですが，現在では数百都市から数千都市の例が勝負になっています[1]．これは，この問題集に載っている例が割合自然なものであるためで，たとえば1000都市の例が解けたプログラムであっても意地悪く作ったものであれば数十都市の例にも歯がたちません．けれどもよっぽど運が悪くなければ現実の場面でそういう意地悪な例に遭遇することはないだろうという楽観論は1つの行き方です．

もう1つの行き方は，正確に最短な巡回路でなくてもよいよ，近似解でよいよという行き方です．たとえば，真に最短な巡回路の1パーセント増しぐらいの長さの巡回路であればたいていの場合は満足できるでしょう．こういうふうに要求をゆるめると問題はだいぶやさしくなります．解の候補は

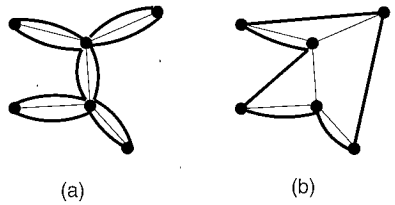

図-3
最小全域木を用いたTSPの2倍以内の近似解

指数関数的にたくさんあるのですが，十分よい近似解の個数も普通は莫大なので，その中の1つを見つけるのはそれほど難しくないというわけです．ここでも意地悪な例はあまり現実的には生じないという楽観的な立場にたった技法(ヒューリスティクス)を駆使するとTSPLIBにあるような数万都市の問題でも解けてしまいます．ヒューリスティクスとしては，TSPに特有のものもありますしまた一般の組合せ最適化問題に適用できる，いわゆるメタヒューリスティクスもたくさんあります[4]．

この「近似解はやさしい」という命題は理論的に裏付けられているのでしょうか？　たとえば，「最適解の1.1倍以内の長さのセールスマン巡回路を求めよ」という問題は多項式時間で解けるのでしょうか？　これが今日のお話しの本題です．

まず「最適解の2倍以内の近似解を求めよ」という問題ならば簡単に解けます．これは，最小全域木問題を解いてできた木に沿って都市を回っていけばよいのです(**図-3**(a))．よほどへまをしないかぎり，木を構成している各辺は往復1回通るだけですみますから木の辺の総長の2倍以内の長さの巡回路ができます．同じ都市を2度訪れることが気になるならば，2度目の訪問はスキップするように巡回路を修正します(**図-3**(b))．三角不等式の仮定から，このような修正によって巡回路の長さが長くなることはありません．一方，最小生成木の辺の総長はTSPの最適解の長さ以下のはずです．なぜならば，TSPの解の方がもし短かったらその巡回路から辺を1本をとりさって，さらに総長の短い木(この場合点を1列につないでいる)を作ることができるからです．したがってこの方法で求まる巡回路はは最適解の2倍以内の長さであるということが保証されます．

Christofidesは1976年の論文でこの方法にさらに工夫を加えて，必ず最適解の1.5倍以内の近似解を求めるアルゴリズムを考案しました．これ自体非常に面白いのですが，紙数の都合で省略します．以来20年間この1.5という数は多くの努力にもかかわらず改良されませんでした．つまり，1.1倍はおろか，必ず1.49倍以内の近似解を求めるという保証のある多項式時間のアルゴリズムは見つからなかったのです．そこに登場したのがAroraです．

Aroraの結果は都市間の距離が直線距離で計られるという特別な場合を対象としています．こういう場合のTSPを幾何的TSPと呼ぶことにします．三角不等式を満たしさえすればよいという一般的な場合と比べると，幾何的TSPでは点集合の幾何学的な性質が使えますから問題がやさしくなっても不思議はなさそうです．にもかかわらず，幾何的TSPに対してもChristofidesの1.5倍以内の近似よりもよい方法はArora以前には見つからなかったのです．Aroraの結果によると，幾何的TSPに対する，最適解の1.1倍以内の近似解は多項式時間で求めることができます．もっと一般にεを任意の正定数とするとき，最適解の(1+ε)倍以内の近似解を多項式時間で求めることができます．たとえば1.0001倍の近似もOKということで，Christofidesの1.5倍に比べると飛躍的な前進です．

この「多項式時間」をもう少し詳しくいうと，Aroraの1996年の論文では$O(n^{10/\varepsilon})$ぐらい，それに続くArora自身の1997年の論文[3]では$O(n(\frac{\log n}{\varepsilon})^{100/\varepsilon})$ぐらいになっています．理論的に見るとこれは驚異的な急進展です．というのは，$(\frac{\log n}{\varepsilon})^{100/\varepsilon}$という関数は$n$を大きくしていくと，たとえば$n^{0.01}$よりもいずれは小さくなってしまうので，$O(n(\frac{\log n}{\varepsilon})^{100/\varepsilon})$はほとんど$O(n)$と見ることができるからです．したがって，「幾何的TSPを0.1%の誤差で解くことはほぼ入力に比例する時間以内で解ける」ということができます．

いや，「そんなのは理論家のつく嘘だ」という人もいるでしょう．$(\frac{\log n}{\varepsilon})^{100/\varepsilon}$という関数の実際の値は$n$=3に対してもすでに天文学的な数になるからです☆．一般にアルゴリズムの理論的結果について似たような場合がたくさんあるために理論研究に不信感を持っている人も多いかもしれません．そういう疑問や不信に反論するかわりに，この解説では理論的な壁を越えるためにいかに豊かで美しいアイディアが生み出されるかをAroraの仕事にじかに語ってもらおうと思います．そして，たとえ$O(n(\frac{\log n}{\varepsilon})^{100/\varepsilon})$時間という結果自体が実用的な意味を持たずとも生み出されたアイディアは実用上の可能性を秘めたものであることも．

アローラによる超高精度近似アルゴリズム

まず問題の仮定と結果を整理しておきましょう．都市の数はnで，そのすべてが1つの平面上に置か

☆ 対数の底はこの解説では常に2です．

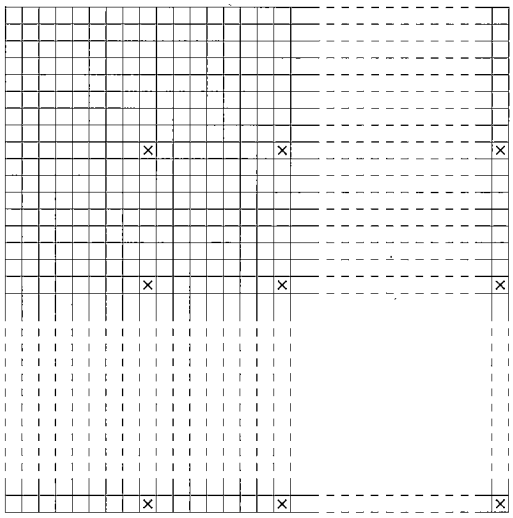

図-4
正方枠の格子化：1辺の長さ $L=2^k$

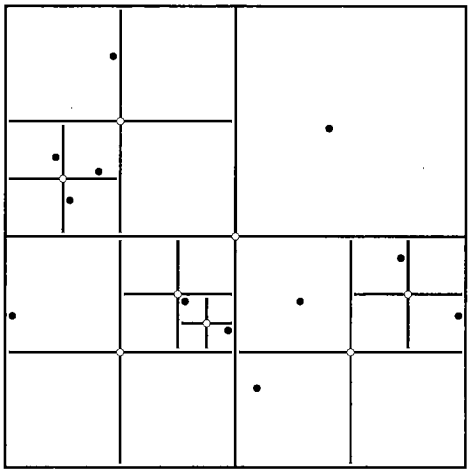

図-5
4分木分割：白丸は分割の中心を表す

れているとします．3次元やそれ以上の次元の空間にも結果は拡張できるのですが，ここでは平面に話を限ります．2つの都市の間の距離は直線距離で計ります．近似解の精度ε>0は入力として与えられているものとします．アルゴリズムは最短の巡回路の(1+ε)倍以内の長さの巡回路を求めなければなりません．具体的に考えたい人は以下でεが出てきたら常に0.1を代入してみてください．これから説明するのはAroraの2つ目のアルゴリズムで，その計算時間は $n(\frac{\log n}{\varepsilon})^{O(\frac{1}{\varepsilon})}$ 以下です．$O(\frac{1}{\varepsilon})$ という表現が肩にのっているので戸惑うかもしれませんが，これは「$(\frac{100}{\varepsilon})$ といってもよいのだけれど，この100は解析をがんばれば小さくできるかもしれないし，その値自体にはあまり意味はない，だからぼかしてただ $\frac{1}{\varepsilon}$ の定数倍以内といっておこう」というほどの意味です．ここでやる解析では分かりやすさを優先するので，100より大きい定数が出てきます．

このアルゴリズムは乱数を使ってふるまいを決める「確率的アルゴリズム」で，出てきた答えの精度が要求を満たさない(つまり，最短の巡回路の(1+ε)倍より長くなってしまう)可能性があります．でも，そのような失敗の確率は1/2以下であることが保証されているので，たとえば10回アルゴリズムを実行してみて一番よい結果をとることにすれば，10回全部失敗する確率は非常に小さいので心配いらないのです．

解析の便宜上，都市の配置について次のような仮定を置きます．まず，すべての都市は1辺の長さが$L=2^k$の正方枠の内部にあり，しかもこの枠を間隔1の横線と縦線で格子に区切ったときに，各都市を置ける位置は縦座標と横座標が共に8の倍数であるような小正方形の中心のみとします．**図-4**で×印が都市を置いてもよい位置です．随分強い仮定に見えるかもしれませんが，Lを大きくしてやると(つまり格子の網目を細かくしてやると)任意の都市の配置がこのような配置によって十分よく近似できます．具体的には都市の配置が与えられたとき，まず全部の都市を含むできるだけ小さい正方枠を描きます．次に$80n/\varepsilon$と$160n/\varepsilon$の間で2^kの形をした数Lを選び，枠を$L\times L$の格子に切って，格子の間隔を長さの単位と定めます．つまり，枠の1辺の長さがLになります．最後に各都市を一番近い×印まで動かします．この移動によって生じる巡回路の長さの増減はどんなに多く見積もっても1都市あたり16(もとの巡回路上の都市の位置から移動先の位置まで片道の長さがたかだか8の寄り道をすると考えます)，合計$16n$以下です．一方，正方枠のとり方から，巡回路の長さ自体は少なくとも正方枠の1辺の長さの2倍，すなわち$160n/\varepsilon$はあります．つまり，このような都市の移動によって生じた巡回路の長さの相対誤差はε/10で，全体で誤差をε以下におさえるという我々の目標からは十分に無視することができます．この，都市を置く位置についての制約から，次の命題が出てきます．

> **命題1**：最適な巡回路の長さをTとするとこの巡回路が格子線を横切る回数は$2T$以下である．

格子の間隔が1であることに注意して，都市と都市を結ぶ直線ごとに考えれば納得できるでしょう．都市と都市の距離が十分離れているという仮定がこの命題のために必要なことを確認してください．

準備は整いました．アルゴリズムの説明にはいりましょう．

アルゴリズム：動的計画法

これから説明するアルゴリズム自体にはそれほど目新しいアイディアがあるわけではありません．なぜこのアルゴリズムでうまくい

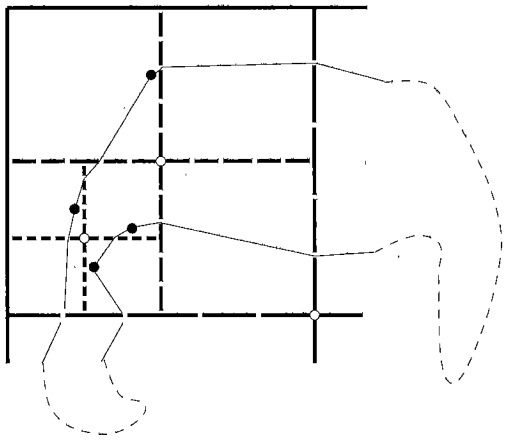

図-6
抜け穴：m＝4の例

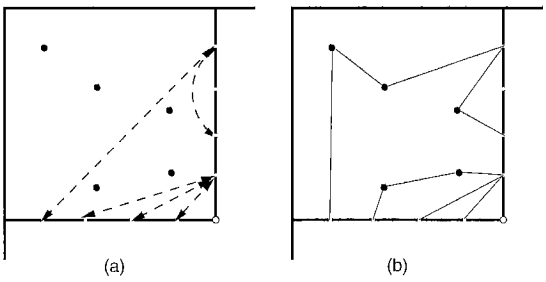

図-7
左上の正方形に対する1つの部分問題（a）とその解（b）

くかを説明するのが次節で述べる構造定理で，そちらの方が核心なのですが，まずアルゴリズムを理解しないと構造定理のご利益がピンときません．

アルゴリズムはまず上で説明した格子枠に対して，4分木分割と呼ばれる再帰的な分割を行います（**図-5**）．全体の正方枠をまず4つの正方形に分け（第1レベルの分割），その1つ1つに対してまた4分割を施す（第2レベルの分割）…ということを繰り返し，正方形のなかにある都市の数が1または0になったらその正方形はそれ以上分割しません．枠の1辺が$L=2^k$という2の冪乗の形なので，分割線は必ず格子線の上に載ります．第1レベルの分割で使う4本の分割線それぞれの長さは2^{k-1}，第2レベルの分割で使う分割線の長さは2^{k-2}，というように半分ずつ短くなって第iレベルの分割線の長さは2^{k-i}になります．このことは後で重要になりますので記憶にとどめてください．

この4分木分割に基づいて，巡回路の候補を一種のしらみつぶし法で調べるのですが，調べる巡回路に制約を設けて調べる範囲を減らすことを考えます．1つの制約は分割線との交わり方で，**図-6**のように各分割線に何カ所か抜け穴を開けて巡回路はこの抜け穴のところだけで分割線を横切ることができるとします．抜け穴の開け方は各分割線の上では等間隔で，その個数は分割線の長さによらないとします．この個数の定め方は後で述べますが，とりあえずmと呼ぶことにしておきましょう．図-6はm=4の例です．このような制約を加えると，最短の巡回路の長さが長くなってしまい解に誤差が生じますが，この誤差が目標のεより十分小さくなるように，穴の間隔を十分小さくしてやる，すなわちmを十分大きく決めるのです．もう1つ，大事な制約は巡回路が1つの分割線を横切る回数がある定数r以下だとすることです．このrの選び方も後で述べます．rを小さくすると，調べなければならない巡回路の候補数が減って計算時間の上では有利ですが，調べる範囲が狭くなるために目標の解の精度を達成できないおそれがあります．

この2つの制約を満たす巡回路を「やさしい巡回路」と呼ぶことにします．アルゴリズムはやさしい巡回路の中で一番短いものを求めます．このようなやさしい巡回路だけに候補を絞って考えてもよいということを保証するのが次節の構造定理です．

アルゴリズムは動的計画法と呼ばれる方法の典型的な例ですが，ここでは動的計画法の予備知識を仮定せずに説明します．我々の，最適なやさしい巡回路を求める問題をまず，4分木分割の第1レベルの分割で得られる4つの正方形に対応する部分問題に分解して解くことを考えます．つまり，左上，右上，左下，右下のそれぞれの正方形に対する最適解を組み合わせて全体の最適解を求めようとするのです．しかし，「左上の正方形に対する最適解」を求めるとき残りの正方形のことを忘れて左上の正方形だけに都合のよい解を求めると，残りの正方形もそれぞれの都合があるわけですから，4つの解をつなぎあわせてうまく1つの解にすることができなくなってしまいます．そこで，あらかじめつなぎめのところは，巡回路が分割線を横切る横切り方のすべての可能性を考えてそのすべて場合のそれぞれに対して，各正方形における部分問題を 考えます．**図-7**(a)は左上の正方形に対して考えなければならない部分問題を1つ示しています．1本の矢線は巡回路からこの正方形によって切りとられる1本の経路の出入口を指定しています．「このような出入りの指定を満たしつつ，左上の正方形内の都市をすべて通るような5本の経路で長さの和が最短なものを求めよ」というのが，この部分問題の意味です．**図-7**(b)はこの部分問題に対する解を示しています．

巡回路が1つの分割線を横切る回数がr回以下と仮定していますから，左上の正方形に対して考えなければならない部分問題の個数は有限です．そのすべてに対して解が求まっていて，さらに残りの3つの正方形に対しても部分問題の解がすべて求まっているとします．そうすると，全体の問題に対する答えはこれらの部分問題の解の組合せから求めることができます．具体的には，4つの正方形のそれぞれに対して部分問題を1つずつとり，つなぎめの部分のつじつまがあっていたら，それぞれの部分問題の解をつなぎ合わせて1つの巡回路を作ります．そのようなつじつまのあう部分問題の組合せはたくさんありますが，すべてを調べてその中で一番短い巡回路を答えとして選びます．

上では，4つの正方形に対する部分問題がすべて解けていると仮定しましたが，これらはどうやって解くのでしょうか？　同じように，各正方形を4分して(第2レベルの分割)，それらにたいする部分問題に帰着するのです．これを続けていくと，前に述べた4分木分割に対応した部分問題の階層ができます．動的計画法のポイントはこの階層の下のほうからボトムアップに解いていくことにあります．そうすることにより，同じ部分問題を何回も繰り返し解くことが避けられるからです．階層の一番下にある部分問題は都市をたかだか1つしか含まない正方形に対するものなので，ごく簡単に解けます．

このアルゴリズムの実行時間はどのぐらいでしょうか？　まず，解かなければならない部分問題の個数がどのくらいかを考えましょう．4分木分割に現れる1つの正方形に注目すると，この正方形に対する部分問題は何通りあるでしょうか？　各分割線を巡回路が横切る回数についての仮定から，巡回路がこの正方形を出入りする回数は$4r$回以下です．したがって，部分問題を指定する矢線の本数は$2r$以下としてよいことになります．矢線の両端はたかだか$4m$個の抜け穴のどれかを指すわけですから，部分問題の指定の仕方は荒っぽく見積もって，$(4m)^{4r}$通り以下となります．それでは，1つの正方形Sに対する部分問題をすべて解くのにはどれぐらいの時間がかかるでしょうか？　これは，その下の部分問題がすべて解けていると仮定しての話です．この正方形の4つの部分正方形S_1，S_2，S_3，S_4に対する部分問題はそれぞれたかだか$(4m)^{4r}$個なので，S_1に対する部分問題を1つ，S_2に対する部分問題 を1つ，S_3に対する部分問題を1つ，S_4に対する部分問題を1つ選んできて組み合わせるやり方はたかだか$((4m)^{4r})^4=(4m)^{16r}$通りです．この組合せをすべて調べれば，正方形Sに対するすべての部分問題に対する解が求まるわけです．1つの組合せを調べる手間はたいしたことがないので，ちょっと不正確ですが，1つの正方形に対する計算時間は$O((4m)^{16r})$であるとしておきましょう．4分木分割の中に現れる正方形の数は$O(n \log n)$なので(考えてみてください)結局，このアルゴリズムの計算時間は$O(n \log n(4m)^{16r})$でおさえられます．

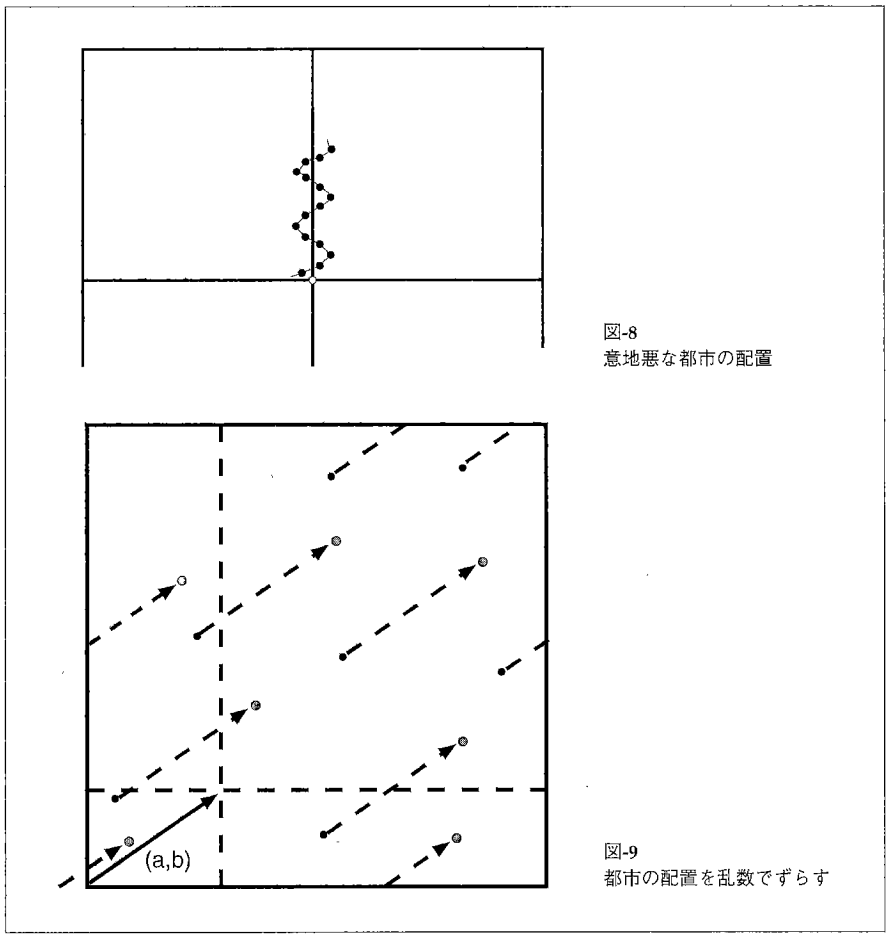

図-8
意地悪な都市の配置

図-9
都市の配置を乱数でずらす

構造定理：乱数の魔術

いよいよ核心の構造定理です．構造定理の言わんとすることは，「$m=\frac{10}{\varepsilon}\log n$, $r=\frac{12}{\varepsilon}$ と設定すると前節で定義した意味での“やさしい巡回路”で最適解の$(1+\varepsilon)$倍以内の長さのものが存在する．」ということです．したがって，mとrをこの値に設定して前節のアルゴリズムによりやさしい巡回路の中で最短のものを求めれば近似の目的が達成されることになります．実行時間の見積もり$O(n \log n(4m)^{16r})$にこれらのm，nの値を

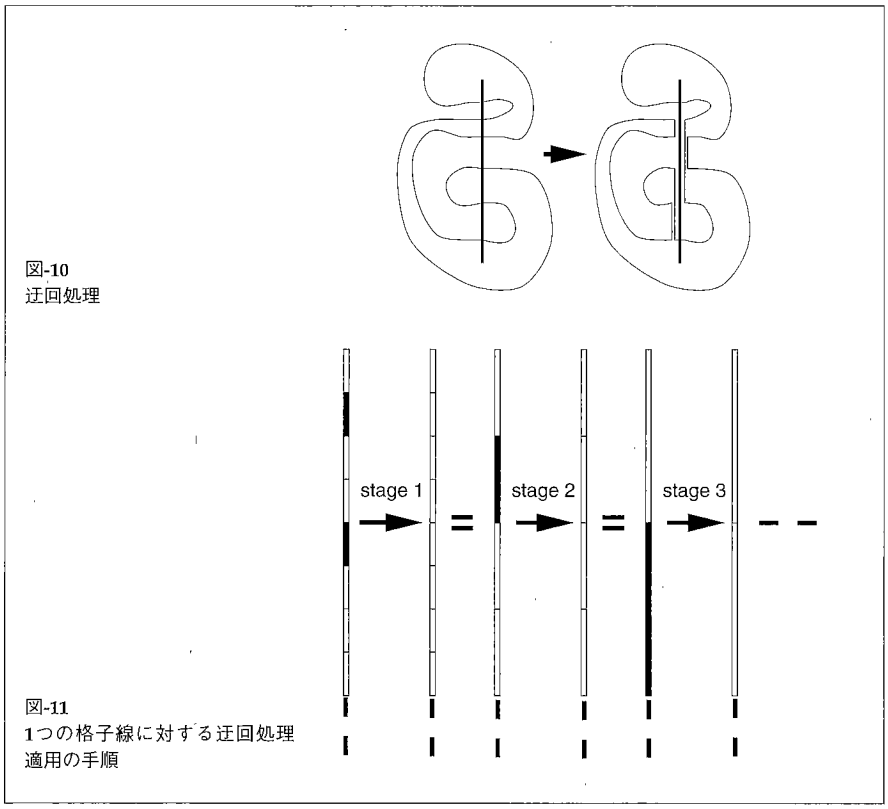

図-10
迂回処理

図-11
1つの格子線に対する迂回処理適用の手順

代入すると約束通り $O(n(\frac{\log n}{\varepsilon})^{O(\frac{1}{\varepsilon})})$ になります．

ところが，残念ながら上のままの形では構造定理は成立しません．神様は，**図-8**のように第1レベルの分割線に沿って都市を意地悪く配置するかもしれません．そうすると分割線ををたくさん横切らないとよい巡回路ができなくなってしまいます．Aroraは，乱数を使って意地悪な神様を出し抜くことを考えました．やり方は簡単で，$-L/2$と$L/2$の間の整数a, bをランダムに選び，**図-9**のようにすべての都市を一様に横にa，縦にbだけずらすのです．枠の外に出てしまった都市は座標にLを足すか引くかして枠の中に戻します．こうすると，意地悪な神様が第1レベル(あるいはそれに近いレベル)の分割線のために用意しておいた都市の配置が低い(第1レベルから遠い)レベルの分割線，つまり短い分割線の付近に落ちる可能性が高くなります．そうなれば，分割線を巡回路が横切る回数は1つの分割線あたりでは少なくなるので困りません．都市を動かしたので，問題を変えてしまったように思われるかもしれませんが，もとの枠がどこにずれたかはもちろん覚えておいて，もとの枠に当たる格子線のところは横切れない(図で破線のところ)，そうでないところは横切れる，ということをアルゴリズムがきちんと管理するようにすることは難しくありません．

こういう乱数化の考え方は最近のアルゴリズムの設計にはごく普通に使われていて，それ自体はさほど珍しくありません．私が感動したのは，これでうまくいくという証明の美しさです．$r = O(\frac{\log n}{\varepsilon})$ でよいならば，だれでもここまでくれば証明できるのですが，$r = O(\frac{1}{\varepsilon})$ にしても十分近似できるという証明は見事というほかありません．この感動を伝えたい一心で書いています．できれば最後までおつきあいをお願いします．

証明は，最適なTSP巡回路が与えられたとして，それをやさしい巡回路に変換する方法を示すことにより行います．そのときに巡回路の長さが伸びてしまうわけですが，その増加分がもとの巡回路の長さのε倍以内であるようにしなければなりません．変換は，まず(1)各分割線を横切る回数がr以下であるようにし，次に(2)分割線を横切る場所は抜け穴だけであるようにする，という順に進みます．(2)の部分は比較的単純なので，(1)の部分を主に説明します．

巡回路の変換用の基本的な手続きとして，次のような「迂回処理」を用います．これは，巡回路がある線分をr回以上横切っているときに，巡回路をつなぎ換えて横切る回数を2回以内にするというものです．**図-10**のように，線分の両側に迂回路を加えてこれを実現します．図から「この目的を達するためのコスト(巡回路の長さの増分)はいつでも線分の長さの3倍以内にできる」ということを納得してください．証明は省きます．

1つの格子線に着目したとき，この格子線に対する迂回処理は**図-11**のような手順で進んでいきます．この図のなかで，巡回路がr回以上横切っている線分は「汚い」線分として黒く塗って示してあります．黒い線分が白い線分に変わっているのは，迂回処理を施したということです．第1ステージでは，長さ1の線分に区切ってそれぞれに対して，もし「汚ければ」迂回処理を施します．第2ステージでは，長さ2の線分に区切って同じことを行います．第1ステージでは「きれい」であっても2つまとめたために「汚い」ということは当然あり得ます．何番目のステージまで処理を続ける必要があるかどうかは，この格子線の上にどのレベルの分割線が載るかによって決まります．第1レベルの分割線が載るのであれば，分割線の長さは$L/2=2^{k-1}$になるので第$(k-1)$ステージまで続ける必要があります．反対に第kレベルの，つまり長さ1の，分割線が載るならば第1ステージで終わりにしてよい理屈です．乱数化によって都

市を移動するとき，この格子線も一緒にずれると見なすべきなので，この格子線にどのレベルの分割線が載るかは確率的な事象であることを注意しておきましょう．

さて，もとの最適な巡回路の長さをTとしましょう．アルゴリズムの説明の前に述べた命題1から，この巡回路が格子線を横切る回数は$2T$回以下です．迂回処理を1回施すたびに格子線を横切る回数はすくなくとも$r-2$減りますから，迂回処理の回数は全部で$2T/(r-2)$以下です．rはかなり大きい数を考えていますから，この回数は$2T/r$以下であるといってもまあよいでしょう．

それでは，1回の迂回処理のコストはどれぐらいでしょうか？つまり，1回の迂回処理をすると，巡回路の長さはどれぐらい増えるでしょうか？　これはステージによります．第1ステージならば，迂回処理を適用する線分の長さが1なのでコストは3以下です．すべてのステージにおいて同じコストならば，万々歳，全体のコストは$3\cdot 2T/r=\varepsilon T/2$以下で，もとの巡回路の長さの$\varepsilon/2$倍以内におさまります(こうなるように$r=12/\varepsilon$を選んでおきました.)．しかしながら，ステージが進むにつれて，線分の長さが倍々になるため1回の迂回処理のコストの上限値も倍々になっていきます．さてどうするか？

いよいよ大団円です．あるステージの迂回処理が実際に実行されるかどうかは乱数に依存していることを思い出してください．第1ステージの迂回処理は必ず実行されなければなりません．第2ステージの迂回処理は，その格子線が一番底の第kレベルの分割線に当たったときは実行しなくてよいので確率1/2で実行されます．同様にして，ステージが進むにつれて実行される確率は半分半分になっていきます．したがって，1回の迂回処理のコストの期待値はステージにかかわらず一定値3(！)でおさえられることになります．コストの合計の期待値は，迂回処理の回数が$2T/r$だから$\varepsilon T/2$以下となります．ある正の量の実際の値がその期待値の倍以上になってしまう確率は1/2以下ですから，確率1/2以上で変換による巡回路の長さの増加はεT以下になります．

(2)の，抜け穴だけを通るようにするための処理の解析にも似たような期待値の議論を使いますが，こちらは省略します．この部分の処理にかかるコストのために，上の議論の定数を少し修正しなければなりませんが，本質はかわりません．

類似の問題

TSPのように，距離の決め方に制約がないと難しいが幾何的な距離に話を限ると近似解が(理論的な意味で)うまく解けるという例はほかにもあります．Aroraの論文[3]にはTSPに対するのと同じ技法によって解ける類似の問題がいろいろ出ています．名前だけ挙げると，(すべて幾何的な)最小シュタイナー木問題，k-最小木問題，k-TSP問題などです．

また，私達はTSPと似た問題で少し毛色の変わったk-巡回路被覆という問題を考えました[2]．これは，TSPのようにすべての都市を尋ねるのですが，k個の都市を尋ねたら必ず決まった都市(基地)に一度戻ってこなければならないという条件がついています．kの値は入力で指定します．経路の総長を最小化したいという目的は同じです．おおまかにいって，$k=O(\log\log n)$ならば精度εの近似が多項式時間でできることが知られていたところを私達は$k=O(\log n)$でも精度εの近似が多項式時間でできることを示しました．その解法について解説(宣伝)したいのですが，紙数の都合で省略します．Aroraの技法だけではなぜうまくいかないか，興味のある方は考えてみてください．

おわりに

解析によって得られたAroraのアルゴリズムの実行時間の見積もりは，実用的観点からはばかげたほど大きいものです．解析のいくつかの場面で最悪を想定しているためこうなりますが，現実の例に対しては構造定理がずっと小さいmとrの値に対して成立している可能性は十分あります．アルゴリズムにしても，動的計画法を正直に実行しないさぼったインプリメンテーションがいろいろ考えられそうです．理論的な飛躍のために使われたアイディアが現実の場面にも活用されることが大いに期待されます．

参考文献

1) Applegate, D., Bixby, R., Chvá , V. and Cook, W.: Finding Cuts Int the TSP (A Preliminary Report, Report No. 95-05, Center for Discrete Mathematics and Theoretical Computer Science (DIMACS), Rutgers University, Piscataway, NJ (1995).
2) Asano, T., Katoh, N., Tamaki, H. and Tokuyama, T.: Covering Points in the Plane by k-tours: Towards a Polynomial Time Approximation Scheme for General k, in Proc. 29th ACM Symposium on Theory of Computing, pp.275-283 (1997).
3) Arora, S.: Nealy Linear Time Approximation Schemes for Euclidean TSP and Other Geometric Problems, in Proc. 38th IEEE Symposium on Foundations of Computer Science, pp.554-563 (1997).
4) Johnson, D.S. and McGeoch, L.A.: The Traveling Salesman Problem: A Case Study in Local Optimization, in Local Search in Combinatorial Optimization, E. H. L. Aarts and J. K. Lenstra (editors), John Wiley and Sons, Ltd., pp.215-310 (1997).
5) Lawler, E.L., Lenstra, J.K., G. Rinnooy Kan, A.H. and Shmoys, D.B.: The Traveling Salesman Problem, John Wiley & Sons, Chichester (1985).
6) Reinelt, G.: The Traveling Salesman Problem: Computational Solutions for TSP Applications, Lecture Notes in Computer Science 840, Springer-Verlag, Berlin (1994).

(平成10年4月4日受付)