第3回最適化ゼミ

指導 渡邉 (D1)　　佐野 (M1)
担当　チーフ：花田　サブチーフ中村，實田

1　離散最適化問題の概要

1.1　組合せ問題とは

離散最適化問題は，目的関数や，制約条件が離散的な点の集合である最適化問題のことである．離散的な問題に，組合せ最適化問題というものがある．

組合せ最適化問題とは，いろいろなものの組合せの中で，与えられた制約条件を満たし，かつ，目的関数を最小化するものをみつける問題でである．組合せ最適化問題には TSP[1]，スケジューリング問題，最大流問題に代表されるネットワーク上の最適化問題，およびナップザック問題のような組合せ最適化問題の解の効率的列挙などがある．ここでは，TSP，スケジューリング問題について紹介し，それらを解くためのアルゴリズムにはどのようなものがあるか簡単に紹介する．

1.2　TSP

TSP とは，Fig. 1 のように，都市数とそれぞれの都市間の道のりが分かっているとき，セールスマンが N 個ある都市を必ず 1 度だけ通って巡回したとき，総経路長を最短にするような都市の巡回経路を見つける問題である．都市数が N 個のとき，考えられる巡回経路の種類は，N 個の要素の最適な並び替え（順列）であるので (N-1)！/2 個である．例えば，N=10 の場合，約 18 万だが，N=20 となると約 6×10^16 となってしまう．このことから分かるように，N が大きくなると，この組合せは爆発的に多くなる．このように現象を組合せ爆発という．この場合もすべての巡回経路を調べることは実際には不可能で，都市の配置に法則性がなければ厳密解を求めることはできない．

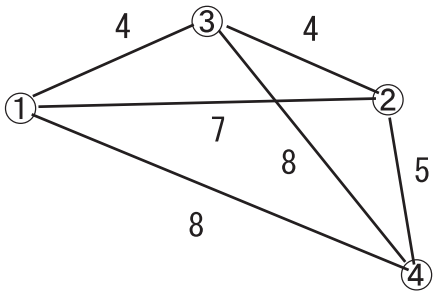

Fig. 1　TSP

TSP のバリエーションとして，球面 TSP，非対称 TSP がある．

- 球面 TSP
  都市が球面上に配置されていると考える問題. 平面上では, もっとも右上（左上）に位置する都市ともっとも左下（右下）に位置する都市は, 直線距離のコストが大きいが, 球面では逆に小さい場合もある.

- 非対称 TSP
  都市 a から都市 b に行く時と, 都市 b から都市 a に行く時で巡回経路が異なる問題. 一般的には, 都市 a と都市 b の直線距離に重みをかけた値をその都市間コストとし, 巡回の方向によって異なる重みを設定する. 一方通行用の道路でつながれた都市間の道で，異なる道のりと考えるとイメージしやすい.

[1]Traveling Salesman Probrem，巡回セールスマン問題

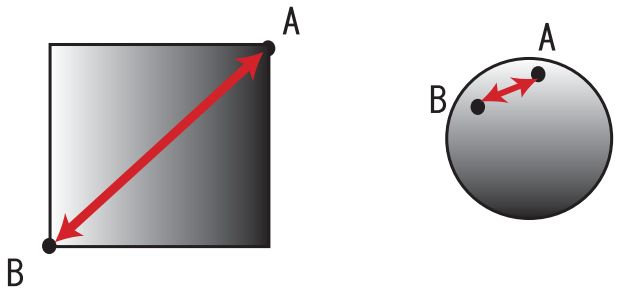

Fig. 2　球面 TSP のイメージ

1.3　スケジューリング問題

1.4　生産スケジューリング問題

生産活動を円滑に行うためには，製品を作る行程での作業の所要時間や順序関係を考慮しなければならない．その実際の生産加工にあたって，どの生産設備（機械）で，いつ加工するかという，個々の要素作業のための具体的で詳細な時間日程が必要となる．その最適ないし実行可能な詳細実施プログラム（スケジュール）を決定することを「生産スケジューリング」と呼ぶ．この中に，ジョブショップスケジューリング問題がある．

1.4.1　ジョブショップスケジューリング問題

ジョブショップスケジューリング問題とは，Fig. 3 で表されるように複数の作業の直列な接続によって接続されており，資源である機械を用いて製品（ジョブ）を加工する順序を，いくつかの評価基準に基づいて決定する問題である．このとき，資源を同時に使用することはできないものとされる．

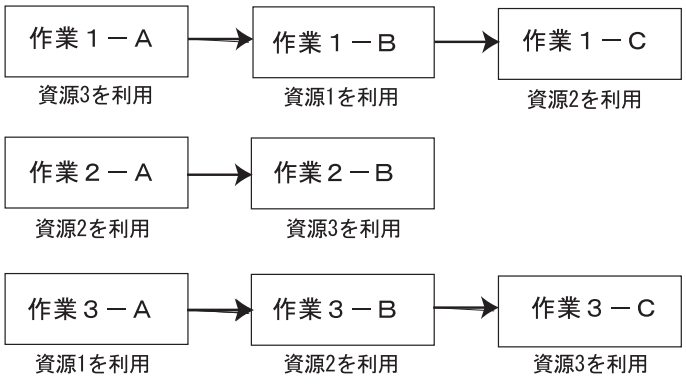

Fig. 3　ジョブショップ・スケジューリング問題

ジョブショップ・スケジューリング問題のうち，Fig. 4 に示すようにすべてのジョブが同じ順に資源である機械を占有するような問題をフローショップスケジューリング問題と呼ぶ．この問題は，現実の工場では，もっともよく存在するものであり，コンベアを用いた流れ作業などがこれに相当する．

1.4.2　プロジェクトスケジューリング問題

プロジェクトスケジューリング問題は，ジョブショップスケジューリング問題をさらに拡張した問題で，工場におけるスケジューリング問題のみではなく，建築工事や製品開発などより広い適用範囲をあげることができる．この問題では，行程に属する作業が直列である必要がなく，分岐や合流があってもかまわない．また，利用する資源は同時に複数の作業で共有することが可能であるものとする．プロジェクトスケジューリング問題は，PERT（Problem Evaluaton and Review Technique）によって一般的に扱われる．

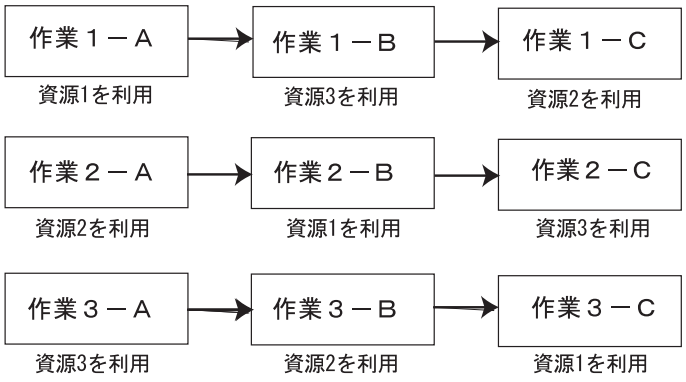

Fig. 4　フローショップ・スケジューリング問題

1.5　プロセススケジューリング

コンピュータに対して複数の実行可能な仕事がある場合，どの順序で CPU に割り当てるかという問題である．そして、この時にこの問題を担当するプログラムを ”プロセススケジューラ ”と呼び，これがプロセス選択に用いるアルゴリズムを ”スケジューリング・アルゴリズム ”と呼ぶ.

1.6　ナーススケジューリング問題

ナーススケジューリング問題とは，昼夜３交代制で２０名程度の看護婦達の一ヶ月勤務表をいくつかの制約条件のもとで，各個人の不満が最小限になるように作成する問題である．(Fig. 5 参照)

Fig. 5　ナーススケジュール問題

1.7　組合せ最適化問題に用いられるアルゴリズム

組合せ問題において，それぞれの場合に対する計算が，問題の規模（例えば問題の要素数）に関する多項式時間で実行できるアルゴリズム[2]があるならば，その問題はクラス NP[3]に属するという．ただし，組合せの総数が爆発的に多くなる問題については，仮に多項式時間で実行できるアルゴリズムが存在したとしても，すべてに対する解を多項式時間で調べ尽くすことは不可能である．このように，クラス NP の問題であったとしても，現実的には解くことが不可能であるものもある．一方，クラス NP に含まれるどの問題と比べても難しさ[4]が同等か，もしくはそれ以上であり，しかも，多項式時間アルゴリズムがまだ見つかっていない問題のクラスを NP 困難とよぶ．NP 困難の問題に関しては，一般に計算量が問題の規模に対して指数関数的に増加する．TSP，スケジューリングの問題は，NP 困難でかつ組合せが爆発的に増える非常に難しい問題である．

[2]アルゴリズムの計算量が，問題の規模，例えば要素 N 個を考えたとき，$N^5 + N^2$ のような,N のべき乗（logN を含む）の式で表されるならば，そのアルゴリズムは多項式時間アルゴリズムという．
[3]Nondeterministic Polynomial time，クラスとは集まり
[4]ここでは問題の難しさの厳密な定義は省略する

これらのような組合せ最適化では，連続最適化問題のように微分の概念に基づく古典的な最適化手法を直接利用することはできない．したがって，組合せ最適化問題の解法は連続最適化問題のアルゴリズムと本質的に異なるものとなり，基本的には，解となり得るすべての場合について調べ上げるという列挙的手法を取らざるを得ない．しかし，NP困難で，しかも組合せの総数は有限ではあっても膨大な数にのぼる問題は多い．

これらの問題に関しては，厳密な最適解を求めることはあきらめ，良い近似最適解が比較的短時間で得られればそれで満足しようという考え方が一般的には受け入れられている．その方法はヒューリスティック解法あるいは発見的解法といわれており，その有力な手法として欲張り法がある．欲張り法は，有限である実行可能解に対する目的関数から値を計算して，求まった値がその段階で最適であればその値を解として取り入れるという方法を繰り返し行い，最適解を求めていくという方法である．この方法では，解となり得る点についてすべて調べるわけではないので，問題の（大域的）最適解が得られる保証はないが，組み合わせが多い問題に対しては実際に解を効率的に得ることができる．

ヒューリスティック解法は，一度局所解に陥るとそこから抜け出すことができなくなる．そこで，改悪となるような解への移動も許すことによって，好ましくない局所解に捕捉されることを避ける考え方がある．その手法にメタヒューリスティック解法がある．

メタヒューリスティック解法の基本型は局所探索法であるが，得られた近似最適解に対して，さらに近傍で部分的な修正を繰り返し加えることにより，より良い解近似最適解を得る方法である．ただし，最終的に得られる解の善し悪しは近傍の定め方による．一般に，近傍を大きく取れば，小さい近傍を用いた場合に比べ，現時点における解よりも，より良い解が見つかる可能性が大きいので，最終的に得られる局所最適解の質は良くなる．

局所探索法を基にした代表的な手法としては，以下のようなものがある．

- タブー探索法
  近傍内での最良の解を求め，これが改悪であっても， 現時点での解を更新することを基本とする． しかしこの操作をそのまま実行すると循環が生じやすいので，タブーリスト T を用意しておき (例えば， 過去何回かの繰り返しで，訪れた解のリスト)，T 内への遷移を禁止することで循環を抑制している

- SA（Simulated Annealing)
  EC ゼミで説明済み

- GA (遺伝的アルゴリズム，Genetic Algorithm)
  EC ゼミで説明済み

- AL（人工生命，Artificial Life）
  基本的な考え方は GA と相違ないが,「染色体の長さに制限が無い」,「全ての生物が自分をコピーすることができる (交配がない)」,「より優れた生物はいつまでも生き残ることができる」という点で異なる．

1.8 TSP に用いられるアルゴリズム

1.8.1 TSP と列挙的手法

TSP に用いられる列挙法には動的計画法などがある．

動的計画法

動的計画法は，大きな問題を小さな問題に分けて，それをひとつひとつ解いていくことで大きな問題を解く方法である．ここでは，動的計画法について簡単に説明する．

動的計画法は次のような考え方である．

あらかじめ出発点（=終点) を定めておいて，巡回路の最終点に戻る直前に訪問する可能性のあるすべての点 x について，出発点から点 x までの最短な部分巡回路を考える．そして，その部分巡回路を求めるには，同様の考え方を用いて，点 x にたどり着く直前の点として考えられるすべての点 y までの最短な部分巡回路を調べればよい．この考え方を用いて，出発点までさかのぼっていくと，最終的には，例えば，出発点が都市 1 ならば，都市 1 から個々の都市（ここでは都市 2,3,4) への距離のみを用いた形にまで分解することができる．

TSP は組合せ爆発を起こしてしまう問題なので，都市数に対して指数関数的に計算量が増加するということは前節で述べた．動的計画法は，ある程度の規模の問題なら適用可能であるが，同じ，列挙法で解くならば，より効率の良

い分枝限定法[5]を用いたほうが良い．しかし，分枝限定法でも解けないような大規模な問題になると，列挙法ではなく，次に説明するヒューリスティック解法を用いると良い．

1.8.2 TSP とヒューリスティック解法

ヒューリスティック解法の 1 つである欲張り法に基づく手法に最近近傍法という方法がある．この方法では，ある都市から出発して，まだ訪問していない隣接した都市の中で，現在の都市に最も近い接点を次々と移動していく．Fig. 1 の例の最適解は 21 である．この例に対して，まず，都市 1 を出発点として最近近傍法を適用すると，巡回路 $1 \to 3 \to 2 \to 4 \to 1$，巡回路長 21 が得られる．これはこの問題の最適解になっている．次に都市 3 を出発点とする場合を考える．都市 3 から都市 1,2 は等距離にあるので，どちらを選ぶこともできる．例えば，そこで都市 2 を選ぶと $3 \to 2 \to 4 \to 1 \to 3$ が得られるが，これは先ほどの巡回路と等価であるから最適解である．しかし，都市 2 のかわりに都市 1 を選ぶと，得られる巡回路は $3 \to 1 \to 2 \to 4 \to 3$ となる．この巡回路は長さ 24 であるから最適解ではない．最近近傍法は禁止的に巡回路を構成していくので，得られる巡回路は一般に最適化解と比べてかなり悪くなることが多いが，次に説明するメタヒューリスティック解法で，さらに部分的に修正することができる．

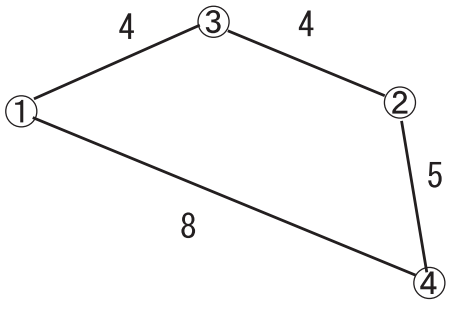

Fig. 6　巡回路 $1 \to 3 \to 2 \to 4 \to 1$

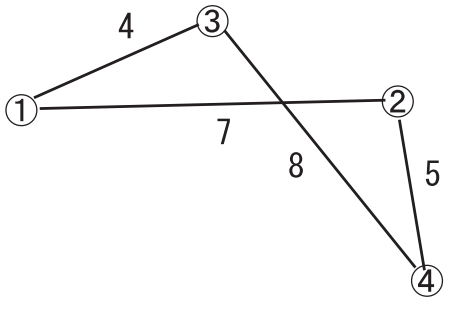

Fig. 7　巡回路 $3 \to 1 \to 2 \to 4 \to 3$

1.8.3 TSP とメタヒューリスティック解法

メタヒューリスティック法は，前に述べたように得られた局所解を部分的に修正を加えるものである．メタヒューリスティック解法の種類については，前節で述べた．ここれは TSP についてのメタヒューリスティック解法における近傍の取り方について述べる．巡回路が Fig. 8 のように得られているとする．そのとき隣り合わない任意の 2 本の枝

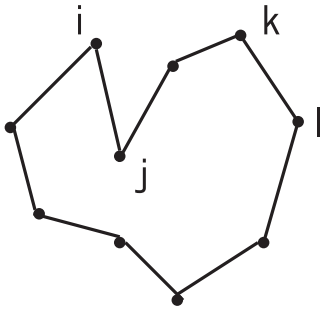

Fig. 8　巡回路

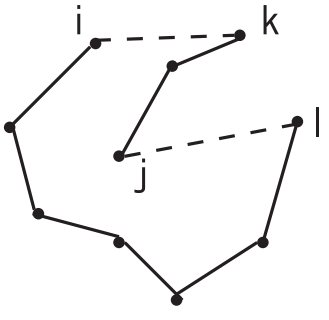

Fig. 9　新たに生じた巡回路

を巡回路 Fig. 9 から取り去り，別の 2 本の枝を付け加えて得られる巡回路を考える．ここで述べている 2 本というのは，TSP における近傍のことであり，2 本の場合は 2-opt 近傍[6]という．2-opt の場合，都市数が N 個の場合新たに考えられる巡回路の数は N(N-3)/2 個である．この操作を何度か繰り返していると，徐々に解が良くなる．先ほどの例の解である Fig. 7 の巡回路 $3 \to 2 \to 4 \to 1 \to 3$ についてこの方法を用いる．都市数が 4 個であるので新たにできる巡

[5]第 1 回で紹介した

[6]連続最適化問題の場合は，ある点を中心とした十分小さな半径 d の円の内部を近傍と呼ぶが，離散的な問題については，ある点 x に隣接した点のうちいくつの点を x の近傍とするという定義になっている

回路は 4(4-3)/2=2 個で Fig. 10，Fig. 11 である．このうち，Fig. 10 は，最適解である．ただし，2-opt 近傍の場合，

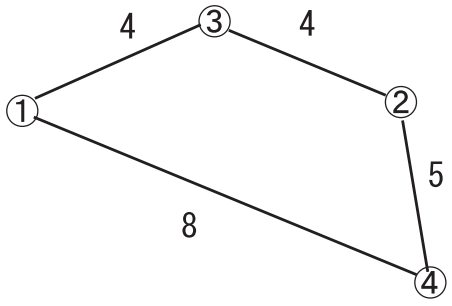

Fig. 10　巡回路 $1 \to 3 \to 2 \to 4 \to 1$

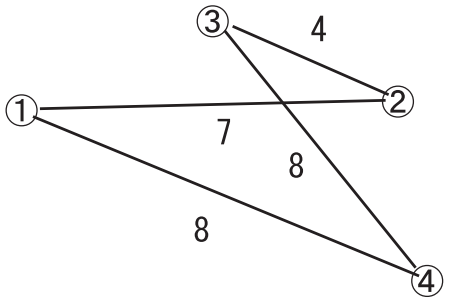

Fig. 11　巡回路 $3 \to 4 \to 1 \to 2 \to 3$

近傍がとても小さいので，都市数が大きくなると，良い解が得られないことが多々ある．問題の規模に合わせて 3-opt 近傍，4-opt 近傍など，近傍を広げるとより，厳密な解に近づくが，近傍を大きくしすぎると，新たに生じる巡回路の個数が爆発的に増加する．そのため 4-opt 近傍以上の近傍が用いられるのは非常に稀である．

1.9　プロセススケジューリングのアルゴリズム

プロセススケジューリング問題をとくにあたって，重要なことは以下の四点であると考えられている．

- スループットの向上
- 応答時間の短縮
- 公平な割り当て
- 応答時間が予測可能であること

スケジューリングアルゴリズムには，いったいどのようなものがあるのだろうか．

CPU を割り当てられたプロセスが，実行を終了するか入出力を行うまで CUP を手放さないとしたら， 他のプロセスの応答時間がそのプロセスの特性に大きく依存してしまうことになる．よって，現在の多くのオペレーティングシステムでは，オペレーティングシステムがプロセスの実行を途中で中断し，別のプロセスの実行を開始させるということを行う．これを**横取り** *(preemption)* という。 スケジューリングは横取りを許すか許さないかで性質が大きく異なる．横取りを許すスケジューリングを横取りのある (preemptive) スケジューリング，横取りを許さないスケジューリングを横取りのない (nonpreemptive) スケジューリングという．プロセススケジューリングにおける代表的なスケジューリングのアルゴリズムを示す．

到着順スケジューリング

システムに到着したプロセスから順に処理する，横取りのないアルゴリズムである．プロセスは実行可能になると実行可能待ち行列の最後におかれる．スケジューラは CPU を，実行可能待ち行列の先頭のプロセスから順に割り当てる．

処理時間順スケジューリング

処理時間の短いプロセスから順に CPU を割り当てる，横取りのないアルゴリズムである．処理時間順は前プロセスの平均応答時間を最小にする．横取りを適用した処理時間順アルゴリズムの 1 つに，残余処理時間順（SRPT:Shortest remaining processing time first）がある．

優先度順スケジューリング

各プロセスに実行の優先度を与え，優先度の高い順に実行するというものである．優先度は様々な基準によって定義される．優先度順では，優先度の高いプロセスを優先的に扱うという性質から，優先度の低いプロセスは実行可能であってもなかなか実行されないと言う，いわゆる**飢餓状態（***starvation***）**に陥ることがある．

ラウンドロビンスケジューリング

横取りのあるスケジューリングの方針として最も広く用いられているのは，一定の時間ごとにプログラムの実行を切り替えるというものである．プログラムの実行が開始され定められた時間が経過すると，次のプログラムが開始される．いったん実行を終えたプログラムは待ち行列の最後に回される．すなわち，プログラムは待ち行列を巡回しながら順番に実行されるわけである．このようなスケジューリングアルゴリズムをラウンドロビンという．

1.10　ナーススケジューリング問題のアルゴリズム

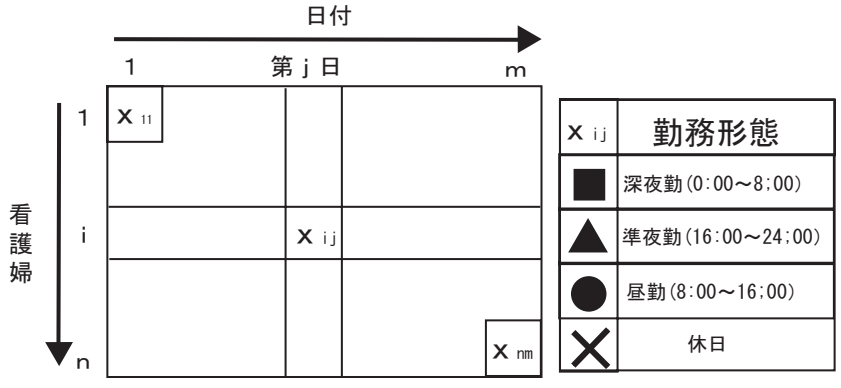

Fig. 12　NSP のスケジュール表現と勤務形態

ナーススケジューリング問題は主に GA によって研究が行われている[7]ため，以下に GA による解法の概要を説明する．ナーススケジューリング問題は，看護婦ｉのｊ日の勤務 x_{ij} を要素とするｎ×ｍ行列を決定する問題であるとも言える．ここに，ｎは看護婦数，ｍは当該月の日数，x_{ij} は基本的には深夜勤，準夜勤，日勤，年休，公休等が入る．(Fig. 12 参照)

そして例えば, 看護婦の数は 15 名，スケジュール日数は 30 日，深夜勤者数は 1 日につき 2 人，日勤→深夜勤の組合せは禁止，希望の休みを指定するというような各種制約条件を設定し，各看護婦と全体の評価項目を与える．そして初期集団を生成し各個人のスケジュールを親とし GA 操作を行い最適な評価値を計算する．

参考文献

1) 福島雅夫，数理計画入門，朝倉書店，1996

2) http://home.intercity.or.jp/users/sasaki/reserch/pre2.html

3) http://www.mmm.muroran-it.ac.jp/ advcomp/ch5/index.html

4) http://www.kecl.ntt.co.jp/as/reports/JPN/yamada.html

5) http://www.pa.airnet.ne.jp/ tomsato/SCM/SchDFree0.html

6) http://www.al.cs.kobe-u.ac.jp/research/opt/

7) http://endeavor.eng.toyo.ac.jp/ takeshi/www/node2.html

8) http://kecsglue.kj.yamagata-u.ac.jp

9) 遺伝的アルゴリズム　北野宏明，産業図書，1995

[7]GA 以外では数理計画法，発見的探索，ニューラルネットなど