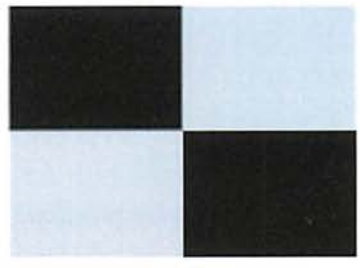
2015 年度修士論文報告書

# 割当型設備配置問題のための対称性と二次元交叉を利用した進化的解法に関する研究

指導教員　　山本 久志



首都大学東京
システムデザイン研究科
経営システムデザイン学域
学修番号：14892511
胡 巍

# 目次


**第1章　序論 …… p.1**

1.1. はじめに……p.1

1.2. 研究の目的……p.1

1.3. 本論文の構成……p.2

**第2章　先行研究 …… p.3**

2.1. 設備配置問題の種類……p.3

2.1.1. 設備配置問題の分類 ……p.3

2.1.2. 設備配置問題の分類 ……p.4

2.2. 設備配置問題の解法……p.5

2.2.1. 厳密解を求めるアルゴリズム ……p.6

2.2.2. 近似解を求めるアルゴリズム ……p.7

2.3. *m* 行 *n* 列格子状設備配置問題……p.8

2.3.1. 問題定義……p.8

2.3.2. 定式化……p.9

2.3.3. 評価関数……p.10

**第3章　対称性の特徴を取り入れた遺伝的アルゴリズム ……**

**…… p.11**

3.1. 先行研究におけるアルゴリズムとの関連……p.11

3.1.1. 対称性の説明……p.11

3.1.2. 遺伝的アルゴリズム …… p.12
3.2. 対称性の特徴を取り入れた遺伝的アルゴリズム …… p.14
3.2.1. アルゴリズムの説明 …… p.14
3.2.2. 対称性操作の評価 …… p.18
3.2.3. アルゴリズムの評価 …… p.23
**第4章　部分一致交叉と隣接選好を取り入れた二次元交叉のハイブリットなアルゴリズム…… p.27**
4.1. 先行研究におけるアルゴリズムとの関連…… p.27
4.1.1. 2 点交換を用いた近傍探索法…… p.27
4.1.2. 隣接選好を考慮した近傍作成法 …… p.28
4.1.3. 近傍探索手順を考慮した SA アルゴリズム …… p.31
4.2. 部分一致交叉と隣接選好を取り入れた二次元交叉のハイブリットなアルゴリズム…… p.32
4.2.1. アルゴリズムの説明 …… p.32
4.2.2. アルゴリズムの評価 …… p.38
**第5章　結論 …… p.45**
**参考文献 …… p.47**
**謝辞 …… p.50**
**付録 …… p.51**

〔36 × 30 = 1080 字〕

# 第1章　　序論

本章では，本論文で対象とした設備配置問題の概略やその背景について説明し，続いて本論文の目的と構成について述べる．

## 1.1.　研究背景

設備配置問題とは複数の設備の空間的配置を決定する問題の総称であり，対象とする問題によって様々な問題の特徴や制約条件が存在する．例えば工場や事業所等の場所に多数の目的を持つ生産現場に設備を配置する際には，生産の効率をもっと向上できる設備の配置方法を求めることは必要である．また効率的な配置は原料運搬等のコストを 30%削減することができる[16]．さらにこれらの配置決定の過程において，プロセス上のフローや経済性，外観，環境，法規，将来の拡張性といった様々な制約条件や考慮事項が存在し，これらの複雑な要因を含む問題全てを 1 つの設備配置問題としてモデル化することは非常に困難である．そのため，工学的にこれらの問題を対象とする場合はこれらの問題を分割し，単純化することによって，より小規模で局所的な設備配置問題としてモデル化し，研究対象とすることが多い．これらの分割，単純化された設備配置問題は，プラントレイアウト，職場レイアウト，ファシリティレイアウト等とも呼ばれ，その解法の適用範囲の広さや重要性の高さから，オペレーションズリサーチ，システム工学，コンピュータ科学等の理工系の分野や，数理経済学や社会学，心理学まで，様々な分野で研究が進んでいる[1]．

## 1.2.　研究の目的

本論文では「工場に設備をどのように配置すれば好ましいか」を考慮した中大規模の設備配置問題を中心に扱っている．設備配置問題は，規模が大きく（設備数が多く）なると組み合わせ数も増大し，厳密解法による最適解の算出が困難とされている．そこで本論文では，それを一般化した形で数学的な問題としてモデル化し，中大規模の問題だけではなく，小規模の問題に対しても適用可能である，近似解による配置算出アルゴ

リズムを提案することを目的としている.

また本論文で提案したアルゴリズムの有効性の評価のため, 実際にプログラムを作成して数値実験を行い, その結果から考察を行う.

## 1.3. 本論文の構成

本論文の構成を以下に示す.

第 2 章において, 実際に設備配置問題の分野で考えられている問題の種類や性質, また本論文で対象とした設備配置問題の問題定義を行い, その特徴についての説明を行う.

第 3 章では, 本論文で中大規模の問題を解決するために, 先行研究の解法を基に提案した近似解を求める対称性の特徴を取り入れた遺伝的アルゴリズム(Symmetry in Genetic Algorithm ; SGA)についての説明と数字実験の結果を述べる.

第 4 章では, 本論文で提案したもう 1 つの近似解を求めるアルゴリズムである部分一致交叉と隣接選好を取り入れた二次元交叉のハイブリットのアルゴリズム(Hybrid method based on Genetic Algorithm ; HGA)の説明を行う. また提案したアルゴリズムを用いて実際に行った数値実験の結果について述べ, その評価を行う.

最後に, 第 5 章で本論文のまとめを行う.

# 第2章　　設備配置問題に関する研究背景

本章では，設備配置問題の分野で実際に考えられている問題の種類や解法，また本論文で扱った設備配置問題の説明を行う．まず 2.1 節で設備配置問題の種類について説明し，続いて 2.2 節ではその代表的な解法について説明する．最後 2.3 節では対象とする設備配置問題についての説明を行う．

## 2.1.　設備配置問題の種類

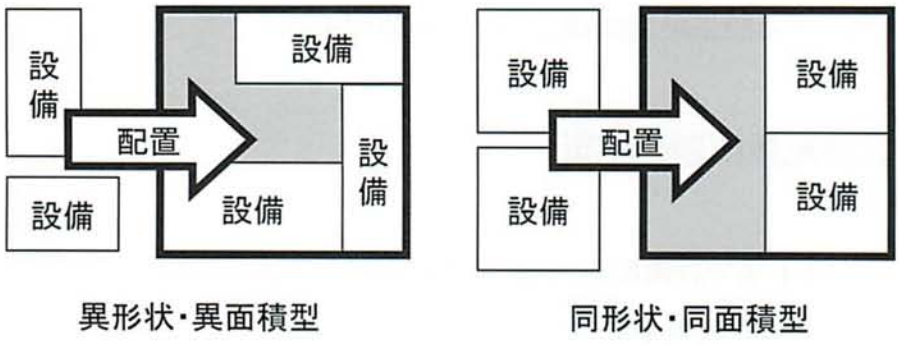
本節では 2.1.1 項で設備配置問題の分類について説明し，2.1.2 項で設備配置問題の評価基準について説明する．

### 2.1.1　設備配置問題の分類

設備配置問題とは有限個の設備を想定した同じ数のセルに分けられた工場に配置する際の組み合わせを求めるという数理計画問題として表現される．またこれらの問題は，配置する設備の形状，面積によって図 2.1 に示すような「異形状・異面積型」と「同形状・同面積型」の 2 種類に分類することができる[11]．前者の「異形状・異面積型」の設備配置問題は，各設備の配置位置と形状を 2 次元座標で表現する座標方の設備配置問題として取り扱うことが可能である[25]が，設備を配置する際の組み合わせに加えて，例えば長方形状の設備を縦，横のどちらに配置するかといった，設備の回転等の組み合わせを考慮する必要があり，設備数の増大と共に計算量が爆発的に増大し，最適な組み合わせを得ることが非常に困難であるという性質を持つ．

本論文では，より単純な問題を対象とし，後者の「同形状・同面積型」の設備配置問題を研究の対象とした．この「同形状・同面積型」の問題は，工場への設備配置の際に，製品を加工する際の役割や，安全性の考慮等から同系統の設備をグループ化することによって，同形状・同面積の単一の正方形の設備に近似して表現し，これを工場敷地全体に仮想的に長方形状また正方形に区画分けした同じ形状と面積のセルに配置することが仮定できる．この問題についても「異形状・異面積型」の問題と同様に，設備数

の増加とともにその組み合わせ数が爆発的に増加し，最適な組み合わせを得ることが非常に困難であるという性質を持つ．逆に設備が同形状・同面積であることにより配置案の表現が単純化され，異形状と異面積の設備で発生される複数の配置案を同一の配置案として表現して探索する組み合わせを減らすことが可能であるという特徴を持つ．




異形状・異面積型　　同形状・同面積型

**図2.1　設備配置問題の分類例**

### 2.1.2　設備配置問題の評価基準

工場での設備配置を目的とした設備配置問題は通常，運搬性を考慮するため物流の際に発生するコストを評価基準として用いることが多く，配置された設備間の距離に応じて増加，減少する物流コストが最小になるような配置案の探索を目的とした設備配置問題を研究対象とすることが多い．

設備が隣接の場合
→ 監視員1人

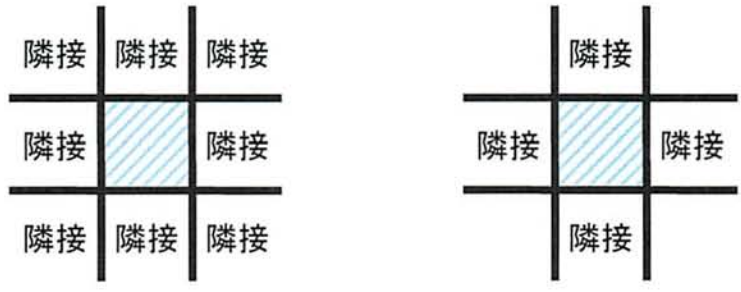
設備が非隣接の場合
→ 監視員2人

**図2.2　監視員が必要な設備の隣接・非隣接の例**

しかし，例えば図 2.2 に示すような 2 つの設備を互いに隣り合う位置に配置すれば，

作業員 1 人で済むが，逆に 2 つの設備が離れれば作業員が 2 人必要となり，それによってコストも増える．そのため，本研究は設備が非隣接に配置されることによって配置案が非効率だと考えられる．隣接条件が不満足の場合，あるペナルティのコストを与えて物流コストと共に評価基準として配置案を評価する[18]．Foulds[6]や Houshayar ら[13]によって従来の研究では，隣接は設備間距離を考慮し，連続値で評価しているため，隣接・非隣接を考慮した離散的な問題である隣接性を評価しているとは言い難い．よって本論文では，連続的な物流コストと離散的な隣接性の 2 種類の基準を同時に考慮している場合を前提とした設備配置問題を対象とする．

隣接の定義にも図 2.3 のように，Rosenblatt[15]の研究に代表される隣接性（1 つの設備と周りの 9 つのセルに配置された設備）と，Dutta と Sahu[5]の研究に代表される隣接性（1 つの設備と周りの 4 つのセルに配置された設備）の 2 種類が存在するが，本論文での隣接性は，後者の Dutta と Sahu の研究と同様の配置を隣接な配置と定義した．




Rossenblattの隣接の定義　　Dutta and Sahuの隣接の定義

**図2.3　隣接の定義の違い**

## 2.2.　設備配置問題の解法

本論文で対象とする「同形状・同面積型」の設備配置問題は二次割当問題（QAP）として定式化され[3]，Bazaraa[2]や Burkard[4]らによって解法も古くから研究されている．QAP は Golany[9]の整数計画法の分野での研究や，Lackxonen[14]の組合せ最適化の分野での研究等，様々な研究が続けられているが，問題の制約条件やその特徴に依存して計算効率，結果が左右されて，あらゆる問題に対して万能な解法を求めるのは難しい現状にある．

### 2.2.1 厳密解を求めるアルゴリズム

本論文で対象とする問題では，考えられるすべての組み合わせを調べてその中から最良の解を得るアルゴリズムを適用することによって，厳密な最適解（厳密解）を求めることが可能となる．本項では，厳密解を求めるアルゴリズムについての説明を行う．

厳密解を求めるアルゴリズムの中で最も単純なアルゴリズムとして，全数列挙法があげられる．この全数列挙法について簡単に説明すると，対象とする組み合わせ最適化問題の全ての解を列挙し，その中の最適なものを選べば，それが厳密解であるというものである．原理的にはこの解法を用いることで全ての組み合わせ最適化問題が解けることになる．しかし，例として本論文での設備配置問題を例にとってみると，$m$ 行 $n$ 列格子状問題の場合$(mn)!$通りの解を求めることになる．このとき，問題のサイズが大きくなるにつれて組み合わせ数が急激に増加し，高速な計算機を用いても現実的には解くのが難しいという問題点がある．

そこで設備数が少なく問題規模の小さな場合に全数列挙法を現実的な手法とするために，個々の問題特有の構造を利用して，不要な列挙を考察から除外していくという考えの下に生み出された計算原理が分枝限定法（branch and bound method）である．

分枝限定法[24]とは，問題をいくつかの小規模な部分問題に分割して，その部分問題全てを解くことで等価的にもとの問題を解くという考え方に基づいた手法で，問題を小規模な部分問題に分割する操作を分解操作と呼ぶ．いくつか存在する部分問題の中から，次の操作において分解操作を行う部分問題を選択する方法を探索法と呼ぶ．無駄な部分問題の生成を抑制するために，ある状態から最適解が求まる場合，ある状態から最適解が得られないことが何らかの理由によって結論できる場合には，直ちに部分問題を分解する操作を終了する．これを限定操作と呼ぶ．すべての部分問題において分解操作と限定操作を行った時点で探索は終了し，それまでの最良の解を厳密解として得るという手法である．

これらを踏まえて先行研究の山本ら[26]では，分枝限定法を基にした問題の特徴を考慮した最適配置算出アルゴリズムの提案が行われている．2 次元の設備配置問題において設備数少ない場合（先行研究では設備個数が 14 以下） に有効が確認されている．ただ 15 以上の場合に分枝限定法を利用すれば，計算時間が 11 時間以上となっている．本研究は主に大規模の問題を対象として解決するため，莫大の組み合わせに

対して，分枝限定法を利用すれば困難な面がある．

### 2.2.2　近似解を求めるアルゴリズム

本論文で対象とする設備配置問題では，設備数の増加と共にその組み合わせ数も爆発的に増加し，2.2.1 項で説明した分枝限定法を適用した場合でも，その計算効率化の作用が組み合わせ数の増加に対応できずに現実的な時間内での厳密解の探索が困難になる．このような設備数の多い大規模な設備配置問題を対象に現実的な時間内で解探索を行う場合，近年，シミュレーテッドアニーリング（Simulated Annealing ;SA）[7]，遺伝的アルゴリズム（Genetic Algorithm ; GA）[12]やタブサーチ（Tabu Search ; TS）[8]等の発見的なアルゴリズムを用いて解探索を行うことが注目されている．これらのアルゴリズムは，広大な解空間から各アルゴリズム特有の解探索方法によってよりよい解を探索するものであり，これによって得られた最良解は全数列挙法や分枝限定法で得られた厳密な最適解である厳密解とは異なり，各アルゴリズム適用時の探索内で得られた，解の最適性は保障されないが優れた解である近似解を得ることになる．

SA とは，鉄などの物を高温に熱し，徐々に冷却をすることで原子が始めは活発に動き出すが，冷却されることで動きが収まる性質を数学的に表現したアルゴリズムである．始めに高温で，解空間内で探索をおこなって局所解に行き詰ったら遷移をする．遷移してよい解に進んだら解の更新を行う．そして徐々に温度が下がっていき遷移が起こりにくくなるアルゴリズムである．そのため，局所解には陥りにくいアルゴリズムである．

GA とは，生物進化の原理から着想を得た近似解を求めるアルゴリズムである．解を個体として遺伝子表現する事により，複数個体をグループ化した世代内で選択，交叉，突然変異等の解操作を行う．これらの操作を世代の更新と共に繰り返しながら解探索を行うアルゴリズムである．また GA を適用する場合，世代数や世代内の個体数，選択率，交叉率，突然変異率等のパラメータを問題に応じて事前に設定する必要があるが，広大な解空間を持つ問題に有効であるという特徴を持つ．

TS とはタブーリストを保持する事により近傍探索に特化した近似解を求めるアルゴリズムである．近傍探索を行う場合，ある解の近傍を探索することによってその解周辺の最良解を効率的に探索することができる反面，その問題の解空間の構造によって局所解に陥ってしまうことも考えられる．この時 TS では，タブーリストにそれまで探索した解

を記憶しておくことにより効率よく近傍探索を行い，局所解に陥ることを防ぐことが可能である．TS を解法として問題に適用する際にも，タブーリストの記憶領域の量であるタブーリスト長を問題に応じて事前に設定する必要がある．

## 2.3. *m* 行 *n* 列格子状設備配置問題

本節では，対象とする設備配置問題についての説明を行う．まず 2.3.1 節で，本論文で対象とする *m* 行 *n* 列格子状設備配置問題についての説明を行い，次に 2.3.2 項で，目的関数の定式化を行う．2.3.3 項でその評価関数について述べ．

### 2.3.1 問題の定義

本論文では 2.1 節で述べたように，「同形状・同面積型」の設備配置問題を研究の対象とした．このため複数の設備を同面積，同形状に仮定し，これを同数の同じ格子状の配置場所であるセルに配置する問題を考察する．つまり図 2.4 のように，*m* 行 *n* 列の格子状のセルに，セルの数と同数の *mn* 個の設備を配置するという割当問題として表現する．1 つのセルには複数の設備が配置されることはなく，1 個の設備のみが配置される．ここで，$i = 1,2,3,...,m$ と $j = 1,2,3,...,n$ に対して行 $i$ と列 $j$ のセルをセル $(i, j)$ と呼び，配置される設備をそれぞれ，設備 1，設備 2，…，設備 *mn* とする．

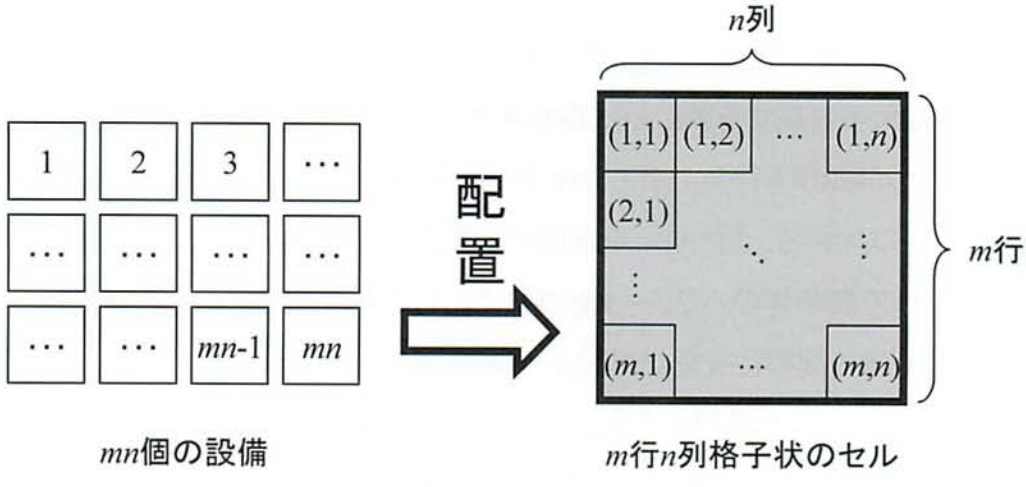



**図2.4　*m* 行 *n* 列格子状設備配置問題の概念図**

### 2.3.2 定式化

本節では，目的関数の定式化を行う．まず$i_1 = 1,2,...,m$，$j_1 = 1,2,...,n$，$i_2 = 1,2,...,m$と$j_2 = 1,2,...,n$に対して，次の記号を定義する．

| | |
|---|---|
| $p(i_1, j_1)$ | ：セル$(i_1, j_1)$に配置された設備の番号. |
| $P$ | ：$mn$次元ベクトル，配置$(p(1,1), p(1,2),...p(m,n))$を表す．また以後，配置$P$と呼ぶ. |
| $F(p(i_1, j_1), p(i_2, j_2))$ | ：設備$p(i_1, j_1)$から設備$p(i_2, j_2)$への単位距離あたりの単位物流コスト，ただし$F(p(i_1, j_1), p(i_2, j_2)) \geq 0$. |
| $D((i_1, j_1),(i_2, j_2))$ | ：$D((i_1, j_1),(i_2, j_2)) = \lvert i_1 - i_2 \rvert + \lvert j_1 - j_2 \rvert$で表される，セル$(i_1, j_1)$，セル$(i_2, j_2)$間の直交距離. |
| $G(p(i_1, j_1), p(i_2, j_2))$ | ：設備$p(i_1, j_1)$に対して設備$p(i_2, j_2)$が隣接していない場合に，その不満足の度合いに応じて与えられる正のペナルティ，ただし$G(p(i_1, j_1), p(i_2, j_2)) \geq 0$. |
| $U((i_1, j_1),(i_2, j_2))$ | ：セル$(i_1, j_1)$，セル$(i_2, j_2)$間が隣り合っている時には0，そうでない時には1をとる0-1変数. |

また上記より，本論文で対象とする設備配置問題の目的関数$I(P)$は，次の(2.1)式で表される.

$$I(P) = \sum_{i_1=1}^{m}\sum_{i_2=1}^{m}\sum_{j_1=1}^{n}\sum_{j_2=1}^{n}\Big(F(p(i_1, j_1), p(i_2, j_2))D((i_1, j_1),(i_2, j_2)) + G(p(i_1, j_1), p(i_2, j_2))U((i_1, j_1),(i_2, j_2))\Big) \tag{2.1}$$

この時，本論文で考察する設備配置問題は，

$$I(P^*) = \min_{P}(I(P)) \tag{2.2}$$

となる設備の配置$P^*$（最適配置）を求める問題となる.

### 2.3.3　評価関数

本論文では対象とした設備配置問題の配置案に対する評価として，2.1 節で述べた様に，設備間の距離に依存する連続的な評価基準である物流コストと，設備間の配置の位置関係に依存する離散的な評価基準である隣接選好の 2 種類を考える．

次に上記で挙げた 2 種類の評価基準について説明する．まず連続的な評価基準である物流コストについては，対象設備間に単位距離あたりの単位物流コストを与え，それに設備間の距離を掛けたものとして表す．このとき，設備間の距離については，セルの中心から中心までの直交距離を取り，図 2.5 のように，10×10 の大きさのセルの集合の場合，セル(1,1)とセル(1,3)との距離は 20 となり，セル(1,1)とセル(2,2)の距離も同様に 20 となる．また離散的な評価基準である隣接選好については，対象設備間に非隣接の場合の隣接選好条件が不満足の度合いに応じた正のペナルティを設定し，対象設備間が非隣接ならそのペナルティを加算し，隣接していればペナルティを加算しないこととする．またこの時の隣接の定義は，2.1.2 項で述べたように，DuttaとSahuの研究と同様の配置を隣接な配置と定義した．

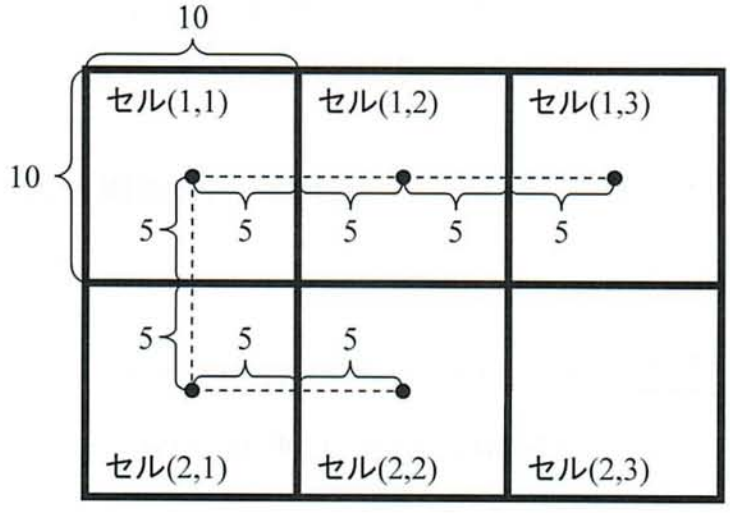



**図2.5　距離の概念図**

# 第3章　　対称性の特徴を取り入れた遺伝的アルゴリズム

本章では，大規模な問題に対して準最適解を求めるための「対称性の特徴を取り入れた遺伝的アルゴリズム」(Symmetry in Genetic Algorithm ; SGA)の説明を行う．まず3.1 節では本論文との関連する先行研究についての説明を行い，続いて 3.2 節では提案するアルゴリズムについての説明と数値実験の結果から得られた評価を行う．

## 3.1.　先行研究におけるアルゴリズムの関連

本論文で 1 つ目の提案アルゴリズムを説明する前に，この節において本論文に関連する先行研究を紹介する．まず 3.1.1 項で設備配置問題では考慮した配置の対称性について説明し，3.1.2 項で部分一致交叉について説明する．

### 3.1.1　対称性の説明

先行研究[22][23]で考慮された対称性の操作とは，対称の配置を排除することによって，計算量を減少することである．その理由は設備の相対的な位置関係が変わらないので，配置を回転また反転させても(図 3.1 に示す)，目的関数も変わらない．そのため，同じ目的関数を持っている配置を排除して探索時間を短縮できる．また，長方形 $(m \neq n)$ と正方形 $(m = n)$ の対称配置がそれぞれ 4 個と 8 個存在する．

もしある配置の目的関数値を算出すれば，長方形の場合は同じ目的関数値を持つ残り 3 個の配置の目的関数値を，正方形の場合は同じ目的関数値を持つ残りの 7 個の配置の目的関数値を算出する必要がなくなり，本来の解空間に対して，長方形の場合は 1/4 に，正方形の場合は 1/8 にそれぞれ解空間が狭くなる．これにより上記で述べたような計算量の増大や探索中の振幅による計算効率の悪化等の問題点を解消し，アルゴリズムの解探索性能の向上が可能であると考えられる．

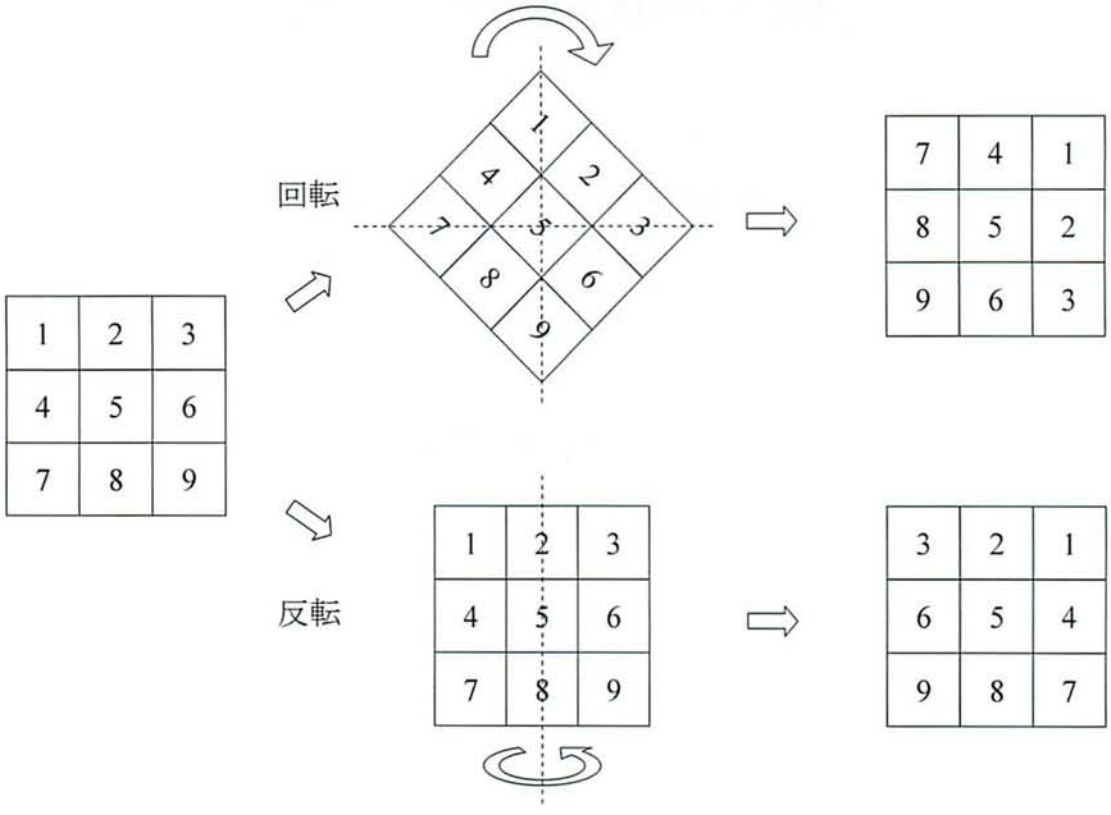



**図3.1　対称配置の例**

### 3.1.2　遺伝的アルゴリズム

第 2 章でも説明したように，GA とは生物の進化における遺伝子染色体の変化をメタロジーとした経験的な解法である[20]．本論文で適用された部分一致交叉（PMX: Partially Mapped Crossover）[10]は一般的な遺伝的アルゴリズム（Genetic Algorithm）の基に提案された 1 つの交叉手法である．この方法は一方の親からある部分の遺伝子をそのまま受け継ぎ，他の親からもできるだけ多くの遺伝子を受け継ぐことを目的にしている．その特徴は設備番号で直接交叉しても，重複した遺伝子の入れ替える作業によって，遺伝子の衝突の状況も起きないような一つの交叉手段である．本論文で適用した部分一致交叉は一般的な遺伝的アルゴリズムの基に提案されたため，まず一般的な遺伝的アルゴリズムの操作手順を示す．

一般的な遺伝的アルゴリズムの計算順概略

Step1　：初期集団を生成して現世代集団とし，各個体の目的関数値と適応度を計算

する. 世代カウンタが 0 とする.

Step2 :(エリート保存)現世代の優れた個体を上位 1 位にする.

Step3 :(選択)現世代集団からルーレット戦略で 2 個体を生成する.

Step4 :(交叉)単純交叉で新たな 2 個体を生成する.

Step5 :(突然変異)突然変異率に応じて, 任意の 2 つの遺伝子を置換し, 次世代集団に加える.

Step6 :次世代集団が集団サイズに達したら現世代とし, 各個体の目的関数値と適応度を計算する. 世代カウンタを 1 増加する.

Step7 :最終世代に達したら終了. そうでなければ Step2 に戻る.

続いて, 一般的な遺伝的アルゴリズムとの異なる部分である交叉の操作（手順 Step4）を設備数が 9 の場合の例で説明する. 図 3.2 に例を示す.

部分一致交叉の交叉操作の計算順概略

<STEP 1> : 2 つの親個体で単純交叉を行い, 2 つの子孫を生成する.

<STEP 2> :生成された子孫で重複する遺伝子があれば, 2 つの子孫間で入れ替える. この作業を重複の遺伝子がなくなるまでやり続ける.

<STEP 3> :2 つの子孫に重複の遺伝子がなくなれば操作を終了する.

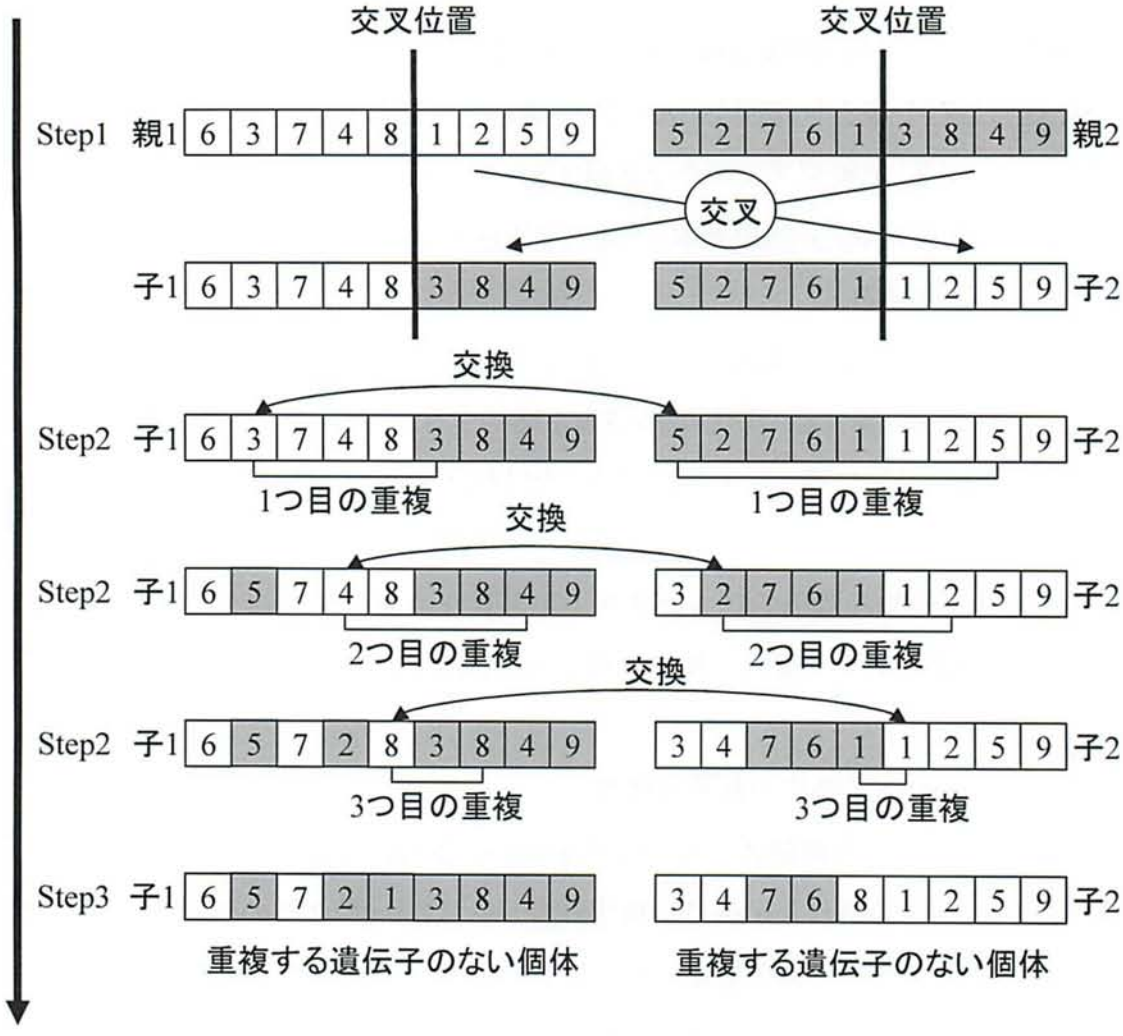



**図3.2　部分一致交叉の例**

## 3.2.　対称性の特徴を取り入れた遺伝的アルゴリズム

本節では，提案アルゴリズムについての説明と，数値実験の結果を行った評価について述べる．まず 3.2.1 項で，アルゴリズムの説明を行い，続いて 3.2.2 項で，数値実験の結果を評価する．

### 3.2.1　アルゴリズムの説明

本節では，GA を基に対称性排除の特徴を取り入れた提案するアルゴリズム SGA[付録 1]の説明を行う．まず，GA と違うオリジナルな部分を示すために従来の GA を説明

する. GA は交叉による代替案の作成で広い領域で探索することによって効果を上げられるという特徴を持つ. また生成された子孫が親個体の遺伝子を保ち, 親たちの優れた部分を引き継ぐことができるが, 劣る遺伝子も子孫の中に引き継がれる可能性もある. そのため, 交叉方法と選択方法を改善し, 集団サイズ, 交叉率, 突然変異率などのパラメータを調整し, 特に子孫が局所解からの脱出のための操作に工夫が必要である. その計算手順は 3.1.2 項に従って, フローチャーを以下の図 3.3 に示す.

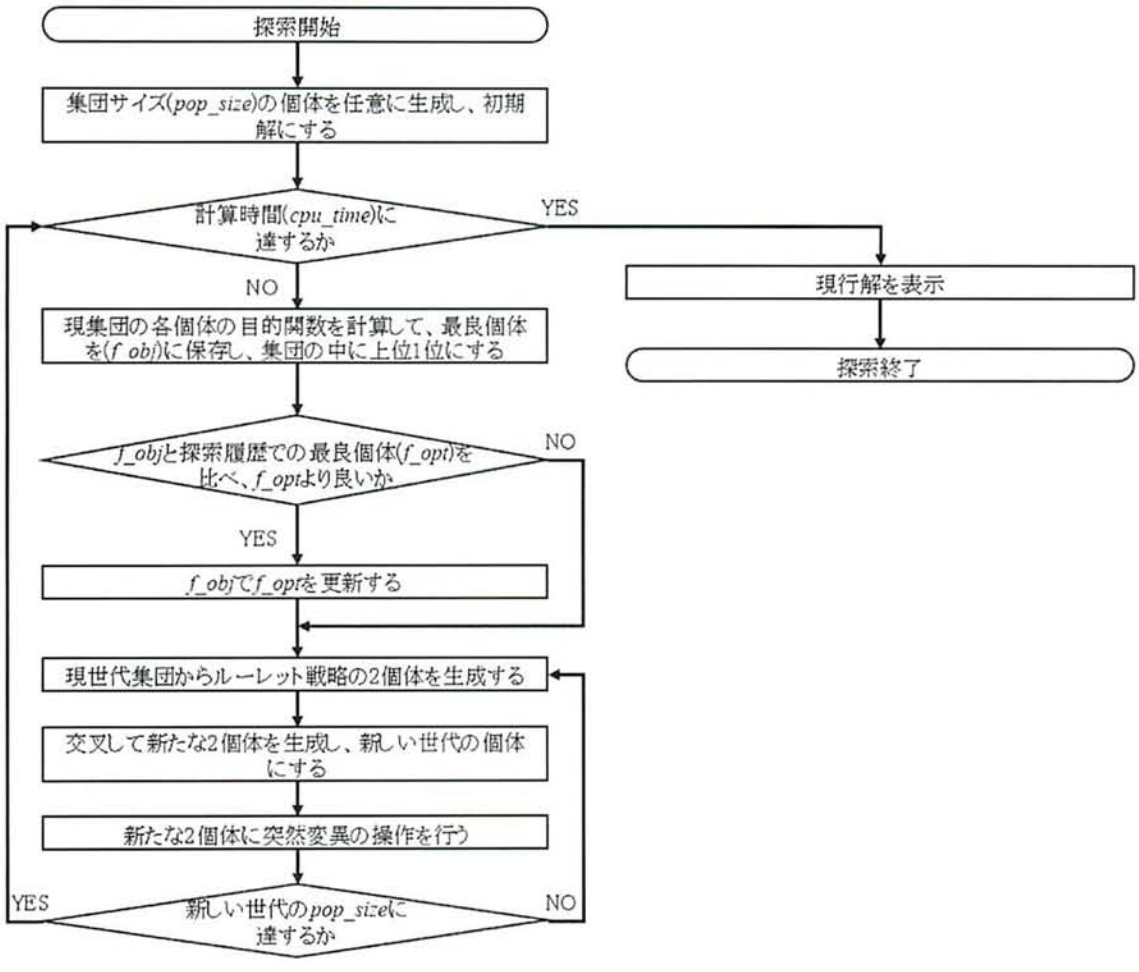



**図3.3　GA の概略フローチャート**

本論文では, 近似解を求める新たなアルゴリズム, SGA に対してよりよい配置案を探すため, 対称性の操作を取り入れた. SGA の操作手順としては, GA と同様に, まず, 初期集団を生成する. 続いて, 現集団の中での最良値と前世代の最良値を比較して, 現集団の最良値がよかったら保存する. もし前世代の最良値が悪い(言い換えると, 前世代の最良結果が更新されていない). 且つ最良値の未更新回数が $N$ 回(事前に設定されたパラメータ)に達すれば, 対称性の操作で現集団を更新する(現集団にある遺伝

子の並び方を変える). SGA では, 対称性の操作を行うために一次元の配置を二次元の列に変える, その方法は以下の2通りがある. ①:遺伝子を行で並べる. ②:遺伝子を列で並べる. 以下設備数が9の配置例で説明する. 図3.4に示すように, 左は配置を左から右まで1行ずつ, 右は配置を上から下まで1列ずつ遺伝子列に変換する. このように作成された遺伝子で交叉の操作を行う. また, 対称性の操作が終わると, 二次元の列を一次元の配置に戻す際, 逆の手順で操作する. 全ての個体に対称性の操作を行うと, 生成した新たな集団で元の集団を更新する. このように繰り返して探索を続けていく. SGA の概略フローチャートを図 3.5 に示す, 色付の部分が GA に対して SGA で改善また追加された操作である.

上記したように, SGA は GA の探索能力を上げるため, 2 つのオリジナルの部分を GA に入れた. 1 つ目は次元変換の操作である. SGA では, 対称性の操作を行う際, 図3.4に示すように2つの方法で遺伝子(一次元の配置)を変えるので, より広い領域で探せると考える. 2 つ目は対称性の操作である. この操作は対称の配置を排除して範囲を縮小することではなく, 逆に対称の配置を利用し, 一次元の遺伝子列に戻る際, 異なる遺伝子が作られることにより, 子孫たちの遺伝子を別の探索領域に移し, 遺伝子で保存された情報を破壊しなくても探索能力を上げることができる.

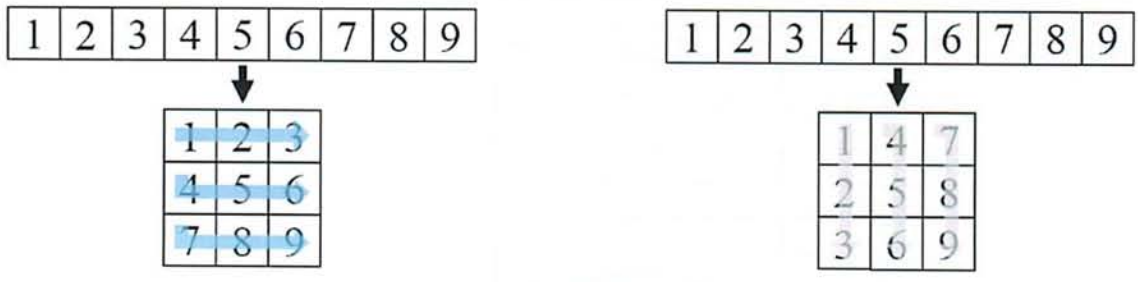



**図3.4　次元変換の例**

本論文で提案した SGA では, 部分一致交叉の基に提案したアルゴリズムである. その操作は以下の手順に従う.

Step1　:初期集団を生成して現世代集団とし, 各個体の目的関数値と適応度を計算する. 世代カウンタを 0 とする.

Step2　:現世代の最良個体を *fobj* に保存し, 集団での上位 1 位にする.

Step3　: *fobj* と探索履歴の最良個体 *fopt* の目的関数を比較し, *fobj* が良い場合, *fobj* で *fopt* を更新する.

Step4 ：*fobj* が *fopt* より悪い場合，*fopt* が更新されい回数が *N* 回（事前に設定されたパラメータ）に達せば，全現集団の個体に対称性の操作を行い，各個体の遺伝子を変換する．

Step5 ：現世代集団からルーレット戦略で 2 個体を選択する．

Step6 ：部分一致交叉で新たな 2 個体を生成する．

Step7 ：次世代集団が集団サイズに達したら現世代とし，各個体の目的関数値と適応度を計算する．世代カウンタを 1 増加する．

Step8 ：最終世代に達したら終了．そうでなければ Step2 に戻る．

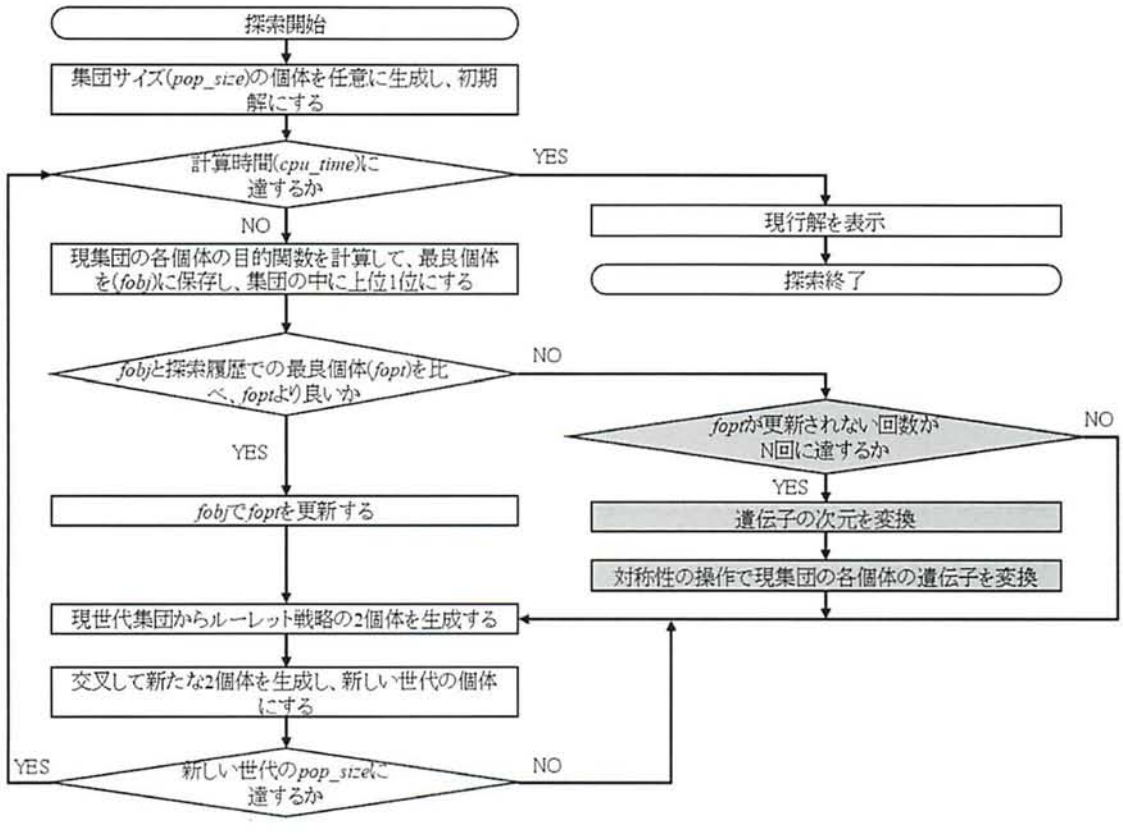



**図**3.5　SGA **の概略フローチャート**

また，ここで，上記の Step4 を説明する．まず，探索履歴での目的関数が最良の個体が *N* 回連続的に更新されていなかったら，対称性の操作に移り，操作を行う．続いて，現集団での個体に対称性で遺伝子の並び方を変換する．最後，作成された遺伝子を新しい集団とし，探索を続ける．その操作は以下の手順に従う．

<STEP 1> :現集団で 1 つの個体を選択する.

<STEP 2> :選択された個体(一次元の配置)をランダムに 1 つの次元変換方法(図 3.4 に示す)で二次元の配置に変換し, その二次元の配置に対して 1 つの対称性操作(反転, また回転させる)を行う.

<STEP 3> :二次元の配置を一次元の遺伝子列に戻して元の遺伝子列を更新する.

<STEP 4> :集団で全ての個体に対称性の操作を行ったら, 終了. そうでなければ <STEP 1>に戻る.

本論文では, 対象とした配置の形状が正方形と長方形 2 つあり, それぞれに対する対称の配置が 7 つと 3 つある. 正方形の各対称配置が左右反転, 上下反転, 90 度回転, 180 度回転, 270 度回転, 対角反転 1 と対角反転 2, 7 つ(表 3.1 に示す)ある. また, 長方形の場合は長さと幅と違うので, 対称の配置が左右反転, 上下反転と 180 度回転 3 つしかない.

### 3.2.2 対称性操作の評価

本論文で紹介, また提案された一連のアルゴリズムは, 全て以下の計算機環境で実験を行った.

**CPU**: Core(TM)i3-3240 (3.40GHz)

**RAM**: 4GB

**OS** : Microsoft Windows 7

**Language**: Microsoft Visual C++ 6.0 日本語版

対称性の操作とは, 上記でしたようにある配置を反転また回転させることである。それらの対称性操作の中から最もよい操作を求めるため, 先行研究で扱った最適解既知問題[21]を対象として対称性を考察する. 今回の実験問題の形状が正方形( $m = n$ ), 問題サイズは 100(10×10), 225(15×15)と 400(20×20)を対象として, どちらの対称性に対して同じ 1 問を作成し, 実験を 3 回行った. 実験の対象とした正方形の 7 つの対称性を

表 3.1 に示す. また必要なパラメータの意味と設定値をまとめると表 3.2 のようになる. 表 3.3 の結果は最良目的関数値と平均値である.

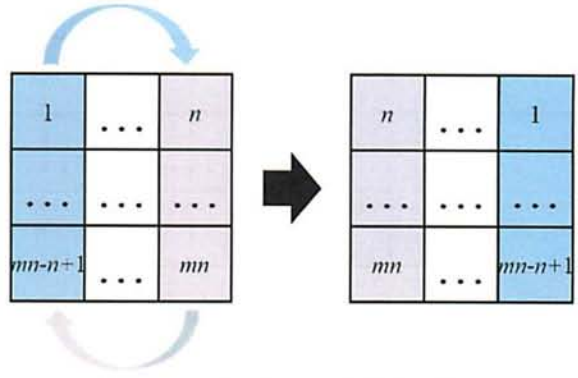



**図**3.6　**左右反転**

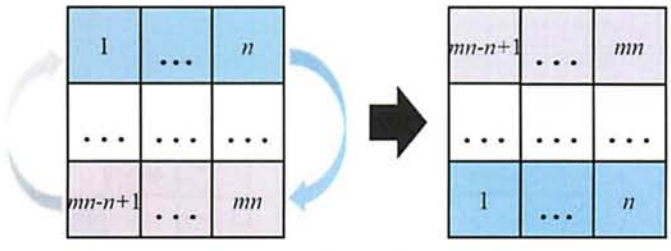



**図**3.7　**上下反転**

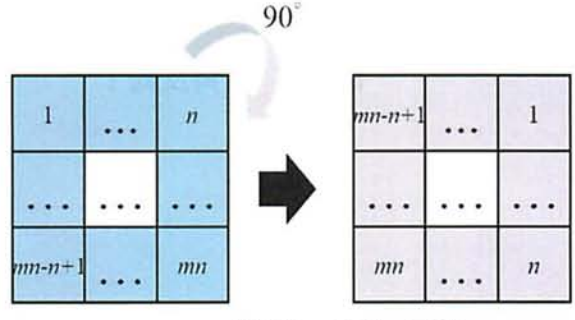



**図**3.8　90°　**回転**

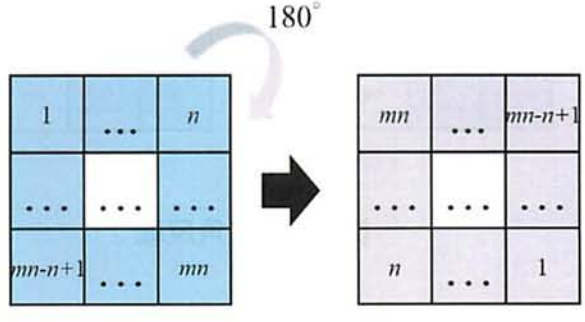



**図**3.9　180°　**回転**

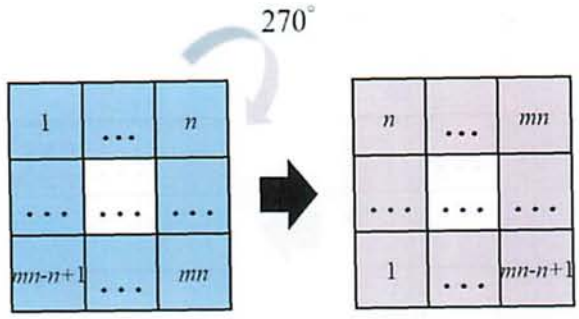



図3.10　270°回転

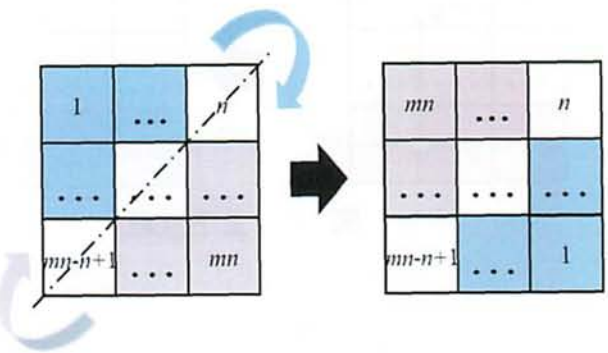



図3.11　対角反転 1

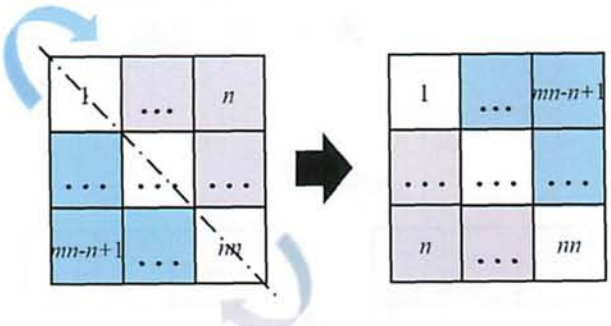
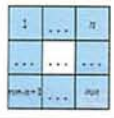
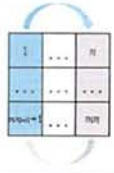
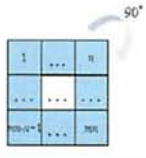
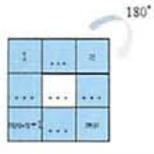
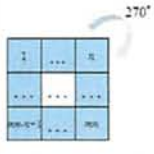
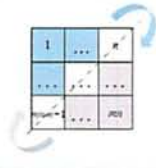
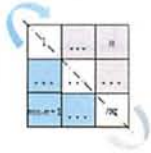
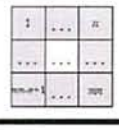



図3.12　対角反転 2

**表3.1　対称性の操作一覧**

| 操作名 | 図 | 操作の内容 |
|---|---|---|
| 対称性 1 |  | 対称性の操作を行わない |
| 対称性 2 |  | 二次元の配置を中央の垂直線で均一に分けられた左右の部分を入れ換える |
| 対称性 3 | | 二次元の配置を中央の水平線で均一に分けられた上下の部分を入れ換える |
| 対称性 4 |  | 二次元の配置を中心に右に 90° を回す |
| 対称性 5 |  | 二次元の配置を中心に右に 180° を回す |
| 対称性 6 |  | 二次元の配置を中心に右に 270° を回す |
| 対称性 7 |  | 二次元の配置を右上から左下までの対角線で均一に分けられた 2 つの部分を入れ換える |
| 対称性 8 |  | 二次元の配置を右上から左下までの対角線で均一に分けられた 2 つの部分を入れ換える |
| 対称性 9 |  | 上記の対称性を毎回ランダムに使う |

ここで，表 3.2 に示した値にパラメータを設定した理由を説明する．集団サイズと交叉率は先行研究[20]と同じ数値を設定した．また，予備実験の結果により $N$ と $P$ に 800 と 50 を設定すれば，実験の結果が最良であるので，今回の実験は 800 と 50 を設定した．

**表3.2　パラメータ設定値(SGA)**

| パラメータ | パラメータの意味 | パラメータの設定値 |
|---|---|---|
| *cpu _time* | 実験の計算時間 | 4000 |
| *pop _size* | 集団サイズ(集団の個体数) | 40 |
| *px* | 交叉率 | 90 |
| $N$ | *fopt* の未更新回数 | 800 |
| $P$ | 対称性の操作を行う確率 | 50 |

**表3.3　実験結果(目的関数平均値)**

| アルゴリズム名 | 問題サイズ=100 | 問題サイズ=225 | 問題サイズ=400 |
|---|---|---|---|
| 対称性 1 | 9860 | 34013 | 97926 |
| 対称性 2 | 10193 | 31780 | 98120 |
| 対称性 3 | 10460 | 33440 | 107493 |
| 対称性 4 | 9146 | 30633 | 97846 |
| 対称性 5 | 9766 | 33280 | 101480 |
| 対称性 6 | 9273 | 31960 | 93606 |
| 対称性 7 | 9440 | 32420 | 98426 |
| 対称性 8 | 9426 | 35046 | 101006 |
| 対称性 9 | 9326 | 33366 | 100173 |

表 3.3 に示すように，対称性 4(90° 回転)と対称性 6(270° 回転)が他の操作より，より優れた結果が得られるということがわかった．したがって，本論文で提案した SGA は 90° 回転の対称性(対称性 4 と対称性 6 は互いに対称する)を選択する．

### 3.2.3　アルゴリズムの評価

本節では，3.2.1 項で説明した本論文で提案したSGAと，従来の解法GA，SA2[17]，FSA[19]（SA と FSA の詳細は第 4 章で説明する）を比較対象とし，セル形状が正方形（$m$=$n$）の 100(10×10)，225(15×15)と 400(20×20)の問題サイズを扱った．この問題においては物流コスト $F$ と隣接ファクター $G$ の各行列要素の値は 0 が与えられる率（スパース度）[1]を設定し，スパースが 0 以外の場合は 1～5 の整数をランダムに与えることで例題データを作成した．スパース度は式（3.1）で定義される．

$$\text{スパース度}=\frac{\text{0である物流コスト}F\text{と隣接ファクター}G\text{の個数}}{\text{物流コスト}F\text{と隣接ファクター}G\text{の個数}} \tag{3.1}$$

また，それぞれの問題に対して 0%，30%，60%及び 90%とした．計算時間が同程度のように 4000 秒を設定して，各アルゴリズムで 3 回ずつ解いた．実験結果については表 3.4，表 3.5，表 3.6 と表 3.7 に示す．また，提案するアルゴリズムであるSGA の特徴をさらに明らかに示すため，1 つの問題に対して実験経過のグラフを作成した．そのグラフを図 3.13 に示す．また SGA のパラメータ設定値を表 3.2 に示す．

**表3.4　目的関数比較表 1**

①:問題サイズ　　②:試行番号　　$\delta$:スパース度

| ① | ② | GA | | | | SA2 | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | $\delta$=0 | $\delta$=30 | $\delta$=60 | $\delta$=90 | $\delta$=0 | $\delta$=30 | $\delta$=60 | $\delta$=90 |
| 100 | 1 | 2222900 | 1537180 | 855070 | 206820 | 2208480 | 1517550 | 843090 | 201470 |
| | 2 | 2219450 | 1533220 | 865980 | 207450 | 2207050 | 1517670 | 840300 | 200700 |
| | 3 | 2219730 | 1533250 | 859740 | 207860 | 2207260 | 1518140 | 840460 | 201260 |
| 225 | 1 | 16378590 | 11421550 | 6497300 | 1663660 | 16240080 | 11292430 | 6364820 | 1598410 |
| | 2 | 16384600 | 11438140 | 6475380 | 1652060 | 16237810 | 11286280 | 6354640 | 1595410 |
| | 3 | 16379510 | 11457960 | 6472310 | 1651940 | 16240170 | 11294180 | 6356800 | 1594750 |
| 400 | 1 | 67925120 | 47728840 | 27110780 | 6890670 | 67280500 | 46835680 | 26587070 | 6690640 |
| | 2 | 67975728 | 47884480 | 27147710 | 6881120 | 67279840 | 46847020 | 26601940 | 6687410 |
| | 3 | 67939280 | 47868212 | 27153590 | 6888220 | 67286000 | 46850620 | 26590320 | 6673810 |

**表3.5　目的関数比較表 2**

①:問題サイズ　　②:試行番号　　$\delta$:スパース度

| ① | ② | FSA | | | | SGA | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | $\delta$=0 | $\delta$=30 | $\delta$=60 | $\delta$=90 | $\delta$=0 | $\delta$=30 | $\delta$=60 | $\delta$=90 |
| 100 | 1 | 2209020 | 1519960 | 848130 | 209950 | 2203740 | 1510900 | 835340 | 198530 |
| | 2 | 2207890 | 1518850 | 851320 | 211380 | 2203310 | 1511620 | 834910 | 199800 |
| | 3 | 2208620 | 1519590 | 850470 | 209950 | 2204930 | 1513940 | 835670 | 198310 |
| 225 | 1 | 16272110 | 11310540 | 6386940 | 1620820 | 16212280 | 11261680 | 6305020 | 1564450 |
| | 2 | 16265250 | 11326250 | 6383430 | 1630550 | 16208140 | 11242420 | 6315290 | 1553750 |
| | 3 | 16256650 | 11321280 | 6373410 | 1620170 | 16196660 | 11243250 | 6310630 | 1554060 |
| 400 | 1 | 67367310 | 46908290 | 26674420 | 6889700 | 67238912 | 46832056 | 26544280 | 6627390 |
| | 2 | 67358390 | 46918790 | 26641660 | 6910750 | 67247288 | 46749808 | 26493010 | 6639660 |
| | 3 | 67353250 | 46898900 | 26705590 | 6936710 | 67268152 | 46808412 | 26573620 | 6656790 |

**表3.6　実験結果 1(平均値と標準偏差)**

①:問題サイズ　　②:目的関数　　③:平均値　　④:標準偏差　　$\delta$:スパース度

| ① | ② | GA | | | | SA2 | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | $\delta$=0 | $\delta$=30 | $\delta$=60 | $\delta$=90 | $\delta$=0 | $\delta$=30 | $\delta$=60 | $\delta$=90 |
| 100 | ③ | 2220693 | 1534550 | 860263 | 207377 | 2207596 | 1517786 | 841283 | 201143 |
| | ④ | 1471.1 | 1753.3 | 3811.1 | 371.1 | 588.9 | 235.6 | 1204.4 | 295.6 |
| 225 | ③ | 16380900 | 11439217 | 6481663 | 1655887 | 16239353 | 11290963 | 6358753 | 1596190 |
| | ④ | 2466.7 | 12495.6 | 10424.4 | 5182.2 | 1028.9 | 3122.2 | 4044.4 | 1480 |
| 400 | ③ | 67946709 | 47827177 | 27137360 | 6886670 | 67282113 | 46844440 | 26593110 | 6683953 |
| | ④ | 19345.8 | 65558.2 | 17720 | 3700 | 2591.1 | 5840 | 5886.7 | 6762.2 |

表3.7　実験結果 2(平均値と標準偏差)

①:問題サイズ　②:目的関数　③:平均値　④:標準偏差　$\delta$:スパース度

| ① | ② | FSA | | | | SGA | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | $\delta$=0 | $\delta$=30 | $\delta$=60 | $\delta$=90 | $\delta$=0 | $\delta$=30 | $\delta$=60 | $\delta$=90 |
| 100 | ③ | 2208510 | 1519466 | 849973 | 210426 | 2203993 | 1512153 | 835306 | 198880 |
| | ④ | 413.3 | 411.1 | 1228.9 | 635.6 | 624.4 | 1191.1 | 264.4 | 613.3 |
| 225 | ③ | 16264670 | 11319357 | 6381260 | 1623846 | 16205693 | 11249117 | 6310313 | 1557420 |
| | ④ | 5346.7 | 5877.8 | 5233.3 | 4468.9 | 6022.2 | 8375.6 | 3528.9 | 4686.7 |
| 400 | ③ | 67359650 | 46908660 | 26673890 | 6912386 | 67251451 | 46796759 | 26536970 | 6641280 |
| | ④ | 5106.7 | 6753.3 | 21486.7 | 16215 | 11134.2 | 31300.4 | 29306.7 | 10340 |

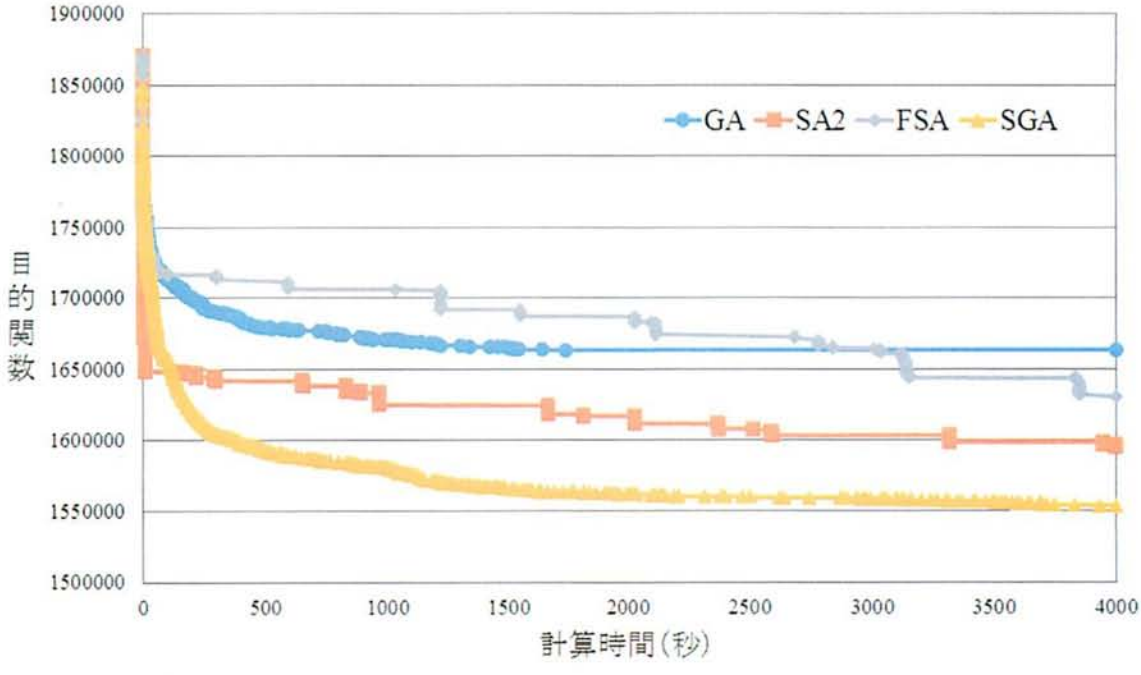



図3.13　実験経過の 1 例(問題サイズ 225, $\delta$=90, 試行番号 2 の問題)

本実験から得られた結果について説明する. 表 3.4 と表 3.5 は各サイズの問題に対して, 各々3 回実行して得られた結果である. 表 3.6 と表 3.7 は目的関数の平均値と標準偏差値である. 実験結果から SGA は 3 つの従来解法よりよい解を得ることができた. 提案するアルゴリズムである SGA が優れた結果になったのは, 二次元配置の対称性とい

う特徴を利用した結果であると思われる. 本論文では, 従来の解法と違って, 解空間を縮小することではなく, アルゴリズムの探索能力を高めるために, 対称の配置を利用すると考えられる. その改善結果を図 3.13 で明らかに示す. SGA は一旦探索が止まったら, 対称の配置を使い, 新たな領域(同じ目的関数を持っている対称の配置)に移動し, 探索を続ける. そのため, 探索の後半に行っても, 他の解法より短い時間でよい結果を出し続けられる. しかしながら, 標準偏差により, SGA はSA2よりばらつきが大きく, 良い結果が探し出せるが, 悪い結果が出るリストも高い. そのため, 偏差をもっと抑えられるもう 1 つの部分一致交叉と隣接選好を取り入れた二次元交叉であるハイブリッド解法を提案した. また, スパースについての考察結果としては, 今回の実験結果により, どんなスパースの問題に対しても, SGA は他の解法よりよい結果を得ることができる. そのため, SGA は執行時間 4000 秒までスパースとサイズにかかわらず, 中大規模の問題に対して, 1 つの効率的解法と考えられる.

# 第4章　一致交叉と隣接選好を取り入れた二次元交叉のハイブリットなアルゴリズム

本章では，本論文で提案したもう 1 つのアルゴリズムである「部分一致交叉と隣接選好を取り入れた二次元交叉のハイブリットなアルゴリズム」(Hybrid method based on Genetic Algorithm ; HGA)の説明を行う．まず 4.1 節では本提案アルゴリズムに関連する先行研究についての説明を行い，続いて 4.2 節ではアルゴリズムの詳細な説明と数値実験の結果からの評価を行う．

## 4.1.　先行研究におけるアルゴリズムとの関連

本節では，提案するアルゴリズム HGA を説明する前に，本論文に用いられたもう 1 つの先行解法，シミュレーテッド・アニーリング法(Simulated Annealing ; SA)を紹介する．SA は第 2 章で説明した通り，高温で熱した金属の温度を徐々に下げて冷却する過程を計算機でシミレーションした最適手法である．まず 4.1.1 項で先行研究の 2 点近傍探索方法についての説明を行う．続いて 4.1.2 項で今ら[17]による隣接選好を考慮し近傍作成方法についての説明を行う．

### 4.1.1　2 点近傍探索方法

2 点近傍探索方法とは SA で近傍探索方法として 2 点交換法を用いた方法である．また，2 点交換法とは任意的に 2 箇所のセルに配置された設備を入れ替えることによって近傍探索を行う手法である．その操作の例を図 4.1 に示す．

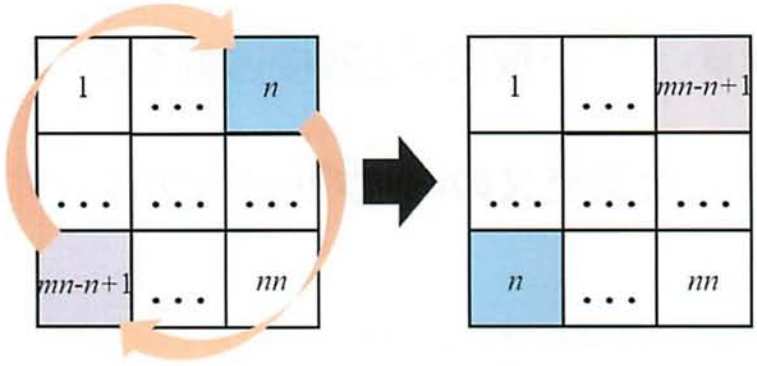



**図4.1　2 点交換法**

2 点近傍探索方法の計算順概略

Step1　:温度を初期化する.

Step2　:初期解(初期状態)を生成して, 現行解として設定する.

Step3　:終了判定(事前に設定した計算時間を超えた場合)を行う. 条件を満たしていれば現行解を表示して終了, 満たしていなければ Step4 に進む.

Step4　:2 点交換法による近傍探索方法で評価をする.

Step5　:SA 法の受理確率に基づいて受理判定を行う. 受理の場合 Step6 へ, そうでなければ Step7 へ進む.

Step6　:現行解を更新する.

Step7　:設定した繰り返し回数に達していれば次の温度を設定し, そうでなければそのままで Step3 に戻る.

### 4.1.2　隣接選好を考慮した近傍作成法

本項では, 先行研究で今ら[17]によって提案した隣接選好を考慮した近傍作成法（SA2）について説明を行う. この方法は 4.1.1 項で紹介した方法に基づいて 2 点交換法を隣接選好に変えて開発された方法である. ここで, 異なる隣接選好の部分を紹介する. 隣接選好とは, ある設備$k_1$において別の設備$k_2$が隣接すべき度合いであり, 設備$k_1$-$k_2$間の単位距離あたりの物流コスト F および隣接ペナルティ G の総和（$v(k_1,k_2)$）であり, 次の(4.1)式で定義される.

$$v(k_1,k_2)=F(k_1,k_2)+F(k_2,k_1)+G(k_1,k_2)+G(k_2,k_1) \tag{4.1}$$
$$k_1 \neq k_2, \quad k_1=1,2,...,mn, \quad k_2=1,2,...,mn$$

設備$k_1$に対し，$v(k_1,k_2)$が高い順から，隣接選好順位が高い設備$k_2$となる．隣接選好を考慮した近傍作成法とは，ランダムに 1 つの設備を選択してこの設備の周囲に隣接選好に基づいて設備を配置する．図 4.2～図 4.5 に示すように周りの設備個数が 4 つの場合を説明する．また，隣接設備数が 2 または 3 の場合も，隣接設備数が異なるだけで同様に隣接選好順位の高次の設備を用いて，近傍を作成する．

隣接選好を考慮した近傍作成法の計算順概略

<STEP 1> ：ランダムに設備の選択をする（今回は設備$a$にする）．

<STEP 2> :設備番号$a$の上下左右の配置されている$a-n$，$a-1$，$a+1$，$a+n$があることを確認する．

<STEP 3> :設備$a$の隣接選好順位上位 4 位の設備番号，つまり来て欲しい設備を探し出す($b$，$c$，$d$，$e$)．

<STEP 4> :$a$の設備に隣接されている Step2 の 4 つの設備と実際に来て欲しい Step3 の 4 つの設備が同じ場合，Step5 に進み，そうでなければ，設備$a$の周りがない設備と来て欲しい設備を交換する．

<STEP 5> :来て欲しい設備を上下左右に配置して全ての配置方を列挙して，その中から目的関数値が一番低い配置案を探す．

| | | | |
|---|---|---|---|
| 1 | ... | a-n | m |
| b | a-1 | a | a+1 |
| c | ... | a+n | ... |
| mn-n+1 | d | e | mn |

**図4.2　ランダムに設備を選択する**

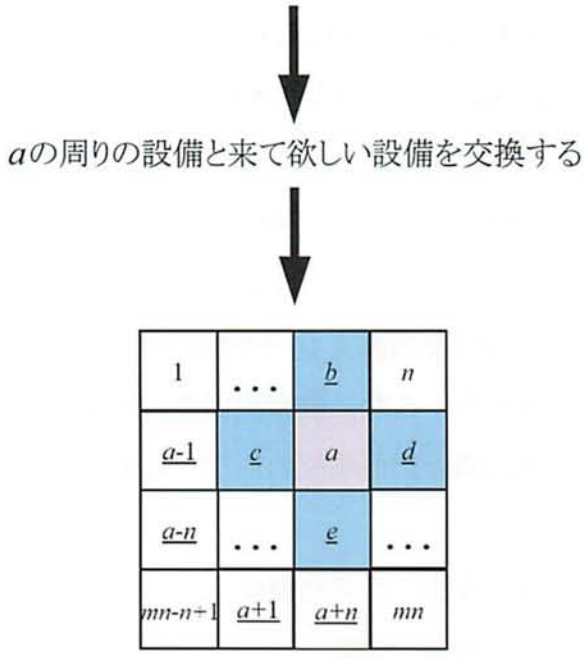



**図4.3　近傍探索の手順(a)**

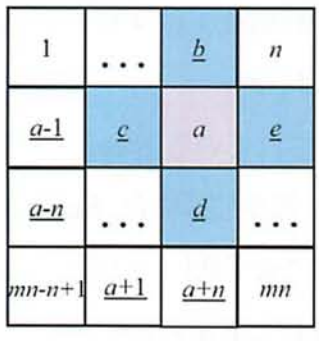



**図4.4　近傍探索の手順(b)**

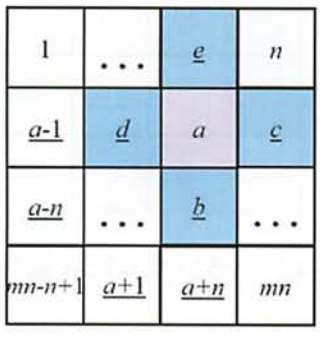



**図4.5　近傍探索の手順(c)**

### 4.1.3　近傍探索手順を考慮した SA アルゴリズム

本項では実験対象としたもう 1 つの先行解法である近傍探索手順を考慮した SA アルゴリズム(FSA)を説明する. FSA は鈴木ら[19]により提案され, 上記で紹介した SA2 を基に探索順次を工夫した探索方法である. FSA では, 探索する前まず, 各設備 $i$ の関連度 $w_i$ を単位距離あたりの物流コスト $F$ および隣接ペナルティ $G$ で表示された次の式(4.2)で算出する.

$i$=1,2, ...,$mn$に対して,

$$w_i = \sum_{j=1}^{mn} (F(i,j) + F(j,i) + G(i,j) + G(j,i)) \tag{4.2}$$

この関連度が小さいほど他の設備との関連性が低く, 関連が高い設備の近くに置かれなくてもよい場合が多いと経験的に判断できる[19]. また, 探索順序を F 形(図 4.6)で配置案を作成していく.

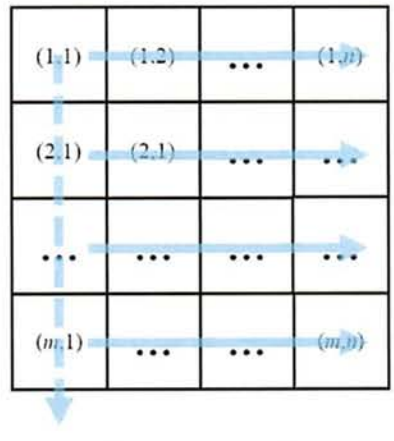



**図4.6　近傍探索の順序**

FSA では, 最初にセル(1,1)に関連度が最小の設備をおいて, その隣接セルに置かれるべき設備の関連度上位の 2 つの設備を, セル(1,2)と(2,1)に置かれている設備と交換して配置案を作成する. また, セル(1, 2)と(2, 1)に置かれた設備を相互に入れ換えて新たな配置案を列挙して, それら配置案のうち, 目的関数が最小の配置案を保存しておき, 次の受理判定においてアニーリングの基準で受理または棄却とする. この操作

をセル(1, 2), (1, 3), ... ,(1, $n$), (2, 1), ... , ($m$, $n$)と行っていくことで，局所的に好ましい配置が順次形成され，全体として優れた配置に組み上がっていく．

## 4.2. 一致交叉と隣接選好を取り入れた二次元交叉のハイブリットアルゴリズム

本節では，本論文では提案した 2 つ目の解法，一致交叉と隣接選好を取り入れた二次元交叉のハイブリットなアルゴリズム(Hybrid method based on Genetic Algorithm ; HGA)についての説明を行う．まず 4.2.1 項で，アルゴリズムの説明を行い，続いて 4.2.2 項で，実際に行った数値実験の結果とその評価について述べる．

### 4.2.1 アルゴリズムの説明

3.2 節で説明したアルゴリズム SGA は GA を基に提案したが，交叉の手順を行う前に，二次元の配置を必ず染色体のような一次元の列に戻す．そのため，交叉する際に，配置全体が一列とし，すべての設備に対して左右のセルに配置された設備との関係だけ考えて配置案を評価して，上下の設備との関係が考慮されていない．結局，交叉すると，二次元の状態で保存された設備間の関係が破られて，探索の効率が悪くなる．そのため，網のような横と縦の生産ラインを両方最適化できるもう 1 つの解法，二次元の交叉方法，HGA[付録 2]を提案した．このアルゴリズムの特徴は交叉が行われる対象が一次元の列ではなく，二次元の領域であるので，横と縦の優れた遺伝子(配置)を共に保存し，次世代に引き継ぐことができる．また，本論文では，主に大規模の問題を解決するため，中小規模の問題に対して，密接的に探索できる先行研究の解法，SA2 を利用すると考える．もし設定した時間を超えて，よりよい結果が探し出せなかったら，SA2 に切り替えて，より厳密的な探索を続ける．そこで，ハイブリッドなアルゴリズムとして HGA がいずれかの問題に対してもよりよい結果が出せると思う．

HGA は GA を基に提案したので，操作の手順が GA とほぼ同じである．まず，初期集団を作成して，その中の最良値を保存する．続いて，集団からランダムに 2 個体を選択して交叉させる．ただし，ここで二次元の交叉方法で交叉する．また，二次元交叉の終了条件を満たせば，現時点まで探し出された最良個体の目的関数と配置を SA2 に

バトンタッチし，最後まで探索を続ける．HGA の概略フローチャートを図 4.7 に示す，色付の部分が GA に対して HGA で改善また追加された操作である．

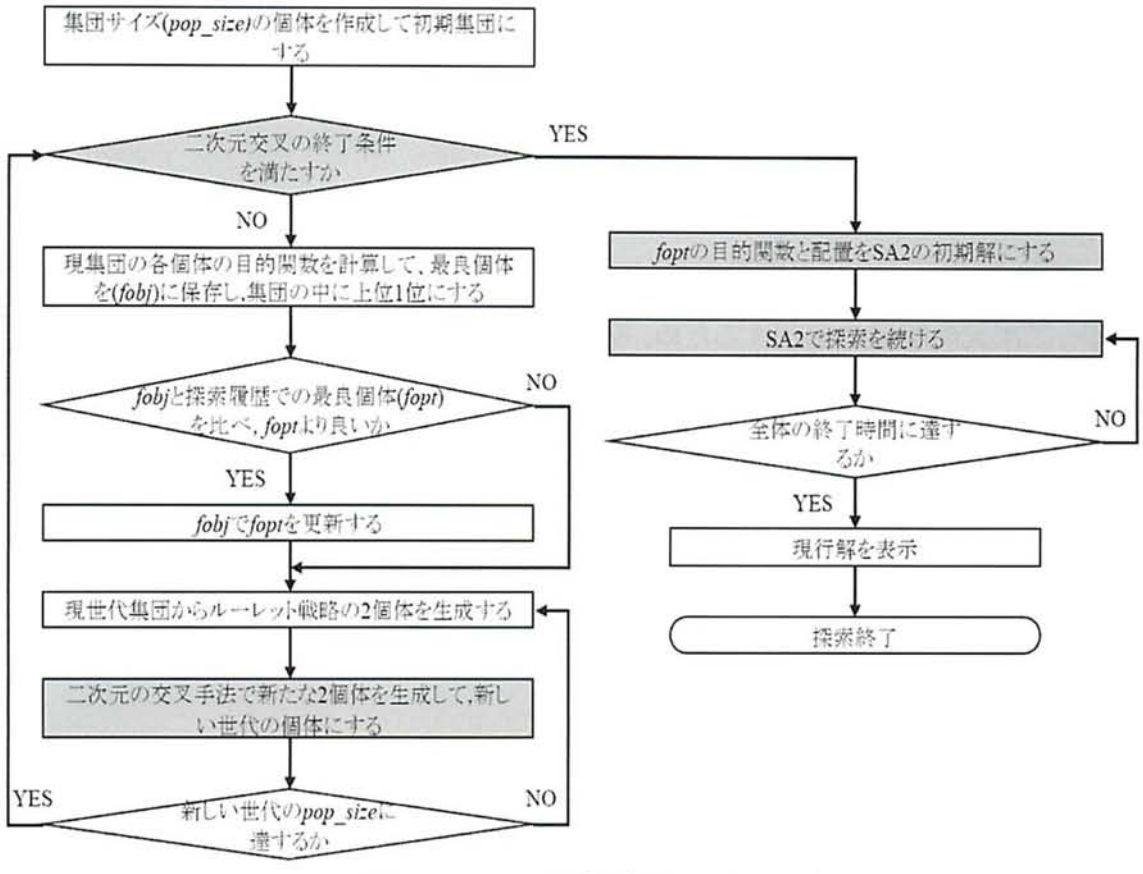



**図**4.7　HGA **の概略フローチャート**

上記説明したように，HGA は本論文で提案した二次元交叉の手法で子孫を作成する．その操作は以下の手順に従う．

Step1　:初期集団を生成して現世代集団とし，各個体の目的関数値と適応度を計算する．世代カウンタが 0 とする．

Step2　:二次元交叉部分の終了条件（*M* 秒以内，よりよい結果が探し出せなかったら，SA2 に移る）が満たされれば，Step9 へ，もしそうでなければ，Step3 に進む．

Step3　:現世代の最良個体を *fobj* に保存し，集団での上位 1 位する．

Step4　: *fobj* と探索履歴の最良個体 *fopt* の目的関数を比較し，*fobj* がよい場合，*fobj* で *fopt* を更新する．

Step5　:現世代集団からルーレット戦略で 2 個体を選ぶ．

Step6　:二次元交叉の手法で新たな 2 個体を生成する．

Step7　:次世代集団が集団サイズに達したら現世代とし，各個体の目的関数値と適

応度を計算する. 世代カウンタを 1 増加する.

Step8 :最終世代に達したら Step2 に戻り, そうでなければ Step5 に戻る.

Step9 : *fopt* の目的関数と *fopt* の配置を SA2 の初期解にし, SA2 で探索を続ける.

Step10 :プログラムの全体終了時間(*cpu_time*)に達したら終了. そうでなければ Step9 に戻る.

本論文では, HGA の交叉手法は 3.1.2 項で説明した部分一致交叉を基に提案したが, 二次元の配置に適用するため, 交叉する際, 二次元の配置が"染色体"として直接交叉するので, 交叉される部分も二次元の"遺伝子"の形にしなければならない. すなわち, 二次元交叉方法の遺伝子とは, 図4.8と図4.9のような領域である. 遺伝子を一次元から二次元に変換することにより, 配置全体がよく考慮されることだけではなく, GA において遺伝子の次元が頻繁に変えられることも省略できるので, 効率性が高まると考えられる. ここで, そのオリジナルな交叉手法(上記の Step6)を以下に説明する. 操作の手順を図 4.8 と図 4.9 に示す.

<STEP 1> :1 個体に対して, ランダムに交叉点とするセルを選択する.

<STEP 2> :交叉の領域(交叉する設備の個数)の長さ$a$と幅$b$をランダムに決める.

<STEP 3> :もう 1 つの個体で交叉点を選択し(ただし, 今回の交叉点は$a \times b$の積が入るに選択する), $a \times b$の領域で交叉する範囲を確定する.

<STEP 4> :2 つの交叉領域(選択された領域での設備)を入れ換える.

<STEP 5> :重複した設備番号を解消する.

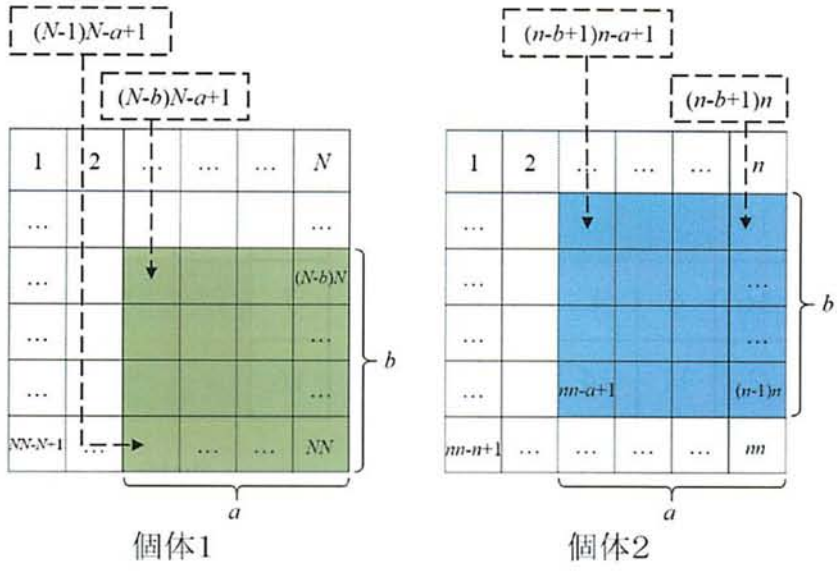



**図4.8　HGA の交叉手順(交叉する前)**

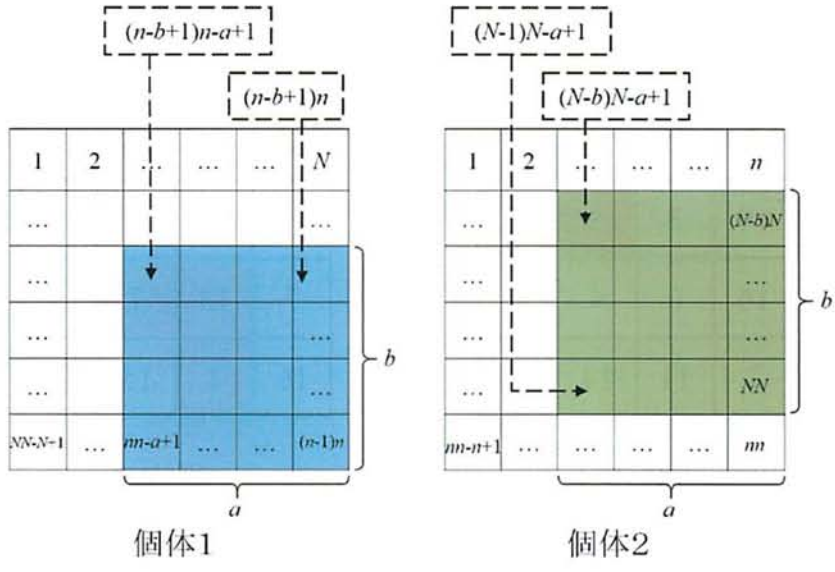



**図4.9　HGA の交叉手順(交叉した後)**

HGA において，致死遺伝子(1 つの設備が 2 回以上配置された設備)を解消するため，部分一致交叉と同じ方法で処理する．ここで，その操作(上記の<STEP 5>)を説明するため，問題サイズが 4×4 の一般例を挙げる．

▸ step 1：2 個体に対して，同じ交叉の領域を選択する．今回は長さ a＝3，幅 b＝2 に

し，図 4.10 に示す．

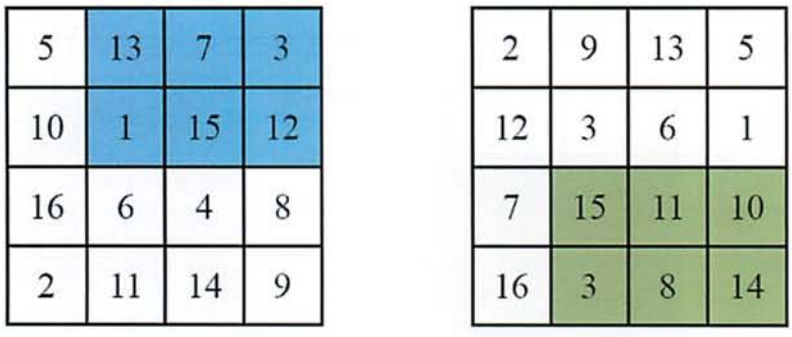



**図4.10　致死遺伝子の解消(*a*)**

▸ step 2:領域内の設備を交換後に，重複した設備の番号を確認する，図 4.11 に示す．（線に囲まれる部分が交換されてきた設備，赤色と黄色の設備がそれぞれの配置において，もとの設備と重複した設備．）

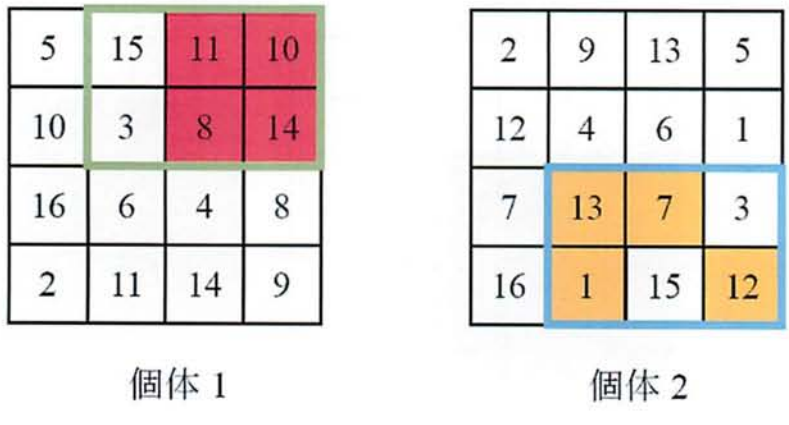



**図4.11　致死遺伝子の解消(*b*)**

▸ step 3:生成された新たな 2 個体に対して，それぞれの衝突した設備を重複がなくなるまで 1 個ずつ交換する(交換はセルの並ぶ順番に従って，すなわち左上から右下までの順で行う．ここで，例においては，11と13，10と7，8と1，14 と 12 のようにチェンジする．その交換の過程を図 4.12～図 4.15 に示す)．

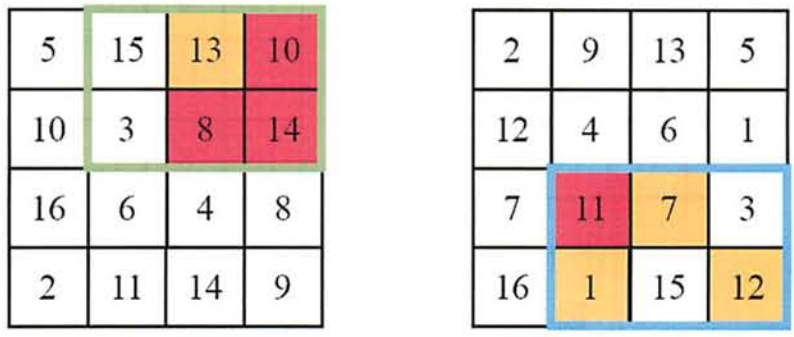



**図4.12　設備 11 と設備 13 を交換する**

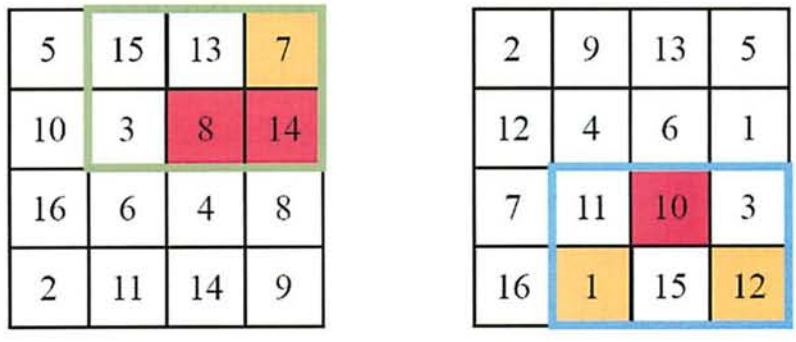



**図4.13　設備 10 と設備 7 を交換する**

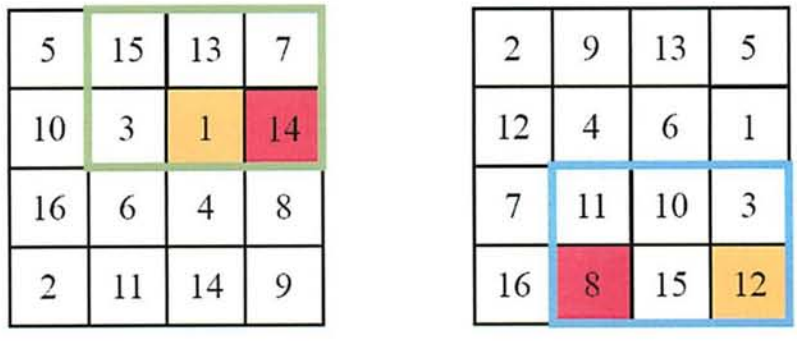



**図4.14　設備 8 と設備 1 を交換する**

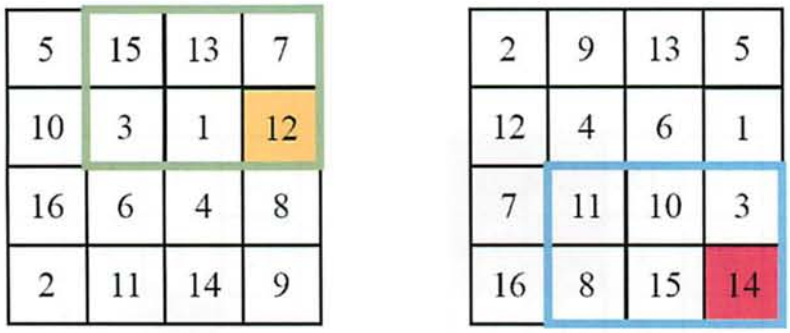



**図4.15　設備 14 と設備 12 を交換する**

### 4.2.2　アルゴリズムの評価

本節では，提案した SGA，HGA と，従来の解法 (SA2)[17]，(FSA)[19]，を比較対象とし，セル形状が正方形($m$=$n$)，100(10×10)，169(13×13)，225(15×15)，289(17×17)と 400(20×20)の 5 つサイズの問題を扱った．また，3.2.2 項と同じようなスパース度がそれぞれ 0%，30%，60%及び 90%の 4 つ，合計 20 個の問題を作成する．今回の計算時間を 8000 秒に設定して，各アルゴリズムで 3 回ずつ解いた．HGA のパラメータは表 4.1 のように設定する．実験結果は表 4.2，表 4.3，表 4.4 及び表 4.5 に示す．また，各アルゴリズムの探索過程を明確するために 1 つの問題に対して実験経過のグラフを作成した．図 4.16 に示す．

**表4.1　パラメータ設定値(HGA)**

| 解法 | パラメータ | パラメータの意味 | パラメータの設定値 |
|---|---|---|---|
| GA | *cpu _time* | 実験の計算時間 | 8000 |
| | *pop _size* | 集団サイズ(集団の個体数) | 40 |
| | *px* | 交叉率 | 100 |
| | *pm* | 突然変異率 | 90 |
| | *M* | SA2 に切り替える条件(秒) | 180 |
| SA2 | *t* | 初期温度 | 5000 |
| | *n _2opt* | 2 点近傍探索回数 | 11 |
| | *bt* | クーリング係数 | 0.99 |
| | *n _ep* | エポック数 | 20*mn* |

本実験では, SGA と比較するため, HGA においての GA に関する部分のパラメータを SGA と同じ数値を設定した. *M* の設定理由は数値実験により, 二次元交叉の方法が最悪の場合, 180 秒で現時点の最良解よりよい結果を出すので, 180 秒を超えたら, なかなかよい結果が出せない. そのため, 今回の実験は 180 秒に設定した. また, SA 部分のパラメータは先行研究[17]と同じ数値で設定された.

**表4.2　目的関数比較表 3**

①:問題サイズ　　②:試行番号　　$\delta$:スパース度

| ① | ② | FSA | | | | SA2 | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | $\delta$=0 | $\delta$=30 | $\delta$=60 | $\delta$=90 | $\delta$=0 | $\delta$=30 | $\delta$=60 | $\delta$=90 |
| 100 | 1 | 2188730 | 1493780 | 818710 | 183400 | 2188860 | 1493080 | 818340 | 182980 |
| | 2 | 2188130 | 1493680 | 818100 | 182750 | 2188140 | 1493300 | 818390 | 182870 |
| | 3 | 2187980 | 1493580 | 818050 | 182890 | 2189230 | 1492870 | 818000 | 183150 |
| 169 | 1 | 7951960 | 5474760 | 3020670 | 710850 | 7951700 | 5473350 | 3017880 | 709240 |
| | 2 | 7954090 | 5477990 | 3013980 | 711030 | 7953700 | 5478390 | 3012950 | 710040 |
| | 3 | 7953660 | 5476860 | 3017420 | 711090 | 7950420 | 5474220 | 3017890 | 711040 |
| 225 | 1 | 16220690 | 11257990 | 6328790 | 1573380 | 16203110 | 11249340 | 6325940 | 1563680 |
| | 2 | 16215160 | 11262780 | 6319450 | 1576030 | 16210720 | 11245770 | 6309810 | 1569060 |
| | 3 | 16213320 | 11250150 | 6317530 | 1568370 | 16205160 | 11246180 | 6304270 | 1565420 |
| 289 | 1 | 30150020 | 20945040 | 11854930 | 2941310 | 30122050 | 20930940 | 11825960 | 2899320 |
| | 2 | 30142120 | 20953160 | 11856150 | 2931960 | 30122580 | 20935740 | 11830220 | 2892180 |
| | 3 | 30137030 | 20957950 | 11836550 | 2944200 | 30117840 | 20923120 | 11824310 | 2887430 |
| 400 | 1 | 67275910 | 46809340 | 26632030 | 6800240 | 67249260 | 46789000 | 26552710 | 6675740 |
| | 2 | 67282410 | 46846740 | 26641660 | 6822250 | 67256720 | 46818570 | 26557020 | 6677220 |
| | 3 | 67288080 | 46824720 | 26643770 | 6827040 | 67244590 | 46796800 | 26565310 | 6659840 |

**表4.3　目的関数比較表 4**

①:問題サイズ　　②:試行番号　　$\delta$:スパース度

| ① | ② | SGA | | | | HGA | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | $\delta$=0 | $\delta$=30 | $\delta$=60 | $\delta$=90 | $\delta$=0 | $\delta$=30 | $\delta$=60 | $\delta$=90 |
| 100 | 1 | 2196400 | 1508800 | 826540 | 188540 | 2187630 | 1493400 | 818370 | 182480 |
| | 2 | 2196520 | 1507210 | 829210 | 188010 | 2187580 | 1493200 | 818340 | 182520 |
| | 3 | 2195880 | 1510280 | 828500 | 188030 | 2187690 | 1493030 | 818450 | 182650 |
| 169 | 1 | 7966810 | 5502980 | 3049360 | 729010 | 7949790 | 5472250 | 3015830 | 709240 |
| | 2 | 7960640 | 5504210 | 3035390 | 724620 | 7950830 | 5474780 | 3015850 | 710140 |
| | 3 | 7965020 | 5481070 | 3032350 | 723890 | 7951450 | 5473360 | 3015200 | 709550 |
| 225 | 1 | 16189220 | 11231290 | 6293200 | 1555320 | 16188990 | 11224780 | 6291530 | 1557580 |
| | 2 | 16197910 | 11222420 | 6307910 | 1547780 | 16182700 | 11226050 | 6290370 | 1553070 |
| | 3 | 16201360 | 11234060 | 6302950 | 1545380 | 16184420 | 11208830 | 6287440 | 1539700 |
| 289 | 1 | 30070690 | 20907560 | 11783710 | 2913850 | 30024980 | 20854770 | 11744190 | 2816480 |
| | 2 | 30092950 | 20897600 | 11752420 | 2913970 | 30043880 | 20839810 | 11727680 | 2824690 |
| | 3 | 30097480 | 20860370 | 11745370 | 2911500 | 30030390 | 20859700 | 11691250 | 2816280 |
| 400 | 1 | 67197152 | 46825104 | 26497800 | 6573260 | 67005544 | 46600232 | 25941270 | 6521650 |
| | 2 | 67162360 | 46735104 | 26477770 | 6598930 | 67040436 | 46581096 | 26410100 | 6494920 |
| | 3 | 67202256 | 46776008 | 26565220 | 6602620 | 67063564 | 46589632 | 26457390 | 6503470 |

**表4.4　実験結果 3(平均値と標準偏差)**

①:問題サイズ　②:目的関数　③:平均値　④:標準偏差　$\delta$:スパース度

| ① | ② | FSA | | | | SA2 | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | $\delta$=0 | $\delta$=30 | $\delta$=60 | $\delta$=90 | $\delta$=0 | $\delta$=30 | $\delta$=60 | $\delta$=90 |
| 100 | ③ | 2188280 | 1493680 | 818287 | 183013 | 2188743 | 1493083 | 818243 | 183000 |
| | ④ | 300 | 66.7 | 282.2 | 257.8 | 402.2 | 144.4 | 162.2 | 100 |
| 169 | ③ | 7953237 | 5476537 | 3017357 | 710990 | 7951940 | 5475320 | 3016240 | 710107 |
| | ④ | 851.1 | 1184.4 | 2251.1 | 93.3 | 1173.3 | 2046.7 | 2193.3 | 622.2 |
| 225 | ③ | 16216390 | 11256973 | 6321923 | 1572593 | 16206330 | 11247097 | 6313340 | 1566053 |
| | ④ | 2866.7 | 4548.9 | 4577.8 | 2815.6 | 2926.7 | 1495.6 | 8400 | 2004.4 |
| 289 | ③ | 30143057 | 20952050 | 11849210 | 2939157 | 30120823 | 20929933 | 11826830 | 2892977 |
| | ④ | 4642.2 | 4673.3 | 8440 | 4797.8 | 1988.9 | 4542.2 | 2260 | 4228.9 |
| 400 | ③ | 67282133 | 46826933 | 26639153 | 6816510 | 67250190 | 46801457 | 26558347 | 6670933 |
| | ④ | 4148.9 | 13204.1 | 4748.9 | 10846.7 | 4353.3 | 11408.9 | 4642.2 | 7395.6 |

**表4.5　実験結果 4(平均値と標準偏差)**

①:問題サイズ　②:目的関数　③:平均値　④:標準偏差　$\delta$:スパース度

| ① | ② | SGA | | | | HGA | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | $\delta$=0 | $\delta$=30 | $\delta$=60 | $\delta$=90 | $\delta$=0 | $\delta$=30 | $\delta$=60 | $\delta$=90 |
| 100 | ③ | 2196267 | 1508763 | 828083 | 188193 | 2187633 | 1493210 | 818387 | 182550 |
| | ④ | 257.8 | 1035.6 | 1028.9 | 231.1 | 37.8 | 126.7 | 42.2 | 66.7 |
| 169 | ③ | 7964157 | 5496087 | 3039033 | 725840 | 7950690 | 5473463 | 3015628 | 709643 |
| | ④ | 2344.4 | 10011.1 | 6884.4 | 2113.3 | 600 | 877.8 | 284.4 | 331.1 |
| 225 | ③ | 16196163 | 11229257 | 6301353 | 1549493 | 16185370 | 11219887 | 6289780 | 1550117 |
| | ④ | 4628.9 | 4557.8 | 5435.6 | 3884.4 | 2413.3 | 7371.1 | 1560 | 6944.4 |
| 289 | ③ | 30087040 | 20888510 | 11760500 | 2913107 | 30033083 | 20851427 | 11721040 | 2819150 |
| | ④ | 10900 | 18760 | 15473.3 | 1071.1 | 7197.8 | 7744.4 | 19860 | 3693.3 |
| 400 | ③ | 67187256 | 46778739 | 26513597 | 6591603 | 67036515 | 46590320 | 26269587 | 6506680 |
| | ④ | 16597.3 | 30910.2 | 34415.6 | 12228.9 | 20647.1 | 6608 | 218877.8 | 9980 |

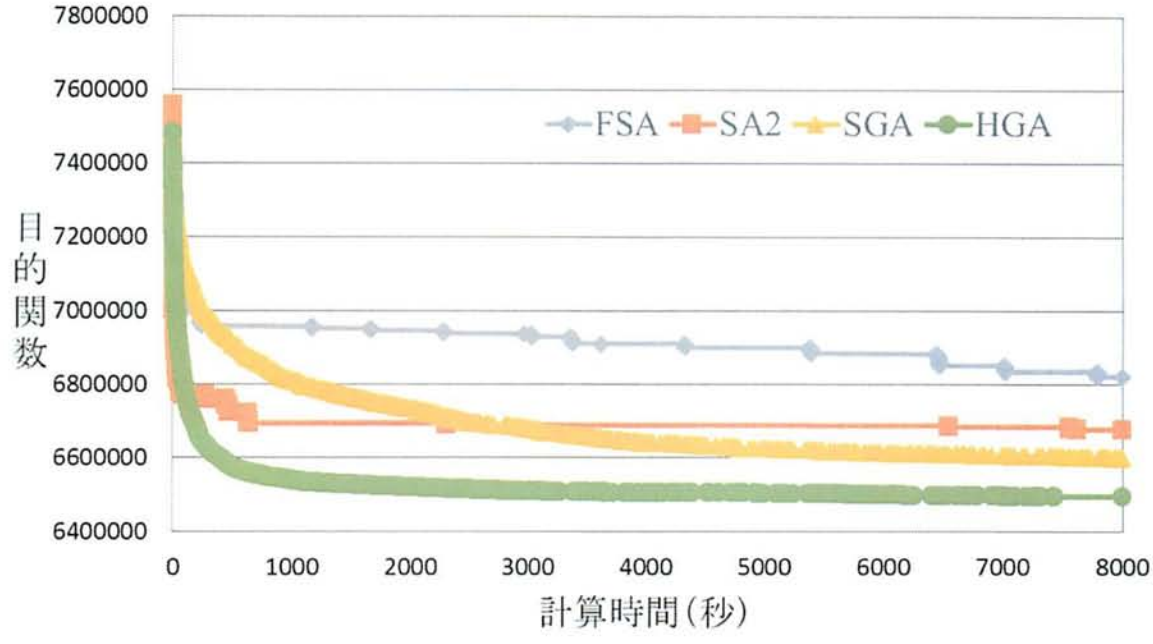



**図4.16　実験経過の 1 例(問題サイズ 400, $\delta$=90, 試行番号 2 の問題)**

実験から得られた結果について説明する. 今回は提案アルゴリズムの効率性を確認するため, 計算時間を 8000 秒に設定し, 5 つサイズの問題を作成し, 3 回ずつ執行して得られた結果を表 4.2 と表 4.3 に示す. 表 4.4 と表 4.5 は目的関数の平均値と標準偏差値である. その結果により, HGA は SGA より探索効率を更に高められるということがわかった.

問題サイズが 100 の場合, HGA は SA2 とほぼ同じ結果を示す. その原因は, 中小規模の問題に対して, SA2 が厳密に探索できるという特徴がよく発揮される. HGA の探索後半において SA2 で探すため, 最後まで SA2 とほぼ同じ程度の結果が出せる. 逆に, サイズが 169 以上になると, HGA はほぼ全ての問題に対して, その有効性が示されてきた. その原因は大規模の問題に対して, 二次元交叉が広い解空間で探すことが得意ので, HGA において二次元交叉が主に利用され, SA2 より優れた結果が得られる. それらのことにより, 厳密探索に強い SA と広域探索に強い GA の特徴を合わせてどのような問題に対しても, ハイブリッドな解法である HGA も適用できると考える. 図 4.16 で SGA, HGA と他の実験対象を比較してより早く収束することが示された. また, 1000 秒まで HGA で得られた結果もう他の解法より優れるということがわかった. そのため, HGA は目的関数だけではなく, 計算時間にも優位性が示せるという結論が得られた.

スパースの考察については，SGAとHGAがスパースの増加に伴って，優越性がより明らかになる．3.2.3 項で説明したように，スパースは問題の複雑さを控えるために設定したパラメータである．スパースが低ければ低いほど，設備間の関係がより強くなり，問題の全体も複雑になる．それにより，FSA と SA2 という設備間の関係によって配置案を探索する方法の優越性がますます悪くなる．逆に，SGAとHGAの場合は前世代から残された遺伝子の情報により探索するので，単なる設備を狙って 1 個ずつ満足させる配置案を探すことと違い，全体が優れる配置案を求めることがメインである．そのため，サイズが大きく，複雑な問題に対して，本論文で提案した 2 つのアルゴリズムがより効率に解決できると思う．

# 第5章　　結論

本論文では，「工場に設備を配置する」場合を想定した2つの評価基準である物流コストと隣接ファクターにおいて $m$ 行 $n$ 列格子状設備配置問題を対象とし，GA の基に対称性排除の特徴を取り入れたアルゴリズム(SGA)と部分一致交叉と隣接選好を取り入れた二次元交叉のハイブリットなアルゴリズム(HGA)である近似解を求めるアルゴリズムの提案を行った．

第 3 章において SGA は配置の対称性という特徴を用いて本論文で提案された新たな解法である．配置の対称について先行研究は各設備間の相対的な位置関係が変わらないために同じ目的関数値を持つ配置を排除し，探索空間が減少できるというものである．しかし SGA では，近似解を求める GA の探索能力を更に向上させてよりよい近似解が得られるという発想から，対称の配置を造ってかく遺伝子（配置）の多様性を拡張して探索ルートを増加し，より効率的なアルゴリズムが得られるというアイディアが得た．そのため，GA の基に配置の対称性を取り入れた SGA を提案した．この解法の最大の特徴とは，GA の早めに収束することを有効的に防ぐことができる．第 3 章で説明したように，GA が一旦止まると対称性を利用する条件を満たせば，対称の配置で現目的関数を維持して遺伝子の並び方だけ変えて他の領域に移り，探索を続ける．このような操作を繰り返すので，全探索空間で移動し，よりよい結果を徐々に探し出すことができる．また，実験結果と探索経過から見ると SGA は予想通り近似解を求める大規模の問題に対してより効率的に解決することができるということがわかった．

第4章では，本論文で提案する2つ目のアルゴリズム HGA についての説明と実験を行った．HGA は本論文で扱った問題の特徴に着目して，先行研究である SA2 を取り入れた提案したハイブリッドな解法である．従来の GA は交叉を行う際，遺伝子として二次元の配置を一次元に戻さなければならないので，一次元の列に戻すと，二次元の工場に配置された設備間の情報（設備間の物流関係）が失うという欠点がある．そのため，HGA は二次元の配置を直接遺伝子にして，交叉を行う．このような操作をすれば，単一の染色体のような遺伝子を平面のような領域に変えて任意の設備に対して上下左右に置かれる設備との関係を考察でき，特に本論文で扱った隣接ファクターも考慮することができる．実際に行った数値実験の結果により，HGA は実験を行ったアルゴリズムの

中で最良の結果が得られるということがわかった．また，中小規模の問題に優位性を持つSA2の特徴も用いるので，どんな規模の問題に対してもHGAも適用できると考える．しかし，HGA は交叉の領域が正方形と長方形しかないので，他の領域の形を決める方法が残る課題である．

今後の課題としては，より複雑な問題を扱うことが考えられる．例えば，多品種，少量生産を念願において，動的施設が注目されている生産要求の変動に応じられる動的な問題を扱うことが考えられる．

また，第 4 章で提案したHGA はSA2 に切り替えると良い結果を探し出すまで準備の時間が必要だが，設備個数の増加に伴ってその時間がさらに長くなり，特に 169 以上になると，SA2 を使っても最後までより良い結果を探すのが困難という欠点がある．そのため，どんな状況においてSA2 に切り替えるべきか，またSA2 で探し出せない場合になったらどのように判断し，GA に戻って探索を続けるかなどの検討する必要があると考えられる．

# 参考文献


[1] Tompkins, J. A., White, J. A.: "*Facilities Planning*," Wiley(1984).

[2] A.Suzuki., H.Yamamoto., T.Nakamura., Y.Tsujimura: "Efficient Algorithms for Solving the Facility Layout Problem by using Symmetries," Proceedings of Eighth International Conference on Manufacturing and Management, Vol.2, pp.1155-1162 (2004).

[3] Heragu, S.: "Recent models and techniques for solving the layout problem," European Journal of Operational Research, vol.57, pp.136-144. (1992).

[4] 屓旭，鈴木淳，松井正之: "配置可能なエッジを用いた座標型設備レイアウト法に関する研究" ,日本経営工学会論文誌, Vol.50, No.6, pp.381-388 (2000).

[5] 鈴木淳: "プロセスプラントの設備レイアウト計画と多基準評価法"，日本設備管理学会誌，第13巻，第3号, pp.152-157 (2002).

[6] Foulds, L. R.: "Techniques for Facilities Layout: Deciding Which Pairs of Activities Should Be Adjacent," Management Science, Vol.29, No.12, pp.1414-1426 (1983).

[7] Houshayar, A., White, B: "Exact Optimal Solution for Facility Layout: Deciding Which Pairs of Locations Should Be Adjacent," Computers and Industrial Engineering, Vol.29, No.2, pp.177-187 (1993).

[8] Rosenblatt. M. J.: "The Facilities Layout Problem: A Multi-goal Approach," International Journal of Production Research, Vol.17, No.4, pp.323-332 (1979).

[9] Dutta, K., Sahu, N. S.: "A Multigoal Heuristic for Facilities Design Probrem: MUGHAL," International Journal of Production Research, Vol.20, No.2, pp.147-154 (1982).

[10] Beckman, M., Koopmans, T. C.: "Assignment Problems and the Location of Economic Activities," Econometrica, Vol.25, pp.53-76 (1957).

[11] Bazaraa, M. S.: "Computerized Layout Design: A Branch and Bound Approach," AIIE Transactions, Vol.7, No.4, pp.432-438 (1975).

[12] Burkard, R. E.: "Quadratic Assignment Problems," European Journal of Operational Research, Vol.15, pp.283-289 (1984).

[13] Golany, B., Rosenblatt, M. J.: "A Heuristic Algorithm for the Quadratic Assignment Formulation to the Plant Layout Problem," International Journal of Production Research, Vol.27, No.2, pp.293-308 (1989).

[14] Lacksonen, T. A., Enscore, E. E. Jr. "Quadratic Assignment Algorithms for the Dynamic Layout Problem," International Journal of Production Research, Vol.31, No.3, 503-517 (1993).

[15] 西川禕一，三宮信夫，茨木俊秀:「岩波講座情報科学19 最適化」，岩波書店. (1982).

[16] 山本久志，鈴木淳，中村智明:"対称性を利用した設備最適配置問題解法のための分枝限定アルゴリズム"，日本経営工学会論文誌，Vol.55, No.5, pp.293-301 (2004).

[17] GelettJr. C.D., Vecchi, M.P. Kirkpatrick, S.: "Optimization by simulated annealing," Science, Vol.220, No4598, pp.671-680 (1983).

[18] Holland, J.: "Adaptation in Natural and Artificial Systems," MIT Press (1992).

[19] Glover, F.: “Tabu Search　Part I,” ORSA Journal on Computing, pp.190-206 (1989)

[20] 鈴木淳，山本久志，中村智明:“対称性を考慮した設備配置問題のためのタブサーチを用いた解法”，日本設備管理学会平成16年度春季研究発表大会論文集，pp.95-96 (2004).

[21] 中村智明，山本久志，鈴木淳:“対称性及びセルの配置順序を考慮した設備配置問題に関する分枝限定解法”，日本設備管理学会平成16年度秋季研究発表大会論文集，pp.190-191 (2004).

[22] 鈴木淳，山本久志，中村智明:“セル型設備配置問題のための遺伝的アルゴリズムの提案”，日本経営工学会平成16年度秋季大会予稿集，pp.234-235 (2004).

[23] Goldberg, D.E. and Lingle, R.:　“Alleles, Loci and the Traveling Salesman Problem,” Proceedings of the First International Conference on Genetic Algorithms, pp.154-159 (1985).

[24] 鈴木淳，山本久志，中村智明:“セル型設備配置問題のための進化的なアルゴリズムによる解法”，日本設備管理学会誌，Vol.17, No.2，pp.25-33. (2005).

[25] 今 剛士，鈴木 淳，福士達也，山本久志:“隣接選好順位を利用した設備配置問題の解法”，日本設備管理学会平成19年度春季研究発表大会論文集，B-1-4, pp.93-96 (2007).

[26] 鈴木 淳，山本久志，辻村泰寛:“スパースな設備間関連構造を持つ設備配置問題のための近傍探索手順を考慮したSAアルゴリズム”，日本設備管理学会誌，Vol.23, No.2, p37-42 (2007).

# 謝辞

本研究を行うにあたって，終始熱心にご指導，ご教示下さいました山本久志先生ならびに獨協大学の鈴木淳先生に厚く御礼申し上げます．

また私を支えてくれた山本研究室の皆様にも，この場を借りて御礼申し上げます．

皆様本当にありがとうございました．

# 付録

付録1　プログラムソース 1 SGA …… p.52

付録2　プログラムソース 2 HGA …… p.67

# 付録1 プログラムソース1 SGA

```
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>
#include <time.h>
#define N1 401
#define N2 2
#define c_times 100

int rand_n(int a);
void read_data(void);
void ga_param(void);
void init_chrm_new(void);
void fit_calc(void);
void top_keep(void);
void dsp_opt(void);
void select(void);
void crossover(void);
void existence(void);
void mutation(void);
void ga(void);

void virus(void);
void factorial(void);

        FILE *fp1;
        int tb_leng;
        double cnt_max;
        int rand_iter=1000;
        int i,j,k,kk,l,m,n,xx;
        int     p[N1][N1];
        int c[N2+1][N1][N1]; /*  c[kk][i][j]: 評価基準 kk における unit i - j 間の費用係数 */
        int d[N2+1][N1][N1]; /*  d[kk][i][j]: 評価基準 kk における unit i - j 間の距離 */
        int dist[N2+1][N1][N1]; /* dist[kk][i][j]: 評価基準 kk における site i - j 間の距離 */
        int site[N1];   /* site[i]: unit i が置かれるサイト番号 */
        float fobj=0, fobj_crnt=0,fobj_min, fopt,count;  /* objective function */
        int cont=1,cnt_term=0,cnt_a; /* continue(cont=1) or terminate(cont=0) */
        int r1,r2, flg, flg_tl;
        double cnt_all=0.;
        double  cpu_time; /*計算時間*/
        double  cpu_max; /*計算時間上限*/
        double  t_ini; /*初期温度(SA 用)*/
        double  tb;                     /*冷却係数(SA 用)*/
        clock_t start,finish,lap;
/** GA 用変数 **/
```

```
        int cnt=0, cnt_new=0, rn;
        int chrm[60][N1], chrm_new[60][N1];  /*遺伝子, 新遺伝子 */
        int p1,p2; /*選択された個体*/
        double fitness[N1], fit_total, f_obj[N1], p_select[N1];
        int pop_size, px, pm, n_iter;
//      int c_times;
        int tmp_ga[N1], x_point, c1, c2, flag, flag_all;
        int chrm_p[3][N1], chrm_c[3][N1];
        int n_row,n_column;
        int pm_last;
        double alp;
/**factorial number 変数**/
        int
C,D,e,f,g,h,o,q,fac,column_s,column_l,row_l,site_m[N1][N1],integer[N1],integer_p[N1],site_p[N1]
[N1],site_pp[N1];
        float f_opt,F_obj;


/***************************
 乱数発生(1 から a までの整数)
***************************/
int rand_n(int a)
{
        int r;
        double rr;
        rr=(double)(rand() % a);
        r=(int)rr + 1;
        return r;
}


void read_data()
{
        fscanf(fp1,"%d",&n);
        fscanf(fp1,"%d",&n_row);
        fscanf(fp1,"%d",&n_column);

        for(kk=1;kk<=N2;kk++){
                for(i=1;i<=n;i++){
                        for(j=1;j<=n;j++){
                                fscanf(fp1,"%d",&c[kk][i][j]);
                        }
                }
                for(i=1;i<=n;i++){
                        for(j=1;j<=n;j++){
                                fscanf(fp1,"%d",&dist[kk][i][j]);
```

```
                        }
                }
        }

        for(i=1;i<=n;i++){fscanf(fp1,"%d",&site[i]);}
        fscanf(fp1,"%lf",&cpu_max);
        fscanf(fp1,"%lf",&cnt_max);
        fscanf(fp1,"%d",&rand_iter);
//      printf("%.0lf¥n",cpu_max);

        fscanf(fp1,"%lf",&t_ini);
        fscanf(fp1,"%f",&tb);

        fscanf(fp1,"%d",&pop_size);
        fscanf(fp1,"%d",&px);
        fscanf(fp1,"%d",&pm);
        fscanf(fp1,"%d",&n_iter);
//      fscanf(fp1,"%d",&c_times);
//      printf("%d ¥n",c_times);

        fobj=0.;
        for(kk=1;kk<=N2;kk++){
                for(j=1;j<=n;j++){
                        for(k=1;k<=n;k++){
                                fobj=fobj+(float)(c[kk][site[j]][site[k]]*dist[kk][j][k]);
                        }
                }
        }
        fopt=fobj;
//      printf("世代=%d, ",cnt);
//      lap=clock();
//      cpu_time=(double)(lap-start)/CLOCKS_PER_SEC;
//      printf("GA, %.3f, ",cpu_time);
//      printf("%.0f, ",fopt);
//      printf("%d ¥n",cnt);
}


void init_chrm_new()
{
        for(i=0;i<=pop_size-1;i++){
                for(j=0;j<=N1-1;j++){
                        chrm_new[i][j]=0;}
        }
}
```

```
void fit_calc()
/*適応度，選択率計算*/
{
/*                  printf("==chrm 順序表現→site 配置へ変換==¥n");   */
                    for(i=1;i<=pop_size;i++){
                              for(j=1;j<=n;j++){tmp_ga[j]=j;}
                              for(j=1;j<=n-1;j++){
                                        site[j]=tmp_ga[chrm[i][j]];
                                        for(k=chrm[i][j];k<=n;k++){
                                                  tmp_ga[k]=tmp_ga[k+1];}
                              }
                              site[n]=tmp_ga[1];
/*                            for(j=1;j<=n;j++){
                                        printf("%3d",site[j]);}
                              printf("¥n");   */
                              fobj=0.;
                              for(kk=1;kk<=N2;kk++){
                                        for(j=1;j<=n;j++){
                                                  for(k=1;k<=n;k++){


          fobj=fobj+(float)(c[kk][site[j]][site[k]]*dist[kk][j][k]);
                                                  }
                                        }
                              }
                              f_obj[i]=(float)fobj;
                    }

                    fit_total=0;
                    for(i=1;i<=pop_size;i++){
                              fitness[i]=(float)(1./(1.+f_obj[i]));
                              fit_total=fit_total+fitness[i];

                    }
                    p_select[0]=0.;
                    for(i=1;i<=pop_size;i++){
                              p_select[i]=p_select[i-1]+(double)(fitness[i]/fit_total);
                    }

}


void top_keep()
{
          fobj_min=f_obj[1];
```

```
        for(j=1;j<=n-1;j++){
                chrm_new[1][j]=chrm[1][j];}
        for(i=2;i<=pop_size;i++){
                if(f_obj[i]<fobj_min){
                        fobj_min=f_obj[i];
                        for(j=1;j<=n-1;j++){
                                chrm_new[1][j]=chrm[i][j];}
                }
        }
        cnt_new=1;
}


void    hrote(void)
{
        int i,j,n_column2,tmp;
        n_column2=n_column/2;
        for(i=1;i<=n_row;i++){for(j=1;j<=n_column2;j++){
                tmp=p[i][n_column-j+1];p[i][n_column-j+1]=p[i][j];p[i][j]=tmp;}}
}

void    vrote(void)
{
        int i,j,n_row2,tmp;
        n_row2=n_row/2;
        for(j=1;j<=n_column;j++){for(i=1;i<=n_row2;i++){
                tmp=p[n_row-i+1][j];p[n_row-i+1][j]=p[i][j];p[i][j]=tmp;}}
}

void    xrote(void)
{
        int i,j,tmp;
        for(i=1;i<=n_row-1;i++){for(j=i+1;j<=n_row;j++){
                tmp=p[i][j];p[i][j]=p[j][i];p[j][i]=tmp;}}
}


void dsp_opt()
{
        double ran;

        if(f_obj[1]<fopt){
                count=f_obj[1];
                fopt=f_obj[1];
                xx=1;
//              printf("世代=%d, ",cnt);
```

```
                lap=clock();
                cpu_time=(double)(lap-start)/CLOCKS_PER_SEC;
                printf("sga, %.3f, ",cpu_time);
                printf("%.0f, ",f_obj[1]);
//              for(j=1;j<=n-1;j++){printf("%d,",chrm[1][j]);}
                printf("%d ¥n",cnt);
//              printf("¥n");

                for(j=1;j<=n;j++){tmp_ga[j]=j;}
                for(j=1;j<=n-1;j++){
                        site[j]=tmp_ga[chrm[1][j]];
                        for(k=chrm[1][j];k<=n;k++){
                                tmp_ga[k]=tmp_ga[k+1];}
                }
                site[n]=tmp_ga[1];

                k=1;
                for(i=1;i<=n_row;i++){
//                      printf("                                                ,,,");
                        for(j=1;j<=n_column;j++){
//                              printf("%4d,",site[k]);
                                k++;
                        }
//                      printf("¥n");
                }
        }

        if(fopt==count){xx++;}

//      if(xx==100){
//      if(xx==50){
        if(xx==c_times){
//              kkk=rand_n(7);
                for(m=1;m<=pop_size;m++){
                        ran=(double)rand() / RAND_MAX * 100.;
                        if(ran < 50){
                                //                      for(m=1;m<=2;m++){
                                for(j=1;j<=n;j++){tmp_ga[j]=j;}
                                for(j=1;j<=n-1;j++){
                                        site[j]=tmp_ga[chrm[m][j]];
                                        for(k=chrm[m][j];k<=n;k++){
                                                tmp_ga[k]=tmp_ga[k+1];}
                                }
                                site[n]=tmp_ga[1];

                                k=1;
```

```
for(i=1;i<=n_row;i++){
        for(j=1;j<=n_column;j++){
                p[i][j]=site[k];
                k++;
        }
}

if(n_row!=n_column){kk=1;}
else{kk=2;}

if(kk==1){
        k=rand_n(3);
        if(k==1){hrote();   /* printf("左右反転¥n"); */}
        if(k==2){vrote();   /* printf("上下反転¥n"); */}
        if(k==3){hrote();   vrote();   /*  printf("180°回転
¥n"); */}
}

if(kk==2){

        k=rand_n(2);
        if(k==1){hrote();   vrote();   xrote();   vrote();}
        if(k==2){hrote();   vrote();   xrote();   hrote();}

        /*
        k=3;
        if(k==1){hrote(); } // printf("左右反転¥n");
        if(k==2){vrote(); } // printf("上下反転¥n");
        if(k==3){hrote();   vrote();   xrote();   vrote();} //
printf("90°回転¥n");//

        if(k==4){hrote();   vrote(); } //  printf("180°回転
¥n");

        if(k==5){hrote();   vrote();   xrote();   hrote();}

        // printf("270°回転¥n"); //

        if(k==6){xrote();}  // printf("左上右下対角軸回
転¥n");

        if(k==7){hrote();   vrote();   xrote(); } //   printf("
右上左下対角軸回転¥n");

        */

}


k=1;
for(i=1;i<=n_row;i++){
        for(j=1;j<=n_column;j++){
```

```
                                                site[k]=p[i][j];
                                                k++;
                                        }
                                }

                                for(j=1;j<=n;j++){tmp_ga[j]=j;}
                                for(j=1;j<=n-1;j++){
                                        for(k=1;k<=n;k++){
                                                if(site[j]==tmp_ga[k]){
                                                        chrm[m][j]=k;
                                                        for(l=k;l<=n;l++){

        tmp_ga[l]=tmp_ga[l+1];
                                                        }
                                                }
                                        }
                                }
                        }
//                      xx++;

        }



//      if(xx==500){
        if(xx==c_times+100){
//                      row_l=rand_n(n_row);
//                      for(i=1;i<=pop_size;i++){
//                              virus();
//                      }
                        xx=1;
        }


}


void select()
/*選択 (2007/07/04 修正中)*/
{
        int h,i,sp[2],s_cont;
        double pb;

        for(h=0;h<=1;h++){
                pb=(double)rand() / RAND_MAX;   /* 0～1の乱数発生 */
```

```
                // printf("pb=%.3f¥n",pb);
                i=0; s_cont=1;
                do{
                        i++; sp[h]=i;
                        //printf(" sp[%d]=%d p_select[%d]=%.3f¥n",h,sp[h],i,p_select[i]);
                        if(i>pop_size){sp[h]=pop_size; s_cont=0;}
                        if(pb>p_select[i]){s_cont=0;}
                }while(s_cont==1);

                //printf("sp[%d]=%d¥n",h,sp[h]);
        }
        p1=sp[0]; p2=sp[1];
}


void crossover()
{
        double ran;

        int i,i1,i2,j,k,w_cnt,tmp;
        int ga_pr[3][N1],ga_ch[3][N1];
        int w_ch[3][N1]={0};

/*      printf(" p1=%d p2=%d¥n",p1,p2);   */
        for(i=1;i<=n-1;i++)
        {
                chrm_p[1][i]=chrm[(int)p1][i];
                chrm_p[2][i]=chrm[(int)p2][i];
        }
/*
                printf("p1: ");
        for(i=1;i<=n-1;i++){
                printf(" %d",chrm_p[1][i]);}
        printf("¥n");
        printf("p2: ");
        for(i=1;i<=n-1;i++){
                printf(" %d",chrm_p[2][i]);}
        printf("¥n");
*/
        ran=(double)rand() / RAND_MAX * 100.;
        if(ran < px){

                i1=rand_n(n-2);
                do{
                        i2=rand_n(n-1);
                }while(i1==i2);
```

```
if(i1>i2){tmp=i1; i1=i2; i2=tmp;}


for(i=1;i<=2;i++){
        for(j=1;j<=n;j++){tmp_ga[j]=j;}
        for(j=1;j<=n-1;j++){
                ga_pr[i][j]=tmp_ga[chrm_p[i][j]];
                for(k=chrm_p[i][j];k<=n;k++){
                        tmp_ga[k]=tmp_ga[k+1];}
        }
        ga_pr[i][n]=tmp_ga[1];
}


for(i=1;i<=n;i++){
        if((i>=i1)&&(i<=i2)){
                ga_ch[1][i]=ga_pr[2][i];
                ga_ch[2][i]=ga_pr[1][i];
        }
        else{
                ga_ch[1][i]=ga_pr[1][i];
                ga_ch[2][i]=ga_pr[2][i];
        }
}


for(k=1;k<=2;k++){
        w_cnt=0;
        for(i=1;i<=n;i++){
                if( (i<i1)||(i>i2) ){
                        j=i1;
                        do{
                                if(ga_ch[k][i] == ga_ch[k][j]){
                                        w_cnt++;
                                        w_ch[k][w_cnt]=i;
                                }
                                j++;
                        }while(j<=i2);
                }
        }
}


for(i=1;i<=w_cnt;i++){
        tmp=ga_ch[1][w_ch[1][i]];
```

```
                ga_ch[1][w_ch[1][i]]=ga_ch[2][w_ch[2][i]];
                ga_ch[2][w_ch[2][i]]=tmp;
        }


        for(i=1;i<=2;i++){
                for(j=1;j<=n;j++){tmp_ga[j]=j;}
                for(j=1;j<=n-1;j++){
                        for(k=1;k<=n;k++){
                                if(ga_ch[i][j]==tmp_ga[k]){
                                        chrm_c[i][j]=k;
                                        for(l=k;l<=n;l++){
                                                tmp_ga[l]=tmp_ga[l+1];
                                        }
                                }
                        }
                }
        }

/*
        x_point=rand_n(n-2);

        for(i=1;i<=n-1;i++){
                if(i<=x_point){
                        chrm_c[1][i]=chrm_p[1][i];
                        chrm_c[2][i]=chrm_p[2][i];}
                else{
                        chrm_c[1][i]=chrm_p[2][i];
                        chrm_c[2][i]=chrm_p[1][i];}
        }
*/

    }
    else
    {
        for(i=1;i<=n-1;i++){
                chrm_c[1][i]=chrm_p[1][i];
                chrm_c[2][i]=chrm_p[2][i];}
    }

/*  printf("c1: ");
    for(i=1;i<=n-1;i++){
            printf(" %d",chrm_c[1][i]);}
    printf("¥n");
    printf("c2: ");
    for(i=1;i<=n-1;i++){
```

```
                printf(" %d",chrm_c[2][i]);}
        printf("¥n");
*/
}


void existence(void)
{
/* chrm_new になければ chrm_c を追加する */
                        for(k=1;k<=2;k++){
                                flag_all=0;
                                if(cnt_new>=1){
                                        for(i=1;i<=cnt_new;i++){
                                                flag=0;
                                                for(j=1;j<=n-1;j++){
                                                        c1=chrm_new[i][j];
                                                        c2=chrm_c[k][j];
                                                        if(c1==c2)flag=flag+0;
                                                        else flag=flag+1;
                                                }
                                                if(flag==0) flag=0;
                                                else            flag=1;

                                                flag_all=flag_all+flag;
                                        }
                                        if(flag_all==cnt_new){
                                                for(j=1;j<=n-1;j++){

        chrm_new[cnt_new+1][j]=chrm_c[k][j];}
                                                cnt_new=cnt_new+1;
                                        }
                                }
                                else{
                                        for(j=1;j<=n-1;j++){
                                                chrm_new[1][j]=chrm_c[k][j];}
                                        cnt_new=cnt_new+1;
                                }
                        }
}


void mutation(void)
{
        int k;
        int i,j,i1,i2,kk,kkk,a,b,j3,tmp;
        for(i=1;i<=2;i++){
```

```
                for(j=1;j<=n;j++){tmp_ga[j]=j;}
                for(j=1;j<=n-1;j++){
                        site[j]=tmp_ga[chrm_c[i][j]];
                        for(k=chrm_c[i][j];k<=n;k++){
                                tmp_ga[k]=tmp_ga[k+1];}
                }
                site[n]=tmp_ga[1];


                i1=rand_n(n-2);
                do{i2=rand_n(n-1);}while(i1==i2);
                if(i1>i2){tmp=i1; i1=i2; i2=tmp;}
                j3=(int)(i2-i1)/2+0.5; //printf("%d,%d¥n",i2-i1,j3);
                for(j=1;j<=j3;j++){
                        a=j+i1-1; b=i2-j+1;
                        tmp=site[a]; site[a]=site[b]; site[b]=tmp;
                }


                for(j=1;j<=n;j++){tmp_ga[j]=j;}
                for(j=1;j<=n-1;j++){
                        for(k=1;k<=n;k++){
                                if(site[j]==tmp_ga[k]){
                                        chrm_c[i][j]=k;
                                        for(l=k;l<=n;l++){
                                                tmp_ga[l]=tmp_ga[l+1];
                                        }
                                }
                        }
                }

                /*
                k=rand_n(n-1);
                chrm_c[i][k]=rand_n(n-k+1);
                */
        }
}


void ga(void)
{
        double ran;

        init_chrm_new();
        pm_last=(int)(100-pm)/2;      //printf("pm_last=%d ",pm_last);
```

```
//        alp=(double)(pm_last-pm)/pm; //printf("alp=%.3f¥n",alp);

/*        printf("==初期集団作成 ==¥n"); */
          for(i=1;i<=pop_size;i++){
                    for(j=1;j<=n-1;j++){
                              chrm[i][j]=rand_n(n-j+1);
                    /*        printf("%3d",chrm[i][j]); */
                    }
          /*        printf("¥n"); */
          }

//        while(cnt < n_iter)
          while(cpu_time < cpu_max) /* 計算時間上限 cpu_max を超えない間は繰り返し */
          {
                    fit_calc();
                    top_keep();
                    dsp_opt();

                    while(cnt_new < pop_size){
                    /*        printf("==選択==¥n");   */
                              select();
                              while(p1==p2){
                                        select();
                              }
/*                            printf("==交叉==¥n");   */
                              crossover();
                              existence();

/*                            printf("==突然変異==¥n");   */

                              ran=(double)rand() / RAND_MAX * 100.;
                              if(ran < (double)pm ){mutation(); existence();}
                    }
                    /*        printf("==chrm_new を chrm に転写==¥n");   */
                    for(i=1;i<=pop_size;i++){
                              for(j=1;j<=n-1;j++){
                                        chrm[i][j]=chrm_new[i][j];
                              }
                    }
                    cnt=cnt+1; cnt_new=0;       /*        exit(1); */
          }
//        fit_calc();

/* chrm→site 変換 */

          /*
```

```
        for(i=1;i<=pop_size;i++){
                for(j=1;j<=n;j++){tmp_ga[j]=j;}
                for(j=1;j<=n-1;j++){
                        site[j]=tmp_ga[chrm[i][j]];
                        for(k=chrm[i][j];k<=n;k++){
                                tmp_ga[k]=tmp_ga[k+1];}
                }
                site[n]=tmp_ga[1];
                for(j=1;j<=n;j++){
                        chrm[i][j]=site[j];}
                lap=clock(); cpu_time=(double)(lap-start)/CLOCKS_PER_SEC;
        } //   end of i-loop
        */

//      for(i=1;i<=pop_size;i++){
//              for(j=1;j<=n;j++){printf("%d,",chrm[i][j]);}printf("¥n");
//      }
/*** 終了処理 ***/
//      printf("GA 終了¥n");
        finish=clock();
//      printf("computation time = %10.3lf(sec.)¥n", (double)(finish-start)/CLOCKS_PER_SEC);
//      printf("%.3lf, ¥n", (double)(finish-start)/CLOCKS_PER_SEC);
        printf(",%.3lf, ", (double)(finish-start)/CLOCKS_PER_SEC);
        printf("%.0f, ",f_obj[1]);
        printf("%d ¥n",cnt);
}
void main(int argc, char *argv[])
{
        /* ファイルエラー処理 */
        if(argc!=2){
                printf("引数の数が違います. ex) C:¥>program.exe datafile.txt ¥n");
                exit(1);
        }
        if((fp1=fopen(argv[1],"r"))==NULL){
                printf("入力ファイルがオープンできません.¥n");
                exit(1);
        }
        srand((unsigned)time(NULL));            /* 乱数の初期化 */
        start=clock();                  /* time count start */
        read_data();
        fclose(fp1);
        ga();
        printf("引数の数が違います. ex) C:¥>program.exe datafile.txt ¥n");
} /* end of main */
```

# 付録2 プログラムソース2 HGA

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <conio.h>
#include <time.h>
#include <math.h>
#include <limits.h>
#define   MMAX   21
#define   NMAX   21
#define   LARGE   100000000
#define   LSTIME  25
#define   PB2OPT  70
#define N1 401
#define N2 2
#define tim 180

int rand_n(int a);
void read_data(void);
void ga_param(void);
void init_chrm_new(void);
void fit_calc(void);
void top_keep(void);
void dsp_opt(void);
void select(void);
void crossover(void);
void existence(void);
void mutation(void);
void ga(void);

void      perm4(void);
void      perm3(void);
void      find_i2j2(int prr);
void      f_obj_p(void);


          FILE *fp1;
          int tb_leng;
          double cnt_max;
          int n_2opt;
          int h,i,ii,j,l,m,n,k,kk,kkk,s;

          int a,b,x,y,aaa,iii,lll,i1,i2,j2,i3,j3,xx,yy;

          int                  fg[MMAX*NMAX][MMAX*NMAX]={0};
```

```
        int             fg2[MMAX*NMAX][MMAX*NMAX]={0};
        int             pr[MMAX*NMAX][MMAX*NMAX]={0};
        int             q[25][6]={0};
        int             q3[7][5]={0};
        int
p[MMAX][NMAX],p_min[MMAX][NMAX],fopt_p[MMAX][NMAX],p_crnt[MMAX][NMAX];


        int wp[N1],tmp;
        int c[N2+1][MMAX*NMAX][MMAX*NMAX];
//      int d[N2+1][N1][N1];
        int d[3][MMAX][NMAX][MMAX][NMAX];
        int dist[N2+1][MMAX*NMAX][MMAX*NMAX];
        int site[N1];
        float fobj=0, fobj_crnt=0,fobj_min, fopt,f,fmin,fc;
        int cont=1,cnt_term=0,cnt_a;
        int r1,r2, flg, flg_tl;
        double cnt_all=0.;
        double   cpu_time;
        double   cpu_max;
        double   t, bt,t_ini;
        int             n_ep, cnt_ep;
    int     almax, tleng;
        double no_update=0;
//      double  tim;
        clock_t  start,finish,lap;
        double   the_time_1,the_time_2;

/** GA 用变数 **/
        int cnt=0, cnt_new=0, rn;
        int chrm[60][MMAX*NMAX], chrm_new[60][MMAX*NMAX];
        int p1,p2;
        int w_cnt,w_ch[3][MMAX*NMAX]={0},ww_ch[3][MMAX*NMAX]={0};
        double          fitness[MMAX*NMAX],          fit_total,          f_obj[MMAX*NMAX],
p_select[MMAX*NMAX];
        int pop_size, px, pm, n_iter;
        int tmp_ga[MMAX*NMAX], x_point, c1, c2, flag, flag_all;
//      int chrm_p[3][N1];
        int chrm_p[3][MMAX*NMAX][MMAX*NMAX];
        int chrm_c[3][MMAX*NMAX];
        int m,n;
        int pm_last;
        double alp;

/**TS**/
double f1,f2,tim1,tim2;
```

```
void dsp_tl(int pp[MMAX][NMAX]);
int       chk_tl(int pp[MMAX][NMAX]);
void      update_tl(int pp[MMAX][NMAX]);
int                 tl[102][MMAX][NMAX]={0};

/**SA**/
void      kinbo_adj4(int i1, int j1);
void      kinbo_select(void);
void      kinbo_2opt(void);
void      sa(void);
void      p_copy(int p1[MMAX][NMAX],int p2[MMAX][NMAX]);
double    f_obj_sa(int pp[MMAX][NMAX]);
int      p_opt[MMAX][NMAX];
void      p_dsp(int pp[MMAX][NMAX]);


double   rand_big;

/*****************************
 乱数発生(1 から a までの整数)
*****************************/
int rand_n(int a)
{
          int r;
          double rr;
          rr=(double)(rand() % a);
          r=(int)rr + 1;
          return r;
}
/*****************************
 乱数発生(1 から a までの整数)
*****************************/
int rand_nn(int a)
{
          int r;
          double rr;
          rr=(double)(rand() % a);
          r=(int)rr;
          return r;
}
/*****************************
 データファイル読み込み
*****************************/
void read_data()
{
          fscanf(fp1,"%d",&s);
```

```
fscanf(fp1,"%d",&m);
fscanf(fp1,"%d",&n);

for(kk=1;kk<=N2;kk++){
        for(i=1;i<=s;i++){
                for(j=1;j<=s;j++){
                        fscanf(fp1,"%d",&c[kk][i][j]);
                }
        }
        for(i=1;i<=m;i++){
                for(j=1;j<=n;j++){
                        for(k=1;k<=m;k++){
                                for(l=1;l<=n;l++){
                                        fscanf(fp1,"%d",&d[kk][i][j][k][l]);
                                }
                        }
                }
        }

}

for(i=1;i<=m;i++){
        for(j=1;j<=n;j++){
                fscanf(fp1,"%d",&p[i][j]);
        }
}
fscanf(fp1,"%lf",&cpu_max);
almax=s*s;
fscanf(fp1,"%d",&tleng);
fscanf(fp1,"%d",&n_2opt);
//      printf("%.0lf¥n",cpu_max);

fscanf(fp1,"%lg",&t);
fscanf(fp1,"%lg",&bt);

fscanf(fp1,"%d",&pop_size);
fscanf(fp1,"%d",&px);
fscanf(fp1,"%d",&pm);
fscanf(fp1,"%d",&n_iter);

n_ep=s*20;
t_ini=t;


fobj=0.;
for(h=1;h<=2;h++){
```

```
                for(i=1;i<=m;i++){
                        for(j=1;j<=n;j++){
                                for(k=1;k<=m;k++){
                                        for(l=1;l<=n;l++){

        fobj=fobj+c[h][p[i][j]][p[k][l]]*d[h][i][j][k][l];
                                        }
                                }
                        }
                }
        }
        fopt=fobj;

        fc=fobj;

        for(i=1;i<=m;i++){
                for(j=1;j<=n;j++){
                        p_crnt[i][j]=p[i][j];
                }
        }

        for(i=1;i<=m;i++){
                for(j=1;j<=n;j++){
                        p_opt[i][j]=p[i][j];
                }
        }


//      printf("世代=%d, ",cnt);
//      lap=clock();
//      cpu_time=(double)(lap-start)/CLOCKS_PER_SEC;
//      printf("GA, %.3f, ",cpu_time);
//      printf("%.0f, ",fopt);
//      printf("%d ¥n",cnt);
}


/*************************************
  Metropolis 受理確率関数
*************************************/
int     ap(double f1,double f2)
{
        double r;
        double delta,x;
        delta=f1-f2;
```

```
//          printf("f1=%.3f f2=%.3f f1-f2=%.3f    ",f1,f2,f1-f2);

            if(delta<=0.){r=100.;}
            else{
//                      printf("t_ini=%.3f ",t_ini); printf("x=%.8f ",delta/t_ini);
            x=delta/t_ini;
//          printf("x=%.8f ",x);
            r=(exp(-x))*100.;   }

//          printf("r=%.3f ¥n",r);

            return (int)r;

}
/*****************************
 horizontal 回転
*****************************/
void        hrote(void)
{
            int i,j,n2,tmp;
            n2=n/2;

            for(i=1;i<=m;i++){
                        for(j=1;j<=n2;j++){
                                    tmp=p[i][n-j+1];p[i][n-j+1]=p[i][j];p[i][j]=tmp;
                        }
            }
}
/*****************************
 vertical 回転
*****************************/
void        vrote(void)
{
            int i,j,m2,tmp;
            m2=m/2;
            for(j=1;j<=n;j++){for(i=1;i<=m2;i++){
                        tmp=p[m-i+1][j];p[m-i+1][j]=p[i][j];p[i][j]=tmp;}}
}
/*****************************
 cross 回転
*****************************/
void        xrote(void)
{
            int i,j,tmp;
            for(i=1;i<=m-1;i++){for(j=i+1;j<=m;j++){
                        tmp=p[i][j];p[i][j]=p[j][i];p[j][i]=tmp;}}
```

```
}


void    calc_fg(void)
{
        int h,i,j,k,max_value,max_no;

        for(h=1;h<=2;h++){
                for(i=1;i<=m*n;i++){
                        for(j=1;j<=m*n;j++){
                                fg[i][j]=fg[i][j]+c[h][i][j];
                        }
                }
        }

        for(i=1;i<=m*n;i++){
                for(j=1;j<=m*n;j++){
                        fg2[i][j]=fg[i][j]+fg[j][i];
                }
        }

        for(i=1;i<=m*n;i++){
                for(k=1;k<=4;k++){
                        max_value=-1;
                        for(j=1;j<=m*n;j++){
                                if(fg2[i][j]>max_value){
                                        if(j != i){
                                                max_value=fg2[i][j];
                                                max_no=j;
                                        }
                                }
                        }
                        pr[i][k]=max_no;
                        fg2[i][max_no]=-1;
                }
        }
/*
        for(i=1;i<=s;i++){
                printf("%3d:",i);
                for(j=1;j<=4;j++){
                        printf(" %3d",pr[i][j]);
                }
                printf("¥n");
        }
*/
}
```

```
void	p_copy(int p1[MMAX][NMAX],int p2[MMAX][NMAX])
/* p1[][] から p2[][] に転写 */
{
	int i,j;
	for(i=1;i<=m;i++){
		for(j=1;j<=n;j++){
			p2[i][j]=p1[i][j];
		}
	}
}


/*************************************
  4 の順列を q[][]に入力する
*************************************/
void	perm4(void)
{
	int i,j,k,jj,kk,h=0, n4=4,w;
	int t[6]={0};

	for(i=1;i<=n4;i++){ t[i]=i;}
	do{
		h=h+1;
		for(i=1;i<=n4;i++){q[h][i]=t[i];}

		j=n4-1;
		while(t[j]>t[j+1]){j=j-1;}

		k=j+1;
		while(t[k]>t[j]){k=k+1;}

		w=t[j];
		t[j]=t[k-1];
		t[k-1]=w;

		jj=j+1;
		kk=n4;

		while(jj<kk){
			w=t[jj];
			t[jj]=t[kk];
			t[kk]=w;
			jj=jj+1;
			kk=kk-1;
```

```
                }
        }
        while(j!=0);

}
/*************************************
  3 の順列を q3[][]に入力する
*************************************/
void    perm3(void)
{
        int i,j,k,jj,kk,h=0, nn=3,w;
        int t[5]={0};

        for(i=1;i<=nn;i++){ t[i]=i;}
        do{
                h=h+1;
                for(i=1;i<=nn;i++){q3[h][i]=t[i];}

                j=nn-1;
                while(t[j]>t[j+1]){j=j-1;}

                k=j+1;
                while(t[k]>t[j]){k=k+1;}

                w=t[j];
                t[j]=t[k-1];
                t[k-1]=w;

                jj=j+1;
                kk=nn;

                while(jj<kk){
                        w=t[jj];
                        t[jj]=t[kk];
                        t[kk]=w;
                        jj=jj+1;
                        kk=kk-1;
                }
        }
        while(j!=0);
/*
        printf("3 の全順列¥n");
        for(i=1;i<=6;i++){
                for(j=1;j<=3;j++){printf("%3d",q3[i][j]);}
                printf("¥n");
        }
```

```
*/
}
/**************************************
  p[][]から prr を見つけ、その i,j を返す
**************************************/
void find_i2j2(int prr)
{
  int i,j;
  for(i=1;i<=m;i++){
    for(j=1;j<=n;j++){
      if(p[i][j]==prr){i2=i; j2=j; goto L_END;}
    }
  }
  L_END:
  //printf("i2=%d j2=%d ¥n",i2,j2);
  return;
}


void init_chrm_new()
{
        for(i=1;i<=pop_size;i++){

//              srand((unsigned)time(NULL));
                l=1;
                for(j=0; j<s; j++){
                        wp[j]=j;
                }
                for(j=0; j<s; j++)                      ///ﾉ螃ﾉ n ｸ□貊摠□
                {
                        k=rand() % s;                           //ｱ萩ｿ k ﾋ萼ﾉ rand()ｺｯﾊ□ﾉ螃ﾉ」
ｬｵｫﾋ・ｻﾊﾇﾊ□ﾗ鰆ｪﾋﾘｵﾄｵﾘﾖｷ

                        kk=wp[k];                               //ｱ萩ｿ n ｲﾅﾊﾇﾊ□ﾗ鰆ｪﾋﾘｵﾄ
ﾖｵ
                        while(kk==-1)                   //ﾈ郢瓶ﾎｪ-1」ｬﾋｵﾃ□篋□ｪﾋﾘｱｻﾊｹﾓﾃｹ
□」ｬﾒｪﾖﾘﾐﾂﾑｰﾕﾒ」ｬﾖｱﾖﾁﾕﾒｵｽﾎｴﾓﾃｹ□ｵﾄﾎｪﾖｹ
                        {
                                k=rand() % s;
                                kk=wp[k];
                        }
                        chrm[i][l]=kk+1;
//                      printf(" %3d",chrm[i][l]);
                        l++;
                        wp[k]=-1;                               //ﾁ「ｼｴｽｫｸﾃﾔｪﾋﾘﾖｵﾖﾃﾎｪ-
1」ｬｱ桃ｾﾒﾑｱｻﾓﾃｹ□
```

```
                }
//              printf("¥n");
                lap=clock();
                cpu_time=(double)(lap-start)/CLOCKS_PER_SEC;
        }
}


void fit_calc()
{
                for(kk=1;kk<=pop_size;kk++){
                        k=1;
                        for(i=1;i<=m;i++){
                                for(j=1;j<=n;j++){
                                        p[i][j]=chrm[kk][k];
                                        k++;
                                }
                        }

                        fobj=0.;
                        for(h=1;h<=2;h++){
                                for(i=1;i<=m;i++){
                                        for(j=1;j<=n;j++){
                                                for(k=1;k<=m;k++){
                                                        for(l=1;l<=n;l++){

        fobj=fobj+c[h][p[i][j]][p[k][l]]*d[h][i][j][k][l];
                                                        }
                                                }
                                        }
                                }
                        }

                        f_obj[kk]=(float)fobj;

//                      printf("kk=%d %1f",kk,f_obj[kk]);
//                      printf("¥n");
                }


                fit_total=0;
                for(i=1;i<=pop_size;i++){
                        fitness[i]=(float)(1./(1.+f_obj[i]));
                        fit_total=fit_total+fitness[i];

                }
```

```
		p_select[0]=0.;
		for(i=1;i<=pop_size;i++){
			p_select[i]=p_select[i-1]+(double)(fitness[i]/fit_total);
		}

}


void f_obj_p()
{
	fobj=0.;

	for(h=1;h<=2;h++){
		for(ii=1;ii<=m;ii++){
			for(j=1;j<=n;j++){
				for(k=1;k<=m;k++){
					for(l=1;l<=n;l++){

	fobj=fobj+c[h][p[ii][j]][p[k][l]]*d[h][ii][j][k][l];
					}
				}
			}
		}
	}
//	printf("%.3f ¥n",fobj);

}


void top_keep()
{
	fobj_min=f_obj[1];
	for(j=1;j<=s;j++){
		chrm_new[1][j]=chrm[1][j];
	}
	for(i=2;i<=pop_size;i++){
		if(f_obj[i]<fobj_min){
			fobj_min=f_obj[i];
			for(j=1;j<=s;j++){
				chrm_new[1][j]=chrm[i][j];}
		}
	}
	cnt_new=1;
}
```

```
void dsp_opt()
{
//          int j1;

          if(f_obj[1]<fopt){
                    fopt=f_obj[1];
                    lap=clock();
                    cpu_time=(double)(lap-start)/CLOCKS_PER_SEC;
                    printf("GA, %.3f, ",cpu_time);
                    printf("%.0f, ",f_obj[1]);
                    printf("%d ¥n",cnt);

                    the_time_1=cpu_time;

                    k=1;
                    for(i=1;i<=m;i++){
                              for(j=1;j<=n;j++){
//                                        p[i][j]=chrm_new[1][k];
                                        p_opt[i][j]=chrm_new[1][k];
                                        p_crnt[i][j]=chrm_new[1][k];
                                        k++;
                              }
                    }

                    fopt=f_obj[1];
                    fc=f_obj[1];


          }

}


void select()
{
          int h,i,sp[2],s_cont;
          double pb;

          for(h=0;h<=1;h++){
                    pb=(double)rand() / RAND_MAX;
                    i=0; s_cont=1;
                    do{
                              i++; sp[h]=i;
                              if(i>pop_size){sp[h]=pop_size; s_cont=0;}
                              if(pb>p_select[i]){s_cont=0;}
                    }while(s_cont==1);
```

```
        }
        p1=sp[0]; p2=sp[1];
}


void crossover()
{
        double ran;
        int i1,j1,i2,j2,p_t[N1][N1],p_tt[MMAX][NMAX];

//      int chrm_p[3][MMAX*NMAX][MMAX*NMAX];

        ran=(double)rand() / RAND_MAX * 100.;
        if(ran < px){
                k=1;
                for(i=1;i<=m;i++){
                        for(j=1;j<=n;j++){
                                chrm_p[1][i][j]=chrm[(int)p1][k];
//                              printf("-%3d ",chrm[(int)p1][k]);
                                k++;
                        }
//                      printf("¥n");
                }
//              printf("¥n");

                k=1;
                for(i=1;i<=m;i++){
                        for(j=1;j<=n;j++){
                                chrm_p[2][i][j]=chrm[(int)p2][k];
//                              printf("+%3d ",chrm[(int)p2][k]);
                                k++;
                        }
//                      printf("¥n");
                }
//                      printf("¥n");

/*
if(cpu_time>tim){
        i1=rand_n(m);j1=rand_n(n);

                                for(k=1;k<=2;k++){
                                        do{
                                                x=rand_n(m);y=rand_n(n);
                                        }while(x==i1 && y==j1);
                                        tmp=chrm_p[k][i1][j1];
                                        chrm_p[k][i1][j1]=chrm_p[k][x][y];
```

```
                                        chrm_p[k][x][y]=tmp;

                                        kk=0;
                                        for(i=1;i<=m;i++){
                                                for(j=1;j<=n;j++){
                                                        kk++;

        chrm_c[k][kk]=chrm_p[k][i][j];
                                                }
                                        }
                                }
}
*/


//else{
                i1=rand_n(m);j1=rand_n(n);

                if(i1==m || j1==n){
                        if(i1==m && j1!=n){a=0;b=rand_nn(n-j1+1);}
                        if(j1==n && i1!=m){b=0;a=rand_nn(m-i1+1);}
                }
                if(i1==m && j1==n){a=0;b=0;}
                if(i1!=m && j1!=n){a=rand_nn(m-i1+1);b=rand_nn(n-j1+1);}

//              printf("a=%d,b=%d¥n",a,b);

                if(a==0 && b==0){
//              printf("~~~~~~~~~~~~~~~~~~~~~~~~~¥n");
                        for(k=1;k<=2;k++){
                                do{
                                        x=rand_n(m);y=rand_n(n);
                                }while(x==i1 && y==j1);
                                tmp=chrm_p[k][i1][j1];
                                chrm_p[k][i1][j1]=chrm_p[k][x][y];
                                chrm_p[k][x][y]=tmp;
                        }
                }


                else{
                        x=1;y=1;
//                      printf("i1=%d,j1=%d¥n",i1,j1);
//                      printf("a+i1=%d,b+j1=%d¥n",a+i1,b+j1);

                        for(i=i1;i<=a+i1;i++){
```

```
                                for(j=j1;j<=b+j1;j++){
                                        p_t[x][y]=chrm_p[1][i][j];
//                                      printf("//%3d ",chrm_p[1][i][j]);
                                        x++;y++;
                                }
//                              printf("¥n");
                        }

                        i2=rand_n(m-a);j2=rand_n(n-b);
                        x=1;y=1;
                        for(i=i2;i<=a+i2;i++){
                                for(j=j2;j<=b+j2;j++){
                                        p_tt[x][y]=chrm_p[2][i][j];
                                        chrm_p[2][i][j]=p_t[x][y];
                                        x++;y++;
                                }
                        }
//              printf("***********~~~~~~~~~~~~~~********¥n");

                        x=1;y=1;
                        for(i=i1;i<=a+i1;i++){
                                for(j=j1;j<=b+j1;j++){
                                        chrm_p[1][i][j]=p_tt[x][y];
                                        x++;y++;
                                }
                        }

                        for(k=1;k<=2;k++){
                                if(k==1){xx=i1;yy=j1;}
                                if(k==2){xx=i2;yy=j2;}
                                w_cnt=0;
                                for(i=1;i<=m;i++){
                                        for(j=1;j<=n;j++){
                                                if(i<xx || i>a+xx || j<yy || j>b+yy){
                                                        for(x=xx;x<=a+xx;x++){

        for(y=yy;y<=b+yy;y++){

        if(chrm_p[k][i][j]==chrm_p[k][x][y]){

        w_cnt++;

        w_ch[k][w_cnt]=x;

        ww_ch[k][w_cnt]=y;
//
```

```
        goto L_END;
                                                                                }
                                                                            }
                                                                        }
                                                                    }
//                          L_END:
                                                    }
                                            }
//                                          printf("¥n");
                                    }
                                    for(i=1;i<=w_cnt;i++){
                                            tmp=chrm_p[1][w_ch[1][i]][ww_ch[1][i]];

        chrm_p[1][w_ch[1][i]][ww_ch[1][i]]=chrm_p[2][w_ch[2][i]][ww_ch[2][i]];
                                            chrm_p[2][w_ch[2][i]][ww_ch[2][i]]=tmp;
                                    }
                            }

                            for(k=1;k<=2;k++){
                                    kk=0;
                                    for(i=1;i<=m;i++){
                                            for(j=1;j<=n;j++){
                                                    kk++;
                                                    chrm_c[k][kk]=chrm_p[k][i][j];
//                                                  printf("*%3d ",chrm_c[k][kk]);
                                            }
                                    }
//                                  printf("¥n");
                            }

//}

        }

        else{
                    for(i=1;i<=s;i++){
                            chrm_c[1][i]=chrm[(int)p1][i];
                            chrm_c[2][i]=chrm[(int)p2][i];}
        }


}


void existence(void)
{
```

```
                    for(k=1;k<=2;k++){
                        flag_all=0;
                        if(cnt_new>=1){
                            for(i=1;i<=cnt_new;i++){
                                flag=0;
                                for(j=1;j<=s;j++){
                                    c1=chrm_new[i][j];
                                    c2=chrm_c[k][j];
                                    if(c1==c2)flag=flag+0;
                                    else flag=flag+1;
                                }
                                if(flag==0) flag=0;
                                else        flag=1;

                                flag_all=flag_all+flag;
                            }
                            if(flag_all==cnt_new){
                                for(j=1;j<=s;j++){

        chrm_new[cnt_new+1][j]=chrm_c[k][j];
                                }
                                cnt_new=cnt_new+1;
                            }
                        }
                        else{
                            for(j=1;j<=s;j++){
                                chrm_new[1][j]=chrm_c[k][j];
                            }
                            cnt_new=cnt_new+1;
                        }
                    }
}


void mutation(void)
{
        /*
        int j1,k1,k2,flgt,n6;
        int adj[5],prf[5],flag[5],flag_a[5]={0};
        int p_t[MMAX][NMAX],p_p[MMAX][NMAX];
        int adj_x[MMAX*NMAX],adj_y[MMAX*NMAX];
        */
        double ran;
        int wp[3]={0};
```

```
for(kkk=1;kkk<=1;kkk++){
        ran = (double)rand( ) / RAND_MAX;   // 0～1 乱数発生
        if(ran*100 < pm){

                j=1;
                for(i1=1;i1<=m;i1++){
                        for(i2=1;i2<=n;i2++){
                                p[i1][i2]=chrm_c[kkk][j];
//                              printf(" %3d",p[i1][i2]);
                                j++;
                        }
//                      printf("¥n");
                }
//              printf("¥n");

                if(m!=n){kk=1;}
                else{kk=2;}

                if(kk==1){
                        k=rand_n(3);
                        if(k==1){hrote();   }
                        if(k==2){vrote();   }
                        if(k==3){hrote();   vrote();   }
                }

                if(kk==2){
                        wp[kkk]=rand_n(7);
//                      k=3;
//                      if(kkk==1){k=3;}
//                      if(kkk==2){k=5;}

                        do{
                                if(k==1){hrote(); } // printf("左右反転¥n");
                                if(k==2){vrote(); } // printf("上下反転¥n");
                                if(k==3){hrote();vrote();xrote();vrote();}          //
printf("90°回転¥n");
                                if(k==4){hrote();   vrote(); } //  printf("180° 回 転
¥n");
                                if(k==5){hrote();   vrote();   xrote();   hrote();}
        // printf("270°回転¥n");
                                if(k==6){xrote();} // printf("左上右下対角軸回
転¥n");
                                if(k==7){hrote();   vrote();   xrote(); } //  printf("
右上左下対角軸回転¥n");
                        }while(wp[1]==wp[2]);
```

```
                        }


                        j=1;
                        for(i1=1;i1<=m;i1++){
                                for(i2=1;i2<=n;i2++){
                                        chrm_c[kkk][j]=p[i1][i2];
//                                      printf(" %3d",chrm_c[kkk][j]);
                                        j++;
                                }
//                              printf("¥n");
                        }
//                      printf("¥n");
                }

        }



}


/*************************************
  pp[][]が tl にあるかチェックする関数
        ある=1, ない=0 を返す
*************************************/
int     chk_tl(int pp[MMAX][NMAX])
{
        int i,j,k,r;
        if(tleng==0){r=0; goto END2;}
        for(k=1;k<=tleng;k++){
                r=1;
                for(i=1;i<=m;i++){
                        for(j=1;j<=n;j++){
                                if(pp[i][j]!=tl[k][i][j]){r=0; goto END1;}
                        }
                }
                END1:
                if(r==1){goto END2;}
        }
        END2:
        return r;
}
```

```
/************************************
  Tabu List 更新関数
************************************/
void    update_tl(int pp[MMAX][NMAX])
{
        int i,j,k;
        if(tleng!=0){
                for(k=tleng;k>=1;k--){
//                      printf("k=%2d    ",k);
                        for(i=1;i<=m;i++){
                                for(j=1;j<=n;j++){
                                        tl[k+1][i][j]=tl[k][i][j];
//                                      printf("%3d ",tl[k][i][j]);
                                }
                        }
//                      printf("¥n");
                }
//              printf("¥n");
                for(i=1;i<=m;i++){
                        for(j=1;j<=n;j++){
                                tl[1][i][j]=pp[i][j];
                        }
                }
        }

/*
        for(k=tleng;k>=1;k++){
                for(i=1;i<=m;i++){
                        for(j=1;j<=n;j++){
                                printf("%3d ",tl[k][i][j]);
                        }
                }
                printf("¥n");
                printf("¥n");
        }
*/

}



void    p_dsp(int pp[MMAX][NMAX])
{
        for(i=1;i<=m;i++){
                for(j=1;j<=n;j++){
                        printf("%4d",pp[i][j]);
```

```
                }
                printf("¥n");
        }
}


double  f_obj_sa(int pp[MMAX][NMAX])
{
        double z=0.;

        for(h=1;h<=2;h++){
                for(i=1;i<=m;i++){
                        for(j=1;j<=n;j++){
                                for(k=1;k<=m;k++){
                                        for(l=1;l<=n;l++){

        z=z+c[h][pp[i][j]][pp[k][l]]*d[h][i][j][k][l];
                                        }
                                }
                        }
                }
        }
        return z;
}



void    kinbo_select()
{
        int i1,j1;

  i1=(int)(rand_n(m));
  j1=(int)(rand_n(n));

  if( ((i1>=1)&&(i1<=m))&&((j1>=1)&&(j1<=n)) )
        {kinbo_adj4(i1,j1);}
  else {
                printf("ERROR: i1=%d, j1=%d¥n",i1,j1); //scanf("%d",&idmy);
                exit(1);
        }
}
```

```
void	kinbo_adj4(int i1,int j1)
{
	int i,j,i3,j3,k1,k2,flgt,tmp,n6,a;
	int adj[5],prf[5],flag[5],flag_a[5]={0};
	int p_t[MMAX][NMAX];
	int adj_x[MMAX*NMAX],adj_y[MMAX*NMAX];

	for(i=1;i<=20;i++){p[21][i]=0;}

	p_copy(p_crnt,p);

//	p_dsp(p);printf("¥n");

	k=1;

	if(p[i1-1][j1]!=0){adj[k]=p[i1-1][j1];adj_x[k]=i1-1;adj_y[k]=j1;k++;}
	if(p[i1][j1+1]!=0){adj[k]=p[i1][j1+1];adj_x[k]=i1;adj_y[k]=j1+1;k++;}
	if(p[i1+1][j1]!=0){adj[k]=p[i1+1][j1];adj_x[k]=i1+1;adj_y[k]=j1;k++;}
	if(p[i1][j1-1]!=0){adj[k]=p[i1][j1-1];adj_x[k]=i1;adj_y[k]=j1-1;k++;}

	n6=k-1;

	for(i=1;i<=n6;i++){prf[i]=pr[p[i1][j1]][i];}

	for(k1=1;k1<=n6;k1++){
		flag[k1]=0;
		for(k2=1;k2<=n6;k2++){
			if(adj[k2]==prf[k1]){
				flag[k1]=k2;
				flag_a[k2]=1;
				break;
			}
		}
	}
	flgt=0;
	for(i=1;i<=n6;i++){flgt=flgt+flag[i];}

	a=0;
	for(i=1;i<=n6;i++){a=a+i;}

	if(flgt!=a){
		for(k1=1;k1<=n6;k1++){
			if(flag[k1]==0){
				find_i2j2(prf[k1]);
				for(k2=1;k2<=n6;k2++){
					if(flag_a[k2]==0){
```

```
                                                i3=adj_x[k2];j3=adj_y[k2];
                                                tmp=p[i2][j2];          p[i2][j2]=p[i3][j3];
p[i3][j3]=tmp;
                                                flag[k1]=k2; flag_a[k2]=1;
                                                break;
                                        }
                                }
                        }
                }
        }

//      p_dsp(p);printf("¥n");

        p_copy(p,p_t);

        if(n6==2){
                p_copy(p_t,p);
                f=f_obj_sa(p);   cnt++; no_update++; cnt_ep++;
                fmin=f; p_copy(p,p_min);

                p_copy(p_t,p);
                tmp=p[adj_x[1]][adj_y[1]];
                p[adj_x[1]][adj_y[1]]=p[adj_x[2]][adj_y[2]];
                p[adj_x[2]][adj_y[2]]=tmp;
                f=f_obj_sa(p);   cnt++; no_update++; cnt_ep++;
                if(f<fmin){
                        fmin=f; p_copy(p,p_min);
                }

        }
        if(n6==3){
                for(i=1;i<=6;i++){
                        p_copy(p_t,p);
                        for(j=1;j<=3;j++){
                                if(q3[i][j]==1){p[adj_x[1]][adj_y[1]]=prf[j];}
                                else if(q3[i][j]==2){p[adj_x[2]][adj_y[2]]=prf[j];}
                                else if(q3[i][j]==3){p[adj_x[3]][adj_y[3]]=prf[j];}
                        }
//                      p_dsp(p);printf("¥n");

                        f=f_obj_sa(p);   cnt++; no_update++; cnt_ep++;
                        if(f<fmin){
                                fmin=f; p_copy(p,p_min);
                        }
                }
        }
```

```
        if(n6==4){
                for(i=1;i<=24;i++){
                        p_copy(p_t,p);
                        for(j=1;j<=4;j++){
                                if(q[i][j]==1){p[adj_x[1]][adj_y[1]]=prf[j];}
                                else if(q[i][j]==2){p[adj_x[2]][adj_y[2]]=prf[j];}
                                else if(q[i][j]==3){p[adj_x[3]][adj_y[3]]=prf[j];}
                                else if(q[i][j]==4){p[adj_x[4]][adj_y[4]]=prf[j];}
                        }
//                      p_dsp(p);printf("¥n");

                        f=f_obj_sa(p);   cnt++; no_update++; cnt_ep++;
                        if(f<fmin){
                                fmin=f; p_copy(p,p_min);
                        }
                }
        }
//              printf("****************¥n");
//              p_dsp(p);

//      printf("%.3f, %.3f ¥n",fmin,fopt);

}



/*************************************
  近傍を 2opt で作成する
*************************************/
void    kinbo_2opt()
{
        int i1,i2,j1,j2,k,tmp;
        for(k=1;k<=n_2opt;k++){
                i1=rand_n(m);       j1=rand_n(n);
                do{i2=rand_n(m);  j2=rand_n(n);}while((i1==i2)&&(j1==j2));
                p_copy(p_crnt,p);
                tmp=p[i1][j1];   p[i1][j1]=p[i2][j2];   p[i2][j2]=tmp;
                f=f_obj_sa(p);   cnt++; no_update++; cnt_ep++;
                if(f<fmin){fmin=f; p_copy(p,p_min);}
        }
}
void sa()
{

//      rand_rand=rand_n(100);
//      printf("%d ¥n",rand_rand);
```

```
		rand_big=(double)rand( )/RAND_MAX;
//		printf("%.3f ¥n",rand_big);

		if(fmin<=fc){
			fc=fmin; p_copy(p_min,p_crnt);
			if(fc < fopt){p_copy(p_crnt,p_opt); fopt=fc; no_update=0;
			lap=clock(); cpu_time=(double)(lap-start)/CLOCKS_PER_SEC;
//			sym_check();
			printf("SA, %.3f, %.1f, %.3f ¥n",cpu_time,fc,t_ini);
			}
		}

//		else if( rand_n(100) <ap(fmin,fc) ){
		else if( rand_big <ap(fmin,fc) ){
			fc=fmin;

			/*
			printf("**********************");
			printf("fmin=%.3f ¥n",fmin);
			printf("fc=%.3f ¥n",fc);
			*/

			p_copy(p_min,p_crnt);
			if(fc < fopt){p_copy(p_crnt,p_opt); fopt=fc; no_update=0;}
			if(cnt_ep>n_ep){t_ini=t_ini*bt;		cnt_ep=0;}
			else{cnt_ep++;}
		}
}



void ga(void)
{

		init_chrm_new();

//		pm_last=(int)(100-pm)/2;

		while(cpu_time < cpu_max){

//			if(cpu_time<tim){
			if(the_time_2-the_time_1<tim){
				fit_calc();
				top_keep();
```

```
                dsp_opt();

                while(cnt_new < pop_size){

                        select();
                        while(p1==p2){
                                select();
                        }

                        crossover();
                        existence();

//                      ran=(double)rand() / RAND_MAX * 100.;
//                      if(ran < (double)pm ){mutation(); existence();}
//                      mutation(); existence();
                        }


                        for(i=1;i<=pop_size;i++){
                                for(j=1;j<=s;j++){
                                        chrm[i][j]=chrm_new[i][j];
//                              printf("%3d ",chrm[i][j]);
                                }
//                      printf("¥n");
                        }


                        /*
                        for(k=1;k<=pop_size;k++){
                                kk=1;
                                for(i=1;i<=m;i++){
                                        for(j=1;j<=n;j++){
                                                tl[k][i][j]=chrm_new[k][kk];
                                                kk++;
                                        }
                                }
                        }
                        */


                        cnt=cnt+1; cnt_new=0;
                }


                else{
```

```
                        fmin=LARGE;
                        if(rand_n(100)<=PB2OPT){kinbo_2opt();}
                        else{kinbo_select();}
                        //                  kinbo_2opt();

                        lap=clock(); cpu_time=(double)(lap-start)/CLOCKS_PER_SEC;
                        sa();

//                                          printf("%.3f, %.3f, %.3f ¥n",fmin,fc,fopt);
                }


                lap=clock();
                cpu_time=(double)(lap-start)/CLOCKS_PER_SEC;
                the_time_2=cpu_time;

        }

        finish=clock();
        printf("FINISH, %.3lf, ", (double)(finish-start)/CLOCKS_PER_SEC);
        printf("%.0f ¥n",fopt);

        for(i=1;i<=m;i++){
                for(j=1;j<=n;j++){
                        printf("%4d",p_opt[i][j]);
                }
                printf("¥n");
        }

}

void main(int argc, char *argv[])
{
        /* ファイルエラー処理 */
        if(argc!=2){
                printf("引数の数が違います. ex) C:¥>program.exe datafile.txt ¥n");
                exit(1);
        }
        if((fp1=fopen(argv[1],"r"))==NULL){
                printf("入力ファイルがオープンできません.¥n");
                exit(1);
        }
        srand((unsigned)time(NULL));          /* 乱数の初期化 */
        start=clock();                  /* time count start */
        read_data();
        fclose(fp1);
```

```
        cnt_ep=0;
        cnt=1.;

        f2=fc;
        tim2=tim;

        perm4(); perm3();
        calc_fg();
/*
        f=f_obj_sa(p);
        p_copy(p,p_crnt);   fc=f;
        p_copy(p,p_opt);    fopt=f;
*/
        ga();
} /* end of main */
```