二次割当問題に対する可変深度近傍探索の適用

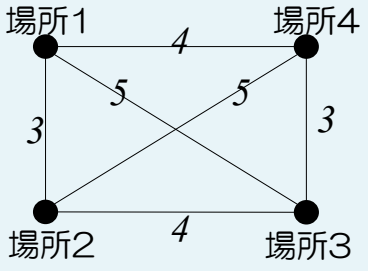
東京理科大学工学部経営工学科
沼田研究室
4410048　清水達朗

目次

1．はじめに

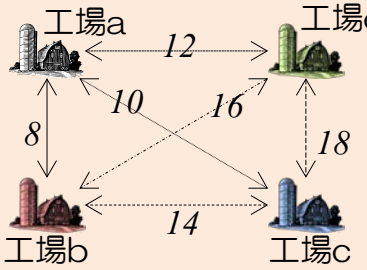
1.1　例（工場最適配置問題）

割当候補場所間の距離



	1	2	3	4
1	0	3	5	4
2	3	0	4	5
3	5	4	0	3
4	4	5	3	0

各工場間で資材のやり取り



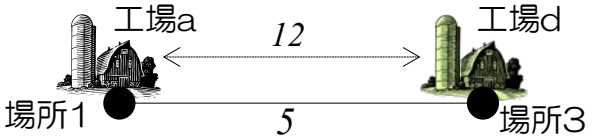
	a	b	c	d
a	0	4	5	6
b	4	0	7	8
c	5	7	0	9
d	6	8	9	0

①各場所に各工場を割当てる

コスト は 距離 × 流量

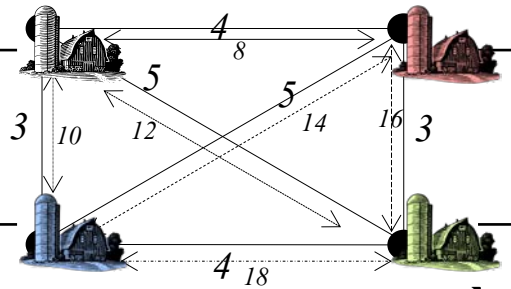
二次式の計算

ex.)



【コスト＝ 12×5 ＝ 60】

②コストの総和を 最小化する割当！ を求める



【最小化割当の例(コスト総和312)】

1．はじめに

1.2 記号化（二次割当問題 Quadratic Assignment Problem ：以下QAP）

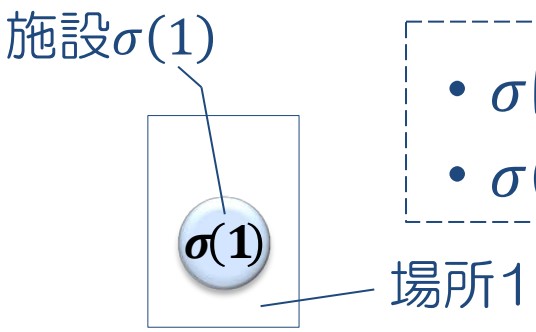
- n個の場所にn個の施設を割当てる
- 次のコストを最小化するように割当てる問題

$$\sum_{i=1}^{n}\sum_{j=1}^{n} d_{ij}\, f_{\sigma(i)\sigma(j)}$$

場所間距離 と 施設間流量 の積の総和

- 場所の集合： $N = \{1,2,\ldots n\}$
- 施設の集合： $N' = \{1,2,\ldots n\}$
- 施設の割当： $N(=N')$ 上の置換 σ

- σは解を示す　ex.) $\sigma = (1,2,5,3,4)$
- $\sigma(i)$ で場所 i に置かれる施設を表す



1．はじめに
1.3 定式化のポイント

- 定式化は置換 σ ではなく0-1変数 x_{ik} を使用する

QAP

非線形整数計画問題

目的関数が2次式

$$min \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n}\sum_{\ell=1}^{n} d_{ij}f_{k\ell}x_{ik}x_{j\ell}$$

補助変数を用いれば

線形整数計画問題（ＬＰ）
として定式化可能

場所 i に施設 k を割当てる(1)
割当てない(0)

既存の0-1変数 x_{ik}

ただし，変数の個数は
n^2からn^4に増加

$$y_{ijk\ell} = x_{ik}\ x_{j\ell}$$

新たに0-1変数 $y_{ijk\ell}$ の導入

場所 i に施設 k を，かつ，
場所 j に施設 ℓ を割当てる(1)
割当てない(0)

1．はじめに
1.4 定式化

(QAP)

min. $$\sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n}\sum_{\ell=1}^{n} d_{ij}f_{k\ell}y_{ijk\ell} \quad (1)$$

subject to $$\sum_{k=1}^{n} x_{ik} = 1 \qquad i = 1,2,\dots n \quad (2)$$

$$\sum_{i=1}^{n} x_{ik} = 1 \qquad k = 1,2,\dots n \quad (3)$$

$$y_{ijk\ell} \geq x_{ik} + x_{j\ell} - 1 \quad (4)$$

$$y_{ijk\ell} \leq x_{ik} \quad (5)$$

$$y_{ijk\ell} \leq x_{ik} \qquad i,j,k,\ell = 1,2,\dots n \quad (6)$$

$$x_{ik} \in \{0,1\} \qquad i,\ k = 1,2,\dots n \quad (7)$$

$$y_{ijk\ell} \in \{0,1\} \qquad i,j,k,\ell = 1,2,\dots n \quad (8)$$

各場所には
1つの施設が配置される

各施設をいずれか
1つの場所に配置する

(4)～(6)式全てで
$y_{ijk\ell} = x_{ik}x_{j\ell}$ と等価

1．はじめに
1.5　問題点

定式化を用い，ベンチマーク問題[1]を汎用ソルバーGurobiで計算した
(実験環境： Windows7，intel core i5 2.67GHz，メモリ32GB)

chr12a($n = 12$) →約18分で最適解を求めることができた

chr15a($n = 15$) →約4時間半で最適解を求めることができた

bur26a($n = 26$) →1週間（！？）計算しても最適解が求まらなかった

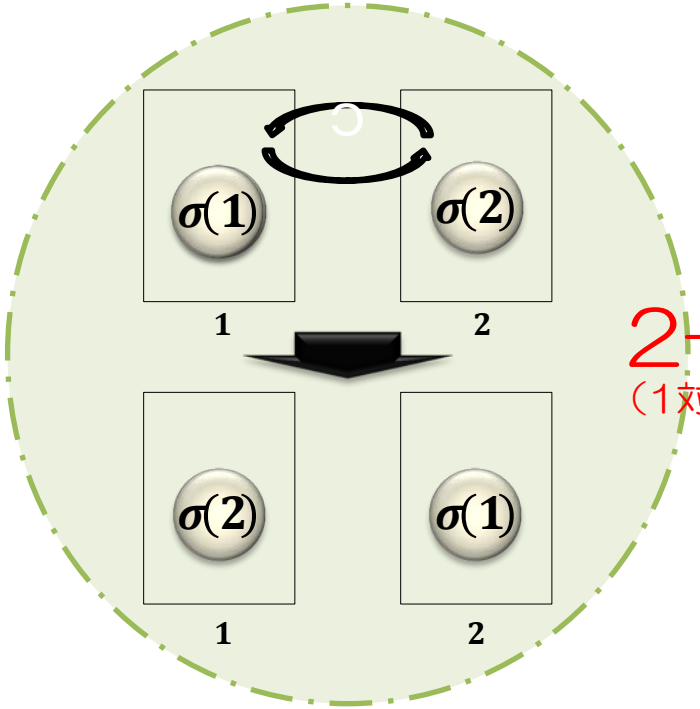
ＱＡＰは組合せ最適化の中でも“難しい”問題[2]
→ n が大きくなると厳密に解くことは困難になる

発見的解法の研究が重要!
n が大きいときにも，十分な精度の解を求める解法の研究

1．はじめに

1.6 既存の発見的解法[3]

- 局所探索法
- タブー探索法　etc..



基本探索
2-swap
（1対の施設交換）

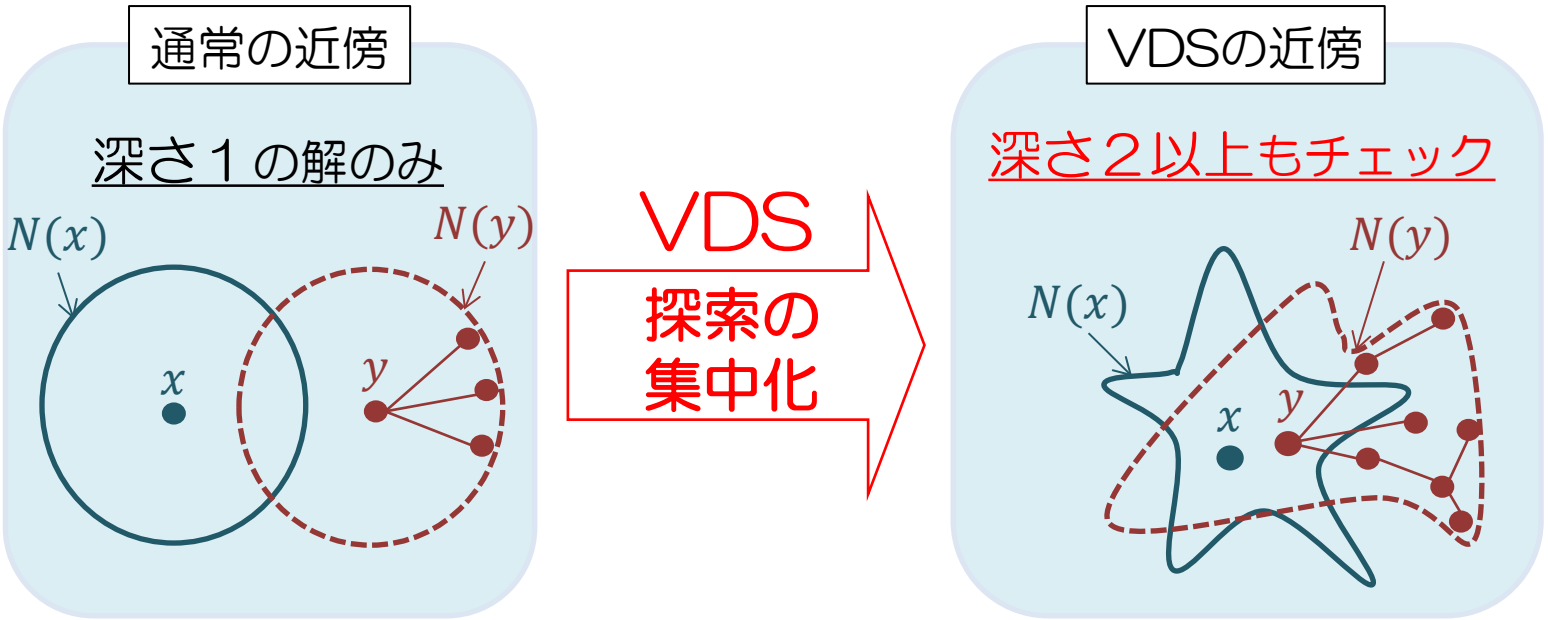
強化

可変深度
近傍探索

1．はじめに

1.7 可変深度近傍探索（Variable Depth Search：以下VDS）

- 高度な近傍の構成法の１つ[3]
- 単純な近傍操作を連鎖的に複数回反復することで生成され得る解集合 ⟹ あらためて近傍と定義



１．はじめに

1.8　QAPに対するVDSの適用

組合せ最適化の経験則に鑑みる

QAPに
対する適用

とても良い改善を見出す可能性!!

2．本研究の目的

二次割当問題(QAP)に対する

- 可変深度近傍探索(VDS)の具体化
- VDSの性能評価，考察

３．提案解法

3.1　方針

VDSのポイント１

小さな変更を連鎖して大きな変更を実現する

VDSのポイント２

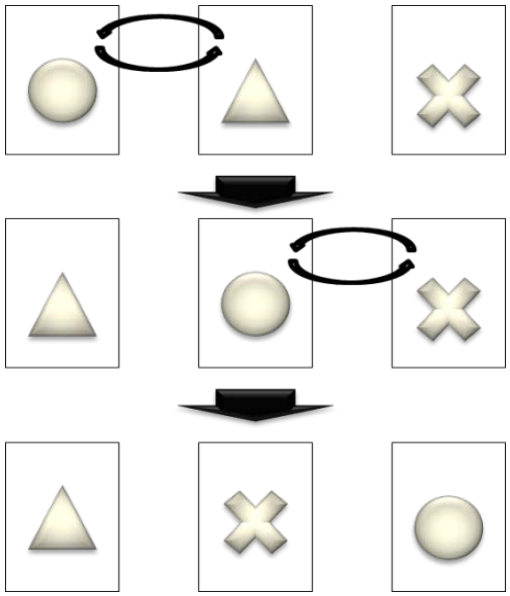
連鎖の長さを動的に制御する

3．提案解法
3.2 具体化(1)

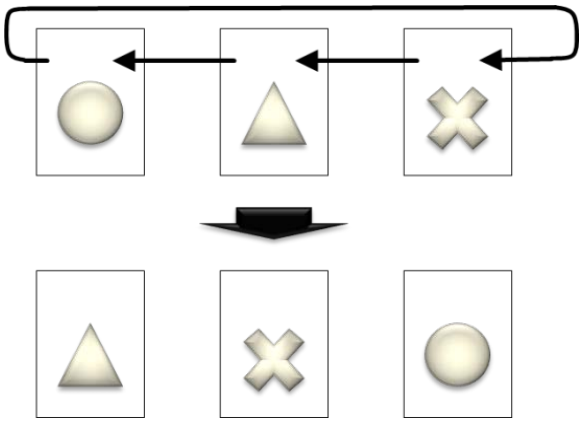
VDSのポイント１

小さな変更を連鎖して大きな変更を実現する

2-swapの連鎖
（1対の施設交換）



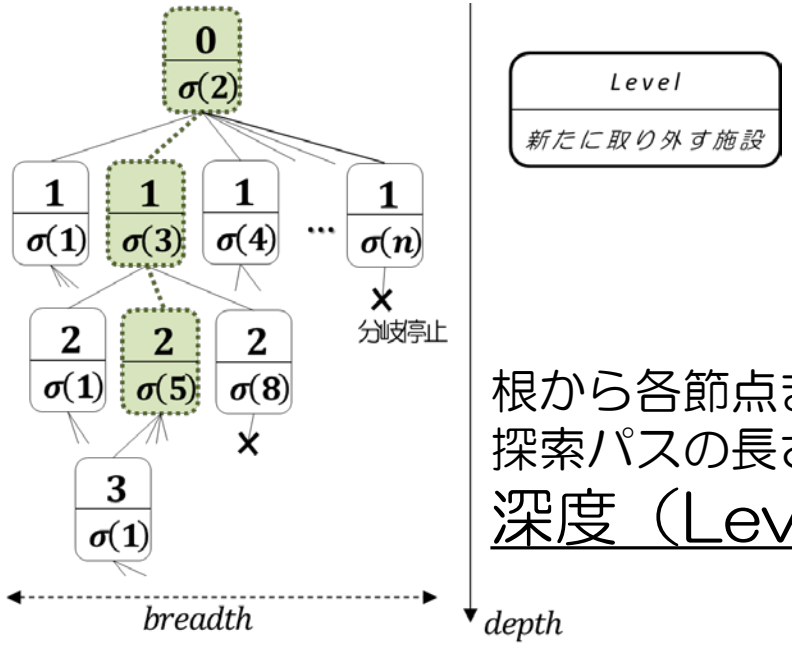
巡回交換



3．提案解法
3.3 具体化(2)

VDSのポイント２

連鎖の長さを動的に制御する
⇒探索木の導入



根から各節点までの
探索パスの長さ
深度（Level）

ルール
見込みのある枝は採用
見込みのない枝は不採用

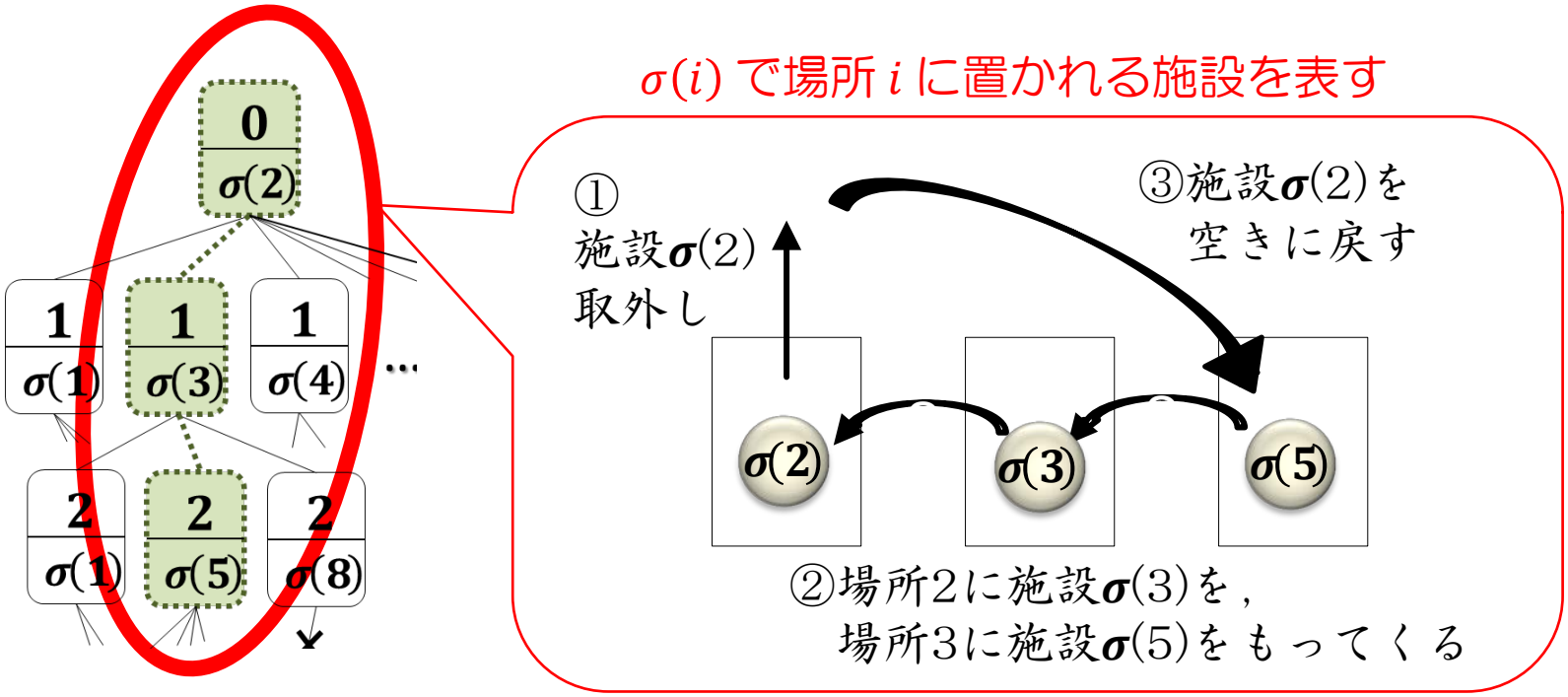
交換前と交換後の
コストの差 $(= Gain)$ で
深度を
増やすかどうかを常時判断

3．提案解法
3.4 方法

VDSのポイント１ ＋ VDSのポイント２

探索木の頂点から $Level = \alpha$ （ α は0以上の整数）まで辿っていく

探索木の節点に対応するように施設を巡回交換させる



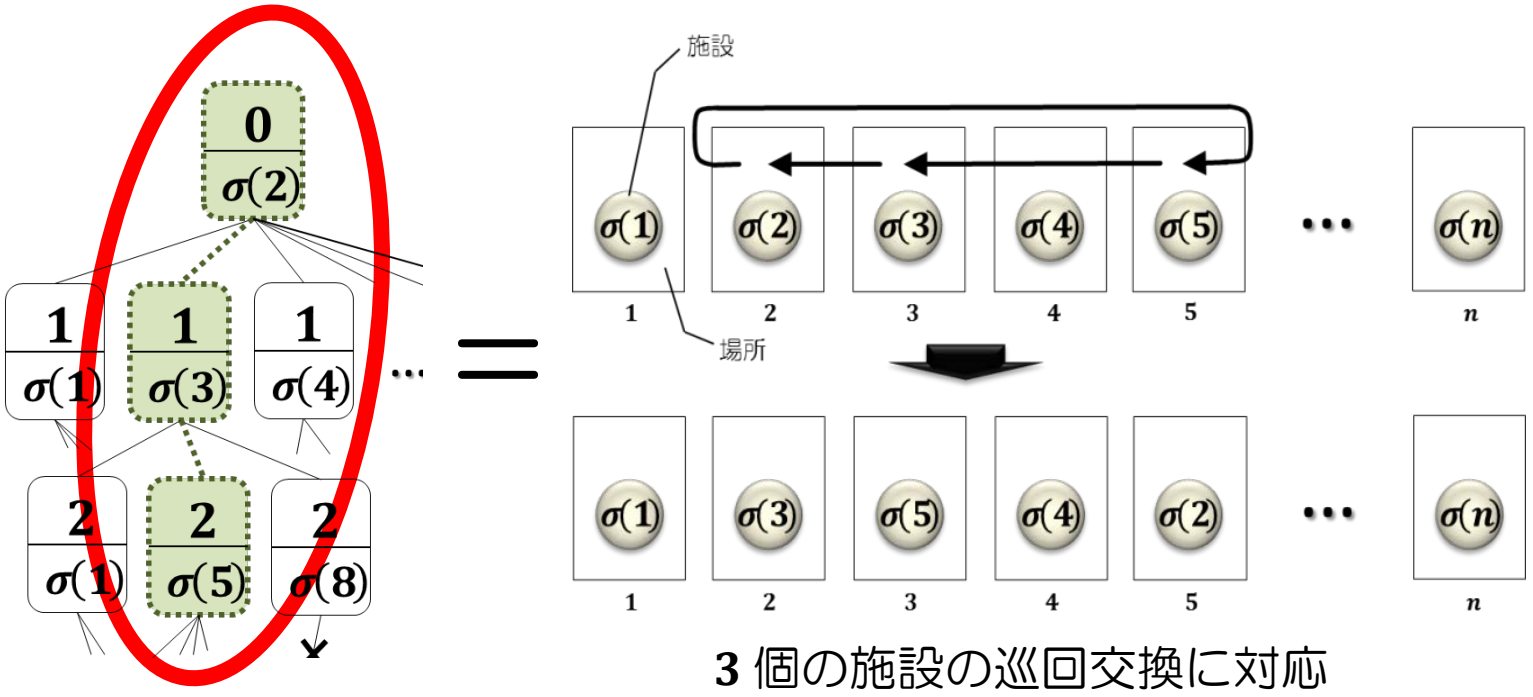
3．提案解法

3.4 方法

VDSのポイント１ ＋ VDSのポイント２

探索木の頂点から $Level = \alpha$ （ α は0以上の整数）まで辿っていく

探索木の節点に対応するように施設を巡回交換させる



3 個の施設の巡回交換に対応

3. 提案解法
3.5 方法の問題点と対策

特に制約がない場合
⇒探索木の大きさが際限なく広がる

計算量の観点から，
- 最大深度(定数)
- 各深度毎に最大幅(定数)　を設定

それぞれ１回の巡回交換に関わる施設の最大個数と，
その深度における施設の最大選択数にあたる

3．提案解法

3.6　解法（Step1～Step4）

Step1：初期解 σ をランダムに作成

Step2：巡回交換に関わる施設のリスト$[L^{(0)}, L^{(1)}, \ldots L^{(n)}]$の初期化

$L^{(0)} = 1;\ \alpha = 1;$

Step3：$\alpha > 0$の間，Step3-aもしくは Step3-bを繰り返す

（Step3-a）深度 α 増加の手続き．深度αにおける $Gain$のリスト作成

$Gain$のリストを降順にソートし，

$Gain$が大きく未探索の施設の交換を実施，コスト計算

改善時は，即座に解を更新し，Step2へ戻る

（Step3-b）深度 α が(において）最大深度(幅)を超えたとき，

$Gain < 0$ のときは，深度 α 減少の手続き

Step4：$L^{(0)}{+}{+}$；$\alpha = 1;\ L^{(0)} \leq n$ ならば，step3へ戻る

$n < L^{(0)}$ ならば，VDS終了につき，解の出力

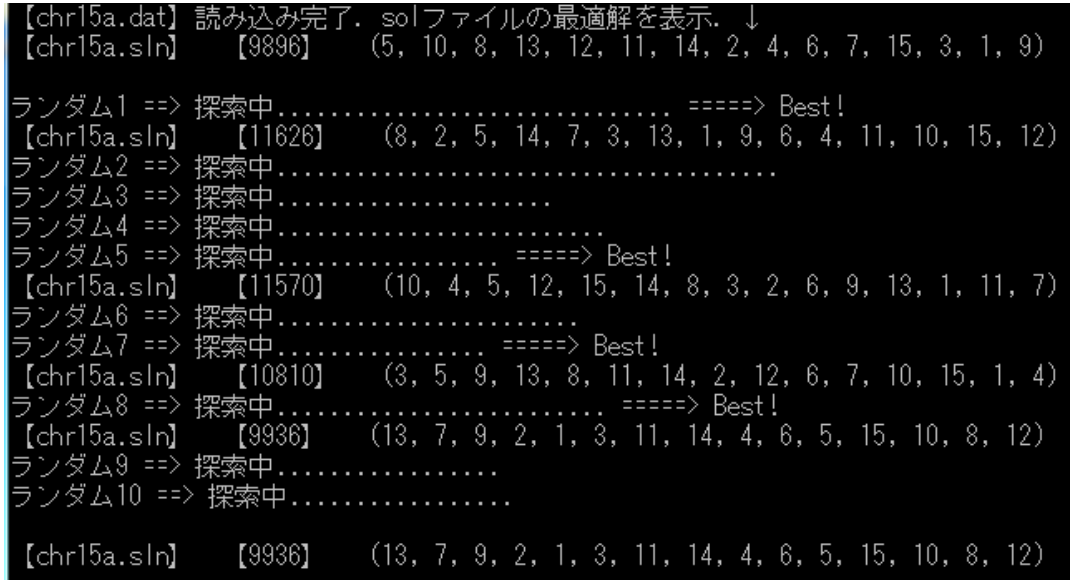
3．提案解法

3.7　解法（Step5）

Step5：Step1～4を10～100回程度繰返し最良解を出力
（多スタート探索）

4．数値実験
4.1　Delphi６での実験画面

操作画面



実行画面（chr15a）

４．数値実験
4.2　実験結果

- 交換近傍よりも良い解を発見できた
- chr15aやnug30は最適解を見出した
- 表以外でも,
  $n \leq 20$ の多くの問題で最適解を見出した

問題名	10通りの初期解 交換近傍, BestMove 誤差（%）	10通りの初期解 VDS 誤差（%）	100通りの初期解 VDS 誤差（%）	実験1回あたり VDS 平均実行時間（s）
chr15a	23.88400	0.40420	0.00000	0.06
nug30	2.25343	0.29393	0.00000	6.45
tai80a	3.03751	1.52485	1.51288	948.49
sko90	1.48181	0.66993	—	50582.08

注：sko90は計算に時間がかかった為，100通りの初期解の測定はしていない.

最大深度４，幅は深度順にn,n,5,5,5

5．考察

n (サイズ)が大きくなると実行時間がかなり増大する

【理由】
コスト計算に時間がかかることが挙げられる

探索木の最大幅を設定して余計なチェックを減らし，
最大深度を制限することで見込みの低い探索をしないようにはしているものの，実行時間が指数関数的に増加してしまった

【対策】
「探索木の最大幅を小さくする」などで高速化が可能

⇒数値実験との比較の為に，sko90で試したところ
（最大深度４，最大幅n,1,5,5,5）
既存の最良解との誤差は　約0.67%→約1.13%　（増加↑）
1回あたりの計算時間は　約14時間→約1分　（減少↓）　となった

６．まとめ

- VDSの動作を具体化し，QAPに適用した
- $n \leq 20$ の多くの問題で最適解を見出すことに成功
- $20 < n \leq 30$ の一部の問題は最適解を見出せた
- n が大きいときは，実行時間が大きく増える

今後の課題

- 最大深度（幅）の調節法を調べる必要がある
- 深度の増減時の評価式を工夫して
  高速化する必要がある

参考文献

[1]QAPLIB,
http://www.opt.math.tu-graz.ac.at/qaplib/inst.html(2013.12.25)
[2]藤澤克樹，梅谷俊治(2009)，
応用に役立つ50の最適化問題，朝倉書店，134pp.
[3]柳浦睦憲，茨木俊秀(2001)，
組合せ最適化－メタ戦略を中心として－，朝倉書店，237pp.
[4]沼田一道，濱川大志(2008)，
自動倉庫の入出個スケジューリング問題に対する可変深度近傍を用いた発見的解法，システム制御情報学会論文誌，21(5),139-141.

ご清聴ありがとうございました