日本オペレーションズ･リサーチ学会
サプライチェーン･ネットワーク研究会
2005.11.04

サプライチェーン(SCM)ネットワーク最適化の課題
Part II：最適化技術の適用性

(株) 数理モデリング研究所

野末 尚次

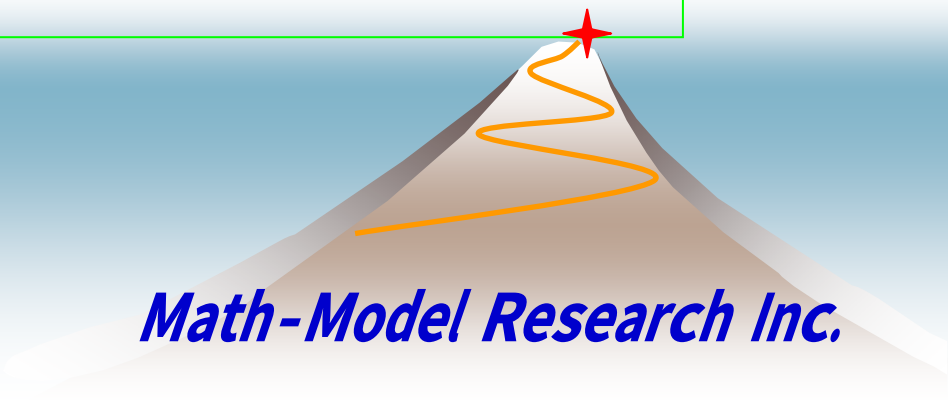
URL:www.math-model.co.jp

今日の話題（PartⅡ）

PartⅠ：SCNシステムの動的安定性
- 1）サプライチェーン・ネットワークの課題
- 2）サプライチェーン・ネットワークの安定性（CPFR)
- 3）業務の意思決定支援システム（DSS)
- 4）まとめ（PartⅠ）

PartⅡ：最適化技術の適用性
- **5）最適化技術の現状（整数計画法と制約プログラミングを中心）**
- **6）最適化技術の適用性**
- **7）まとめ（PartⅡ）**

PartⅢ：鉄道への応用
- 8）鉄道にみるSCNの課題
- 9）まとめ（PartⅢ）



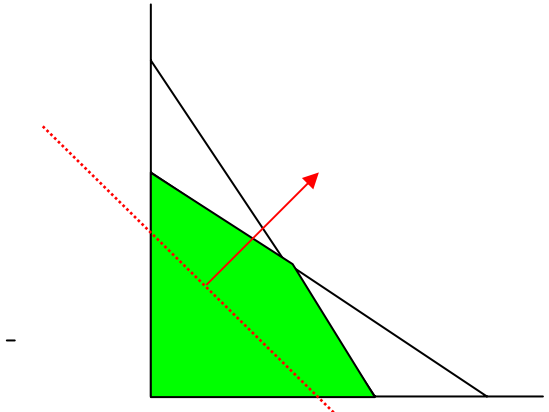
Math-Model Research Inc.

最適化問題の分類 (形式的)

問題の定義要因	性質	分類	英語表記 (略称)
変数のタイプ	実数	連続型	Continuous
	実数と整数	混合整数型	Mixed Integer (MIP)
	整数	整数型	Integer Programming(IP)
	記号	非数値型	Symbolic
制約条件 目的関数	線形	線形計画	Linear Programming(LP)
	2次式	2次計画	Quadratic Programming(QP)
	非線型	非線形計画	Non-linear Programming(NLP)
	論理／非数値	シンボリック	Symbolic
制約条件の有無	制約なし	非制約最適化	Unconstrained
	制約あり	制約付最適化	Constrained
データ	確定	確定的	Deterministic
	不確定	確率的	Stochastic Programming

最適化問題の分類 (質的相違)

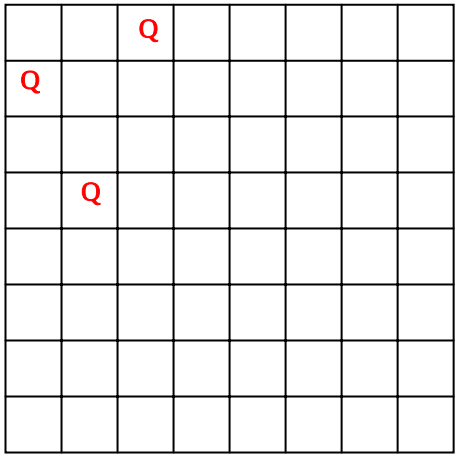
資源配分問題



$$2x + 3y \leq 12$$
$$3x + 2y \leq 12$$
$$x \geq 0,\ y \geq 0$$

$$x + y \rightarrow Max$$

8-Queen配置問題



$$x[i] \neq x[j],\ for \quad i \neq j$$
$$x[i]+i \neq x[j]+j,\ for \quad i \neq j$$
$$x[i]-i \neq x[j]-j,\ for \quad i \neq j$$

宣言的アプローチ(1):制約論理、制約プログラミング

- 論理言語(Logic Programming):Prolog
  - 論理的な制約条件を宣言的に記述可能
  - ユニフィケーションとバックトラックによる探索

  ⇒ 計算時間がかかる。
- 制約充足問題(Constraint Satisfaction)
  - 有限領域を対象とした効率的な探索方式

  ⇒ 問題の定式化に自由度ない。
- 平行処理(Parallel Processing)
  - 複数のプロセスにより計算状況を監視

制約論理言語:CHIP(Constraint Handling in Prolog)

制約ツールキット:ILOG(C++で制約ベースの処理を実現)

制約充足理論入門

制約充足問題：制約条件が与えられた時、それを満たす解を求める。

4－Queen問題：縦、横、斜め方向にQueenが重ならない配置を求める。

	A	B	C	D
1			Q	
2	Q			
3				Q
4		Q		

定式化：各行のQueenの位置を変数にする

変数： x1、x2、x3、x4

制約条件：

1) **縦：** x1≠x2、x1≠x3、x1≠x4、
x2≠x3、x2≠x4、x3≠x4

2) **右上：** x2＋1 ≠x1
x3＋2 ≠x1、x3＋1 ≠x2、x4＋3 ≠x1、‥

2) **右下：** x2－1 ≠x1
x3－2 ≠x1、x3－1 ≠x2、x4－3 ≠x1、‥

バックトラック法による制約充足

	A	B	C	D
1	Q_1			
2	$\times_2$	$\times_3$	Q_4	
3	$\times_5$	$\times_6$	$\times_7$	$\times_8$
4				

	A	B	C	D
1	Q_1			
2	$\times_2$	$\times_3$	$\times_9$	Q_{10}
3	$\times_{11}$	Q_{12}		
4	$\times_{13}$	$\times_{14}$	$\times_{15}$	$\times_{16}$

	A	B	C	D
1	Q_1			
2	$\times_2$	$\times_3$	$\times_9$	Q_{10}
3	$\times_{11}$	$\times_{17}$	$\times_{18}$	$\times_{19}$
4				

	A	B	C	D
1	$\times_{20}$	Q_{21}		
2	$\times_{22}$	$\times_{23}$	$\times_{24}$	Q_{25}
3	Q_{26}			
4	$\times_{27}$	$\times_{28}$	Q_{29}	

1) 1行目は、(1、A) に置く
2) 2行目は、(2、A)、(2、B)はダメ、(2、C)に置く
3) 3行目は、全部ダメ ⇒ バックトラック
4) 2行目の、(2、C)を中止して、(2、D)に置く
5) 3行目は、(3、A)はダメ、(3、B)に置く
6) 4行目は全部ダメ ⇒ バックトラック
7) 3行目の、(3、B)を中止して、(3、C)、(3、D)も全部ダメ ⇒ バックトラック
8) 2行目の、(2、D)を中止して、残りなし ⇒ バックトラック
9) 1行目の、(1、A)を中止して、(1、B)に置く

バックトラック法の問題点

問題点：　無駄な探索が多く、時間が掛かる

あるQueenの位置を決めた時に、

・そのQueenと同じ列には、他のQueenは置けない。

・そのQueenと対角線上にある場所は、置けない。

↓

整合性：　可能性のない値を事前にできる限り削除すればよい

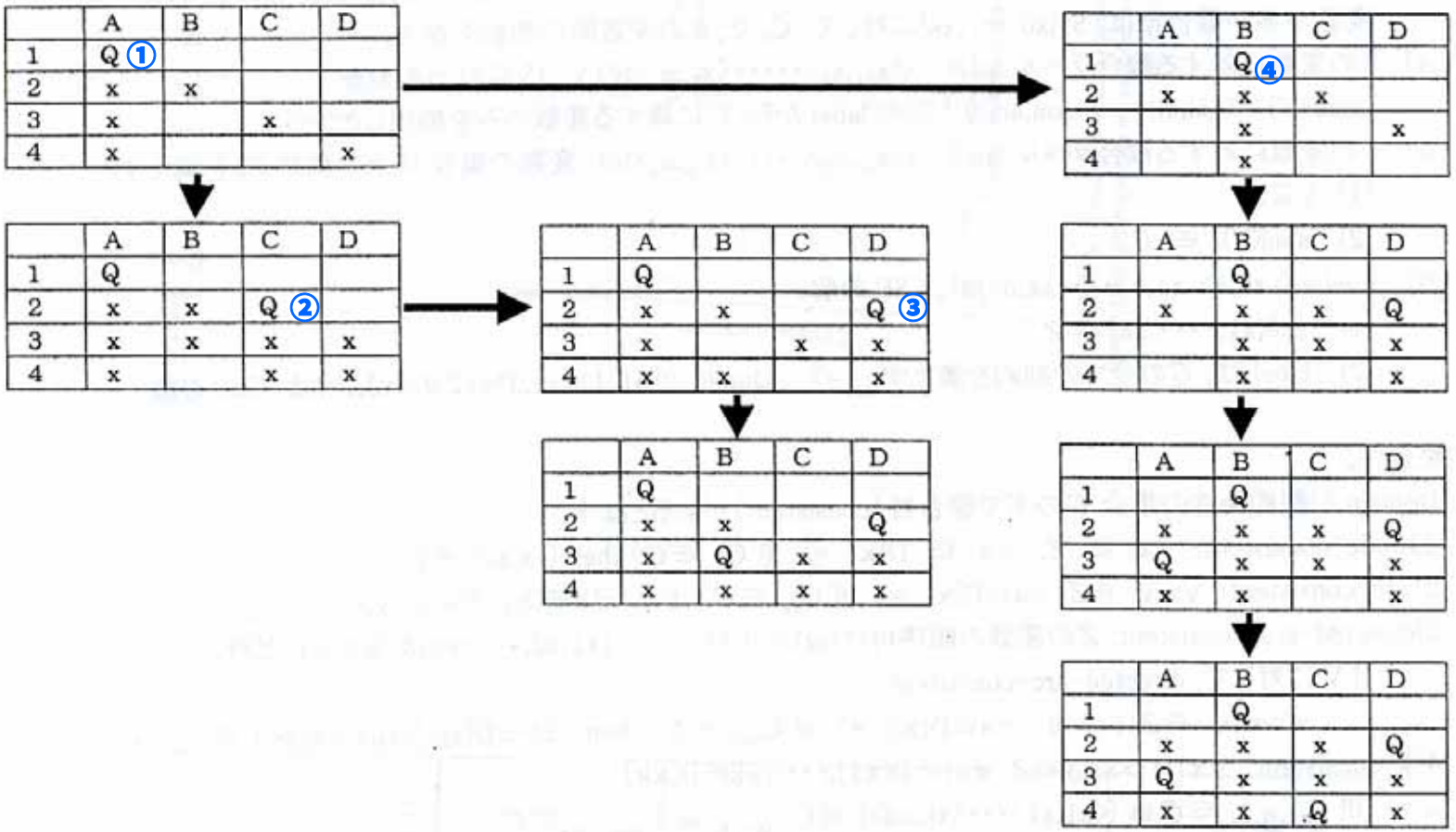
・アーク整合性、・k－整合性、・・・・

↓

制約伝播：効率的に整合性を維持するアルゴリズム

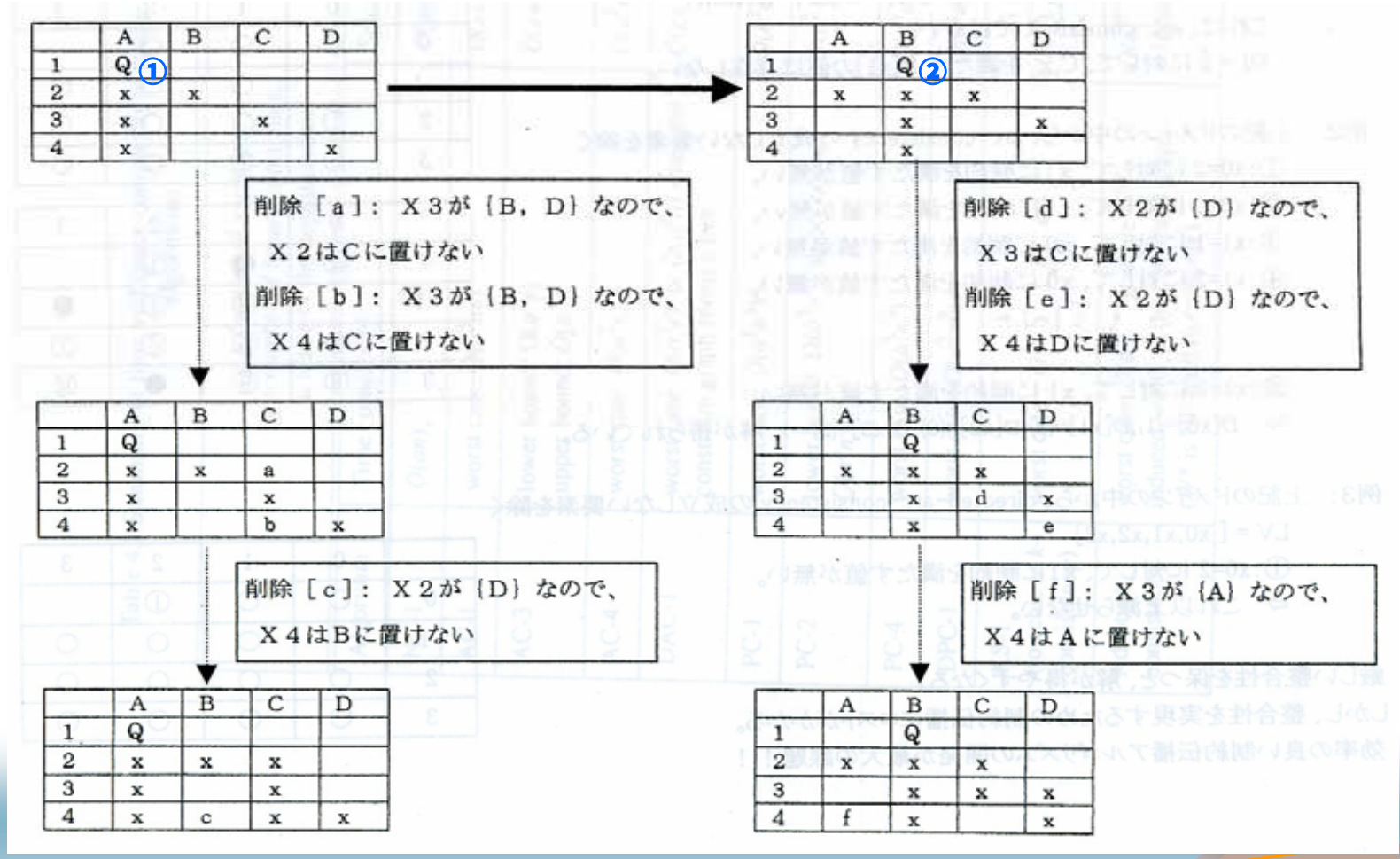
制約伝播（前方検査）を併用した制約充足

前方検査：1つの変数の値が決まったら、他の変数の無効な値を削除する。

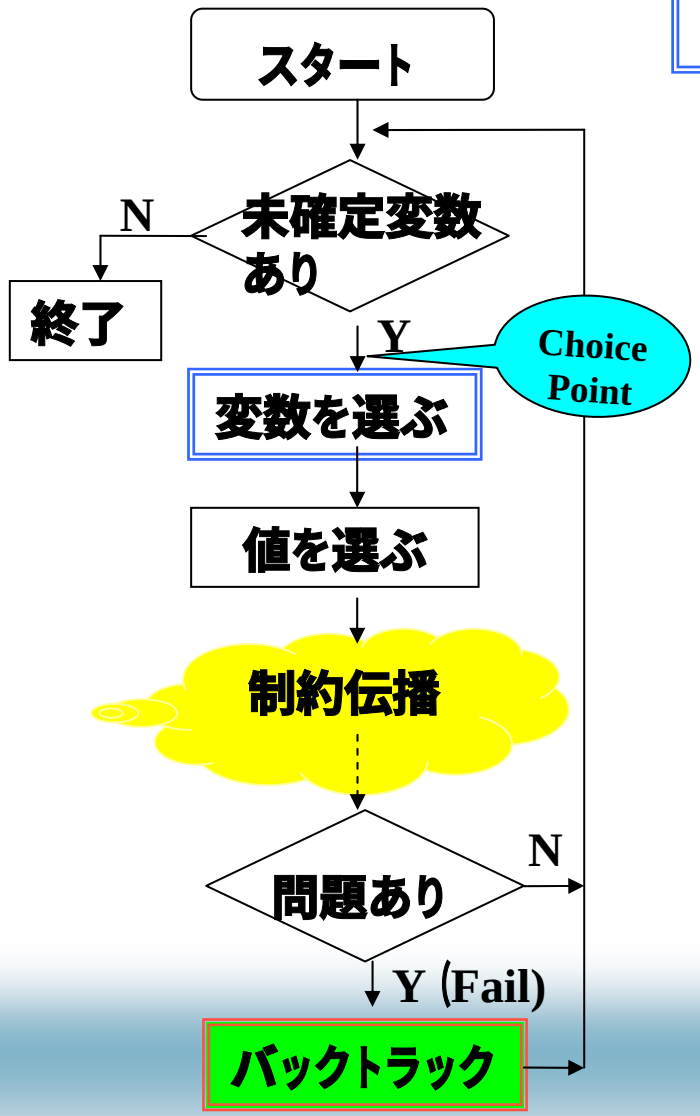


制約伝播（アーク整合性）を併用した制約充足

アーク整合性：2つの変数の組に対して、可能性のない値を削除する。



CPの基本構造と探索方式の重要性



Choice Point:変数（値）の選択順序が重要

N-Queen問題のILOG Solverによるテスト結果

素朴な探索方式：
変数の並び順に決める。
x1->x2->x3->

N=30:　Fail=7,472,978　Time=163 sec

First-Fail 探索方式：
その時点で
可能な値の最も少ない変数から決める

N=30:　Fail=32　Time=0.14 sec

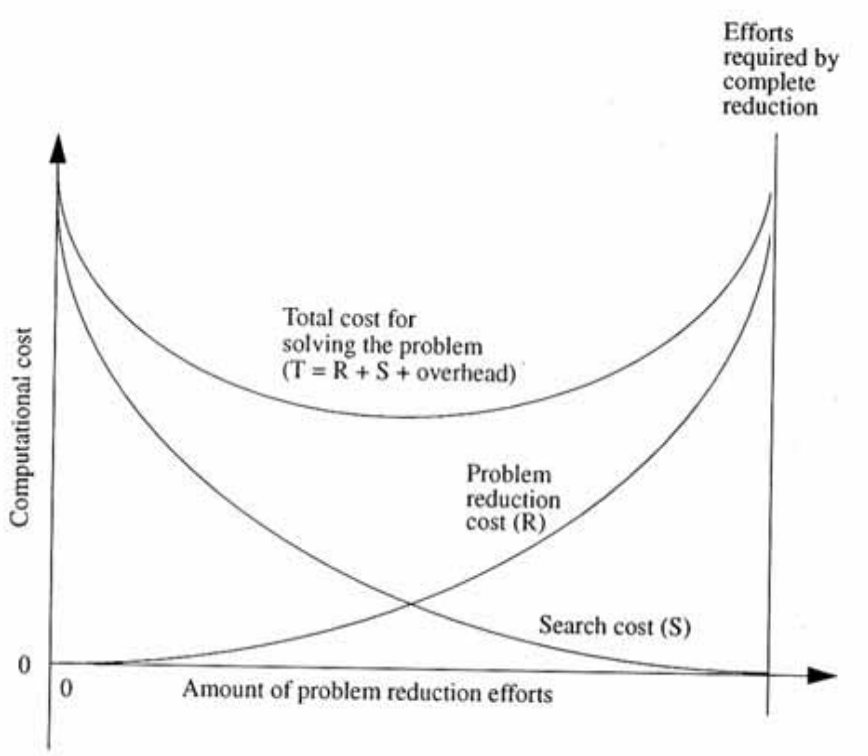
N=3000:　Fail=8　Time=19 sec

Math-Model Research Inc.

制約伝播と探索コスト

n = number of variables;
e = number of binary constraints;
a = size of the largest domain

Algorithm	Time complexity	Space complexity
NC-1	$O(an)$	$O(an)$
AC-1	worst case: $O(a^3ne)$	$O(e+na)$
AC-3	lower bound: $\Omega(a^2e)$ upper bound: $O(a^3e)$	$O(e+na)$
AC-4	worst case: $O(a^2e)$	$O(a^2e)$
DAC-1	worst case: $O(a^2e)$; or $O(a^2n)$ when the constraint graph forms a tree	$O(e+na)$
PC-1	worst case: $O(a^5n^5)$	$O(n^3a^2)$
PC-2	lower bound: $\Omega(a^3n^3)$, upper bound: $O(a^5n^3)$	$O(n^3+n^2a^2)$
PC-4	worst case: $O(a^3n^3)$	$O(n^3a^3)$
DPC-1	worst case: $O(a^3n^3)$	$O(n^2a^2)$
KS-1 (to achieve k-consistency)	worst case : $O\left(\sum_{i=1}^{k} ({}_nC_i \cdot a^i)\right)$	$O(\sum_{i=1}^{k} ({}_nC_i \cdot a^i))$
Adaptive_ consistency	worst case: $O(a^{W^*+1})$, where W^* = induced-width of the constraint graph (W^* is NP-hard to compute)	$O(a^{W^*})$, where W^* = induced-width



出典：Tsang,Foundations of Constraint Satisfaction,1993,Academic Press

制約プログラミング入門

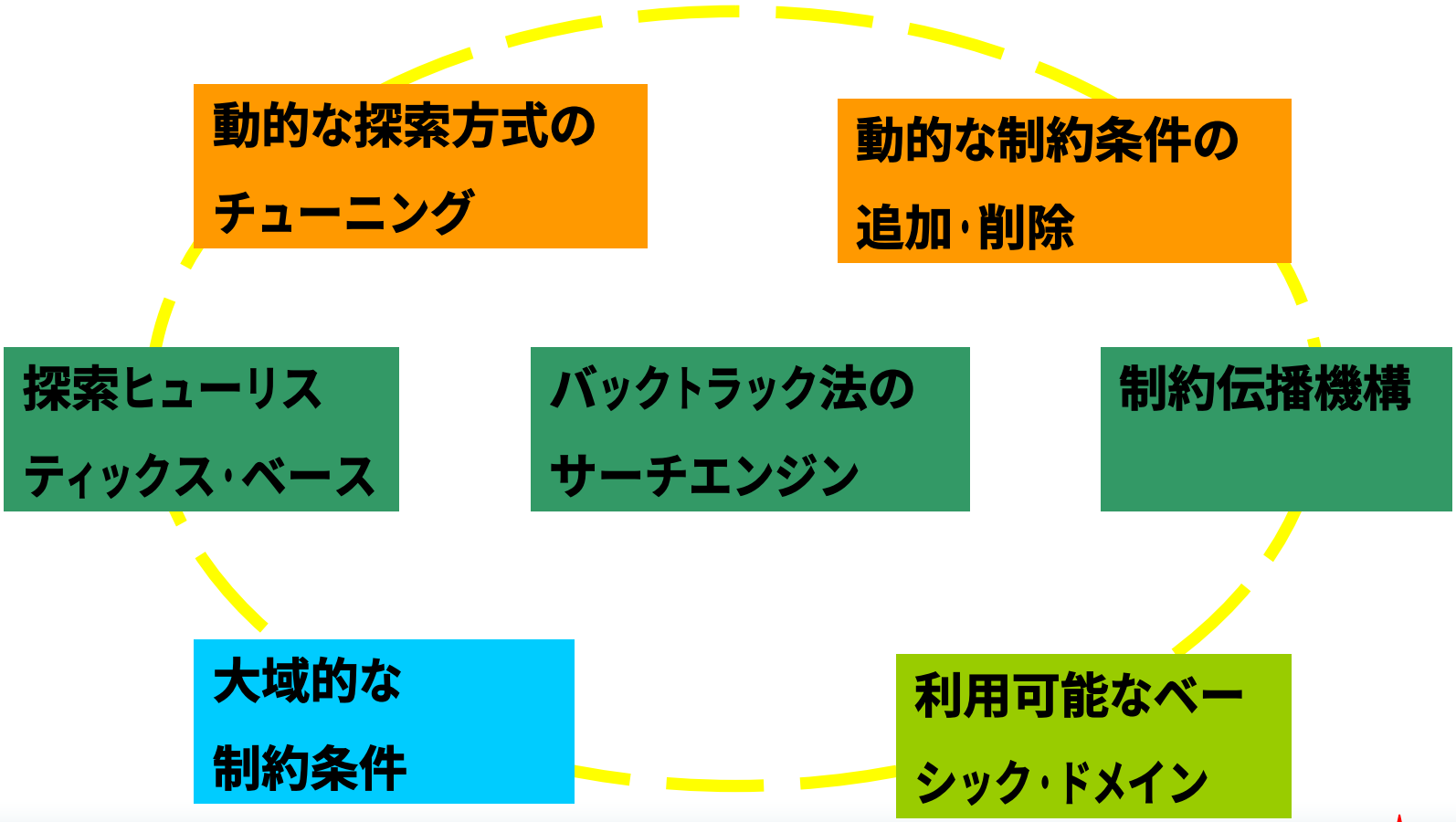
- **記述力が高い**
  - モデル記述：モデル記述言語（ＯＰＬ，ＡＭＰＬ，ｅｔｃ）
  - プログラム：　基本的な記述形式　m.add(2*x+3*y ≦5)；
- **制約の例**
  - スケジューリング用オブジェクト：activityとresourceによる制約表現
  - actA.startsAfterEnd(ActB,5)：　相互の時刻に制約
  - actA.requires(resourceX)　：同じリソースを要求するものの競合管理
  - AllDifferent：全ての変数は、互いに同一の値を取れない。
  - Path制約：　全てのノードをパスでカバーする。
  - 分配制約：複数の変数が同時にとり得る値に上下限を設定する。
  - 集合演算に関する制約（共通部分、直和制約、集合サイズ制約）
- **ユーザに自身による制約伝播が書ける。**
  - デーモンによる並列処理
  - 指定した変数の可能な値が変化した時
    （whenDomain,whenRange,whenValue）

Math-Model Research Inc.

制約プログラムの例 (部分)

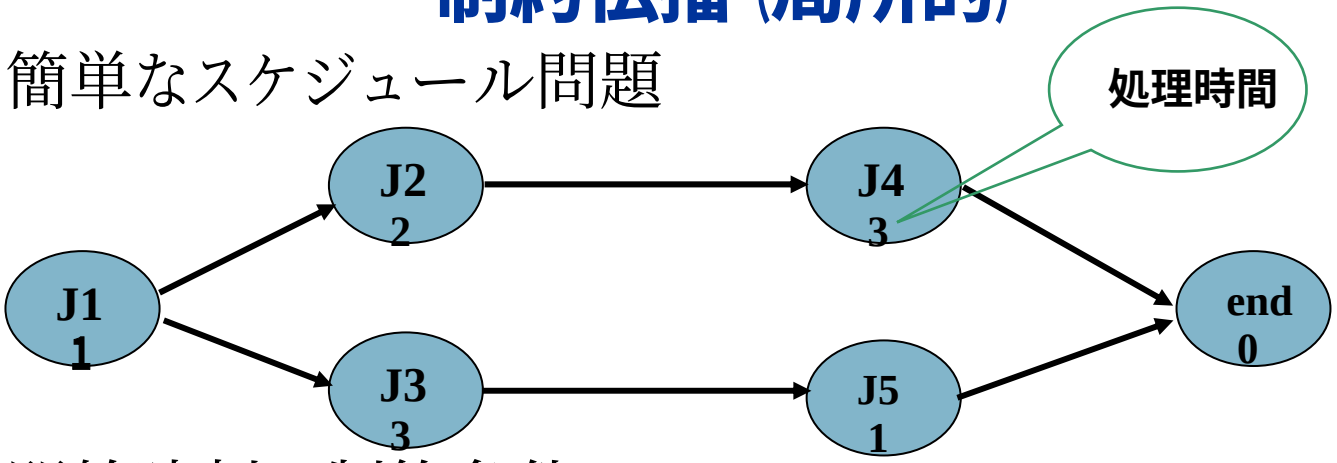
```
IlcManager m(IlcNoEdit);                                    マネージャの定義

IlcIntVarArray x(m, nqueen, 0, nqueen-1),                   変数の定義
                    x1(m, nqueen),    x2(m, nqueen);
IlcInt i;
for (i = 0; i < nqueen; i++) {
          x1[i] = x[i] + i;    x2[i] = x[i] - i;  }         変数間の制約

m.add( IlcAllDiff(x) );                                     制約条件の定義 (制約伝播)
m.add( IlcAllDiff(x1) );
m.add( IlcAllDiff(x2) );

m.add( IlcGenerate(x, IlcChooseMinSizeMin) );               探索法の指定 (First Fail)

if ( m.nextSolution() ) {                                   解の探索
          for (i=0; i < nqueen ; i++ )    m.out() << x[i].getValue()  << " ";
          m.out() << endl;
}  else m.out() << "No solution" << endl;

m.printInformation();
m.end();                                                    Version 4
```

制約プログラミングの構造



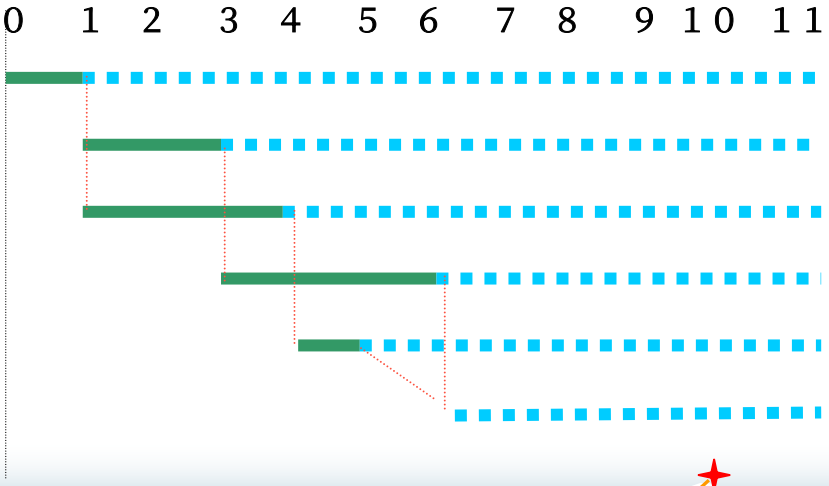
制約伝播 (局所的)

- 簡単なスケジュール問題



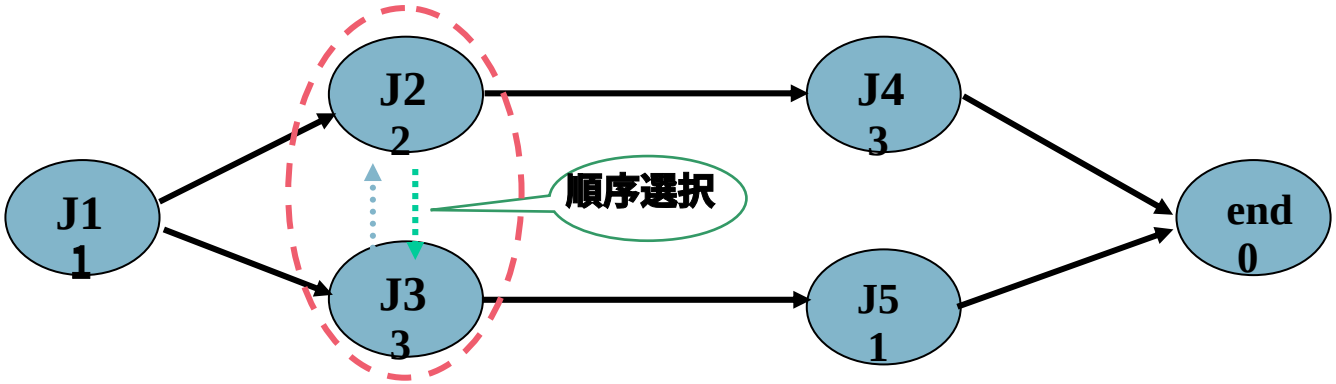
- 開始時刻の制約条件
  - **t1 ≧ 0**
  - **t2 ≧ t1 + 1**
  - **t3 ≧ t1 + 1**
  - **t4 ≧ t2 + 2**
  - **t5 ≧ t3 + 1**
  - **te ≧ t4 + 3**
  - **te ≧ t5 + 1**

制約伝播



制約伝播(リソース制約)

- リソース制約のあるスケジュール (J2とJ3が同時実行不可)



- リソースの制約条件
  - J2 ー＞ J3

    t3 ≧ t2 + 2
  - J3 ー＞ J2

    t2 ≧ t3 + 3

制約伝播

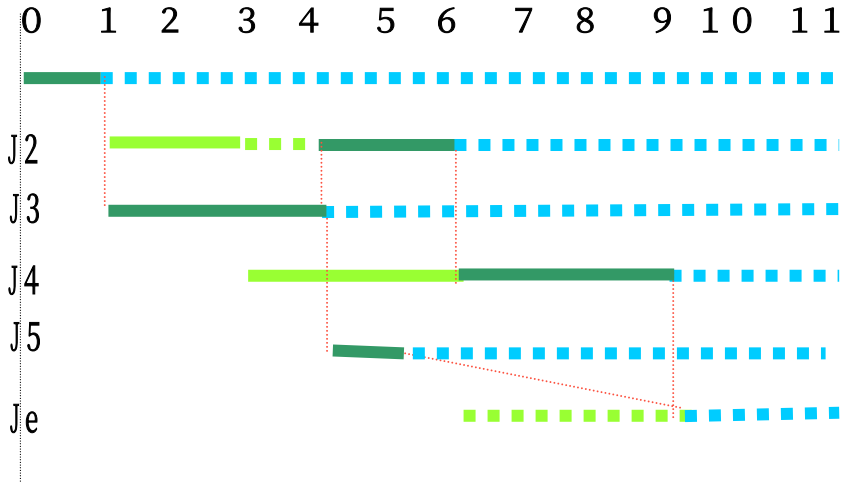
Math-Model Research Inc.

制約伝播(排他的な条件)

- リソース制約のあるスケジュール (逆向きの場合)

- J3 ー＞ J2
  t2 ≧ t3 + 3

制約伝播



J2 ー＞ J3 : **te ≧ 7**
J3 ー＞ J2 : **te ≧ 9**

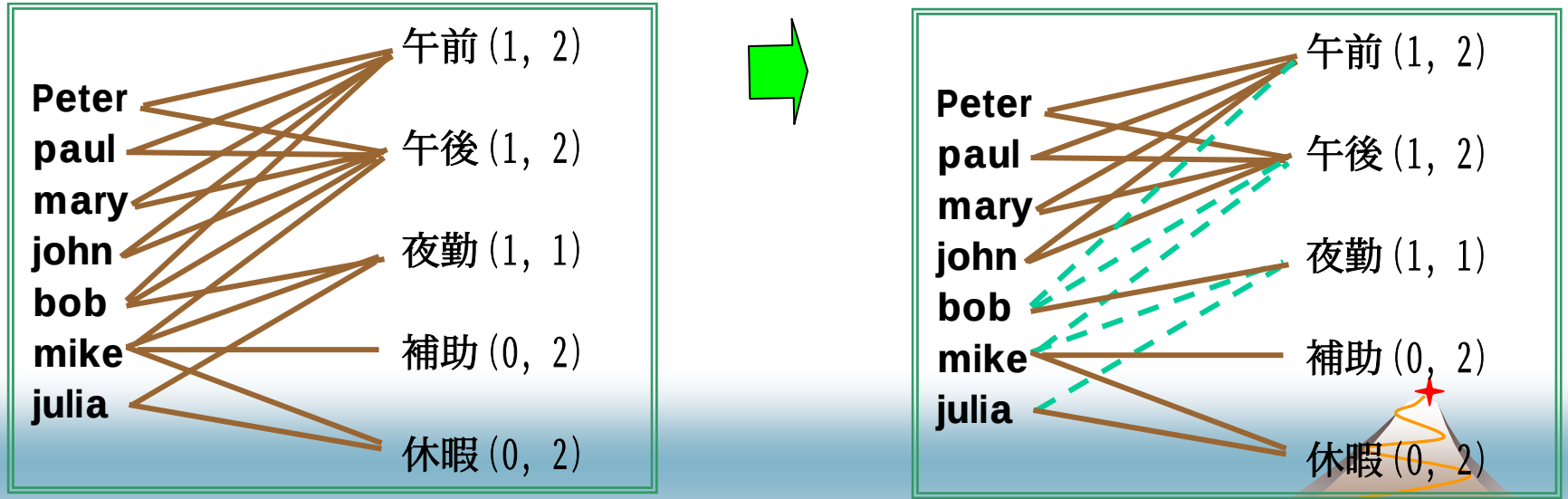
納期制約がある場合 （ **te ≦ 8** ）
もし制約伝播が完全なら、J3ー＞J2 は納期を満たさないから、
自動的に J2ー＞J3 が選択される。

このレベルでの制約伝播は計算が大変

制約（論理）プログラミングだけでは、解けない

- 制約伝播の不完全性
  - 制約（論理）プログラミングの固有の制約伝播だけでは、複雑な制約条件に対して、完全な制約伝播を行うことは、計算時間上、不可能である。
- 大域的な制約条件の導入
  - オペレーションズ・リサーチ等の理論を応用することにより、効率良く制約伝播が可能な汎用の**大域的制約条件**の導入　－＞　**メーカーの差異化**
- 例：勤務計画で、勤務毎の人数の制約：　勤務（最小、最大）

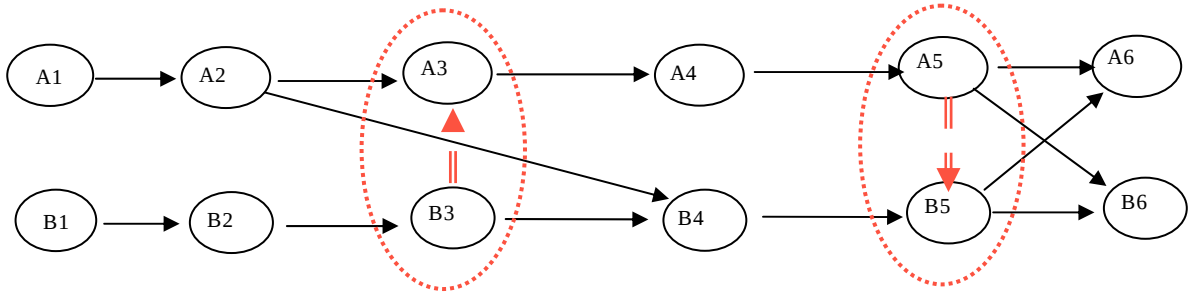
最大フロー問題を利用した制約伝播：　J.　Regin　(ILOG)



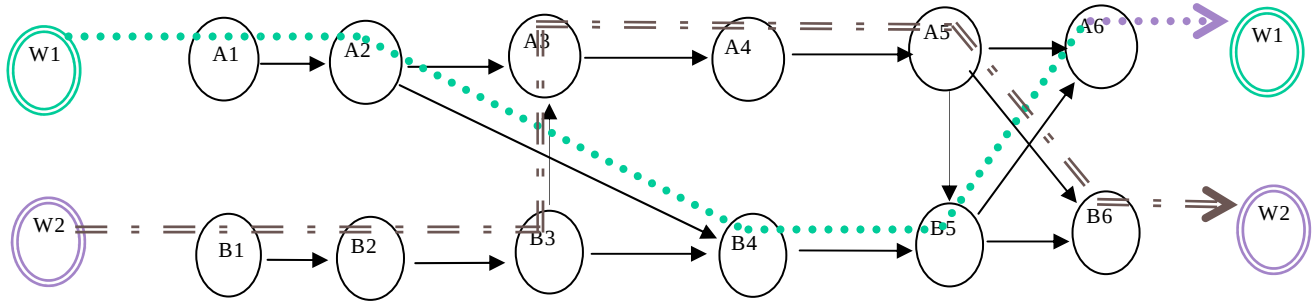
Math-Model Research Inc.

代表的な大域的制約条件とモデル

1) 各作業者の仕事の順序が与えられているが、仕事間の優先順序やリソースの競合があるモデル



2) 順序付けられた仕事に対して、作業者を割り当てるモデル



3) 日々に必要な資格別の要員数、及び、各作業員の勤務種別の時系列に対する制約のモデル

氏名	月	火	水	木	金	土	日
A	N	R	N	D	N	R	R
B	D	N					
C	R	D					
日勤	1	1	2	1	1	1	0
夜勤	1	1	1	1	1	1	1

ユーザによる制約（制約伝播）の定義

対象とする計画の
緩和問題、固有な性質、ユーザの習慣等を制約として定義

ユーザによる制約伝播の記述

[例]　X＝{1, 2, 3, 4, 5}、Y＝{1, 2, 3, 4, 5}

新しい制約条件　Less:　X ＜ Y

- 整合性の条件
  - max (X) ＜ max (Y)
  - min (X) ＜ min (Y)
- 制約伝播用関数の定義
  - ReduceX () {
    X.setMax (Y.getMax () − 1) ;
    }
  - ReduceY () {
    Y.setMin (X.getMin () ＋ 1) ;
    }
- 実行時の制約伝播条件

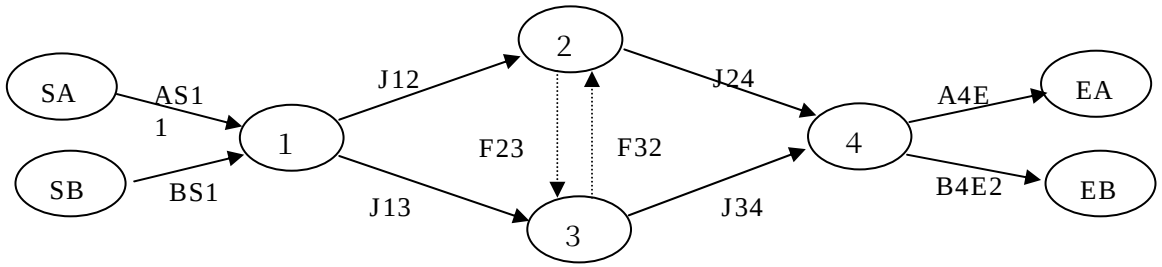
  このいずれかの変数の値域に変化が生じたら
  他の変数の整合性のチェックする条件を事前に登録:

  X.whenDomain (ReduceY):　値が一つでも欠けたら
  X.whenRange (ReduceY):　上限か下限が変化したら
  X.whenValue (ReduceY):　値が確定したら

CPの鍵：Choice Pointを少なく
=> 制約伝播で絞込（決定）

多種流問題によるスケジューリングの表現

仕事の集合**M**は、{J12,J13,J24,J34} の4種類で、次の順序関係がある。

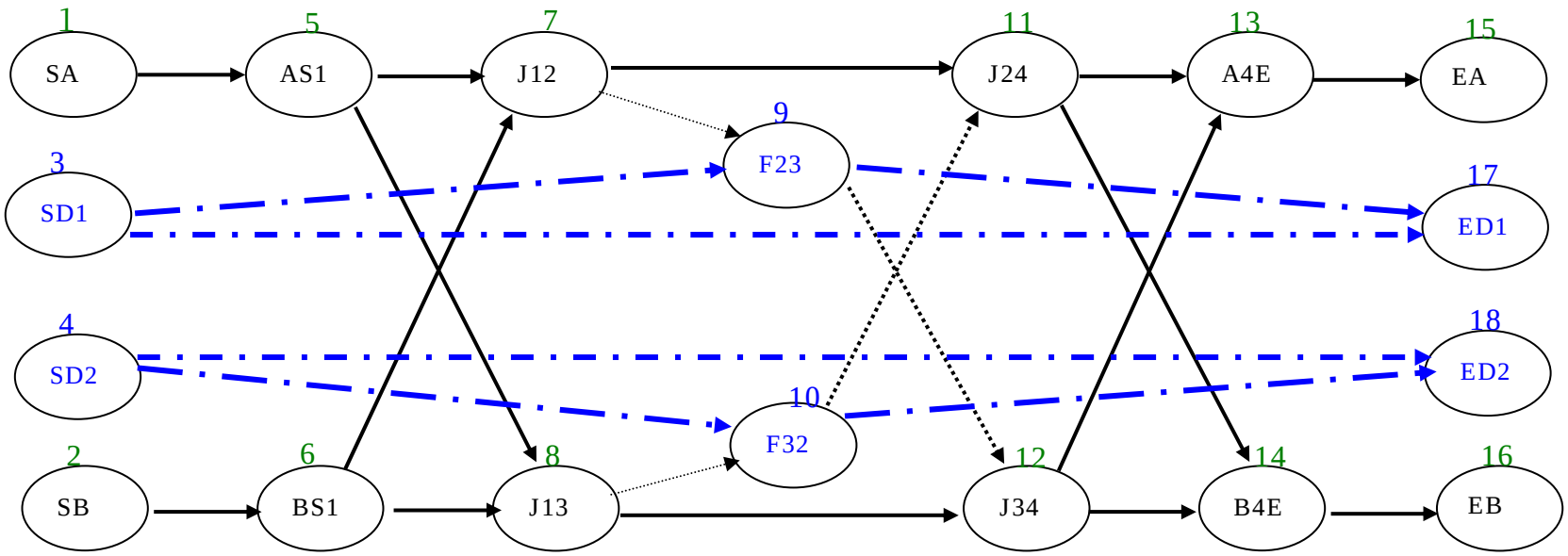


・使用可能な作業者の集合Kは、{A、B} の2人である。
・{F23,F32} は、補助作業で不要なら行う必要はない。
・各仕事には、作業者毎に、開始可能な時間帯の制約がある。

Aの作業計画：SAからEAへの許容な道で表現：AS1->J12->F23->J34->A4E
Bの作業計画：SBからEBへの許容な道で表現：BS1->J13->F32->J24->B4E
→ **M**の全ての枝が一度だけカバーされていれば実行可能な計画となる。

制約プログラミングによる定式化

- ネットワーク表現(ダミー・ノードの利用)



Suc[s]：ノードsに隣接した後続ノードの集合　-> Suc[7] = { 9,11}
　　P：作業者の集合　->　P = { 1,2,3,4}　但し　3,4はダミー

SA～EA、SB～EB、SD1～ED1、SD2～ED2の道で、全てのノードを1回だけカバーすれば、実行可能解となる。

Math-Model Research Inc.

制約条件の定義

- **変数の定義**：
  - $X[s]$：番号ｓの仕事を行った作業者が次ぎに行う仕事の番号
  - $T[s]$：番号ｓの仕事の開始時刻
  - $Y[s]$：番号ｓの仕事を行った作業者の番号
- **制約条件の定義**：Suc[s] は「ｓの直後に可能な仕事」の集合、Pは作業者の集合

$X[s] \in \mathrm{Suc}[s]$	$s \in N$	許された後続作業の選択
$Y[s] \in P$	$s \in N$	作業者は、限定
$Y[X[s]] = Y[s]$	$s \in N$	続行作業は、同じ人
$T[X[s]] \geqq T[s] + d[s]$	$s \in N$	次ぎの開始時刻の計算
$X[s] \neq X[u]$	$s \neq u \quad s,u \in N$	直前の仕事は、1つだけ
$a_s^{X[s]} \leqq T[s] \leqq b_s^{X[s]}$		開始時刻の制約
$X[15]=1, X[16]=2, X[17]=3, X[18]=4$		開始と終了の一致

非常にコンパクトな表現が可能

Math-Model Research Inc.

大域的制約の利用

1）全ての変数の値が異なる。

局所的制約条件： $X[s] \neq X[u] \quad s \neq u \quad s,u \in \mathrm{N}$

大域的制約条件： IloAllDiff（X）

2）全てのノードが、時間制約を満たすパスでカバーされる

局所的制約条件： $Y[s] \quad \in \quad \mathrm{P} \quad (|\mathrm{P}| = 4)$

局所的制約条件： $X[s] \quad \in \quad \mathrm{Suc}[s] \quad s \in \mathrm{N}$

局所的制約条件： $X[s] \neq X[u] \quad s \neq u \quad s,u \in \mathrm{N}$

局所的制約条件： $T[X[s]] \geqq T[s] + d[s] \quad s \in \mathrm{N}$

大域的制約条件： IloPathLength（$X, T, 4,$ ・・・）

CP成功の鍵：問題の本質的な
大域的制約が利用可能。

Math-Model Research Inc.

宣言的なアプローチ(2):LP

- 線形計画法(リニアプログラミング:LP)
  - 基本の定式化

目的関数: $c_1X_1 + c_2X_2 + c_3X_3 \rightarrow Max$

制約条件: $a_{11}X_1+a_{12}X_2+a_{13}X_3 <= b_1$

$a_{21}X_1+a_{22}X_2+a_{23}X_3 <= b_2$

$a_{31}X_1+a_{32}X_2+a_{33}X_3 <= b_3$

$X_1>= 0 \quad X_2>=0 \quad X_3>=0$

☆大規模な問題も解ける(CPLEX,XPRESS-MP,SOPT,NUOPT等)

  - (混合)整数線形計画法

整数制約: X_1, X_2, X_3 (の一部)は、整数のみ可能

★簡単には解けない。Branch & Bound法

整数計画による4－Queenの定式化

4－Queen問題：縦、横、斜め方向にQueenが重ならない配置を求める。

	A	B	C	D
1			Q	
2	Q			
3				Q
4		Q		

$x_{13} = x_{21} = x_{34} = x_{42} = 1$

その他の $x_{ij} = 0$

定式化：各行のQueenの位置を示す論理変数 x_{ij}を用いる

変数： $x_{ij} = 1$ if i行目のQueenはj列にある時

$x_{ij} = 0$ if i行目のQueenはj列にない時

制約条件：

1) **横**：$x_{11}+x_{12}+x_{13}+x_{14} = 1$

……

2) **縦**：$x_{11}+x_{21}+x_{31}+x_{41} = 1$

……

3) **右下**：$x_{11}+x_{22}+x_{33}+x_{44} \leqq 1$

$x_{21}+x_{32}+x_{43} \leqq 1$

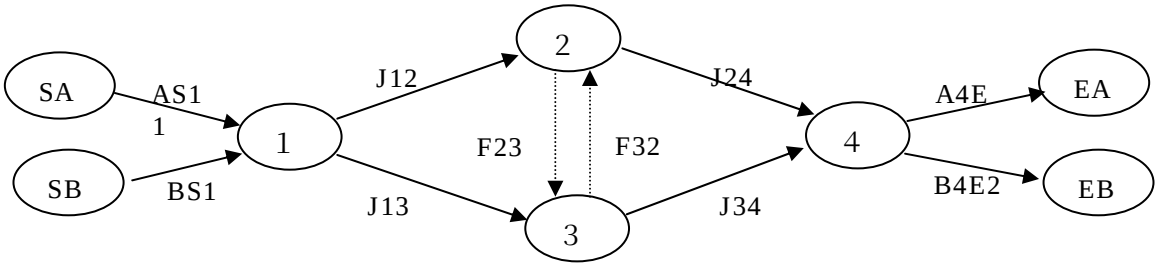
……

2) **右上**：$x_{41}+x_{32}+x_{23}+x_{14} \leqq 1$

$x_{31}+x_{22}+x_{13} \leqq 1$

……

ＬＰの集合分割問題の基本的な考え方



右側の行列で全ての行に1回だけ1が現れるように、複数の列を選べば実行可能解

道の列挙

P11: AS1->J12->J24->A4E

P12: AS1->J12->F23->J34->A4E

P13: AS1->J13->F32->J24->A4E

P14: AS1->J13->J34->A4E

P21: BS1->J12->J24->B4E

P22: BS1->J12->F23->J34->B4E

P23: BS1->J13->F32->J24->B4E

P24: BS1->J13->J34->B4E

	P11	P12	P13	P14	P21	P22	P23	P24			
AS1	1	1	1	1	0	0	0	0		Z1	1
BS1	0	0	0	0	1	1	1	1		Z2	1
J12	1	1	0	0	1	1	0	0		Z3	1
J13	0	0	1	1	0	0	1	1	=	Z4	1
J24	1	0	1	0	1	0	1	0		Z5	1
J34	0	1	0	1	0	1	0	1		Z6	1
A4E	1	1	1	1	0	0	0	0		Z7	1
B4E	0	0	0	0	1	1	1	1		Z8	1

Z1=0,Z2=1,Z3=0,Z4=0,Z5=0,Z6=0,Z7=1,Z8=0

Math-Model Research Inc.

線形計画法による定式化（集合分割問題）

- 作業者毎に実行可能な仕事の順序を複数作成する。
  - s(k)からe(k)への道として表現される。
  - 変数 Y_p^k ：作業者kのp番目の道が選ばれたとき、1とする。

$$Min \sum_k \sum_{p \in P(k)} C'_p * Y_p^k$$

$$\sum_k \sum_{p \in P(k)} R_{ij,p} Y_p^k = 1 \qquad (i, j) \in M$$

枝(i,j)を1回だけカバー

$$\sum_{p \in P(k)} Y_p^k = 1 \qquad k \in K$$

kの道は、1回だけ選択

$$Y_p^k = \{0,1\} \qquad k \in K, p \in P(k)$$

$[\ 0 \le Y_p^k \le 1$ ：LP 緩和問題

但し $C'_p = \sum_{(i,j) \in E(p)} C_{ij}$

道に対応したコスト

- $P(k)$：$s(k)$～ $e(k)$のパスの集合
- $E(p)$：p番目のパスの枝の集合
- $R_{ij,p} = 1 \quad (i, j) \in E(p)$
  $= 0 \quad (i, j) \notin E(p)$

$R_{ij,p}$は、枝(i、j)が道p上に有る時、1となる。

Math-Model Research Inc.

ネットワーク フローによる定式化

多種流問題として、ネットワークの均衡条件を記述する。

変数X_{ij}^{k}：仕事(i,j)が、作業者kで実施されるとき、1となる。

変数T_i^k ：作業者k の仕事(i,*)の開始時刻

$$Min \quad \sum_{k} \sum_{ij} C_{ij} * X_{ij}^{k}$$

$$\sum_{k} X_{ij}^{k} = 1 \qquad (i, j) \in M$$

(i,j)が1回だけ1となる。

$$\sum_{(s(k), j) \in M} X_{s(k) j}^{k} = 1 \qquad k \in K, s(k) : k\text{の 開始ノード}$$

kが1回だけスタート

$$\sum_{(i, j) \in M} X_{ij}^{k} - \sum_{(j, i) \in M} X_{ji}^{k} = 0 \qquad k \in K, j \in N - \{s(k), e(k)\}$$

kの流れの均衡条件

$$\sum_{(i, e(k)) \in M} X_{ie(k)}^{k} = 1 \qquad k \in K, e(k) : k\text{の 終了ノード}$$

kが1回だけ終了

$$X_{ij}^{k} (T_i^{k} + t_{ij} - T_j^{k}) \leq 0 \qquad k \in K, (i, j) \in E$$

$$a_i^{k} \leq T_i^{k} \leq b_i^{k} \qquad k \in K, i \in N$$

kのiの開始時刻の制約

$$X_{ij}^{k} = \{0,1\} \qquad k \in K, (i, j) \in E$$

Column Generation法

基本的なアルゴリズム

- 1）パスの部分集合Pを与えて、集合分割問題のLP緩和を解き、対応する双対変数（シャドープライス）β_{ij}、γ_kを求める。
- 2） 各人(k)毎に下記のコストによる時刻制約を満たす最短経路を求め、コストが負で絶対値が最大のパスをPに加え、1）へ戻る。

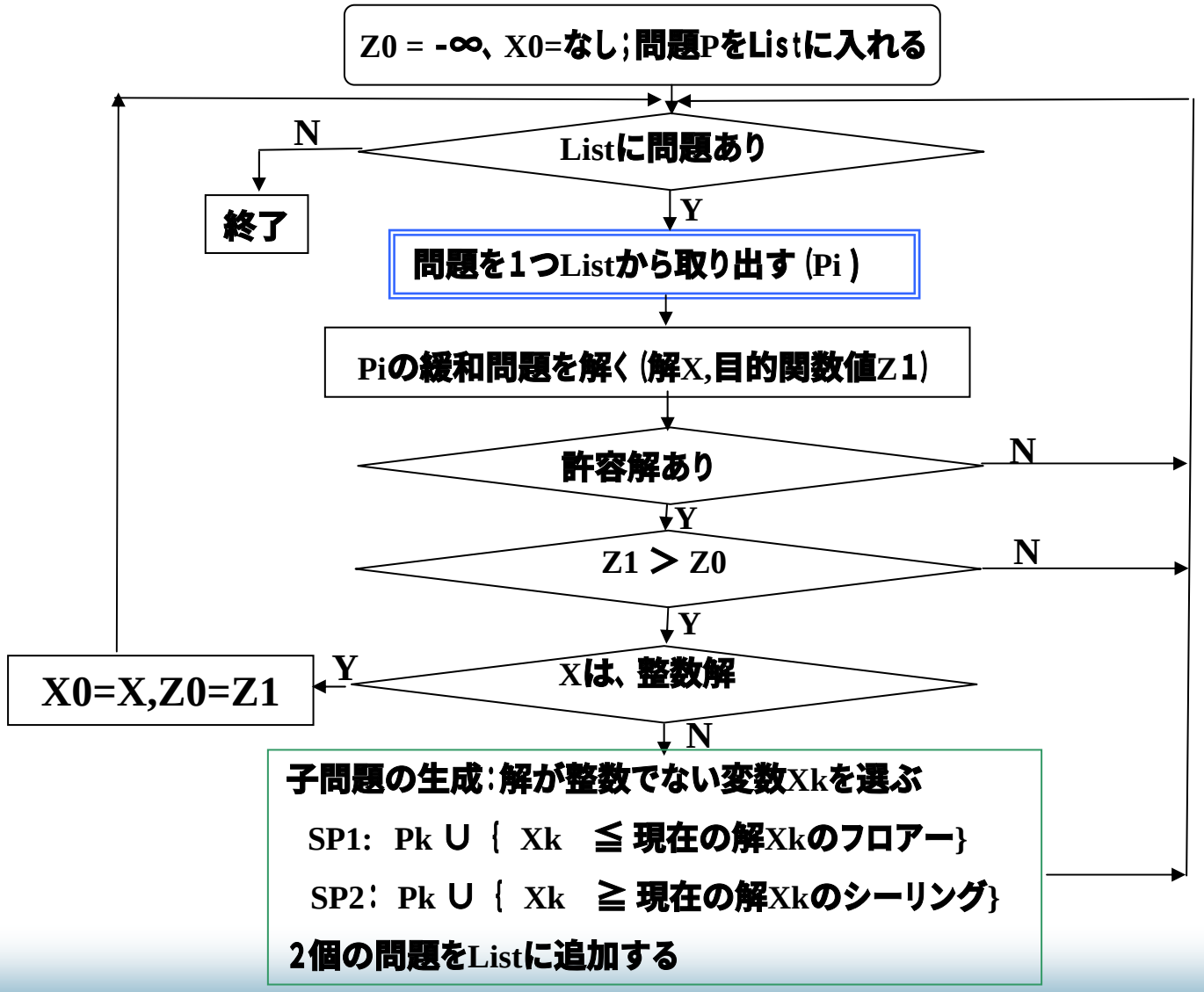
$$
\begin{aligned}
C_{ij}^{k} &= C_{ij} - \beta_{ij} && (i,j) \in M \\
C_{s(k)j}^{k} &= C_{s(k)j} - \gamma_k && s(k): k\text{の}start\ node \\
C_{ij}^{k} &= C_{ij} && \text{上記以外}
\end{aligned}
$$

この部分で、多種流問題の定式化を利用

- 3）コストが負のパスが存在しなければ、LPは最適解である。
- 4）Y_p^kが全て0，1にならない場合には、Branch and Bound法で整数解を探索する（ Branch and Cut法、Branch and Price法が強力 ）。

- 実用規模の多くの問題が解かれている。
- 計算規模が膨大になる傾向がある。

Branch and Bound法 (標準)



基本的なアルゴリズム

Branch & Cut法：Cutを利用した領域の削除

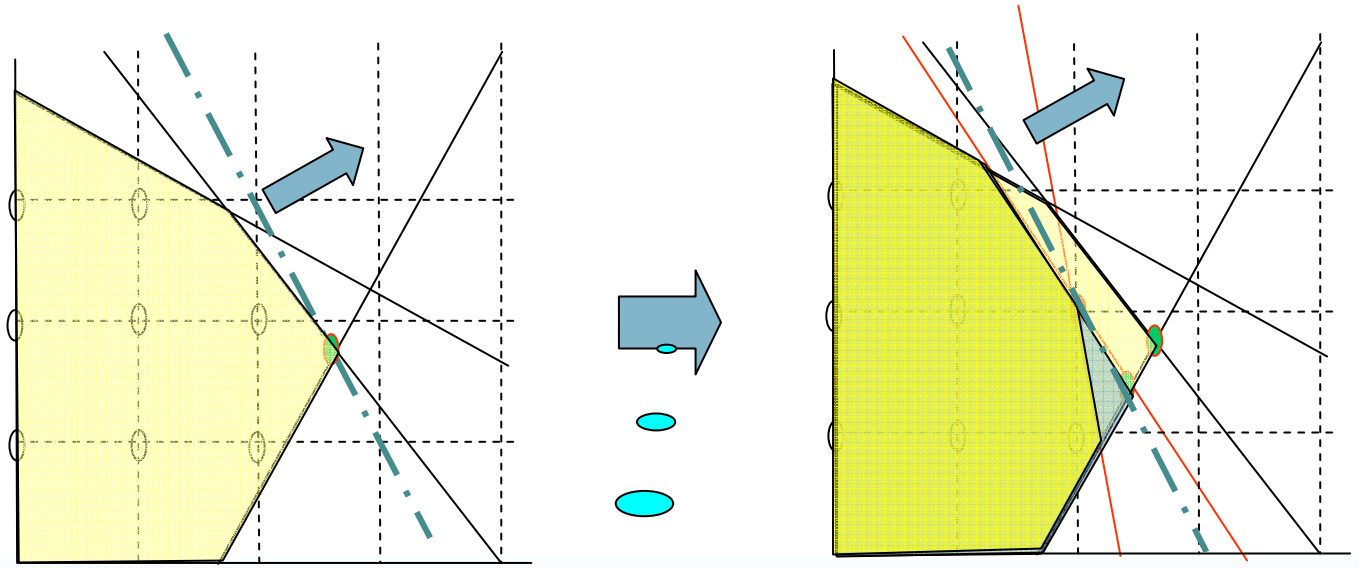
Cutの例：x1,x2,x3は、非負の整数

Gomory Cut: x1 + 2.2x2 + 1.8x3 ≦ 2.9

x1 + 2x2 + x3 ≦ 2

Clique Cut: x1 + x2 ≦ 1, x2 + x3 ≦ 1, x1 + x3 ≦ 1

x1 + x2 + x3 ≦ 1



出来るだけ
整数解に！

ハイブリッド・アプローチ (CP＋LP)

制約プログラミング (CP):

長所: 複雑な論理的制約条件を表現可能

短所: 大域的な評価 (最適解、要素の感度) が困難

線形計画法 (LP):

長所: 大域的な評価 (最適解、要素の感度) が可能

短所: 利用可能な制約条件が限定

CPとLPの長所を利用

緩和問題:

CPの制約条件を緩和した**LP**問題の利用

ハイブリッド・アプローチ:

1) **CP**により厳密に問題を定式化する

2) **LP**緩和問題を解いて、上限(下限) 値、感度情報を得る

3) 上限 (下限) 値、感度情報により、**CP**を高速に解く

ハイブリッド・アプローチのイメージ

制約プログラミング (CP):

目的: $F = aX_1 + bX_2 + \cdots$ -> 最大

変数: X_1、X_2、$\cdots$

条件:

C1: $cX_1 + dX_2 + \cdots \leqq b_1$

C2: $\{X_1、X_2、X_3、X_4\}$ の非ゼロは2個以内

線形計画 (LP):

目的: $G = aX_1 + bX_2 + \cdots$ -> 最大

変数: X_1、X_2、$\cdots$

条件:

C1: $cX_1 + dX_2 + \cdots \leqq b_1$

C2: 削除

CPの解探索:

現在までの最も良い許容解 (F_0)

緩和情報を利用して、
一部の変数を設定 (部分解)

現在の部分解の評価:
$G_0 \leqq F_0$ => バックトラック

探索情報:
1) 厳しい制約を先に扱う
2) 初期値としてY_0近傍を利用

LPの解探索:

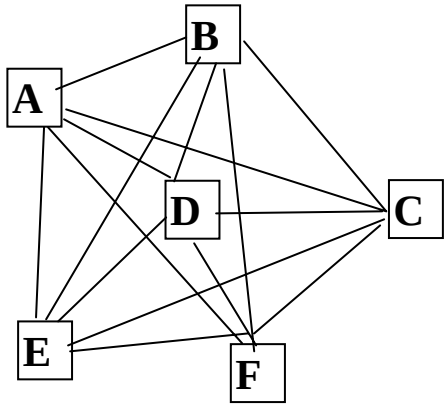
(CPで設定された値は、固定)

1) 目的関数: G_0
2) 最適化を達成するXの値 (Y_0)
3) 制約条件のシャドープライス

Math-Model Research Inc.

遺伝子アルゴリズム(GA)の特徴

問題の解を遺伝子の形で表現
例：巡回配送問題（基地はA）
ルート1： 順路 (A,E,D,F,C,B,A) コスト：○○円
ルート2： 順路 (A,C,B,E,F,D,A) コスト：××円



1) 初期集団の生成

P0 ＝ {(A,E,D,F,C,B,A), (A,C,B,E,F,D,A) , (A,D,C,F,B,E,A),・・・・}

2) 新規集団の候補の生成： Pkからランダムに遺伝子を選択

・突然変異

(A,E,D,F,C,B,A) -> (A,C,D,F,E,B,A)

・交差（サイクル）

(A,E,D,F,C,B,A) -> (A,E,D,F,C,B,A)

(A,D,C,F,B,E,A) -> (A,D,C,F,E,B,A)

これが簡単に
出来るか？

3) 新しい集団の生成

・既存の集団と新規集団の候補から、新しい集団 (Pk_{+1})を形成

4) 集団Pk_{+1}の評価

・十分収束していれば、終了

・ステップ2) へ

シミュレーテッド・アニーリング（SA)の特徴

問題の解を順路の形で表現
例：巡回配送問題（基地はA)
ルート1：順路（A,E,D,F,C,B,A) コスト：○○円

1）初期解の生成：温度 T0、繰返回数 M、冷却率 αの設定

S0 ＝（A,E,D,F,C,B,A)； S_cur ＝ S_best ＝ S0

2）温度のコントロール

・Time ≧ Time_Limitなら終了

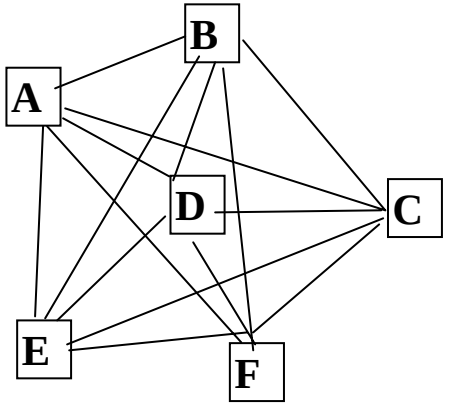
T0 ＝ αT0

3）近傍の探索(M回)： S_curの近傍解（S'）を1つ選ぶ

```
if( Cost(S') < Cost(S_cur) ){
  S_cur = S'
  if( Cost(S') < Cost(S_best)) { S_best = S'}
 }else{
   if( Random < exp( -(Cost(S_cur) – Cost(S'))/T0) ){ S_cur = S'}
   }
 }
```

4） Time ＝ Time + M

ステップ2へ



これが簡単に出来るか

悪くてもある確率で選択

Math-Model Research Inc.

最適化のアルゴリズム

探索方式	アルゴリズム	特徴
逐次的探索	ローカルサーチ メタ･ヒューリスティックス (タブ･サーチ、ガイディッド･サーチ) エキスパート･システム	･色々な問題に適用可能で、比較的簡単 ･最後の方になってダメと判明する！) ･探索空間が大きくなり、時間がかかる。 ･局所的な解に陥りやすい。
数理的探索	数理計画法 (LP、QP、NLP) ネットワーク理論 混合整数計画 (Branch and Bound 法) Column Generation 法	･方程式系により大域的な実行可能性を保証 ･探索空間の縮小が可能 ･適用可能な制約条件に制限がある。 ･専門的な知識による抽象的な定式化が必要 ･大規模な問題に適用可能
制約充足的探索	制約論理 CHIP (COSYTEC) 制約ツールキット SOLVER (ILOG)	･大域的制約条件の導入と制約伝播による実行可能性の確保 ･制約伝播による探索空間の縮小 ･複雑な制約条件に対応可能 ･多少の専門知識で、具象的な定式化が可能 ･大規模な問題には、適用が難しい。
確率的探索	遺伝子アルゴリズム シミュレーティッド･アニーリング	･確率的に多くの解を生成して、良いものを求める。 ･実行可能解が沢山あり、簡単に求まる場合に適用が可能 ･遺伝子の定義が難しい。 ･最適化の達成度が判らない。

最適化ソフトウェアの総合情報サイト

http://www.ece.nwu.edu/OTC/

松井知己 (東大)： 最適化ソフトウェアとテスト問題集

http://www.misojiro.t.u-tokyo.ac.jp/~tomomi/opt-code.html

Math-Model Research Inc.

最適化アルゴリズムの適用基準

	逐次的		**確率的**		MIP		ColGen		CP		CP+LP	
変数の値域	**小**	**大**	**小**	**大**	**小**	**大**	**小**	**大**	**小**	**大**	**小**	**大**
許容解が簡単に得られる												
線形制約	○		○		○	○			○			
論理制約	○		○		△				○			
許容解を得るのが困難												
大域的な制約はない												
複雑な論理制約なし	○		○		○	○			○			○
複雑な論理制約あり	○				△		○	○	○		○	○
大域的な制約がある												
複雑な論理制約なし	△		△		○	○			○		○	○
複雑な論理制約あり	△				△		○	○	○		○	○

(CP+ColGen)

最適化アルゴリズムの適用性

Problem category	Do's	Don't's
Set covering, set partitioning including possibly some additional knapsack constraints Set selection (coverage)	Mixed Integer Programming Column generation, for very combinatorial problems, allows an efficient technical decomposition CP with partial search (LDS) local optimisation	Constraint propagation with depth-first search
Scheduling Disjunctive scheduling (single tasks) Scheduling with complex resources and tasks	B&B + constraint propagation using literature shaving tabu heuristics Read Demeulemeester' thesis [Demeulemeester 92], B & B + resource histograms + partial search	MIP, genetic algorithms, simulated annealing
Production scheduling Big size (multi-product, multi-machines), linear processes Short range problem, including fine set up time, non-linear routes (job-shop problems), small size	MIP can bring satisfactory solutions, combined sometimes with heuristics for post-processing. CP + heuristics. If big size, decompose problems	Propagation on disjunctive constraints MIP
Time-tabling Fixed set of activities Variable schedules	Global constraints, flow algorithms. LP model, local optimisation LP, tabu, coverage techniques, Column generation, CP for schedules with global analysis	MIP Propagation on local constraints

出典：The Chic-2 Methodology for Combinatorial Applications

Staff scheduling Shifts construction: Roster grids construction:	MIP with set covering/partitioning approach. CP and heuristics. Decompose and use hybrid approaches. Column generation.	Don't try to solve the whole problem in one round with one approach.
Matching Assignment, resource allocation	Use adhoc algorithm (matching flows) or LP (simpler), search for good model relaxation => lower bounds and local optimisation	Constraint propagation
Assignment problems Big and linear : Small and non linear:	MIP CP or hybrid	
Routing problems Travelling Salesman Problem	CP for small complex problems only, Local optimisation by default, Branch & Cut is an option	Don't build ad-hoc solutions - there are many published algorithms which work well
Problem category	**Do's**	**Don't's**
Vehicle routing problems	Local optimisation	B&B
Tour problems	Column generation MIP approaches proved to optimally solve such problems. Heuristics, although sub-optimal, proved to solve them very efficiently	If max duration constraints, don't use network flow approach with LP
Flow problems Network optimisation Optimisation problems with Boolean choices (multi-periodic, multi-products with inventories and intermediate processes)	LP models + flows + global analysis Good heuristics & local optimisation MIP	Pure CP, genetic algorithms,
Melting problems	Pure LP (MIP)	Others
Cutting problems Bin Packing	1 dimension: MIP can be OK. Global constraint propagation Several dimensions (2 or 3): simulated annealing proved to work in some cases.	MIP for more than 2 dimension problems

出典：The Chic-2 Methodology for Combinatorial Applications

業務と問題のカテゴリー

Field	Application	Category	Classification
Airline industry	Crew scheduling Fleet scheduling Shift planning	Set covering Set partitioning Network flow Assignment	*var:* Integer *constr:* Linear *data:* Deterministic
Production manufacturing	Cutting stock	Knapsack Bin packing	*var:* Mixed integer *constr:* Non-linear *data:* Deterministic
Production scheduling	Car sequencing Resource allocation Task scheduling	Resource constrained project scheduling	*var:* Integer *constr:* Non-linear *data:* Deterministic
Transportation industry	Facility location Vehicle routing Tour problems	Set covering TSP	*var:* Integer *constr:* Linear *data:* Deterministic
Telecommunication and network industry	Network flow Network optimisation Frequency allocation	Matching	*var:* Integer *constr:* Linear *data:* Stochastic
Wireless Telecommunication	Frequency allocation	Coloration, maximum independent set	*var:* Integer *constr:* Quadratic *data:* Deterministic
Personnel scheduling	Time-tabling Work force scheduling	Assignment Matching flow	*var:* Integer *constr:* Linear *data:* Deterministic
Finance	Portfolio optimisation	Knapsack	*var:* Continuous *constr:* Quadratic *data:* Stochastic

出典：The Chic-2 Methodology for Combinatorial Applications

まとめ（その2）

- 最適化技術：
  - 産学連携により、実際のデータによる最適化技術の評価が必要。
  - 〃　　、CPとLPのハイブリッド化のような新しい試みが必要。
  - 高性能の商用のパッケージが開発され、非常に安価になっているが、実務ベースで使える人材が不足している。
  - 企業に開かれた相談窓口が必要
- SCN環境：
  - CPFRのような試みは、現実のシステムを構築する上で非常重要だが、ORの研究者の枠外にあり、この面での研究が遅れている。
  - システム的視点から、統合モデルのような研究が必要
- コンサルティング：
  - コンサルティングの面でも、専門的な技術者は少なく、ソフトウェア開発者と大学等との連携による早期の育成が必要である。
  - 計画系システムに対する制約ベースの開発方式の普及が必要

ILOGの最適化ソフトウェアのデモ

Math-Model Research Inc.

ILOGの最適化ソフトウェア

1) 整数計画法:

CPLEX:LPのパッケージ

- 整数計画、混合整数計画、ネットワーク計画等

2) 制約プログラミング

Solver:基本となる制約モデル

- ドメイン変数の管理
- 基本となる制約条件
- 解の探索方式

Scheduler:スケジューリング問題に特化した制約

- スケジューリング問題に特化したオブジェクト
- リソースの制約