最適化技術の現状と計画系システムの開発

(株) 数理モデリング研究所
野末 尚次

ＵＲＬ：www.Math-Model.Co.JP

なぜ今最適化か

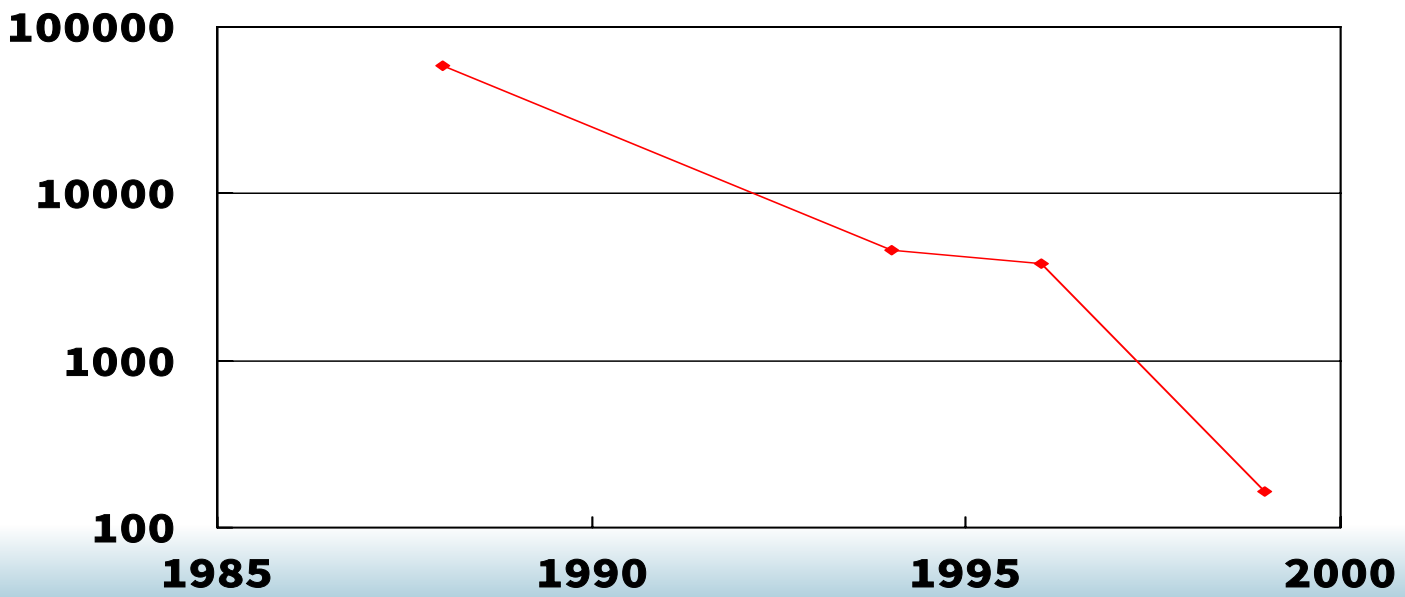
- 社会からの要請：顧客ニーズの多様化
  - 多品種少量、短いリードタイム、高い価格弾力性
- 企業のIT化の進展：企業内・企業間の統合化
  - ロジスティックスも含めた統合的な生産計画
  - データベースの完備／on lineの生産管理体制
- コンピュータの性能向上：高性能パソコンの出現
  - 驚異的な計算速度の達成
  - 大量のメモリー空間が利用可能
- 計画系ソフトウェアの進歩：商用ソフトウェアの出現
  - 組合せ問題に対する制約プログラミング等の出現
  - 線計画法（LP）の高速化と緩和アプローチの発達

LPソフトウェアの進歩：CPLEX（ILOG社）

- ＬＰのヴァージョン別実行速度の比較

  （CPLEX1.0～CPLEX6．0）

  - 問題規模：　４９９４４行，１７７６２８列　計算時間（秒）
  - 計算時間：　単位（秒），マシーンは同一



実用規模の計画問題の特徴

計画問題

機械・人間・材料・在庫・納期等の制約条件の下で、

- 制約条件を満たす計画案（実行可能案）を作成する。
- 実行可能案の中で、最も効率の良い案を求める。

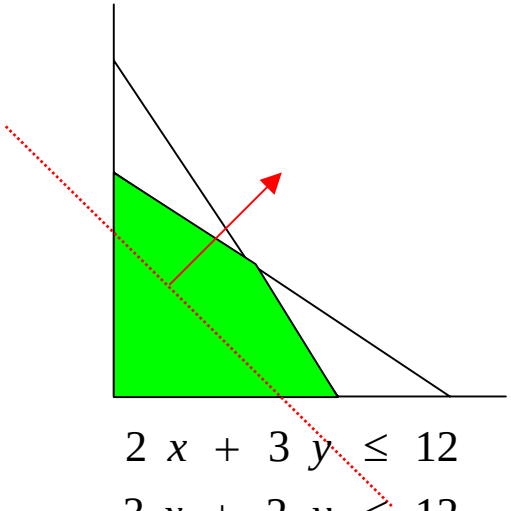
- どのようなタイプの制約条件があるか？
  - 絶対守らなければならない制約（ハードな制約）
  - ペナルティは有るが、緩和が可能な制約（ソフトな制約）
- 評価指標はなにか？
  - 実行可能解が得られれば良い（制約充足問題）。
  - 実行可能解の中から、（準）最適解を求める（最適化問題）。

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



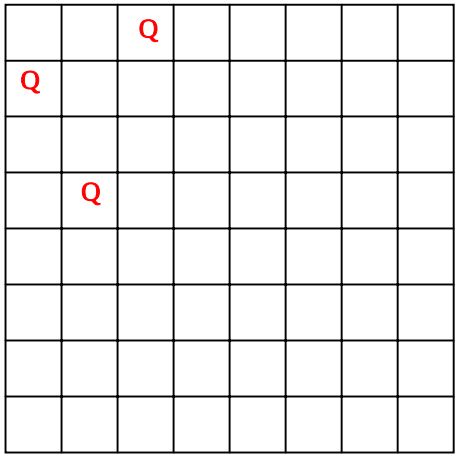
$$2x + 3y \leq 12$$

$$3x + 2y \leq 12$$

$$x \geq 0,\ y \geq 0$$

$$x + y \rightarrow Max$$

8-Queen配置問題



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

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

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

計画問題へのアプローチ　(開発手法)

- 計画問題に対する2つのアプローチ
  - 手続的なアプローチ (従来のプログラム)
  - 宣言的なアプローチ (制約の定義と解探索の分離)

つる･かめ算の計算

鶴と亀が足して5匹、足の合計が14本の時、鶴と亀の数は？

手続的アプローチ

1) 全部鶴と仮定すると、足は10本

2) 余った足の数　14－10＝4本

3) 亀の数　＝　4本／2　＝2匹

4) 鶴の数＝5－2＝3匹

宣言的アプローチ

1) 鶴と亀の数を　X、Yとする

2) X＋Y＝5　2X＋4Y＝14　(定義)

3) 連立方程式を解く　(探索)

4)　X＝3　Y＝2

計画問題へのアプローチ　(整合性の維持)

エキスパート･システムは、なぜ失敗したか

- 知識工学の新しい手法
  - 局所的なルールにより熟練者の知識を組織化する。

- 診断系のシステム開発で大きな成果

- 計画系のシステム開発に適用
  - 多くの制約がある実用規模の開発は、実質的に失敗

- 局所的な情報で有効と思われる方策も、
  トータルに見ると有害な場合がある。
  - 制約条件全体を考慮した方策が選択されていない

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/

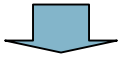
宣言的アプローチ(1):制約論理、制約プログラミング

- 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)

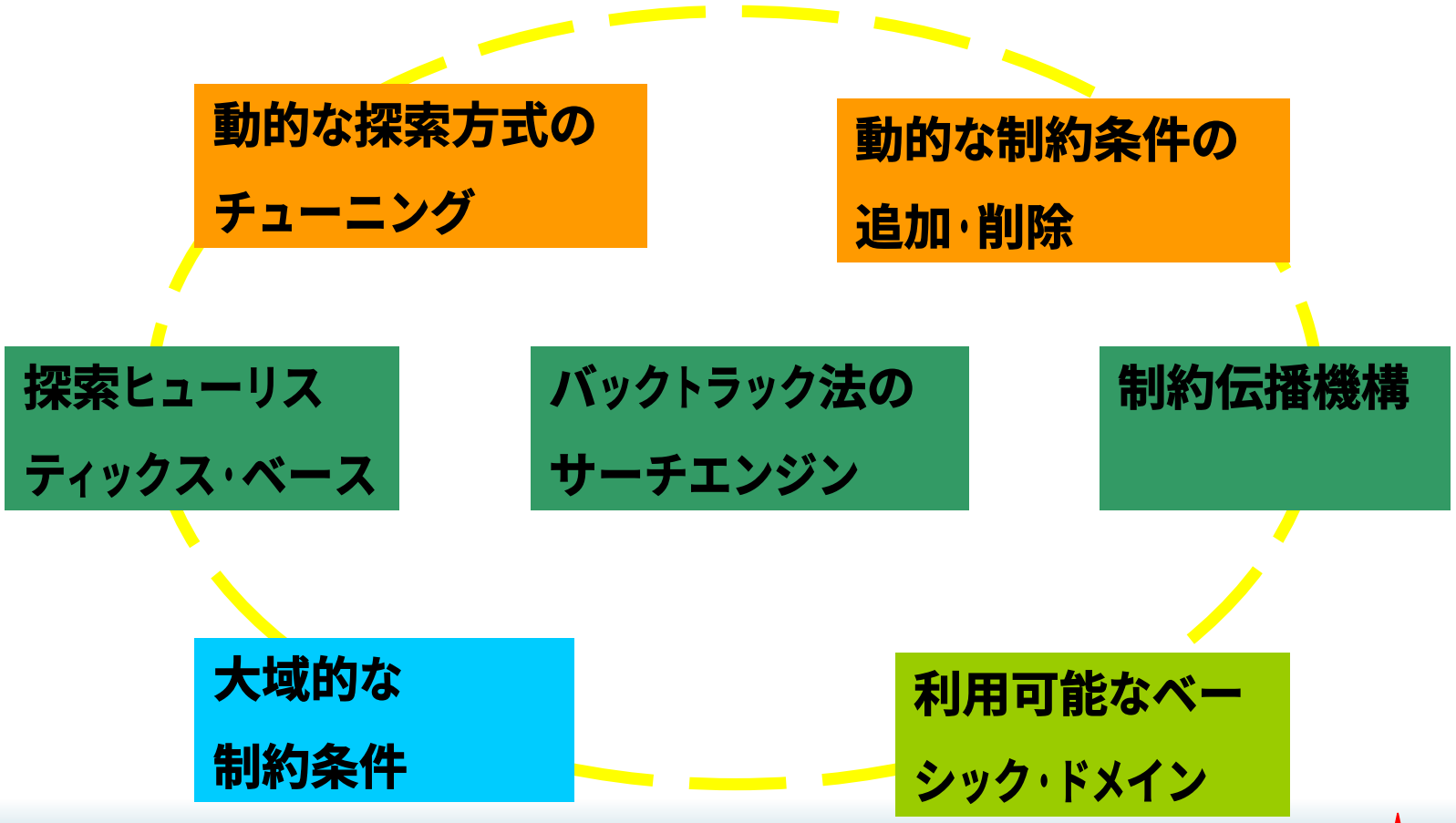
制約プログラミング入門

- **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**
  - 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）

制約プログラムの例 (部分)

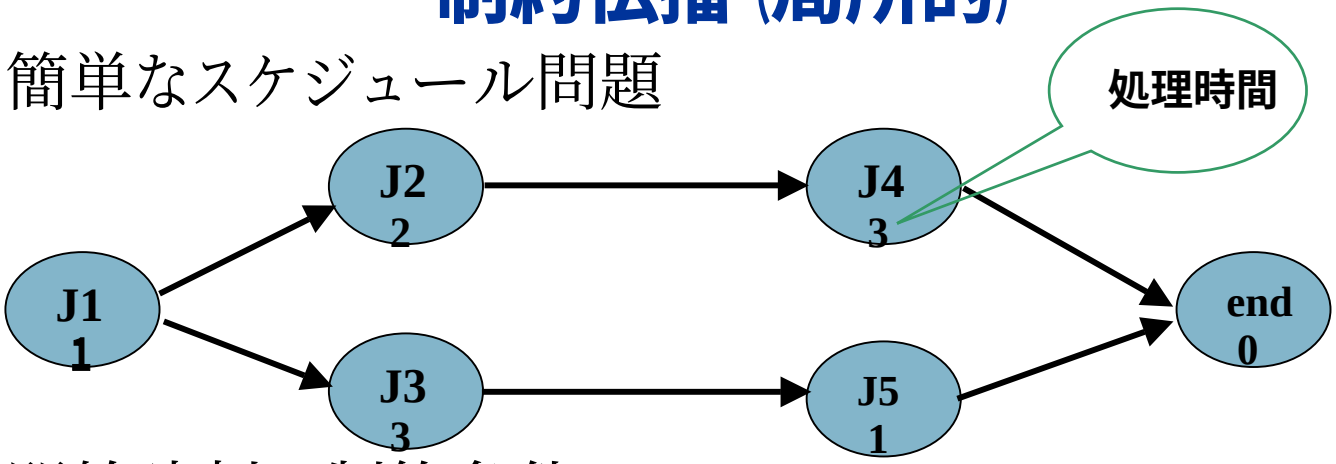
```
IlcManager m(IlcNoEdit);                                    マネージャの定義

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
m.add( IlcAllDiff(x1) );
m.add( IlcAllDiff(x2) );

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

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



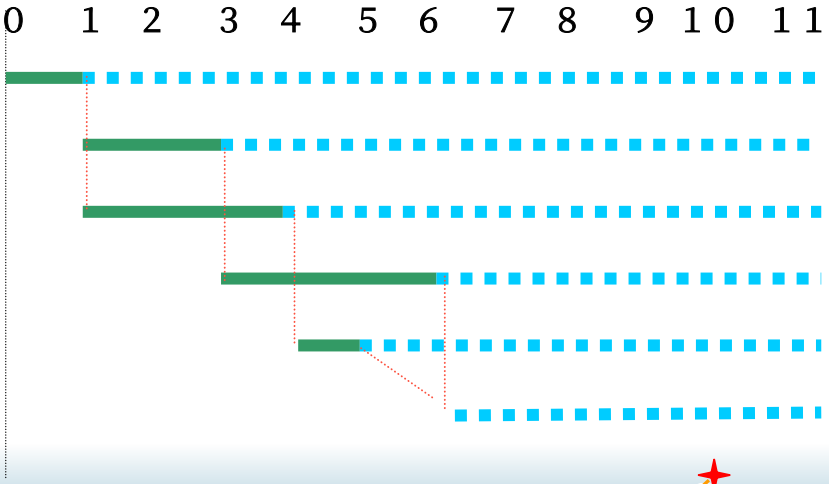
制約伝播 (局所的)

- 簡単なスケジュール問題



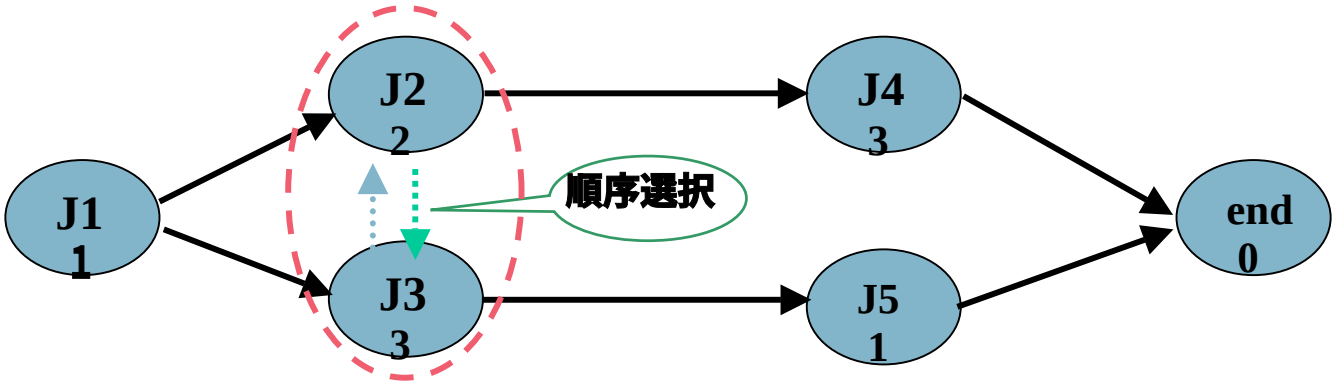
- 開始時刻の制約条件
  - **t1 ≧ 0**
  - **t2 ≧ t1 + 1**
  - **t3 ≧ t1 + 1**
  - **t4 ≧ t2 + 2**
  - **t5 ≧ t3 + 1**
  - **te ≧ t4 + 3**
  - **te ≧ t5 + 1**

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

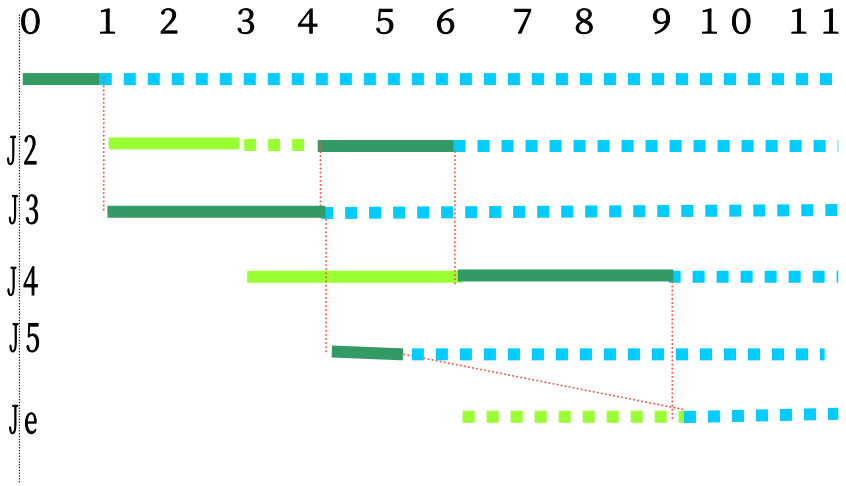
Math-Model Research Inc.

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**

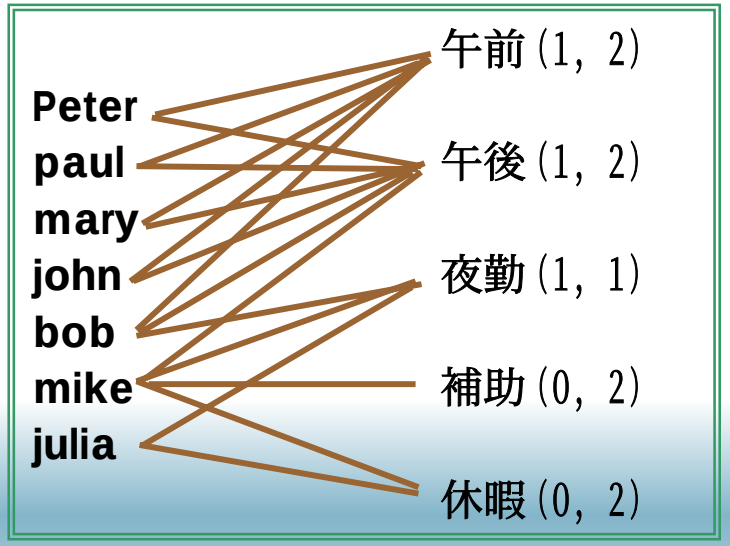
納期制約がある場合 （ **te ≦ 8** ）
もし制約伝播が完全なら、J3ー＞J2 は納期を満たさないから、
自動的に J2ー＞J3 が選択される。

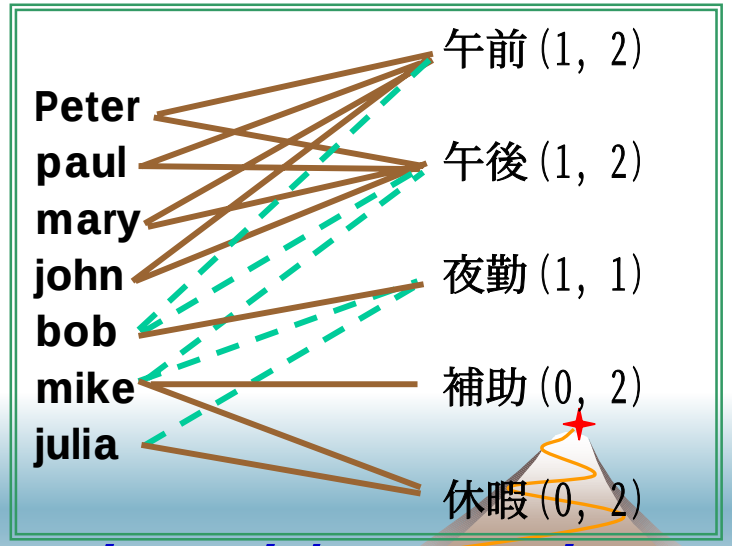
このレベルでの制約伝播は計算が大変

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）

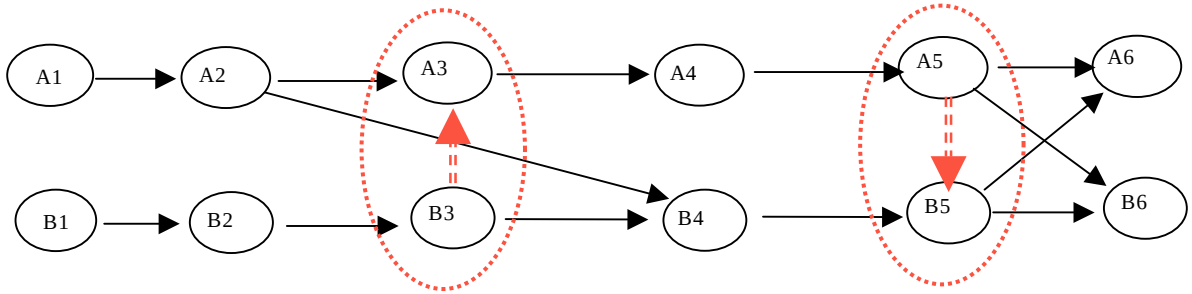
最大フロー問題を利用した制約伝播：　J.　Regin　(ILOG)



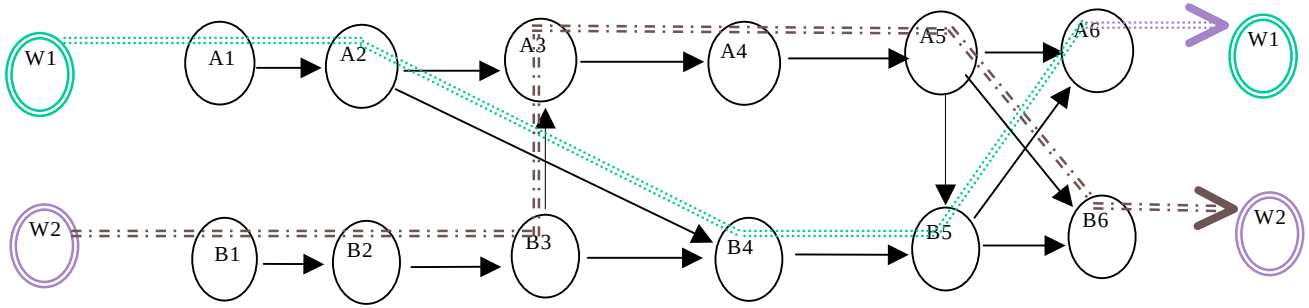


代表的な大域的制約条件とモデル

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

制約伝播の定義

- 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)
  - 関数の定義
    - ReduceX () {

      X.setMax (Y.getMax () − 1) ;

      }
    - ReduceY () {

      Y.setMin (__X.getMin () + 1) ;

      }
  - 実行時の伝播条件

    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

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)

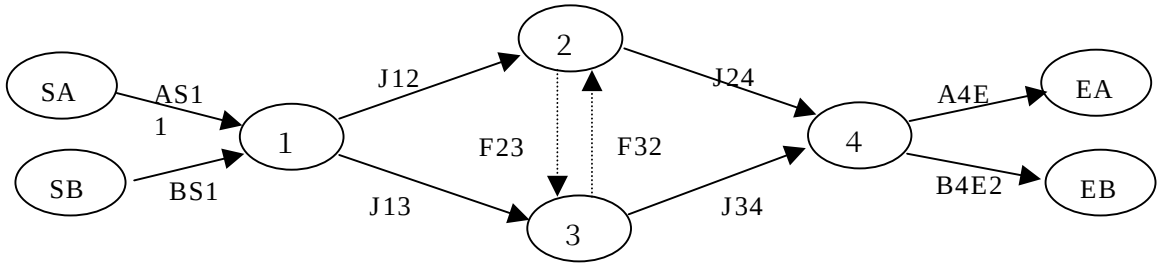
  - (混合)整数線形計画法

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

多種流問題によるスケジューリングの表現

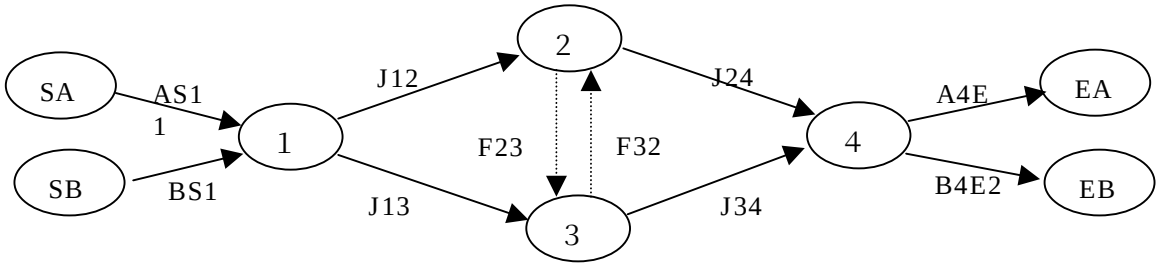
仕事の集合**M**は、{J12,J13,J24,J34} の4種類で、次の順序関係がある。



・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。

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

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
- $R_{ij,p} = 1 \qquad (i, j) \in E(p)$
  $= 0 \qquad (i, j) \notin E(p)$

$R_{ij,p}$は、枝(i、j)が道p上に有る時、1となる。

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。

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

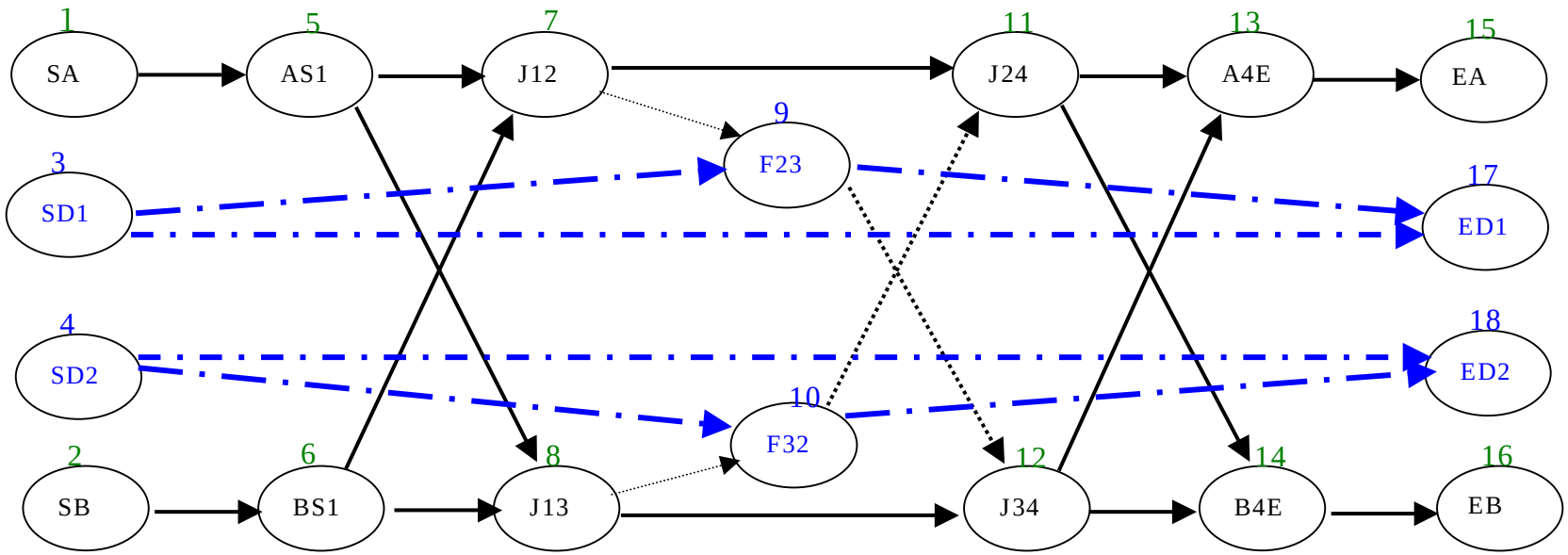
この部分で、多種流問題の定式化を利用

- 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。**

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。

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：

$X[s] \in \mathrm{Suc}[s]$	$s \in \mathrm{N}$	許された後続作業の選択
$Y[s] \in \mathrm{P}$	$s \in \mathrm{N}$	作業者は、限定
$Y[X[s]] = Y[s]$	$s \in \mathrm{N}$	続行作業は、同じ人
$T[X[s]] = T[s] + d[s]$	$s \in \mathrm{N}$	次ぎの開始時刻の計算
$X[s] \neq X[u]$	$s \neq u \quad s,u \in \mathrm{N}$	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

制約をベースとした計画系システムの開発

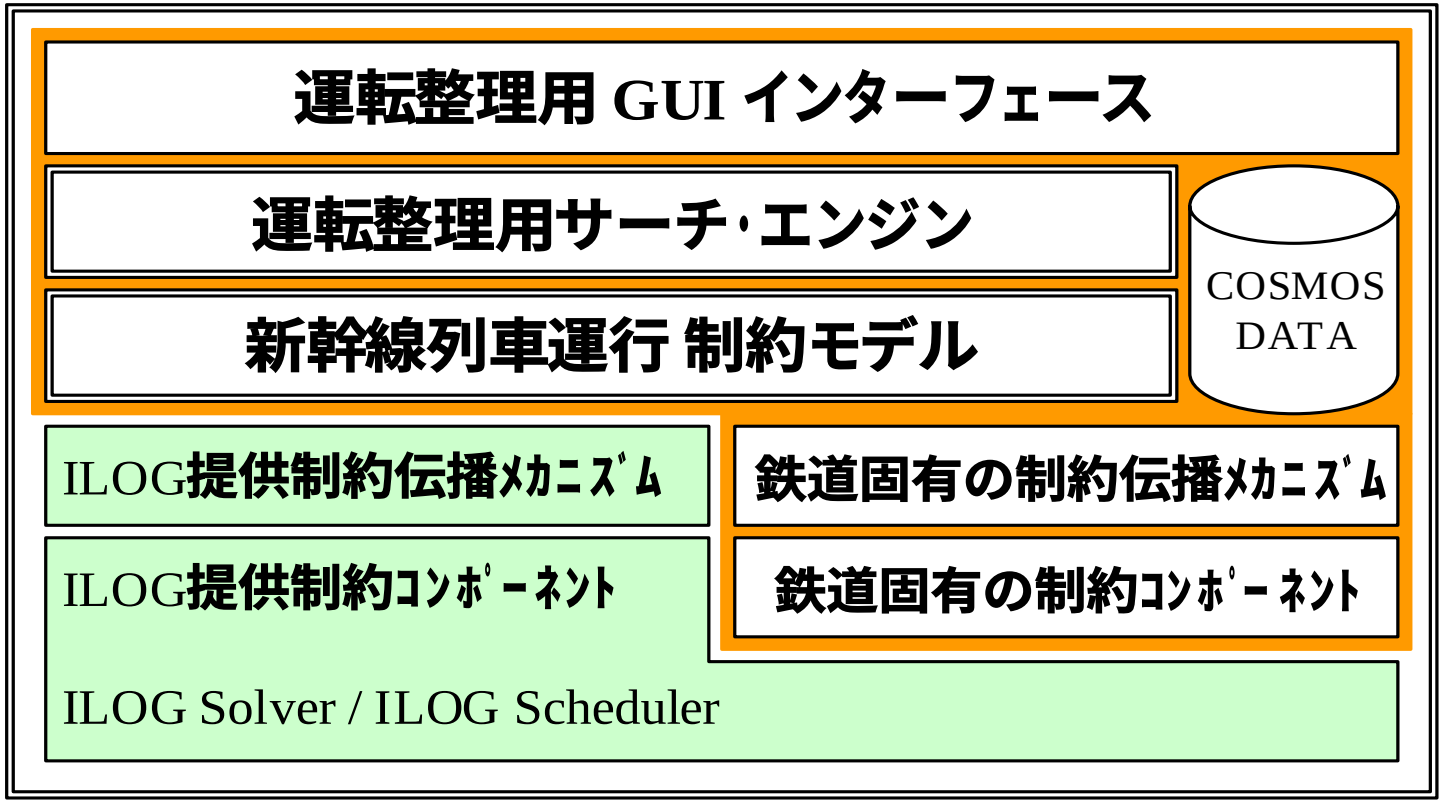
- 制約プログラミングと宣言的プログラミング
  - 最適化エンジンが主要な話題となるが、システム開発の面からは、宣言的なプログラミングが非常に重要である。
  - 制約論理自体が、宣言的な言語であるPrologの処理の非効率性の改善をターゲットに開発されてきた経緯を考えれば当然のことであるが、日本の現状では、この点が理解されていない。
- 新幹線の列車ダイヤの混乱を回復するための「新幹線運転整理プロトタイプ・システム　CIBERS」をILOGのSolver／Schedulerを用いて開発したが、この制約（＋オブジェクト）ベースのシステム開発法の有効性が実証された。

•制約の定義と探索法の分離

•制約ベース思考の革新性

新幹線運転整理プロトタイプ･システムの開発

CIBERSシステム構成図



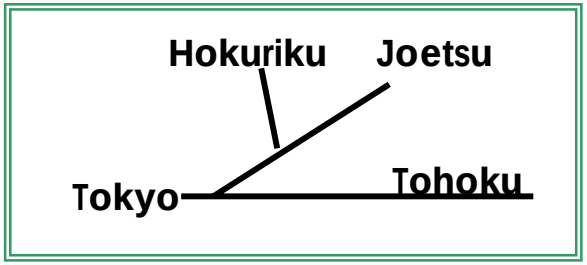
CIBERS

(株) JR東日本情報システム
(株) ニューメディア総研　　共同開発

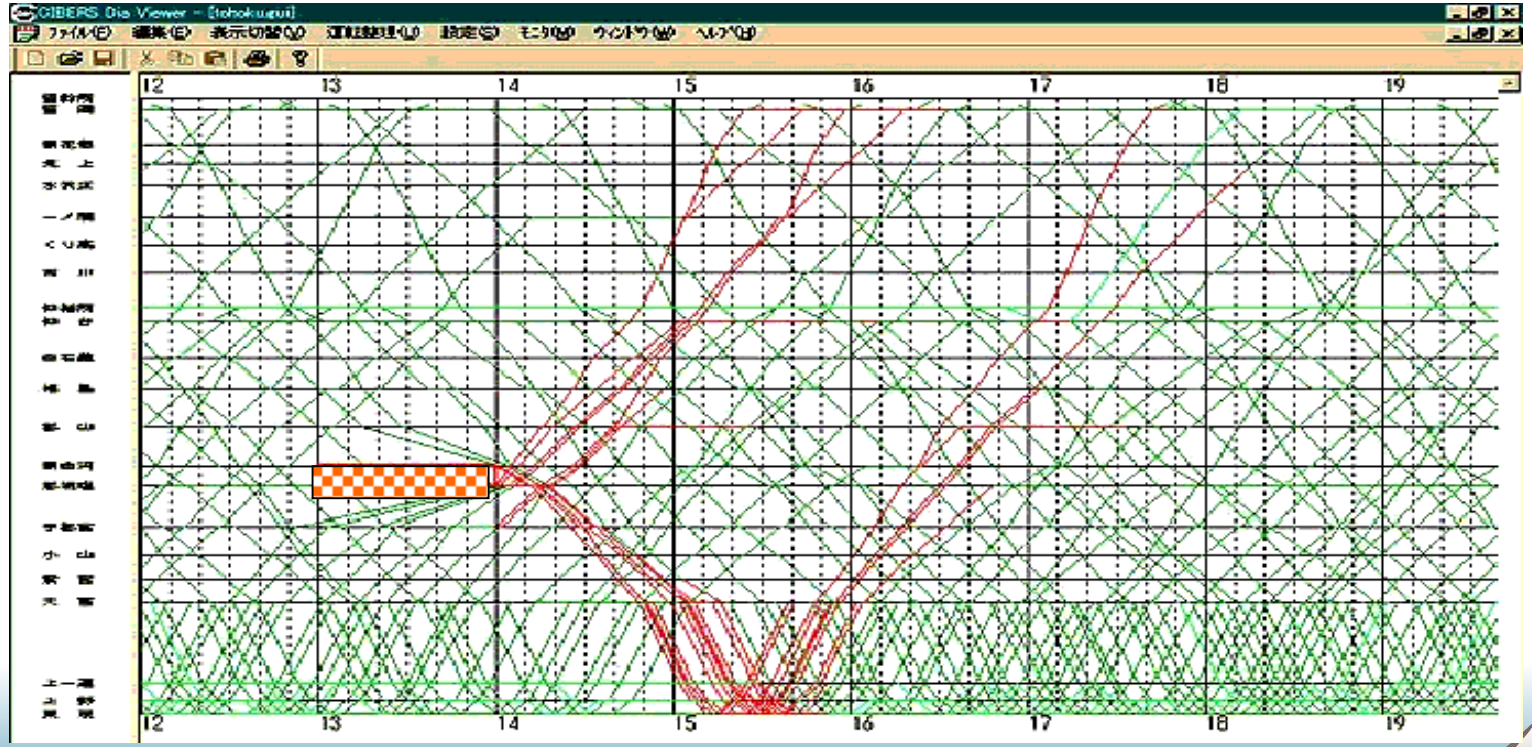
Math-Model Research Inc.

Case Study

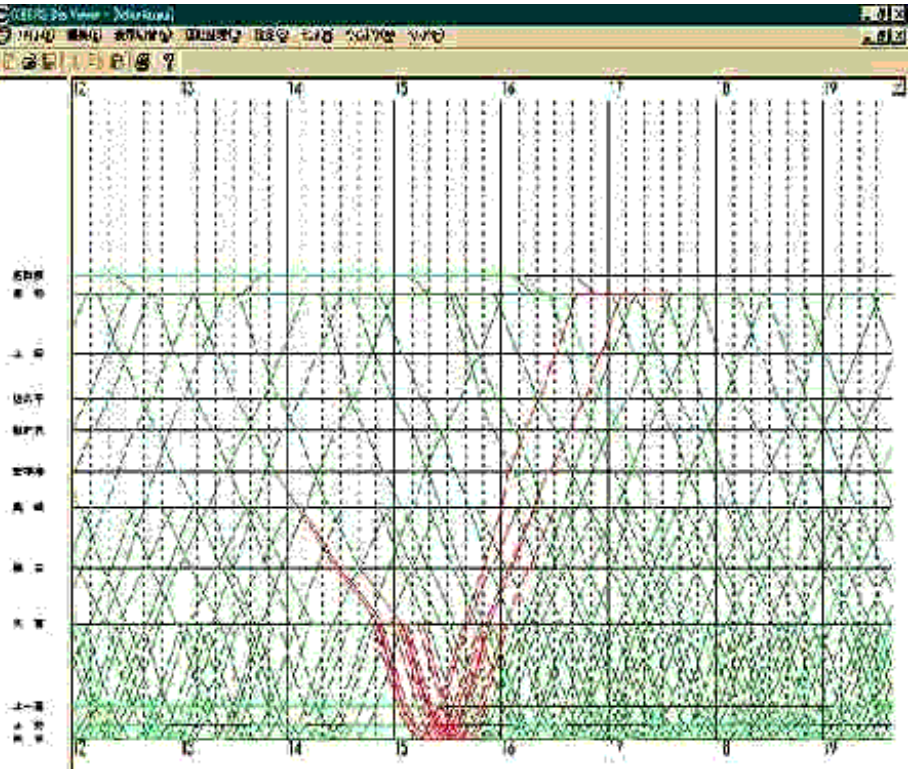
- JR-East Shinkansen Network
- 60 minutes suspension



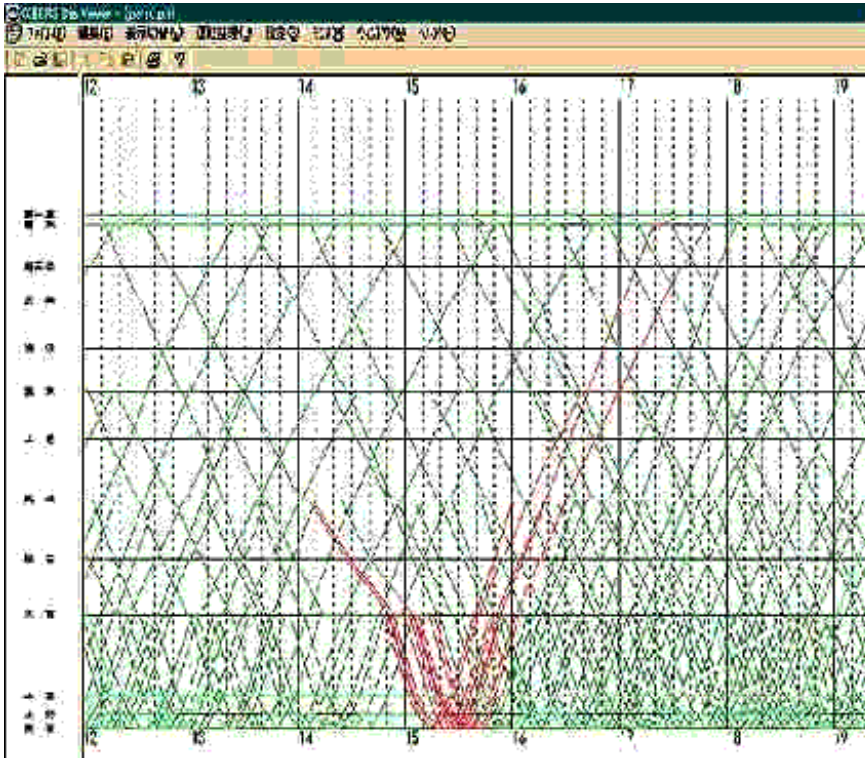
Tohoku line: Original SOT



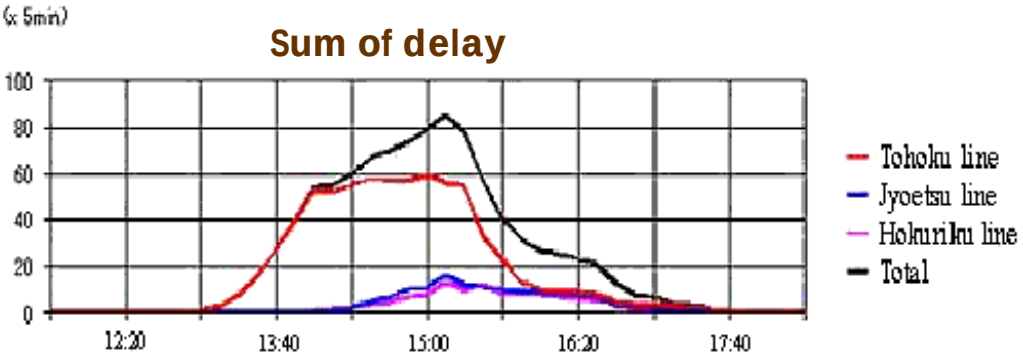
Hokuriku Line: original SOT



Joetsu Line: original SOT

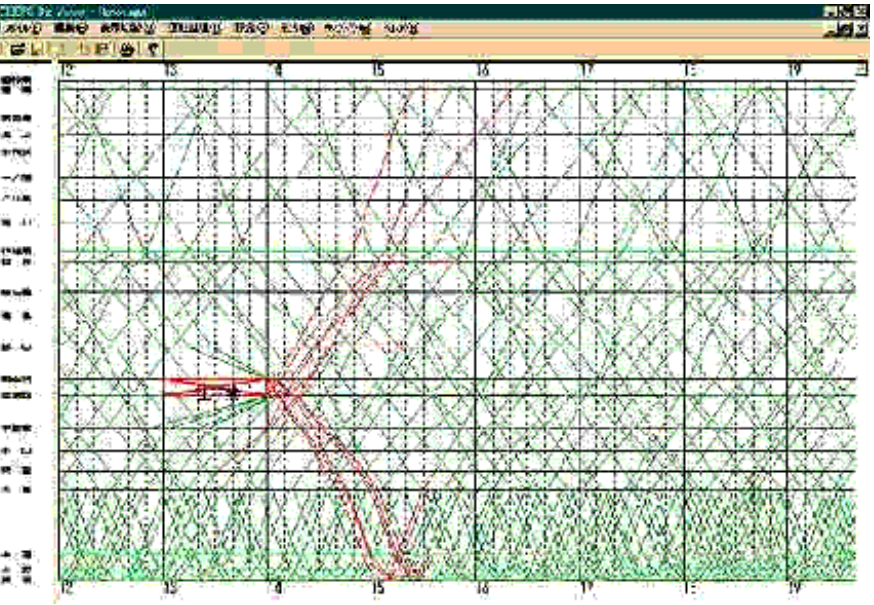


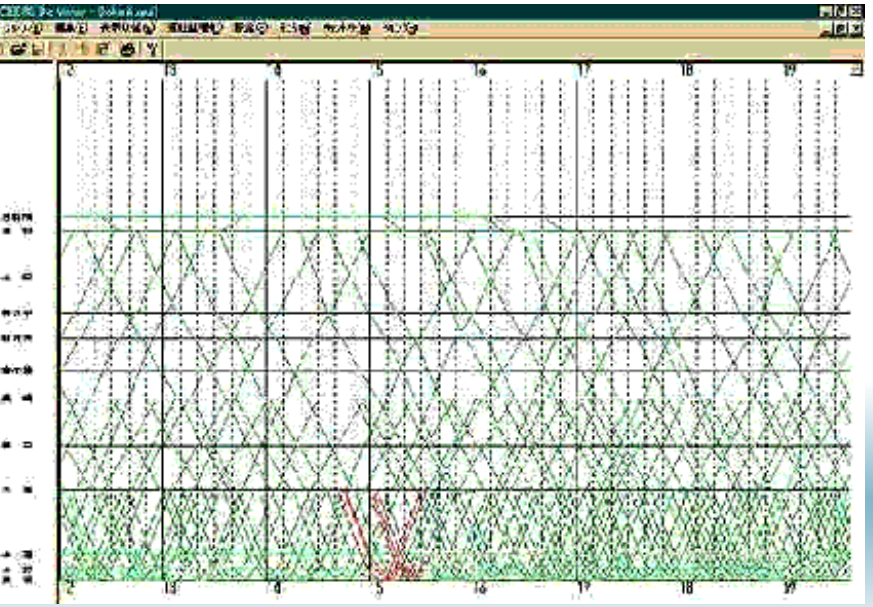
Sum of delay

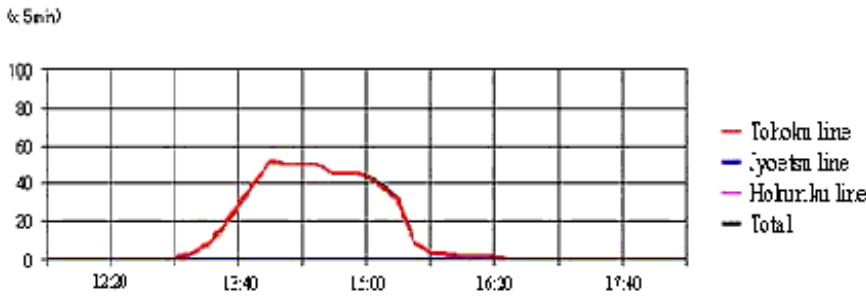


Math-Model Research Inc.

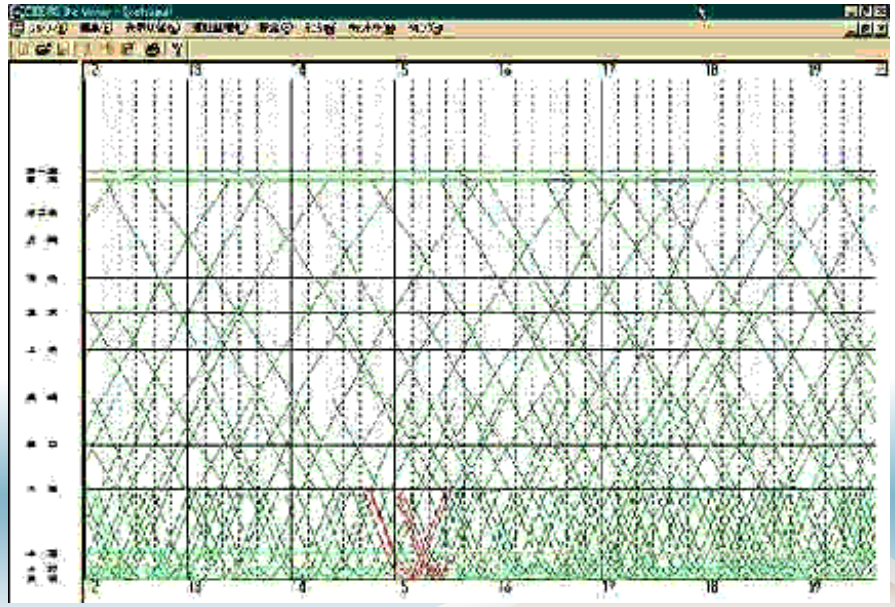
Result of Reschedule





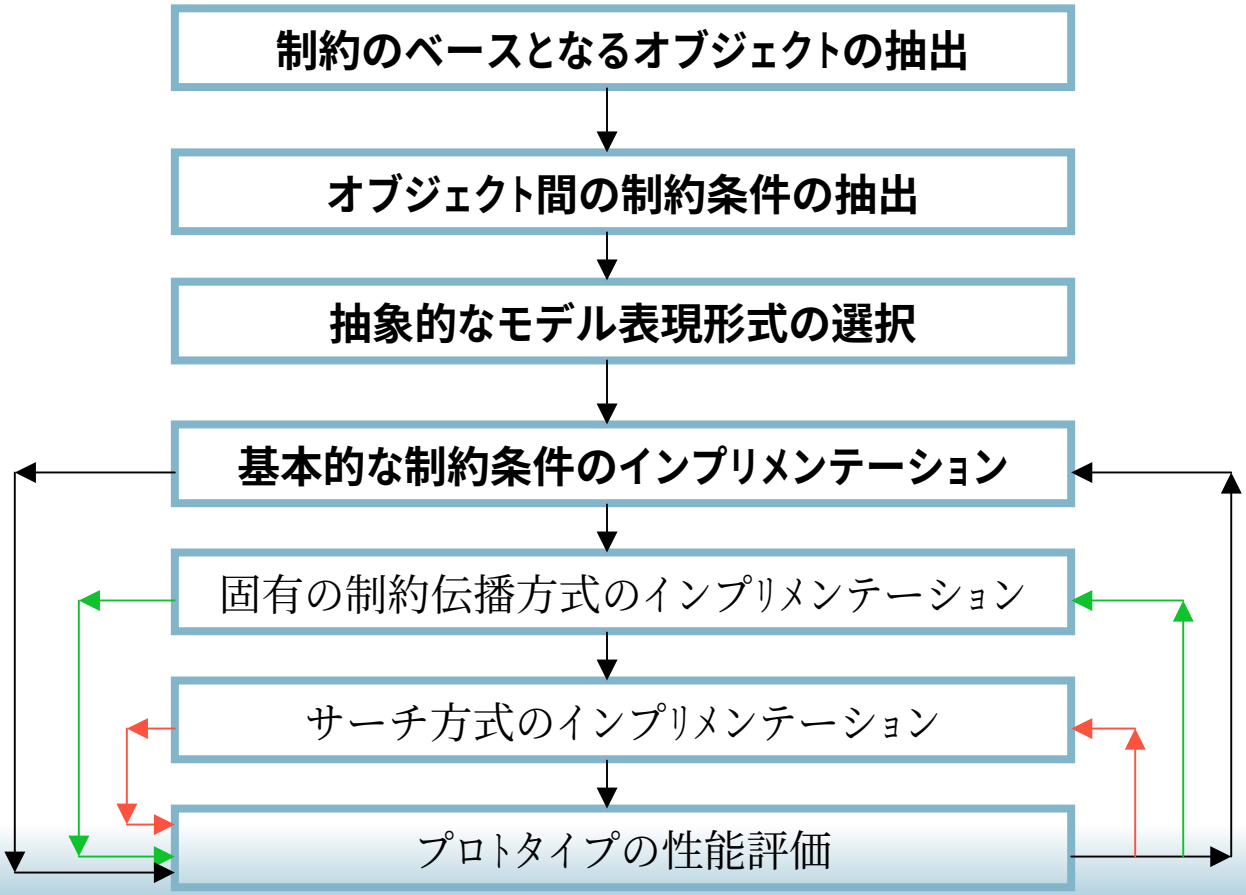


	Change of "SOT" [sections]		Total Delay [minutes]
	Down	Up	
Before Rescheduling	0	0	4496
After Rescheduling	35	32	2219



制約ベースの開発手順（スパイラル）

独立したフィードバックが可能



これからの最適化

- 最適化技術:
  - 高性能の商用のパッケージが開発され、非常に安価になった結果、これを利用した新しいアルゴリズムの開発が進められている。
  - 線形計画法(LP)と制約プログラミング(CP)のハイブリッド化
  - タブー・サーチや遺伝子アルゴリズムとCP、LPのハイブリッド化
  - 新しいタイプのモデル(BN、etc)
- 企業環境:
  - 最適化の欲求は強くなり、また、商用パッケージの能力も非常に発達して、実現可能な時期に至るが、日本企業には、最適化の文化が欠如している場合が多く、これが大きな障壁となる。これを早くクリアした企業が生き残ってゆくと思われる。
- 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。

- あとがき

このスライドは、下記のセミナーで使用されました。

最新製造業ソリューションセミナー

～最適化とWeb技術で変革する製造～

日　　時：2002年2月22日（金）　13：30～17：00

会　　場：東京コンファレンスセンター　セミナールーム202

東京都千代田区飯田橋3-13-1　TEL03-5214-2020

主　　催：伊藤忠テクノサイエンス株式会社

協　　賛：アイログ株式会社

「基調講演 ～ 最適化技術の現状と計画系システムの開発 ～

（株）数理モデリング研究所 代表取締役 野末 尚次　　」

- **連絡先**

（株）数理モデリング研究所　　代表取締役 野末 尚次

〒186-0002 東京都国立市東1－18－10 パールプラザ ビル

TEL／FAX **042－572－0584**

E-mail　　nozue@bk9.so-net.ne.jp