意思決定支援システムの開発と統合モデリング

野末　尚次

1. はじめに

企業の意思決定問題に対して，実務的にORを応用する場合には，その利用者が必ずしもORの専門家とは限らないため，その業務に対応した意思決定支援システム（DSS: Decision Support System）を構築して利用される場合が多い．

現実の意思決定問題では，対象が複雑ですべての制約条件が明確ではなかったり，評価も複数の曖昧な基準がある場合が普通であり，単純な（答えの得られる）最適化問題に定式化することが困難な場合が多い．

このような問題は，「非構造的な問題」（節2で定義）として認知されており，コンピュータによる解の自動作成を狙った開発を行っても，失敗するケースが多い．

したがって，人間と協調した解の作成が不可欠であり，人間の思考プロセスに適合した探索プロセスをコンピュータにより支援する必要がある．

この探索プロセスは，3種類のモデルで表現される．最初のモデルは，この思考プロセスのモデルである．これには，Simonにより提示された「発見段階」，「設計段階」，「選択段階」の3段階よりなる「**思考プロセス・モデル**」を用いている．

次のモデルは，人間とコンピュータのインタフェースのモデルである．これには，次に述べる二つの重要な課題がある．

非構造的な意思決定問題では，人間の総合力・直感力による判断が不可欠であり，これらが有効にはたらく計算結果の提示が重要である．次に，この人間の判断をコンピュータにフィードバックすることが必要となるが，通常，計画者はORの専門家ではないため，数学的なモデルを直接操作することは困難である．

この問題を解決するために，計画者が考える**ユーザ空間**と数学的な定式化が行われる**モデル空間**を明確に分離して考えると同時に，これらの空間相互のインタフェースの構造を定義した「**インタフェース・モデル**」を導入している．

最後のモデルは，実際にOR理論等を用いて問題を定式化し，ソフトウェアを開発する段階にある．すでに述べたように，非構造的な問題では，制約条件や評価基準が後から追加・修正される場合が多いため，手続的なアプローチによるソフトウェア・ロジック（例：鶴亀算）では，改修等（例：百足を追加？）が困難のため，迅速な対応が難しい．

このような状況では，宣言的なアプローチ（例：連立方程式）が望ましい．宣言的なアプローチとしては，PROLOGが1970年代から提案されているが，基本的には，全数探索をベースとしているため，計算量が指数爆発を起こしてしまい，実用面では有効ではなかった．しかし，現在は，制約論理や制約プログラミングの発展により，無駄な探索空間を**制約伝播**により事前に削除する技術が発達し，効率の良い市販のソフトウェア・パッケージが利用可能となっている．

数理的なアルゴリズムを開発する際の基本的な考え方として，「**制約ベース・モデル**」を採用している．対象となる制約条件を抽出して定義し，これを充足する解を制約伝播を利用しながら探索する方式である．

実際の使用場面では，軽微な条件の変更に対しては，できるだけ事前の計画に近い案が望まれるので，探索をベースとする方式は，この点でも優れている．

ORの理論は，緩和問題として，制約伝播や探索方向の評価で積極的に利用している．

以上に述べたように，実際に有効な意思決定支援システムを開発するためには，この3モデルを十分認識する必要がある．

本稿では，筆者が研究開発してきた意思決定支援システムを例にして，この3モデルの考え方を紹介する．

のずえ　なおつぐ
㈱数理モデリング研究所
〒186-0002 国立市東1-18-10

2. 思考プロセス・モデル

人間の意思決定過程は，Simonによれば，次の3段階よりなる．

1) 発見段階 (Intelligence)：関連情報の収集とそれに基づいた問題点の確認

2) 設計段階 (Design)：問題の定式化，可能解の生成，実現可能性のチェックによる代替案の設定

3) 選択段階 (Choice)：複数の代替案の中から一つの案の選択とそれに基づいた実行案の作成

このような流れを詳細にしたGPPS (Generalized-Problem Processing) と呼ばれるモデル (図1) が提案されている．

問題が**非構造的**と呼ばれるのは，「設計段階において，いまだ明確に把握できない課題が残っていたり，また，選択段階においても，可能な代替案の範囲が確定できないような状況が不可避的に発生する」場合であり，企業の重要な意思決定の多くはこの範疇に属する．この非構造的な問題に対しては，コンピュータによる自動計画作成は不可能であり，人間の直観的な洞察や経験的な知識に基づく総合的な判断が不可欠である．

このGPPSをモデルとしてDSSの開発を行っている．

・問題発見→制約条件の抽出，評価項目の抽出
・代替案作成→制約伝播による不用な代替案の削除
・代替案選択→複数の代替案の評価と選択

これらに関しては，節3以降で言及する．

2.1 人間の直感的な判断

非構造的な問題を解決するためには，人間の直感や経験に基づく判断が不可欠であるが，筆者らが開発した地理情報システム (TRAMPS) の例で示す．

TRAMPSは，交通計画のベースとして，国勢調査等のメッシュ・データを分析・表示可能なシステムである．ここでは，東京圏60 km四方の人口分布の500 mメッシュ・データ (14,400個) を表示する．

図2では，500人単位で階層化して，各メッシュを着色表示している．都心部や鉄道沿線に人口が多いこと，また人口が少ない河川や皇居なども認識できる．

図3では，1,000人単位で階層化した結果を示している．この結果は，図2とは大変様子が異なり，山手通と環状7号線に挟まれた地域が人口の最も稠密な地域であることに気がつく．これは，地図状に表示された人口分布のイメージに対して，経験的に取得している道路情報をマッピングすることにより，知識が抽出されたことを示している．

この知識からは，東京で地震が発生したら，この地域で非常に大きな被害が発生する可能性が高いことを推定できる．これは国の防災計画でも指摘されている．

この14,400個のメッシュデータを地図イメージにマッピングしなければ，このような知識を得ることは

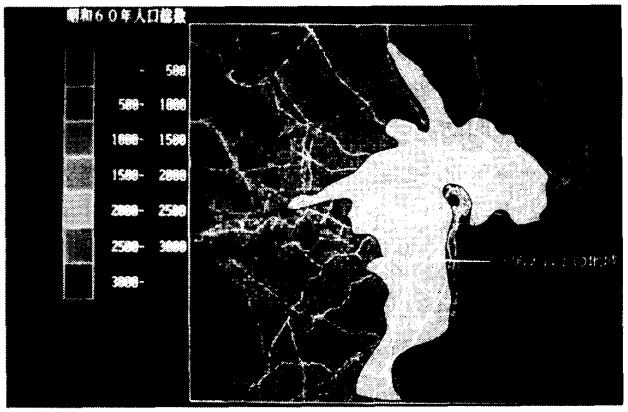

図2 首都圏の人口分布 (TRAMPS: 500人単位)

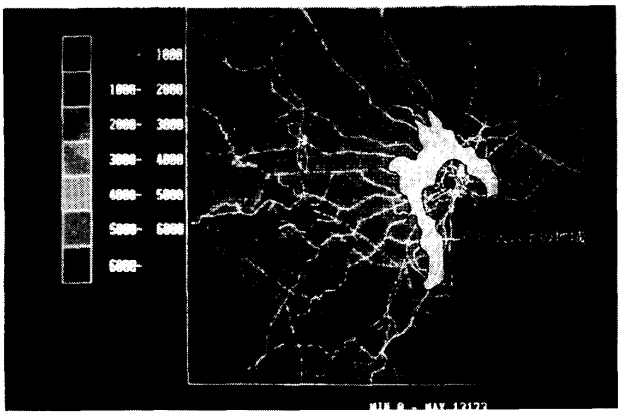

図3 首都圏の人口分布 (TRAMPS: 1,000人単位)

	問題発見	代替案作成	代替案選択
データ収集	1) データ収集	4) 妥当性の検証	7) 実行可能性検証
データ加工	2) 問題認識	5) 分析	8) 実施案の作成
評価・選択	3) 概念モデル構成	6) 解決策の導出	9) 実施案の提案

計算 ↑ ↓ 総合

図1 GPPSの思考プロセスモデル

できないと考えられる．また，コンピュータも人間が持っている経験知識がなくては，また，あったとしても，このような結論を得ることは困難と考えられる．

3. インタフェース・モデル

DSSの利用者は数理計画等の専門家ではないため，専門的な知識を用いたインタフェースは機能しない．

このような状況として，英語は理解できるが，和文英訳はできない人が「日英翻訳システム」を利用するケースを実際に日英翻訳システムを用いて調べた．

図4に示すように，まず「文例1」を入力した．この英文を検討すると，主語が誤っているので，「文例2」のように修正した．この結果から，「幼い兄の子供」をシステムが意味論的に解釈できないようなので，「文例3」のように修正した．この英文からは，「連れて帰る」という動詞が2語に分割されて誤訳になっている．これは，日本語の例文の中の「て」が「買って帰る」のと同じ解釈をされていると考えられるので，「連れ帰る」と修正した「文例4」を入力した．この結果得られた英文は，満足の行くものである．

この説明から明らかなように，翻訳希望者は，「英文和訳」と「日本語の修正」だけで，高度な「和文英訳」に成功している．

この事例を分析することにより，図5に示すようなユーザ空間とモデル空間を持ったモデルを導入した．

今回は，ユーザ空間からモデル空間のアルゴリズムが想定できたので，翻訳がうまく行えた．実際の問題でも，この予測可能性は非常に重要な性質である．

LPで解く時に，解を改良しようとして，ネックとなっている原料を少し増した結果，意図とは全く異なる別解が出て困惑することがあるが，これは，予測可能性がないからである．このような状況では，モデル空間の特性をガイドする支援機能が必要である．

このような構成をテストするために開発した「列車乗継案内システム（EST 3）」を次節で紹介する．

3.1 列車乗継案内システム

このシステムは，図6に示すように，

- 「A駅をt時に出発して，B駅に最も早く着くには？」
- 「B駅にt時迄に到着するには，A駅を何時に出発？」

等の質問に対して，出発/到着時刻や乗継経路を案内するシステムである（このシステムは，1990年に開発されたが，事情があって実使用されなかった）．

図6で「貴方の年齢は？」と問うている部分が，こ

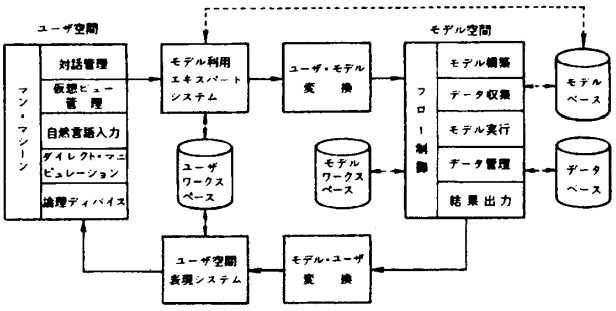

図5 高度なツールを含むDSSの構成方式

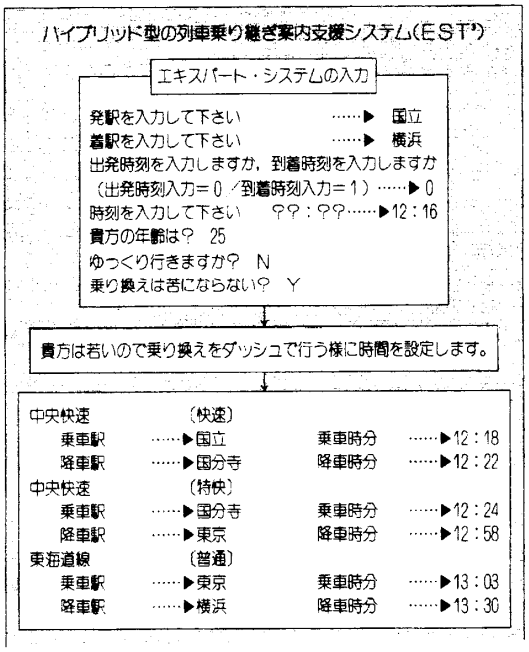

図6 列車乗継案内システム（EST 3）

〔文例1〕 幼い兄の子供を家に連れて帰った。
⟹ Accompanying a child of a young older brother in a house it returned.

〔文例2〕 私は，幼い兄の子供を家に連れて帰った。
⟹ I accompanied a child of a young older brother in a house and returned.

〔文例3〕 私は，私の兄の幼い子供を彼の家に連れて帰った。
⟹ I accompanied a young child of my older brother in his house and returned

〔文例4〕 私は，私の兄の幼い子供を彼の家に連れ帰った。
⟹ I took back a young child of my older brother in his house.

図4 日英翻訳システム利用例

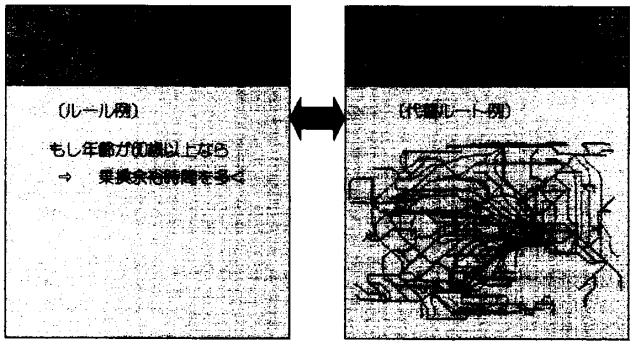

図7　ハイブリッドなシステム構成

変数の定義：仕事の集合Nに対して、以下の変数を定義する
$X[s]$: 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:
$X[s] \in A[s]$ $s \in N$　// A[s]は、sの直後に実行可能な仕事の集合
$Y[s] \in P$　$s \in N$　// Pは、作業者の集合
$Y[X[s]] = Y[s]$　$s \in N$　// sとX[s]は、同じ作業者が担当
$T[X[s]] \geqq T[s] + d[s]$　$s \in N$　// d[s]は、sの作業時間
$X[s] \neq X[u]$　$s \neq u$　$s,u \in N$　// 仕事は、一人に割当
$a_s^{X[s]} \leqq T[s] \leqq b_s^{X[s]}$　$s \in N$　// 開始時刻の制約

図8　スケジュール問題の制約表現

のシステムの最大の特徴である．これは，前節で述べたように，利用者がモデル空間をコントロールする機能を提供している．年齢を調べて，高齢者には，「乗換え回数の少ないルート」や「乗換えは，接続時間が十分ある乗換」等を考慮した乗継案内案を提供する．

このような機能を実現するために，「AIのエキスパート・シェル」と「数理計画の経路生成システム」という2個のサブシステムで構成されている（図7）．

列車検索エキスパート・システムは，ルールベースのエキスパート・シェルで，LISPで開発されている．この役割は，入力された属性に従って，数理計画システムの問題を修正することと，返されてきた複数の案から利用客属性に適した案を抽出することである．

このサブシステムは，ルールにより制御されているので，ルールを修正・追加することにより，利用者の新たなニーズに適合させることができる．

さらに，利用者に対して，文字ベースの時刻，経路情報の外に，可能な代替経路と選択された経路を画面上に同時に表示することにより，直感的に回答の是非を判断できるよう支援している．

代替経路生成・数理計画システムは，ネットワーク理論をベースとしたアルゴリズムにより，複乗禁止条件（同一駅の複数回通過を禁止）を満たす許容経路を複数生成し，時刻表を使って列車乗継案を生成する．許容経路の生成では，複乗禁止条件を除いた緩和問題を考えると，次の不等式が成立することが分かる．

発駅～中継駅～着駅の許容経路の最短距離
≧ 発駅から中継駅迄の最短距離
＋ 着駅から中継駅迄の最短距離

この不等式を利用することにより，実際には使用されない迂回距離の大きい経路の事前削除が可能となり，代替案が絞り込まれて，計算時間が大幅に短縮される．

4. 制約ベース・モデル

一般的に，組合せ問題は，問題の規模に対して計算量が指数関数的増加するため，最適解を求めることは困難なケースが多い．

また，非構造的な問題の特徴でも述べたように，この種の問題では，ある評価基準の下で唯一の最適解が得られても，潜在的な条件等で実施できない可能性が高いこともあり，あまり有効ではない．

むしろ，ある水準を満たす複数の代替案が得られる方が実用上は役に立つ．また，時間の経過とともに条件が変わるので解を修正するが，できるだけ元の解に近い解を生成する機能も重要である．

制約ベースのモデル化では，数上げを基本としており，処理形態上は，このような要望には対処しやすい．

また，ソフトウェアの開発上も制約をベースに設計されるので，新たな制約の追加は容易である．

制約ベースの記述は，複雑な問題も非常にシンプルに表現できる場合が多い．図8に典型的なスケジューリング問題の制約ベースによる定式化の例を示す．

問題：「作業順序に制約のあるN個の仕事をP人の作業者に割り当てる」
条件：一つの仕事を複数の作業者に割当ない．
　　各仕事には，次に実行可能な仕事の集合がある．
　　各仕事には，着手開始時刻の制約がある．

この問題は，$X[s]$（sの次に行う仕事）と$Y[s]$（sを行う人）という変数を導入することにより，非常に簡潔に表現できる．

・sを行う人と$X[s]$を行う人は同じ
・現在の仕事が違えば，次の仕事も違う

この例からも分かるように，制約ベースの開発では，制約条件の追加・削除は非常に簡単である．したがって，スパイラルな開発が可能であり，使用開始後のメ

ンテナンス性も非常に良い．

しかしながら，複雑な問題に対して，制約ベースで解を探索するには，場合によっては，不要な領域を事前に強力に削除する制約伝播ロジックを独自で開発する必要がある．

ここがキーポイントとなるが，これには，OR関連のモデルの知識や運用能力が本質的に問われるので，チャレンジする価値はあると思う．

5. おわりに

実際の問題を解決するためには，その問題を数学的に処理して解を得るプロセスも重要だが，計画者と協調して問題を解くプロセスも非常に重要である．

数学的にモデル化ができても，その入力データが存在しない場合もあり，その時は，データの生成法を考案することからスタートしなければならない．

現実の問題を解決するためには，GPPSで示されている総合的なアプローチが必要であり，これが私の基本モデルである．

参考文献

[1] 野木尚次：http://www.math-model.co.jp.