経済学研究 48－2
北海道大学 1998.9

＜研究ノート＞

OR/MSを活用した意思決定を支援するモデリング・システムの現状

向原 強

1．はじめに

組合せ最適化問題のように効果的な汎用解法の存在しない問題領域に対して，数理モデルを作成し，それを解くことを支援するための実用性の高いモデリング・システムの構築は，挑戦的な試みである。そのようなモデリング・システムでは，専用ソルバーを利用できることが必要であると考えられる。よって本ノートでは，既存モデリング・システムを調査し，専用ソルバーに対応するためのモデリング・システムの構築について検討する。

1．1．モデリング・システムの役割と拡張可能性

Fourer[1993]によると，**数理モデル**（mathematical model）の表記には**モデル設計者**（modeler）が設計し，理解しやすい形式にした**モデル設計者形式**（modeler's form）と，その解法プログラムであるソルバーの入力形式である**アルゴリズム形式**（algorithm's form）がある。一般にこれらは全く異なるもので，それ故，モデルを解くためにはモデル設計者形式で記述された数理モデルをアルゴリズム形式に変換することが必要になる。しかし，これには時間的・金銭的にコストがかかり，間違いも起こしやすく困難な作業である。**モデル記述言語**[1]（modeling language）とは，モデル設計者形式を表現するために設計されたコンピュータ実行型の言語であり，近年多数提案されている。モデル記述言語を利用するとコンピュータによるアルゴリズム形式への自動変換が可能となるため，前述の労力を省くことができる。この変換を実行するソフトウェアは**モデル・トランスレータ**（model translator）[2]と呼ばれる。モデル・トランスレータはアルゴリズム形式の数理モデルの作成が使命であるので，ソルバーの入力形式が分かっていればソルバー本体が付属していなくとも実行することができる[3]。よって，これからの議論で対象とされるソルバーは，モデル・トランスレータが機能するのに十分なほど入力仕様が明らかであり，モデル・トランスレータ

1）本ノートでとりあげるAMPLやGAMSなどのmodeling languageに対する訳語は確立しているとはいえないが，ウィリアムス[1995]，柳浦[1998]などで利用されている呼称の“モデル記述言語”を採用する。モデリング・システム機能のうち，モデル記述の方法を提供する言語体系という意味で，適当な呼称であると考えられるためである。

2）Fouer[1990]による呼称を採用した。

3）ただし，実際上モデル・トランスレータはアルゴリズム形式数理モデルを直接ソルバーに渡すことが一般的であるので，モデル・トランスレータのみの実行が常に可能というわけではない。しかし，例えば2．1で検討されるMPL（アカデミック版）では，CPLEX以外のソルバー本体は付属していないが，対応するソルバーの入力形式を出力できる。また，2．2で検討されるGAMSに付属するMPSWRITEのように，アルゴリズム形式数理モデルを出力するだけのプログラムをソルバーに含めている場合すらある。これらは本ノートの表記法を採用するならば，ファイル出力機能を備えたモデル・トランスレータと呼ぶべきものであり，この場合，モデル・トランスレータのみの実行が可能であるといえる。

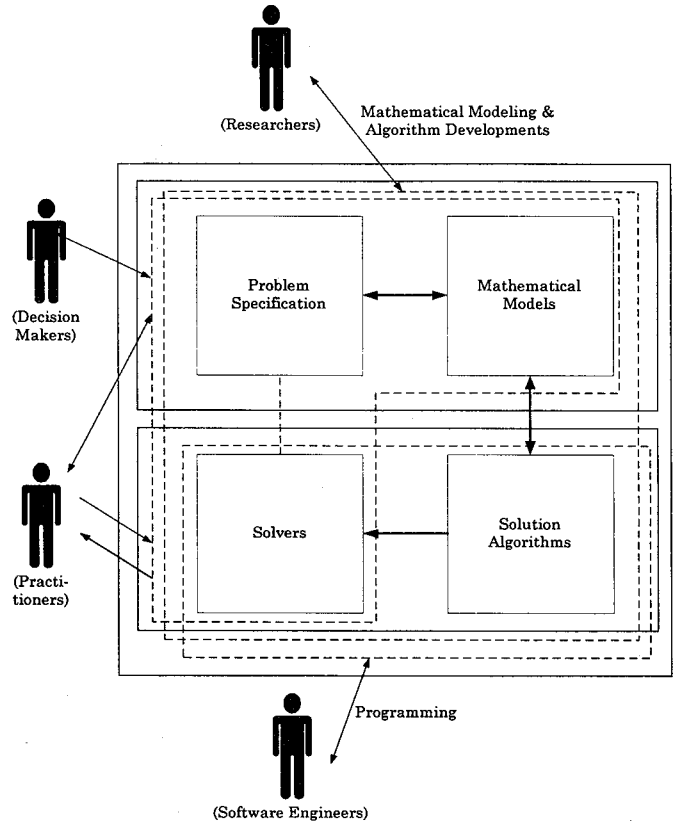

図1：MS／OR を利用する意思決定プロセス

により生成されるアルゴリズム形式数理モデルを入力データとして実行することが可能であることを前提とする。このように，数理モデルの表現方法を提供するにすぎないモデル記述言語を利用して数理モデルを解くためには，モデル・トランスレータやソルバーなどを利用可能にする統合的なシステム環境が必要とされる。このようなシステム環境を，本ノートではGeoffrion [1987]やMaturana[1994]らの呼称を採用し，**モデリング・システム（modeling system）**[4]と呼ぶことにする。本ノートで取り上げるMPL[5], GAMS[6], AMPL[7], ASCEND[8]は，代表的なモデル記述言語であるが，ここではそのシステム環境について議論するため，モデリング・システムとして扱うことにする。モデリング・システムには，モデル記述言語，モデル・トランスレータ，ソルバーに加えて，モデル記述の編集機能，モデルやソルバーの管理機能などが必要とされる。

4）Maturana[1994]では，modeling environment（モデリング環境）とも呼んでいる。この呼称は，ASCENDでも採用されている。

5）Maximal Software, Inc[1994]参照のこと。このマニュアルは，http://www.maximalsa.com/mplman/mplwtoc.htmlよりダウンロードできる。

6）Brooke, et al.[1996]参照のこと。

7）Fourer, et al.[1993]参照のこと。

8）The research group of A. Westberg[1997]参照のこと。

モデリング・システムは，OR／MSを利用する意思決定支援の効率性を高めるための有効なツールとなることが期待される。OR／MSを活用する意思決定プロセスを，関口[1997]は図1のように示し，次のように解説している。“*意思決定者は，OR／MSの十分な知識を持った専門家（OR実践者あるいは・およびOR研究者）の支援を得つつ問題の切り出しを進め，結果を問題定義なる記述に整理する。専門家はこれを基にして数理モデルないしシミュレーションモデルを構築する。このモデルが既に利用可能な解法を持ち，かつ，そのインプリメンテーションであるソルバーを活用できるなら，必要なデータを整備して解き，意思決定者に提示することになる。もし，解法とソルバーの一方ないし両方が利用可能でないなら，それを開発することが必要となる。場合によっては既存の解法やソルバーを利用可能なようにモデル自体を修正するかもしれず，それはさらに，問題定義の変更につながるかもしれない。…*”モデリング・システムの果たすべき役割は，この意思決定プロセスのうち数理モデルの構築からソルバーの起動と解の分析までのプロセスの支援であり，この一連の意思決定プロセスを支援するための有効なワークベンチになりうると考えられる。また，その利点として以下の点があげられる。

- モデル記述言語を利用することから，モデル設計者に対するインターフェイスが優れており，数理モデルの記述は比較的容易である。
- 構築されたモデルに対して適当なソルバー本体が組み込まれているならば，必要なデータを整備さえすればモデリング・システムは意思決定者に解を提示できる。
- 構築されたモデルに対する適当なソルバー本体が組み込まれていなくとも，適当なソルバーのモデル・トランスレータが組み込まれていれば，ソルバー入力形式のデータを容易に形成できる。このとき，ソルバーはモデリング・システムと必ずしも同一環境（OS，プラットホーム，プログラミング言語，GUI）である必要はない[9]。
- モデル・トランスレータが組み込まれていれば，異なるソルバーの入力データ（すなわちアルゴリズム形式数理モデル）を，同じモデル記述言語を利用して記述することができる。これによりモデルの変更や修正によるソルバーの切り替えを容易に行うことができる。ソルバーの切り替えによりアルゴリズム形式数理モデルを新たに構築する必要はない。また，統一したモデル記述言語を利用することにより，モデルの標準化を図ることも可能となる[10]。

9）例えば，パーソナル・コンピュータ（PC）上のモデリング・システムの中でモデルを記述したが，ハードウェア的な制限から，ワークステーション（WS）上のソルバーでないと解を導くことができないこともあり得る。一般にモデリング・システムの中で構築された数理モデルが，そのシステム環境でソルバーを実行し解を導出できるとは限らない。この場合，モデリング・システム上で解を提示することにならないかもしれないが（後述のNEOSクライアント利用のような例外もある），もしWS上で起動するソルバーへの入力データ（すなわち，アルゴリズム形式の数理モデル）を生成する性能を持ち合わせていれば，ソルバーを入手もしくは開発することにより，WS上で容易にソルバーを実行することができる。特にネットワーク環境が発展した今日では，別のシステム環境にあるコンピュータをサーバとして利用できる環境が整ってきているので，こうした使い方（PC上で作成したデータをWS上で実行する）は容易になってきている。NEOSサーバ（http://www-c.mcs.anl.gov/home/toc/Server/）のように，AMPL（2．3節参照）で記述したモデルをWebもしくはe-mailで送信すれば解を返信してくれるサービスなどもある。これも異なるシステム環境にあるソルバー利用の一例といえる。また，NEOSクライアント（http://www.ampl.com/cm/cs/what/ampl/NEOS/）というソフトウェアを組み込めば，モデリング・システム（UNIX上のAMPL）の中でNEOSサーバーのソルバーを起動することも可能である。

10）モデル標準化のメリットはいくつかある。モデル設計に関わる専門家同士の意思疎通を容易にすることや，これまで蓄積された過去の事例を再利用するこ

モデリング・システムが対応可能な（すなわちモデル・トランスレータが備わる）ソルバーは豊富であることが望ましいが，ソルバーが豊富であると，それだけその選択は困難なものとなる。そこで，既存のモデリング・システムには適用可能な領域が比較的広い幾つかの**汎用ソルバー(general-purpose solver)**[11]に対応したモデル・トランスレータを備えている[12]。ソルバーの汎用性により，本来専門家によって行われる高度な知的作業であるソルバー選択は容易になる。線形計画問題におけるシンプレックス法のような汎用解法が有効なクラスの数理モデルにとって，汎用ソルバーに対応可能なモデリング・システムは有効である。

ところが，例えばスケジューリング問題をはじめとする組合せ最適化問題には有効な汎用の解法が存在しない。既存のモデリング・システムは有効なソルバーのためのモデル・トランスレータが利用可能であることを前提としているので，このクラスの問題に対しモデリング・システムを利用することは一般には困難である。ソルバーが利用可能でない場合，上記の意思決定プロセスに従えば，構築された数理モデルに対応したソルバーを独自に開発するか，問題状況に対応した汎用ソルバーのカスタマイズを試みることになる。こうして開発するソルバーを**専用ソルバー(problem-specific solver)**[13]と呼ぶことにする。このようなソルバーは組織の中でOR／MSによる意思決定支援の実践にともない蓄積されていく。これを**ソルバーベース(solver-base)**と呼ぶことにする。生産スケジューリング問題のように直面する問題毎に専用ソルバーを開発する必要があるクラスでは，モデルやそれに対応できるソルバーのデータベース化に関する議論[14]がなされている。モデリング・システムにおけるソルバーベースはソルバーをデータベース化するための一つの手法と考えられる。モデリング・システムにとってソルバーベースは単なるソルバー自身の集合体とは異なる。モデル記述言語で構築された数理モデルはモデル・トランスレータを通してソルバーを実行できるので，ソルバーベースの中にはモデル・トランスレータが含まれているべきであり，モデリング・システムの観点からは，モデル・トランスレータこそがソルバーベースにとって本質的である。

もし，モデリング・システムが豊富な専用ソルバーに対応したモデル・トランスレータを備えることができるならば，次のような利点がある。

● モデリング・システムが対応できる問題領域

とが容易になる。コンサルタント会社や製造企業におけるOR／MS専門家集団のように，多種多様なOR／MSプロジェクトに関わる組織にとって有効である。

11) ここでいう汎用ソルバーとは，特定の問題構造（例えば，ＬＰ問題における目的関数や制約条件が線形関数で構成されるという構造）を持つクラスの任意の問題に対して適用可能なソルバーである。そのクラスのある特定問題に対しては有効であるが，別の特定問題に対しては有効でないような手続きは汎用ソルバーには組み込まれない。そのため，汎用ソルバーが扱うことのできるデータのサイズや実行速度は，特定問題用に開発された専用ソルバーのそれよりも一般に性能が劣る。

12) 2節で紹介するアカデッミク版のモデリング・システムには機能制限付きソルバー本体が組み込まれているが，製品版のモデリングシステムを購入した場合にはソルバー本体は別途購入する必要がある。

13) 特定構造をもつクラスの問題のうち特定問題にのみ有効な手続きが組み込まれているソルバー。例えば，2．1節で紹介される輸送問題のみを対象として有効なソルバーを開発した場合，そのようなソルバーは専用ソルバーといえる。このソルバーを利用すれば，輸送問題の特殊ケースである割り当て問題を解くことができるが，輸送問題の属するクラス（ＬＰ問題）の任意の問題を解けるとは限らない。しかし，輸送問題を解くときにはその特殊構造を利用して計算に要する記憶容量や実行速度に関し，汎用のＬＰソルバーよりも性能を高めることが期待できる。

14) 例えば，生産スケジューリング・シンポジウム'97，パネル討論，「スケジューリング問題・解法の複雑度とそのデータベース化」，オーガナイザ：藤本英雄，による議論。

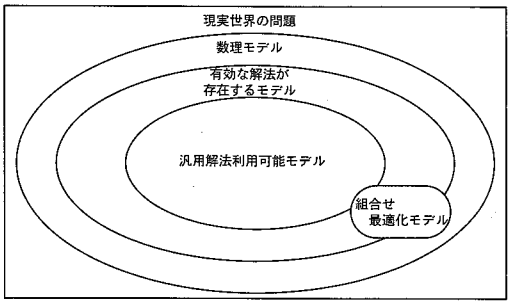

図２：ソルバーの観点から見たモデル領域

を広げることができる（図２参照）ことに加え，汎用ソルバーよりも効率的な解法を選択することが可能になる。

- 既存のモデルと同型であることが分かれば，そのモデルに対して有効なソルバーに関する情報は既知であるので，既存の専用ソルバーを直接活用することができる。
- 問題構造が明確になり，当該問題の関連モデルに関するソルバー情報の知識の獲得が促進され，専用ソルバー開発の支援に有効である。

以上の点から，専用ソルバーに対応したモデリング・システム活用の意義は大変大きい。しかし，汎用ソルバー用のモデル・トランスレータのみが付属している既存のモデリング・システムに，専用ソルバーを組み込むためのシステム環境が整備されているかは定かではない。そこで本ノートでは，既存のモデリングシステムが専用ソルバーを組み込むのに十分な適応能力を持つかどうかを検討する。

１．２．モデリング・システムの評価項目

専用ソルバーを取り込むために必要な適応能力を，既存のモデリング・システムについて評価するために，モデリング・システムに備わる以下の３つの機能を評価項目として取り上げた。

ソルバーベース

ソルバーベースは本来ソルバー本体の集合体と考えるのが自然であるが，本ノートではモデリング・システムについて議論するために，モデル・トランスレータをソルバーベースの主体として考えることにする。新たな専用ソルバーを利用するためにはソルバーに関する必要情報およびソルバー自身をモデリング・システムに柔軟に追加修正する必要があり，特にモデル・トランスレータの役割は重要である。また，備えられるモデル・トランスレータは多種多様になるので，ソルバーベースから適当なソルバー・トランスレータを柔軟に選択するために十分な管理機能を持つことが必要となる。既存のソルバーを利用する場合には新たにモデル・トランスレータを追加する必要はないが，汎用ソルバーをカスタマイズするときには，そのカスタマイズ情報を維持・管理できる機能を必要とする。

モデルベース

専用ソルバーを必要とする問題領域は，一般に大規模で複雑である。モデルは大規模で複雑になるほど，モデルベースから利用可能なモデルを検索し，再利用できることが望ましい。しかし，扱うモデルは多種多様であるので，構造化されたモデルベースが必要となる。モデルベースで管理されるモデルは，その抽象度を高めた方が一般性を維持することができる。例えば数式表現の場合，$y=3x+5$のように，具体的数字を利用した表現よりも，aやbの文字列を使ったパラメータを利用し，$y=ax+b$と抽象化して表現した方が一般性を維持できる。$a=3$, $b=5$のような情報は具体的データとして別に保存した方が数式の管理にとっては都合がよい。このように，抽象化されたモデルを具体的データとは独立して管理する方式を**モデル－データ独立(model-data independence)**と呼ぶ。モデルの一般性の維持という観点から，モデルベースは，モデル－データ独立方式が望ましい。

また専用ソルバーは特定の数理モデルに応じて開発されるので，ソルバー本体には問題毎の特殊性も含め，その問題構造が組み込まれていなければならない。よって，専用ソルバーに入力すべきデータの数や種類は少なくてすむ。この意味でもモデルとデータを独立して扱うこと

は専用ソルバーにとって望ましい。

モデル表現能力

扱う問題領域は広範囲に広がるので，極めて柔軟で豊かなモデル表現力が要求される。特に，組合せ最適化の問題にはインデックスや集合の柔軟な表現が必要となる。

1．3．本ノートの内容

第2節では，代表的な既存の代数的モデリング・システムであるMPL, GAMS, AMPLに加えて,オブジェクト指向言語であるASCENDについて，上記の評価項目に基づいて，サーベイを行う。第3節では，専用ソルバーをとりこむという観点から，これまでのモデリング・システムの問題点と将来的な解決策について展望する。

2．サーベイ

1．2でとりあげた評価項目について，既存の代表的モデリング・システムをサーベイする。数理モデルは一般に代数記述され，OR／MSに精通したモデル設計者にとって代数記述は親しみがある。代数記述からモデルを生成する場合，**代数的モデル記述言語(algebraic modeling language)**[15]は便利であると考えられる。よってこれを使用するモデリング・システムをまず取り上げる。代数的モデリング・システムは，MPL,GAMS,AMPLなどが代表的であるので，これらについてサーベイする。また，あわせて，オブジェクト指向のモデル記述言語であるASCENDを取り上げる。これは，代数表記の数理モデルを基にモデル記述を行うことが易しいモデル記述言語ではないが，特に，継承とよばれるオブジェクト指向技術を利用しており，大規模な数理モデルを生成するのに向いている。例えば最小費用流問題は線形計画モデルの特殊ケースであり，輸送問題は最小費用流問題の特殊ケースとみなすことができる。数理計画モデルは継承関係という観点から構造的に把握することができるので，オブジェクト指向を利用した数理計画の記述は有効であると考えられる。モデル記述言語には他にビジュアル指向やスプレッドシート指向のものがあるが，ここで議論するような大規模で複雑な問題領域の場合，これらの言語は有効性が低いと考えられることからここでは取り上げない。

1．2で取り上げた評価項目について，これらのモデリング・システムを調査した結果は，表1のようにまとめることができる。

2．1．MPL

MPLは，Maximal Software Inc.で開発されたモデリング・システムである。この学生用版は，インターネット上で公開[16]されており，無償でダウンロードできる。以下の**輸送問題(Transportation Problem;TP)**を，MPLによりモデルを記述すると図4[17]のようになる。比較が容易になるように，本ノートでとりあげたモデリング・システムはすべてTPを例題として使用した。

> (TP)
> ある会社が1種類の製品をm個の倉庫からn個の販売店に輸送しようとしている。
> a_i：倉庫iにある製品の総量
> b_j：販売店jでの製品の必要量
> d_{ij}：倉庫iから販売店jへの製品1単位あたりの輸送費用
> とする。輸送にかかる総費用を最小化するにはどうしたらよいか？

これを代数的な数式表現で記述すると図3のようになる。

15) この呼称については，Jones[1996]を参照のこと。

16) http:／／www. maximal-usa.com／

17) 付属のサンプルモデルを修正することによりインプリメントした。

表１：モデリング・システムの比較

		MPL	GAMS	AMPL	ASCEND
ソルバーベース	領域	LP,(M)IP	LP,NLP (M)IP	LP,NLP,(M)IP	LP,NLP,(M)IP, 方程式系
	独自ソルバーのためのインターフェイス	なし	あり	あり	なし
	管理ファイル	なし	モデル内記述	プロジェクトファイル記述	スクリプトファイル記述
モデルベース	保存形式	ファイル	ファイル	ファイル	ファイル
	識別子	ファイル名	ファイル名	ファイル名	クラス名
	検索機能	なし	なし	なし	なし
	モデル•データ独立	併記	併記	分離表記	併記
モデル表現力	領域	LP,(M)IP	LP,NLP (M)IP	LP,NLP,(M)IP Network	LP,NLP,(M)IP, 方程式系
	インデックス•集合	集合表記	集合表記	分離表記	集合表記
	順序付き集合	あり	あり	あり	なし
	変数インデックス	なし	なし	なし	なし
	宣言ステートメント	なし	あり	なし	あり

LP: 線形計画モデル
NLP: 非線形計画モデル（制約付きなども含む）
(M)IP: (混合)整数計画モデル
Network: ネットワーク計画モデル

ソルバーベース

MPLでは代表的なソルバーであるCPLEXに対するもののほか，Lindo, XA, Turbo-Simplex, FortMPなどに対するモデル・トランスレータが内蔵[18]している。これらは，線形計画，および（混合）整数計画ソルバーである。また，MPLがサポートしているソルバーの範囲内で，ユーザがソルバーを追加削除するための機能も持っている。サポート外のソルバーは追加できない。つまり，モデル・トランスレータは追加できない。(混合）整数計画に対して，CPLEXは整数条件を緩和した線形計画問題の解を界値として利用し，分枝限定法を構成する。分枝限定法は，活性問題の選択方法や界値の計算などのオプショナルな要素が多い。ユーザーはソルバーやオプションといったカスタマイズ情報をモデリング・システムの中で指定する必要があるが，それらの情報はモデリング・システムの中では保存されず，管理の対象とならない。

モデルベース

MPLではサンプルモデルがファイル形式で一つのフォルダにまとめられている。これはモデルベースとみなすことができる。モデルはファイル名により識別することになる。また，プロジェクトファイルを利用することにより，任意のフォルダにモデルと作業フォルダの場所を設定することができる。MPLでは，モデルと同時に具体的データを入力しておく必要がある。具体的データを入力しないとモデルを生成できないという意味で，MPLはモデルーデータ独立が達成されていない。

18) ただし，学生版には，CPLEX以外のソルバー本体は付属していない。つまりモデル・トランスレータ機能だけが実行できる。（脚注3）参照のこと。）

タイトル：
　　輸送問題（Transportation Problem, TP）;
定義：
I ：倉庫の集合。{1 ... m}
J ：販売店の集合。{1 ... n}
a_i ：倉庫iにある製品の総量
b_j ：販売店jでの製品の必要量
d_{ij} ：倉庫iから販売店 j への製品1単位あたりの輸送費用
x_{ij} ：倉庫iから販売店 j への商品の輸送量
Z ：輸送にかかる総費用

Minimize　$Z = \Sigma_{i \in I} \Sigma_{j \in J} d_{ij} x_{ij}$
Subject to
$\Sigma_{j \in J} x_{ij} \leq a_i$, for $i \in I$
$\Sigma_{i \in I} x_{ij} \geq b_j$, for $j \in J$
$x_{ij} \geq 0$　　, for $i \in I$, $j \in J$

図3 ：輸送問題(TP)の数式表現

```
    {  Transp.  mp1  }
TITLE
   Transportation  Problem ;
INDEX
   SUP : = (seattle, san_diego) ;
   DEM : = (new_york, chicago, topeka) ;
DATA
   a [SUP] : = (350, 600) ;
   b [DEM] : = (325, 300, 275) ;
   d [SUP, DEM] : = (2.5, 1.7, 1.8, 2.5, 1.8, 1.4) ;
DECISION  VARIABLES
   x [SUP, DEM]
MODEL
   MINIMIZE  Z=SUM (SUP, DEM : d*x) ;
SUBJECT  TO
   Supply [SUP] : SUM (DEM : x) <=a ;
   Demand [DEM] : SUM (SUP : x) >=b ;
BOUNDS
   x>= 0 ;
END
```

図4 ：MPLによるTPのモデル記述

モデル表現能力

MPLは非線形モデルを記述することができない。例えば，

```
DECISION VARIABLE
        x;
MODEL
        Z=x*x;
```

のような記述をすると，コンパイルエラーを起こす。すなわち，MPLが表現できるモデルは，線形計画モデル，（混合）整数計画モデルだけである。

MPLは，インデックスにはj∈{1...n}などのように，数値データも利用可能であるが，

```
DEM:=(new_york,chicago,topeka);
```

のように意味のある文字データも扱うことができる。このような機能は，後で解の解釈をするときに有効である。これはこれから扱うGAMS, AMPL, ASCENDも同様である。また，インデックスは，常に集合として扱われる。よって，下に示すように，変数やパラメータにはインデックスを添えて表記する必要がなく，簡易な表現が可能になる。

```
MINIMIZE Z=SUM(SUP, DEM:d*x);
```

この性質は，モデルの表現力という点では不都合である。例えば，あるインデックスi∈SUP, k, l∈ DEMに対して，

```
x[i,k]+x[i,l]
```

のような表現は許容されない。これでは組合せ最適化問題のようなモデルを表記することは困難である。また上記の通り，MPLはベクトルや配列にカッコ[]を利用するが，その中身はインデックスセクションで宣言されたものでなければならないので，変数は利用できない。これは，ここで扱う全てのモデル記述言語について同様である。

2．2．GAMS

1970年代後半に開発されたGAMSは，おそらく最も広範囲に利用されるモデリング・システムである。本システムはインターネット上で公開されていないが，GAMS Development Corporationにリクエスト[19]すれば，機能制限付きデモ版（学生版）を安価に入手できる。図5[20]はTPをGAMSによって記述した例である。

ソルバーベース

GAMSが有するモデル・トランスレータには，CONOPT, CPLEX, DICOPT, LAMPS, MILES, MINOS, MPSWRITE, OSL, PATH, XA, ZOOMに対するものがあげられる。線形計画モデル，（混合）整数計画モデルに加え，非線形モデルにも対応している。また，ユーザ独自のソルバーを利用するためのインターフェイス（つまりモデル・トランスレータの仕様）を公開している。ソルバーやそのオプション情報を指示することができ，それはモデルファイルに書き込むことにより保存される。また，線形計画モデルにはlp，非線形計画モデルにはnlpなどのモデル型を指定することにより，それぞれのモデル型に対応できるデフォルトのソルバーをシステムが自動的に選択するほか，モデル・トランスレータに対するオプション指定により任意のソルバーを起動することも可能である。

モデルベース

MPLと同様に，GAMSではサンプルモデルがファイル形式で一つのフォルダにまとめられ，これをモデルベースとみなしている。モデルを

19) mailto: sales@gams. com.

20) 付属のサンプルモデルを，修正し，MPLと同じモデル（TP）を生成した。

```
Sets
    SUP  canning  plants /seattle, san_diego/
    DEM  markets /new_york, chicago, topeka/ ;
Parameters
    a (SUP)  capacity  of  plant  SUP  in  cases
      /  seattle  350
         san_diego  600/
    b (DEM)  demand  at  market  DEM  in  cases
      /  new_york  325
         chicago   300
         topeka    275  / ;
Table  d (SUP, DEM)  distance  in  thousands  of  miles
                      new_york      chicago      topeka
    seattle             2.5           1.7          1.8
    san_diego           2.5           1.8          1.4 ;
Variables
    x (SUP, DEM)  shipment  quantities  in  cases
    Z  total  transportation  costs  in  thousands  of  dollars ;
Positive  Variable  x ;
Equations
    dist  define  objective  function
    Supply (SUP)  observe  supply  limit  at  plant  SUP
    Demand (DEM)  satisfy  demand  at  market  DEM  ;
dist..            Z=e=sum (SUP, DEM), d (SUP, DEM)*x (SUP, DEM) ;
Supply (SUP)..    sum (DEM, x (SUP, DEM) =l= a (SUP) ;
Demand (DEM)..    sum (SUP, x (SUP, DEM) =g=b (DEM) ;
Model  transport /all/ ;
Solve  transport  using  lp  minimizing  z  ;
Display  x. l, x. m ;
```

図5：GAMSによるTPのモデル記述

ファイル名により識別することもMPLと同じである。また，MPLのようにプロジェクトファイルを利用することはできない。モデル記述に，モデルと同時に具体的データを入力しておく必要があるのはMPLと同様である。

モデル表現能力

MPLとは異なり，GAMSでは非線形計画モデルの記述が可能である。また，MPLと同様に，インデックスとインデックス集合との区別を設けないので簡素な表現となるが，GAMSでは，例えば，

```
k('1991')=e=k('1991')+l('1990');
```

のように，特定のインデックス要素に対する表記[21]が可能である。これはモデル記述のための制限が緩和されていることを示す。また，宣言的ステートメントに加え，for文などの手続き的な実行ステートメントを扱うことができる。displayコマンドに見られるx. l, x. mは，それぞれ，決定変数のカレント値，双対価格を示すものである。変数に対しては，その他に上限，下限などの属性を持たせることができる。変数に対して複数の属性を持たせるこのような表記は，オブジェクト指向技術の流れをくむものと

21) ただし，=e=は等号関係を示す。

```
#transp. mod
set SUP ;    #origins
set DEM ;    #destinations
param a   {SUP} >= 0 ;   #amounts available at origins
param b   {DEM} >= 0 ;   #amounts required at destinations
param d   {SUP, DEM} >= 0 ;   #shipment costs per unit
var x   {SUP, DEM} >= 0 ;   #units to be shipped
MINIMIZE Z :
   sum {i in SUP, j in DEM} d [i, j]*x [i, j] ;
subject to Supply   {i in SUP} :
   sum {j in DEM}   x [i, j] <=a [i] ;
subject to Demand   {j in DEM} :
   sum {j in SUP}   x [i, j] >=b [j] ;
#transp. dat
param :  SUP :  a  := #defines set "SUP" and param"a"
         seattle      350
         san_diego    600 ;
param :  DEM :  b  := #defines "DEM" and "b"
         new_york   325
         chicago  300
         topeka   275 ;
param d :
                        new_york  chicago  topeka  :=
         seattle          2.5      1.7      1.8
         san_diego        2.5      1.8      1.4 ;
```

図6：AMPLによるTPのモデル記述

みられる。

2．3．AMPL

AMPLは，Fourerらによって開発されたモデリング・システムであり，Fourer, et al [1993][22]を購入すれば，機能制限付き学生版であるAMPL Plus Student Edition for Microsoft Windowsを入手することができる。TPを，AMPLにより記述すると図6[23]のようになる。

ソルバーベース

AMPLはXLSOL，LX-XLSOL，LSGRG，CONOPT, MINOSなど幅広いソルバーに対するモデル・トランスレータが利用できる。線形計画モデル，（混合）整数計画モデルに加え，非線形モデルにも対応している。また，ユーザ独自のソルバーを利用するためのインターフェイス（モデル・トランスレータの仕様）が公開[24]されている。ソルバーやそのオプション情報を指示することができるが，GAMSとは異なり，ソルバーは何を利用し，ソルバーオプションに何を設定するか？などといった情報は，モデルファイルとは別にプロジェクトファイルとして管理する。

モデルベース

AMPLではサンプルモデルがファイル形式

22) http://netlib.bell-labs.com/cm/cs/what/ampl/BOOK/index.html.

23) 付属のサンプルモデルを，修正し，MPLと同じモデル（TP）を生成した。

24) Gay[1997]参照のこと。

```
REQUIRE"atoms. a41" ;
REQUIRE"trans_atoms. a41" ;
MODEL  plant ;
        a  IS_A  supply  capacity ;
        DEM  IS_A  set  OF  symbol_constant ;
        x [DEM], supply  IS_A  flow ;
        d [DEM]   IS_A  unitCost ;
        supply=SUM [x [DEM]] ;
        supply<=a ;
        ShipmentCost  IS_A  cost ;
        ShipmentCost=SUM [x [i]*d [i] i  IN  DEM] ;
END    plant ;
MODEL  test_plant  REFINES  plant ;
        DEM : == ['new_york', 'chicago', 'topeka'] ;
END  test_plant ;
MODEL  customer ;
        b  IS_A  demand ;
        SUP  IS_A  set  OF  symbol_constant ;
        x [SUP]   IS_A  flow ;
        SUM [x [SUP]] >=b ;
END    customer ;
MODEL  test_customer  REFINES  customer ;
        SUP   : == ['seattle', 'san_diego'] ;
END  test_customer ;
MODEL  transportation 2 ;
        SUP, DEM  IS  A  set  OF  symbol_constant ;
        p [SUP]   IS  A  test_plant ;
        c [DEM]   IS  A  test_customer ;
        FOR  i  IN  DEM  CREATE
          c [i]. SUP   : == [j  IN  SUP   |  i  IN  p [j]. DEM] ;
        END  FOR ;
        FOR  i  IN  plantID  CREATE
              FOR  j  IN  p [i]. DEM  CREATE
                   p [i]. x [j], c [j]. x [i]   ARE_THE_SAME ;
              END  FOR ;
        END  FOR ;
        obj : MINIMIZE  SUM [p [i]. shipmentCost |  i  IN  SUP] ;
END  transportation 2 ;
MODEL  transportation  REFINES  transportation 2 ;
        SUP  : == ['seattle', 'san_diego'] ;
        DEM : == ['new_york', 'chicago', 'topeka'] ;
END  transportation ;
```

図7：ASCEND IVによるTPのモデル記述（モデル定義部分）

```
MODEL  test_transportation  REFINES  transportation ;
                    p [1]. a : =350 ;
                    p [2]. a : =600 ;

〜〜〜〜〜〜〜〜〜〜〜〜〜〜〜〜〜〜〜〜〜〜〜〜〜〜〜〜〜〜〜〜〜〜
                         省略
〜〜〜〜〜〜〜〜〜〜〜〜〜〜〜〜〜〜〜〜〜〜〜〜〜〜〜〜〜〜〜〜〜〜

                p [2]. c [1] : =2.5 ;
                p [2]. c [2] : =1.8 ;
                p [2]. c [3] : =1.4 ;
END  test_transportation ;
```

図8：ASCEND IVによるTPのモデル記述（データ入力部分）

で一つのフォルダにまとめられ，モデルをファイル名により識別する。これはMPL, GAMSと同様である。しかし，モデルの記述部分とデータの入力部分は完全に分離され，モデル－データ独立が達成されている。プロジェクトファイルは，上記のソルバー情報に加えて，どのモデルファイルを利用するか？どのデータファイルを利用するか？などの情報を管理できるので，純粋なモデル情報をモデルベースに蓄積させることができる。

モデル表現能力

AMPLは線形計画モデル，（混合）整数計画モデル，非線形モデルを記述することができる。

インデックスの利用法で，MPLやGAMSと異なる点は，本来的な意味での集合とインデックスを区分した利用が可能になっている点である。インデックスは明示的に表現する必要がなく，宣言された集合の要素であれば利用可能である。例えば，

```
MINIMIZE Z:sum {i in SUP, j in DEM} d[i, j]*x[i, j];
```

のi, jがインデックスであり，SUP, DEMは集合である。集合データは数値データとは限らないので，インデックスとして利用するためには，本来，その順序が必要となる。そのような場合は，順序付き集合を設定すればよい。例えば，製造，在庫，売上に関する均衡関係は次のような方程式で示すことができる。

前期末在庫数量＋当期製造数量＝
当期売上数量＋当期末在庫数量

これをAMPLで記述すれば，次のようになる。

```
set WEEKS ordered;
subject to balance{t in WEEKS:ord(t)> 1}
   :Inv[prev(t)]+Prod[t]=Sell[t]+Inv[t];
```

WEEKSは，orderedというキーワードを入れることにより，順序付き集合となる。前期なる概念は，第2期以降という条件が暗黙条件として必要となる。“:ord(t)〉 1”の部分は，それを意味する。集合からの条件付要素取り出しは，この方法で実現できる。ord(t)は，インデックスtの先頭からの順序を返す。前期は，prev(t)によって表現している。方法こそ異なるが，MPLやGAMSでも，このような順序付き集合を扱うことができる。

また，arc, nodeというキーワードを利用することができ，ネットワーク計画問題を表現するのに有効である。

2．4．ASCEND

ASCENDは，D. Benjaminによって最初に

開発されたモデリング・システムであるが，ここで議論するACSEND IVは，Carnegie Mellon大学のA. Westerbergらによって開発された。これはインターネット上でソフトウェアが公開[25]されており，無償でダウンロードできる。ASCEND IVにより，TPをモデル記述すると図7[26]，図8のようになる[27]。

ソルバーベース

ASCENDのモデル・トランスレータには，Slv，QRSlv，LSODE，MINOS，LSSlv，Opt(SQP)，CONOPT，MakeMPSに対するものがある[28]。数理計画用としては，線形計画モデル，(混合）整数計画モデル，非線形モデルに対応している。しかし，ユーザ独自のソルバーを利用するためのインターフェイス（モデル・トランスレータの仕様）は公開されていない。ソルバーやそのオプション情報は，後述のスクリプトファイルで保存，管理できる。

モデルベース

ASCENDはオブジェクト指向言語であるので，一つの数理モデルはmodelと呼ばれる複数のクラステンプレートから構成され，それもまたクラステンプレートになる。クラステンプレートは，クラスライブラリに蓄積されるので，これをモデルベースとして管理できる。しかし，どのクラステンプレートがどのクラスライブラリにあるかなどの検索ルーチンは持っていない。また，クラステンプレートの継承の中で，データを入力することもある[29]という点で，モデルとデータを完全に独立に管理することは難しい。

モデル表現能力

化学実験のシミュレーションを対象とした，複雑で大規模な方程式系を解くためのシステムとして開発されたもので，元来，数理モデル記述を対象としていたわけではないが，数理モデルとしては，線形計画モデル，(混合）整数計画モデル，非線形モデルなどの最適化モデルを記述することができる。

GAMSと同様に，集合とインデックスは分離せず記述する形式をとっている。インデックスは文字データとして扱われる。GAMSやAMPLのように，順序付き集合を定義することはできない。

ASCENDでは，クラステンプレートなるモジュール化されたコンポーネントの組み立てや，それらの継承などの特徴を利用し，単純な数理モデルから大規模で複雑な数理モデルを容易に築き上げることが期待される。使われるクラスライブラリは複数のファイルに分割されていることが通常であるので，どのクラスライブラリからどのクラステンプレートを利用するか？，ソルバーは何を利用しオプションは何を利用するか？などを指定する。この情報はスクリプトファイルとして保存できる。これはAMPLのプロジェクトファイルに相応する。

3．検討

第2節では，代表的なモデリング・システムについてサーベイしたが，いずれも，専用ソルバーを必要とする問題分野のためのモデリング・

25) http://www.cs.cmu.edu/~ascend/ascend-download.htm.

26) P.Pielaetal.[1993]より抜き出したものを修正し，MPLと同じモデル（TP）を生成した。

27) ただし，ソルバーは非線形方程式のものが付属しているだけである（脚注28参照）。よって，ASCEND IVによるTPのモデル記述に対する文法的なチェックは可能であるが，ソルバーを実行しそれを解くことはできなかった。数理計画ソルバーとしてMINOS本体はGAMSに付属していたが，インターネットで公開されているASCNEDの方にMINOS用のモデル・トランスレータが組み込まれていなかったため，それを利用できなかった。

28) インターネット上で公開（脚注25参照）しているASCEND IVにはモデル・トランスレータおよびソルバー本体としてQRSlvのみが付属している。

29) 例えば割り当て問題は，輸送問題の特殊ケースとみなすことができ，輸送問題モデルにおける需要量と供給量のパラメータに，すべて1というパラメータデータを代入することによって，割り当て問題モデルを構築できる。

$$\text{Minimize} \quad \Sigma_{j \in J} c_j x_j$$
$$\text{Subject to}$$
$$\Sigma_{j \in J} a_j x_j \leq b$$
$$x_j \in \{0,1\} \quad j \in J$$

図9：0－1ナップサック問題のモデル記述（0－1変数利用）

$$\text{Minimize} \quad \Sigma_{j \in X^{in}} c_j$$
$$\text{Subject to}$$
$$\Sigma_{j \in X^{in}} a_j \leq b$$
$$X^{in} \in J$$

図10：0－1ナップサック問題のモデル記述（変数集合利用）

システムとしては不十分といえる。この節ではその問題点を議論し，各々の問題に対する解決策を提案する。

3．1．問題点

3．1．1．ソルバーベースに関する問題点

第1の問題点として，構築したモデルに対して適当なモデル・トランスレータを検索し選択できる効果的なソルバーベースを持たないことがあげられる。ここでとりあげたモデリング・システムに付属されモデル・トランスレータを通して利用可能なソルバーが汎用的であることから，ソルバー選択のための機能はほとんど備わっていないし，その必要がないことがこれらのモデリング・システムのメリットということもできる。よって，利用可能なソルバーにはどういうものがあり，どういうモデルに適用可能か？という情報はいずれのシステムでも（オンライン）マニュアルに頼るほかないし，あるソルバーがモデルベースの中のどのモデルを解くのに使用されたか？というソルバーの利用履歴情報は保存されていない。しかし，この情報は，モデルに対して適当なソルバーやそのオプションを選択するときに，参考にできる情報であり，実用上重要であると考えられる。モデルに対する依存性の高い専用ソルバーを組み込むためには当然必要となる機能である。

3．1．2．モデルベースに関する問題点

第2の問題点として効果的なモデルベースがないことがあげられる。上述の通り，代数記述の定式化が完成していれば，代数的モデル記述言語のインターフェイスの利点を活かして実行型のモデルを構築することは容易である。しかし，一般に，定式化自体がやさしい作業ではない。モデルが大規模で複雑になるほど，ゼロ状態から定式化しモデルを構築するのは困難な作業になる。このような場合，モデル構築はモデルベースから適当なモデルを取り出し，それを再利用する方が便利になる。モデリング・システムでも，そのような機能を有するモデルベースを持つ必要があると考えられる。モデルベースは，構造化され再利用に便利な形式であるべきである。ところが，ここでとりあげたモデル記述言語のモデリング・システムは，そのような構造化されたモデルベースを持たない。単なるファイルのコレクションという形式でモデルは保存され，ユーザ（もしくはモデル設計者）が要求するモデルを取り出すためには，ファイル名により識別できる関連モデルを全てオープ

ンして調べなければならず，困難な作業である。また，モデルベースの構造化という観点からすれば，MPLやGAMSのように，モデルーデータ独立が達成されないものは好ましくない。ASCENDはオブジェクト指向言語であり，モデルの再利用という観点では優れているが，やはり，再利用すべきモデルを検索するルーチンは組み込まれていない。

3．1．3．モデル表現能力に関する問題点

第3の問題点は，表現力が限定されていることである。モデル記述言語で表現するモデルは，実はソルバーに依存しており，どのソルバーでも解くことのできないモデルは，表現することができない。組合せ最適化モデルは，汎用的なソルバーがないために，既存のモデリング・システムで扱うことが難しいことを指摘したが，実はソルバーだけの問題ではなく，モデリング・システムに備わるモデル記述言語の表現力が低いことも，その一因と考えられる。例えば，0－1ナップサック問題は，図9の形式で記述されたモデルなら，モデル記述言語を利用して表現することできるが，変数集合を導入した図10のような形式のモデルは表現できない。これは，インデックスや集合を変数とするような表現はできないという，制限によるものである。

各モデル記述言語は，インデックスと集合の表記方法，宣言ステートメントの扱いに，それぞれ工夫や特徴を持つものの，すべての代数記述をインプリメントできるわけではないことを意味する。このことは，モデル設計者にとって必ずしも理解し操作しやすい形式に定式化できることを保障しないということであり，大きな欠点である。

上記のような問題はモデルを適当な形に変換することにより，既存のモデル記述言語で扱えるようにはなるし，汎用ソルバーのデータ入力性能が向上すれば，このような表現が可能となるようなモデル記述言語も出てくるであろう[30]。しかし何れにせよ，そのモデル表現力はソルバー性能のために限定されていることには変わりがない。モデル設計者はソルバー性能を考慮しながらモデルを構築する必要がある。モデル記述言語の役割が解を求めることではなく，数理モデルを記述することであることを考慮すれば，このことはモデル記述言語の魅力を低減させる

3．2．提案するモデリング・システム

3．2．1．ソルバーベースに関する提案

適当な専用ソルバーのためのモデル・トランスレータを検索し，選択できる効果的なソルバーベースのためのシステム環境について考える。専用ソルバーの利用は，解法の効率性の点からも実用上有利である。専用ソルバーを利用するためには，構築されたモデルに対して，“どのような具体的データを利用するか？どのソルバーが適用可能か？ソルバーにはどのようなオプションが必要か？”などといった情報が蓄積されるべきだと考えられる。そのような情報を**結合情報**，その蓄積物は**結合ベース**と呼ぶことにする。例えば，GAMSのソルバーオプションのような情報は，結合情報である。そのような結合ベースは，利用されることにより事例ベースとして充実され，特定のモデルに対して適応可能なソルバーやそのオプション，適用可能なデータのサイズ（すなわち，結合情報）を，そこから知ることができる。ある特定のモデルに対する結合情報は，データサイズにも依存し，複数存在することが通常である。よって，結合情報をモデル情報やデータ情報に埋め込むと無駄が大きくなる。例えば図5をみると，GAMSのモデル記述の中でソルバーの指定をしている。もしこのモデル情報を保持したまま同一モデルに対して別ソルバーを利用しようとするならば，このモデル記述ファイルをまるごとコピーした上で，ソルバー指定部分を修正するという作業が

30) 例えばFourer[1998]を参照のこと。この論文は，http://iems.nwu.edu/～4er/からダウンロードできる。

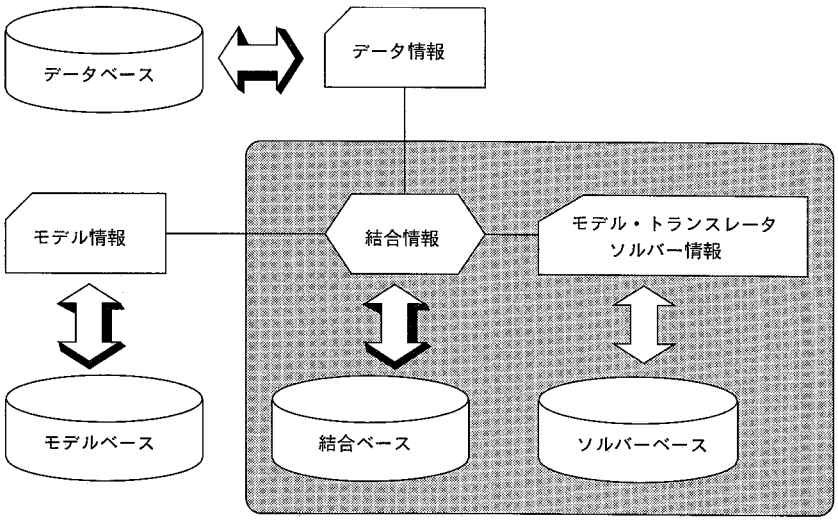

図11：結合ベースによりソルバーベースを取り込んだ提案モデリング・システム概念図

結合情報

ID	1
モデル	transp. mod
データ	transp. dat
ソルバー	tp_solver
オプション指定	Ho　　；ハウザッカー法
説明	2×3の輸送問題を，ハウザッカー法で初期解を求め，飛び石法で改善し最適化する。

図12：結合情報の一例（TP）

モデルトランスレータ／ソルバー情報

ID	tp_solver
モデル	tp_trans
ソルバー名	tp_opt
プラットフォーム	PC／AT，Windows95
ソースプログラム	C言語
説明	小規模（<=50*50）モデルの輸送問題に有効

図13：モデル・トランスレータ／ソルバー情報の一例（TP）

必要となる。ソルバー指定部分以外は全く同じ情報であるのに，別ファイルとして2重に管理することになる。しかも同一モデルを利用しているのに，その関係に関する情報はどこにも保存されない。この問題を解決するためには，結合情報をモデル情報やデータ情報とは別に管理すべきである。モデル情報やデータ情報には識別子を付与し，この識別子を利用した結合情報を管理すれば上記のような問題を解決できる。AMPLのプロジェクトファイルで記述される

内容も結合情報といえるが，このようなAMPL方式はモデル情報やデータ情報との独立管理という観点から望ましい。

既に述べた通り，専用ソルバーを扱うためには，それを構造的に管理するソルバーベースが必要になる。モデリング・システムのユーザは，適当なソルバーをソルバーベースから選択するか，もしくは，ソルバーを自分で構築するという作業が必要になる。しかし，そのような作業を，専門家ではないユーザが実行することは困難であると思われる。よって，事例ベースである結合ベースを通してソルバーを選択するか，OR／MS専門家にソルバーの構築を委ねることになろう。ただし，ユーザーが構築したモデルに対する適当なソルバーが存在したとしても，その結合情報は必ずしも存在するとは限らない。結合情報が存在しない場合は，適当なソルバーやそのオプションを選択することにより，ユーザーが結合情報を生成する必要がある。このとき，専門家でないユーザーが適切なソルバーやそのオプションを選択できるか，また，ユーザー自身が選択の妥当性を確信できるか，は疑問である。したがって，結合ベースにないモデルを扱う場合には，OR／MS専門家の関与が必要になると考えられる。

また既存の専用ソルバーを利用し数理モデルを解く場合には問題毎の特殊性も含めてその問題構造が特定化される。よって，問題定義に従って数理モデルを作成する場合，結合情報によって参照することのできる適当なモデル・トランスレータ／ソルバー情報を選択することができるならば，モデル設計者形式とアルゴリズム形式の対応関係が分かるので，構築すべきモデル設計者形式数理モデルのテンプレートを作成できる。ユーザはテンプレートに基づくことによって容易に数理モデルを記述することができる。専用ソルバーを利用する問題の場合，テンプレートへの入力はデータ情報（例えば，AMPLのモデル記述（図６）中のtransp.datの部分）にのみ行えばよい。逆にテンプレートに基づいて記述することができなければ，そのソルバーが利用できないということを知ることができる。このことはソルバーベースを利用したモデル記述方法の有効性を示すものである。

上記を概念図としてまとめたのが図11である。また，図11における結合情報とモデル・トランスレータ／ソルバー情報の一例を図12と図13に示した。

これまでのモデリング・システムでは，ソルバーが汎用であるために結合ベース，ソルバーベースの管理機能（中の網掛け部分）はあまり必要とされず貧弱であった。もちろん汎用ソルバーに対するモデル・トランスレータもソルバーベースに含めることができることは言うまでもない。この場合，オプション情報が重要となるかもしれないが，それは結合情報として管理されることになる。ソルバー情報は，結合情報に従属して取り出されるので，ソルバーベースとソルバー情報の間の矢印は陰付きになっていない。

3．2．2．モデルベースに関する提案

モデル情報は，モデルベースから検索し再利用できることが望ましい。また，代数記述に親しんでいるモデル設計者にとって代数記述方式は好ましいかもしれない。しかし，代数記述は，インデックス，パラメータ，決定変数などから複雑に構成され，文法的な点からそれを構造的に把握することは難しい。よって，モデルベースは，モデル情報を検索し再利用するためには，意味的な点で構造化されるべきである。関口[1996]は，汎実体－関連モデル(GERM)というアプローチから，構造化された問題定義の方法を提唱し，独立に定義される数理モデルを結合条件という概念で結びつけることを提案している。また，鮑[1997]はそのような事例ベースをインプリメントしている。この場合扱われる問題情報は意味的であり，かつ構造化されている。モデルベースから直接，モデルを検索，再利用するのではなく，構造化された問題のデー

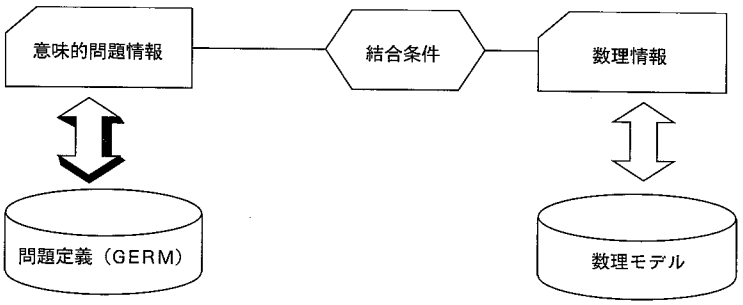

図14：問題定義を通した数理モデルの管理

```
#knap_ext. mod
Var  X  within  J ;

MAXIMIZE  total_Value :
        sum {j  in  X} c [j]  ;
subject  to  Weight_Limit :
        sum {j  in  X} e [j] <=b ;
```

図15：拡張されたAMPL表現による０－１ナップサック問題

タベースから情報を取り出し，結合条件によって結合されている数理情報を管理できれば，構造化されたモデルベースを間接的に利用することができる。この場合，図11中のモデルベースからのモデル情報の取り出し部分は図14のように図示することができる。数理情報と数理モデルの間の矢印に陰がないのは，数理モデルの取り出しが直接行われるのではなく，意味的問題情報を取り出したときに従属して取り出される情報であることを示す。もちろん，この場合の数理モデルはここで議論しているモデル記述言語のような実行型の形式であるべきである。異なる形式で記述された数理モデル（例えば，図９と図10）であっても，それらが意味的情報から構成される同一モデルから派生していることが分かれば，それらの数理モデルの同値関係を知ることができる。そこで，図10のように記述された数理モデルを既存ソルバーのアルゴリズム形式に変換することが困難な場合，ユーザーは数理モデル図10を同値関係にある図９に変換することができれば，その問題（０－１ナップサック問題）を解くことができる。一般に数理モデルは様々な形式で記述することができるので，意味的情報から構成されるモデルベースは重要である。

３．２．３．モデル表現能力に関する提案

モデル記述言語の役割は解を求めることではなくモデルを記述することであるので，モデル記述言語はそれが持つ既存のソルバー性能に依存しない表現力を持つことが要求される。例えば，図10の表現に対応した記述は許容されるべきである。Fourer[1998]では図15のようなAMPL記述[31]を提案している。

この表現はこれまでのAMPL表現に加えて，変数集合Xとキーワードwithinの導入という拡張によって実現できる。もし，変数集合を扱うことのできるソルバーと，ソルバー入力デー

31）ただし，図10に対応するように，変数名およびパラメータ名を修正した。

タへの変換ソフトウェアを備えつけることができれば，図15のようにモデル記述言語で記述されたモデルの解を，モデリングシステムを利用して求めることができる。しかし，たとえソルバーが開発されていなくとも，つまりソルバー開発とは独立して，図15のような表現を可能にするモデル記述言語の拡張の試みは行われるべきである[32]。ソルバーが対応していなければ，ソルバーに対応したモデル記述形式（例えば，図9）への変換をユーザーかモデリング・システム自体が試みればよいし，直接対応したソルバーが開発されれば，それを利用すればよい。ソルバーとは独立したモデル記述言語の拡張によるモデル表現能力の向上の意義は大きい。

謝　辞

本ノートを作成するにあたり，有益なコメントを頂いた関口恭毅教授および鮑金源助手に心から厚くお礼申し上げたい。さらに本稿を査読していただいたレフリーには本稿執筆の最終段階で貴重なコメントを頂いた。記して感謝したい。

参考文献

[1]A. Brooke, D. Kendrick, and A. Meeraus, *GAMS RELEASE 2.25 A USER'S GUIDE*, GAMS Development Corporation, 1996.

[2]R. Fourer, D. M. Gay, and B. W. Kernighan, "A Modeling Language for Mathematical Programming," *Management Science*, Vol. 36, No. 5, pp. 519−554, 1990.

[3]R. Fourer, D. M. Gay, and B. W. Kernighan, *AMPL: Modeling Language for Mathematical Programming*, boyd & fraser publishing company, 1993.

[4]R. Fourer, "Extending a General-Purpose Algebraic Modeling language to Combinatorial Optimization: A Logic Programming Approach," In D. L. Woodruff, ed. , *Advances in Computational and Stochastic Optimization, Logic Programming, and Heuristic Search: Interfaces in Computer Science and Operations Research*, Kluwer Academic Publishers, Dordrecht, The Netherlands, pp. 31−74, 1998.

[5]D. M. Gay, "Hooking Your Solver to AMPL," Technical Report 97−4−06, Computing Science Research Center, Bell Laboratories, 1997.

[6]A. M. Geoffrion, "An Introduction to Structured Modeling, "*Management Science*, Vol. 33, No. 5, pp. 547−588, 1987.

[7]C. V. Jones, *Visualization and Optimization*, Kluwer Academic Publishers, 1996.

[8]S. V. Maturana, "Issues in the design of modeling languages for mathematical programming," *European Journal of Operational Research*, Vol. 72, pp. 243−261, 1994.

[9]Maximal Software, Inc, *MPL User Manual, Maximal Software*, Inc, 1994.

[10]P. Piela, R. McKelvey, and A. Westerberg, "An Introduction to the ASCEND Modeling System: Its Language and Interactive Environment," *Journal of Management Information Systems*, Vol. 9, No. 3, pp. 91−121, 1993.

[11]The research group of A. Westerberg, *ASCEND IV Advanced System for Computations in Engineering Design A portable mathematical modeling environment Release 0. 8*, Department of Chemical Engineering, Carnegie Mellon University, 1997.

[12]H. P. ウィリアムス　著　(前田　監訳，小林　訳)，"数理モデルの作成方法，" 産業図書，1995年。

[13]関口　恭毅，"問題定義の一方法：実体−関連アプローチの拡張，" 北海道大学ディスカッションペーパ・シリーズB, No. 18, 1996年。

[14]関口　恭毅，"意思決定支援システムにおける問題定義について，"オフィスオートメーションVol. 18,

32) ただし，モデル記述言語の拡張はその文法体系を拡張するので文法のチェック機能やモデル・トランスレータの機能も拡張する必要があり，困難な作業である。よって言語体系の拡張の是非は慎重に検討されるべきである。

No. 4－2, 1997年。

[15]鮑　金源, “生産スケジューリング事例ベース構築に関する問題定義／モデルの記述形式の研究,” 経済学研究（北海道大学）, Vol. 47, No. 1, pp. 58－80, 1997年。

[16]柳浦　睦憲, “数理計画ソフト初体験,” オペレーションズリサーチ, Vol. 43, No. 2, pp. 94－99, 1998年。