1997年度日本オペレーションズ・リサーチ学会
秋季研究発表会

# 数理計画のためのモデリング言語 SIMPLE　II　応用

01307380　数理システム　　*田辺隆人　TANABE Takahito
01701240　数理システム　　山下浩　YAMASHITA Hiroshi
数理システム　　徐良為　XU Liangwei

本稿では実際の応用例をとりあげながら SIMPLE におけるモデル記述について述べる.

## 1. ナップサック問題

集合とその要素の概念を用いる例としてナップサック問題:

$$\begin{aligned} &\text{変数} && x_i \in \{0,1\} \quad (i \in S) \\ &\text{最大化} && \sum_{i \in S} c_i x_i, \\ &\text{制約式} && \sum_{i \in S} a_i x_i \leq b \end{aligned}$$

を取り上げる. SIMPLE によるこの問題の記述は以下の様になる.

```
Set S;
Element i(set=S);
IntegerVariable x(index=i,type=binary);
Parameter c(name="c",index=i)
         ,a(name="a",index=i)
         ,b(name="b");
Objective weight(type=maximize);
weight = sum(c[i]*x[i],i); // 目的関数
sum(a[i]*x[i],i) <= b;  // 制約式
```

SIMPLE のオブジェクトである Set や Element, また sum(c[i]*x[i],i) などの表現 (要素が取り得る範囲すべてにわたる和) が通常の数学的表現と対応している. IntegerVariable は整数変数に対応するクラスで, Variable と同様に扱われる.

定数を表す Parameter というオブジェクトとして宣言された目的関数/ 制約式の係数, 右辺の具体的な値はモデル定義に含まれている必要はない. データファイルから例えば以下の様に与えることができる.

```
c = [1] 42 [2] 12 [3] 45 [4] 5  [5] 2
    [6] 61 [7] 89 [8] 32 [9] 47 [10] 18;
a = [1] 39 [2] 13 [3] 68 [4] 15 [5] 10
    [6] 20 [7] 31 [8] 15 [9] 41 [10] 16;
b = 121;
```

SIMPLE は問題 (より一般的には「モデル」) の解釈時にデータファイルの内容を対応するオブジェクトにあてはめて, 実際の解析が可能な計算グラフを作成して解析プログラム (ソルバ) へと送る. 直接与えられていない集合 S の内容はパラメータ a,b の添字から自動的に決定される.

この例では問題の具体的な規模 (この場合は 10 変数) はデータファイルの内容によって決定され, モデルに記述されているのは数式に類似した形で与えられる問題の構造や規則のみであることに注意されたい. 大規模になり得るモデル, 例えばネットワーク問題や離散化された偏微分方程式系もその殆どが単純なパターンの構造や規則の繰り返しで構成されていることが多い. SIMPLE の集合と要素の概念はこれら一般の大規模問題をモデル記述とデータを分離して簡潔に記述することを可能にしている.

## 2. ネットワーク問題

グラフを用いた問題記述はネットワーク問題などへの応用上重要である. SIMPLE では有向グラフに対応するクラス Graph を用いた問題記述が可能である. 例えば最小コスト流問題:

$$\begin{aligned} &\text{変数} && x_{ij} \\ &\text{最小化} && \sum_{(i,j) \in \mathcal{A}} a_{ij} x_{ij} \\ &\text{制約式} && \sum_{\{j \mid (i,j) \in \mathcal{A}\}} x_{ij} - \sum_{\{j \mid (j,i) \in \mathcal{A}\}} x_{ji} = s_i, \ \forall i \in \mathcal{N} \\ & && b_{ij} \leq x_{ij} \leq c_{ij}, \quad \forall (i,j) \in \mathcal{A} \end{aligned}$$

は以下の様に記述される.

```
Graph g;   // グラフに対応するオブジェクト
Variable x(name="x",index=g.arcs);
Parameter a(name="a",index=g.arcs);
Parameter s(name="s",index=g.nodes);
Element i(set=g.nodes) // 点
  ,e(set=g.arcs) // 辺
  ,eout(set=out(g,i)) // 点 i に入る辺
```

```
  ,ein(set=in(g,i));  // 点 i から出る辺
Objective cost(type=minimize);
cost = sum(a[e] * x[e], e); //目的関数
// 制約式
sum(x[eout], eout)
                - sum(x[ein], ein) == s[i];
x[e] >= 0;
```

この場合にも，モデルには目的関数の係数の具体的な値やグラフの具体的な連結情報は含まれない．それらはデータファイルから

```
a = [1,2] 1 [2,4] 3 ...
s = [1] 1 [2] 0 ...
```

の様に与えられ，モデルとデータの分離が実現されている．

## 3. 多目的最適化

SIMPLE は上記の様にモデルを記述する他にも，ソルバの起動を明示的に指示したり，その計算結果を利用して逐次的にモデルの定義を行う機能を備えている．ここでは次の多目的最適化問題:

$$\begin{array}{ll} \text{変数} & x_1, x_2, x_3 \\ \text{最小化} & f_1 \equiv x_1, f_2 \equiv x_2, f_3 \equiv x_3 \\ \text{制約式} & (x_1-1)^2+(x_2-1)^2+(x_3-1)^2 \le 1 \end{array}$$

の希求水準法による補助最適化 [3]

$$\begin{array}{ll} \text{最小化} & z+\alpha\sum_{i=1,2,3} w_i f_i \\ \text{追加制約式} & w_i(f_i-\bar{f}_i) \le z, \quad i=1,2,3 \end{array}$$

を SIMPLE にて記述した例を示す．ただし $\bar{f}_i$ は各目的関数 $f_i$ に対して設定された希求水準．重み $w_i$ は各目的関数 $f_i$ の最小値 $f_i^*$ から

$$w_i = \frac{1}{\bar{f}_i - f_i^*}$$

として定められるものとしている．

```
Variable x1,x2,x3;
pow(x1-1,2)+pow(x2-1,2)+pow(x3-1,2) <= 1;
Set S = "1 2 3"; Element i(set=S);
Expression f(index=i,type=minimize);
// f1,f2,f3 の定義
f[1] = x1; f[2] = x2; f[3] = x3;
Parameter fmin(index=i);
// f1,f2,f3 の最小値の設定
Objective obj1(type=minimize)
         ,obj2(type=minimize)
         ,obj3(type=minimize);
obj1 = f[1]; obj2 = f[2]; obj3 = f[3];
solve(obj1); fmin[1] = f[1].val;
solve(obj2); fmin[2] = f[2].val;
solve(obj3); fmin[3] = f[3].val;
// 希求水準 (入力パラメータ)
Parameter fbar(index=i,name="fbar");
// 補助問題の定義
Objective obj(type=minimize);
Variable z;
Parameter alpha = 1.0e-6;
Parameter w(index=i); // 重み
obj = z + alpha*sum(w[i]*f[i],i);
w[i] = 1/(fbar[i]-fmin[i]); // 重み設定
w[i]*(f[i]-fbar[i]) <= z; // 追加制約式
solve(obj); // 求解
// 表示
cout << "f = " << f[i].val << "\n";
```

SIMPLE の命令 solve() はソルバを起動して求解を行うことを指示する命令であり，.val はそのオブジェクトの現在値を示す属性 (メンバ) を示す．すなわち上記での

```
solve(obj1); fmin[1] = f[1].val;
```

は f[1] を目的関数とする最適化を行い，その結果を fmin[1] に格納することを指示している．補助最適化問題はこうして得られた結果から計算された重み w の値に基いて定義され，最後の solve() の呼び出しによって実行される．最後の行は標準出力ストリーム (cout) にオブジェクトの現在値を表示させるための命令である．

例えばデータファイルから希求水準の値:

```
fbar = [1] 0.35 [2] 0.4 [3] 0.5;
```

を与えると，4 回の最適化ののち，この行の効果により

```
f = (0.357995 0.409137 0.51142)
```

という出力が得られる．

## 参考文献


[1] 茨木俊秀，福島正夫
*FORTRAN 77* 最適化プログラミング，岩波コンピュータサイエンス,1991.

[2] Dimitri P. Bertsekas
*Linear Network Optimization Algorithms and Codes*, MIT Press,1991.

[3] 中山弘隆，谷野哲三
多目的計画法の理論と応用，計測自動制御学会,1994.