# AI言語Prologによる線形計画法問題の解法プログラムの開発

# AI言語Prologによる線形計画法問題の解法プログラムの開発

# AI言語Prologによる線形計画法問題の解法プログラムの開発

吉福　功美

（受理　平成7年5月31日）

# The Development of a Program of Linear Programming Problem Solution by AI Language Prolog

Isami YOSHIFUKU

In our laboratory, the application of Prolog language to chemical engineering problems have been studied : the flowgraph making program by Prolog and the continuous system simulation program by Prolog for process control problems.

In this report, a program for linear programming problem solution by AI language Prolog, based on the treatment of extreme points, has been developed. This technique is expected as an approach to linear and nonlinear programming problems. In this study, the programming technique in Prolog language using the connection of predicates added in an appendix is shown.

## 緒　言

我々の研究室ではAI言語Prologの化学工学の諸問題への適用について幾つかの報告をしてきた。

化学工学の設計問題に現れる非線形代数方程式群の解法については，これをFlowgraphという流れ図に表現することによって解を得ることができることは既に示した[1]。このFlowgraphは複雑な問題になると人の手で作成することは大変な作業となる。これをコンピュータで行うという試みについて報告した[2]。そこではProlog言語で書かれたFG_MAKEというプログラムを作成した。

またプロセス制御問題への適用として同じくProlog言語を用いて書かれたCSPPというプログラムを作成した[3]が，これはあるプロセス制御システムのブロック線図が与えられたとき，ステップ応答曲線を得るための微分方程式および代数方程式群を求めるものである。

このようにProlog言語は数値計算よりもリスト処理等の論理演算に適した言語であることは論を待たないが，場合によっては数値計算も可能である。ここでは化学工学の生産計画問題への適用として，線形計画法と呼ばれる分野においてプログラムLP_Bを開発し，数値計算を行った結果について報告する。これはいわゆる端点が最適解の候補であるという点に着目して提案したアルゴリズムに基ずいてProlog言語で書かれたものである。このような試みは線形計画法並びに非線形計画法等の特殊な数値計算問題の解法へのアプローチとなることが期待できる。

## 1. 問題の提起

次の生産計画問題を取り上げる[4]。ある会社で製品A，Bを製造するのにエネルギー源として重油か電力のいずれか一方のみが利用でき，重油を使用する場合はそれぞれ製品1kgあたり4および5kℓ必要であり，電力の場合にはそれぞれ9および4kWh必要とする。さらに製品A，Bを製造するのに労力がそれぞれ3人日および10人日必要である。ところがその会社で現在利用できるのは重油200kℓ，電力360kWh，労力が300人日までであるとする。製品AおよびBは1kgについてそれぞれ7万円および12万円の利益を生む。このとき利益を最大にする製品A，Bの生産高を求めよ。

いま，製品A，Bの生産高をx，yとすると，上の問題は制約条件

$$x >= 0 \quad (1)$$
$$y >= 0 \quad (2)$$
$$4x+5y =< 200 \quad (3)$$
$$9x+4y =< 360 \quad (4)$$
$$3x+10y =< 300 \quad (5)$$

の下で Z = 7x+12y を最大にするような x, y を見いだせということになる。これを

$$x, y : z = 7x+12y = \max \quad (6)$$

と表現する。

このような問題の解法についてはすでにSimplex法および図解法が知られている。ここではこの線形計画法問題に対して新しい解法のアルゴリズムを開発し，Prolog言語でプログラムを書くことを試みる。

## 2. 解法のアルゴリズム

ここでは線形計画法問題に対する解法として Simplex法，図解法に次ぐ別な方法を提案し，そのアルゴリズムを Prolog 言語でプログラムにする。図解法での2直線の交点や Simplex 法では基低解と呼ばれるもの，いわゆる端点（extreme Point）が最適解の候補であることに着目し，次のようなアルゴリズムを提案する。

(a) 不等式を全て等式に変換し，その全ての2個の式の組み合わせについてこれを解き，候補解を求める。
(b) この候補解について不等式の条件を当3てはめ，条件にはずれるものは除外する。
(c) 残った候補解の中で最適解を選ぶ。

## 3. Prolog言語での記述

### 3. 1 データベース

次に上述のアルゴリズムを Prolog 言語で表す。ここでは Lifeboat 社の Arity Prolog を用いた。まず，プログラム全体として次のような構造を作る。

データベース---> プログラム本体---> 解

すなわちベータベース，プログラム本体を別々に作っておき，プログラム本体にデータベースを組み込んだとき，解が得られるという構造である。まず，データベースは ab. ari という名前にするが，本例題の場合は次のようになっている。

```
% ab. ari
lst ([[1, 0, 62, 0], [0, 1, 62, 0], [4, 5, 60, 200], [9,
    4, 60, 360], [3, 10, 60, 300]]).
lstz ([[7, 12], [x1, x2], max]).
```

ここで述語 lst ([L1]) において引数のリスト L1 は5個のリスト要素から成っている。1番目の要素 [1, 0, 62, 0] は(1)式 x >= 0 を表現していて，1は x の係数を，0は y の数を，62は > の ASCII コードを，0は不等式の右辺の値を示している。ここで >= の代わりに > のコードを用いた。=< も < のコード60を用いることにする。2番目以下も同様で(2)－(5)式を表している。

次に述語 lstz ([LZ]) では引数のリスト LZ は3個の要素から成るが，1番目の要素 [7, 12] は(6)式の x, y の係数を，2番目 [x1, x2] は x, y を x1, x2 としていることを表している。最後の要素 max は本問題が最大値を求めることを表現している。

### 3. 2 プログラム本体

このデータベースを迎えるプログラム本体を作成するに当たり，付録の述語群を参照した。これらは必要に応じて作成し集めたもので述語の定義，内容および適用例から成っている。

本プログラムは次の通りである。

```
consult ('ab'), lst (L1),
lst (LZ), LZ=[Z1, Z2, Z3],
combi_a (L1, L2),
det_ 2 (L2, L3),
select_ 8 (L3, L1, L4),
case ([
Z3=min->(cal_z (L4, LZ, L5),
    min_num 3 (L5, Min, L6),
Z3=max->(cal_z (L4, LZ, L5),
    max_num 3 (L5, Max, L6))]),
write ('L6='), write(L6).
```

1，2番目はデータベースの取り込みで L1，LZ が作られ，さらに Z1，Z2，Z3 が作られている。ここで consult ('ab') は使用した Prolog に組み込まれた述語[5]で，ファイルから ab. ari を読み込むものである。3番目の combi_a (L1, L2) はその詳細は付録に記載してあるが，リストL1の要素のペアを要素とするリストL2を作成するものである。すなわち不等式を等式にしたときの2直線をペアにしている。4番目の det_ 2 (L2, L3) はそれぞれの2直線の交点の座標を求めるもので，ここ迄がアルゴリズム(a)に対応している。

次に5番目の select_ 8 (L3, L1, L4) はアルゴリズ

ム(b)に対応し，L3の中で不等式の条件L1に適合するものL4を得るものである。最後にアルゴリズム(c)であるが，6番目はリストL4の中の各変数x1, x2を目的関数(6)式に代入してzの値を求め(cal_z(L4, LZ, L5))，さらにその中で例えば最大値を求めるものである。ここでは最小値，最大値の2通りとなっている。解はL6である。このプログラムをLP_Bと名付ける。

### 3. 3 適用例

本例題のデータベースab. ariにプログラムLP-Bを適用した結果は次の通りであった。

2直線の交点の座標のリストは

L3=[[0,0], [0,40], [0, 90], [0, 30], [50, 0], [40, 0], [100, 0], [34.4, 12.4], [20, 24], [30.7, 20.7]]

で，条件に合うものは

L4=[[0, 0], [0, 30], [40, 0], [20, 24]]

これらを代入した(6)式の値は

L5=[[0,0,0], [360,0,30], [280, 40, 0], [428, 20, 24]]

解リストはL6=[428, 20, 24]である。すなわちx1=20，x2=24が解で，最大値Max=428である。

この他幾つかの線形計画法の問題に本プログラムを適用したが，簡単に解が得られた。線形計画法のような特殊な数値計画には，このProlog言語で書かれたプログラムは有用であると考えられる。

## 結　論

線形計画法問題の解法について，新しくアルゴリズムを提案し，プログラムLP_Bを開発した。これはProlog言語で書かれており，Prolog言語の数値計算法への適応性を検討することを目的とするものである。線形計画法のような特殊な数値計算には十分適合することが分かった。

Prolog言語には普通，数十個の組み込み述語が備わっているが，不十分である。本報告には付録として，述語群すなわち述語について定義，その内容および適用例を集めたものを準備したが，実際にProlog言語でプログラムを作成する場合に便利であるというよりは必要不可欠であると考えられる。

## 引用文献


1) 吉福功美：化学工学論文集，6，546 (1980)
2) Yoshifuku, I. & R. Fukumura : J of Chemical Engineering of Japan, 24, 677 (1991).
3) Yoshifuku, I., M. Motoura, K. Ijichi : Proceedings of Taipei-Kyushu Joint Symposium on Chemical Engineering, 243 (1994).
4) 化学工学会："化学工学のための応用数学(丸善)"，80 (1994)
5) Lifeboat : "Users Manual Arity Prolog Ver. 5"，139 (1987)


### 付録　述語群(述語の定義，内容および適用例から成っている)

(1) append (L1, L2, L3) -→ L3は2個のリストL1, L2を連結したリストである。

```
append ([], L2, L2).
append ([L | L1], L2, [L | L3]) : -
  append (L1, L2, L3).
```

〈eg〉 L1=[1, 2, 5, 7], L2=[11, 15, 18]のときはL3=[1, 2, 5, 7, 11, 15, 18]となる。

(2) cal_z (L1, LZ, L2) -→ x1, x2の組のリストL1と式a×1+b×2=minを表すリストLZがある。x1, x2を式に代入したときの値を第1要素とするリストL2を作る。

```
cal_z (L1, LZ, L2) : - cal_z (L1, LZ, [], L2).
cal_z ([], LZ, L2, L2).
cal_z (L1, LZ, LS, L2) : -
  L1=[H | T1], H=[H1, H2],
  LZ=[A, B, C], A=[A1, A2],
  Z is A1*H1+A2*H2,
  LL=[Z, H1, H2], append (LS, [LL], LT),
call_z (T1, LZ, LT, L2).
```

〈eg〉 式-7x1 -12x2をminとすることをリストLZ=[[-7, -12], [x1, x2], min]と表す。x1, x2のデータの組L1=[[0, 0], [0, 30], [40, 0], [20, 24]]があり，x1, x2を式に代入したときの値を第1要素とするリストL2はL2=[[0,0,0], [-360,0,30], [-280, 40, 0], [-428, 20, 24]]である。

(3) combi_b (A, L1, L2) -→ AとリストL1の要素とのペア要素とするリストL2を作る。

```
combi_b (A, L1, L2) : -
  combi_b (A, L1, [], L2).
combi_b (A, [], LL2, LL).
combi_b (A, L1, LS, L2) : -
                    L1=[H | T], LL=[A, H],
  append (LS, [LL], LT),
  combi_b (A, T, LT, L2).
```

〈eg〉 A=v, L1=[a, b, c, d]のときは

L2=[[v, a], [v, b], [v, c], [v, d]]である。

(4) combi_a (L1, L2) -→リストL1の要素のペアを要素とするリストL2を作る。

```
combi_a (L1, L2) : - combi_a (L1, [], L2).
combi_a ([], L2, L2).
combi_a (L1, LS, L2) : - L1=[H | T],
  combi_b (H, T, LL), append (LS, LL, LT),
  combi_a (T, LT, L2).
```

〈eg〉 L1=[a, b, c, d]のときはL2=[[a, b], [a, c], [a, d], [b, c], [b, d], [c, d]]である。

(5) det_1 (L1, L2, L3) -→2個の不等式 a1x+a2y=〈a3, b1x+b2y=〈b3 に対してこれらを等式としたとき，2直線の交点の座標の計算

```
det_1 (L1, L2, L4) : - det_1 (L1, L2, L3, L4).
det_1 (L1, L2, L3, L4) : -
  L1=[A1, A2, A3, A4], L2=[B1, B2, B3, B4],
  C is A1*B2-A2*B4, C1 is A4*B2-A2*B4,
  C2 is A1*B4-A4*B1, L3=[C, C1, C2],
  D1=C1/C, D2 is C2/C, L4=[D1, D2].
```

〈eg〉 4x-5y=〈8, 2x-4y〉=2 のときはL1=[4, -5, 60,8], L2=[2,-4, 62, 2]で，このときはL3=[3.666, 1.333]すなわち交点の座標は x=3.666, y=1.333である。

(6) det_2 (L1, L2) -→2個の不等式の組が幾つかある時（そのリストL1）2直線の交点を要素とするリストL2を求める。

```
det_2 (L1, L2) : - det_2 (L1, [], L2).
det_2 ([], L2, L2).
det_2 (L1, LS, L2) : - L1=[H1 | T1], H1=
  [L11, L12], det_1 (L11, L12, LL),
  append (LS, [LL], LT), det_2 (T1, LT, L2).
```

〈eg〉 2個の不等式組2組がある。x1=〈0, x21〉=0 とx=〈0, 3x+10y=〈300。これはL1=[[1, 0, 62,0], [0,1,62,0]], [[1, 0, 60, 0], [3, 10, 60, 300]]と表す。これらを等式にするときそれぞれの交点の座標はL2=[[0, 0], [0, 30]]すなわち点 x=0, y=0 と点 x=0, y=30 である。

(7) member (E, L) -→EはリストLの要素である。

```
member (X, [X | T]).
member (X, [Y | T]) : - member (X, T).
```

〈eg〉 E=1はリストL=[2, 1, 3]の要素である。

(8) max_num 3 (L1, Max, A) -→1番目が数字であるリストを要素とする2重リストL1について，その数字が最大のものAを探す。

```
max_num 3 (L1, Max, A) :-
  max_num 3 (L1, Max),
    member (A, L1), A=[max, _, _].
max_num 3 ([[X, _, _]], X) : - !.
max_num 3 ([H | L], Max) : -
  max_num 3 (L, M), H=[H1 | T],
  ifthenelse (H1>M, Max is H1, Max is M), !.
```

〈eg〉 L1=[[1,a,2], [3,b,3], [-5, d, 4], [6, r, 3], [-8, w, 1], [10, f, 5], [5, g, 0]]の要素リストの1番目の要素で最大のものはA=[10, f, 5]であり，最大値 Max は10である。

(9) mix_num 3 (L1, Min, A) -→1番目が数字であるリストを要素とする2重リストL1について，その数字が最大のものAを探す。

```
min_num 3 (L1, Min, A) : -
min_num 3 (L1, Min),
  member (A, L1), A=[Min, _, _].
min_num 3 ([[X, _, _]], X) : - !.
min_num 3 ([H | L], Min) : -
  min_num 3 (L, M), H=[H1 | T],
  ifthenelse (H1<M, Min is H1, Min is M),!.
```

〈eg〉 L1=[[1,a,2], [3,b,3], [-5, d, 4], [6, r, 3], [-8,w,1], [10,f,5], [5,g,0]]の要素リストの1番目の要素で最小のものはA=[-8,w,1]であり，最大値 Min は-8である。

(10) select_7 (L1, LA, L2) -→不等式 ax+by〈cをリストにしたLA=[a, b, 60, c]と点x, yを表すリストを要素とするリストL1があるとき，不等式を満足する点 x, y のリストL2を作る。

```
select_7 (L1, LA, L2) : -
  select_7 (L1, LA, [], L2).
select_7 ([], LA, L2, L2).
select_7 (L1, LA, LS, L2) : -
  L1=[H | T], H=[L13, L14],
  LA=[A1, A2, A3, A4],
  X3 is A1*L13+A2*L14,
case ([
  (A3=60, X3=〈A4)-〉(LY=H,
        append (LS, 0 [LT], LL)),
  (A3=62, X3〉=A4-〉(LT=H,
        append (LS, [LT], LL))
  | LL=LS]),
  select_7 (T, LA, LL, L2).
```

〈eg〉 不等式 9x+4y=〈360をリストL1=[9, 4, 60,

360]とし，点[x, y]を要素とするリストL1=[[0, 0], [0, 40], [0, 90], [0.30], [50, 0], [40, 0], [100, 0], [4, 4], [12, 4], [20, 24], [30.7], [20.7]] があるとき不等式を満足する点のリストはL2=[[0, 0], [0, 40], [0, 90], [0, 30], [40, 0], [4, 4], [12, 4], [20, 24], [30.7, 20.7]] である。すなわち点 x=0, y=0 や点 x=0, y=4 等は不等式を満足する。

(11) select_8 (L1, LA, L2) -> 幾つかの不等式をリストにした場合の selext_7 と同じもの
select_8 (L2, [], L2).

select_8 (L1, LA, L2) : - LA=[H | T],
select_7 (L1, H, LL), select_8 (LL, T, L2).

〈eg〉 不等式 4x+5y=〈200, 9x+4y=〈360, 3x+10y=〈300 に対して次の点 x, yのリスト
L1=[[0, 0], [0, 40], [0, 90], [0, 30], [50, 0], [40,0], [100,0], [4.4, 12.4], [20, 24], [30.7, 20.7]] の中でその全てを満足する点のリスト L2 は
L2=[0,0], [0, 30], [40, 0], [4.4, 12, 4], [20, 4]] である。