관계형 데이터베이스를 이용한 배낭문제 해법기의 구현

서창교*, 송구선**

Development of Knapsack Problem Solver Using Relational DBMS

Chang-Kyo Suh* · Gu-Seon Song**

ABSTRACT

Knapsack problems represent many business application such as cargo loading, project selection, and capital budgeting. In this research we developed a knapsack problem solver based on Martello-Toth algorithm using a relational database management system on the PC platform. The solver used the menu-driven user interface. The solver can be easily integrated with the database of decision support system because the solver can access the database to retrieve the data for the model and to store the result directly.

1. 서 론

배낭에 넣을 수 있는 능력이 주어져 있고, 이를 초과하지 않는 범위 내에서 n 개의 물건 중에서 어떤 물건을 선정해야 할 것인가와 관련된 배낭문제는 보통 이용할 수 있는 물건의 량은 제한되어 있고, 그 물건들은 중량, 부피, 유용성 등 각각의 특징을 지니고 있다. 따라서, 이 때의 목적은 주어진 배낭의 크기에 비해서 유용성을 가장 크게 할 수 있도록 여러 물건의 최적 조합을 구하는 것이다[4].

이러한 배낭문제는 정수계획법의 특수한 경우로서 화물의 선적시 화물선의 용량이 정해져 있으므로 가장 가치 있는 화물을 선적해야 하는 화물 선적 문제, 다수 기간 동안 투자 재원이 제한되어 있을 때 전기간에 걸쳐 총이익을 최대화하기 위한 기간별 투자 대안을 선택해야 하는 자본예산문제, 예상 이용가치에 따라 과학 잡지를 구매할 때 현재 및 장래 예상 자금을 최적 할당해야 하는 도서관 잡지 구입 결정 등 제한된 제약자원하에서 주어진 대안들을 선정해야 하는 다양한 분야[7]에 이용되고 있다.

* 계명대학교 경영정보학과 조교수
** 포항선린전문대학 사무자동화과 조교수

일반적으로 사용되는 배낭문제의 해법은 (1) 열거법(implicit enumeration), (2) 동적계획법(dynamic programming), (3) 네트워크 접근법(network approaches), (4) 일반 라그랑지법(generalized Lagrangian methods) 등을 들 수 있다[6].

본 연구는 실제적인 자원 배분과 관련된 경영상의 문제(예를 들면, 장기 연구개발 계획을 위한 자원 배분[9], 중장기 투자 계획[10] 등)들을 풀기 위한 의사결정지원시스템의 한 모듈로 관계형 데이타베이스를 이용하여 개발된 배낭문제 해법기를 공개하여 최근에 활성화되고 있는 수리계획 소프트웨어의 국내 자체 개발 능력의 확보[1, 2, 3]에 기여하고자 한다.

관계형 데이타베이스를 이용한 모델베이스의 해법기 개발은 의사결정지원시스템이 데이타베이스를 직접 접근하여 필요한 수리모형의 기초 자료를 가져오고, 모형의 해법결과를 직접 데이타베이스에 다시 저장하여 다른 수리모형에서 이들 자료를 재사용할 수 있도록 지원하여 의사결정지원시스템 구성요소간의 접속을 원활하게 하여 줄 뿐만 아니라, 상용화된 수리계획 소프트웨어에 보편화되고 있는 데이타베이스 접속능력[8]을 활용할 수 있어 효과적이다.

본 논문의 구성은 다음 장에서 배낭문제 해법기의 구조를 설명하고, 제3장에서는 개발된 해법기의 특징을 논하였다. 또한 개발된 해법기의 원시프로그램을 부록에 첨부하여 유사한 응용 프로그램의 개발에 이식하여 사용할 수 있도록 하였다.

2. Martello-Toth의 알고리즘과 프로그래밍

0-1 배낭문제의 일반적인 형태는 다음과 같다.

최대화 $P=\sum_{j=1}^{m} p_j x_j$

제약 $W=\sum_{j=1}^{m} w_j x_j \leq M$

$x_j=0$ 또는 $1 (j=1,2, \cdots, m)$

여기서 p = 현재 해$[x_j=(x_1, x_2, \cdots, x_m)]$와 관련된 이익$(\sum_{j=1}^{m} p_j x_j)$

Π = 현재 가장 좋은 해 $[X_j=(X_1, X_2, \cdots, X_m)]$ 와 관련된 이익$(\sum_{j=1}^{m} p_j X_j)$

1. 제1단계(초기화)

$p_j..$와 w_j를 단위당 이익 p_j/w_j에 따라 내림차순으로 정리한다.

$IP=\sum_{j=1}^{l} p_j$를 계산한다. 여기서 l는 $IW=\sum_{j=1}^{l} w_j \leq M$인 최대 지수(index)

만약 $IW=M$이면 $\Pi=IP$, $(X_j=1, j=1, 2, \cdots, l)$, $(X_j=0, j=l+1, \cdots, m)$가 최적해이므로 끝낸다.

그렇지 않으면, $Min_j=\min\{w_k | j<k \leq m\}$, $j=1, 2, \cdots,$ m-1, $Min_m=M+1$을 계산한다. 또한, Martello-Toth의 하한정리[5]에 따라 $B_1=\sum_{j=1}^{l} p_j + [(M-\sum_{j=1}^{l} w_j) p_{l+2}/w_{l+2}]$와 $B_2=\sum_{j=1}^{l} p_j + [p_{l+1} - (w_{l+1} - (M-\sum_{j=1}^{l} w_j)) p_l/w_l]$일 때 0-1 배낭문제에 대한 상한해는 $UB_2=\max\{B_1, B_2\}$가 되므로 Lim=UB_2, $p=\Pi=0$, $(x_j=0, j=1, \cdots, m)$, $i=1$, $l_{old}=m$라 두고 제4단계로 간다.

```
CASE kstep = '1'
  FOR j = 1 TO karraySize
    IF kIW + kaW[j] <=kM
      kIW += kaW[j]
    ELSE
      kl = j - 1
      EXIT
    ENDIF
```

```
  NEXT

  FOR j=1 TO kl
    kIP += kaP[j]
  NEXT
  ksolution = 0
  IF kIW = kM
    ksolution = kIP
    FOR j=1 TO kl
      kFixedX[j] = 1
    NEXT
  IF kl < karraySize
    FOR j = kl + 1 TO karraySize
      kFixedX[j] = 0
    NEXT
    ENDIF
    EXIT          //kfixedX의 값이 해이다.
  ENDIF
aTemp := ARRAY(karraySize - 1)
FOR j = 2 TO karraySize
    @ 6, 10 say "j="+substr(str(j),-3)
    ACOPY(kaW, aTemp, j, karraySize-j+1)
    ASORT(aTemp, , , {|A, B| a>b})
    kaMin[j-1] := ATAIL(aTemp)
    ASIZE(aTemp, karraySize - j)
NEXT
kaMin[karraySize] = kM+1
STORE 0 TO kp, ksolution
ki = 1
klold=karraySize
kB1=kIP+(kM-kIW)*kaP[kl+2]/kaW
      [kl+2]
kB2=kIP+kaP[kl+1]-(kaW[kl+1]-kM
      +kIW)*kaP[kl]/kaW[kl]
kUB2 = INT(MAX(kB1, kB2))
kLim = kUB2
FOR j = 1 TO karraySize
  kaX[j] = 0
NEXT
kstep = '4'
LOOP
```

2. 제2단계(i 번째 요소를 현재해에 저장)

만약 $w_i \leq M$이면 제3단계로 간다.

그렇지 않고 $\Pi \geq p+[Mp_{i+1}/w_{i+1}]$이면 제5단계로 가고, 아니면 $i=i+1$로 놓고 제2단계를 반복한다.

```
CASE kstep = '2'
  IF kaW[ki] <= kM
    kstep = '3'
    LOOP
  ENDIF
  kValue1=kp+kM*kaP[ki+1]/kaW
          [ki+1]
  IF ksolution > = kValue1 //=kp+kM*
          //kaP[ki + 1]/kaW[ki+1]
    kstep = '5'
    LOOP
  ELSE
    ki += 1
    kstep = '2'
    LOOP
  ENDIF
```

3. 제3단계(새로운 현재해를 구성)

$IP=\overline{p}_i+\sum_{j=z_i}^{i} p_j$를 계산한다.

여기서 l은 $IW=\bar{w}_i+\sum_{j=z_i}^{l} w_j \le M$이고, $l \le m$인 최대 지수.

(만약, $w_{\bar{z}_i} > M$이면 $l=\bar{z}_i-1$)

여기서는 두 가지 경우가 있을 수 있다.

(a) $IW < M$이고 $l < m$ 일 때 :

만약, $\Pi \ge p+IP+\overline{UB}$이면 제6단계로 그렇지 않으면 제4단계로 간다.

여기서 $\overline{UB}=\max\{[(M-IW)p_{l+2}/w_{l+2}],\ [p_{l+1}-(w_{l+1}-(M-IW))p_l/w_l]\}$

(b) $IW = M$이거나 $l = m$일 때 :

만약, $\Pi \ge p+IP$이면 제6단계로 간다.

그렇지 않으면 $\Pi=p+IP$, $(X_j=x_j,\ j=1, \cdots, i-1)$, $(X_j=1,\ j=i,\ i+1,\ \cdots,\ l)$, $(X_j=0,\ j=l+1\ \cdots,\ m)$로 둔다.

만약 Π=Lim이면 끝내고, 그렇지 않으면 제6단계로 간다.

```
CASE kstep = '3'
  kt = kaZbar[ki]
  IF kaW[kt] > kM
    kl = kaZbar[ki] - 1
  ELSE
    kIP = kaPbar[ki]
    kIW = kaWbar[ki]
    FOR j= kt TO karraySize
      IF kIW + kaW[j] <= kM
        kIP += kaP[j]
        kIW += kaW[j]
        kl := j
      ELSE
        EXIT      // 해를 구했다.
      ENDIF
    NEXT
  ENDIF
  IF kIW < kM . AND. kl < karraySize
    IF kl+2 <= karraySize
      kB3 = (kM-kIW)*kaP[kl+2]/kaW
            [kl+2]
    ELSE
      kB3 = 0
    ENDIF
    kB4 = kaP[kl+1]-(kaW[kl+1]-
          (kM-kIW))*kaP[kl]/kaW[kl]
    kUB = INT(MAX(kB3, kB4))
    IF ksolution > kp + kIP + kUB
      kstep = '6'
      LOOP
    ELSE
      kstep = '4'
      LOOP
    ENDIF
  ENDIF
  IF kIW =kM . OR. kl = karraySize
    IF ksolution >= kp + kIP
      kstep = '6'
      LOOP
    ELSE
      ksolution := kp + kIP
      FOR j=1 TO ki - 1
        kFixedX[j] = kaX[j]
      NEXT
      FOR j = ki TO kl
        kFixedX[j] = 1
      NEXT
      IF kl+1 <= karraySize
        FOR j = kl + 1 TO karraySize
          kFixedX[j] = 0
        NEXT
      ENDIF
      IF ksolution = kLim
```

```
        EXIT      // 해를 구했다.
      ELSE
        kstep = '6'
        LOOP
      ENDIF
    ENDIF
  ENDIF
```

4. 제4단계(현재해를 저장)

$M=M-IW$, $p=p+IP$, ($x_j=1$, $j=i$, · · ·, l)라 둔다.

$\bar{w}_i=IW$, $\bar{p}_i=IP$, $\bar{z}_i=l+1$, ($\bar{w}_j=\bar{w}_{j-1}-w_{j-1}$, $\bar{p}_j=\bar{p}_{j-1}-p_{j-1}$, $\bar{z}_j=l+1$ for $j=i+1$, · · ·, l), ($\bar{w}_j=\bar{p}_i=0$, $\bar{z}_j=j$ for $j=l+1$, · · ·, l_{old})를 계산한다. $l_{old}=l$로 둔다.

이 때 다음의 세 가지 경우가 있다.

(a) $l < m-2$인 경우 : $i=l+2$로 둔다.

만약 $M < Min_{i-1}$이면 제5단계로 그렇지 않으면 제2단계로 간다.

(b) $l=m-2$인 경우

만약, $M \geq w_m$이면, $M=M-w_m$, $p=p+p_m$, $x_m=1$, $i=m-1$로 두고 제5단계로 간다.

(c) $l=m-1$인 경우 : $i=m$으로 두고 제5단계로 간다.

```
CASE kstep = '4'
  kM -= kIW
  kP += kIP
  FOR j = ki TO kl
    kaX[j] = 1
  NEXT
  kaPbar[ki] = kIP
  kaWbar[ki] = kIW
  kaZbar[ki] = kl + 1
  FOR j = ki + 1 TO kl
    kaPbar[j] = kaPbar[j-1] - kaP[j-1]
    kaWbar[j] = kaWbar[j-1] - kaW[j-1]
    kaZbar[j] = kl + 1
  NEXT
  IF kl + 1 <= klold
    FOR j = kl + 1 TO klold
      kaPbar[j] = 0
      kaWbar[j] = 0
      kaZbar[j] = j
    NEXT
  ENDIF
  klold = kl
  IF kl < karraySize - 2
    ki = kl + 2
    IF kM < kaMin[ki-1]
      kstep = '5'
      LOOP
    ELSE
      kstep = '2'
      LOOP
    ENDIF
  ENDIF
  IF kl = karraySize - 2
    IF kM >= kaW[karraySize]
      kM -= kaW[karraySize]
      kp += kaP[karraySize ]
      kaX[karraySize] = 1
    ENDIF
    ki = karraySize - 1
    kstep = '5'
    LOOP
  ENDIF
  IF kl = karraySize - 1
    ki = karraySize
```

```
    kstep = '5'
    LOOP
  ENDIF
```

5. 제5단계(현재의 최적해를 저장)

만약 $\Pi < p$이면 $\Pi=p,(X_j=x_j,\ j=1,\ \cdots,\ m)$로 두고, 이때 Π=Lim이면 끝낸다. 그렇지 않으면서(즉, $\Pi \geq p$ 또는 $\Pi \neq$Lim) $x_m=1$이면 $M=M+w_n,\ p=p-p_m,\ x_m=0$로 둔다.
제6단계로 간다.

```
CASE kstep = '5'
  IF ksolution < kp
    ksolution = kp
    FOR j=1 TO karraySize
      kFixedX[j] := kaX[j]
    NEXT
    IF ksolution = kLim
      EXIT // 해를 구했다.
    ENDIF
  ENDIF
  IF kaX[karraySize] = 1
    kM+= kaW[karraySize]
    kp -= kaP[karraySize]
    kaX[karraySize] = 0
  ENDIF
  kstep = '6'
  LOOP
```

6. 제6단계(역추적 : Backtrack)

$x_k=1$인 최대의 $k(<i)$ 값을 찾는다. 만약, 그러한 k가 존재하지 않으면 끝낸다.
그렇지 않으면 $R=M,\ M=M+w_k,\ p=p-p_k,\ x_k=0$로 둔다. 만약 $R \geq Min_k$이면, $i=k+1$로 놓고 제2단계로 간다. 그렇지 않으면 $i=k,\ n=k+1$로 놓고 제7단계로 간다.

```
CASE kstep = '6'
  kk = 0
  FOR j=1 TO ki - 1
    IF kaX[j] = 1
      kk = j
    ENDIF
  NEXT
  IF kk = 0
    EXIT     // 해를 찾았다.
  ENDIF
  kR = kM
  kM += kaW[kk]
  kP -= kaP[kk]
  kaX[kk] = 0
  IF kR >= kaMin[kk]
    ki = kk + 1
    kstep = '2'
    LOOP
  ELSE
    ki = kk
    kn = kk + 1
    kstep = '7'
    LOOP
  ENDIF
```

7. 제7단계(n번째 요소를 k번째 요소와 대체)

만약 $n > m$이거나 $\Pi \geq p+[Wp_n/w_n]$이면 제6단계로 간다. 그렇지 않으면 $D=w_n - w_k$로 둔다. 이 때 다음의 세 가지 경우가 있다.
(a) D=0인 경우 : n=n+1로 놓고 제7단계를

반복한다.

(b) D〉0인 경우 : 만약 $D > R$이거나 $\Pi \geq p+p_n$이면 $n=n+1$로 놓고 제7단계를 반복한다. 그렇지 않으면 $\Pi=p+p_n$, $(X_j=x_j,\ j=1, \cdots, k)$, $(X_j=0,\ j=k+1, \cdots, m,\ j\neq n)$, $X_n=1$로 놓는다. 그리고 Π=Lim이면 끝내고 그렇지 않으면 $R=R-D$, $k=n$, $n=n+1$로 놓고 제7단계를 반복한다.

(c) D〈0인 경우 : $T=R-D$로 둔다. 만약 $T < Min_n$이면 $n=n+1$로 놓고 제7단계를 반복한다. 그렇지 않고, $\Pi \geq p+p_n+[Tp_n/w_n]$이면 제6단계로 가고, 아니면, $M=M-w_n$, $p=p+p_n$, $x_n=1$, $i=n+1$, $\bar{w}_n=w_n$, $\bar{p}_n=p_n$, $\bar{z}_n$=n+1, $(\bar{w}_j=\bar{p}_j=0$, $z_j=j$ for $j=n+1, \cdots, l_{old})$, $l_{old}=n$으로 놓고 제2단계로 간다.

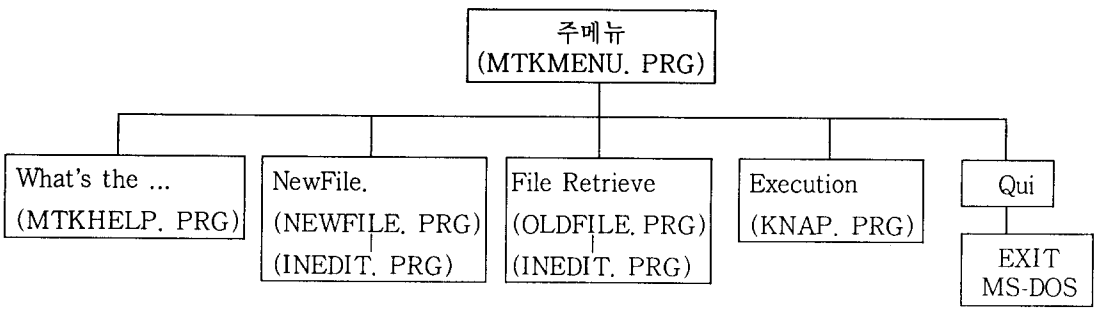
```
CASE kstep = '7'
  IF kn > karraySize
    kstep = '6'
    LOOP
  ENDIF
  kValue1=kP+kM*kaP[kn]/kaW[kn]
  IF ksolution >= kValue1
    kstep = '6'
    LOOP
  ENDIF
  kD = kaW[kn] - kaW[kk]
  IF kD = 0
     kn += 1
     kstep = '7'
     LOOP
  ENDIF
  IF kD > 0
    IF kD > kR
      kn += 1
      kstep = '7'
      LOOP
    ENDIF
    IF ksolution >= kp + kaP[kn]
      kn += 1
      kstep = '7'
      LOOP
    ENDIF
    ksolution = kp + kaP[kn]
    FOR j=1 TO kk
      kFixedX[j] = kaX[j]
    NEXT
    FOR j = kk + 1 to karraySize
      kFixedX[j] = 0
    NEXT
    kFixedX[kn] = 1
    IF ksolution = kLim
      EXIT      // 해를 찾았다.
    ELSE
      kR -= kD
      kk = kn
      kn += 1
      kstep = '7'
      LOOP
    ENDIF
  ENDIF
  IF kD < 0
    kTT = kR - kD
    IF kTT < kaMin[kn]
    kn += 1
    kstep = '7'
    LOOP
  ENDIF
  kValue1=kp+kaP[kn]+kTT*kaP[kn]
          /kaW[kn]
```

```
IF ksolution >= kValue1
  kstep = '6'
  LOOP
ENDIF
kM -= kaW[kn]
kp += kaP[kn]
kaX[kn] = 1
ki = kn + 1
kaWbar[kn] = kaW[kn]
kaPbar[kn] = kaP[kn]
kaZbar[kn] = kn + 1
IF kn + 1 <= klold
  FOR j = kn+1 to klold
    KaWbar[j] = 0
    kaPbar[j] = 0
    kaZbar[j] = 0
  NEXT
ENDIF
  klold = kn
  kstep = '2'
  LOOP
  ENDIF
ENDCASE
```

3. 개발된 해법기의 기능과 특징

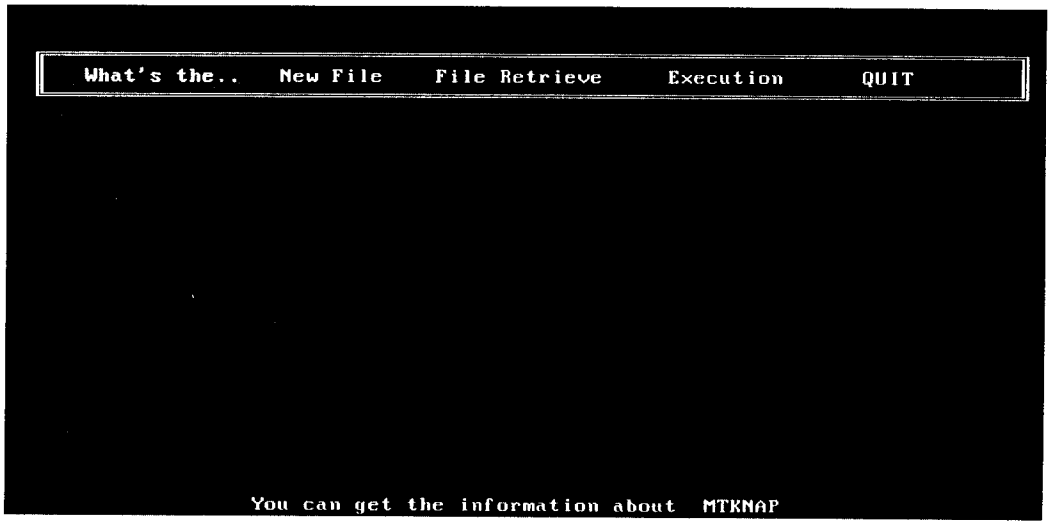
1. 기능

개발된 해법기는 초기 메뉴가 "What's the.", "New File", "File Retrieve" "Execution", "Quit"로 구성된 메뉴기반 인터페이스를 가지고 있다. 본 해법기의 메뉴 구조는 〈그림 1〉과 같으며, 각 메뉴에 해당되는 프로그램 소스화일명은 괄호 속에 포함되어 있다. 또한 초기 화면은 〈그림 2〉와 같다.



〈그림 1〉 해법기의 메뉴 구조

먼저, 각 화일의 기능을 보면 MTKMENU. PRG는 초기 화면을 표시하여 메뉴 막대로 각 메뉴를 선택하도록 해주며, "Quit" 메뉴를 직접 지원한다. NEWFILE. PRG는 새로운 0-1 배낭문제를 입력할 때 사용되며, OLDFILE. PRG는 기존의 데이터베이스 화일을 검색하여 풀다운 형태로 화면에 표시하고, 이 중 원하는 화일을 메뉴 막대로 선택하면 그zzzzz면에 표시해 준다. INEDIT. PRG는 NEWFILE. PRG와 OLDFILE. PRG에서 자료의 화면 표시를 실제 담당하며 편집과 저장이 가능케 하는 기능을 한다. KNAP. PRG는 본 해법기의 핵심으로 Martello-Toth 해법에 따라 문제의 해를 구하여 그 결과를 화면에 표시하는 프로그램이다.



〈그림 2〉 해법기의 초기화면

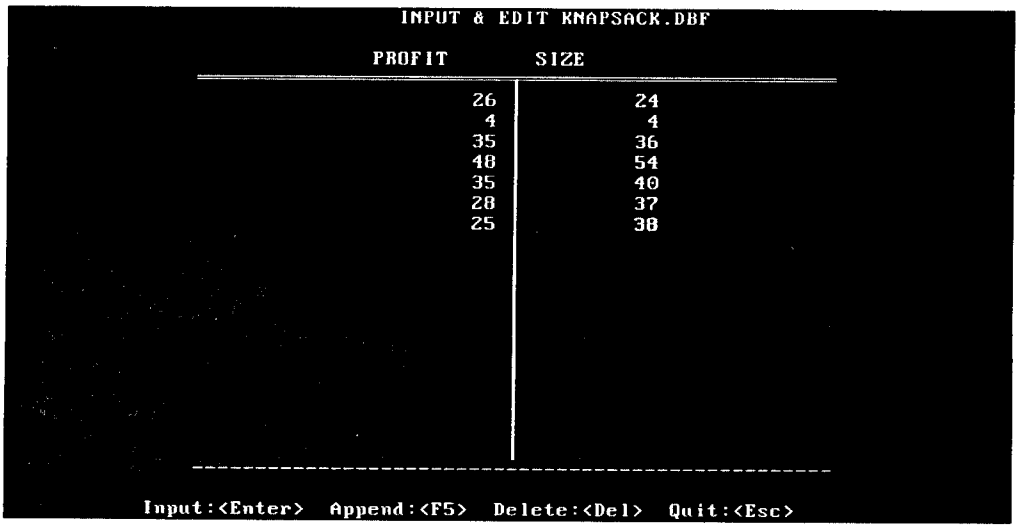
각 메뉴의 기능은 다음과 같다.

(a) What's the...

해법기의 기능과 사용 방법에 대한 설명으로서 사용도중 의문나는 점을 찾아볼 수 있다.

(b) New File

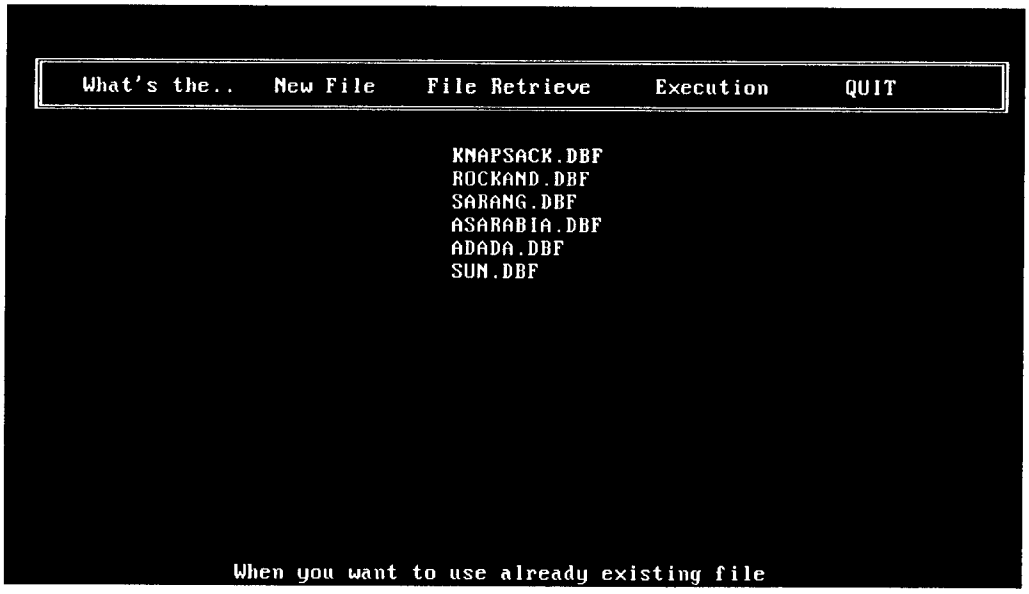
해법기에서 새로운 문제를 구성할 때 사용한다. 이 메뉴를 선택하면 화일의 이름을 묻는 부메뉴가 나오고, 이 때 화일명을 입력하면 입력된 화일명이 이미 존재하는지를 확인하여 화일명이 존재하면 새로운 화일명을 입력토록 요구하고, 그렇지 않으면 변수의 개수를 입력하도록 요구한다. 변수의 개수를 입력하면 〈그림 3〉과 같은 자료편집기가 화면에 나타나 이 편집기를 이용하여 자료를 입력하고 필요시 수정할 수 있다. 자료입력을 마치면 상위화면으로 되돌아온다.



〈그림 3〉 자료편집 화면

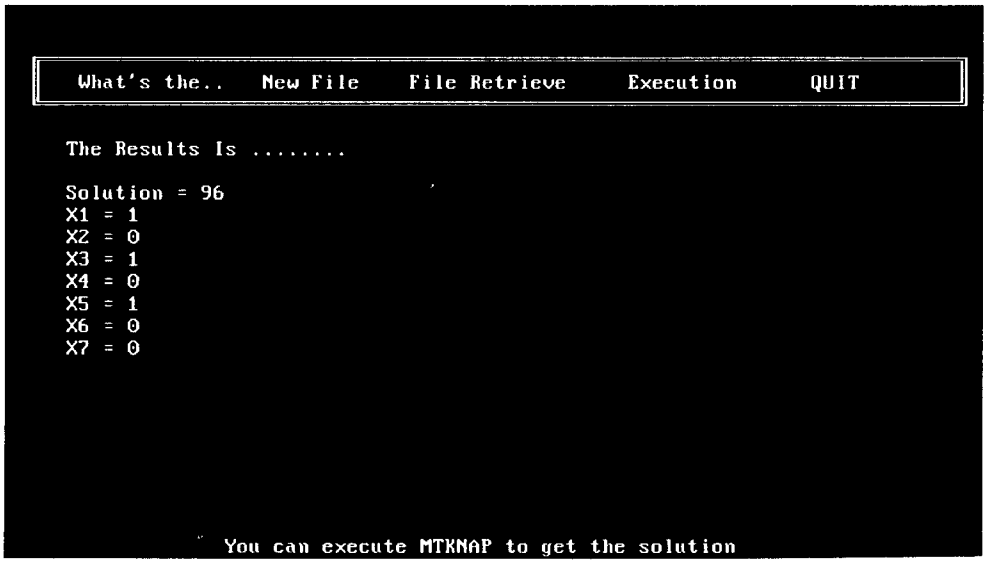
(c) File Retrieve

이미 화일로 저장되어 있는 문제를 불러오는 기능이다. 이 메뉴를 실행하면 디렉토리에 존재하는 문제의 화일 목록이 〈그림 4〉와 같이 풀다운 형태로 화면에 나타나고, 나열된 화일 목록 중에서 선택 메뉴를 이용하여 화일을 선택하면 〈그림 3〉과



〈그림 4〉 파일검색 화면

같은 자료편집 화면에 입력된 자료가 나타난다. 이와 같이 단순히 원하는 화일을 선택하는 기능 외에 즉시 자료의 확인과 자료의 편집이 가능하도록 하였다. 원하는 자료편집을 끝내면 상위메뉴로 되돌아오게 된다.



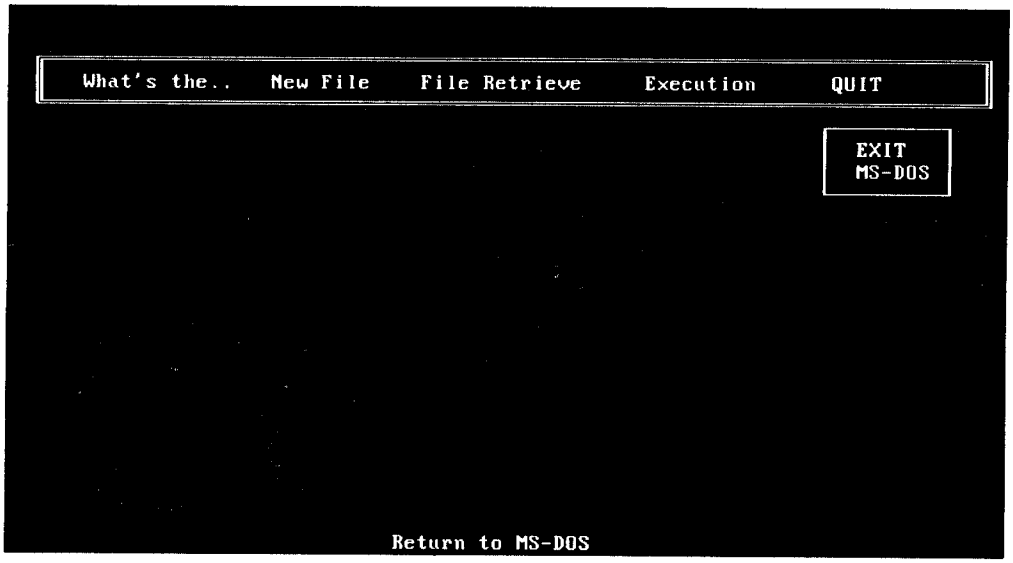
〈그림 5〉 실행결과 화면

(d) Execution

문제를 새로운 화일에서 편집하거나 기존의 문제를 불러들였을 경우 문제를 실행하여 답을 구하는 기능이다. 실행결과의 예는 〈그림 5〉와 같다. 화면에서 답을 확인하고 임의의 키를 누르면 상위메뉴로 되돌아간다.

(e) Quit

Quit는 〈그림 6〉과 같이 두 가지 하위 메뉴로 구성되어 있다. "EXIT" 부메뉴는 해법기의 사용을 완전히 마치고 DOS 상태로 빠져나가는 기능을 한다. "MS-DOS" 부메뉴는 해법기의 사용도중에 DOS 상으로 일시 빠져나가는 기능이다. 이 때 DOS 상에서 EXIT를 누르면 다시 해법기로 되돌아온다.



〈그림 6〉 Quit 화면

2. 개발된 해법기의 특징

의사결정지원시스템은 사용자가 사용자 인터페이스를 통해 필요한 모델과 자료를 이용하여 의사결정과정에 도움을 받도록 지원해 준다. 개발된 해법기는 자원 배분과 관련된 경영상의 문제들을 풀기 위한 의사결정지원시스템의 한 모듈로 PC용 관계형 DBMS인 CA-Clipper 5. 2를 이용하여 개발되었다.

본 해법기에서 문제를 구성하는 데이타는 〈표 1〉과 같은 구조의 데이타베이스에 저장된다. 이 데이타베이스는 PROFIT, SIZE, P_OVER_S, SACK 등 4개의 필드를 갖는다. 모든 필드는 수치형으로 그 길이는 10자이고 P_OVER_S는

〈표 1〉 KNAPSACK. DBF의 구조

필드명	자료형태	자료길이	소숫점 아래 자리수	비 고
PROFIT	N	10	0	가치
SIZE	N	10	0	무게
P_OVER_S	N	10	2	단위당 가치
SACK	N	10	0	해법결과(1 또는 0)

소수점 이하 2자리 실수를, 나머지는 정수를 입력하도록 하였다. 물론 필요시 데이타베이스의 구조를 적절히 조정할 수 있다.

본 해법기는 사용의 편의성을 향상시키고자 풀다운 메뉴 방식을 이용하였다. 특히 "File Retrieve"에서 이미 존재하는 문제의 목록이 나타나도록 하여 화일명을 기억하지 못한 경우나 화일명 입력의 번거로움을 없앴다. 또한 시스템의 사용도중 일시적으로 DOS로 빠져나갈 수 있도록 하였고, 사용 경험이 없는 이용자들을 위해 도움말 기능을 두었다.

그러나, 본 해법기에서는 CA-Clipper를 위해 상업용으로 지원되고 있는 여러 라이브러리들이 대부분 한글과 그래픽을 지원하고 있으므로 한글 지원을 위한 특정 라이브러리의 이용은 개발자들의 손에 맡기고 해법기의 신속한 개발을 위해 CA-Clipper의 표준 라이브러리만을 사용하였다.

5. 결 론

본 연구에서는 관계형 데이타베이스를 이용한 배낭문제의 해법기를 Martello-Toth의 알고리즘을 이용하여 구현하였다. 이 해법기는 PC용 DBMS인 CA-Clipper를 이용하여 개발되었으며, 사용자 도움말 기능과 새로운 자료의 입력 기능, 기존 화일을 화면상에서 메뉴 막대로 선택하는 기능, 일시적인 운영체제 명령어 사용 기능 등을 갖추고 있다. 개발된 해법기는 관계형 데이타베이스를 매개로 하여 다른 수리적 모형들과의 원활한 접속을 통해 자원 배분 등의 실질적인 경영문제들을 해결하기 위한 의사결정지원시스템의 개발에 활용될 수 있다.

이러한 수리계획 소프트웨어의 개발은 최근의 "소규모 한글 선형계획법 소프트웨어 K-LP의 개발[1]", "MS-DOS용 선형계획법 통합환경 소프트웨어의 개발[2]", "수리계획 소프트웨어 LinPRO의 설계 및 구현[3]" 등과 함께 개발된 해법기의 원시프로그램을 공개하여 DBMS를 이용한 수리계획법의 각종 알고리즘을 프로그램으로 개발하고자 하는 사람들이 사용할 수 있도록 하였다.

참 고 문 헌

[1] 김세헌, 김성륜, 장근녕, 여재현, "소규모 한글 선형계획법 소프트웨어 K-LP의 개발", 「경영과학」, 제11권 제3호(1994년 10월), pp. 27-34

[2] 설동렬, 박찬규, 서용원, 박순달, "MS-DOS용 선형계획법 통합환경 소프트웨어의 개발", 「경영과학」, 제12권 제1호(1995년 2월), PP. 126-138

[3] 양광민, "수리계획 소프트웨어 LinPro의 설계 및 구현", 「경영과학」, 제12권 제1호(1995년 2월), pp.139-156

[4] 이상문, 백종현, 어퍼레이션즈 리서치, 2판, 경문사, 1984

[5] Martello, S. and Toth, P., "An Upper Bound for the Zero-one Knapsack Problem and a Branch and Bound Algorithm", *European Journal of Operational Research*, Vol.1(1977), pp.169-175

[6] Martello, S. and Toth, P., "Algorithm for the Solution of the 0-1 Single Knapsack Problem", *Computing*, Vol.21(1978), pp. 81-86

[7] Salkin, H. M. and Kluyver, C. A., "The Knapsack Problem : A Survey", *Naval Research Logistics Quarterly*, Vol.22 (1975), pp.127-144

[8] Sharda,R., "Linear Programming Solver

Software for Personal Computers : 1995 Reports", *OR/MS Today*(October 1995), pp.49-57

[9] Suh, C.-K., Suh, E.-H., and Baek, K.-C., "The Prioritizing Telecommunication Technologies for Long-range Planning to the Year 2006", ***IEEE Transactions on Engineering Management***, Vol.41, No.3 (1994), pp.264-275.

[10] Suh, E.-H., Suh, C.-K., and Do, N.-C., "A Decision Support System for Investment Planning on Microcomputer", *Journal of Microcomputer Application*, Vol.15, No.4(1992), pp.297-311.

부 록

```
**************************************************
* 프로그램명 : KNAP. PRG
* 사용 DB : knapsack. DBF
* 내 용 : 0-1 Knapsack Problem
*   Martello and Toth, "An upper bound for
*   the zero-one knapsack problem and a
*   branch and bound algorithm", European
*   Journal of Operational Research, 1, 1977,
*   169-175
**************************************************
sizeLimit = 0
scroll(05,00,23,78)
@05,05 say "Input the Knapsack Size :" get
     sizeLimit;read

fName="&nfILE"
REPLACE ALL p_over_s WITH profit /size
INDEX ON p_over_s TO fName DESCENDING
STORE 0 TO karraySize, kl, ksolution, klold,
          ki, kp,;
          kUB, kLim, kB1, kB2, kB3,
          kB4, kt,;
          kR, kD, kTT, kn, kValue1,
          kIW, kIP
  kM=sizeLimit
karraySize=RECCOUNT()
kaP      :=ARRAY(karraySize)
kaW      :=ARRAY(karraySize)
kaMin    :=ARRAY(karraySize)
kaPbar   :=ARRAY(karraySize)
kaWbar   :=ARRAY(karraySize)
kaZbar   :=ARRAY(karraySize)
kaX      :=ARRAY(karraySize)
kFixedX  :=ARRAY(karraySize)

For j=1 to karraySize
  kaP[j] = profit
  kaW[j] = size
  skip
NEXT
USE
kstep = '1'
  krow = 7 ; kline = 1

DO WHILE . T.
   IF kline > 10
      kline = 1
      @ 7, 2 CLEAR TO 17, 78
 @ krow + kline, 10 SAY "STEP " + kstep
    ELSE
     kline += 1
   ENDIF
```

```
DO CASE
  CASE kstep = '1'
    FOR j = 1 TO karraySize
      IF kIW + kaW[j] <=kM
         kIW += kaW[j]
      ELSE
         kl = j - 1
         EXIT
      ENDIF
    NEXT

    FOR j=1 TO kl
      kIP += kaP[j]
    NEXT
    ksolution = 0
    IF kIW = kM
      ksolution = kIP
      FOR j=1 TO kl
      kFixedX[j] = 1
    NEXT
    IF kl < karraySize
     FOR j = kl + 1 TO karraySize
        kFixedX[j] = 0
      NEXT
    ENDIF
    EXIT    //kfixedX의 값이 해이다.
 ENDIF
aTemp := ARRAY(karraySize - 1)
FOR j = 2 TO karraySize
     @ 6, 10 say "j=" + substr(str(j),-3)
     ACOPY(kaW, aTemp, j, karraySize-j+1)
     ASORT(aTemp, , , {|A, B| a>b})
     kaMin[j-1] := ATAIL(aTemp)
     ASIZE(aTemp, karraySize - j)
NEXT
kaMin[karraySize] = kM+1
STORE 0 TO kp, ksolution
ki = 1
klold = karraySize
  kB1=kIP+(kM-kIW)*kaP[kl+2]/kaW
       [kl+2]
  kB2=kIP+kaP[kl+1]-(kaW[kl+1]-kM+;
       kIW)*kaP[kl]/kaW[kl]
  kUB2 = INT(MAX(kB1, kB2))
kLim = kUB2
FOR j = 1 TO karraySize
     kaX[j] = 0
NEXT
kstep = '4'
LOOP
WAIT "Error In Step 1"

CASE kstep ='2'
  IF kaW[ki] <= kM
    kstep = '3'
    LOOP
  ENDIF
kValue1=kp+kM*kaP[ki+1]/kaW[ki+1]
  IF ksolution >= kValue1
    //=kp+kM*kaP[ki+1]/kaW[ki+1]
    kstep = '5'
    LOOP
  ELSE
    ki += 1
    kstep = '2'
    LOOP
  ENDIF
  WAIT "Error in step 2"
```

```
CASE kstep = '3'
  kt = kaZbar[ki]
  IF kaW[kt] > kM
    kl = kaZbar[ki] - 1
  ELSE
    kIP = kaPbar[ki]
    kIW = kaWbar[ki]
    FOR j= kt TO karraySize
      IF kIW + kaW[j] <= kM
        kIP += kaP[j]
        kIW += kaW[j]
        kl := j
      ELSE
        EXIT    // 해를 구했다.
      ENDIF
    NEXT
  ENDIF
  IF kIW < kM . AND. kl < karraySize
    IF kl+2 <= karraySize
kB3 = (kM - kIW)*kaP[kl+2]/kaW[kl+2]
    ELSE
      kB3 = 0
    ENDIF
kB4 = kaP[kl+1]-(kaW[kl+1]-(kM -;
        kIW))*kaP[kl]/kaW[kl]
    kUB = INT(MAX(kB3, kB4))
    IF ksolution > kp + kIP + kUB
      kstep = '6'
      LOOP
    ELSE
      kstep = '4'
      LOOP
    ENDIF
  ENDIF
  IF kIW =kM . OR. kl = karraySize
    IF ksolution >= kp + kIP
      kstep = '6'
      LOOP
    ELSE
      ksolution := kp + kIP
      FOR j=1 TO ki - 1
        kFixedX[j] = kaX[j]
      NEXT
      FOR j = ki TO kl
        kFixedX[j] = 1
      NEXT
      IF kl+1 <= karraySize
        FOR j = kl + 1 TO karraySize
          kFixedX[j] = 0
        NEXT
      ENDIF
      IF ksolution = kLim
        EXIT    // 해를 구했다.
      ELSE
        kstep = '6'
        LOOP
      ENDIF
    ENDIF
  ENDIF
  WAIT "Error in step 3"

CASE kstep = '4'
  kM -= kIW
  kP += kIP
  FOR j = ki TO kl
    kaX[j] = 1
  NEXT
  kaPbar[ki] = kIP
  kaWbar[ki] = kIW
  kaZbar[ki] = kl + 1
```

```
FOR j = ki + 1 TO kl
  kaPbar[j] = kaPbar[j-1] - kaP[j-1]
  kaWbar[j] = kaWbar[j-1] - kaW[j-1]
  kaZbar[j] = kl + 1
NEXT
IF kl + 1 <= klold
  FOR j = kl + 1 TO klold
    kaPbar[j] = 0
    kaWbar[j] = 0
    kaZbar[j] = j
  NEXT
ENDIF
klold = kl
IF kl < karraySize - 2
  ki = kl + 2
  IF kM < kaMin[ki-1]
     kstep = '5'
     LOOP
  ELSE
     kstep = '2'
     LOOP
  ENDIF
ENDIF
IF kl = karraySize - 2
  IF kM >= kaW[karraySize]
     kM -= kaW[karraySize]
     kp += kaP[karraySize ]
     kaX[karraySize] = 1
  ENDIF
  ki = karraySize - 1
  kstep = '5'
  LOOP
ENDIF
IF kl = karraySize - 1
   ki = karraySize
     kstep = '5'
     LOOP
  ENDIF
  WAIT "Error in step 4"

CASE kstep = '5'
  IF ksolution < kp
    ksolution = kp
    FOR j=1 TO karraySize
      kFIxedX[j] := kaX[j]
    NEXT
    IF ksolution = kLim
      EXIT    // 해를 구했다.
    ENDIF
  ENDIF
  IF kaX[karraySize] = 1
    kM+= kaW[karraySize]
    kp -= kaP[karraySize]
    kaX[karraySize] = 0
  ENDIF
  kstep = '6'
  LOOP

CASE kstep = '6'
  kk = 0
  FOR j=1 TO ki - 1
    IF kaX[j] = 1 //
       kk = j
    ENDIF
  NEXT
  IF kk = 0
    EXIT    // 해를 찾았다.
  ENDIF
  kR = kM
  kM += kaW[kk]
```

```
  kP -= kaP[kk]
  kaX[kk] = 0
  IF kR >= kaMin[kk]
    ki = kk + 1
    kstep = '2'
    LOOP
  ELSE
    ki = kk
    kn = kk + 1
    kstep = '7'
    LOOP
  ENDIF

CASE kstep = '7'
  IF kn > karraySize
    kstep = '6'
    LOOP
  ENDIF
  kValue1 = kP + kM*kaP[kn]/kaW[kn]
  IF ksolution >= kValue1
    kstep = '6'
    LOOP
  ENDIF
  kD = kaW[kn] - kaW[kk]
  IF kD = 0
    kn += 1
    kstep = '7'
    LOOP
  ENDIF
  IF kD > 0
    IF kD > kR
      kn += 1
      kstep = '7'
      LOOP
    ENDIF
    IF ksolution >= kp + kaP[kn]
      kn += 1
      kstep = '7'
      LOOP
    ENDIF
    ksolution = kp + kaP[kn]
    FOR j=1 TO kk
      kFixedX[j] = kaX[j]
    NEXT
    FOR j = kk + 1 to karraySize
      kFixedX[j] = 0
    NEXT
    kFixedX[kn] = 1
    IF ksolution = kLim
      EXIT    // 해를 찾았다.
    ELSE
      kR -= kD
      kk = kn
      kn += 1
      kstep = '7'
      LOOP
    ENDIF
  ENDIF
  IF kD < 0
    kTT = kR - kD
    IF kTT < kaMin[kn]
    kn += 1
    kstep = '7'
    LOOP
  ENDIF
  kValue1=kp+kaP[kn]+kTT*kaP[kn]
            /kaW[kn]
  IF ksolution >= kValue1
    kstep = '6'
    LOOP
```

```
      ENDIF
      kM -= kaW[kn]
      kp += kaP[kn]
      kaX[kn] = 1
      ki = kn + 1
      kaWbar[kn] = kaW[kn]
      kaPbar[kn] = kaP[kn]
      kaZbar[kn] = kn + 1
      IF kn + 1 <= klold
        FOR j = kn+1 to klold
          KaWbar[j] = 0
          kaPbar[j] = 0
          kaZbar[j] = 0
        NEXT
      ENDIF
      klold = kn
      kstep = '2'
      LOOP
      ENDIF
  ENDCASE
ENDDO

? "End of step"
wait " "
scroll(05, 01, 23, 78)
@06, 05 SAY "The Results Is....."
@08, 05 SAY "Solution = " + alltrim(str
(ksolution))
? "      "
INKEY(0)

srow=9 ; scol = 5
USE knapsack INDEX fname
FOR j = 1 TO karraySize
   REPLACE sack with kFixedX[j]
@srow, scol SAY "X"+LTRIM(STR(J))+;
   " = " + STR(kFixedX[j],1,0)
    SKIP
    srow +=1
   IF srow > 20
     WAIT " "
     srow = 9
   ENDIF
NEXT
USE
INKEY()
CLOSE ALL
RETURN

**************************************************
* 풀다운식 주메뉴
* 화일명 : MTKMENU. PRG
**************************************************
# INCLUDE "INKEY. CH"
# INCLUDE "BOX. CH"

SET(_SET_WRAP, .T.)
SET(_SET_MESSAGE, 24)
SET(_SET_MCENTER, .T.)

DO WHILE .T.
   SCROLL()
   SETCOLOR("GR+/B,GR+/bg+, r, ,,")
   SCROLL(0, 0, 4, 78)
   DISPBOX(02, 02, 04, 78, B-DOUBLE)
   _ATPROMPT(03, 05, "What's the.", "You
        can"+;" get the information about
        MTKNAP")
   _ATPROMPT(03, 20, "New File", "If
        you want"+;" to create new file")
```

```
   _ATPROMPT(03, 32, "File Retrieve",
        "When you want to use already
        existing file")
   _ATPROMPT(03, 50, "Execution", "You
        can execute MTKNAP to get the
        solution")
   _ATPROMPT(03, 65, "Quit", "Quit OR
        EXIT TO MS-DOS")
  MENU TO nMenuNo1

PUBLIC abc := SPACE(8)
abc = DBF()+". DBF"

  DO CASE
     CASE nMenuNo1 = 1
          DO MTKHELP
          WAIT " "
     CASE nMenuNo1 = 2
      DO NewFile; @01, 67 SAY abc COLOR
                "R/W"
          WAIT " "
     CASE nMenuNo1 = 3
      DO OldFile; @01, 67 SAY abc COLOR
                "R/W"
          WAIT " "
          CASE nMenuNo1 = 4
            DO Knap
            WAIT " "
          CASE nMenuNo1 = 5
             SCROLL(05, 00, 24, 79)
        DISPBOX(05, 65, 08, 75, B-SINGLE)
           _ATPROMPT(06, 67, "EXIT",
"Return to MS-DOS")
           _ATPROMPT(07, 67, "MS-DOS",
"MS-DOS commands")
     SETKEY(K_RIGHT, {||Rightkey()})
     SETKEY(K_LEFT, {||LeftKey()})
     MENU TO nMenuNo2
     SETKEY(K_LEFT, NIL)
     SETKEY(K_RIGHT, NIL)
  DO CASE
     CASE nMenuNo2=0;LOOP
     CASE nMEnuNo2=1;QQOUT
                    (CHR(7))
       SCROLL(); RETURN
     CASE nMenuNo2 = 2 ; SCROLL()
       QOUT("EXIT to Quit")
         RUN c:\command. com
  ENDCASE
  _KEYBOARD(CHR(K-ENTER))

  ENDCASE
ENDDO

*-----------------------------------------------------------
PROCEDURE RightKey

_KEYBOARD(CHR(K_ESC)+CHR(K_
RIGHT)+CHR(K_ENTER))
RETURN

*-----------------------------------------------------------
PROCEDURE RightKey

_KEYBOARD(CHR(K_ESC)+CHR(K_
RIGHT)+CHR(K_ENTER))
RETURN
```

```
*************************************************
* KNAPSACK 문제 해결을 도움말
* 제공 프로그램
* 프로그램명 : MTKHELP. PRG
*************************************************
SET SCOREBOARD OFF
SCROLL()
DISPBOX(05, 02, 23, 78)
@07, 04 SAY "MTKNAP은 0-1 배낭문제를 해
결하기 위한 시스템입니다."
@08, 04 SAY SPAC(58)
@09, 04 SAY "여기서 사용하는 문제의 형식은
다음과 같습니다."
@10, 04 SAY SPACE(58)
@11, 04 SAY "목적식"
@12,04 SAY "p1x1+p2x2+・・・+pnxn"+;
            SPACE(28)
@13, 04 SAY SPACE(58)
@14, 04 SAY "제약조건"+SPACE(48)
@15, 04 SAY "w1x1+w2x2+・・・+wnxn<=
            W"+; SPACE(24)
@16, 04 SAY SPACE(58)
@17, 04 SAY SPAC(10)+ "xj=1 or 0"+
            SPACE(35)
@18, 04 SAY "여기서 pj는 대상 j의 가치 또는
이익을, wj는 대상 j의"
@19, 04 SAY SPACE(58)
@20, 04 SAY "무게를 나타냅니다."
@21, 04 SAY SPACE(58)
RETURN
```

```
*************************************************
* 자료입력화면
* 프로그램명 : INEDIT. PRG
*************************************************
#include "inkey. ch"
#INCLUDE "setcurs. ch"
#INCLUDE "dbedit. ch"
#TRANSLATE    endMemo() =>
_KEYBOARD(CHR(K_CTRL_W))

SCROLL()
SET(_SET_SCOREBOARD, . F.)
PRIVATE aArray1 :={"PROFIT", "SIZE"}
PRIVATE aArray6 := {"-","-"}
DEVPOS(00, 31):DEVOUT("INPUT & EDIT
&FNAME", "GR+ /R")

DBEDIT(02, 15, 22, 65, aArray1, "User",,,,,
aArray6)

*-------------------------------------------------------------
FUNCTION User(nMode, nFieldNo)
  LOCAL cCurField := aArray1[nFieldNo]

DO CASE
  CASE LASTKEY() = K_ENTER
  SETCURSOR(1)
  @ROW(), COL() GET &cCurField
  SCROLL(24,00)
  DEVPOS(24,33) : DEVOUT("INPUT DATA")
  READ
  SETCURSOR(SC-NONE)
  _KEYBOARD(CHR(K_RIGHT))

  CASE LASTKEY()=K_F5 .AND. cCurField
="PROFIT"
```

```
    IF nMode = DE_EXCEPT
        DBAPPEND()
      ENDIF

  CASE LASTKEY() = K_DEL
    DEVPOS(ROW(), COL())
    DEVOUT(&cCurField, "N/W")

     cAns = SPACE(1)
      SCROLL(24,00)
  @24, 28AY "Are You Sure ? (Y/n)"
        GET cAns
        PICTURE "Y" COLOR "W*/N"
        READ
    IF cAns = "Y"
      DBDELETE()
      _DBPACK()
      RETURN DE-REFRESH
    ENDIF
  CASE LASTKEY() = K-ESC
    RETURN DE-ABORT
ENDCASE
    DEVPOS(24,11)
DEVOUT("Input:〈Enter〉 Append:〈F5〉 De-
lete:〈Del〉 Quit:〈Esc〉")
RETURN DE_CONT

************************************************
* 새로운 화일명으로 자료를 입력할 때 사용
* 프로그램명 : NEWFILE. PRG
************************************************
PUBLIC aaa
STORE SPACE(12) TO aaa
DISPBOX(06, 20, 08, 48)
DO WHILE .T.
nFile = SPACE(8)
@07, 21 SAY "New File Name:" GET nFile
PICT "XXXXXXXXX"
READ

IF FILE("&nFile"+".DBF")
  @23, 20 SAY "That File Nmae Already
Exist. Input Other Name"
  Wait " "
  LOOP
ELSE
  EXIT
ENDIF
ENDDO

USE KNAPSACK
COPY STRUCTURE TO &nFile
USE &nFile
DISPBOX(06,20,08,50)
nVarNum = SPACE(4)
@07,21 SAY "Number of Variables :" GET
nVarNum PICT "####" ;READ
FOR i=1 TO VAL(nVarNum)
  APPEND BLANK
NEXT

aaa = DBF()+".DBF"
@01,67 SAY aaa COLOR "R/W"
cScr1 := SAVESCREEN(00,00,24, 79)
DO INEDIT
RESTSCREEN(00,00, 24, 79, cScr1)
@01, 67 SAY aaa COLOR "R/W"
RETURN
```

```
**************************************************
* 기존의 DBF 화일을 불러서 수정하거나
* 곧바로 EXECUTION 하는 프로그램
* 프로그램명 : OLDFILE. PRG
**************************************************
LOCAL aFiles[ADIR("*. DBF")]
ADIR("*. DBF", aFiles)
STORE SPACE(12) TO ccc
nMenuNo3  :=  ACHOICE(06,  35,  20,  48,
aFiles)

fname := afiles[nMenuNo3]
fname =alltrim(fname)

USE &fname
PUBLIC ccc
fold= DBF()+".DBF"
@01, 67 SAY fold COLOR "R/W"
cScr2 = SAVESCREEN(00,00, 24, 79)

DO INEDIT

RESTSCREEN(00,00, 24, 79, cScr2)
@01, 67 SAY fold COLOR "R/W"
RETURN
```