

# Generalisations of non-linear knapsack problems

W. O. Riha<sup>a</sup>, J. Walker<sup>b</sup>

*<sup>a</sup>School of Computer Studies, University of Leeds, Leeds, LS2 9JT, UK*

*<sup>b</sup>Nanyang Business School, Nanyang Technological University, Singapore 639798*

## **Abstract**

This paper presents an efficient algorithm for solving the Lagrangean dual of non-linear knapsack problems with additional nested constraints. The dual solution provides a feasible primal solution (if it exists) and associated lower and upper bounds on the optimal objective function value of the primal problem. Computational experience is cited indicating computation time, number of dual iterations, and "tightness" of the bounds.

*Keywords:* Non-linear Knapsack; Distribution Of Effort; Nested Constraints.

## 1. Introduction

### 1.1 Problem definition

Consider the problem PRIMAL:

$$\begin{aligned} \text{maximise } f(\mathbf{x}) &= \sum_{i=1}^n f_i(x_i) \\ \text{subject to } g(\mathbf{x}) &= \sum_{i=1}^n g_i(x_i) \leq b_0, \end{aligned} \quad (1)$$

$$\sum_{i=1}^j x_i \leq b_j, \quad j = 1, \dots, n, \quad (2)$$

$$0 \leq x_i \leq b_i \text{ and integer}, \quad i = 1, \dots, n, \quad (3)$$

where  $f_i$  ( $g_i$ ) is a concave (convex) real-valued function defined on  $\{0, \dots, b_i\}$ ,  $b_i$  a non-negative integer,  $i = 1, \dots, n$ ,  $0 \leq b_1 \leq \dots \leq b_n$  and  $b_0$  real.

Variations on problem PRIMAL obtained by omitting either constraints (1) or (2) have received some attention in the literature.

### 1.2 Omitting constraints (2)

PRIMAL reduces to a non-linear knapsack problem. The reader is referred to Dyer, Riha and Walker[2], and the references therein, for a summary of applications. Reference [2] also provides a description and computational experience of a hybrid dynamic programming/branch-and-bound algorithm for non-linear knapsack problems. The algorithm was shown to be highly effective even with large problems involving arbitrary  $f_i$  and  $g_i$  functions.

### 1.3 Omitting constraint (1)

(a)  $0 < b_1 = \dots = b_n$  PRIMAL reduces to the well known distribution of effort problem. For  $f_i$  concave Frederickson and Johnson[6] provide an algorithm which runs in  $O(n \log(b_n/n))$  time.

(b)  $0 \leq b_1 \leq \dots \leq b_n$  PRIMAL reduces to a form in which the nested constraints represent the cumulative bounding of the variables. Such a form can be used to model production planning problems, Dyer and Walker[3], Tamir[10] and the management of a bank's bond portfolio, Mavrides[9]. Dyer and Walker[4] provide an algorithm which runs in  $O(n \log(n) \log^2(b_n/n))$  time.

### 1.4 Incorporating constraints (1) and (2)

Incorporating constraints (1) and (2) provides greater generality and applications include nested knapsack problems, examples of which are provided by Tamir[11] and Armstrong, Sinha and Zoltner[1]; distribution of effort and production planning problems where an additional resource constraint has to be considered. PRIMAL may itself be regarded as a relaxation of some larger combinatorial optimisation problem in which constraint (1) represents a surrogate constraint, see for example Dyer[5]. In this paper an algorithm is presented for solving the Lagrangean dual of problem PRIMAL. The dual solution provides a feasible primal solution (if it exists) and associated lower and upper bounds on the optimal objective function value of the primal problem. The algorithm embeds one Lagrangean relaxation within another and has a "simple and elegant" structure which is easy to code. Computational experience is cited indicating computation time, number of dual iterations, and "tightness" of the bounds. Obtaining tight bounds in reasonable amounts of computing time is of prime importance in the development of efficient branch-and-bound based algorithms for combinatorial optimisation problems and also in problems where a "good" feasible solution is

sufficient. The computational experience reported in this paper clearly demonstrates such a pattern.

## 2. Lagrangean Relaxation

### 2.1 Dual algorithm

To avoid a lengthy review of Lagrangean relaxation the reader is referred to Fisher[7]. For a given multiplier  $y \geq 0$  one possible relaxation of PRIMAL may be written as RELAX(y):

$$h(y) = \text{maximum}\{f(\mathbf{x}) - y[g(\mathbf{x}) - b_0] : \text{subject to (2) and (3)}\}.$$

$h(y)$  provides an upper bound on the optimal objective function value of PRIMAL (as is conventional  $h(y) = -\infty$  if PRIMAL is infeasible). The minimum such upper bound is determined by solving the Lagrangean dual DUAL:

$$h(y^0) = \text{minimum}\{h(y) : \text{subject to } y \geq 0\}.$$

The following procedure for solving DUAL is a simple modification to that proposed by Greenberg[8] ("tangential approximation") for one-dimensional generalised Lagrange multiplier problems. A proof of convergence to the optimal dual solution for such procedures is given by Shapiro[7]. On exit from the procedure the following elements are available:  $y^0$  the optimal dual multiplier,  $\mathbf{x}^0$  an associated optimal solution of RELAX( $y^0$ ),  $f^0$  the value  $f(\mathbf{x}^0)$ ,  $g^0$  the value  $g(\mathbf{x}^0)$ , and  $h^0$  the value  $h(y^0)$ . If PRIMAL is feasible then  $\mathbf{x}^0$  represents a feasible solution with  $g^0 \leq b_0$  and  $f^0, h^0$  provide lower and upper bounds on the optimal objective function value of PRIMAL.

**procedure dual**( n, f, g, b,  $y^0$ ,  $x^0$ ,  $f^0$ ,  $g^0$ ,  $h^0$  )

**begin**

$y^0 \leftarrow \text{LARGE}$

$x^{\text{prev}} \leftarrow 0$

**call relax**(  $n, f, g, b, y^0, x^{\text{prev}}, 0, x^0, f^0, g^0, h^0$  )

**if**  $g^0 \geq b_0$  **then** "PRIMAL infeasible or  $x^0$  trivially optimal."

**stop**

**endif**

$y \leftarrow 0$

**call relax**(  $n, f, g, b, y, x^{\text{prev}}, 0, x^y, f^y, g^y, h^y$  )

**if**  $g^y \leq b_0$  **then** " $x^y$  trivially optimal."

$\{ y^0, x^0, f^0, g^0, h^0 \} \leftarrow \{ y, x^y, f^y, g^y, h^y \}$

**stop**

**endif**

$t \leftarrow 0$

**while**  $h^y > f^0 - y[g^0 - b_0]$  **do**

**if**  $g^y > b_0$  **then**

$\{ f^1, g^1 \} \leftarrow \{ f^y, g^y \}$

**else**

$\{ x^0, f^0, g^0 \} \leftarrow \{ x^y, f^y, g^y \}$

**endif**

$y \leftarrow (f^1 - f^0) / (g^1 - g^0)$

$x^{\text{prev}} \leftarrow x^0$

$t \leftarrow t + 1$

**call relax**(  $n, f, g, b, y, x^{\text{prev}}, t, x^y, f^y, g^y, h^y$  )

**endwhile**

$\{ y^0, h^0 \} \leftarrow \{ y, h^Y \}$  "Optimal DUAL solution."

**if**  $g^Y = b_0$  **then** "Optimal PRIMAL solution."

$\{ y^0, x^0, f^0, g^0, h^0 \} \leftarrow \{ y, x^Y, f^Y, g^Y, h^Y \}$

**endif**

**end dual**

Central to the procedure **dual** is the procedure **relax** for solving RELAX(y) for a given y. The procedure **relax** for solving RELAX(y) is based on a further Lagrangean relaxation formed by multiplying each constraint (2) by a non-negative multiplier and appending to the objective function. The correctness of the procedure is derived in section 2.4. On exit the following elements are available:  $\mathbf{x}^Y$  an associated optimal solution of RELAX(y),  $f^Y$  the value  $f(\mathbf{x}^Y)$ ,  $g^Y$  the value  $g(\mathbf{x}^Y)$ , and  $h^Y$  the value  $h(y)$ .

## 2.2 Initialisation heuristic

In order to guarantee that **relax** will determine the optimal solution to RELAX(y),  $\mathbf{x}^Y$ , it is necessary to initialise **relax** with a feasible vector  $\mathbf{x}^{\text{prev}}$  such that  $\mathbf{x}^{\text{prev}} \leq \mathbf{x}^Y$ . Clearly  $\mathbf{x}^{\text{prev}} = \{0\}$  would suffice. However, with the application of **relax** within **dual** it is found that as the dual iterations increase the feasible solution resulting from the t-1 th iteration,  $\mathbf{x}^0(\mathbf{t-1})$ , is "closer" to the  $\mathbf{x}^Y(\mathbf{t})$  than the null vector and  $\mathbf{x}^Y(\mathbf{t}) - \mathbf{x}^0(\mathbf{t-1}) \rightarrow \{0\}$ . Therefore, initialising **relax** at the t th dual iteration to  $\mathbf{x}^0(\mathbf{t-1})$  should result in quicker termination. In computational testing the initialisation  $\lfloor \mathbf{x}^0(\mathbf{t-1}) \cdot t / (t+1) \rfloor$  always provided the correct  $\mathbf{x}^Y(\mathbf{t})$ . However, the more conservative initialisation indicated in **relax** was used. Computational experience with the initialisation heuristic is described below and in all cases tested, the initialisation heuristic

determined the correct solution to RELAX(y). Clearly in applications with characteristics different from the generated problems described below some preliminary testing may need to be performed.

### 2.3 Relaxation algorithm

The following procedure for solving RELAX(y) is essentially a “greedy” algorithm utilising marginal analysis. At each iteration the variable that has the largest increment in objective function value and also retains feasibility in the nested constraint set (2) is increased by 1. In order to efficiently maintain a record of variables which can be increased by 1 and yet retain feasibility in (2) the parameter  $i^s$  is used to denote the largest index in (2) for which the “slack” is zero. That is, only variables  $x_i$  with  $i > i^s$  can be increased by 1 and retain feasibility.

**procedure relax**( n, f, g, b, y,  $x^{\text{prev}}$ , t,  $x^y$ ,  $f^y$ ,  $g^y$ ,  $h^y$  )

"In all heap operations the keys are given by  $\Delta p_i(x_i^y + 1) = \Delta f_i(x_i^y + 1) - y \Delta g_i(x_i^y + 1)$  (where  $\Delta$  denotes the backward difference operator) with associated indices  $1 \leq i \leq n$ . In the case of a tie in comparing keys the key with the least index is placed on 'top'."

**begin**

$x_i^y \leftarrow \text{maximum}\{0, \lfloor x_i^{\text{prev}} \cdot t / (t+1) \rfloor - 10\}$ ,  $i = 1, \dots, n$  "Initialisation heuristic."

$s_i \leftarrow b_i - \sum_{j=1}^i x_j^y$ ,  $i = 1, \dots, n$

**call formmaxheap**("Form the keys  $\Delta p_i(x_i^y + 1)$ ,  $i = 1, \dots, n$ , into a maxheap.")

$i^s \leftarrow 0$  "Largest index  $i$  for which the slack  $s_i = 0$ ."

**while**  $i^s \neq n$  **do**

L1           **call topofheap**("Let  $k$  denote the index of the key at the top of the heap.")

```

if  $\Delta p_k(x_k^y+1) \leq 0$  then
L2       $i^s \leftarrow n$ 
L3      else if  $k > i^s$  then " $x_k^y+1$  is feasible."
           $x_k^y \leftarrow x_k^y+1$ 
          for  $m \leftarrow k$  to  $n$  do
               $s_m \leftarrow s_m-1$ 
              if  $s_m = 0$  and  $m > i^s$  then
                   $i^s \leftarrow m$ 
              endif
          endfor
          call updateheap("Update key of index  $k$  to  $\Delta p_k(x_k^y+1)$  and adjust maxheap
                           accordingly.")
          else if  $k = n$  then
               $i^s \leftarrow n$ 
          else
L4      call modifyheap("Modify maxheap by deleting all keys with index  $\leq k$ .")
          endif
endwhile
          {  $f^y, g^y$  }  $\leftarrow$  {  $f(x^y), g(x^y)$  }
           $h^y \leftarrow f^y - y[g^y - b_0]$ 
end relax

```

#### 2.4 Correctness of relax

Sufficiency conditions are developed under which an optimal solution to a Lagrangean relaxation of RELAX(y) also provides an optimal solution to RELAX(y). Note should be made of the remarks made above regarding the initialisation heuristic. That is, it is assumed that an initial  $x^{\text{prev}} \leq x^y$  is available. Clearly,  $x^{\text{prev}} = \{0\}$  would suffice. For notational convenience the superscript y is omitted.

For a given  $y \geq 0$  let  $p_j(x_j) = f_j(x_j) - y g_j(x_j)$ ,  $x_j^{\text{prev}} \leq x_j \leq b_j$ ,  $j = 1, \dots, n$ . Since  $f_j$  ( $g_j$ ) is concave (convex) and  $y \geq 0$   $p_j$  is concave. For given multipliers  $u_j \geq 0$ ,  $j = 1, \dots, n$ , consider the Lagrangean relaxation of RELAX(y):

$$\text{maximise } \left\{ \sum_{j=1}^n p_j(x_j) - \sum_{j=1}^n u_j \left[ \sum_{i=1}^j x_i - b_j \right] : \text{subject to } x_j^{\text{prev}} \leq x_j \leq b_j \text{ and integer, } j = 1, \dots, n \right\}.$$

The maximisation decomposes into n separate problems (P<sub>j</sub>):

$$\text{maximise } \left\{ p_j(x_j) - \sum_{i=j}^n u_i x_j : \text{subject to } x_j^{\text{prev}} \leq x_j \leq b_j \text{ and integer} \right\}.$$

It is easily demonstrated, see for example, Fisher[7], that if the  $u_j$ ,  $x_j$ ,  $j = 1, \dots, n$ , are such that:

$$u_j \geq 0, \quad (4)$$

$$x_j \text{ solves } (P_j), \quad (5)$$

$$\sum_{i=1}^j x_i \leq b_j, \text{ and} \quad (6)$$

$$u_j \left[ \sum_{i=1}^j x_i - b_j \right] = 0, \quad (7)$$

then  $x_j$  is optimal in RELAX(y). The sufficiency conditions can be rewritten in a more convenient form. Let  $v_n = u_n$  and  $v_j = u_j + v_{j+1}$ ,  $j = 1, \dots, n-1$ .

The requirement (4) that the multipliers  $u_j$ ,  $j = 1, \dots, n$ , be non-negative is equivalent to the requirement that:

$$\infty > v_1 \geq \dots \geq v_n \geq 0 \quad (8)$$

With the substitution of  $(v_j - v_{j+1})$  for  $u_j$ ,  $j = 1, \dots, n-1$ , and  $v_n$  for  $u_n$  the problem (Pj) may be rewritten:

$$\text{maximise } \{Q_j(v_j, x_j) : \text{subject to } x_j^{\text{prev}} \leq x_j \leq b_j \text{ and integer}\}$$

where  $Q_j(v_j, x_j) = p_j(x_j) - v_j x_j$ . Since  $p_j$  is concave so is  $Q_j$  and the condition that  $x_j$  maximises  $Q_j$  is that  $\Delta Q_j(v_j, x_j) \geq 0$  and  $\Delta Q_j(v_j, x_{j+1}) \leq 0$  where  $\Delta$  is the backward difference operator with:

$$\Delta Q_j(v_j, x_j) = \begin{cases} Q_j(v_j, x_j) - Q_j(v_j, x_{j-1}) & x_j > x_j^{\text{prev}}, \\ +\infty & x_j = x_j^{\text{prev}}. \end{cases}$$

Defining  $\Delta p_j(x_j)$  in a similar fashion:

$$\Delta Q_j(v_j, x_j) = \Delta p_j(x_j) - v_j \text{ and } \Delta Q_j(v_j, x_{j+1}) = \Delta p_j(x_{j+1}) - v_j.$$

The requirement (5) that  $x_j$  solve (P<sub>j</sub>) is then equivalent to the requirement that:

$$\Delta p_j(x_j) \geq v_j \geq \Delta p_j(x_{j+1}) \quad , \quad j = 1, \dots, n. \quad (9)$$

The requirement (6) is simply that  $\mathbf{x}$  be feasible in RELAX(y).

With the substitution of  $(v_j - v_{j+1})$  for  $u_j$ ,  $j = 1, \dots, n-1$ , and  $v_n$  for  $u_n$  the requirement (7) for complementary slackness is then equivalent to the requirement that:

$$\left. \begin{aligned} (v_j - v_{j+1}) \left[ \sum_{i=1}^j x_i - b_j \right] &= 0 & \} \\ & & \} \\ v_n \left[ \sum_{i=1}^n x_i - b_n \right] &= 0 & \} \end{aligned} \right\} \quad (10)$$

To summarise, a solution  $\mathbf{x}$  is optimal in RELAX(y) if it is feasible and there exists a vector  $\mathbf{v}$  such that conditions (8), (9) and (10) are satisfied.

*Proposition 1* The procedure **relax** terminates with a solution  $\mathbf{x}$  that is feasible in RELAX(y) and an implicit vector  $\mathbf{v}$  satisfying conditions (8), (9) and (10).

*Proof* To prove the proposition note the following:

(a) Only feasible solutions are constructed and the procedure iterates  $i^S$  and  $k$  the next variable for possible increase;  $i^S$  is updated when some constraint  $m$ ,  $k \leq m \leq n$ , is binding and termination occurs when either

- (i)  $m = n$ ; or
- (ii) the values  $p_j(x_j)$  are non-increasing for further increases in  $x_j$ .

(b) Immediately before each execution of label L4 let  $i = i^S + 1$  and  $v_j = \Delta p_k(x_{k+1})$ ,  $j = i, \dots, k$ .

Thus

$$v_1 \geq v_2 \geq \dots \geq v_i = v_{i+1} = \dots = v_k \geq v_{k+1} \geq \dots \geq v_n \geq -\infty.$$

This follows from the fact that at label L1 the maximum key is non-increasing and  $p_j$  is concave. If the procedure subsequently terminates with constraint  $n$  binding then  $v_n > 0$ . If the procedure terminates with the values  $p_j(x)$  non-increasing for  $x \geq x_j$ ,  $j = i, \dots, n$ , then immediately before the execution of label L2 let  $i = i^S + 1$  and  $v_j = 0$ ,  $j = i, \dots, n$ . Thus

$$v_1 \geq v_2 \geq \dots \geq v_i = v_{i+1} = \dots = v_n = 0.$$

Thus condition (8) is satisfied at termination.

(c)  $\Delta p_j(x_j) \geq v_j \geq \Delta p_j(x_{j+1})$ ,  $j = i^S + 1, \dots, k$ .

The left-hand inequality follows from the fact that if

- (i)  $x_j = x_j^{\text{prev}}$  then by definition  $\Delta p_j(x_j) = +\infty$ , or
- (ii)  $x_j > x_j^{\text{prev}}$  implies that  $x_j$  has been increased to its current value at some earlier iteration according to the criterion of label L1.

The right-hand inequality follows directly from the execution of label L1 in determining  $k$ .

Since  $i^s$  is set initially to 0 and at termination  $k$  is set (implicitly) to  $n$  conditions (8) are satisfied for all  $j = 1, \dots, n$

(d) Immediately before each execution of label L4 let  $i = i^{s+1}$  and  $v_j = \Delta p_k(x_{k+1})$ ,  $j = i, \dots, k$ , i.e.  $v_i = v_{i+1} = \dots = v_k$  Each constraint  $j$ ,  $j = i, \dots, k-1$ , is feasible and may be non-binding ,

with  $\sum_{r=1}^j x_r \leq b_j$  and  $(v_j - v_{j+1})[\sum_{r=1}^j x_r - b_j] = 0$ . It is known (from the test at label L3) that some

constraint  $m$ ,  $k \leq m \leq n$ , is binding. For a binding constraint  $m$ , with  $\sum_{r=1}^m x_r = b_m$  then  $(v_m -$

$v_{m+1})[\sum_{r=1}^m x_r - b_m] = 0$  for  $m \neq n$  and  $v_m[\sum_{r=1}^m x_r - b_m] = 0$  for  $m = n$ . If the procedure terminates

after the execution of label L2 let  $i = i^{s+1}$  and  $v_j = 0$ ,  $j = i, \dots, n$ . Thus  $(v_j - v_{j+1})[\sum_{r=1}^j x_r - b_j] = 0$

for  $j = i, \dots, n-1$  and  $v_n[\sum_{r=1}^n x_r - b_n] = 0$ . Thus conditions (10) are satisfied.

### 3. Computational Experience

#### 3.1 Generation of test problems

The algorithm described in this paper was implemented in ANSI-C on a Silicon Graphics workstation. In order to evaluate the quality of the bounds generated by the **dual** procedure a set of computational experiments was designed and conducted. All times reported are in seconds. Time for I/O and generation of random test data is not included.

Nine groups of test problems were generated; each group using fixed values of the following data:  $n$  the number of variables and  $B$  the maximum difference in  $b_i - b_{i-1}$ ,  $i = 2, \dots, n$ . Each group contained ten problems generated in the following manner ( $U[c,d]$  denotes the discrete uniform distribution between  $c$  and  $d$ ).

$b_1$  is randomly drawn from  $U[B,2B]$  and  $b_i$  from  $U[b_{i-1}, b_{i-1} + B]$ ,  $i = 2, \dots, n$ . Note that  $B \leq b_i \leq (i+1)B$ ,  $1 \leq i \leq n$ , and the expected value of  $b_i$  is  $(i+2)B/2$ .

$F_i$ , the number of linear segments defining the  $f_i$  function, is drawn randomly from  $U[1, b_i]$ ; the slope for segment 1 from  $U[nB/2, 3nB/2]$  and the slope for segment  $m$  from  $U[\text{slope}_{m-1}, \text{slope}_{m-1} + 4]$ ,  $m = 2, \dots, F_i$ ;  $i = 1, \dots, n$ . With this method of generating slope<sub>1</sub> there is a possibility of generating negative slopes for the larger values of  $m$  and  $i$ . However, the problems generated were checked to ensure that this did not occur and the method does ensure that the initial slopes are not unreasonably large.

$G_i$  the number of linear segments defining the  $g_i$  function is drawn randomly from  $U[1, b_i]$ ; the slope for segment 1 from  $U[1, 4]$  and the slope for segment  $m$  from  $U[\text{slope}_{m-1}, \text{slope}_{m-1} + 4]$ ,  $m = 2, \dots, G_i$ ;  $i = 1, \dots, n$ .

Finally, in order to avoid infeasibility or trivial optimality,  $b_0 = (g^0 + g^y)/2$  where  $g^0$  and  $g^y$ ,  $y = 0$ , are as generated in **dual**.

The quality of the bounds is measured by  $100[h^0 - f^0]/f^0$  which represents an upper bound on the percentage deviation of the optimal objective function value from that of  $f^0$ . The effectiveness of any bounding procedure is also related to the effort needed to compute it. Table 1 makes an attempt to capture both these aspects. Table 1 contains five rows for each group of problems. The first/second ( third/fourth ) row reports the minimum, mean and maximum of the computation time/number of dual iterations when not using (using) the initialisation heuristic. The number of dual iterations required is of importance in instances of PRIMAL when constraint (1) represents a " soft goal" constraint ( in this case the iterations could be stopped as soon as a specified tolerance has been achieved) or a second criterion (and the value of  $b_0$  is implicitly defined by a decision maker's value function; the comparison of  $g^y$  and  $b_0$  within **dual** is then replaced by interactive questioning of the decision maker's preferences as outlined in Walker[12]). The fifth row reports the minimum, mean and maximum of the relative gap. It can be seen that in all cases high quality feasible solutions were found very quickly. The bounds generally improve as the number of variables and/or the range of the Uniform distribution from which the  $b_i$  were drawn increase. The number of dual iterations is small and seems insensitive to changes in both the number of variables and the range of the Uniform distribution from which the  $b_i$  were drawn. The initialisation heuristic provides a significant reduction in solution time. However, for those problems where constraint (1) represents a "soft goal" or second criterion the number of dual iterations is significantly higher when the initialisation heuristic is used. For such problems, it is recommended that no such heuristic be used.

**Table 1**  
**Computation time, number of dual iterations and relative gap**

		32			n 64			128		
		min	mean	max	min	mean	max	min	mean	max
<b>B</b>	<b>64</b>	0.16	0.27	0.42	0.61	0.79	0.93	2.21	2.43	2.74
		11	12.10	13	11	13.20	15	13	14.30	15
	<b>128</b>	0.07	0.13	0.17	0.28	0.36	0.44	0.90	1.14	1.46
		14	17.80	24	16	22.00	30	23	30.00	40
		0.00018	0.00438	0.01405	0.00007	0.00116	0.00352	0.00003	0.00061	0.00177
	<b>256</b>	0.35	0.48	0.61	1.37	1.65	1.76	4.17	4.90	5.92
		12	12.90	14	13	14.20	15	14	15.10	17
	<b>128</b>	0.16	0.20	0.24	0.49	0.64	0.76	1.41	2.06	2.43
		15	22.90	28	22	29.40	39	23	39.40	66
		0.00017	0.00230	0.00972	0.00015	0.00058	0.00125	0.00001	0.00021	0.00054
	<b>256</b>	0.82	1.21	1.53	2.32	3.05	3.69	6.30	10.53	13.72
		13	14.10	15	14	14.90	16	15	16.30	17
	<b>128</b>	0.27	0.42	0.52	0.91	1.10	1.41	2.51	4.20	5.36
		27	33.00	46	33	40.70	50	43	63.20	103
		0.00003	0.00077	0.00257	0.00000	0.00012	0.00036	0.00001	0.00009	0.00021

Note that the problems generated are fairly large and if formulated as 0-1 integer programming problems with a multiple-choice constraint representing the  $b_i+1$  alternative values of the integer variable  $x_i$  the number of 0-1 variables required for the "average" problem generated is given by  $\sum_{i=1}^n \{(i+2)B/2+1\} = (n^2+5n)B/4+n$ . Thus, for  $n = 128$  and  $B = 256$  the expected number of 0-1 variables is 1089664.

#### 4. Conclusions

In all cases high quality feasible solutions were found very quickly. In conclusion the procedure **dual** seems to be an efficient tool for solving problems of the type described in the paper. Further, it is self-contained and coded in widely supported ANSI-C.

## References

- [1] R.D. Armstrong, P. Sinha and A.A. Zoltners, The nested multiple-choice knapsack model, *Management Sci.* **28** (1982) 34-43.
- [2] M.E. Dyer, W.O. Riha and J. Walker, A hybrid dynamic programming / branch and bound algorithm for the multiple choice knapsack problem, *J. Comput. Appl. Math.* **58** (1995) 43-54.
- [3] M.E. Dyer and J. Walker, A simple graphical method for a production/purchasing problem, in: K.L. Chew, H.L. Ong, F.S. Chong et. al. (Editors) *Operational Research for Decision Support* (Operational Research Society of Singapore Press, 1985) 26-35.
- [4] M.E. Dyer and J. Walker, An algorithm for a separable integer programming problem with cumulatively bounded variables, *Discrete Applied Mathematics* **16** (1987), 135-149.
- [5] M.E. Dyer, Calculating surrogate constraints, *Math. Prog.* **19** (1980) 255-278.
- [6] G.N. Frederickson and D.B. Johnson, The complexity of selection and ranking in  $X + Y$  and matrices with sorted columns, *J. of Computing and System Sciences* **24** (1982) 197-208.
- [7] M.L. Fisher, The Lagrangian relaxation method for solving integer programming problems, *Management Sci.* **27** (1981) 1-18.
- [8] H J Greenberg, The one-dimensional generalised Lagrange multiplier problem, *Oper Res.* **25** (1977) 338-345.

- [9] L.P. Mavrides, *Nonlinear programming with cumulatively bounded variables*, *J. Comput. Appl. Math.*, **5** (1979) 163-169.
- [10] A. Tamir, Efficient algorithms for a selection problem with nested constraints and its application to a production-sales planning problem, *SIAM J. Control Optimisation*, **18** (1980) 282-287.
- [11] A. Tamir, Further remarks on selection problems with nested constraints, Department of Statistics, Tel Aviv University, 1979.
- [12] J. Walker, An interactive method as an aid in solving bicriterion mathematical programming problems, *J. Opl. Res. Soc.* **29** (1978) 915-922.