KMITL Sci. J. Vol. 4 No. 2 Jul-Dec. 2004

CLOSE-FORM EXACT ALGORITHM FOR MINIMIZING MAXIMUM PROCESSING TIME IN THE UNBOUNDED KNAPSACK PROBLEM

Chanin Srisuwannapa[1*] and Peerayuth Chansethikul[2]

[1]Department of Applied Statistics, Faculty of Science,
King Mongkut's Institute of Technology Ladkrabang,
Chalongkrung Rd., Ladkrabang, Bangkok, 10520, Thailand.
[2]Department of Industrial Engineering, Faculty of Engineering, Kasetsart University,
50 Phahonyothin Rd., Lardyaw, Jatujak, Bangkok, 10900, Thailand.

ABSTRACT

We address a variant of the unbounded knapsack problem (UKP) into which the processing time of each item is also put and considered, referred as MMPTUKP problem. The problem is a decision of allocating amount of n items such that the maximum processing time of the selected items is minimized and the total profit is gained as at least as determined without exceeding capacity of knapsack (budget). In this paper, we proposed two new modified exact algorithms for this problem. The first algorithm, CFMMPTUKP algorithm, has applied the close-form method with the original algorithm, MMPTUKP, and the second one, CTCFMMPTUKP algorithm, has applied the coordinate transformation with CFMMPTUKP algorithms. We present computational experiments with 7 different type of problems for which data were generated to validate our ideas and demonstrate the efficiency of the proposed algorithms. It can be concluded that, for all type of problems, the proposed CTCFMMPTUKP algorithms performs in term of solution time better than the 5 other algorithms.

KEYWORDS: Linear programming, Simplex method, Integer linear programming, Branch and bound algorithm, Unbounded and bounded knapsack problem, Processing time

1. INTRODUCTION

One of the practices complexity that most of operations research must deal with in numerous decisions is to decide which subset of n items or projects should be selected such that the total profit sum of the selected items or projects is maximized, without exceeding the capital budget, called generally the knapsack problem (KP). This problem can be formulated as mathematical model and is one of an NP-hard combinatorial optimization problem which can be solved successfully by various exact algorithms. The commonly used techniques are the (old fashion) dynamic programming and branch-and-bound methods and the (latest fashion) branch-and-cut and branch-and-price methods as described in Toth [1]. This kind of problem can be applied to many real world situations. In the following, we will provide a short historical overview of many kinds of this problem and various recent algorithms.

For the 0-1 knapsack problem (KP), it involves with selecting items or projects to maximize total profit and a hybrid algorithm has been recently proposed by Martello *et al.* [2]. The 0-1 multidimensional knapsack (0-1 MKP) is a generalization of the 0-1 knapsack problem. Heuristics can be found in Freville [3]. The Multiple Knapsack problem (MKP) is

*Corresponding author. Tel: 662-7373000 ext 6163, E-mail: chanin_sri@yahoo.com the problem of assigning a subset of n items to m distinct knapsacks to maximize total profits. The new exact algorithm is proposed by Psinger [4]. A different algorithm is presented by Martello and Toth [5]. The Multiple-Choice Knapsack Problem (MCKP) is defined as a 0-1 Knapsack Problem with the addition of disjoined multiple-choice constraints. Several algorithms for MCKP have been presented during the last two decades: e.g., Psinger [6]. The budgeting problem with bounded multiple choice constraints (BBMC) is a generalization of the multiple choice knapsack problem (MCKP) and the algorithms for it was presented in Psinger [7]. The Unbounded knapsack problem (UKP) is one of knapsack problem. A new algorithm (Dynamic programming revisited) was presented by Andonov *et al.* [8].

Problem of minimizing maximum processing time in unbounded knapsack problem (MMPTUKP) is about allocating amount of n items such that the maximum processing time of the selected items is minimized and the total profit is gained as at least as determined without exceeding knapsack capacity (budget). To the best our knowledge, this problem has been studied only by Srisuwannapa and Charnsettikul [9-11] in the literature. So the objective of this paper is to propose the two new modified algorithms. The first algorithm, CFMMPTUKP algorithm, has applied the close-form method with the original algorithm, MMPTUKP, and the second one, CTCFMMPTUKP algorithm, has applied the coordinate transformation with CFMMPTUKP algorithms. We also made computational experiments with 7 different types of generated data to validate our ideas and demonstrate the efficiency of new proposed algorithms. Coordinate transformation applied with linear and integer linear programming problem to rearrange data before solving was stated in section 2. Section 3 is about problem model. The original, other and new proposed algorithms were explained in section 4 and 5. Computational experiment and conclusions were stated in section 6 and 7 respectively.

2. COORDINATE TRANSFORMATION WITH LINEAR AND INTEGER LINEAR PROGRAMMING

Coordinate transformation is method of changing the coordinate system of the LP and ILP to be the transformed LP and ILP. It has been applied in the pre-solve step to rearrange data first before solving the transformed LP and ILP, since we believe that doing this may speed up the solution time. This reason is based on the following below concepts.

As applying two-phase method in solving LP, phase I of the simplex method can end at a basic feasible solution (bfs) that is far from the optimal solution, in which case, phase II will require many iterations before obtaining an optimal solution. To reduce the effort, we propose changing the coordinate system of the LP in such a way that phase I generates an initial extreme point on the "proper" side of the feasible region, that is, geometrically closer to the optimal solution.

Now we describe how to convert the original LP and ILP problem in standard form into the transformed problem. Consider an LP in the standard form(for ILP, all x are integer):

$$\begin{aligned} & \min \ cx \\ & s.t. \ Ax = b \\ & l \le x \le u \end{aligned}$$

the coordinate transformation can be implemented by using the following substitution.

$$x_j = \begin{cases} l_j + v_j \ , \ if \ c_j \ge 0 \\ u_j - v_j \ , \ if \ c_j < 0 \end{cases}$$

After performing this change, the result will be the following equivalent LP and ILP, transformed LP and ILP (TLP/TILP), in the *v*-coordinate system:

$$\begin{aligned} &\min\ \bar{c}^T v \\ &s.t.\ \bar{A}v = \bar{b} \\ &0 \le v \le \bar{u} \end{aligned} \qquad \text{(TLP/TILP)}$$

where

$$\bar{c}_j = \begin{cases} -c_j, & if\ c_j < 0 \\ c_j, & if\ c_j \ge 0 \end{cases}$$

$$\bar{A}_j = \begin{cases} -A_j, & if\ c_j < 0 \\ A_j, & if\ c_j \ge 0 \end{cases}$$

$$\bar{b} = b - \sum_{j=1}^{n} A_j * \begin{cases} u_j, & if\ c_j < 0\ and\ u_j < +\infty \\ l_j, & if\ c_j \ge 0\ and\ l_j > -\infty \end{cases}$$

$$\bar{u}_j = \begin{cases} u_j - l_j, & if\ c_j < 0\ and\ u_j < +\infty \\ u_j + l_j, & if\ c_j \ge 0\ and\ l_j > -\infty \end{cases}$$

We then solve TLP/TILP with the simplex algorithm to get the optimal solution, v^*, for the transformed problem. Finally, the optimal solution for the original problem is:

$$x_j^* = \begin{cases} l_j + v_j^*, & if\ c_j \ge 0 \\ u_j - v_j^*, & if\ c_j < 0 \end{cases}$$

3. PROBLEM MODEL

The problem studied in this paper can be formulated as follows. The problem is given a knapsack with capacity *C* and expected profit of at least *B* into which we may put *n* types of objects. All parameters corresponding to each object of type *i* such as the profit, p_i, the weight, w_i, and the processing time, t_i and *n, B,* and *C* are all positive integers. The number of copies of each object type are unbounded. The problem calls for selecting the set of n items with minimizing the maximum processing time, and must gain profit of at least *B* without exceeding the knapsack capacity (budget, *C*). Mathematically, the problem can be expressed as the following integer linear programming formulation.

$$\begin{aligned} &\min\ T \\ &s.t. \quad T \ge t_i x_i \\ &\sum_{j=1}^{n} p_j x_j \ge B \\ &\sum_{j=1}^{n} w_j x_j \le C \\ &\quad x_j \ge 0\ and\ integer,\ j = 1,\dots,n \end{aligned}$$

4. EXACT ALGORITHMS

4.1 MMPTUKP ALGORITHM

Its steps are as follows [9]:

Step 1: Solve the following unbounded sub problem, unbounded knapsack problem, by an exact algorithm, branch-and-bound method.

$$\begin{aligned} max \quad & \sum_{j=1}^{n} p_j x_j = z^* \\ s.t. \quad & \sum_{j=1}^{n} w_j x_j \le C \\ & x_j \ge 0 \text{ and integer}, \ j = 1,\dots,n \end{aligned}$$

Then check whether z^* is greater than to B or not. If yes, let $To = \underset{\forall j}{Max}\{t_j x_j\}$, $Tu = To$, $Tl = 0$, then proceed to *Step* 2. If not, there is no feasible solution, and then stop.

Step 2: Let $T^*=(Tu+Tl)/2$ and solve the unbounded knapsack problem from *Step 1* with the bounded variable constraints, $0 \le x_j \le \lfloor T^* / t_j \rfloor$ and integer, $j = 1,2,\dots, n$, called bounded knapsack problem (BKP), by using the well-known branch-and-bound method, then proceed *Step 3*.

Step 3: Check if z^* is greater than or equal to B, then let $Tu=T^*$, and proceed *Step 4*, otherwise let $Tl=T^*$, and then proceed *Step2*.

Step 4: Check if $Tu-Tl$ is less than or equal ε, then the last solution obtained from *Step 2* is an optimal solution with T^* under tolerance ε, and stops. If not, proceed *Step 2*.

4.2 APPLICATION OF COORDINATE TRANSFORMATION WITH MMPTUKP (CTMMPTUKP)

We proposed application of coordinate transformation with MMPTUKP, referred as CTMMPTUKP algorithm [10]. Before applying coordinate transformation, first calculate upper bound for each x, ub[x_j] = $\lfloor C/w_j \rfloor$ and transform the bounded sub problem, bounded knapsack problem (ILPUKP), into the transformed bounded knapsack problem (TILPUKP) and then solve it by an branch-and-bound method. For other its steps, since it have already had upper bound from the previous step, it then only transforms the ILPUKP into TILPUKP and then branch and bound method and repeat until getting optimal solution.

4.3 APPLICATION OF LINEAR PROGRAMMING WITH MMPTUKP (LPMMPTUKP)

LPMMPTUKP's steps are as follows [11]:

Step 1: Solve the following unbounded sub problem, unbounded knapsack problem, by an exact algorithm, branch-and-bound method.

$$\begin{aligned} max \quad & \sum_{j=1}^{n} p_j x_j = z^* \\ s.t. \quad & \sum_{j=1}^{n} w_j x_j \le C \\ & x_j \ge 0 \text{ and integer}, \ j = 1,\dots,n \end{aligned}$$

Then check whether z^* is greater than to B or not. If yes, let $To = \underset{\forall j}{Max}\{t_j x_j\}$, $Tu = To$, $Tl = 0$, then proceed to *Step* 2. If not, there is no feasible solution, and then stop.

Step 2: Let $T^*=(Tu+Tl)/2$ and solve the relaxed bounded knapsack problem from *Step 1* with the new bounded variable constraints, $0 \le x_j \le \lfloor T^*/t_j \rfloor$, $j = 1,2,\dots,n$, by using simplex method. After that, let $x_j^* = \lfloor x_j \rfloor$, $\sum_{j=1}^{n} p_j x_j^* = z^*$ and then proceed to *Step 3*.

Step 3: Check if z^* is greater than B or not. If yes, let $Tu=T^*$ and then proceed to *Step 4*. If not, solve the knapsack problem form *Step 1* with the bounded variable constraints, $0 \le x_j \le \lfloor T^*/t_j \rfloor$ *and integer*, $j = 1,2,\dots,n$, by using the branch-and-bound method. Check if z^* is greater than or equal to B or not. If yes, then let $\tilde{T}u=T^*$, and proceed to *Step 4*. If not, let $Tl=T^*$, and then proceed to *Step 5*.

Step 4: Check if Tu-Tl is less than or equal ε, then the last obtained solution from *Step 2* or *3* is an optimal solution with T^* under tolerance ε and stop. If not, proceed to *Step 2*.

Step 5: Let $T^*=(Tu+Tl)/2$ and solve the relaxed bounded knapsack problem from *Step 1* with the new bounded variable constraints, $0 \le x_j \le \lfloor T^*/t_j \rfloor$, $j = 1,2,\dots,n$, by using simplex method and then proceed to *Step 6*.

Step 6: Check if z^* is greater than B or not. If not, let $Tl=T^*$, and then proceed to *Step 5*. If yes, solve the knapsack problem form *Step 1* with the bounded variable constraints, $0 \le x_j \le \lfloor T^*/t_j \rfloor$ *and integer*, $j = 1,2,\dots,n$, by using the branch-and-bound method. Check if z^* is greater than or equal to B or not. If yes, let $Tu=T^*$, and proceed to *Step 7*. If not, let $Tl=T^*$, and then proceed to *Step 5*.

Step 7: Check if Tu-Tl is less than or equal ε, then the last obtained solution from *Step 6* is an optimal solution with T^* under tolerance ε and stop. If not, proceed to *Step 5*.

4.4 APPLICATION OF COORDINATE TRANSFORMATION WITH LPMMPTUKP (CTLPMMPTUKP)

We proposed application of coordinate transformation with MMPTUKP, referred as CTLPMMPTUKP algorithm [11]. Before applying coordinate transformation, first calculate upper bound for each x, ub[x_j] = $\lfloor C/w_j \rfloor$ and transform the bounded sub problem, bounded knapsack problem (ILPUKP), into the transformed bounded knapsack problem (TILPUKP) and then solve it by an branch-and-bound method. For other its steps, since it have already had upper bound from the previous step, it then only transforms the ILPUKP or LPUKP, relaxed bounded knapsack problem, into TILPUKP or TLPUKP, the relaxed transformed bounded knapsack problem, and then branch and bound method and repeat until getting an optimal solution.

5. TWO NEW MODIFIED EXACT ALGORITHMS

5.1 APPLICATION OF CLOSE-FORM SOLUTION TECHNIQUE WITH MMPTUKP (CFMMPTUKP)

For LPMMPTUKP algorithm, simplex method was used for solving some relaxed bounded knapsack problems (RBKP). For solving these RBKP problems, there is also another technique for solving it. That technique is called close-form technique and then will be applied in CFMMPTUKP algorithm.

CFMMPTUKP's steps are as follows:

Step 1: Solve the following unbounded knapsack problem, UKP, by branch and bound method

$$\begin{aligned} & max \sum_{j=1}^{n} p_j x_j = z^* \\ & s.t. \sum_{j=1}^{n} w_j x_j \le C \qquad \text{(UKP)} \\ & \quad x_j \ge 0 \text{ and integer}, j = 1,2,\dots,n \end{aligned}$$

and then check whether z^* is greater than B or not. If yes, then let $\text{To} = \underset{\forall j}{\text{Max}}\{t_j x_j\}$, $\text{Tu} = \text{To}$, $\text{Tl} = 0$, then proceed to *Step 2*. If not, there is no feasible solution and stop.

Step 2: from the above problem, compute ratio, $\frac{p_j}{w_j}$, for all *j*, and then sort those above ratios from largest to smallest value by using quick sort.

Step 3: Let $T^*=(Tu+Tl)/2$ and solve the relaxed bounded knapsack problem from *Step 1* with the new bounded variable constraints, $0 \le x_j \le \lfloor T^* / t_j \rfloor$, $j = 1,2,\dots,n$, by using close-form solution method as follows.

Close-form solution method

choose x_j having the biggest value of ratio $\frac{p_j}{w_j}$, let $x_j = min\{C, \lfloor T^* / t \rfloor\}$, compute remaining C, $C_{reml} = C - min\{C, \lfloor T^*/t \rfloor\}$, and then repeat by choosing x_j having the value of ratio $\frac{p_j}{w_j}$ which is less than the previous one until there is no C remaining.

After that, let $x_j^* = \lfloor x_j \rfloor$, $\sum_{j=1}^{n} p_j x_j^* = z^*$, and then proceed to *Step 4*.

Step 4: Check if z^* is greater than B or not. If yes, let $Tu=T^*$ and then proceed to *Step 5*. If not, solve the knapsack problem form *Step 1* with the bounded variable constraints, $0 \le x_j \le \lfloor T^* / t_j \rfloor$ and integer, $j = 1,2,\dots,n$, by using the branch - and - bound method, by using the branch-and-bound method. Check if z^* is greater than or equal to B or not. If yes, then let $Tu=T^*$, and proceed to *Step 5*. If not, let $Tl=T^*$, and then proceed to *Step 6*.

Step 5: Check if *Tu-Tl* is less than or equal ε, then the last obtained solution from *Step 2* or *3* is an optimal solution with T^* under tolerance ε and stop. If not, proceed to *Step 3*.

Step 6: Let $T^*=(Tu+Tl)/2$ and solve the relaxed bounded knapsack problem from *Step 1* with the new bounded variable constraints, $0 \le x_j \le \lfloor T^* / t_j \rfloor$, $j = 1,2,\dots,n$, by using close-form solution method and then proceed to *Step 7*.

Step 7: Check if z^* is greater than B or not. If not, let $Tl=T^*$, and then proceed to *Step 5*. If yes, solve the knapsack problem form *Step 1* with the bounded variable constraints, $0 \le x_j \le \lfloor T^* / t_j \rfloor$ *and integer*, $j = 1,2,\dots,n$, by using the branch-and-bound method. Check if z^* is greater than or equal to B or not. If yes, let $Tu=T^*$, and proceed to *Step 8*. If not, let $Tl=T^*$, and then proceed to *Step 6*.

Step 8: Check if *Tu-Tl* is less than or equal ε, then the last obtained solution from *Step 2* or *3* is an optimal solution with T^* under tolerance ε and stop. If not, proceed *Step 6*.

5.2 APPLICATION OF COORDINATE TRANSFORMATION WITH CFMMPTUKP (CTCFMMPTUKP)

We proposed application of coordinate transformation with CFMMPTUKP, referred as CTCFMMPTUKP algorithm. For step that must solve the bounded sub problem, bounded knapsack problem (ILPUKP), coordinate transformation was applied by transforming ILPUKP into the transformed bounded knapsack problem (TILPUKP) and then solve it by an branch-and-bound method and repeat until getting optimal solution.

To verify the correctness of the proposed algorithms, the following theorems are proven as follows.

This following proportion 1 verifies the correctness at step 3 of LPMMPTUKP, CFMMPTUKP algorithm.

Proportion 1: If objective function value obtained from rounding down x_j, Z_{rdown}, is greater than or equal to B, then the objective function value obtained from branch-and-bound method, Z_{bb}, is also greater than B.
Proof: Let the objective function value obtained from rounding down x_j, be Z_{rdown}, the objective function value obtained from branch-and-bound method, Z_{bb}. If $B \leq Z_{rdown}$, since $Z_{rdown} \leq Z_{bb}$ by definition for maximum problem, then $B \leq Z_{bb}$.

This following proportion 2 verifies the correctness at step 6 of LPMMPTUKP, CFMMPTUKP algorithm.

Proportion 2: *I*f objective function value obtained from LP, Z_{LP}, is less than B , then the objective function value obtained from rounding down x_j, Z_{rdown}, is also less than B.
Proof: Let the objective function value obtained from rounding down x_j, be Z_{rdown}, the objective function value obtained from LP, Z_{LP}. If $Z_{LP} < B$, since $Z_{rdown} \leq Z_{LP}$ by definition for maximum problem, then $Z_{rdown} \leq B$.

The following theorem 3 is for verifying the correctness of all algorithms.

Theorem 3: Algorithm terminates with a feasible solution under optimality of T^*
Proof: Let there exist $T^0 < T^* - \varepsilon$ be the optimal value of T^* with a feasible solution, $x_j^0, \forall j$ and the total profits $Z^0 \geq B$. Then, solve the bounded knapsack problem(BKP) with the new upper bound $\lfloor T^*/t_j \rfloor$, with $T^*=T^0$ for all x_j. If $Z^*<B$ then T^0 does not exist as claimed. Otherwise, $Tu=T^0$ which leads tothe impossible case that $Tu > Tl$. By contradiction, T^* is optimal as stated.

6. PERFORMANCE EVALUATION EXPERIMENTS

We have investigated how all algorithms behaves for 7 different types of data generated. All algorithms were coded in C language and solved partly by CPLEX package. The solution time for the same problem for which was solved separately by each algorithm was measured in second. For each problem, 10 instances were generated within any value interval. All experiments were run on a Compaq Presario B2000 Notebook with specification: Intel Pentium M Processor 1.4 GHz, 256 MB RAM, 30 GB hard drive/4200 rpm. The results in term of average and median of solution time in second from 10 generated instances were presented in table.

In this paper, 7 different types of data were used to validate the efficiency of 6 algorithms especially the two new modified algorithms and were as follows.

1. Uncorrelated data
2. Weakly correlated data
3. Strongly correlated data

4. Inverse strongly correlated data
5. Almost strongly correlated data
6. Subset sum data
7. Uncorrelated data with similar weights

Table 1. Average solution time, W's sig., H-test and H-test's sig. from 10 generated samples of uncorrelated data with 10000 variables

Run number	Run label	Factor[a]				MMPTUK	CTMM-PTUK	W sig.	LPMM-PTUK	CTLPMM-PTUK	W sig.	CFMM-PTUK	CTCFMM-PTUK	W sig.	H-test(sig.)
		A	B	C	D										
1	(1)	-	-	-	-	33.35	34.17	0.32	31.29	31.36	0.85	2.0349	2.0248	0.73	0.00**
2	a	+	-	-	-	35.058	35.755	0.34	33.676	33.672	0.90	2.0339	2.0329	0.93	0.00**
3	b	-	+	-	-	64.47	64.90	0.79	76.27	76.35	0.85	19.134	19.582	0.62	0.00**
4	ab	+	+	-	-	38.33	39.16	0.67	36.69	36.72	0.90	2.073	2.0919	0.62	0.00**
5	c	-	-	+	-	35.62	36.48	0.21	33.39	33.45	0.79	2.048	2.065	0.57	0.00**
6	ac	+	-	+	-	37.00	37.92	0.38	35.48	35.50	0.96	2.050	2.042	0.73	0.00**
7	bc	-	+	+	-	72.67	73.79	0.09	84.151	84.272	0.90	21.8759	20.812	0.67	0.00**
8	abc	+	+	+	-	44.481	45.680	0.12	42.600	42.640	0.88	2.127	2.135	0.85	0.00**
9	d	-	-	-	+	34.53	35.31	0.18	32.43	32.46	0.90	2.064	2.045	0.27	0.00**
10	ad	+	-	-	+	35.604	36.340	0.30	34.167	34.187	1.00	2.044	2.037	0.67	0.00**
11	bd	-	+	-	+	54.24	54.86	0.62	59.88	59.84	0.79	13.37	12.40	0.67	0.00**
12	abd	+	+	-	+	39.771	40.626	0.42	38.072	38.105	0.96	2.091	2.104	0.65	0.00**
13	cd	-	-	+	+	39.161	40.148	0.01**	36.788	36.815	0.67	2.070	2.070	0.91	0.00**
14	acd	+	-	+	+	37.653	38.446	0.32	36.066	36.074	0.97	2.076	2.060	0.54	0.00**
15	bcd	-	+	+	+	73.322	74.262	0.18	89.090	92.200	0.52	24.897	26.530	0.82	0.00**
16	abcd	+	+	+	+	40.509	41.516	0.04*	38.730	38.759	0.91	2.103	2.088	1.00	0.00**

*$p<0.05$, **$p< 0.01$,

[a] Factor A = objective function coefficient(c_j): low level(-):c_j = [1,1000], high level(+): c_j = [1,10000], Factor B = matrix coefficient(w_j): low level(-): w_j = [1,1000], high level(+): w_j = [1,10000], Factor C = right hand size(RHS): low level(-): RHS = 0.2*Sum, high level(+): RHS = 0.8*Sum, where Sum= $\sum_{l=j}^{n} w_j$, Factor D = processing time(t_j): low level (-): t_j = [1,1000], high level(+):t_j = [1,10000]

Table 2 Average solution time, W's sig., H-test and H-test's sig. from 10 generated samples of weakly correlated data with 10000 variables

Run number	Run label	Factor[a]			MMPTUK	CTMM-PTUK	W sig.	LPMM-PTUK	CTLPMM-PTUK	W sig.	CFMM-PTUK	CTCFMM-PTUK	W sig.	H test(sig.)
		A	B	C										
1	(1)	-	-	-	32.262	32.902	0.42	28.327	28.327	1.00	1.883	1.871	0.57	0.00**
2	a	+	-	-	41.532	42.708	0.21	39.669	39.730	0.96	2.099	2.113	0.94	0.00**
3	b	-	+	-	34.141	35.087	0.21	30.010	30.034	0.76	1.911	1.917	0.10	0.00**
4	ab	+	+	-	43.500	44.620	0.14	41.520	41.550	0.97	2.086	2.090	0.91	0.00**
5	c	-	-	+	33.125	33.813	0.21	29.027	29.027	0.94	1.860	1.850	0.57	0.00**
6	ac	+	-	+	40.492	41.302	0.21	38.521	38.552	0.85	2.102	2.132	0.17	0.00**
7	bc	-	+	+	35.911	36.808	0.09	31.644	31.653	0.87	1.926	1.914	0.52	0.00**
8	abc	+	+	+	40.739	41.544	0.02**	38.857	38.920	0.73	2.143	2.148	0.70	0.00**

*$p<0.05$, **$p< 0.01$,

[a] Factor A = matrix coefficient(w_j): low level(-): w_j = [1,1000], high level(+): w_j = [1,10000], Factor B = right hand size(RHS): low level(-): RHS = 0.2*Sum, high level(+): RHS = 0.8*Sum, where Sum= $\sum_{l=j}^{n} w_j$, Factor C = processing time(t_j) : low level(-) : t_j = [1,1000], high level(+): t_j = [1,10000], objective function coefficient(c_j): low level(-): c_j = [w_j -100, w_j +100] if w_j = [1,1000], low level(+):c_j = [w_j -1000, w_j +1000] if w_j = [1,10000]

Table 3 Median of solution time, W's sig., H-test and H-test's sig. from 10 generated samples of strongly correlated data with 10000 variables

Run number	Run label	Factor[b] A	B	C	MMPTUK	CTMM-PTUK	W sig.	LPMM-PTUK	CTLPMM-PTUK	W sig.	CFMM-PTUK	CTCFMM-PTUK	W sig.	H-test(sig.)
1	(1)	-	-	-	17.43	8.00(1)[a]	0.00**	1.732	1.753	0.70	0.530	0.536	0.85	0.00**
2	a	+	-	-	47.93	32.00(3)	0.62	17.315	17.290	0.85	1.472	1.482	0.65	0.00**
3	b	-	+	-	7.230	11(1)	0.06	1.890	1.917	0.67	0.570	0.566	0.73	0.00**
4	ab	+	+	-	31.680	32(2)	0.47	18.501	18.487	1.00	1.467	1.462	0.76	0.00**
5	c	-	-	+	14.880	10(3)	0.50	1.8787	1.887	0.57	0.535	0.531	0.80	0.00**
6	ac	+	-	+	36.39	31(3)	0.97	16.4496	16.449	0.73	1.502	1.492	0.82	0.00**
7	bc	-	+	+	7.271	9.439	0.01**	1.933	1.947	0.62	0.571	0.561	0.94	0.00**
8	abc	+	+	+	27.124	29.42	0.01**	16.429	16.599	0.50	1.497	1.527	0.34	0.00**

* $p<0.05$, ** $p< 0.01$, [a] number of instances that spend solution time lager than 3600 second.
[b] Factor A = matrix coefficient(w_j): low level(-):w_j = [1,1000], high level(+):w_j = [1,10000], Factor B = right hand size(RHS): low level(-): RHS = 0.2*Sum, high level(+): RHS = 0.8*Sum, where Sum= $\sum_{l=j}^{n} w_j$, Factor C = processing time(t_j): low level(-): t_j = [1,1000], high level(+): t_j = [1,10000], objective function coefficient(c_j): low level(-): c_j = w_j +100 if w_j = [1,1000], high level(+): c_j = w_j +1000 if w_j = [1,10000]

Table 4 Median of solution time, W's sig., H-test and H-test's sig. from 10 generated samples of inverse strongly correlated data with 10000 variables

Run number	Run label	Factor[b] A	B	C	MMPTUK	CTMM-PTUK	W sig.	LPMM-PTUK	CTLPMM-PTUK	W sig.	CFMM-PTUK	CTCFMM-PTUK	W sig.	H-test(sig.)
1	(1)	-	-	-	3.955	1804(4)[a]	0.00**	1.322	1.387	0.27	0.606	0.641	0.34	0.00**
2	a	+	-	-	13.399	1810(5)	0.00**	8.728	8.823	0.62	1.502	1.542	0.15	0.00**
3	b	-	+	-	4.777	7(1)	0.00**	1.497	1537	0.12	0.610	0.656	0.09	0.00**
4	ab	+	+	-	15.497	18(3)	0.00**	9.894	9.974	0.82	1.532	1.577	0.24	0.00**
5	c	-	-	+	4.186	8(3)	0.00**	1.533	1.527	0.57	0.666	0.691	0.27	0.00**
6	ac	+	-	+	13.594	23(2)	0.00**	8.718	8.803	0.50	1.562	1.602	0.11	0.00**
7	bc	-	+	+	4.942	7(2)	0.00**	1.632	1.652	0.62	0.696	0.706	0.80	0.00**
8	abc	+	+	+	15.146	16.925	0.01**	9.794	9.744	0.97	1.582	1.627	0.15	0.00**

* $p< 0.05$., ** $p< 0.01$, [a] number of instances that spend solution time lager than 3600 second.

[b] Factor A = objective function coefficient(c_j): low level(-) : c_j = [1,1000], high level(+): c_j = [1,10000], Factor B = right hand size(RHS): low level(-): RHS = 0.2*Sum, high level(+): RHS = 0.8*Sum, where Sum= $\sum_{l=j}^{n} w_j$, Factor C = processing time(t_j) : low level(-):t_j = [1,1000], high level(+) : t_j = [1,10000], matrix coefficient(w_j): w_j = c_j +100 if c_j is in [1,1000], w_j = c_j +1000 if c_j is in [1,10000]

Table 5 Median of solution time, W's sig., H-test and H-test's sig. from 10 generated samples of almost strongly correlated data with 10,000 variables

Run number	Run label	Factor[b]			MMPTUK	CTMM-PTUK	W sig.	LPMM-PTUK	CTLPMM-PTUK	W sig.	CFMM-PTUK	CTCFMM-PTUK	W sig.	H test(sig.)
		A	B	C										
1	(1)	-	-	-	22.29	17(1)[a]	0.03**	6.579	6.509	0.85	0.991	0.971	0.85	0.00**
2	a	+	-	-	49.90	48(4)	0.85	38.13	38.16	0.88	2.048	2.052	0.85	0.00**
3	b	-	+	-	17.090	18(1)	0.52	6.790	6.825	1.00	1.021	0.986	0.54	0.00**
4	ab	+	+	-	44.64	45(1)	0.80	41.64	41.67	0.85	2.083	2.083	0.70	0.00**
5	c	-	-	+	18.822	17(1)	0.62	6.560	6.624	1.00	1.026	1.017	0.73	0.00**
6	ac	+	-	+	44.4645	43(4)	0.85	38.270	38.320	0.79	2.083	2.072	0.88	0.00**
7	bc	-	+	+	16.489	18(1)	0.14	6.860	6.875	0.67	1.031	1.031	0.70	0.00**
8	abc	+	+	+	40.808	42(2)	0.21	37.76	37.875	0.67	2.103	2.153	0.17	0.00**

* $p < 0.05$, ** $p < 0.01$, [a] number of instances that spend solution time lager than 3600 second.
[b] Factor A = matrix coefficient(w_j): low level(-): $w_j = [1,1000]$, high level(+): $w_j = [1,10000]$, Factor B = right hand size(C): low level(-): $C = 0.2*\text{Sum}$, high level(+): $C = 0.8*\text{Sum}$, where Sum = $\sum_{l=j}^{n} w_j$, Factor C = processing time(t_j): low level(-): $t_j = [1,1000]$, high level(+): $t_j = [1,10000]$, objective function coefficient(c_j): low level(-): $c_j = [w_j + 98, w_j + 102]$ if w_j is in $[1,1000]$, high level (+): $c_j = [w_j + 980, w_j + 1020]$ if w_j is in $[1,10000]$

Table 6 Median of solution time, W's sig., H-test and H-test's sig. from 10 generated samples of subset sum data with 10000 variables

Run number	Run label	Factor[a]			MMPTUK	CTMM-PTUK	W sig.	LPMM-PTUK	CTLPMM-PTUK	W sig.	CFMM-PTUK	CTCFMM-PTUK	W sig.	H test(sig.)
		A	B	C										
1	(1)	-	-	-	10.30	5.188	0.00**	2.85	1.301	0.00**	2.17	0.450	0.04*	0.00**
2	a	+	-	-	32.70	4.862	0.00**	3.74	2.328	0.00**	1.70	0.435	0.01*	0.00**
3	b	-	+	-	6.55	5.373	0.03*	1.34	1.342	0.67	0.50	0.491	0.85	0.00**
4	ab	+	+	-	5.13	5.213	0.73	2.57	2.58	0.42	0.49	0.495	0.50	0.00**
5	c	-	-	+	12.89	5.328	0.00**	3.18	1.306	0.01*	2.44	0.47	0.01*	0.00**
6	ac	+	-	+	63.7	4.667	0.00**	15.40	2.50	0.04*	13.4	0.47	0.13	0.00**
7	bc	-	+	+	8.19	5.758	0.02*	1.60	1.522	0.54	0.52	0.49	0.30	0.00**
8	abc	+	+	+	20.2	5.413	0.22	6.03	2.649	0.03*	3.66	0.536	0.07	0.00**

* $p<0.05$, ** $p< 0.01$,
[a] Factor A = matrix coefficient(w_j): low level(-): $w_j = [1,1000]$, high level(+): $w_j = [1,10000]$, Factor B = right hand size(RHS): low level(-): RHS $= 0.2*\text{Sum}$, high level(+): RHS $= 0.8*\text{Sum}$, where Sum= $\sum_{l=j}^{n} w_j$, Factor C = processing time(t_j): low level(-): $t_j = [1,1000]$, high level(+): $t_j = [1,10000]$, objective function coefficient(c_j): low level(-): $c_j = w_j$ from $[1,1000]$, high level(+): $c_j = w_j$ from $[1,10000]$

Table 7 Median of solution time, W's sig., H-test and H-test's sig. from 10 generated samples of uncorrelated data with similar weights with 10000 variables

Run number	Run label	Factor[b] A	Factor[b] B	MMPTUK	CTMM-PTUK	W sig.	LPMM-PTUK	CTLPMM-PTUK	W sig.	CFMM-PTUK	CTCFMM-PTUK	W sig.	H test(sig.)
1	(1)	-	-	3600(10)[a]	1.662	0.00**	3600(10)	1.648	0.00*	3600(10)	1.657	0.00**	0.00**
2	a	+	-	2(3)	1.652	0.67	2(3)	1.652	1.00	2(4)	1.647	0.88	0.00**
3	b	-	+	3600(10)	1.648	0.00**	3600(10)	1.652	0.000**	3600(9)	1.652	0.00**	0.00**
4	ab	+	+	1801(5)	1.647	0.49	2(3)	1.657	0.44	2(3)	1.652	0.80	0.00**

* $p< 0.05$, ** $p< 0.01$, [a] number of instances that spend solution time lager than 3600 second.
[b] Factor A = right hand size(RHS): low level(-): RHS = 0.2*Sum, high level(+): RHS = 0.8*Sum, where $\text{Sum}=\sum_{l=j}^{n} w_j$,Factor B = processing time(t_j): low level(-):t_j = [1,10], high level (+): t_j = [1,1000],matrix coefficient(w_j): w_j = [100000, 100100], objective function coefficient(c_j): c_j = [1,1000]

Since statistical test of each data's normal distribution indicated that all of them were not normally distributed, so non parametric statistics were used to test for significances. From above results, it can be concluded that, for all 7 different types of data, CTCFMMPTUKP performs better than the 5 other algorithms. For uncorrelated data, weakly correlated data, strongly correlated data, inverse strongly correlated data and almost strongly correlated data, CFMMPTUKP and CTCFMMPTUKP perform well equally and better the 4 others as well. But for subset sum data, uncorrelated data with similar weights, CTCFMMPTUKP performs better than CFMMPTUKP.

7. CONCLUSION

In this paper, we proposed two new modified exact algorithms, close-form exact algorithm (CFMMPTUKP) and application of coordinate transformation with close-form method (CTCFMMPTUKP), to minimize the maximum processing time of the selected items in unbounded knapsack problem(UKP) in which the total profit is also gained as at least as determined without exceeding capacity of knapsack(budget), called MMPTUKP problem. It can be concluded that, for all 7 different types of data, CTCFMMPTUKP performs better than the 5 other algorithms. For uncorrelated data, weakly correlated data, strongly correlated data, inverse strongly correlated data and almost strongly correlated data, CFMMPTUKP and CTCFMMPTUKP perform well equally and better the 4 others as well. But for subset sum data, uncorrelated data with similar weights, CTCFMMPTUKP performs better than CFMMPTUKP.

REFERENCES

[1] P. Toth, Optimization Engineering Techniques for the Exact Solution of NP-hard Combination Optimization Problem, *European Journal of Operation Research*, *125*, **2000**, 222-238.
[2] S. Martello, D. Psinger, and P. Toth, Dynamic Programming and Strong Bounds for the 0-1 Knapsack Problem, *Management Science*, *45*, **1999**, 414-424.
[3] A. Freville and G. Plateau, Heuristics and Reduction Methods for Multiple Constraints Linear Programming, *European Journal of Operations Research*, *24*, **1986**, 206-215.
[4] D. Psinger, An Exact Algorithm for Large Multiple Knapsack Problem, *European Journal of Operations Research*, *114*, **1999**, 528-541.
[5] S. Martello, and P. Toth, A Branch-and- Bound Algorithm for the Zero-One Knapsack Problem, *Discrete Applied Mathematics*, *3*, **1981**, 275-288.
[6] D. Psinger, A Minimal Algorithm for the Multiple-Choice Knapsack Problem, *European Journal of Operations Research, 83*, **1995**, 394-410.

[7] D. Psinger, Budgeting with Bound Multiple-Choice Constraints, *European Journal of Operations Research*, *129*, **2001**, 471-480.

[8] R. Andonow, V. Poirriez, and S. Rajopadhye, Unbounded Knapsack Problem: Dynamic Programming Revisited, *European Journal of Operations Research*, *123*, **2000**, 394-407.

[9] C. Srisuwannapa, and P. Charnsettikul, An Exact Algorithm for Solving the Unbounded Knapsack Problem with Minimizing Maximum Processing Time, *Proceedings of Seminar in Applied Optimization, A Publication of Industrial Engineering Department, Faculty of Engineering*, Kasetsart University, Bangkhen, Thailand, **2001**, 83-89.

[10] C. Srisuwannapa, and P. Charnsettikul, Application of Coordinate Transformation with MMPTUKP for Minimizing Maximum Processing Time in Unbounded Knapsack Problem, *Proceedings of The 1st KMITL International Conferences on Integration of Science and Technology for Sustainable Development*, KMITL, Ladkrabang, Bangkok, Thailand, *1*, **2004**, 25-28.

[11]C. Srisuwannapa, and P. Charnsettikul, *Modified Exact Algorithms for Minimizing Maximum Processing Time in the Unbounded knapsack Problem with, Proceedings of Meeting in Operation Researches in 2004,* Industrial Engineering Department, Faculty of Engineering, Kasetsart University, Bangkhen, Thailand, **2004**, 207-221.