KMITL Sci. J. Vol. 5 No. 1 Feb. 2005

APPLICATION OF COORDINATE TRANSFORMATION WITH CLOSE-FORM EXACT ALGORITHM FOR MINIMIZING MAXIMUM PROCESSING TIME IN THE UNBOUNDED KNAPSACK PROBLEM

Chanin Srisuwannapa[1*], 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 the new modified exact algorithm for this problem, CTCFMMPTUKP algorithms. It applied the coordinate transformation with CFMMPTUKP algorithm, close-form exact algorithm. We present computational experiments with 4 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 most types of problems, the proposed CTCFMMPTUKP algorithms performs in term of solution time faster 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 business and industrial problems that 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 its variant of this problem and various recent algorithms as follows. For the 0-1 knapsack problem(KP), a hybrid algorithm for this problem has been recently proposed by Martello et al.[2]. For the 0-1 multidimensional knapsack (0-1 MKP), heuristics for it can be found in Freville[3].

[*] Corresponding author. Tel: 02-7373000 ext 6163, e-mail: chanin_sri@yahoo.com

For the Multiple Knapsack problem (MKP), the new exact algorithm for it is proposed by Psinger[4] and a different algorithm is presented by Martello and Toth[5]. For the Multiple-Choice Knapsack Problem(MCKP) , several algorithms for MCKP have been presented during the last two decades: e.g., Psinger[6]. For the budgeting problem with bounded multiple choice constraints (BBMC), the algorithms for it was presented in Psinger[7]. For the Unbounded knapsack problem (UKP), a new algorithm (Dynamic programming revisited) was presented by Andonov et al.[8].

For problem of minimizing maximum processing time in unbounded knapsack problem (MMPTUKP) which 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 our best knowledge, this problem has been studied only by Srisuwannapa *et al.*[9, 10, 11, 12] in the literature. Therefore, the objective of this paper is to propose the new modified exact algorithm, CTCFMMPTUKP algorithm. It applied the coordinate transformation with CFMMPTUKP algorithm, close-form exact algorithm for solving this kind of problem. We also made computational experiments with 4 different types of generated data to validate our ideas and demonstrate the efficiency of new proposed algorithm. 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 new proposed algorithm was explained in section 4. Results and discussions and conclusions were stated in section 5 and 6 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 is applied at the pre-solve step to arrange data first before solving it (the transformed LP or ILP). Doing so might believe that it 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):

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

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

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

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:

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

where

$$\overline{c}_j = \begin{cases} -c_j, \text{ if } \; c_j < 0 \\ \;\; c_j, \text{ if } \; c_j \ge 0 \end{cases}$$

$$\overline{A}_j = \begin{cases} -A_j, \text{ if } \; c_j < 0 \\ \;\; A_j, \text{ if } \; c_j \ge 0 \end{cases}$$

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

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

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

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

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 j such as the profit, p_j, the weight, w_j, and the processing time, t_j 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.

$$\begin{aligned} &\min T \\ &\text{s.t.} \quad T \geq t_j x_j \\ &\sum_{j=1}^{n} p_j x_j \geq \mathrm{B} \\ &\sum_{j=1}^{n} w_j x_j \leq \mathrm{C} \\ &x_j \geq 0 \text{ and integer, } j = 1, 2, ..., n \end{aligned}$$

4. APPLICATION OF COORDINATE TRANSFORMATION WITH CLOSE-FORM EXACT ALGORITHM (CTCFMMPTUKP)

For the close-form exact algorithm (CFMMPTUKP) for solving MMPTUKP problem, in solving sub-problems, integer bounded knapsack problem(ILPUKP), coordinate transformation technique will be used first to transform ILPUKP to be the transformed ILPUKP (TILPUKP) and then it will be solved by branch-and-bound method. CTCFMMPTUKP's steps are as follows:

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

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

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:

<u>Close-form solution method</u>

choose x_j having the biggest value of ratio $\frac{p_j}{w_j}$, let $x_j = min\{C, \lfloor T^*/t_j \rfloor\}$, calculate remaining C, $C_{rem1} = C - min\{C, \lfloor T^*/t_j \rfloor\}$, 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 $x_j^* = \lfloor x_j \rfloor$, $\sum_{j=1}^{n} p_j x_j^* \doteq z^*$, 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, transform 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, ..., n, into the transformed bounded knapsack problem as follows:

$$\begin{aligned} &\min && \bar{c}^T v = z \\ &\text{subject to} && \bar{A} v = \bar{b} \qquad\qquad \text{(TILPUKP)} \\ & && 0 \le v \le \bar{u} \text{ and integer} \end{aligned}$$

and then solve it by the branch-and-bound method. After that find the value of variables by letting

$x_j^* = l_j + v_j$, if $c_j \ge 0$ or $x_j^* = l_j - v_j$, if $c_j < 0$, $z^* = \sum_{j=1}^{n} c_j x_j^*$ and then 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, $0 \le x_j \le \lfloor T^*/t_j \rfloor$, $j=1,2,\ldots,n$, by using close-form solution method as in *Step 3*, find value of variables by letting $x_j^* = \lfloor x_j \rfloor$, $\sum_{j=1}^{n} p_j x_j^* = z^*$, 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 6*. If yes, transform the knapsack problem form *Step* 6 with the bounded variable constraints $0 \le x_j \le \lfloor T^*/t_j \rfloor$ and integer, $j=1,2,\ldots,n$ into the transformed bounded knapsack problem, TILPUKP, as *Step 4* and solve it by the branch-and-bound method. After that find the value of variables by letting $x_j^* = l_j + v_j$, if $c_j \ge 0$ or $x_j^* = l_j - v_j$, if $c_j < 0$, $z^* = \sum_{j=1}^{n} c_j x_j^*$, and 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*.

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

This following lemma 1 verifies the correctness at step 4 of CTCFMMPTUKP algorithm.

Lemma 1: If objective function value obtained from rounding off x_j, Z_{roff} , 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 off x_j, be Z_{roff}, and the objective function value obtained from branch-and-bound method, Z_{bb}. If $B \le Z_{roff}$, since $Z_{roff} \le Z_{bb}$ by definition for maximum problem, then $B \le Z_{bb}$. □

The following theorem 1 is for verifying the correctness of the algorithm.

Theorem 1: 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 to the impossible case that $Tu > Tl$. By contradiction, T^* is optimal as stated. □

5. RESULTS AND DISCUSSION

We have investigated how all algorithms behaves with 4 different types of data generated. All algorithms were coded in C language and solved with CPLEX 6.5 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 median of solution time in second from 10 generated instances were presented in table.

In this paper, 4 different types of data were generated both dependently and independently to validate the efficiency of 6 algorithms especially the new modified algorithm. The tested data were as follows: uncorrelated, weakly correlated, strongly correlated, and subset sum data. However, we believe that level of parameters may affect speed of the solution time of algorithms, so the 2^k factorial design was used to test this hypotheses.

Since statistical test of each data's normal distribution indicated that all of them were not normally distributed, so non-parametric statistics such as Mann-Whitney and Kruskal-Wallis tests were used to test for significances.

Table 1 Median of 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] A	B	C	D	MMPTUK	CTMM-PTUK	W sig.	LPMM-PTUK	CTLPMM-PTUK	W sig.	CFMM-PTUK	CTCFMM-PTUK	W sig.	H-test(sig.)
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]

From test of significance in the above table, it can be concluded that, for all cases, CFMMPTUK and CTCFMMPTUK perform well equally and better, faster than the other algorithms. However, LPMMPTUKP and CTLPMMPTUKP which perform equally still perform better than MMPTUKP and CTMMPTUKP.

Table 2 Median of 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] 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)	-	-	-	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= $\sum_{I=j}^{n} w_j$, 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]

From test of significance in the above table, it can be concluded that, for all cases, CFMMPTUK and CTCFMMPTUK perform well equally and better, faster than the other algorithms. However, LPMMPTUKP and CTLPMMPTUKP which perform equally still perform better than MMPTUKP and CTMMPTUKP.

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] numbet 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]

From test of significance in the above table, it can be concluded that, for all cases, CFMMPTUK and CTCFMMPTUK perform well equally and better, faster than the other algorithms. However, LPMMPTUKP and CTLPMMPTUKP which perform equally still perform better than MMPTUKP and CTMMPTUKP.

Table 4 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	Run	Factor[a]			MMPTUK	CTMM-	W sig.	LPMM-	CTLPMM-	W	CFMM-	CTCFMM-	W	H
number	label	A	B	C		PTUK		PTUK	PTUK	sig.	PTUK	PTUK	sig.	test(sig.)
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*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 from [1,1000], high level(+): $c_j =$ w_j from [1,10000]

From test of significance in the above table, it can be concluded that, for most cases, CTCFMMPTUK performs better than CFMMPTUK and, for all cases, it, of course, performs better and faster than the other four remaining algorithms. For most cases, CTLPMMPTUKP performs better than LPMMPTUKP and CTMMPTUKP performs better than MMPTUKP. However, LPMMPTUKP and CTLPMMPTUKP which perform equally still perform better than MMPTUKP and CTMMPTUKP.

6. CONCLUSION

In this paper, we proposed the new modified exact algorithm(CTCFMMPTUKP), which is application of coordinate transformation with close-form exact algorithm(CFMMPTUKP), 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 uncorrelated, weakly correlated, strongly correlated data, CFMMPTUKP and CTCFMMPTUKP perform well equally and, however, they both perform better and faster than the 4 others as well. But for only subset sum data, application of coordinate transformation with 3 exact algorithms gives faster solution time than not application of coordinate transformation with. Finally, for all 4 different types of data, CTCFMMPTUKP performs faster than the 5 other algorithms. For the further research, we will test efficiency of algorithms with other type of data such as the inverse strongly correlated, almost strongly correlated, and uncorrelated data with similar weights.

REFERENCES

[1] P.toth (2000), Optimization engineering techniques for the exact solution of NP-hard combination optimization problem, *European Journal of Operation Research*, vol 125, pp.222-238.

[2] S. Martello, D. Psinger, P. Toth (1999), Dynamic programming and strong bounds for the 0-1 knapsack problem, *Management Science*, vol 45, pp 414-424.

[3] A. Freville, G. Plateau (1986), Heuristics and reduction methods for multiple constraints Linear Programming, *European Journal of Operations Research*, vol 24, pp.206-215.

[4] D. Psinger (1999), An exact algorithm for large multiple knapsack problem, *European Journal of Operations Research*, vol 114, pp.528-541.

[5] S. Martello, P. Toth (1981), A branch-and- bound algorithm for the zeor-one knapsack problem, *Discrete Applied Mathematics*, vol 3, pp.275-288.

[6] D. Psinger (1995), A minimal algorithm for the multiple-choice Knapsack Problem, *European Journal of Operations Research,* vol 83, pp.394-410

[7] D. Psinger (2001), Budgeting with bound multiple-choice constraints, *European Journal of Operations Research*, vol 129, pp.471-480.

[8] R. Andonow, V. Poirricz, S. Rajopadhye (2000), Unbounded knapsack problem:Dynamic programming revisited, *European Journal of Operations Research*, vol 123, pp.394-407.

[9] Srisuwannapa, C., Charnsettikul, P. (2001), A 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*, pp.83-89.

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

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

[12] Srisuwannapa, C., Charnsettikul, P. (2004), Close-form exact algorithm for Minimizing Maximum Processing Time in the Unbounded knapsack Problem. *KMITL SCIENCE JOURNAL*, KMITL, Thailand, Vol 4, 2004, pp. 310-321.