An Integer Programming-based Local Search for Large-Scale Multidimensional Knapsack Problems

Junha Hwang
Department of Computer Engineering
Kumoh National Institute of Technology
Gumi, Korea

Sungjae Park
Department of Computer Engineering
Kumoh National Institute of Technology
Gumi, Korea

In Yeup Kong
School of Electronic Engineering
Kumoh National Institute of Technology
Gumi, Korea

***Abstract*—Integer programming-based local search (IPbLS) is a metaheuristic recently proposed for solving linear combinatorial optimization problems. IPbLS is basically the same as the first-choice hill-climbing except for using integer programming for neighbor generation. Meanwhile, the multidimensional knapsack problem (MKP) is one of the most well-known linear combinatorial optimization problems and has received wide attention. Integer programming (IP) is very effective for small-scale or mid-scale MKP but suffers from large memory requirement for large-scale MKP. In this paper, we present an IPbLS for solving large-scale MKP. First, an initial solution is generated by IP, and then neighbor solutions are repeatedly obtained by IP using problem reduction. We used the largest 30 problem instances available on the OR-Library as experimental data. The proposed method could find better solutions than the best-known solutions for 6 problem instances. Furthermore, we confirmed that our average result of the best solutions outperforms the result of the best-known method.**

Keywords-Multidimensional Knapsack Problem; Integer Programming; Integer Programming-based Local Search

I. INTRODUCTION

Integer Programming-based Local Search (IPbLS) is a recent method proposed for linear combinatorial optimization problems [1, 2]. It is basically like first choice hill climbing [3] and repeatedly uses integer programming for neighbor generation. In the previous researches [1] and [2], IPbLS was respectively applied to the *N*-Queens maximization problem and the maximal covering problem which are linear combinatorial optimization problems, and also the effectiveness of IPbLS was verified by comparing with other search techniques like pure integer programming, tabu search, simulated annealing, and so on. However, since the search algorithms and the data for experiment were made by on their own, there were not enough evidences about the performance of IPbLS. Thus, in this paper, we are going to prove the superiority of IPbLS once again by applying it to the well-known multidimensional knapsack problem (MKP) and by using public MKP data on the OR-Library [4]. Since many previous studies have used the data for the MKP, this approach allows us to more easily verify the effect of IPbLS.

The MKP is a well-known NP-hard combinatorial optimization, which can be linearly expressed as following equations [5]. A set of n items with profits $p_j > 0$ and m resources with capacities $c_i > 0$ are given. Each item j consumes an amount $w_{ij} \geq 0$ from each resource i. The 0-1 decision variables x_j indicate which items are selected. According to (1), the goal is to choose a subset of items with maximum total profit. Selected items must, however, not exceed resource capacities. This is expressed by the knapsack constraints (2).

(MKP) maximize $$z = \sum_{j=1}^{n} p_j x_j \tag{1}$$

subject to $$\sum_{j=1}^{n} w_{ij} x_j \le c_i, \quad i = 1,\ldots,m, \tag{2}$$

$$x_j \in \{0,1\}, \quad j = 1,\ldots,n. \tag{3}$$

A set of large MKP instances available on the OR-Library have been mainly used for testing the suggested algorithms. It was made by Chu and Beasly [6] and consists of 270 correlated instances generated using the procedure proposed in [7]. The instances were generated by varying combinations of constraints with *m* = 5, 10, 30 and variables with *n* = 100, 250, 500. A set of 30 instances was generated for each *n*-*m* combination. Most of the best-known solutions for the instances were obtained by combined linear programming and tabu search with a variable fixing heuristic [8]. Since then, some advanced results have been reported for relatively small data. We have not, however, seen any better results for the largest instances with *m* = 30 and *n* = 500. So we will apply IPbLS to the largest instances with *m* = 30 and *n* = 500, and compare the results with any other existing algorithms including the best-known research [8]. Experimental results show that the proposed method based on IPbLS outperforms the state-of-the art results.

The structure of the paper is as follows. Section II describes the related works about the MKP and IPbLS. Section III suggests IPbLS for the MKP and Section IV presents experimental results. Finally, in section V we draw the conclusion and discuss future works.

II. RELATED WORK

A. The Multidimensional Knapsack Problem

The MKP has many applications such as capital budgeting, loading problem, resource allocation, and so on [9]. Although the MKP is expressed as a simple formulation, it has been paid attention for a long time due to its complexity and importance. Especially, the MKP has been often used as a test problem to evaluate new metaheuristics. Table I presents a summary of some important research articles and their main approaches.

TABLE I. A SUMMARY OF SOME IMPORTANT RESEARCH ARTICLES

Reference	Year	Author	Approach	Note
[6]	1998	Chu	genetic algorithm	making the MKP data on the OR-Library
[10]	2001	Vasquez	linear programming, tabu search	the first remarkable results
[11]	2004	Alaya	ant colony optimization	
[8]	2005	Vasquez	linear programming, tabu search, variable fixing	improved results of [10]
[12]	2005	Puchinger	variable neighborhood search	
[13]	2005	James	dynamic programming, enumeration	exact method
[14]	2006	Puchinger	core concept, memetic algorithm, variable neighborhood search	
[15]	2007	Akcay	greedy-like heuristic	
[16]	2007	Gallardo	branch and bound, memetic algorithm	
[17]	2008	Nhicolaievna	bee colony optimization	
[18]	2008	Vimont	constraint propagation, implicit enumeration, variable fixing	exact method
[19]	2009	Wilbaut	linear programming, variable fixing	
[20]	2009	Boussier	branch and bound	exact method
[21]	2009	Wilbaut	linear programming, integer programming	better results than [8] in a few instances
[22]	2010	Puchinger	core concept	
[23]	2010	Al-Shihabi	nested partition, ant colony optimization, linear programming	
[24]	2010	Boussier	brach and bound, depth first search	exact method
[25]	2010	Hill	lagrangian relaxation	
[26]	2010	Ke	ant colony optimization	
[27]	2011	Croce	core concept, linear programming	
[28]	2011	Hanafi	linear programming	

Although there are some papers in which the data on the OR-Library are not used, we just included research articles which use the data on the OR-Library in the table I. For example, there was a research using tabu search before the data was made [29], and in recent years, the previous research [30] introduced a computational model simulating the dynamic process of human immune response and [31] introduced a hybridized scheme using

Dantzig algorithm and the genetic algorithm to solve the MKP. However, since it is hard to directly compare our result with their results because they did not use the same data, we do not consider the researches any more.

As you can see from the table I, many methods like linear programming, tabu search, ant colony optimization, branch and bound, and so on, have been applied to find better solutions since the MKP data on the OR-Library had been made to verify the performance of genetic algorithm [6]. The first significant result was made by the hybrid method of linear programming and tabu search [10]. This result is far better than the result of [6]. And then, in the research [8], the improved result was obtained by adding variable fixing heuristic to the hybrid method of [10] by the same researcher. In fact, the result of the paper [8] has been recognized as the best result until now. In particular, we have not seen any similar results as well as any better results on the largest instances with m = 30, n = 500. Only the research by Wilbaut has shown the better results on 3 instances out of 30 instances with m = 30, n = 500 [21]. However, in [21], the results on 18 instances are worse and the average result is also far worse than the result of [8].

The contribution of some papers can be recognized in the following ways. First, some papers like [19, 23, 27] show a little better results or similar results in a relatively small-scale data instances. However, the results on average are not better than the result of [8], and in addition, the results are far worse than the result of [8] on the largest instances with m = 30, n = 500. Another contribution is that the previous researches [14, 22, 28] could find good solutions in a relatively short time. In the research [8], total execution time per data instance with m = 30, n = 500 was 100 hours, and on average it took about 33 hours until the best-known solutions were found. On the other hand, it took a few hours to get their final solutions in the papers [14, 22, 28]. However, their final results are significantly worse than the result of [8]. We guess that to emphasize the short execution time is in itself an evidence that they cannot reach the result of [8], so they may not find the best-known solutions after 100 hours. Several other papers are by and large worse than [8], or it is hard to compare with [8] since they only deal with some parts of the data.

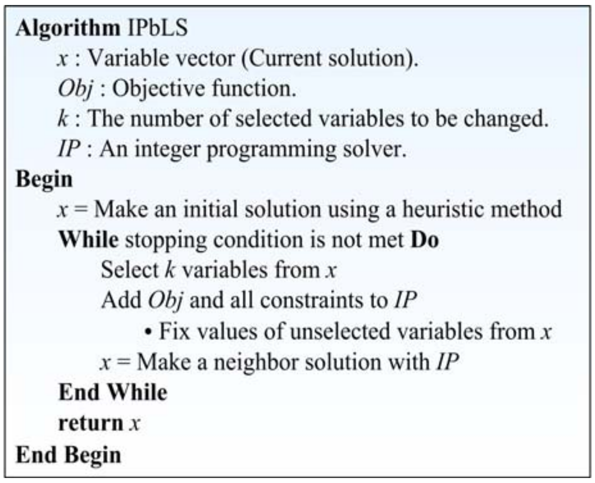
Meanwhile, the previous studies [13, 18, 20, 24] are classified as the exact method which guarantees finding an optimal solution. [13] found the optimal solutions on a number of m = 5, n = 500 instances and [18] obtained several new optimal solutions never published before. The optimal solutions of the instances with m = 10, n = 500 were found in [20] and the optimal solutions of some of m = 30, n = 250 instances were found in [24]. However, the optimal solutions have not been still found for all the largest instances with m = 30, n = 500.

In conclusion, the goal of this paper is to find better solutions than the best-known solutions described in [8] for the largest instances with m = 30, n = 500 using IPbLS. We will compare our results with the results of the previous researches [6, 10, 21] as well as [8] which are the most important studies for the MKP.

B. Integer Programming-based Local Search

IPbLS was first proposed to solve a nurse scheduling problem [32], and it was applied to solve a classic network design problem [33]. IPbLS was also developed to solve N-Queens maximization problem which is a kind of linear constraint satisfaction optimization problem [1], and it was recently used to solve the maximal covering problem [2].

Fig. 1 shows the structure of IPbLS [2]. IPbLS is basically based on first choice hill climbing. Most of the combinatorial optimization problems can be expressed as n variables, and a neighbor solution can be made by changing the values of k variables out of n variables. General first choice hill climbing makes neighbor solutions by randomly changing the values until one is generated that is better than the current solution, and then moves to the neighbor solution [3].



```
Algorithm IPbLS
    x : Variable vector (Current solution).
    Obj : Objective function.
    k : The number of selected variables to be changed.
    IP : An integer programming solver.
Begin
    x = Make an initial solution using a heuristic method
    While stopping condition is not met Do
        Select k variables from x
        Add Obj and all constraints to IP
            • Fix values of unselected variables from x
        x = Make a neighbor solution with IP
    End While
    return x
End Begin
```

Figure 1. General Integer Programming-based Local Search

However, IPbLS selects *k* variables and then moves to a locally optimal neighbor solution since it considers all the neighborhood solutions being able to be made by changing the *k* variables. Integer programming used as a neighbor generation tool makes this possible, which is an exact method for solving linear combinatorial optimization problems based on branch and bound [34]. In addition, *k* of IPbLS can be set to a far larger value unlike general local search where the value of *k* is set relatively small. This is also due to integer programming which can find an optimal solution more efficiently when *k* is relatively large.

III. Integer Programming-based Local Search for the Multidimensional Knapsack Problem

IPbLS for the MKP is very simple. First of all, an initial solution should be made. In the previous studies [1] and [2] , greedy heuristic methods were used to generate an initial solution. A greedy heuristic can be also applied to generate an initial solution for the MKP, however, we use integer programming (IP) for this work. IP may be effective or not according to the given problem but at least IP seems to be very effective for the MCP. Nevertheless, IP exhausts the given memory in a few hours since IP especially needs much memory for large-scale problems. Therefore, execution time limit for generating an initial solution (t_i) is given and the best solution in the time is set to the initial solution.

Next, the problem is reduced by selecting items participating in the next IP execution. When we select items for the next IP, we basically select all the items included in the knapsack of the current solution and, in addition, randomly select some of the items that are not included in the knapsack. The total number of selected items is represented by *k*. The variables for the unselected items are fixed to 0, but this is actually implemented by excluding the variables for the items. Now, IP solves the reduced MKP problem that consists of *k* variables, but at this time IP requires excessive time if the value of *k* is too large. On the other hand, it is hard to improve the current solution if the value of *k* is too small. Accordingly, we set the value of *k* to somewhat large value, and limit the execution time of IP (t_n). After the next solution is obtained, IPbLS for the MKP repeats selecting *k* items from the total *n* items and executing IP until the total execution time reaches the total time limit.

While out method selects all the items included in the knapsack of the current solution, there may be another method in which we remove some of the items included in the current knapsack. However, according to the experimental result, it seems not to be effective for the MKP. The removing allows moving to a worse solution like simulated annealing [3], but it seems useless for IPbLS for the MKP. Therefore, there are 3 parameters for our method: execution time for generating an initial solution - t_i, The number of items joining IP to generate a neighbor solution - *k*, execution time of IP for generating a neighbor solution - t_n. All these parameter values are empirically determined.

IV. Experimental Result

A. *Experimental Environment*

As mentioned previously, IPbLS for the MCP proposed in this paper was tested on the largest 30 data instances with m = 30, n = 500 available on the OR-Library. The 30 instances were generated with different tightness ratios (α = 0.25, 0.5, 0.75). The values of α are 0.25 for the first ten instances, 0.5 for the next ten instances and 0.75 for the remaining ten instances.

All the experiments were carried on a PC with Intel Core2 Duo with 3GHz and 2GB memory. Execution time of one run was limited to 56 hours, and five runs were carried out per each instance. The value of the parameter t_n is 300 seconds that is identical in all data instances, but the values of the parameters t_i and k are different depending on the instance. The value of t_i is various from 300 to 1150 seconds which is the time taken to get the best solution depending on the instance. Thus, in fact, IPbLS starts from the best solution found by IP. The value of *k* is different according to the tightness ratio α. The values are respectively 350, 400, 450 when α = 0.25, 0.5, 0.75. The values of the parameters *k* and t_n were determined based on just several experiments on some instances. Therefore, we think that the performance of IPbLS can be improved by more carefully adjusting the values.

We used IBM ILOG CPLEX 12.1 as an integer programming development tool [35]. IBM ILOG CPLEX is the present most widely used linear and integer programming library in the world for commercial and academic purpose.

B. *Experimental Results*

Table II presents the detailed results obtained by our approach on the 30 instances and the results are compared with five important other approaches. The meaning of the columns is as follows.

- GA: results obtained by genetic algorithm [6]
- LP+TS: results obtained by combined linear programming with tabu search [10]
- LP+TS+Fix: results obtained by combined linear programming with tabu search and variable fixing which is the best-known approach [8]

- IRH: results obtained by iterative relaxation-based heuristic [21]
- IP [CPLEX]: results obtained by integer programming using IBM ILOG CPLEX 12.1
- IPbLS: result obtained by our method
- z^*: best value found by each of the six methods
- t^*: time in seconds to find the best solution by each method
- z_{avg} of IRH: average value of 60 runs by IRH
- z_{avg} of IPbLS: average value of 5 runs by IPbLS

The best values are marked in bold shade on each problem instance. From the best values z^*, we can see that IRH found the better solutions than LP+TS+Fix in only three problems 2, 3, 15 and the improved values are 29, 16, 29 respectively. However, IRH is worse than LP+TS+Fix in other 19 problems and, in addition, the average values z_{avg} of IRH are far worse than LP+TS+Fix. On the other hand, we can see that IPbLS is better than LP+TS+Fix in six problems 2, 3, 4, 7, 10, 29 and the improved values are 29, 41, 28, 55, 4, 10 respectively which are far better results than the best-known solutions. IPbLS shows worse results than LP+TS+Fix in six problems 11, 13, 22, 23, 26, 27 and the values are 3, 25, 4, 17, 4, 8 respectively, but the differences are not relatively large. Also, average values z_{avg} of IPbLS are not so much worse than LP+TS+Fix and, in addition, IPbLS shows better average results in two problems 3 and 4. The execution times to find the best solutions are different in LP+TS+Fix and IPbLS. It seems that IPbLS somewhat takes shorter than LP+TS+Fix, but it is impossible to directly compare the two methods since the experimental environments are differ from each other. We just present the information about the execution time for reference.

TABLE II. EXPERIMENTAL RESULTS FOR THE 30 INSTANCES

Problem	GA [6]	LP + TS [9]	LP + TS + Fix [8]		IRH [20]		IP [CPLEX]	IPbLS		
	z^*	z^*	z^*	t^*	z^*	z_{avg}	z^*	z^*	z_{avg}	t^*
0	115,868	115,991	**116,056**	15,771	115,948	115,512	115,946	**116,056**	116,023	136,917
1	114,667	**114,810**	**114,810**	120,414	114,780	114,338	114,701	**114,810**	114,793	6,014
2	116,661	116,683	116,712	121,769	**116,741**	116,244	116,685	**116,741**	116,710	82,134
3	115,237	115,301	115,329	85,670	115,345	114,856	115,242	**115,370**	115,332	86,403
4	116,353	116,435	116,525	603	116,516	116,052	116,516	**116,553**	116,536	17,393
5	115,604	115,694	**115,741**	616	**115,741**	115,236	115,734	**115,741**	115,737	49,364
6	113,952	114,003	**114,181**	110,873	114,111	113,696	114,058	**114,181**	114,167	33,026
7	114,199	114,213	114,348	282,523	114,344	113,930	114,249	**114,403**	114,339	72,765
8	115,247	115,288	**115,419**	112,849	**115,419**	115,050	**115,419**	**115,419**	115,419	300
9	116,947	117,055	**117,116**	121,248	**117,116**	116,687	117,020	**117,116**	117,106	3,609
10	217,995	218,068	218,104	96,952	218,068	217,683	218,068	**218,108**	218,092	41,496
11	214,534	214,562	**214,648**	167,224	214,645	214,314	214,626	214,645	214,638	16,238
12	215,854	215,903	**215,978**	178,701	215,922	215,592	215,846	**215,978**	215,954	95,939
13	217,836	**217,910**	**217,910**	308,469	217,885	217,538	217,875	217,885	217,879	49,561
14	215,566	215,596	**215,689**	144,793	215,640	215,320	215,604	**215,689**	215,682	77,676
15	215,762	215,842	215,890	117,102	**215,919**	215,555	215,847	215,890	215,875	47,210
16	215,772	215,838	**215,907**	1,637	**215,907**	215,561	215,796	**215,907**	215,902	95,300
17	216,336	216,419	**216,542**	4,775	216,510	216,146	216,444	**216,542**	216,516	185,574
18	217,290	217,305	**217,340**	4,742	217,313	216,935	217,287	**217,340**	217,338	190,574
19	214,624	214,671	**214,739**	109,785	214,690	214,312	214,695	**214,739**	214,730	2,407
20	301,627	301,643	**301,675**	143,430	**301,675**	301,385	**301,675**	**301,675**	301,675	900
21	299,985	**300,055**	**300,055**	218,994	**300,055**	299,742	**300,055**	**300,055**	300,055	1,200
22	304,995	305,028	**305,087**	118,929	305,080	304,751	305,062	305,083	305,082	10,369
23	301,935	302,004	**302,032**	156,377	302,008	301,683	302,004	302,015	302,015	34,518
24	304,404	304,411	**304,462**	238,811	304,425	304,089	304,412	**304,462**	304,449	79,048
25	296,894	296,961	**297,012**	12,191	296,969	296,658	296,946	**297,012**	297,005	45,285
26	303,233	303,328	**303,364**	213,316	303,329	303,025	303,322	303,360	303,337	127,218
27	306,944	306,999	**307,007**	45,781	306,999	306,679	306,893	306,999	306,982	90,171
28	303,057	303,080	**303,199**	235,005	**303,199**	302,906	303,158	**303,199**	303,193	1,352
29	300,460	300,532	300,572	82,949	300,572	300,229	300,538	**300,582**	300,558	167,591

Table III shows a summary of the compared results with LP+TS+Fix according to the tightness ratio, with the number of improved values (#Improved), equal values (#Equal) and lower values (#Lower). The total number of improved values is the same as the total number of lower values. An interesting result is that the number of improved values is much larger than the number of lower values when α = 0.25 and the number of improved values is slightly smaller than the number of lower values when α = 0.5 and α = 0.75.

TABLE III. SUMMARY OF THE RESULTS COMPARED WITH LP+TS+FIX

Problem	**#Improved**	**#Equal**	**#Lower**
0 ~ 9 (α = 0.25)	4	6	0
10 ~ 19 (α = 0.5)	1	7	2
20 ~ 29 (α = 0.75)	1	5	4
Total	6	18	6

Table IV gives a summary of the results obtained by the three methods LP+TS+Fix, IRH and IPbLS according to the tightness ratio. Each line is the average value of the best values z^* over 10 instances. We also present the number of problems with the best value by each method at the last row. The total average value of IPbLS is 211,452 which outperforms not only IRH but also LP+TS+Fix. Especially, IPbLS shows outstanding result when α = 0.25. IPbLS is slightly worse than LP+TS+Fix when α = 0.5 and α = 0.75, but the differences are insignificant. The number of best solutions by IPbLS is 23 which is the same as the one by LP+TS+Fix.

TABLE IV. SUMMARY OF THE AVERAGE RESULTS

Problem	**LP + TS + Fix [8]**	**IRH [20]**		**IPbLS**	
	z^*	z^*	z_{avg}	z^*	z_{avg}
0 ~ 9 (α = 0.25)	115,624	115,606	115,160	**115,639**	115,616
10 ~ 19 (α = 0.5)	**216,275**	216,250	215,896	216,272	216,261
20 ~ 29 (α = 0.75)	**302,447**	302,431	302,115	302,444	302,435
Total Average	211,448	211,429	211,057	**211,452**	211,437
# of best solutions	**23**	9	-	**23**	-

From these results, we can confirm that the overall performance of IPbLS outperforms LP+TS+Fix known as the best method, and especially IPbLS is far better than LP+TS+Fix when α = 0.25. IPbLS is slightly worse than LP+TS+Fix when α = 0.5 and α = 0.75, but it is supposed that IPbLS can be improved by adjusting parameter values or by utilizing problem specific knowledge.

V. CONCLUSION AND FUTURE WORK

In this paper, we have presented a method applying IPbLS for large-scale MKP. We used the MKP data on the OR-Library, which is the most widely used for the MKP, to easily compare with other existing research results. Especially, we focused on the 30 largest instances with m = 30, n = 500 which is the most difficult data set out of the MKP data on the OR-Library. Experimental results showed that the proposed method could find the better solutions than the best-known solutions for 6 data instances, and also on average, outperformed the best-known method. We could confirm the effectiveness of IPbLS more objectively through this research, which is the first goal of this paper.

We could achieve remarkable results by only a simple application of IPbLS. Therefore, it is supposed that we can get much better solutions if we utilize the problem and/or data specific knowledge or finely adjust the parameter values. We will perform additional research on this issue with small or medium-scale MKP data as well as large-scale MKP data. Meanwhile, since IPbLS can be viewed as a kind of metaheuristic for linear combinatorial optimization problems, it can be easily applied to not only the MKP but also many other linear combinatorial optimization problems. Thus, we will continuously develop methods to utilize IPbLS for solving other optimization problems.

REFERENCES

[1] J. Hwang, “Integer programming-based local search technique for linear constraint satisfaction optimization Problem,” Journal of The Korea Society of Computer and Information, vol. 15, no. 9, pp. 47-55, 2010.

[2] J. Hwang, and S. Kim, “An integer programming-based local search for large-scale maximal covering problems,” International Journal on Computer Science and Engineering, vol. 3, no. 2, pp. 837-843, 2011.

[3] S. Russel, and P. Norvig, Artificial Intelligence A Modern Approach, 2nd ed., Prentice Hall, 2003.

[4] J. E. Beasley, “OR-Library: distributing test problems by electronic mail,” The Journal of the Operational Research Society, vol. 41, no. 11, pp. 1069-1072, 1990.

[5] J. Puchinger, G. R. Raidl, and U. Pferschy, “The multidimensional knapsack problem: structure and algorithms,” INFORMS Journal on Computing, vol. 22, no. 2, pp. 250-265, 2010.

[6] P. C. Chu, and J. E. Beasley, “A genetic algorithm for the multidimensional knapsack problem,” Journal of Heuristics, vol. 4, no. 1, pp. 63-86, 1998.

[7] A. Freville, and G. Plateau, “An efficient preprocessing procedure for the multidimensional 0-1 knapsack problem,” Discrete Applied Mathematics, vol. 49, no. 1-3, 1994.

[8] M. Vasquez, and Y. Vimont, “Improved results on the 0-1 multidimensional knapsack problem,” European Journal of Operational Research, vol. 165, no. 1, pp. 70–81, 2005.

[9] C. Wilbaut, S. Hanafi, and S. Salhi, “A survey of effective heuristics and their application to a variety of knapsack problems,” IMA Journal of Management Mathematics, vol. 19, no. 3, pp. 227-244, 2008.

[10] M. Vasquez, and J. K. Hao, “A hybrid approach for the 0–1 multidimensional knapsack problem,” Proceedings of the International Joint Conference on Artificial Intelligence 2001, pp. 328–333, 2001.

[11] I. Alaya, C. Solnon, and K. Ghdira, “Ant algorithm for the multidimensional knapsack problem,” Proceedings of the International Conference on Bioinspired Optimization Methods and their Applications (BIOMA 2004), pp. 63-72, 2004.

[12] J. Puchinger, and G. R. Raidl, “Relaxation guided variable neighborhood search,” Proceedings of the 18th Mini EURO Conference on Variable Neighborhood Search, 2005.

[13] R. J. W. James, and Y. Nakagawa, “Enumeration methods for repeatedly solving multidimensional knapsack sub-problems,” IEICE Transactions on Information and Systems, vol. E88-D, no. 10, pp. 2329-2340, 2005.

[14] J. Puchinger, G. R. Raidl, and U. Pferschy, “The core concept for the multidimensional knapsack problem,” Evolutionary Computation in Combinatorial Optimization - EvoCOP 2006, LNCS, vol. 3906, pp. 195-208, 2006.

[15] Y. Akcay, H. Li, and S. H. Xu, “Greedy algorithm for the general multidimensional knapsack problem,” Annals of Operations Research, vol. 150, pp. 17-29, 2007.

[16] J. E. Gallardo, C. Cotta, and A. J. Fernández, “On the hybridization of memetic algorithms with branch-and-bound techniques,” IEEE Transactions on Systems, Man and Cybernetics, Part B, vol. 37, no. 1, pp. 77-83, 2007.

[17] P. N. Nhicolaievna, and L. V. Thanh, “Bee colony algorithm for the multidimensional knapsack problem,” Proceeding of the International Multi-Conference of Engineers and Computer Scientists, vol. 1, pp. 1-5, 2008.

[18] Y. Vimont, S. Boussier, and M. Vasquez, “Reduced costs propagation in an efficient implicit enumeration for the 0-1 multidimensional knapsack problem,” Journal of Combinatorial Optimization, vol. 15, no. 2, pp. 165-178, 2008.

[19] C. Wilbaut, S. Salhi, and S. Hanafi, “An iterative variable-based fixation heuristic for the 0-1 multidimensional knapsack problem,” European Journal of Operational Research, vol. 199, no. 2, pp. 339-348, 2009.

[20] S. Boussier, M. Vasquez, Y. Vimont, S. Hanafi, and P. Michelon, “Solving the 0-1 multidimensional knapsack problem with resolution search,” VI ALIO/EURO Workshop on Applied Combinatorial Optimization, 2009.

[21] C. Wilbaut, and S. Hanafi, “New convergent heuristics for 0-1 mixed integer programming,” European Journal of Operational Research, vol. 195, no. 1, pp. 62-74, 2009.

[22] J. Puchinger, G. R. Raidl, and U. Pferschy, “The multidimensional knapsack problem: structure and algorithms,” INFORMS Journal on Computing, vol. 22, no. 2, pp. 250-265, 2010.

[23] S. Al-Shihabi, and S. Ólafsson, “A hybrid of nested partition, binary ant system, and linear programming for the multidimensional knapsack problem,” Computers & Operations Research, vol. 37, no. 2, pp. 247-255, 2010.

[24] S. Boussier, M. Vasquez, Y. Vimont, S. Hanafi, and P. Michelon, “A multi-level search strategy for the 0–1 multidimensional knapsack problem,” Discrete Applied Mathematics, vol. 158, no. 2, pp. 97-109, 2010.

[25] R. R. Hill, Y. K. Cho, and J. T. Moore, “Problem reduction heuristic for the 0-1 multidimensional knapsack problem,” Computers & Operations Research, Article in Press, 2010.

[26] L. Ke, Z. Feng, Z. Ren, and X. Wei, “An ant colony optimization approach for the multidimensional knapsack problem,” Journal of Heuristics, vol. 16, no. 1, pp. 65-83, 2010.

[27] F. D. Croce, and A. Grosso, “Improved core problem based heuristics for the 0/1 multi-dimensional knapsack problem,” Computers & Operations Research, Article in Press, 2011.

[28] S. Hanafi, and C. Wilbaut, “Improved convergent heuristics for the 0-1 multidimensional knapsack problem,” Annals of Operations Research, vol. 183, no. 1, pp. 125-142, 2011.

[29] S. Hanafi, A. Freville, “An efficient tabu search approach for the 0-l multidimensional knapsack problem,” European Journal of Operational Research, vol. 106, pp. 659-675, 1998.

[30] M. Gong, L. Jiao, W. Ma, and S. Gou, “Solving multidiemensional knapsack problems by an immune-inspired algorithm,” Proceedings of the 2007 IEEE Congress on Evolutionary Computation, pp. 3385-3391, 2007.

[31] F. Djannaty, and S. Doostdar, “A hybrid genetic algorithm for the multidimensional knapsack problem,” International Journal of Contemporary Mathematical Sciences, vol. 3, no. 9, pp. 443-456, 2008.

[32] S. Hasegawa, and Y. Kosugi, “Solving nurse scheduling problem by integer-programming-based local search,” 2006 IEEE International Conference on Systems, Man and Cybernetics, vol. 2, pp. 1474-1480, 2006.

[33] M. Hewitt, G. L. Nemhauser, and M. W. Savelsbergh, “Combining exact and heuristic approaches for the capacitated fixed charge network flow problem,” INFORMS Journal on Computing, vol. 22, no. 2, pp. 314-325, 2009.

[34] L. A. Wolsey, Integer Programming, Wiley, 1998.

[35] IBM ILOG, CPLEX User’s Manual and Reference Manual, Version 12.1, 2009.

AUTHORS PROFILE

Junha Hwang received his B.S., M.S. and Ph.D. degree in Computer Engineering from Pusan National University, Korea in 1995, 1997 and 2002 respectively. He is an Associate Professor in Dept. of Computer Engineering, Kumoh National Institute of Technology, Korea since 2002. His main research interests are combinatorial optimization, machine learning and artificial intelligence.

Sungjae Park received his B.S. degree in Computer Engineering from Kumoh National Institute of Technology, Korea in 2010. He is pursuing his master’s degree in Dept. of Computer Engineering, Kumoh National Institute of Technology. His main research interests are combinatorial optimization, machine learning and artificial intelligence.

In Yeup Kong received her B.S., M.S. and Ph.D. degree in Computer Engineering from Pusan National University, Korea in 2000, 2002 and 2007 respectively. She is an Full-time Instructor in School of Electronic Engineering, Kumoh National Institute of Technology, Korea since 2008. Her main research interests are context awareness based on ontology and network optimization.