A self-adaptive harmony search combined with a stochastic local search for the 0-1 multidimensional knapsack problem

Abdellah Rezoug*

Department of Computer Science,
Faculty of Sciences,
University Mhamed Bougara of Boumerdes,
Boumerdes, Algeria
Email: abdellah.rezoug@gmail.com
*Corresponding author

Dalila Boughaci

Department of Computer Science,
University of Sciences and Technology Houari Boumedienne (USTHB),
Algiers, Algeria
Email: dalila_info@yahoo.fr

Abstract: This paper presents a new hybrid self-adaptive harmony search combined with a stochastic local search algorithm (SAHS-SLS) to solve the 0-1 multidimensional knapsack problem (MKP). The proposed SAHS-SLS uses SAHS to create harmonies that will be improved with SLS. We propose a dynamic adjustment of the walk probability (wp) in SLS and a technique to compute the bandwidth (bw) and the pitch adjusting rate (PAR) in SAHS. The overall method SAHS-SLS is implemented and evaluated on benchmarks in order to measure its performance in solving the MKP. It is compared to other approaches to show its effectiveness. The numerical results are encouraging and demonstrate the benefit of the proposed approach.

Keywords: multidimensional knapsack problem; MKP; self-adaptive harmony search; SAHS; harmony search; multi-unit combinatorial auctions; winner determination problem; stochastic local search; hybrid heuristic; bio-inspired computation; combinatorial optimisation; evolutionary computation.

Reference to this paper should be made as follows: Rezoug, A. and Boughaci, D. (2016) 'A self-adaptive harmony search combined with a stochastic local search for the 0-1 multidimensional knapsack problem', *Int. J. Bio-Inspired Computation*, Vol. 8, No. 4, pp.234–239.

Biographical notes: Abdellah Rezoug is a PhD student at the University of USTHB. He received his Magister degree in Computer Science from the University of Boumerdes. He is a Lecturer at the Department of Computer Science, University of Boumerdes. His research interests include issues related to the heuristic approaches, optimisation problems, the winner determination problems and the multidimensional knapsack problem.

Dalila Boughaci is a Full Professor of Computer Science at the University of USTHB. She received her PhD from the University of Provence (Marseille-France) in 2008 and her Habilitation Post-doctoral Diploma from USTHB University in 2009. Her current research interests are in the areas of evolutionary computation, artificial intelligence and meta-heuristics. She has published several papers on these research topics in journals and conferences and directed several PhD, MS and BS students' projects. Hei is the Head of the research team: Optimisation, Reasoning and Application of the LRIA Laboratory.

1 Introduction

The multidimensional knapsack problem (MKP) is a NP-hard combinatorial optimisation problem. It is composed of N items and a knapsack with m different capacities b_i with $i \in M = \{1 \cdots m\}$. Each item j with $j \in N = \{1 \cdots n\}$ has a profit c_j and can occupy a_{ij} a portion of capacity i of the knapsack. The goal is to pack the items in the knapsack so as to maximise the profits

of items without exceeding the capacities of the knapsack. The MKP is modelled as the following integer program:

$$\text{Maximise} \quad \sum_{j=1}^{n} c_j x_j \qquad j \in \{1 \dots n\} \tag{1}$$

$$\text{Subject to:} \quad \sum_{j=1}^{n} a_{ij} x_j \leq b_i \qquad i \in \{1 \dots m\} \tag{2}$$
$$x_j \in \{0, 1\}$$

where X is a vector of size n that is the solution vector of the problem, that represents the selected items to be packed in the knapsack. Decision variables x_j are binary where $x_j = 1$ means that the item j is packed in the knapsack, and $x_j = 0$ otherwise.

Existing approaches for solving the MKP can be classified into exact and approximate methods. Among the exact approaches proposed for the MKP, we cite: branch and bound (Vimont et al., 2008; Fukunaga, 2011), relaxation (Freville and Plateau, 1997) and dynamic programming (Volgenant and Zoon, 1990). These approaches have the advantage of efficiency to solve MKP problems of small size and provide exact results. However, the execution time increases exponentially with the size of problem. The approximate approaches can provide good results within reasonable time. Several approaches have been proposed to the MKP such as: tabu search (Aboudi and Jornsten, 1994), simulated annealing (Cho and Kim, 1997), genetic algorithm (Khuri et al., 1994; Chu and Beasley, 1998; Yoon et al., 2005; Lin, 2008), ant-colony (Shi, 2006), harmony search (HS) (Zou et al., 2011), evolutionary algorithm (Liu and Liu, 2009), particle swarm optimisation (Li and Li, 2009) as well as others (Egeblad and Pisinger, 2009; Boussier et al., 2010; Hill et al., 2012).

That is always advantageous to combine an evolutionary-based method with a local search to ensure a balance between the global exploration and the local exploitation of the search space. Motivated by this idea, we propose a hybrid self-adaptive harmony search (SAHS) combined with a stochastic local search (SLS) to solve MKP. SAHS is used to ensure exploration while SLS performs exploitation. First, we improved the SAHS by adding a tuning strategy for the pitch adjusting rate (PAR) and the Bandwidth (bw). Then, we apply the local search SLS on every generated solution with a specified Probability (ESP) strategy. The proposed self-adaptive harmony search-stochastic local search (SAHS-SLS) algorithm is evaluated on the well-known MKP benchmarks proposed by Chu and Beasley (1998).

The rest of the paper is organised as follows. Section 2 gives a background on the solution representation, the encoding technique, the HS, SLS and the SAHS approaches. Section 3 details the proposed new approach SAHS-SLS. Section 4 presents the experimentation and gives some numerical results. In Section 5 we discuss the obtained results. Finally, Section 6 concludes the paper.

2 A background

2.1 The solution representation

Let us assume that A is a feasible solution to MKP problem. The solution A can be represented by a binary vector X of size n (with n is the number of items). Each of those components x_i indicates if the item i is selected to be packed in the knapsack or not.

2.2 The RK encoding

The random key (RK) (Bean, 1994) is used to create feasible solutions. n values ($n \in [0, 1]$) are generated randomly such that each value corresponds to an item. Then, a solution is built by adding items one after one, according to the order. An item is ignored if its addition leads that at least one constraint is broken. This operation continues until the last item. Finally, the fitness of the solution is computed.

2.3 The SLS method

The SLS (Boughaci et al., 2010) is an iterative approach. First an initial solution is generated according to the RK. Second, a neighbour solution is created by adding a selected item to the current solution. The item to be added is whether chosen randomly with a fixed probability $wp > 0$, or the chosen item is the best one in terms of profit. Third, if the created neighbour solution is not a feasible solution, then it will be repaired. The process is repeated until a certain maximum number of iterations $maxiter$ fixed empirically. The SLS algorithm is sketched in Algorithm 1.

Algorithm 1 The SLS method

Require: a feasible solution X, $maxiter$, wp
Ensure: a better feasible solution X^*
  for $i = 1$ to $maxiter$ **do**
    generate r, r $\in [0, 1]$;
    if r $< wp$ **then**
      $x_i = x_{rand}$ (**Step 1)*
    else
      $x_i = x_{max}$ (**Step 2)*
    end if
    $X = X \cup \{x_i\}$; Update (f(X) and somConflit);
    while there is conflict **do**
      $x_i = x_{min}$; $X = X$ - $\{x_m\}$;
      Update (f(X) and somConflit);
    end while
    if $(F(X) > F(X^*))$ **then**
      $X^* = X$;
    end if
  end for

  return the best solution X^*.

2.4 The HS algorithm

The HS (Geem et al., 2001) is a new population-based heuristic. The population or the harmony memory (HM) is composed of harmony memory size (HMS) harmonies coded by vectors representing feasible solutions. Each iteration, a new harmony is improvised by adding an item either randomly or from the HM according to the harmony memory considering rate (HMCR). Then, the item may be adjusted by a Bandwidth (bw) according to the PAR. Finally, the improvised harmony replaces the worst harmony in HM, if its fitness is better.

2.5 The SGHS algorithm

Self-adaptive global best harmony search (SGHS) algorithm ((Pan et al., 2010) proposed a self-adapted tuning of the parameters HMCR, PAR and bw by a learning mechanism, a modified PAR and dynamic generation of bw. SGHS proposed also, in the memory consideration phase, an improved method to avoid getting trapped in local optima. SGHS can be summarised by the pseudo code in Algorithm 2.

Algorithm 2 The SGHS method

Require: $HMS, LP, NI_{max}, n, bw_{max}, bw_{min}, HMCRm$ and$PARm$
Ensure: the better feasible solution X_{best}
1: Initialise and evaluate HM. Set generation counter $lp = 1, NI = 0$.
2: **for** $i = 1$ to NI **do**
3: Generate $HMCR$ and PAR according to $HMCRm$ and $PARm$. Yield bw according to bw_{max} and bw_{min}.
4: **for** $j = 1$ **to** n **do**
5: **if** $r \leqslant HMCR$ **then**
6: $x_{newj} = x_{aj} \pm r1 * bw. where\, a \in \{1, \cdots, HMS\}$;
7: **if** $r2 \leqslant PAR$ **then**
8: $x_{newj} = x_{bestj}$;
9: **end if**
10: **else**
11: $x_{newj} = x_L + (x_U - x_L) * r3$;
12: **end if**
13: **end for**
14: **if** $f(x_{new}) < f(x_{worst})$ **then**
15: update the HM as $x_{worst} = x_{new}$; Record the values of $HMCR$ and PAR.
16: **end if**
17: **if** $lp = LP$ **then**
18: recalculate $HMCRm$, $PARm$ according to the recorded values of $HMCR$, PAR and reset $lp = 1$;
19: **else**
20: $lp = lp + 1$;
21: **end if**
22: **end for**
23: **Note**: $where\, r$, $r1$, $r2 \in [0, 1]$

3 The SAHS-SLS for the MKP

SAHS-SLS initialises the HM population following the RK method before starting the optimisation. After that, the optimisation step consists in improvising a new harmony according to SAHS, then with probability P, apply a local search SLS.

3.1 The bw tuning strategy

In the improvisation step, bw is generated in two phases. First, bw is computed according to the number of iterations (NI). bw starts with a high value and decreases continuously until half NI which allows to explore the search space. In the second phase, for each improvised harmony X_j corresponds a bw_j value set at random in $[bw_{min}, bw_{max}]$. The bw is generated as the following strategy:

$$bw_j = \begin{cases} \dfrac{bw_{max} - bw_{min}}{NI} & i \leq NI/2 \\ bw_{rand} \quad \in [bw_{max}, bw_{min}] & i > NI/2 \end{cases}$$

3.2 The modified PAR

In SAHS-SLS, the PAR is modified as follows: the selected item from X_{best} of HM is slightly adjusted with a bw generated according to the strategy seen previously. The adjustment is applied with a probability $pbw2$. A small value of PAR guaranties the compromise between exploitation and exploration. Contrary, a high value increases the exploration but decreases the exploitation and the obtained solutions will be of bad quality. The proposed PAR is given in Algorithm 3 by lines 8–13.

3.3 The wp parameter adaptation

The wp parameter is dynamically generated in SAHS-SLS. At each iteration, a value of wp is generated according to a normal distribution with a mean value WPm and deviation (0.01). The values of wp allowing SAHS-SLS to obtain a harmony better than the worst harmony among HM are saved. After a number of iterations LP, WPm is adjusted to the average of the saved best wp values.

4 Experiments

The proposed algorithms were coded in C++. The experiments were run on an Intel core 2 duo CPU 2 GHz and 2 GB of RAM and Windows 7 32-bit. Large MKP dataset available in the OR-library (Chu and Beasley, 1998) have been used. The problem set consists of 270 instances. There are nine classes each one composed of three groups, ten instances in every group. The number of constraints m and items n are 5, 10, 30 and 100, 250, 500 respectively. The optimal solution for some instances are yet unknown. We compared the obtained results to those collected the web page (Results, 2012).

Table 1 Impact of HMS in SAHS-SLS

.	.	*10*	*30*	*70*	*100*	*200*	*500*
(5, 10, 30) × 100_0.25	Deviation	1.092	*0.718*	0.763	0.789	0.708	1.054
.	Time	34.063	36.676	50.091	46.386	54.681	163.209
(5, 10, 30) × 250_0.25	Deviation	2.016	*1.831*	1.865	1.891	2.014	2.541
.	Time	68.337	71.698	81.109	83.943	98.091	191.815
(5, 10, 30) × 500_0.25	Deviation	2.538	*1.948*	1.964	1.982	2.399	3.423
.	Time	166.127	159.143	174.019	177.917	194.035	328.02
Average	*Average deviation*	1.882	*1.499*	1.530	1.554	1.707	2.339
.	*Average time*	89.509	89.172	101.739	102.748	115.602	227.681

Algorithm 3 The SAHS-SLS algorithm

Require: Set $HMS, NI, n, HMCRm, PARm, WPm, LP, P, MI, pbw1, pbw2, bw_{min}$ and bw_{max}.
Ensure: the better feasible solution X_{best}
initialise HM according to RK and set $lp = 1, NI = 0$.
for $i = 1$ to NI **do**
  Generate $HMCR$ PAR and wp according to $HMCRm$, $PARm$ and WPm. Yield bw.
  improvise a new harmony X_{new} as follows:
  for $j = 1$ **to** n **do**
    if $r \leqslant HMCR$ **then**
      $x_{newj} = x_{ak} \pm pbw1 * bw$.
      where $a \in \{1, \cdots, HMS\}$;
      if $r1 \leqslant PAR$ **then**
        $x_{newj} = x_{bestj} \pm pbw2 * bw$.
      end if
    else
      $x_{newj} = x_{rand}$;
    end if
  end for
  if $r2 \leqslant P$ **then**
    apply SLS to the new harmony X_{new}
  end if
  if $f(X_{new}) > f(X_{worst})$ **then**
    Update HM as $X_{worst} = X_{new}$;
    Record the values of HMCR, PAR and wp
  end if
  if $lp = LP$ **then**
    Recalculate $HMCRm$, $PARm$ and WPm according to the recorded values of $HMCR$, PAR and wp respectively; $lp = 1$
  else
    $lp = lp + 1$
  end if
end for

4.1 The parameters tuning

This experiment aims to determine the impact of HMS and P on SAHS-SLS. The parameters of the proposed algorithms are empirically fixed. In each experiment, only the tested parameter is changed and one MKP instance is used. The average fitness and CPU time is taken over 20 runs. HMS varies between 10 and 100, and $P \in$ [0, 1].

4.1.1 The impact of HMS

We used six different values of HMS to evaluation its impact on effectiveness and speed of SAHS-SLS. We used the ten first instances from each of the nine MKP benchmark classes (those with $\alpha = 0.25$). The results showed in Table 2 represent the average value of CPU time and the fitness obtained for instances having the same number of items $m = 100$, $m = 250$ and $m = 500$.

Table 2 The appropriate probability of SLS application

	0	*0.1*	*0.2*	*0.4*	*0.8*	*0.9*	*1*
Deviation (%)	1.22	0.6	0.58	0.68	*0.52*	0.59	0.64

Table 3 The values of the algorithms parameters

Parameter	*Value*
HMS	30
$[BW_{min}, BW_{max}]$	[1, 10]
pbw1 and pbw2	0.0001
P	0.8
HMCR	[0.9, 1]
PAR and wp	[0, 1]
HMCRm	0.99
PARm	0.8
WPm	0.7
NI	{30,000, 50,000, 75,000, 100,000}
n	{700, 1,000, 1,500}
maxiter	200
LP	200

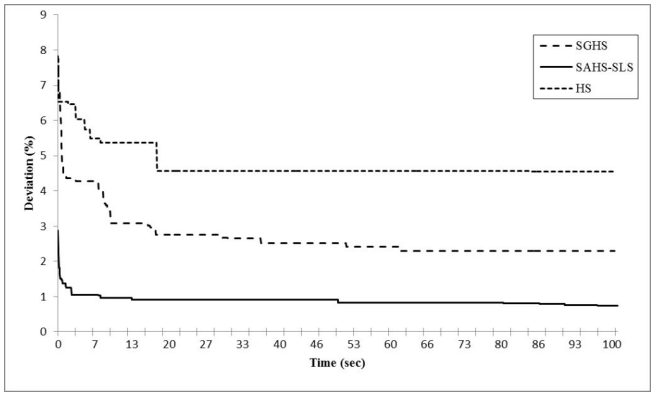
4.1.2 The SLS application probability

This experiment was carried out only with the OR10x250_0.75_1 first instance to determine, in an empirical way, the value of suitable probability for applying SLS following the ESP strategy. For that, the value of probability was varied between seven different values. The results are reported in Table 2.

Table 4 Comparaison of SAHS-SLS, SGHS and HS

	SAHS-SLS		*SGHS*		*HS*	
	Deviation	*Time*	*Deviation*	*Time*	*Deviation*	*Time*
OR5x100_0.25	0.6	18.89	0.775	16.33	5.355	9.545
OR5x100_0.50	0.568	24.79	0.784	22.32	5.148	12.35
OR5x100_0.75	0.316	29.06	0.469	27.56	3.087	16.55
Average 5x100	0.49	24.24	0.676	22.07	4.53	12.817
OR5x250_0.25	2.143	31.54	2.623	26.56	8.411	22.39
OR5x250_0.50	1.477	68.07	2.266	57.16	6.211	36.31
OR5x250_0.75	1.215	99.34	1.766	80.45	3.778	62.32
Average 5x250	1.611	66.31	2.218	54.72	6.133	40.34
OR5x500_0.25	1.854	146.3	3.754	109.06	9.569	66.74
OR5x500_0.50	1.314	248.36	3.595	177.27	7.245	116.41
OR5x500_0.75	1.331	339.97	2.685	269.62	4.673	187.63
Average 5x500	1.499	244.87	3.344	185.32	7.162	123.596
OR10x100_0.25	0.9	29.01	1.261	27.84	5.31	16.46
OR10x100_0.50	0.636	46.6	0.967	33.61	4.896	20.58
OR10x100_0.75	0.358	48.8	0.641	43.51	3.223	27.32
Average 10x100	0.631	41.47	0.956	34.99	4.476	21.458
OR10x250_0.25	1.493	93.28	2.328	74.11	7.168	44.74
OR10x250_0.50	0.913	160.19	2.059	116.26	6.013	76.59
OR10x250_0.75	0.673	261.48	1.592	172.83	3.845	116.62
Average 10x250	1.026	171.65	1.993	121.07	5.675	79.320
OR10x500_0.25	2.113	157.14	4.157	118.50	9.951	78.03
OR10x500_0.50	1.61	267.03	4.226	207.42	7.951	143.54
OR10x500_0.75	1.448	384.42	4.11	297.31	5.04	229.34
Average 10x500	1.723	269.53	4.163	207.74	7.647	150.3
OR30x100_0.25	1.229	76.98	0.93	68.49	2.792	49.69
OR30x100_0.50	0.684	64.93	0.958	52.54	3.904	36.46
OR30x100_0.75	0.334	87.05	0.519	67.913	3.215	46.848
Average 30x100	0.749	76.32	0.802	62.986	3.303	44.33
OR30x250_0.25	1.928	155.91	2.471	91.9	7.586	61.823
OR30x250_0.50	0.837	155.75	2.493	102.97	6.996	71.56
OR30x250_0.75	0.664	231.71	1.772	150.69	4.277	108.27
Average 30x250	1.143	181.12	2.245	115.18	6.286	80.553
OR30x500_0.25	2.24	173.32	5.625	127.13	9.709	88.99
OR30x500_0.50	1.19	264.16	4.82	206.27	8.387	137.07
OR30x500_0.75	0.87	343.99	6.309	285.05	6.435	243.26
Average 30x500	1.433	260.49	5.584	206.15	8.177	156.44
Average all	1.145	148.44	2.438	108.644	6.005	74.006

Figure 1 The convergence speed of SAHS-SLS, SGHS and HS



4.2 The SAHS-SLS speed investigation

By using the same instance OR10x250_0.75_1 we carried out experimental tests in order to compare the speed of convergence of the three algorithms. It consists in running SAHS-SLS, SGHS and HS during 100 seconds. The value of fitness was recovered each 0.05 second. The obtained results were translated into the curves depicted in Figure 1.

4.3 The results on large benchmarks

The constant values were assigned to the parameters to which the values are not dynamically generated. Table 3 summarises the values of those parameters for the three algorithms used to carry out the experiments. Table 4 reports the results of SAHS-SLS, SGHS and HS.

5 Discussion

From Figure 1, we see that SAHS-SLS converges more quickly than SGHS and HS. SAHS-SLS continues to converge up to 100 seconds while SGHS and HS do not converge any more after certain time. The results in Table 2 show that $P = 0.8$ gives best convergence. From Table 4, it can be seen that for the instances with $n = 100$, SAHS-SLS and SGHS succeed to approach the optimal solution. But SAHS-SLS requires only two seconds (in average) to improve SGHS of 0.186%, as long as it largely exceeds HS. For the larger benchmarks (5×250, 5×500 and 10×500), SAHS-SLS was able to converge and avoid the local optimum. The results show that SAHS-SLS exceeds SGHS of 0.273% up to 4.151% and HS of 2.554% up to 6.744%. Furthermore, the comparison shows that the proposed PAR in SAHS-SLS is very effective and it is more effective with bw adjusting. In the same time, the study shows that the proposed bw adjustment allows SAHS-SLS to discover new regions of solutions and then to converge more quickly to the optimal solution.

6 Conclusions

In this paper, we proposed the SAHS-SLS to solve the MKP. In SAHS-SLS new PAR and bandwidth (bw) strategies were proposed. Further, SAHS was hybridised with SLS according to the every solution probability (ESP) strategy with probability $P = 0.8$. Several tests were carried out on a large range of benchmarks to determine the effectiveness and the speed of SAHS-SLS. The obtained results showed that SAHS-SLS was more effective and more speed than both SGHS and HS. In addition, our study showed that the algorithms based on HS are very effective thanks to the HM, the PAR and bw.

References

Aboudi, R. and Jornsten, K. (1994) 'Tabu search for general zero-one integer programs using the pivot and complement heuristic', *ORSA J. on Computing*, Vol. 6, No. 1, pp.82–93.

Bean, J.C. (1994) 'Genetics and random keys for sequencing and optimization', *in ORSA Journal of Computing*, Vol. 6, No. 2, pp.154–160.

Boughaci, D., Benhamou, B. and Drias, H. (2010) 'Local search methods for the optimal winner determination problem in combinatorial auctions', *Journal of Math. Model. Algor.*, Vol. 9, No. 1, pp.165–180.

Boussier, S., Vasquez, M., Vimont, Y., Hanafi, S. and Michelon, P. (2010) 'A multi-level search strategy for the 0-1 multidimensional knapsack problem', *Discrete Applied Mathematics*, Vol. 158, No. 2, pp.97–109.

Cho, J.H. and Kim, Y.D. (1997) 'A simulated annealing algorithm for resource-constrained project scheduling problems', *Journal of the Operational Research Society*, Vol. 48, No. 7, pp.736–744.

Chu, P. and Beasley, J. (1998) 'A genetic algorithm for the multidimensional knapsack problem', *Journal of Heuristics*, Vol. 4, No. 1, pp.63–86.

Duan, Q., Liao, T.W. and Yi, H.Z. (2013) 'A comparative study of different local search application strategies in hybrid metaheuristics', *Applied Soft Computing*, Vol. 13, No. 3, pp.1464–1477.

Egeblad, J. and Pisinger, D. (2009) 'Heuristic approaches for the two- and three-dimensional knapsack packing problem', *Computers and Operations Research*, Vol. 36, No. 4, pp.1026–1049.

Freville, A. and Plateau, G. (1997) 'The 0-1 bidimensional knapsack problem: toward and efficient high-level primitive tool', *Journal of Heuristics*, Vol. 2, No. 2, pp.147–167.

Fukunaga, A.S. (2011) 'A branch-and-bound algorithm for hard multiple knapsack problems', *Annals of Operations Research*, Vol. 184, No. 1, pp.97–119.

Geem, Z.W., Kim, J.H. and Loganathan, G.V. (2001) 'A new heuristic optimization algorithm: harmony search', *Simulations*, Vol. 76, No. 2, pp.60–68.

Hill, R.R., Cho, Y.K. and Moore, J.T. (2012) 'Problem reduction heuristic for the 0–1 multidimensional knapsack problem', *Computers and Operations Research*, Vol. 39, No. 1, pp.19–26.

Khuri, S., Back, T. and Heitkötter, J. (1994) 'The zero/one multiple knapsack problem and genetic algorithms pages', Paper presented in *Proceedings of the ACM Symposium on Applied Computing*, ACM Press, pp.188–193.

Li, Z.K. and Li, N. (2009) 'A novel multi-mutation binary particle swarm optimization for 0/1 knapsack problem', Paper presented at the *Control and Decision Conference*, pp.3042–3047.

Lin, F.T. (2008) 'Solving the knapsack problem with imprecise weight coefficients using genetic algorithms', *European Journal of Operational Research*, Vol. 185, No. 1, pp.133–145.

Liu, Y. and Liu, C. (2009) 'A schema-guiding evolutionary algorithm for 0-1 knapsack problem', Paper presented at the*Int. Association of Computer Science and Information Technology*, Springer, pp.160–164.

Pan, Q-K., Suganthan, P.N., Tasgetiren, M. and Liang, J.J. (2010) 'A self-adaptive global best harmony search algorithm for continuous optimization problems', *Appli. Mathematic & Computation.*, Vol. 216, No. 3, pp.830–848.

Results (2012) *Chu and Beasley Large Benchmarks Best Results* [online] http://www.cs.nott.ac.uk/ jqd/mkp/results.html.

Shi, H.X. (2006) 'Solution to 0/1 knapsack problem based on improved ant colony algorithm', Paper presented at the *International Conference on Information Acquisition*, pp.1062–1066.

Vimont, Y., Boussier, S. and Vasquez, M. (2008) 'Reduced costs propagation in an efficient implicit enumeration for the 01 multidimensional knapsack problem', *Journal of Combinatorial Optimization*, Vol. 15, No. 2, pp.165–178.

Volgenant, A. and Zoon, J. (1990) 'An improved heuristic for multidimensional 0-1 knapsack problems', *Journal of Operational Research Society*, Vol. 41, No. 10, pp.963–970.

Yoon, Y., Kim, Y-H. and Moon, B-R. (2005) 'An evolutionary Lagrangian method for the 0-1 multiple knapsack problem', *in Proceedings of the Genetic and Evolutionary Computation Conference (GECCO '05)*. June, pp.629–635.

Zou, D., Gao, L., Li, S. and Wu, J. (2011) 'Solving 0-1 knapsack problem by a novel global harmony search algorithm', *Applied Soft Computing*, Vol. 11, No. 2, pp.1556–1564.