Volume 64, Issue 1, 2020
Journal of Scientific Research
Institute of Science,
Banaras Hindu University, Varanasi, India.

Solving single and multi-objective 01 Knapsack Problem using Harmony Search Algorithm

Amol C. Adamuthe[1*], Vaishnavi N. Sale[1], and Sandeep U. Mane[2]

[1]Dept. of CS & IT, RIT, Rajaramnagar, Sangli, MS, India. alembantemulu184@gmail.com, vishal.pup@gmail.com
[2]Dept. of CSE, RIT, Rajaramnagar, Sangli, MS, India. manesandip82@gmail.com

Abstract: **Knapsack problem is finding the optimal selection of objects to get maximum profit. Knapsack problem has a wide range of application in different domain such as production, transportation, resource management etc. Knapsack problem varies with change in number of items and number of objectives. 01 knapsack problem is reported as a classical optimization problem under NP category. Harmony search (HS) algorithm is a popular heuristic algorithm investigated to solve different optimization problems. This paper presents harmony search for solving single objective and multi-objective knapsack problem. Performance of HS is tested with 43 instances of single objective knapsack problem taken from three datasets. HS provides optimal results except for three instances. 46 instances of 01 knapsack problem with three, four and five objectives are tested. Experiments show that better results are obtained with an increase in harmony memory with better exploration in objectives.**

Index Terms: **Harmony search algorithm, knapsack problem, optimization problem.**

I. INTRODUCTION

The knapsack problem is a constrained combinatorial optimization problem. This is a classical NP problem in operation research (Bansal & Deep, 2012). It has various applications in different industries. Some of the application areas of knapsack problem are,

- project selection
- resource distribution
- resource management/scheduling
- power allocation management.

Single objective 01 knapsack problem is well known problem. The problem description is presented by many authors. In 01 knapsack problem, items with varying weights and respective profits are given. A knapsack of capacity is given. The objective is to select objects in such a way that maximum profit should be gained with available capacity. Given a set of n objects which are numbered from 1 up to n. Each object i has a weight Wi and associated profit Pi. Maximum weight capacity of knapsack is, M.

Objective Function:

$$\text{Maximize: } \sum P_i X_i \quad \text{for } i=1 \text{ to } n \qquad (1)$$

$$\text{Subject to: } \sum W_i X_i \leq M \quad \text{for } i=1 \text{ to } n$$

$$X_i \in \{0, 1\} \quad \text{for } i=1 \text{ to } n$$

Here X_i has value 1 is it is selected else 0. The objective of the problem is to maximize the sum of the profits of the items selected in the knapsack with sum of the weights less than or equal to the knapsack's capacity.

Multi-objective optimization means optimizing more than one objective function simultaneously. Multi-objective optimization problems are present in different areas such as transportation, engineering, economics etc. The problem is difficult when the objectives are conflicting. The definition of multi-objective knapsack is taken from (Kirlik & Sayın, 2014). For multi-objective knapsack problem equation (1) is the set of p objective functions. Each objective function is the total profit of selected objects. In case of multi-objective knapsack problem, multiple pair of weights and associated profits of the objects are known. The objective is to maximize profit is all given cases.

In literature, different heuristic algorithms have experimented for unconstrained and constrained optimization problems. Different heuristic and metaheuristic algorithms are experimented for solving 01 knapsack problem.

- Evolutionary algorithm (Liu & Liu, 2009)

DOI: http://dx.doi.org/10.37398/JSR.2020.640136

- Genetic algorithm (Zhao et al., 2009; Pradhan et al., 2014)
- Particle swarm optimization (Bansal & Deep, 2012; Li & Li, 2009; Ouyang &Wang, 2012)
- Wolf Pack Algorithm (Gao et al., 2018)
- firefly algorithm (Hajarian et al., 2016; Bhattacharjee & Sarmah, 2015)
- shuffled frog leaping algorithm (Bhattacharjee & Sarmah, 2014)

Genetic algorithms, particle swarm optimization, differential evolution are popular heuristic/meta-heuristic algorithms. Harmony search algorithm is a population-based algorithm that imitates the music improvisation process used by the musicians. Harmony search is used to solve various problems (Geem, 2009; Rao et al., 2010; Fesanghary et al., 2008; Adamuthe & Nitave, 2018).

Harmony search algorithm is investigated to optimize different mathematical functions and real world applications. It is investigated to solve different engineering problems from civil engineering, mechanical engineering, transportation, electrical engineering, telecommunications, image processing etc. (Askarzadeh & Rashedi, 2018; Geem, 2008)

- Rosenbrock's banana function and multiple local optima functions (Lee & Geem, 2005)
- Optimal designing of wireless sensor networks (WSN) (Guney & Onay, 2011).
- Water distribution network design problem was solved in (Geem 2012)
- Transportation problem (Salcedo-Sanz et al. 2013),
- Improve the accuracy of ANN for classification (Kulluk et al. 2012)

This paper presents harmony search algorithm for solving knapsack problem. The objective of this paper is to optimize single objective and multi-objective 01 knapsack problem using harmony search algorithm.

The next section briefly describes harmony search algorithm for solving knapsack problem. Section III describes the experimental details, results and discussion. To end, in Section IV presents conclusions of our work.

II. HARMONY SEARCH ALGORITHM

Harmony search is population-based heuristic algorithm. The algorithm is influenced by the music improvisation process (Wang et al., 2015). In the last decade, harmony search algorithm is investigated to solve various optimization problems. Figure 1 presents the pseudocode of harmony search algorithm for 01 knapsack problem.

The fundamental steps involved in harmony search algorithm are (Wang et al., 2015):

- Step 1: Initialize the parameters of the algorithm. Harmony search algorithms have three important parameters.
  - harmony memory size (HMS),
  - harmony memory consideration rate (HMCR), and
  - pitch adjusting rate (PAR)
- Step 2: Randomly initializing harmony memory (HM). The initial HM consists of randomly generated solutions.
- Step 3: Improvise a new solution from HM. HMCR indicates the probability of selecting a component from initial HM for improvisation. PAR is the probability of mutation for selected solution.
- Step 4: Update the harmony memory. If the improvised solution obtained in step 3 is better than the solution in the HM, then it will replace the later. Otherwise, it is simply neglected.
- Step 5: Repeat steps 3 and 4 until the termination condition is satisfied. Generally, termination condition is maximum iterations.

Algorithm 1. Harmony Search for 01 Knapsack problem

Input: A Number of objects, weights & profits of all objects and knapsack capacity.
Output: Profit and selection of objects.

Initialize HMS, HMCR and PAR and maximum iterations
Define objective function

```
/* Initialization harmony memory strategy */
while i ≤ HMS do
  while j ≤ number_of_objects do
       if capacity ≥ current_obj_size then
         HM (i,j) = 1;
           capacity  = capacity  – current_obj_size
      else
         HM(i,j) = 0;
  end while
end while
 /* Improvise the harmony memory */
while i ≤ max_iterations do
  while j≤ HMS do
    while k ≤ no_of_obj do
      if rand[0,1] < HMCR then
        memory consideration (j);
        if(HM_diversity < threshold)
         PAR =  PAR + (PAR_max- ((PAR_max-PAR_min) x i) /
max_iterations)
          if rand[0,1] < PAR then
              pitch adjustment();
          else
              random solution();
               accept new solution if better than previous;
```

end while
end while
end while
end procedure

Fig. 1. Harmony Search for 01 Knapsack problem

III. Experimental results And Performance comparison

This section gives a detailed explanation about the datasets used and the results obtained. The proposed algorithm is implemented using 'C' programming language and tested on a computer with the following specifications: Windows 7 Professional, Intel core i5-3210M CPU 2.5 GHz and 4 GB RAM. For every dataset, harmony search algorithm was executed for 10 times.

Memory representation: A 1D representation as shown in figure2 is used to solve the knapsack problem. Number of objects indicates the size of the array. The values in the array indicate selection or rejection of the object. Value 1 and 0 indicate selection and rejection respectively. Figure2 shows the sample solution with 10 objects. Objects 3, 4, 5, 7 and 9 are selected.

The implementation consists of the following functions.

1. The memory representation is defined as integer array named allocation [max_no_objects+1] and objective values are stored in array fitness [max_no_obj_fun].
2. Function of import_data() to take the input values.
3. Function of export_data() to show out output values.
4. Initialize_harmony_mem() to initialize population for harmony approach.
5. Calculate_penalty() to evaluate the values for defined constraint violations.
6. Find_worst() function to find the worst value and replace it with next good value, as it is important consideration in harmony search approach.
7. Pitch_adjustment() to adjust the par index.
8. Memory_updation() to update memory after finding the worst to remove it and to insert new best value in harmony memory.
9. Best_fit() to find optimal solution from population.
10. Finally, mean() to calculate mean for overall population and also for population respective of each iteration.

X1	X2	X3	X4	X5	X6	X7	X8	X9	X10
0	0	1	1	1	0	1	0	1	0

Fig. 2. Memory representation

A. Dataset 1:

Eight instances of knapsack are available at (Knapsack_01 Data for the 01 Knapsack problem) presented in table I. Knapsack capacity, the weights of the objects, the profits of each object and the optimal selection of weights is given.

Table I. Dataset 1

Dataset	Dimension	Parameter (capacity, weight, profit)
p_01	10	Capacity: 165 Weights: 23 31 29 44 53 38 69 85 89 8 Profits: 92 57 49 68 60 43 67 84 87 72
p_02	5	Capacity: 26 Weights: 12 7 11 8 9 Profits: 24 13 23 15 16
p_03	6	Capacity: 190 Weights: 56 59 80 64 75 17 Profits: 50 50 64 46 50 5
p_04	7	Capacity: 50 Weights: 31 10 20 19 4 3 6 Profits: 70 20 39 37 7 5 10
p_05	8	Capacity: 104 Weights: 25 35 45 5 25 3 2 2 Profits: 350 400 450 20 70 8 5 5
p_06	7	Capacity: 170 Weights: 41 50 49 59 55 57 60 Profits: 442 525 511 593 546 564 617
p_07	15	Capacity: 750 Weights: 70 73 77 80 82 87 90 94 98 106 110 113 115 118 120 Profits: 135 139 149 150 156 163 173 184 192 201 210 214 221 229 240
p_08	24	Capacity: 6404180 Weights: 382745 799601 909247 729069 467902 44328 34610 698150 823460 903959 853665 551830 610856 670702 488960 951111 323046 446298 931161 31385 496951 264724 224916 169684 Profits: 825594 1677009 1676628 1523970 943972 97426 69666 1296457 1679693 1902996 1844992 1049289 1252836 1319836 953277 2067538 675367 853655 1826027 65731 901489 577243 466257 369261

Table II shows the optimal profits and obtained profit values using harmony search algorithm. For all the instances, harmony search algorithm gives optimal results.

Table II. Results for dataset 1

Dataset	Optimal	Harmony search
p_01	309	309
p_02	51	51
p_03	150	150
p_04	107	107

p_05	900	900
p_06	1735	1735
p_07	1458	1458
p_08	13549094	13549094

B. Dataset 2:

Bhattacharjee & Sarmah (2014) has given ten test problems of knapsack presented in table III. The problem dimension, object weights and respective profits and knapsack capacity is given.

Authors have experimented with Shuffled frog leaping algorithm to solve the problem instances. The dataset mentioned in the paper algorithm by Bhattacharjee & Sarmah (2014) is used test performance of harmony search. The obtained results are presented in table IV. Harmony search gives best results for seven instances. Table V shows that optimal values for dataset f2, f8 and f10 are not obtained. The obtained values are close to the best. Comparison of average results obtained shows that shuffled frog optimization in better than harmony search algorithm.

Table III. Dataset 2 (Taken from algorithm (Bhattacharjee & Sarmah, 2014))

f	Dimension	Parameter(w, p, b)
f1	10	w = {95, 4, 60, 32, 23, 72, 80, 62, 65, 46}; p = {55, 10, 47, 5, 4, 50, 8, 61, 85, 87}; b = 269.
f2	20	w = {92, 4, 43, 83, 84, 68, 92, 82, 6, 44, 32, 18, 56, 83, 25, 96, 70, 48, 14, 58}; p = {44, 46, 90, 72, 91, 40, 75, 35, 8, 54, 78, 40, 77, 15, 61, 17, 75, 29, 75, 63}; b = 878.
f3	4	w = {6, 5, 9, 7}; p = {9, 11, 13, 15}; b = 20.
f4	4	w = {2, 4, 6, 7}; p = {6, 10, 12, 13}; b = 11.
f5	15	w = {56.358531, 80.87405, 47.987304, 89.59624, 74.660482, 85.894345, 51.353496, 1.498459, 36.445204, 16.589862, 44.569231, 0.466933, 37.788018, 57.118442, 60.716575}; p = {0.125126, 19.330424, 58.500931, 35.029145, 82.284005, 17.41081, 71.050142, 30.399487, 9.140294, 14.731285, 98.852504, 11.908322, 0.89114, 53.166295, 60.176397}; b = 375.
f6	10	w = {30, 25, 20, 18, 17, 11, 5, 2, 1, 1}; p = {20, 18, 17, 15, 15, 10, 5, 3, 1, 1}; b = 60.
f7	7	w = {31, 10, 20, 19, 4, 3, 6}; p = {70, 20, 39, 37, 7, 5, 10}; b = 50.
f8	23	w = {983, 982, 981, 980, 979, 978, 488, 976, 972, 486, 486, 972, 972, 485, 485, 969, 966, 483, 964, 963, 961, 958, 959} ; p = {81, 980, 979, 978, 977, 976, 487, 974, 970, 485, 485, 970, 970, 484, 484, 976, 974, 482, 962, 961, 959, 958, 857} ; b = 10000.
f9	5	w = {15, 20, 17, 8, 31}; p = {33, 24, 36, 37, 12}; b =80.
f10	20	w = {84, 83, 43, 4, 44, 6, 82, 92, 25, 83, 56, 18, 58, 14, 48, 70, 96, 32, 68, 92}; p = {91, 72, 90, 46, 55, 8, 35, 75, 61, 15, 77, 40, 63, 75, 29, 75, 17, 78, 40, 44}; b = 879.

Table IV. Comparison of results for dataset 2

Functions	Results obtained by Harmony Search		Results of Shuffled Frog taken from (Bhattacharjee & Sarmah, 2014)	
	Best	Average	Best	Average
f1	295	167.4	295	287.7
f2	**945**	560.16	955	868
f3	35	26.5	35	35
f4	23	14.94	23	23
f5	481.06	390	481.07	408.55
f6	52	40.57	52	51.63
f7	107	52.58	107	106.73
f8	**9731**	7226.6	9759	9733.47
f9	130	71.89	130	130
f10	**889**	693.65	1010	879.9

Table V. Results for dataset 2

Dataset	Optimal (Bhattacharjee & Sarmah, 2015)	Harmony Search
f1	295	295
f2	1024	945
f3	35	35
f4	23	23
f5	481.06	481.06
f6	52	52
f7	107	107
f8	9767	9731
f9	130	130
f10	1025	889

C. Dataset 3:

The third dataset is taken from (Donald L. Kreher). Total 25 test instances are available with objects varying from 8 to 24. Obtained results are presented in table VI.

Table VI. Results for dataset 3

Dataset	Optimal (Bhattacharjee & Sarmah, 2015)	Harmony Search
8a	3924400	3924400
8b	3813669	3813669
8c	3347452	3347452
8d	4187707	4187707
8e	4955555	4955555
12a	5688887	5688887
12b	6498597	6498597

12c	5170626	5170626
12d	6992404	6992404
12e	5337472	5337472
16a	7850983	7850983
16b	9352998	9352998
16c	9151147	9151147
16d	9348889	9348889
16e	7769117	7769117
20a	10727049	10727049
20b	9818261	9818261
20c	10714023	10714023
20d	8929156	8929156
20e	9357969	9357969
24a	13549094	13549094
24b	12233713	12233713
24c	12448780	12448780
24d	11815315	11815315
24e	13940099	13940099

D. Dataset 4:

The multi-objective knapsack problem instances are taken from (Multiobjective optimization library). Instances are available with 3, 4 and 5 number of objective functions (p). The data file names are given in the following format, "KP_p-X_n-Y_ins-Z". X represents the number of objective functions, Y shows the number of objects and Z is the instance number.

Table VII shows the results of 46 instances of knapsack problems with 3, 4 and 5 objectives respectively. For instances with three objective functions, the number of objects considered are 10, 20 and 30. For instances with four and five objective functions, the number of objects considered are 10 and 20. The results are taken with varying harmony memory size from 30 to 240. Better results are obtained with increase in harmony memory size.

Table VII. Results for dataset 4

Dataset Instances	**HM=30**	**HM=60**	**HM=90**	**HM=120**	**HM=150**	**HM=180**	**HM=210**	**HM=240**
KP_p-3_n-10_ins-1	8461	9077	9077	9327	10138	10138	10566	10566
KP_p-3_n-10_ins-2	6687	6779	6779	6779	6779	6779	6998	6998
KP_p-3_n-10_ins-3	9562	9562	10744	10744	10744	10744	10744	10744
KP_p-3_n-10_ins-4	9118	9997	11122	11122	11122	11122	11122	11122
KP_p-3_n-10_ins-5	6996	6996	6996	7241	7241	7241	7241	7241
KP_p-3_n-10_ins-6	7534	9526	9526	9526	9526	9526	9526	9526
KP_p-3_n-10_ins-7	18146	18146	18146	18146	18146	18146	18146	18146
KP_p-3_n-10_ins-8	9658	9658	10457	10457	10457	10457	10457	10457
KP_p-3_n-10_ins-9	8874	9110	9652	9652	9723	10449	10449	10449
KP_p-3_n-10_ins-10	8656	8790	9197	9306	10681	10681	10681	10681
KP_p-3_n-20_ins-1	16215	17775	17775	17775	17775	17775	17775	17775
KP_p-3_n-20_ins-2	14792	15152	16350	16753	16753	16753	16753	16753
KP_p-3_n-20_ins-3	13198	15528	16215	16861	17194	17194	17194	17194
KP_p-3_n-20_ins-4	12923	13617	14085	15815	16413	18231	18231	18231
KP_p-3_n-20_ins-5	12260	13385	14350	14680	15044	15440	16446	16446
KP_p-3_n-20_ins-6	12253	13381	14487	15536	16081	17880	18474	18474
KP_p-3_n-20_ins-7	11871	12580	13759	15828	16023	17488	17488	17488
KP_p-3_n-20_ins-8	12408	14000	14902	15375	16268	17856	18458	18458
KP_p-3_n-20_ins-9	12667	13985	14511	15953	17431	18481	18481	19151
KP_p-3_n-20_ins-10	11984	13375	14138	15915	17491	18651	19346	19346

KP_p-3_n-30_ins-1	22366	23500	24168	24168	24099	24456	24456	24456
KP_p-3_n-30_ins-2	23416	23822	24217	26606	27641	27641	27641	27641
KP_p-3_n-30_ins-3	23239	25030	25890	26627	26786	27472	27472	27472
KP_p-3_n-30_ins-4	21281	22330	23778	25756	25756	27371	27500	27371
KP_p-3_n-30_ins-5	21847	22274	23344	26706	26948	28403	28403	28403
KP_p-3_n-30_ins-6	19812	20622	22620	25266	26556	26556	26556	26556
KP_p-3_n-30_ins-7	20219	21876	22749	23702	23240	24372	24372	24372
KP_p-3_n-30_ins-8	21651	22628	23789	24665	24671	27950	27950	27950
KP_p-3_n-30_ins-9	19171	21479	22700	24041	24277	24974	24974	24974
KP_p-3_n-30_ins-10	18731	19867	20195	23973	26363	26363	26363	26363
KP_p-4_n-10_ins-1	8850	9657	9657	10684	10684	11737	11737	11945
KP_p-4_n-10_ins-2	9906	10955	10955	10955	11103	11103	12118	12118
KP_p-4_n-10_ins-3	9385	10003	10003	10687	10687	11396	12556	13386
KP_p-4_n-10_ins-4	9470	9938	9938	10893	10893	12286	12464	12464
KP_p-4_n-10_ins-5	9983	9983	10570	11920	11920	12146	12146	12437
KP_p-4_n-20_ins-1	17448	18243	21994	21994	23628	23628	24374	25621
KP_p-4_n-20_ins-2	14173	15878	17874	18804	19055	19055	21244	21244
KP_p-4_n-20_ins-3	17483	18929	18929	20458	20458	21100	21100	21659
KP_p-5_n-10_ins-1	10142	10142	11633	11983	12494	14900	14900	15236
KP_p-5_n-10_ins-2	10949	10949	10949	12932	13773	14995	16024	16024
KP_p-5_n-10_ins-3	10968	10968	11363	11363	13933	13933	14818	14818
KP_p-5_n-10_ins-4	9421	9421	10074	10074	11360	11360	12039	12039
KP_p-5_n-10_ins-5	10112	10112	12860	13641	14701	14861	15204	15204
KP_p-5_n-20_ins-1	19619	20344	21976	23895	25748	25748	27812	27812
KP_p-5_n-20_ins-2	23478	24844	24844	27006	28912	28912	29331	29331
KP_p-5_n-20_ins-3	26410	28434	28434	31564	32593	32593	33960	33960

Figure 3 shows that the performance of harmony search changes with the harmonic memory size. Figure 4 shows the objective values for three objective instance. Results indicate that importance is given to all three objectives. Harmony search shows good exploration capability for multi-objective 01 knapsack problem.

CONCLUSIONS

Paper presents harmony search algorithm for 0/1 knapsack problem. Experiments conducted on 43 instances of single objective and 46 instances multi-objective 0/1 knapsack problem. HS gives optimal results with 100% success rate for 40 instances of single objective knapsack problem. Shuffled frog optimization algorithm is found better than harmony search algorithm. There is further scope to improve HS algorithm for improvement in average fitness of population. HS algorithm performs well for multi-objective 01 knapsack problem. The results show that importance is given to all objectives.

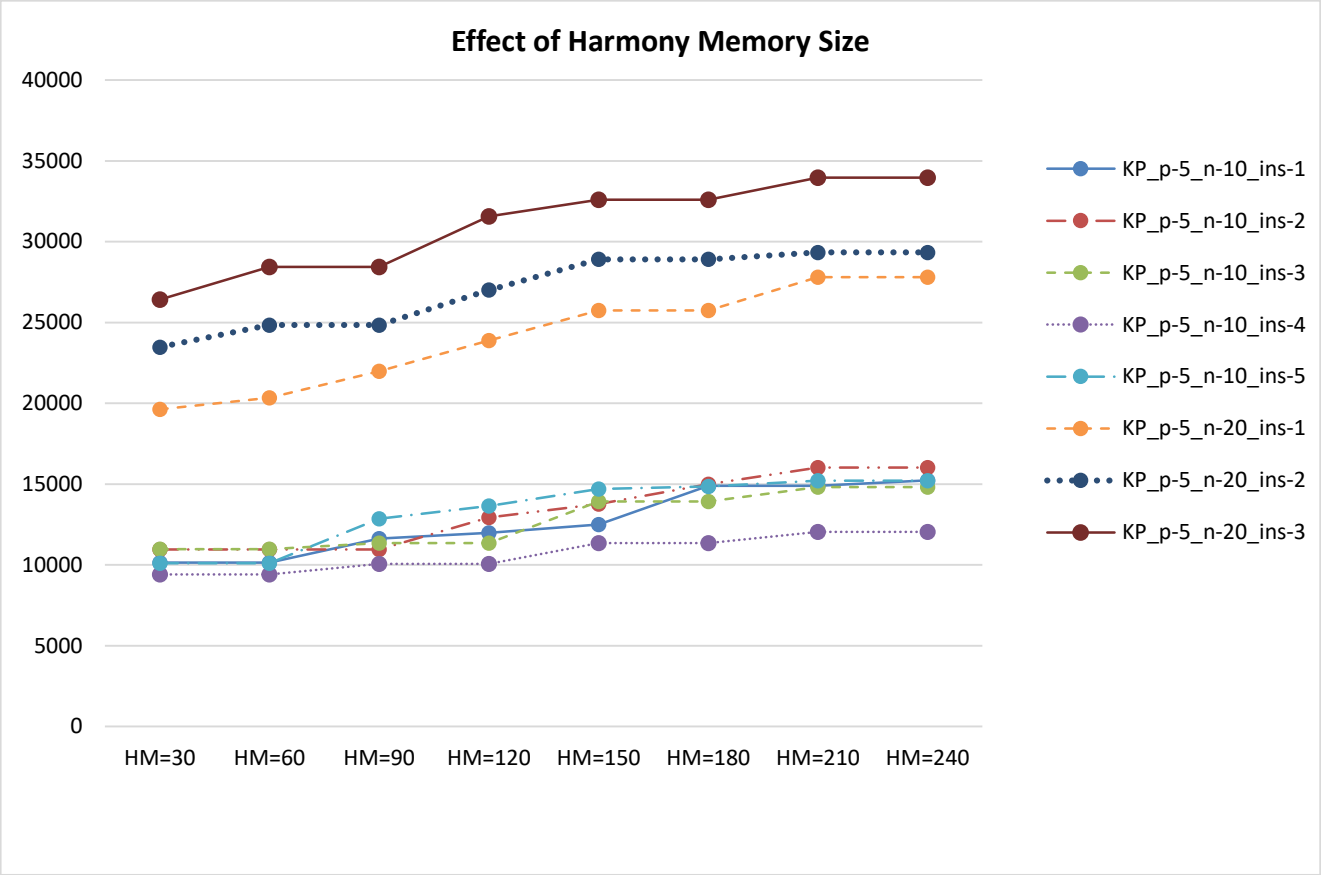

Fig. 3. Effect of Harmony memory size

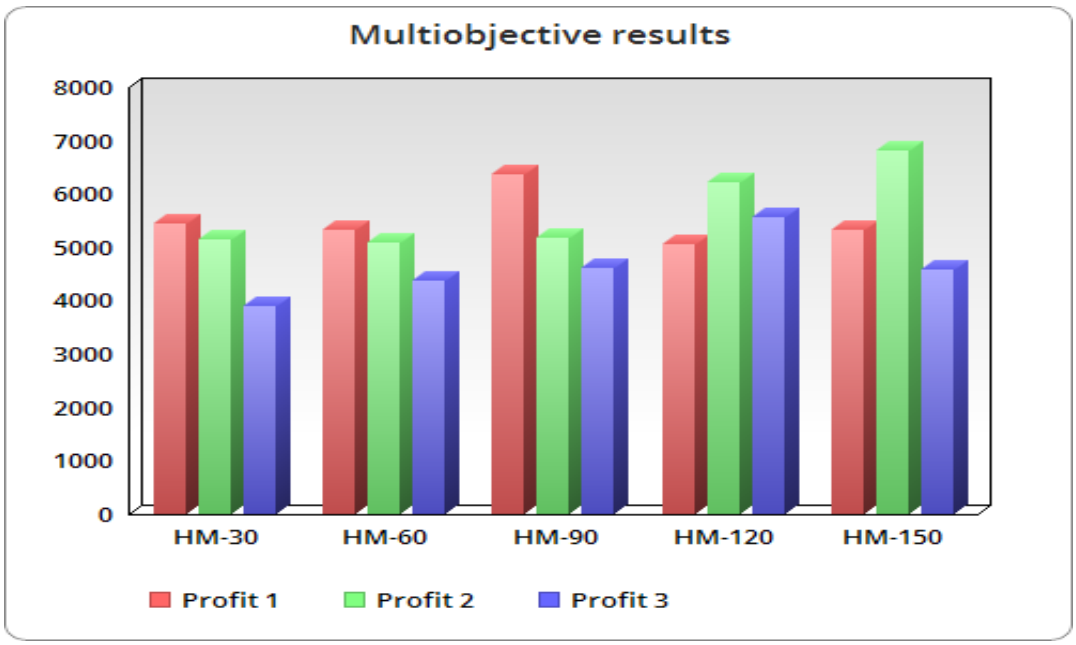

Fig. 4. Results of three objectives for sample instance

REFERENCES

Adamuthe, A. C., & Nitave, T. R. (2018). Adaptive Harmony Search for Optimizing Constrained Resource Allocation Problem. *International Journal of Computing*, *17*(4), 260-269.

Askarzadeh, A., & Rashedi, E. (2018). Harmony Search Algorithm: Basic Concepts and Engineering Applications. In *Intelligent Systems: Concepts, Methodologies, Tools, and Applications* (pp. 1-30). IGI Global.

Bansal, J. C., & Deep, K. (2012). A modified binary particle swarm optimization for knapsack problems. *Applied Mathematics and Computation*, *218*(22), 11042-11061.

Bhattacharjee, K. K., & Sarmah, S. P. (2014). Shuffled frog leaping algorithm and its application to 0/1 knapsack problem. *Applied soft computing*, *19*, 252-263.

Bhattacharjee, K. K., & Sarmah, S. P. (2015, December). A binary firefly algorithm for knapsack problems. In *2015 IEEE International Conference on Industrial Engineering and Engineering Management (IEEM)* (pp. 73-77). IEEE.

Donald L. Kreher. Retrieved from http://www.math.mtu.edu/~kreher/cages/Data.html

Fesanghary, M., Mahdavi, M., Minary-Jolandan, M., & Alizadeh, Y. (2008). Hybridizing harmony search algorithm with sequential quadratic programming for engineering optimization problems. *Computer methods in applied mechanics and engineering*, *197*(33-40), 3080-3091.

Gao, Y., Zhang, F., Zhao, Y., & Li, C. (2018). Quantum-Inspired Wolf Pack Algorithm to Solve the 0-1 Knapsack Problem. *Mathematical Problems in Engineering*, *2018*.

Geem, Z. W. (2008). Harmony search applications in industry. In *Soft Computing Applications in Industry* (pp. 117-134). Springer, Berlin, Heidelberg.

Geem, Z. W. (2009). Particle-swarm harmony search for water network design. *Engineering Optimization*, *41*(4), 297-311.

Geem, Z. W. (2012). Effects of initial memory and identical harmony in global optimization using harmony search algorithm. *Applied Mathematics and Computation*, *218*(22), 11337-11343.

Guney, K., & Onay, M. (2011). Optimal synthesis of linear antenna arrays using a harmony search algorithm. *Expert Systems with Applications*, *38*(12), 15455-15462.

Hajarian, M., Shahbahrami, A., & Hoseini, F. (2016, March). A parallel solution for the 0–1 knapsack problem using firefly algorithm. In *2016 1st Conference on Swarm Intelligence and Evolutionary Computation (CSIEC)* (pp. 25-30). IEEE.

Kirlik, G., & Sayın, S. (2014). A new algorithm for generating all nondominated solutions of multiobjective discrete optimization problems. *European Journal of Operational Research*, *232*(3), 479-488.

Knapsack_01 Data for the 01 Knapsack problem. Retrieved from https://people.sc.fsu.edu/~jburkardt/datasets/knapsack_01/knapsack_01.html

Kulluk, S., Ozbakir, L., & Baykasoglu, A. (2012). Training neural networks with harmony search algorithms for classification problems. *Engineering Applications of Artificial Intelligence*, *25*(1), 11-19.

Lee, K. S., & Geem, Z. W. (2005). A new meta-heuristic algorithm for continuous engineering optimization: harmony search theory and practice. *Computer methods in applied mechanics and engineering*, *194*(36-38), 3902-3933.

Li, Z., & Li, N. (2009, June). A novel multi-mutation binary particle swarm optimization for 0/1 knapsack problem. In *2009 Chinese Control and Decision Conference* (pp. 3042-3047). IEEE.

Liu, Y., & Liu, C. (2009, April). A schema-guiding evolutionary algorithm for 0-1 knapsack problem. In *2009 International Association of Computer Science and Information Technology-Spring Conference* (pp. 160-164). IEEE.

Multiobjective optimization library. Retrieved from http://home.ku.edu.tr/~moolibrary/

Ouyang, L., &Wang, D. (2012, May). New Particle Swarm Optimization algorithm for knapsack problem. In *2012 8th International Conference on Natural Computation* (pp. 786-788). IEEE.

Pradhan, T., Israni, A., & Sharma, M. (2014, May). Solving the 0–1 Knapsack problem using Genetic Algorithm and Rough Set Theory. In *2014 IEEE International Conference on Advanced Communications, Control and Computing Technologies* (pp. 1120-1125). IEEE.

Rao, R. S., Narasimham, S. V. L., Raju, M. R., & Rao, A. S. (2010). Optimal network reconfiguration of large-scale distribution system using harmony search algorithm. *IEEE Transactions on power systems*, *26*(3), 1080-1088.

Salcedo-Sanz, S., Manjarres, D., Pastor-Sánchez, Á., Del Ser, J., Portilla-Figueras, J. A., & Gil-Lopez, S. (2013). One-way urban traffic reconfiguration using a multi-objective harmony search approach. *Expert Systems with Applications*, *40*(9), 3341-3350.

Wang, X., Gao, X. Z., & Zenger, K. (2015). *An introduction to harmony search optimization method*. New York: Springer International Publishing.

Zhao, J., Huang, T., Pang, F., & Liu, Y. (2009, October). Genetic algorithm based on greedy strategy in the 0-1 knapsack problem. In *2009 Third International Conference on Genetic and Evolutionary Computing* (pp. 105-107). IEEE.

***