A SIMPLIFIED BINARY HARMONY SEARCH ALGORITHM FOR LARGE SCALE 0–1 KNAPSACK PROBLEMS

ALGORITHM IMPLEMENTATION REPORT

Supervised by: Dr. Ameera Jaradat

SADEEM AHMAD – 2015781004
Department of Computer Science,
Yarmouk University, Irbid, Jordan

INTRODUCTION:

In this paper, Researchers suggested a methodology to solve a large scaled 0-1 Knapsack problem using a simplified concept of Harmony Search, since 0-1 Knapsack has been one of the most popular NP-Hard problems, it took a great attention and has been studied for decades.

THE HARMONY SEARCH ALGORITHM:

The harmony search algorithm is inspired from the music improvisation process and a solution vector is analogy to a ‘‘harmony’’ here. Similar to other population based heuristic methods, a set of harmony vectors are firstly generated randomly in the variable space and stored in the harmony memory. **HS has three main procedures to improvise a new candidate harmony, that is, harmony memory consideration, pitch adjustment and random search**. The new harmony will replace the worst harmony vector in the whole HM only if it is better (with higher fitness). Continue the improvisation process until a predefined accuracy is achieved or certain number of improvisations has been accomplished.

Mainly it needs five parameters to be initialized in the first step to accomplish its operations:

- harmony memory size (HMS)
- harmony memory considering rate (HMCR)
- Pitch adjusting rate (PAR)
- pitch adjustment step (bw)
- number of improvisations (NI)

The paper mentioned the steps of how original Harmony Search Algorithm works.

RECENT HS VARIANTS FOR 0–1 OPTIMIZATION PROBLEMS

In order to find satisfactory solutions for the 0–1 optimization problems, several variants of HS have been developed to improve its performance. This section briefly reviews four recent extensions of the HS algorithm.

3.1. Binary-coding harmony search (BHS) algorithm by Geem (2005)

3.2. Discrete binary harmony search (DBHS) algorithm by Wang et al. (2010)

3.3. Novel global harmony search (NGHS1) algorithm by Zou et al. (2011)

3.4. Adaptive binary harmony search (ABHS) algorithm by Wang et al. (2013a)

SIMPLIFIED BINARY HARMONY SEARCH ALGORITHM

The paper presented a simplified binary harmony search algorithm (SBHS) to circumvent the drawbacks in current HS variants.

for the 0–1 optimization problems. In SBHS, only two parameters, i.e., HMS and HMCR, need to be set and an ingenious improvisation rule is introduced based on the difference between the best harmony and one randomly chosen harmony stored in the HM to implement pitch adjustment without any parameter such as PAR. HMCR is linearly increasing with the dimension to improve the optimization ability on different problems. To guarantee the feasibility of the solutions, a two-stage greedy procedure is employed to repair the infeasible solution vectors emerged in the HM.

METHODOLOGY:

The paper explained steps divided into two phases, first a Two-stage greedy procedure to repair the infeasible solution, second An ingenious improvisation scheme.

Work Environment:

Programming language: Java
IDE: Netbeans 8.2

Parameters:

HMS = 6

$$HMCR = \begin{cases} (0.9\ , 0.95), n < 100 \\ 1 - 10/n, n \geq 100 \end{cases}$$

- **Repairing Infeasible solution:**

  Two stages including and inserting greedily based on relative profit density, the knapsack can be filled according to the item profit as much as possible.

  It consists the following steps:

  Step 1: Given an Infeasible Harmony denoted as $\mathbf{x}=(x_1,x_2,\ldots,x_n)$. In our case it's all 1's.

  Step 2: Calculate the total volume (weight) $\mathbf{V_t}$ of the items chosen by the infeasible harmony "in our case all items were included", and the value of the constraint violation $\mathbf{V_c}$. (1)

  $$\boldsymbol{Vt} = \sum_{i=1}^{n} \text{vi} \,.\, \text{xi}, \mathbf{Vc} = \text{Vt} - \text{Vmax} \quad (1)$$

  Results: $V_t = 539.0$, $V_c = 244.0$

  Step 3: Get the corresponding item sequence S1 base on the relative profit density (2) in ascending order.

  $$\mathbf{u_i} = p_i/v_i \quad (2)$$

  Results: Sorted problem by ratio:

  v: 80.0 ,p: 8.0 **u:** 0.1
  v: 32.0 ,p: 5.0 **u:** 0.156
  v: 23.0 ,p: 4.0 **u:** 0.174
  v: 95.0 ,p: 55.0 **u:** 0.579
  v: 72.0 ,p: 50.0 **u:** 0.694
  v: 60.0 ,p: 47.0 **u:** 0.783
  v: 62.0 ,p: 61.0 **u:** 0.984
  v: 65.0 ,p: 85.0 **u:** 1.308
  v: 46.0 ,p: 87.0 **u:** 1.891
  v: 4.0 ,p: 10.0 **u:** 2.5

  Step 4: Remove the items in Sequence S1 until the total volume is smaller than the knapsack volume, the indicator in this step is the sign of $\mathbf{V_c}$, if $V_c < 0$ then stop removing items.

Results: Calculating V_c after getting back items from RemovedItems list: $\mathbf{V_c}$: -39.0

Removed items:
v: 80.0 ,p: 8.0 ,u: 0.1 ,x: 0
v: 23.0 ,p: 4.0 ,u: 0.174 ,x: 0
v: 72.0 ,p: 50.0 ,u: 0.694 ,x: 0
v: 62.0 ,p: 61.0 ,u: 0.984 ,x: 0
v: 46.0 ,p: 87.0 ,u: 1.891 ,x: 0

Remaining items:
v: 32.0 ,p: 5.0 ,u: 0.156 ,x: 1
v: 95.0 ,p: 55.0 ,u: 0.579 ,x: 1
v: 60.0 ,p: 47.0 ,u: 0.783 ,x: 1
v: 65.0 ,p: 85.0 ,u: 1.308 ,x: 1
v: 4.0 ,p: 10.0 ,u: 2.5 ,x: 1

Removed items size: 5
Total weight (V_t) after eleminaion:
$\mathbf{V_t}$ = 256.0

Step 5: Calculate the remaining volume of the knapsack V_l (which equals the absolute value of V_c), then judge weather the smallest item that has been eliminated in the previous step, if there aren't any terminate the repair strategy, otherwise continue.
Results: $\mathbf{V_l}$: -39.0

Step 6: Receive the corresponding item sequence S2 based on their volumes ascending.
Step 7: Store the items tabs in a sequence S3, the volume of which is smaller than the remaining volume of the knapsack V_l
Results: sorting S3 items based on their weights:
v: 23.0 ,p: 4.0 ,u: 0.174 ,x: 0
Step 8: Get the corresponding tab sequence S4 of those items contained in S3 based on their relative profit density in ascending order.
Results: sorting S4 items based on their ratio:
v: 23.0 ,p: 4.0 ,u: 0.174 ,x: 0
Step 9: Insert the items into the knapsack following the tab sequence S4 until there are no space in the knapsack.
Results: Repair Work results:
V_t = 279.0
V_c = -16.0

Knapsack items:
v: 4.0 ,p: 10.0 ,u: 2.5 ,x: 1
v: 23.0 ,p: 4.0 ,u: 0.174 ,x: 1
v: 32.0 ,p: 5.0 ,u: 0.156 ,x: 1
v: 60.0 ,p: 47.0 ,u: 0.783 ,x: 1
v: 65.0 ,p: 85.0 ,u: 1.308 ,x: 1
v: 95.0 ,p: 55.0 ,u: 0.579 ,x: 1

After the repair work through this mechanism, it's clear that the infeasible harmonies no longer violates the constraint.

- **An Ingenious Improvisation scheme:**

  Step 1: Set HMS and HMCR according to the dimension of a particular problem.
  Step 2: Initialize the HM, Total profits of each harmony are evaluated according to the item selected.
  Step 3: Determine the maximal Number of Iterations NI.
  Step 4: Improvise a new Harmony $\mathbf{x^{new}}$ ($x^{new}_1,\ldots,x^{new}_i$) as below and repair it if infeasible.
  Step 5: Replace the worst Harmony in the HM with $\mathbf{x^{new}}$ **only if it's better.**
  Step 6: Repeat steps 4-5 until NI is finished.
  Step 7: Output the best Harmonies.

BEST RESULTS: (OPTIMUM SOLUTION IS V_{MAX} = 295)	PROBLEM ITEMS: N = 10
0 ,0 ,1 ,1 ,0 ,0 ,1 ,1 ,1 ,1 ,V_T: 295 ,P_T: 302.0 *BEST RESULT	v: 80.0 ,p: 8.0 ,u: 0.1
0 ,1 ,0 ,0 ,1 ,1 ,1 ,1 ,0 ,1 ,V_T: 295 ,P_T: 258.0	v: 32.0 ,p: 5.0 ,u: 0.15625
0 ,1 ,1 ,0 ,1 ,1 ,1 ,0 ,1 ,0 ,V_T: 295 ,P_T: 254.0 1 ,1 ,0 ,0 ,1 ,0 ,0 ,1 ,1 ,0 ,V_T: 295 ,P_T: 235.0	v: 23.0 ,p: 4.0 ,u: 0.17391304347826086
1 ,1 ,1 ,1 ,0 ,0 ,0 ,1 ,0 ,0 ,V_T: 295 ,P_T: 157.0	v: 95.0 ,p: 55.0 ,u: 0.5789473684210527
	v: 72.0 ,p: 50.0 ,u: 0.6944444444444444
	v: 60.0 ,p: 47.0 ,u: 0.7833333333333333
	v: 62.0 ,p: 61.0 ,u: 0.9838709677419355
	v: 65.0 ,p: 85.0 ,u: 1.3076923076923077
	v: 46.0 ,p: 87.0 ,u: 1.891304347826087
	v: 4.0 ,p: 10.0 ,u: 2.5

OTHER DATA SETS

BEST RESULTS: (OPTIMUM SOLUTION IS V_{MAX} = 375)	PROBLEM ITEMS: N = 15 , KP5 (FROM THE PAPER)
0 ,0 ,0 ,1 ,0 ,0 ,0 ,1 ,1 ,1 ,1 ,1 ,0 ,1 ,0 ,V_T: 374 ,P_T: 374.907681 *BEST RESULT	v: 56.358531 ,p: 0.125126 ,u: 0.00222017852097671
1 ,1 ,0 ,0 ,0 ,0 ,0 ,1 ,0 ,1 ,1 ,1 ,1 ,0 ,0 ,V_T: 369 ,P_T: 364.870143	v: 37.788018 ,p: 0.89114 ,u: 0.02358260758741038
0 ,1 ,1 ,1 ,0 ,0 ,1 ,1 ,0 ,0 ,1 ,0 ,1 ,1 ,1 ,V_T: 372 ,P_T: 305.191198	v: 85.894345 ,p: 17.41081 ,u: 0.20270030582339268
0 ,0 ,0 ,1 ,0 ,1 ,0 ,1 ,0 ,0 ,1 ,1 ,1 ,0 ,1 ,V_T: 371 ,P_T: 347.837763	v: 80.87405 ,p: 19.330424 ,u: 0.23901886946430903
	v: 36.445204 ,p: 9.140294 ,u: 0.25079552305428177
	v: 89.59624 ,p: 35.029145 ,u: 0.39096668565555875
	v: 16.589862 ,p: 14.731285 ,u: 0.8879691102915744
	v: 57.118442 ,p: 53.166295 ,u: 0.9308078641220641
	v: 60.716575 ,p: 60.176397 ,u: 0.991103286046685
	v: 74.660482 ,p: 82.284005 ,u: 1.1021092122068004
	v: 47.987304 ,p: 58.500931 ,u: 1.219091845627
	v: 51.353496 ,p: 71.050142 ,u: 1.38355024554
	v: 44.569231 ,p: 98.852504 ,u: 2.21795399611
	v: 1.498459 ,p: 30.399487 ,u: 20.28716634889
	v: 0.466933 ,p: 11.908322 ,u: 25.50327777219

BEST RESULTS: (OPTIMUM SOLUTION IS V_{MAX} = 10000)	**Problem items: n = 23 , KP8 (from the paper)**
0 ,0 ,0 ,1 ,0 ,1 ,1 ,0 ,0 ,1 ,0 ,1 ,0 ,1 ,0 ,1 ,1 ,1 ,1 ,0 ,1 ,0 ,0 ,V_T: 9732 ,P_T: 9724.0 *BEST RESULT	v: 959.0 ,p: 857.0 ,u: .89

0 ,1 ,1 ,1 ,1 ,0 ,0 ,0 ,0 ,1 ,1 ,0 ,0 ,0 ,0 ,1 ,1 ,1 ,0 ,0 ,1 ,1 ,0 ,VT: 9701 ,PT: 9705.0	v: 961.0 ,p: 959.0 ,u: 1.0
1 ,0 ,0 ,0 ,1 ,1 ,0 ,1 ,1 ,1 ,1 ,1 ,1 ,0 ,0 ,0 ,0 ,0 ,1 ,0 ,1 ,1 ,0 ,VT: 9681 ,PT: 9587.0	v: 963.0 ,p: 961.0 ,u: 1.0
0 ,0 ,1 ,1 ,1 ,0 ,0 ,1 ,0 ,1 ,0 ,1 ,1 ,0 ,1 ,0 ,0 ,1 ,1 ,0 ,0 ,0 ,1 ,VT: 9687 ,PT: 9696.0	v: 964.0 ,p: 962.0 ,u: 1.0
1 ,0 ,0 ,1 ,0 ,1 ,0 ,1 ,0 ,1 ,1 ,1 ,1 ,1 ,1 ,0 ,0 ,0 ,0 ,0 ,1 ,0 ,0 ,VT: 9719 ,PT: 9609.0	v: 483.0 ,p: 482.0 ,u: 1.0
0 ,1 ,1 ,1 ,1 ,0 ,0 ,0 ,0 ,1 ,1 ,0 ,0 ,0 ,0 ,1 ,1 ,1 ,0 ,0 ,1 ,1 ,0 ,VT: 9701 ,PT: 9705.0	v: 485.0 ,p: 484.0 ,u: 1.0
0 ,0 ,0 ,1 ,0 ,1 ,1 ,0 ,0 ,1 ,0 ,1 ,0 ,1 ,0 ,1 ,1 ,1 ,1 ,0 ,1 ,0 ,0 ,VT: 9732 ,PT: 9724.0	v: 486.0 ,p: 485.0 ,u: 1.0
0 ,0 ,0 ,1 ,1 ,0 ,1 ,0 ,1 ,1 ,0 ,0 ,1 ,1 ,1 ,0 ,1 ,1 ,0 ,1 ,0 ,0 ,1 ,VT: 9720 ,PT: 9736.0	v: 972.0 ,p: 970.0 ,u: 1.0
0 ,1 ,1 ,0 ,0 ,1 ,0 ,1 ,0 ,0 ,0 ,0 ,1 ,0 ,1 ,0 ,0 ,1 ,0 ,1 ,1 ,1 ,1 ,VT: 9673 ,PT: 9713.0	v: 486.0 ,p: 485.0 ,u: 1.0
	v: 972.0 ,p: 970.0 ,u: 1.0
	v: 488.0 ,p: 487.0 ,u: 1.0
	v: 976.0 ,p: 974.0 ,u: 1.0
	v: 978.0 ,p: 976.0 ,u: 1.0
	v: 979.0 ,p: 977.0 ,u: 1.0
	v: 980.0 ,p: 978.0 ,u: 1.0
	v: 981.0 ,p: 979.0 ,u: 1.0
	v: 982.0 ,p: 980.0 ,u: 1.0
	v: 983.0 ,p: 981.0 ,u: 1.0
	v: 958.0 ,p: 958.0 ,u: 1.0
	v: 969.0 ,p: 976.0 ,u: 1.01
	v: 966.0 ,p: 974.0 ,u: 1.01
	v: 958.0 ,p: 970.0 ,u: 1.01
	v: 458.0 ,p: 484.0 ,u: 1.06

REFERENCES:

- Xiangyong K. a,⇑, Liqun G. a, Haibin O. a, Steven Li b A simplified binary harmony search algorithm for large scale 0–1 knapsack problems, Expert Systems with Applications (2015)