Solving large-scale multidimensional knapsack problems with a new binary harmony search algorithm

Xiangyong Kong, Liqun Gao, Haibin Ouyang, Steven Li

PII: S0305-0548(15)00098-2
DOI: http://dx.doi.org/10.1016/j.cor.2015.04.018
Reference: CAOR3774

To appear in: *Computers & Operations Research*

Solving large-scale multidimensional knapsack problems with a new binary harmony search algorithm

Xiangyong Kong[a,*], Liqun Gao[a], Haibin Ouyang[a], Steven Li[b]

[a]*School of Information Science and Engineering, Northeastern University, Shenyang 110004, China*
[b]*Graduate School of Business and Law, RMIT University, Melbourne 3000, Australia*

Abstract

Harmony search (HS) is a meta-heuristic method that has been applied widely to continuous optimization problems. In this study, a new binary coded version of HS, named NBHS, is developed for solving large-scale multidimensional knapsack problem (MKP). In the proposed method, focus is given to the probability distribution rather than the exact value of each decision variable and the concept of mean harmony is introduced in the memory consideration. Unlike the existing HS variants which require specifications of parameters such as the pitch adjustment rate and step bandwidth, an ingenious pitch adjustment scheme without parameter specification is executed in the proposed HS according to the difference between two randomly selected harmonies stored in the harmony memory to generate a new candidate harmony. Moreover, to guarantee the availability of harmonies in the harmony memory, a simple but effective repair operator derived from specific heuristic knowledge of the MKP is embedded in the proposed method. Finally, extensive numerical simulations are conducted on two sets of large-scale benchmark problems, and the results reveal that the proposed method is robust and effective for solving the multidimensional knapsack problems with large dimension sizes.

Keywords: Harmony search, Multidimensional knapsack problems, Probability distribution, Ingenious pitch adjustment scheme, Repair operator

1. Introduction

Combinatorial optimization plays a very important role in operational research, discrete mathematics and computer science. The 0-1 knapsack problem is a well-known type of combinatorial optimization problem and it can represent the factory location problem, the production scheduling problem, the assignment problem, or the reliability problem etc. It derives its name from the problem faced by someone who is constrained by a special knapsack with many limitations but wish to pack it with the most valuable items.

The multidimensional knapsack problem (commonly known as the MKP) is generalized from the standard 0-1 knapsack problem. The MKP is a subset selection process of given items with specific profit and resource occupation to fulfill a knapsack without exceeding multidimensional resource capacities. Those items are appropriately chosen to make the cumulative profit packed in the knapsack as large as possible.

Mathematically, an m-dimensional knapsack problem with n items in its standard form can be described as:

$$\begin{aligned} &\text{Max} \;\; f(\mathbf{x}) = \sum_{i=1}^{n} p_i \cdot x_i \\ &s.t. \;\; \sum_{i=1}^{n} r_{i,j} \cdot x_i \leqslant R_{max,j}, j = 1,2,...,m \\ &\qquad x_i \in \{0,1\}, i = 1,2,...n \end{aligned} \tag{1}$$

where each item i (i=1,2,...,n) has a profit value p_i and consumes an amount $r_{i,j}$ for the j-th (j=1,2,...,m) resource. The objective is to maximize the total profits of all the items in a subset while the j-th resource occupation of the subset must be less than its corresponding resource capacity $R_{max,j}$. The value of x_i representing the state of the item i in the knapsack is restricted to be either 0 or 1. If the item i is put into the knapsack, the value of x_i is set to be 1, otherwise, 0. Each item may be chosen at most once and cannot be placed in the knapsack partly. Without loss of generality, it can be assumed that the above parameters in the MKP are non-negative integers and the following constraints, as defined by Eq.(2), must be satisfied.

$$r_{i,j} \leqslant R_{max,j}, \sum_{i=1}^{n} r_{i,j} \geqslant R_{max,j}, j = 1,2,...,m \tag{2}$$

Besides the number of constraints and the number of variables, there is another important parameter in the MKP which determines the resource capacities $R_{max,j}$. The slackness ratio S_j (j=1,2,...,m) introduced by Zanakis [1] defined as Eq.(3) is adopted here to generate different levels of the resource capacities.

$$S_j = \frac{R_{max,j}}{\sum_{i=1}^{n} r_{i,j}}, j = 1,2,...,m \tag{3}$$

Since the MKP is a well-known NP-hard problem which arises various engineering fields, many efficient, exact and approximate algorithms have been developed for obtaining optimal or near-optimal solutions. Early exact methods including dynamic programming [2], hybrid dynamic programming

[*]Corresponding author. Tel.: +86 2483678562.
Email address: `kxy2006@126.com` (Xiangyong Kong)

method [3] and branch and bound algorithm [4, 5] can be applied only to some small-scaled instances in an acceptable computation time. They become insufficient in solving large-scale problems because the computation complexity is rather high and the resulting computational effort and memory requirement can be tremendous. With the development of computational intelligence in recently years, more and more researchers focus on heuristic and meta-heuristic search methods for finding a high quality sub-optimal solution instead of the optimal solution. However, the meta-heuristics are not guaranteed to find the optimum. Relevant algorithms include genetic algorithm [6], simulated annealing [7], tabu search [8], ant colony optimization [9], particle swarm optimization [10], and so on. Most of the above algorithms require solving the linear programming relaxation of the original MKP which becomes difficult even infeasible as the dimension increased. Like the exact methods, these methods fail to be effective and efficient for solving large-scale MKPs. Very recently, an estimation of distribution algorithm based hybrid algorithm named HEDA has been proposed by Wang et al. [11] to cope with large-scale problems by using a new repair operator.

Among numerous heuristic and meta-heuristic methods proposed in the last two decades, the harmony search (HS) algorithm, a simple but powerful stochastic search technique for solving global optimization problems proposed by Geem et al. [12], is worth mentioning. Inspired by the musician attuning such as during rock music improvisation, the harmony search algorithm imitates the improvisation processes to find a perfect pleasing harmony from an aesthetic point of view which is similar to the seeking process in optimization to get a global optimal solution evaluated by an objective function. For an optimization problem, a musical harmony in the HS algorithm can be seen as the variable vector and the best harmony achieved in the end is analogous to the global optimum.

Several characteristics and advantages of HS have been discussed in [13, 14, 15]. The key differences between HS and other meta-heuristics, such as genetic algorithm (GA), particle swarm optimization (PSO) and differential evolution (DE), include: (1) HS enables each existing harmony vector to participate in the generation of a new solution vector. This increases the flexibility and helps to produce better solutions. While GA only considers two selected individuals as the parent vectors, PSO adjusts the trajectory of each particle simply toward its personal best location and the fittest position of the entire swarm found so far, and DE combines just three distinct individuals to form an offspring vector. (2) HS determines the value of each decision variable via improvisation independently. This is in contrast to PSO and DE in which a vector is adjusted according to a fixed rule. (3) A single solution vector is obtained after considering all of the existing vectors in HS, whereas GA, PSO and DE allow the simultaneous production of multiple solutions. (4) The newly generated solution is checked with the worst harmony among the existing ones in the harmony memory. Unlike HS, the offspring solutions compete with their corresponding parent individuals in many other meta-heuristics including GA, PSO and DE.

Owing to its advantage that there are fewer mathematical parameters to adapt than other meta-heuristic algorithms, the HS algorithm has been successfully applied to complex real-world optimization problems including parameter identification [16, 17], structural optimization [18, 19], reliability problems [20, 21], dispatch problems [22, 23], scheduling problem [24], unit commitment problems [25], classification problems [26], network reconfiguration [27], etc.

Most of the existing variants of HS mentioned above are focusing on optimization problems in continuous space. Until now, only a few papers concentrate on discrete problems and binary problems. Geem [28] introduced a binary coded HS without pitch adjustment operator to tackle a discrete problem, i.e., the water pump switching problem. Greblicki and Kotowski [29] analyzed the defect on the search ability degraded by discarding of pitch adjustment operator from the theoretical and experimental results. To circumvent the above weakness, a new pitch adjustment operation was presented by Wang at al. [30] and a novel discrete binary HS algorithm was developed to solve the discrete problems effectively. More recently, a scalable adaptive strategy was developed by Wang at al. [31] in an improved adaptive binary harmony search (ABHS) algorithm. Experimental results on the benchmark functions and 0-1 knapsack problems have demonstrated that it is beneficial to enhance the search ability and robustness in order to solve the binary-coded problems more effectively. Owing to its outstanding performance, ABHS was extended to search the optimal parameters of the fuzzy controller for improving the control performance with the guaranteed stability afforded by LFC [32].

Unlike the previous binary coded HS variants, several real-coded methods have also devoted to solve the discrete problems through specific implementation of transformation from real variables to actual discrete decision values. The most direct and commonly used strategy is to replace a real number with the nearest integer which corresponds to a permissible discrete decision value. Based on this observation, a novel global harmony search algorithm (NGHS) derived from the swarm intelligence of particle swarm was developed by Zou et al. [33] for solving the 0-1 knapsack problems. NGHS introduced a new position updating scheme and genetic mutation strategy to replace the harmony memory consideration and pitch adjusting in the classical HS. Similar to NGHS, a social harmony search algorithm model was presented by Kaveh and Ahangaran [34] for the cost optimization of composite floor system with discrete variables. Besides the conversion of actual discrete decision values, special improvisation operators are also designed to enable the search executed directly in the discrete domain. For example, the new pitch adjustment with neighboring values introduced by Lee et al. [35] helps HS to optimize the structures with discrete-sized members. With the assistance of job-permutation-based representation, Gao et al. [36] employed a novel pitch adjustment rule to produce a feasible solution such that HS is effective for solving the no-wait flow shop scheduling problems with total flow time criteria. Note that a novel partial stochastic derivative [37] was defined for discrete-valued functions and embedded in harmony search to find the optimal solutions for various science and engineering problems. However, the performances of the existing discrete HS variants are not satisfactory. In other words, the research on HS algorithms for discrete problems is still at its infancy.

To the best of the authors' knowledge, there is only one variant of HS method for solving the MKP. That is, Quantum Inspired Harmony Search Algorithm (QIHS) proposed by Layeb [38] which allows successful application of some quantum inspired operators such as measurement and interference in basic harmony search algorithm. Although Layeb demonstrated the feasibility and the effectiveness of the QIHS based on experimental studies with different types of instances and problem sizes, it is not evident enough to claim that the proposed QIHS algorithm performs better than others mentioned in the paper. The numerical experimentation is too limited and the choice of the compared algorithms is not well performed. Besides, it can be seen from the results reported in the paper that QIHS fails to find the optimal solutions of large-scale MKPs and it can only solve several instances with dimensions smaller than 50. Meanwhile, compared to those operators in original HS, the quantum operations introduced in QIHS are too complex and difficult to implement.

This paper aims to fill the gap in the literature by proposing a simple new binary harmony search (NBHS) algorithm for solving the multidimensional knapsack problems. Taking account of the characteristics of discrete problems, the framework of HS is restructured in NBHS. NBHS focuses on the probability distribution rather than the exact value of each decision variable and introduces the concept of mean harmony in the harmony memory consideration. The mean harmony is obtained based on the probability distribution of 0 and 1 and thus it contains the potential information about the objective space of all harmonies stored in the harmony memory. Unlike the existing HS variants, the values of the new candidate harmony vector are not randomly inherited from the historical values stored in the harmony memory. Instead they take the mean harmony with the probability of harmony memory consideration rate. Moreover, the pitch adjustment in NBHS is carried out depending on the difference between two randomly selected harmonies stored in the harmony memory. To be more precise, it depends on the probability distribution of 0 and 1 instead of any specified parameters including the pitch adjustment rate and step bandwidth. Besides, a measurement method named average relative resource occupation is proposed to evaluate the chance of each item to be packed in the knapsack. The measurement embedded in the NBHS algorithm serves as the specific heuristic information to guide an adaptive local search to repair the infeasible harmonies in the harmony memory and enhances the exploitation ability and convergence speed simultaneously. The realization of the repair operator is simple and there is no need to solve the linear programming relaxation of the original MKP which is difficult or even impossible when the problem scale is large. In order to evaluate its viability and effectiveness, NBHS is tested on two sets of various selected large-scale 0-1 MKP instances. Comparisons with the reported results of various other recent approaches are also performed. The experimental results of extensive numerical simulations demonstrate the superiority of the proposed NBHS algorithm. In general, NBHS is suitable for solving the multidimensional knapsack problems with large dimension sizes owing to its robustness and ease of implementation.

The remainder of this paper is organized as follows. In Section 2, a short overview of the classical HS algorithm is provided to ensure that this paper is self-contained. Section 3 describes the proposed new binary harmony search algorithm in detail. Numerical experiments and a series of comparisons are conducted in Section 4 to test the optimization performance of the proposed NBHS algorithm for large-scale 0-1 MKP instances. Finally, Section 5 gives the concluding remarks and comments for further research.

2. Harmony search algorithm

In this section, we set up the research framework and clarify the notation and terminologies used throughout the paper. Without loss of generality, a general optimization problem with n parameters to be optimized is usually represented as a nonlinear programming problem of the following form (in the maximization sense):

$$\begin{aligned} \text{Max} \;\; & f(\mathbf{x}) \\ s.t. \;\; & g_i(\mathbf{x}) < 0, \; i = 1,2,...,p \\ & h_j(\mathbf{x}) = 0, j = 1,2,...,q \end{aligned} \tag{4}$$

where $\mathbf{x} = (x_1, x_2, ..., x_n) \in \Omega \subseteq S$, S is an n-dimensional rectangular space in $\Re^n$ defined by the following parametric constraints: $x_{i,min} \leqslant x_i \leqslant x_{i,max}, i = 1,2,...,n$. The feasible region $\Omega \subseteq S$ is limited by a set of p inequality constraints and q equality constraints. The objective function $f(\mathbf{x})$ does not need to be continuous but it must be bounded. A point $\mathbf{x}$ is called a feasible solution if $\mathbf{x} \in \Omega$ and an infeasible solution, otherwise.

The harmony search algorithm imitates the improvisation processes. Each solution called a "harmony" here is represented by an n-dimension real vector. Initially all of the harmony vectors generated randomly in the variable space are stored in the harmony memory (HM). Then there are three basic phases, including memory consideration, pitch adjustment and random re-initialization, to generate a new candidate harmony from the whole HM. The worst harmony vector in the HM is updated if the new candidate harmony is better. The above improvisation process continues to be executed until the maximal number of improvisations has been accomplished or a certain stopping criterion is met.

The detailed optimization procedure of the classical HS algorithm consists of following steps [12].

Step 1: Initialize the optimization problem and algorithm parameters.

The HS algorithm parameters are specified in this step. They are the harmony memory size (HMS); harmony memory considering rate (HMCR); pitch adjusting rate (PAR); pitch adjustment step (bw); and the maximal number of improvisations (NI), or termination criterion.

Step 2: Initialize the harmony memory.

The initial harmony memory is generated from a uniform distribution in the ranges $[x_{i,min}, x_{i,max}], i = 1,2,...,n$, showed as follows:

$$\mathrm{HM} = \begin{bmatrix} x_1^1 & x_2^1 & \dots & x_n^1 & f(\mathbf{x}^1) \\ x_1^2 & x_2^2 & \dots & x_n^2 & f(\mathbf{x}^2) \\ \dots & \dots & \dots & \dots & \dots \\ x_1^{\mathrm{HMS}} & x_2^{\mathrm{HMS}} & \dots & x_n^{\mathrm{HMS}} & f(\mathbf{x}^{\mathrm{HMS}}) \end{bmatrix} \quad (5)$$

where n represents the dimension of solutions, i.e. the length of harmony vectors; the number of harmonies stored in the HM is denoted as HMS; $f(\cdot)$ is the objective function.

Step 3: Improvise a new candidate harmony from the HM.

Generating a new harmony is called improvisation. The new harmony vector $\mathbf{x}^{new}$ is determined by three rules: memory consideration, pitch adjustment and random selection. The procedure works as follows:

```
for $i$=1 to $n$ do
    if $rand \leqslant$ HMCR
        $x_i^{new} = x_i^r, r \in \{1, 2, \dots, \mathrm{HMS}\}$
        % memory consideration
        if $rand \leqslant$ PAR then
            $x_i^{new} = x_i^{new} \pm rand \cdot \mathrm{bw}$
            % pitch adjustment
        end if
    else
        $x_i^{new} = x_{i,min} + rand \cdot (x_{i,max} - x_{i,min})$
        % random selection
    end if
end for
```

where $rand$ is a uniform random number between 0 and 1.

Step 4: Update the harmony memory.

If the fitness of the improvised harmony vector $\mathbf{x}^{new}$ is better than that of the worst harmony, replace the worst harmony in the HM with $\mathbf{x}^{new}$.

Step 5: Check the stopping criterion.

If the maximal iteration number NI is completed or the termination criterion is satisfied, terminate the computation and return the best harmony vector $\mathbf{x}^{best}$ in the HM. Otherwise, go back to Step 3 and the improvisation process is repeated.

More details on the HS algorithm can be found in [39].

3. A new binary harmony search (NBHS) algorithm for the MKP

This section presents a new binary harmony search algorithm (NBHS) for solving large-scale multidimensional knapsack problems more effectively. Different from the classical HS designed for problems in continuous space, the framework of HS in NBHS is adjusted corresponding to the characteristics of discrete problems.

The proposed method is different from the classical HS in the following four aspects. Firstly, compared to the float coding method used in classical HS, it is more appropriate for the variables to be coded in binary scheme for binary optimization problems in NBHS. Secondly, the probability distribution of each decision variable is the focus instead of the exact value and the concept of mean harmony is introduced in the memory consideration which takes full advantage of the preferable information in the HM. A dynamic parameter adjustment scheme of HMCR is also presented in the memory consideration, which can dynamically update HMCR to better adapt to the evolution of the search process. Thirdly, an ingenious pitch adjustment scheme is executed in NBHS according to the difference between two randomly selected harmonies stored in the HM to generate a new candidate harmony without any specified parameters including PAR and bw utilized in other HS variants. Finally, a new simple but effective repair operator derived from specific knowledge of the MKP is introduced to embed in the NBHS algorithm to enhance the exploitation ability and convergence speed. Through the new repair operator, it can be guaranteed that all harmonies stored in the HM are feasible. The details of the NBHS algorithm are given below.

3.1. *Representation of the harmony*

Compared to the float coding method used in continuous space, the binary coded scheme is more appropriate for the variables in binary optimization problems. In NBHS, harmonies are represented as binary strings, and the harmony memory is initialized with random binary numbers generated by the Bernoulli stochastic process. Each element of the harmony vector is chosen from $\{0, 1\}$.

3.2. *Novel probability distribution based harmony memory consideration rule*

In the classical HS algorithm, generating a new candidate harmony is called improvisation like the musician searching for a better state of harmony. The improvisation consists of three rules: memory consideration, pitch adjustment and random selection. Memory consideration is defined as a process of picking out a value stored in the HM rather than randomly choosing a feasible value not limited to the HM. Memory consideration is utilized to generate a new candidate harmony with a probability of HMCR and the candidate value is chosen from the range of the variable space randomly relying on the complementation of HMCR. HMCR can take any value between 0 and 1. Note that each component obtained by the memory consideration comes from the previous values stored in the HM and it is further determined to be pitch adjusted or not.

In the existing HS variants, only one specific harmony, which is neither the best optimal harmony nor a randomly selected harmony, is considered and utilized to generate a new candidate harmony. The purpose of memory consideration is to obtain the potential and general location information about the optimum in the variable space through retaining the historical location information, i.e., the variable values of the existing harmonies stored in the HM rather than searching randomly in the variable space. In continuous variable space, the location information contained in one harmony can sometimes guide the new candidate harmony searching around the global optimum and lead to it closer and closer. The exact variable value is meaningful and determines the location information contained in one harmony. Therefore, memory consideration on one harmony in continuous variable space is reasonable and effective. However, this strategy becomes unreasonable and useless in 0-1 space. For a 0-1 optimization problem, there are only two values for each variable, that is, 0 and 1. There is no need to judge the exact value of each variable. The choice for each variable is limited to either 1 or 0, the same as each variable in the global optimum similarly. It is difficult to decide one certain variable value of the global optimum to be 0 or 1 from one arbitrary historical harmony. So the exact variable values in 0-1 optimization problems are meaningless and one single harmony can not provide effective potential and general location information about the global optimum.

From another perspective, solving the 0-1 optimization problem is also to decide whether each variable locates in 1 or not (0) in practice. For a given variable, if the corresponding value of one single harmony is 1, it is unconvincing that the corresponding value of the global optimum is 1. However, if 1 is selected as the value for this given variable for most of the harmonies stored in the HM, it is possible that this corresponding variable of the global optimum or better harmony locates in 1 to a certain extent. Similarly, if the great majority of existing harmonies choose 0 as the value for this given variable, the value for this given variable in the global optimum is more likely to be 0. Thus, lots of harmonies rather than one single harmony stored in the HM would lead to generate a better harmony through the probability distribution of each variable value (0 and 1 here). The greater probability one specific variable value emerges in harmonies, the greater probability this variable value is chosen as the corresponding value in the new candidate harmony. Consideration on many harmonies rather than one single harmony employed in continuous space is more reasonable and effective for 0-1 optimization problems.

Based on the above analysis, the probability distribution of each decision variable is the focus instead of exact value and the concept of mean harmony is introduced in the memory consideration which takes full advantage of the preferable information in the HM. On behalf of all harmonies stored in the HM, the mean harmony is a shadow harmony obtained depending on the probability distribution of each decision variable which contains the potential and general location information about the global optimum.

For an n-dimensional 0-1 optimization problem and a given harmony memory with HMS harmonies, the probability vector $P^1 = (P_1^1, P_2^1, \ldots, P_n^1)$ is determined as the probability of harmonies in which 1 is severed as the variable value in each dimension and it is updated according to the following equation:

$$P_i^1 = \frac{n_i^1}{n_i^1 + n_i^0}, i = 1, 2, \ldots, n \tag{6}$$

where n_i^1 and n_i^0 are the numbers of harmonies in which 1 and 0 are the variable value in the i-th dimension, respectively.

Then the mean harmony MH=(MH$_1$, MH$_2$,..., MH$_n$) is constructed as follows:

$$\mathrm{MH}_i = \begin{cases} 1, & if\ \ P_i^1 \geqslant 0.5 \\ 0, & otherwise \end{cases} ; i = 1, 2, \ldots, n \tag{7}$$

where P_i^1 is the probability of harmonies in which 1 is severed as the variable value in the i-th dimension.

In each generation, the mean harmony MH instead of one selected harmony in the HM is considered in the harmony memory consideration of NBHS. This method provides more reasonable and effective information to generate a better harmony than the existing HS variants. Imitating the classical HS, the probability distribution based harmony memory consideration rule is also carried out with the probability HMCR while the probability (1-HMCR) is the rate of stochastically choosing a feasible value from 0 and 1.

3.3. *Ingenious pitch adjustment scheme without any parameter*

Note that the pitch adjustment operator is important and must be contained in the improvisation processes. In the classical HS, the pitch adjustment operator depends on the pitch adjusting rate PAR and pitch adjustment step bw. Since there are just two variable values 0 and 1 to be decided in the MKP, the pitch adjustment step bw is removed from NBHS. The pitch adjusting rate PAR determines whether the pitch adjustment is employed on the new candidate harmony and this has substantial influence on the quality of the final solution. The optimization ability of the HS algorithms partly relies on the parameter setting of PAR. It is important to choose an appropriate value for PAR. Note that various adaptive methods of PAR have been proposed for the HS variants such as the linear increment and the linear decrease, nonlinear increment and nonlinear decrease, random increment etc. and some of them are conflicting with each other. However, it is difficult to determine the best choice for diverse HS algorithms. Meanwhile, the characteristics of the 0-1 optimization problems differ from those of problems in continuous space. Besides, most of the existing adaptive methods are determined regardless of the information containing in the HM. Therefore the tuning method of PAR must be changed to suit the MKPs.

To overcome the difficulty in choosing PAR values, an ingenious pitch adjustment scheme without any parameter is introduced according to the probability distribution of the variable values 0 and 1. It should be noted that the new candidate harmony is improvised through the memory consideration and pitch adjustment in each dimension. For the i-th dimension, two distinct random harmonies $\mathbf{x}^{r1}$ and $\mathbf{x}^{r2}$ are selected and the pitch adjustment scheme is executed if and only if the variables in this dimension of those two harmonies are not equal to each other, i.e., $x_i^{r1} \neq x_i^{r2}$. The pitch adjustment operator is a standard NOT gate applied for the MKP in NBHS because there is only one adjacent value from its structural neighborhood. The new ingenious pitch adjustment scheme depends on the difference between the harmonies stored in the HM. In other words, the probability distribution of the variable values 0 and 1 rather than a specific value PAR determines the accomplishment of the pitch adjustment operator in NBHS. The larger the difference proportion between the two randomly selected harmonies is, the more possibility the pitch adjustment happens.

The influence of the probability distribution on the likehood of the pitch adjustment is depicted in Fig. 1. Here P_i^1 represents the probability of harmonies in which 1 is the variable value in the i-th dimension. P_i^h stands for the likehood of the pitch adjustment.

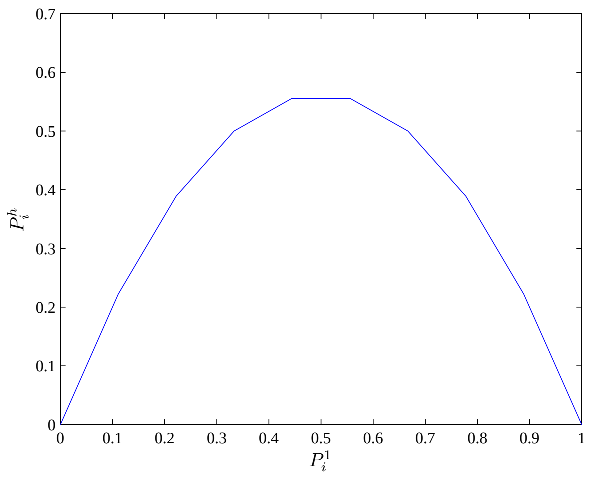

Figure 1: Influence of the probability distribution on the likelihood of the pitch adjustment

It can be seen from Fig. 1 that the more difference between P_i^1 and 1-P_i^1 is, the smaller the P_i^h is. The P_i^h achieves the largest value under the condition that P_i^1 is nearest to 1-P_i^1. That is, the more similar the probability distributions of 0 and 1 are, the larger the likelihood of the pitch adjustment is. On the contrary, if the probability distribution of one is significantly more than the other, the pitch adjustment is seldom carried out in the provision. Combining with the mean harmony based memory consideration proposed in Section 3.2, this ingenious pitch adjustment scheme is consistent with the search process as expected. For a chosen dimension, if most of harmonies in the HM choose 0 (or 1) as their variable value, it is likely that the corresponding value of the global optimum is 0 (or 1). The likelihood of the pitch adjustment is small. This maintains the trend to speed up the convergence of the harmony memory to the real global optimum and enhance the diversity of the harmony memory to effectively avoid it stopping at very poor quality local optima. On the other hand, if 0 and 1 occur similarly frequent in the HM, it is unable to judge whether the corresponding value of the global optimum is 0 or 1. The likehood of the pitch adjustment becomes large and this means that global search is preferred to find a better harmony. Thus the ingenious improvisation scheme proposed in NBHS can appropriately balance the exploration ability and exploitation ability. That is, the capabilities of the global search and the local search have been fully used to the optimization.

Compared to the pitch adjustment methods emerged in existing HS variants, the pitch adjustment scheme proposed in NBHS not only avoids the selection of both PAR and bw parameter values but also considers the internal property of those harmonies stored in the HM. The new pitch adjustment rule can be realized without any parameter and can adaptively adjust the evolution of the search process according to the changes of the harmonies stored in the HM.

3.4. Repair operator based on new heuristic information

As is known, infeasible solutions are unserviceable no matter how high the qualities of their fitness values are for constrainted problems. Only the feasible solutions can delegate the feasible region in the defined variable space for constrainted problems and infeasible solutions mislead the search to be stagnated in the infeasible region. It should be noted that the feasibility of the harmonies is not taken into account in the generation of new candidate harmonies and some of them are likely to be infeasible because of violating some of corresponding resource capacities. Since the proposed NBHS algorithm is rooted in the probability distribution of variable values included in the harmonies stored in the HM, those infeasible harmonies can spoil the optimization ability of NBHS and lead to an unsatisfactory performance for binary optimization problems. To achieve better potential and general location information about the global optimum, it must be ensured that all the harmonies contained in the HM are feasible and a repair operator is thus required.

The simplest way to achieve this is by removing some items from the knapsack and setting the variable value of the corresponding item from 1 to 0. Various heuristic procedures are available to select and remove items to maximize the total profit. Among them, a greedy-like heuristic procedure proposed by Chu and Beasley [6] is the most common used and efficient repair technique which depends on the pseudo-utility ratios determined using the surrogate duality approach [40].

Given a general MKP as described by Eq.(1), the surrogate relaxation problem of the original MKP (denoted by SR-MKP) can be defined as below

$$\begin{aligned} &\text{Max} \;\; f(\mathbf{x}) = \sum_{i=1}^{n} p_i \cdot x_i \\ &s.t. \;\; \sum_{i=1}^{n} \Big(\sum_{j=1}^{m} w_j \cdot r_{i,j} \Big) \cdot x_i \leqslant \sum_{j=1}^{m} w_j \cdot R_{max,j} \\ &\qquad x_i \in \{0,1\}, i = 1,2,...n \end{aligned} \tag{8}$$

where $\mathbf{w} = [w_1, w_2, ..., w_m]$ is a set of weights (or surrogate multipliers here) of some positive real numbers.

Through the surrogate relaxation rule, the original MKP is reduced to a simple knapsack problem with one single surrogate constraint which is easy to deal with. Note that a single constraint knapsack problem is not strong NP-hard and the pseudo-utility ratio for each variable based on the surrogate constraint coefficient can be employed here. Clearly, the effectiveness of SR-MKP strongly depends on the selection of an optimal set of surrogate multipliers which determine the ability of the surrogate constraint to capture the aggregate weighted consumption level of resources for each variable. However, finding the optimal set of surrogate multipliers as weights for SR-MKP is a difficult task and several methods have been suggested to derive these weights. The simplest and reasonable way is to use the values of the dual variables obtained by solving the linear programming (LP) relaxation of the original MKP as the weights. Those weights stand for the weight of each constraint in the surrogate constraint and can also be considered as their shadow prices in the LP relaxation of the original MKP.

Since those weights are given, the pseudo-utility ratio for each variable can be computed as that in single constraint knapsack problem. The pseudo-utility ratios are defined as follows:

$$u_i = \frac{p_i}{\sum_{j=1}^{m} w_j \cdot r_{i,j}}, i = 1, 2, ..., n \tag{9}$$

where the pseudo-utility ratio of the i-th item is denoted by u_i and w_j represents the weight of the j-th constraint in the surrogate constraint. Then the pseudo-utility ratios are embedded to determine the inclusion or exclusion of each item in the infeasible solutions. An item with a higher pseudo-utility ratio will be more desirable for selection. A ranking is obtained according to the pseudo-utility ratios and the repair procedure is performed in two phases. The details of the greedy-like heuristic procedure are illustrated in [6] and [11].

It is clear that solving the LP relaxation of the original MKP is compulsive in the above greedy-like heuristic procedure. It is difficult or even impossible for large-scale problems. In order to circumvent this disadvantage, a new surrogate relaxation model and the definition of relative mean resource occupation is introduced to design a simple but effective repair operator to enhance the exploitation ability and convergence speed. The relative mean resource occupation based repair operator does not require solving the LP relaxation of the original MKP.

The new surrogate relaxation problem of the original MKP is redefined as Eq.(10) and the relative mean resource occupation of each item can be defined as Eq.(11) below.

$$\begin{aligned} \text{Max} \quad & f(\mathbf{x}) = \sum_{i=1}^{n} p_i \cdot x_i \\ s.t. \quad & \sum_{i=1}^{n} \Big(\sum_{j=1}^{m} \frac{r_{i,j}}{m \cdot R_{max,j}} \Big) \cdot x_i \leqslant 1 \\ & x_i \in \{0,1\}, i = 1, 2, ...n \end{aligned} \tag{10}$$

$$o_i = \frac{\sum_{j=1}^{m} \frac{r_{i,j}}{m \cdot R_{max,j}}}{p_i}, i = 1, 2, ..., n \tag{11}$$

where o_i denotes the relative mean resource occupation of the i-th item and it evaluates the relative resource occupation of each item to produce an unit profit. As indicated in Eq.(10) and (11), an item with a smaller relative mean resource occupation will be more desirable for selection intuitively. Inspired by this observation, a new repair operator is developed based on the relative mean resource occupations.

The items can be ranked in the increasing order according to their relative mean resource occupations. The first (last) item has the largest (least) probability to be packed into the knapsack for maximizing the total profit. The repair operator is performed in two stages: removing items and adding items. The stage of removing items seeks a new feasible solution around an infeasible solution by changing the variable values from one to zero. While the stage of adding items aims to raise the total profit of the new feasible solution as much as possible without exceeding any of the resource capacities. The second stage is opposite to the first one and changes the variable values from zero to one.

Specifically, the procedure for the relative mean resource occupations based repair operator is given as below.

Step 1: Let an infeasible harmony be denoted as $\mathbf{x} = [x_1, x_2, ..., x_n]$.

Step 2: Calculate the accumulated values (or constraint values) of each resource R_j (j=1,2,...,m) of the infeasible harmony as

$$R_j = \sum_{i=1}^{n} r_{i,j} \cdot x_i \tag{12}$$

Step 3: Obtain the corresponding sequence S based on the relative mean resource occupation of each item calculated using Eq.(11) in descending order.

Step 4: Remove the items as sequence S until no constraint value R_j exceeds its corresponding resource capacity $R_{max,j}$. The pseudo code of the removing process is given as follows.

```
J={1,2,...,m};
i=1;
While $i \leqslant n$ and $\exists j \in J, R_j > R_{max,j}$
    k=S(i);
    If $x_k$=1
        $x_k$=0;
        $R_j = R_j - r_{k,j}, \forall j \in J$
    End if
    i=i+1;
End while
```

Step 5: Add the items into the knapsack following the reverse order of S until at least one constraint value R_j violates the limitation of its corresponding resource capacity $R_{max,j}$. The pseudo code of the adding process is illustrated as below.

```
J={1,2,...,m};
$i = n$;
While $i \geqslant 1$ and $\forall j \in J, R_j \leqslant R_{max,j}$
    k=S(i);
    If $x_k$ =0
        $x_k$ =1;
        $R_j = R_j + r_{k,j}, \forall j \in J$
    End if
    $i = i - 1$;
End while
$x_k$=0;
```

Through this repair mechanism, it is clear that the infeasible harmony becomes to be legal and no longer violates any constraint. Since the repair operations in the above two stages including removing and inserting are all carried out based on the relative mean resource occupation, the knapsack can be filled up with as much profit as possible. Compared with the pseudo-utility ratio based greedy-like heuristic procedure proposed by Chu and Beasley, this newly proposed repair strategy is very easy to accomplish and there are no difficulty in applying it to solve any MKP with an arbitrary size.

3.5. Self-adaptive mechanism

Due to the elimination of the parameters PAR and bw in NBHS as described detailed in Section 3.3, there are just two parameters left: HMS and HMCR. HMS must be an odd number due to the introduction of the mean harmony and is set to be 9 for all experiments in this paper. The HMCR value which varies between 0 and 1 plays the most important role for the algorithm performance as it controls the balance between the capabilities of the global search and the local search. HMCR=0 means all candidate values are chosen from the range of the variable space randomly. HMCR=1 means there is no chance to choose a value from out-side the HM to improve the harmony.

It is vital to choose a proper adaptive method to improve the performance of the algorithm. Generally speaking, a large HMCR favors the local search. To enhance the exploration ability, HMCR value should be small in order to lead the search in the whole variable space. The best choice of HMCR value is normally in the interval [0.95, 1]. Moreover, various HMCR adaptive methods appear to ameliorate the performance and flexibility of the HS variants, including the linear increment, linear decrease, nonlinear increment, random increment etc. Most of them focus on the search process and base the HMCR value on the current and the maximum iteration number regardless of the problem dimension.

Given an n-dimensional problem, the expected number of elements chosen from the HM in the new candidate harmony is n·HMCR, while the expected number of components reinitialized randomly from the possible range of values relying on the complementation is n·(1-HMCR). For a low-dimensional problem, n·(1-HMCR) is small. However, for a large-scale problem with $n \geqslant 100$, the value n·(1-HMCR) can be so significant that too many randomly selected elements would destroy the optimization ability of the algorithm. In this case, it is not the search process but the problem dimension which has the biggest effect on the performance of the algorithm. Based on this observation, NBHS updates the HMCR value dynamically according to Eq.(13) below:

$$\mathrm{HMCR} = \left(1 - \frac{a - \ln n}{n}\right) + \frac{b}{n} \cdot \frac{k}{\mathrm{NI}} \tag{13}$$

where n represents the dimension of the problem to be solved. k and NI are the current and maximal iteration numbers, respectively. a and b are two constants recommended to 13 and 5, respectively, when the problem dimensionality is between 100 and 2,500.

Note that two terms are included on the right hand side of Eq.(13). The first term can ensure that certain elements are generated by stochastically choosing a feasible value in the feasible variable space. The second term linearly increases in the iteration number. It is worth noting that these two terms both adjust HMCR based on the dimension of the problem to be solved. As can be seen from Eq.(13), the larger the problem dimensionality is, the larger the HMCR value is and lower opportunity for the randomization operator to be selected for the generation of the new candidate harmony. In other words, the expected number of components reinitialized randomly from the possible range of values becomes smaller with the increase of the problem dimension and this can speed up the algorithm to converge around the global optimum. Once the problem dimension is given, HMCR utilized in practice is linearly increasing as search proceeds. In early stage, smaller HMCR value can effectively determine the capabilities of the global search to prevent the algorithm from trapping in a local optimum. Meanwhile, the algorithm would properly converge to the global optimum under a larger HMCR value in later stage. In such a way, this adaptive tuning method can balance the global search and the local search in the searching process.

3.6. Computational procedure of NBHS

With the above specific design, the computational procedure of the NBHS algorithm can be illustrated as follows.

Step 1: Set the parameters including HMS and NI.

Step 2: Initialize the HM using the Bernoulli stochastic process and repair the infeasible harmonies through the repair procedure as discussed in Section 3.4. Each harmony is evaluated according to the total profit of the items selected.

Step 3: Set the current generation k=1 and determine the HMCR value according to the problem dimension and the current generation.

Step 4: Construct the mean harmony MH=(MH$_1$, MH$_2$, ..., MH$_n$) as Eq.(6) and (7) according to the current harmony memory.

Step 5: Improvise a new candidate harmony $\mathbf{x}^{new}(x_1^{new}, x_2^{new}, ..., x_n^{new})$ as below and repair it if infeasible.

```
for $i$=1 to $n$ do
    if $rand \leqslant$ HMCR
        $x_i^{new}$=MH$_i$; % memory consideration
        $r1 \neq r2 \in \{1,2,...,\mathrm{HMS}\}$;
        if $x_i^{r1} \neq x_i^{r2}$
            $x_i^{new} = 1 - x_i^{new}$; % pitch adjustment
        endif
    else
        $x_i^{new} = \begin{cases} 0, & if\ \ rand \leqslant 0.5 \\ 1, & otherwise \end{cases}$ ;%re-initialization
    endif
endfor
```

Step 6: Replace the worst harmony in the HM with $\mathbf{x}^{new}$, if and only if $\mathbf{x}^{new}$ is better than the worst harmony.

Step 7: k=k+1. Repeat steps 4-6 in the improvisation process until NI new harmonies have been generated.

Step 8: Output the best harmony vector $\mathbf{x}^{best}$ in the HM as the optimal solution.

4. Experimental results and comparison analysis

In this section, the performance of NBHS on multidimensional knapsack problems is extensively investigated based on a large number of experimental studies. Two different sets of widely used benchmark instances are chosen to test the performance of NBHS. Test set 1 (named Mk_cb for short) covers 270 large size problems with $m \in \{5, 10, 30\}$ constraints and $n \in \{100, 250, 500\}$ items. They are divided into 9 groups according to 9 m-n combinations and each group contains 30 test problems. In each group, the first ten problems have a tightness ratio of 0.25, the second ten problems have a tightness ratio of 0.50 and the remaining ten problems have a tightness ratio of

0.75. Each group is named as Mk_cb.m.n-S_j in our numerical tests. The other set denoted as Mk_gk consists of 11 larger size problems with m=15 to 100 and n=100 to 2,500. These two sets of well-known benchmark instances can be freely obtained from OR-Library [1]. Two repair operators including the pseudo-utility ratio based greedy-like heuristic procedure proposed by Chu and Beasley and the newly proposed repair strategy in this paper are embedded into NBHS to repair the infeasible harmonies. Here NBHS with Chu and Beasley's repair procedure is denoted as NBHS1 and the other is signed as NBHS2 with our strategy. The maximum number of iteration is set to 100,000 and 30 independent runs are conducted per problem in all experiments, except otherwise clearly declared.

Since most of best feasible solution values of these instances are unknown, it is reasonable to compare with the results obtained from the values of the LP relaxation for these problems to evaluate the performance of other methods. The qualities of the solutions generated are measured using the percentage gap (or deviation) between the optimum x^{LP} of the LP-relaxed problem and them. The percentage gap is calculated as

$$gap(\%) = 100 \times \frac{f(\mathbf{x}^{LP}) - f(\mathbf{x})}{f(\mathbf{x}^{LP})} \tag{14}$$

where $f(\mathbf{x}^{LP})$ represents the optimal value of the LP relaxation of the initial MKP and $f(\mathbf{x})$ denotes the best objective value found by the approach to be measured. The smaller the percentage gap value is, the better performance the approach holds.

The proposed method is implemented in Matlab R2008b using a PC with Intel Core (TM) 2 Duad CPU Q9400 @2.66 GHz, 3.50 GB RAM and 32-bit Windows XP operating system. Subproblems in the referent optimization phase are solved using TOMLAB 7.9.

4.1. Parameter dependence in NBHS

NBHS favors an ingenious pitch adjustment scheme without any parameter and thus the pitch adjusting rate (PAR) and pitch adjustment step (bw) are not required. Since HMCR updates based on two constants, there are three key parameters to be set in the proposed NBHS algorithm: the harmony memory size (HMS) and two constants (a and b) which enables HMCR to dynamically adapt in terms of different problem dimensions and each moment of the optimization process. A detailed study is executed to investigate the influence of these three parameters on the performance of NBHS. Experiments are conducted on the first 5 instances with 10 constraints and 250 items from the Mk_cb.10.250-0.5 set. The algorithm is ended when the iteration is up to 100,000 and the average results for each function in 30 independent runs are recorded for comparison.

First, we investigate the impact of the parameter HMS by keeping the parameters for HMCR unchanged. The HMS value may affect the production of the mean harmony and then impact the convergence of the algorithm. A small value for HMS would mislead the selective probability of the two variable values due to the lack of sufficient information. On the other hand, if HMS is too large, the convergence may be delayed because of the oversized data. Therefore HMS should be determined to balance the accuracy and convergence. It should be noted that HMS should be an odd number such that the selective probabilities of the two variable values are always unequal to each other. If not, when the harmonies that hold the variable value 0 are as many as those hold the variable value 1, it is confusing for the mean harmony to decide which variable value to be. Since the HMCR value has not been investigated, we fix it on 0.96 according to previous empirical studies. A host of HMS values are tested in this paper and the mean fitness values in 30 runs are plotted in Fig. 2 and Fig. 3. Both figures are consistent with the above observations. Fig. 2 indicates that large HMS would destroy the convergence of NBHS drastically. Fig. 3 shows that too small HMS is also not suitable because of insufficient information. A value of 9 for HMS is better for 4 out of the mentioned 5 problems and 11 is better for the one left. Therefore, we set HMS to be 9 for NBHS1 and NBHS2 in this paper.

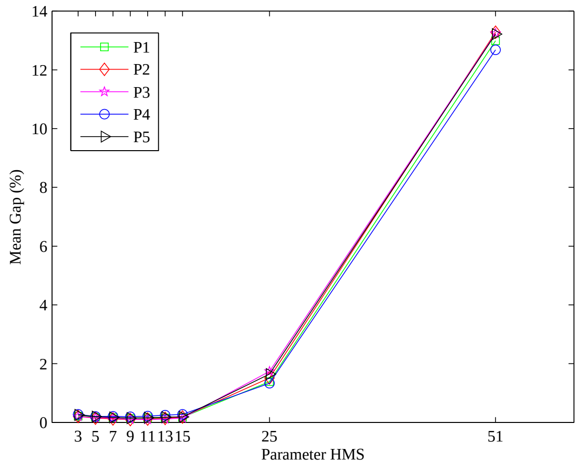

Figure 2: NBHS2 performance with respect to the harmony memory size HMS

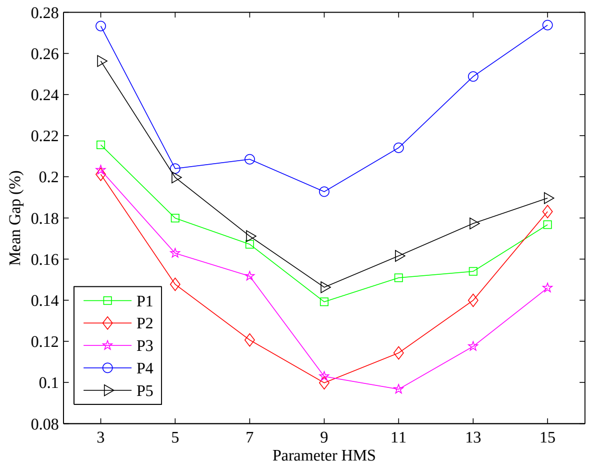

Figure 3: NBHS2 performance with respect to the harmony memory size HMS

[1]http://www.cs.nott.ac.uk/ jqd/mkp/index.html

Next, we focus on the impact of the two parameters a and b for HMCR. HMCR needs to keep the balance between the capabilities of global search and local search. Large HMCR values favor the search around the domain in which the harmony memory is located. On the other hand, the whole possible range of variable values is inclined to be explored with small HMCR values. For the 0-1 optimization problems, only 0 and 1 are acted as the variable values and thus the local search is much more crucial than the global search. From the above perspective, HMCR should be large because small values would result in fluctuation to destroy the search capacity of the algorithm. However, it does not mean that large HMCR values are always better. As an extreme case, when HMCR=1, there is no chance to choose a value from outside the HM to improve the harmony and the harmonies can be easily trapped in the region around the optima.

As can be seen from Eq.(13), a and b (thus) HMCR need to be adapted for different problems and at different stages of the optimization process. HMCR increases linearly over the generations under given values of a and b in this paper. The minimal HMCR value is determined by a and both parameters trigger the maximal value at the last iteration. Two constants are not independent of each other since HMCR is the harmony memory considering rate that should be less than or equal to 1. Therefore, a must be larger than $\ln(n)$ and b should be smaller than $a - \ln(n)$.

We set HMS=9 and b=0 and vary the parameter a to investigate its influence on the performance of the NBHS2 algorithm. Since the dimensions of MKP instances employed in this paper are smaller than 3,000, the minimal value of a is limited to 9 in the experiment. The obtained result for NBHS2 is shown in Fig. 4. On the whole, the mean gap values become larger as the parameter a increases. For a general configuration, the value between 11 and 15 is good for all of the test functions. Hence, the value of 13 is adopted as parameter a in NBHS.

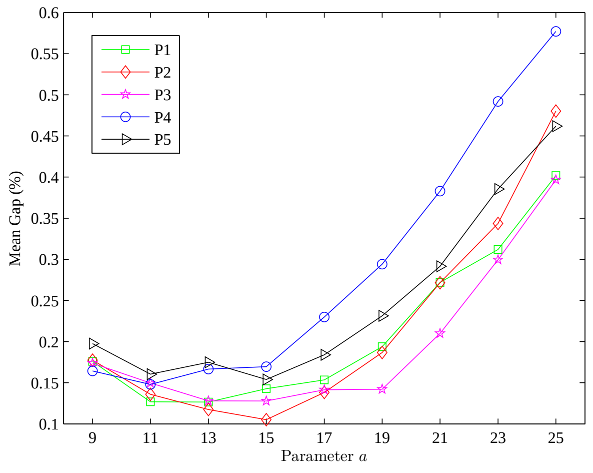

Figure 4: NBHS2 performance with respect to the parameter a

Finally, the impact of parameter b is studied with HMS=9 and a=13. Since b should be smaller than $a - \ln(n)$, the maximal value of a is limited to 5 in the following experiment. The obtained result for NBHS2 is shown in Fig. 5. There are no clear trends for most instances. As can be seen from Fig. 4 and Fig. 5, the parameter a has a major impact on the performance of NBHS2 according to the variations of mean gap values for different choices. Since HMCR is better to approach 1 for convergence in the later search process, it is beneficial to set b close to $a - \ln(n)$ as much as possible. From Fig. 5 and the above analysis, the value of 5 may be a good choice for parameter b.

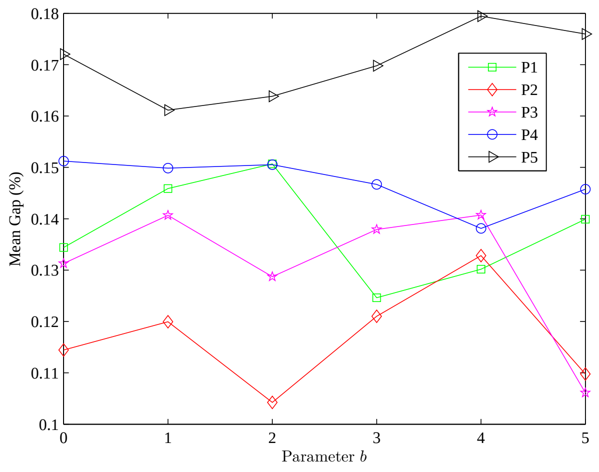

Figure 5: NBHS2 performance with respect to the parameter b

4.2. *Comparison of NBHS with QIHS*

Firstly, NBHS is compared with the quantum inspired harmony search algorithm named QIHS proposed in [38] to solve the MKPs. Experiments are conducted on 5 instances with 5 constraints and 100 items as well as 5 instances with 10 constraints and 100 items taken from the Mk_cb set. The objective values of the best solutions found by QIHS and NBHS are summarized in Table 1. As we can be seen, the performance of NBHS is clearly better than QIHS. QIHS fails in finding the optimal value in every instance and the second instance is the only instance of which optimal value can be located correctly when the repair operator is introduced into the QIHS method (QIHS-R). While NBHS1 is able to find the best solutions for 4 tests out of ten MKP instances and NBHS2 is successful in 5. On the other hand, results obtained by NBHS1 and NBHS2 are much better or at least equal to those by QIHS. In short, NBHS have demonstrated better performance compared to QIHS on solving the multidimensional knapsack problems.

4.3. *Comparison of NBHS with other methods on Test set 1*

In this subsection, NBHS is tested on Mk_cb set. The number of resource constraints, the number of decision variables, and the slackness ratios of the knapsack constraints are all selected as the alterable parameters to investigate the performance of NBHS for the MKPs with different characteristics. Various methods selected to make comparison with NBHS include PIR [41], GA [6], ADP-H2 [2], PECH [42], HDP-LBC [3], NR(P)

Table 1: Comparison of NBHS with QIHS

Problem	Best known	QIHS	QIHS-R	NBHS1	NBHS2
Mk_cb.5.100	24381	23915	24219	24381	24381
Mk_cb.5.100	24274	23817	24274	24274	24274
Mk_cb.5.100	23551	23210	23450	23551	23551
Mk_cb.5.100	23534	23308	23366	23484	23497
Mk_cb.5.100	23991	23020	23673	23966	23966
Mk_cb.10.100	23064	22306	23057	23057	23064
Mk_cb.10.100	22801	22610	22565	22753	22743
Mk_cb.10.100	22131	21594	21927	22131	22131
Mk_cb.10.100	22772	22407	22562	22763	22717
Mk_cb.10.100	22751	22342	22387	22697	22697

[43], CH [44], FPLS [44] and MLH [45], which are briefly described below.

PIR is a dual surrogate relaxation method with an efficient branch and bound code which significantly reduces the solution times compared to the commercial code. GA incorporates the standard genetic algorithm approach and a problem-specific knowledge based heuristic operator to obtain high-quality solutions for problems of various characteristics with a modest amount of computational effort. As an approximate dynamic programming approach, ADP-H2 approximates the value function using parametric and nonparametric methods and a base-heuristic. The proposed fixing heuristic can adaptively round the solution of the linear programming relaxation. Different from the existing greedy-like heuristics, PECH introduces a primal effective capacity based method to add decision variables to the solution in batches. The new greedy-like heuristic improves computational efficiency significantly and generates robust and near-optimal solutions especially for large-scale problems. HDP-LBC is constructed with a dynamic-programming algorithm (HDP) and so-called limited-branch-and-cut method (LBC), where HDP gives a better solution with a quite good processing time and LBC receives a better approximation of the optimal bound provided by HDP with less time than an exact method. With respect to the problem size reduction approach NR(P), Hill et al. use the dual variables to formulate a Lagrangian relaxation of the original problem and solve the estimated core problem to achieve a heuristic solution. NR(P) introduces an efficiency measure range approach to dynamic estimate the core problem size that diverges from the standard practice of a problem size percentage or fixed size core problem estimate. CH denotes a constructive method using Lagrange multipliers focusing on the goal of finding a good Lagrangian capacity and its corresponding Lagrange multiplier. Unlike other Lagrangian heuristics for discrete problems, the Feasibility-Pursuing Lagrangian Search (FPLS) algorithm primarily pursues finding feasible solutions rather than good upper bounds. Since it is not easy but vital to find the optimal Lagrange multipliers nearest to the capacity constraints for the MKP, a memetic algorithm (MLH) is applied to optimize the Lagrange multipliers.

The comparative results of 90 instances with 5, 10 and 30 constraints in Test set 1 are tabulated in Tables 2-4, respectively. The mean relative gap represents the mean percentage gap of the best found solution from the corresponding best-known value for each instance. It should be noted that each value in Tables 2-4 is an average over 10 instances. For measuring the improvement of NBHS on other methods, the degree of improvement (DI) is used.

DI is described as:

$$\mathrm{DI}(\%) = 100 \times \frac{\mathrm{Gap}_{\mathrm{other}} - \mathrm{Gap}_{\mathrm{NBHS}}}{\mathrm{Gap}_{\mathrm{other}} - \mathrm{Gap}_{\mathrm{best}}},$$

where $\mathrm{Gap}_{\mathrm{NBHS}}$ represents the average of mean relative gap obtained by NBHS and $\mathrm{Gap}_{\mathrm{other}}$ represents the average of mean relative gap obtained by other method. The best ideal gap is 0 and thus $\mathrm{Gap}_{\mathrm{best}}$is equal to 0. In the following experiments, the best results among the compared methods are highlighted in boldface.

Table 2 gives the mean relative gap values of 9 groups of problems with 5 constraints and different number of items and slackness ratios. For each number of n, the final line gives the average of the mean relative gap values over all values of slackness ratios. The third line from the bottom of Table 2 provides the overall average gap values over all categories of n.

As can be observed from Table 2, NBHS2 obtains the smallest mean relative gap values on problems in Mk_cb.5.100-0.5 and Mk_cb.5.100-0.75 while NBHS1 outperforms in other 7 groups of instances. Since the mean relative gap is a measure of how close the obtained solution is to the linear programming optimum, the smaller the mean relative gap value is, the better the solution is. Therefore, NBHS1 and NBHS2 achieve the better results compared to other methods on each instance according to the overall average gap values of each method in Table 2. PECH has the largest average gap value of 4.05% which is 45 times over the smallest average gap value of 0.09% obtained by NBHS1. The average gap value of NBHS2 is also less than half of that obtained by GA which ranks the third in performance among all the methods in this experiment. On the other hand, the distribution of the mean relative gap values of 9 groups of problems with 5 constraints is summarized in Fig. 6 for each method except PECH which is far inferior to the others. The box plot indicates that NBHS1 and NBHS2 have much stronger exploration ability and stability on finding better solutions.

From another point of view, if we consider the degree of improvement of NBHS1 and NBHS2 on other methods which have been given in the last two rows in Table 2, NBHS1 and NBHS2 are clearly better. The DI values of NBHS1 are located between 65.38% on GA and 97.78% on PECH while those for NBHS2 are 57.69% and 97.28% also on GA and PECH. This shows that PECH is the worst and GA is the best one except NBHS1 and NBHS2 in many comparison approaches. The corresponding average improvements made by the proposed NBHS1 and NBHS2 approaches are 77.63% and 72.66%.

Table 3 reports the mean relative gap values of 9 groups of problems with 10 constraints and different number of items and slackness ratios. For each number of n, the last row gives the average of the mean relative gap values over all values of slackness ratios. The third last row of Table 3 provides the overall average gap values over all categories of n.

As can be observed from Table 3, NBHS1 obtains the smallest mean relative gap values on each class of problems. The sec-

Table 2: Mean relative gap of each group of problems with 5 constraints

Instance			Mean relative gap							
m	*n*	S_j	PIR	GA	PECH	HDP-LBC	NR(P)	MLH	NBHS1	NBHS2
5	100	0.25	1.59	0.99	7.34	-	0.94	2.32	**0.19**	0.23
		0.50	0.77	0.45	3.47	-	0.44	1.08	0.12	**0.11**
		0.75	0.48	0.32	2.02	-	0.22	0.72	0.09	**0.08**
		Average	0.95	0.59	4.28	0.57	0.53	1.37	**0.13**	0.14
5	250	0.25	0.53	0.23	7.12	-	0.46	0.92	**0.17**	0.19
		0.50	0.24	0.12	3.20	-	0.17	0.49	**0.07**	0.10
		0.75	0.16	0.08	1.77	-	0.10	0.25	**0.04**	0.06
		Average	0.31	0.14	4.03	0.16	0.24	0.56	**0.09**	0.12
5	500	0.25	0.22	0.09	6.41	-	0.15	0.48	**0.08**	0.13
		0.5	0.08	0.04	3.43	-	0.06	0.20	**0.04**	0.07
		0.75	0.06	0.03	1.70	-	0.03	0.14	**0.03**	0.04
		Average	0.12	0.05	3.85	0.07	0.08	0.27	**0.05**	0.08
Average			0.46	0.26	4.05	0.27	0.28	0.73	**0.09**	0.11
DI of NBHS1 (%)			80.43	65.38	97.78	66.67	67.86	87.67	77.63	-
DI of NBHS2 (%)			76.09	57.69	97.28	59.26	60.71	84.93	-	72.66

Table 3: Mean relative gap of each group of problems with 10 constraints

Instance			Mean relative gap							
m	*n*	S_j	PIR	GA	PECH	HDP-LBC	NR(P)	MLH	NBHS1	NBHS2
10	100	0.25	3.43	1.56	8.20	-	2.05	2.27	**0.46**	0.50
		0.50	1.84	0.79	3.73	-	0.81	1.14	**0.25**	0.30
		0.75	1.06	0.48	1.82	-	0.44	0.69	**0.10**	0.19
		Average	2.11	0.94	4.58	0.95	1.10	1.37	**0.27**	0.33
10	250	0.25	1.07	0.51	5.86	-	0.88	0.88	**0.25**	0.37
		0.50	0.57	0.25	2.56	-	0.39	0.45	**0.12**	0.20
		0.75	0.33	0.15	1.52	-	0.19	0.24	**0.08**	0.14
		Average	0.66	0.30	3.31	0.32	0.48	0.52	**0.15**	0.24
10	500	0.25	0.52	0.24	5.06	-	0.34	0.50	**0.13**	0.25
		0.50	0.22	0.11	2.45	-	0.14	0.24	**0.07**	0.15
		0.75	0.14	0.07	1.23	-	0.10	0.14	**0.05**	0.09
		Average	0.29	0.14	2.92	0.16	0.19	0.29	**0.08**	0.17
Average			0.77	0.46	3.6	0.48	0.59	0.73	**0.17**	0.24
DI of NBHS1 (%)			77.92	63.04	95.28	64.58	71.19	76.71	74.79	-
DI of NBHS2 (%)			70.13	50.00	93.61	52.08	61.02	68.49	-	65.89

Table 4: Mean relative gap of each group of problems with 30 constraints

Instance			Mean relative gap										
m	*n*	S_j	PIR	GA	ADP-H2	PECH	HDP-LBC	NR(P)	CH	FPLS	MLH	NBHS1	NBHS2
30	100	0.25	9.02	2.91	-	6.83	-	2.24	11.75	3.1	3.19	0.99	**0.94**
		0.5	3.51	1.34	-	3.19	-	1.32	7.51	1.39	1.43	**0.39**	0.40
		0.75	2.03	0.83	-	1.91	-	0.8	4.2	0.86	0.89	0.33	**0.25**
		Average	4.85	1.69	-	3.98	1.81	1.45	7.82	1.78	1.84	0.56	**0.53**
30	250	0.25	3.7	1.19	1.61	4.83	-	1.27	11.36	1.27	1.28	**0.55**	0.71
		0.5	1.53	0.53	0.69	2.08	-	0.75	6.71	0.57	0.59	**0.28**	0.36
		0.75	0.84	0.31	0.58	1.16	-	0.38	3.56	0.32	0.34	**0.21**	0.24
		Average	2.02	0.68	0.96	2.69	0.77	0.8	7.21	0.72	0.74	**0.35**	0.44
30	500	0.25	1.89	0.61	0.98	3.69	-	0.89	9.39	0.69	0.68	**0.29**	0.58
		0.5	0.73	0.26	0.43	1.7	-	0.36	6.17	0.3	0.29	**0.14**	0.30
		0.75	0.48	0.17	0.29	0.88	-	0.23	3.34	0.19	0.19	**0.11**	0.20
		Average	1.03	0.35	0.57	2.09	0.42	0.49	6.3	0.39	0.39	**0.18**	0.36
Average			2.63	0.91	0.77	2.92	1	0.91	7.11	0.96	0.99	**0.36**	0.44
DI of NBHS1 (%)			83.31	60.44	53.25	87.67	64.00	60.44	94.94	62.50	63.64	70.35	-
DI of NBHS2 (%)			83.27	51.65	42.86	84.93	56.00	51.65	93.81	54.17	55.65	-	63.77

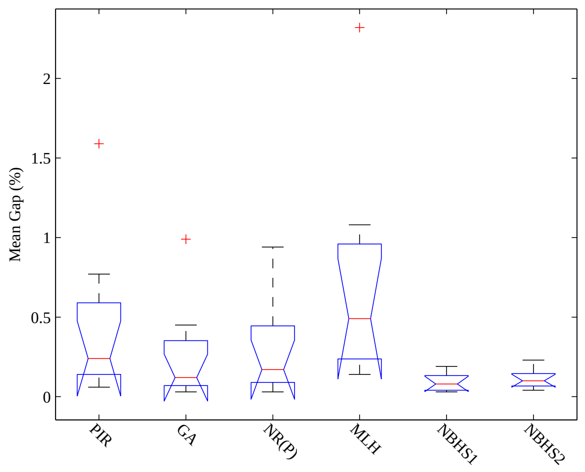

Figure 6: Box plots for the distribution of the mean relative gap values of 9 groups of problems with 5 constraints

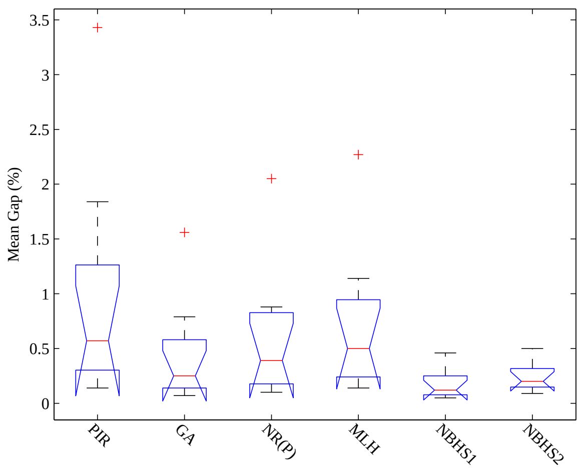

Figure 7: Box plots for the distribution of the mean relative gap values of 9 groups of problems with 10 constraints

ond best results are achieved by NBHS2 on the first six groups of problems while GA finds the last three ones. That is to say, GA has better performance than NBHS2 on problems with 500 items but the advantage is not significant in most cases. However, GA relies on solving the LP relaxation problem of the MKP which is very difficult to solve when the numbers of constraints and items are very large. Meanwhile, the repair operator utilized in NBHS2 is simple to implement and almost unaffected by the problem dimensionality. Therefore, NBHS2 can solve any larger scale MKP but GA cannot solve those problems due to the limitation on the MKP dimensions.

The overall average gap values of each method are listed in the third last row of Table 3. The largest average gap value of 3.60% is obtained by PECH which is 21.2 times over the smallest average gap value of 0.17% obtained by NBHS1. The average gap value of NBHS2 is also half of that obtained in GA which ranks third in performance among all the methods mentioned in this experiment. Fig. 7 summarizes the distribution of the mean relative gap values of 9 groups of problems with 10 constraints. The box plot also indicates that NBHS1 and NBHS2 have much stronger exploration ability and stability on finding better solutions.

On the other hand, the superiorities of NBHS1 and NBHS2 have been shown by the degree of improvement on other methods in Table 3. The greatest improvement made by NBHS1 is 95.28% on PECH and the smallest improvement is 63.04% on GA. The performance of these two methods also been improved by NBHS2 with ratios of 93.61% and 50.00%, respectively. Overall, the corresponding average improvements made by the proposed NBHS1 and NBHS2 approaches are 74.79% and 65.89%, respectively.

Experiments on problems with more constraints are carried out to further study the performance of the proposed algorithms. It can be seen in Table 4 that NBHS1 and NBHS2 still yield in general the best results among the ones considered. NBHS2 is more effective in solving problems contained by the first three groups and better computational results for the other problems are achieved by NBHS1. ADP-H2 instead of GA obtains the best performance other than NBHS1 and NBHS2. CH instead of PECH has the largest average gap value of 7.11% which is 19 times of that by NBHS1 and 16 times of that by NBHS2. These observations can be seen clearly from the distribution of the mean relative gap values in Fig. 8. ADP-H2 and HDP-LBC are absent from the box plot due to the lack of complete data.

To quantify the superiorities of NBHS1 and NBHS2, the degree of improvement on other methods have been computed. The greatest and smallest improvements made by NBHS1 are 94.94% on CH and 53.25% on GA, respectively. The average improvement on all methods is 70.35%. For NBHS2, the corresponding numbers are 93.81%, 42.86% and 63.77%, respectively.

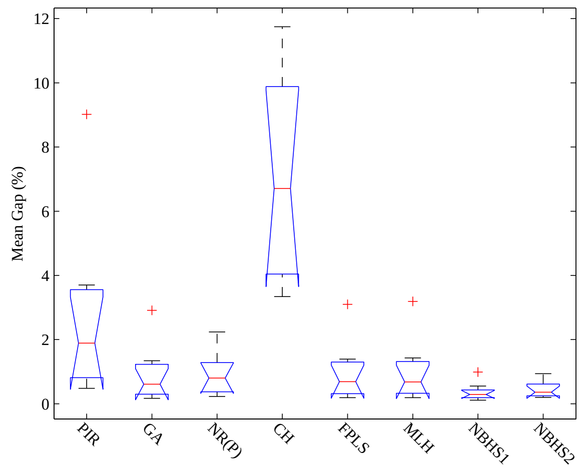

Figure 8: Box plots for the distribution of the mean relative gap values of 9 groups of problems with 30 constraints

Finally, a collection of results of some state-of-the-art methods for Test set 1 is listed in Table 5, which includes some presented in the above three tables. Each value in Table 5 is an average over 30 instances contained in one of the 9 groups with different numbers of constraints and items. The slackness ratio is ignored in this table but considered in the above. As seen

Table 5: Mean relative gap of instances in MKP_cb set

Instance		Mean relative gap							
m	*n*	PIR	GA	PECH	HDP-LBC	NR(P)	MLH	NBHS1	NBHS2
5	100	0.95	0.59	4.28	0.57	0.53	1.37	**0.13**	0.14
	250	0.31	0.14	4.03	0.16	0.24	0.56	**0.09**	0.12
	500	0.12	0.05	3.85	0.07	0.08	0.27	**0.05**	0.08
10	100	2.11	0.94	4.58	0.95	1.1	1.37	**0.27**	0.33
	250	0.66	0.3	3.31	0.32	0.48	0.52	**0.15**	0.24
	500	0.29	0.14	2.92	0.16	0.19	0.29	**0.08**	0.17
30	100	4.85	1.69	3.98	1.81	1.45	1.84	0.56	**0.53**
	250	2.02	0.68	2.69	0.77	0.8	0.74	**0.35**	0.44
	500	1.03	0.35	2.09	0.42	0.49	0.39	**0.18**	0.36
Average		1.37	0.54	3.46	0.58	0.6	0.82	**0.21**	0.27
DI of NBHS1 (%)		84.67	61.11	93.93	63.79	65.00	74.39	73.82	-
DI of NBHS2 (%)		80.29	50.00	92.20	53.45	55.00	67.07	-	66.34

in Table 5, both NBHS1 and NBHS2 perform on all instances clearly better than other approaches. NBHS1 obtains 8 out of 9 best results of all groups except the seventh group with 30 constraints and 100 items, where NBHS2 outperforms NBHS1 and the value of mean relative gap is best. On the other hand, if we consider the largest mean relative gap of each method, the largest value 0.56% for NBHS1 on the seventh group is less than one third of 1.69% for GA on the sixth group.

In terms of the average values of mean relative gaps on all instances in Test set 1, the ranking order of the performances of these methods from the best to the worst is: NBHS1, NBHS2, GA, HDP-LBC, NR(P), PIR, MLH and PECH. The average of the mean relative gap of the best performer NBHS1 is only 0.21%, almost less than a third of that under GA which is the most comparable to our approaches and recognized to be the best method by now. The average values of the mean relative gap between our best solutions and the best-known solutions of benchmarks are 0.21% for NBHS1 and 0.27% for NBHS2, respectively, whereas for other algorithms are from 0.54% to 3.46%. NBHS1 and NBHS2 are the leading algorithms, followed by GA, HDP-LBC and NR(P). The results of these three approaches are quite close to each other and differences between them are very small. The performances of PIR, PECH and MLH are far behind the other 5 methods.

The distribution of the mean relative gap values of 9 groups of problems plotted in Fig. 9 shows that NBHS1 and NBHS2 have smaller values and shocks of mean relative gaps which demonstrate much stronger exploration ability and stability on finding better solutions. The greatest and smallest improvements made by NBHS1 are 93.93% on PECH and 61.11% on GA, respectively. The average improvement is 73.82% on all methods. For NBHS2, the corresponding numbers are 92.20%, 50.00% and 66.34%, respectively.

In summary, all comparisons above suggest that NBHS1 and NBHS2 are the most leading algorithms and clearly outperform all other mentioned state-of-the-art approaches for large-scale MKPs like those in the Mk_cb set.

4.4. Further comparison of NBHS with HEDA on Test set 2

Though the performance of NBHS has been assessed on a large-scale MKP set in Section 4.3, this section aims to further study the capacity of NBHS on MKP with more items and

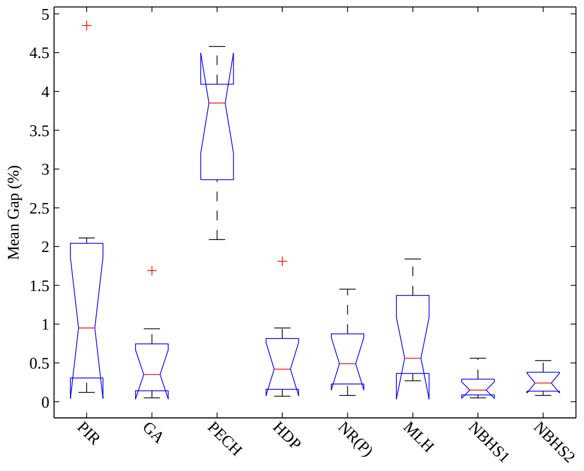

Figure 9: Box plots for the distribution of the mean relative gap values of instances in Mk_cb set

constraints. Thus we shall restrict our performance comparison solely to larger scale problems and the Mk_gk set including 11 larger size problems with *m*=15 to 100 and *n*=100 to 2,500 is selected for this purpose. An effective hybrid algorithm based on estimation of distribution algorithm (HEDA) has been proposed recently to solve the multidimensional knapsack problem (MKP) and it shows better performance than some existing algorithms. We thus focus on comparing our NBHS with HEDA below. For the comparisons to be fair, the maximum number of iteration is set to be 100,000 and 20 independent runs are conducted per instance. That is, the setting is the same in HEDA. It should be noticed that HEDA1 employs the pseudo-utility ratio based greedy-like heuristic procedure proposed by Chu and Beasley to repair the infeasible solutions and a new repair operator based on the specific heuristics knowledge of the MKP is introduced in HEDA2.

The computation results on the 11 large-scale instances of the second test set Mk_gk are reported in Tables 6 and 7, where the percentage gap is utilized to evaluate the performance of different approaches. In Tables 6 and 7, Min.Gap and Ave.Gap denote the minimum and average percentage gaps from the best-known values in 20 runs, respectively, and Var.Gap represents the variance of the gap. The better results are highlighted in

Table 6: Comparisons of NBHS1 with HEDA1

Problem	$n \times m$	Best known	HEDA1			NBHS1		
			Min.Gap	Ave.Gap	Var.Gap	Min.Gap	Ave.Gap	Var.Gap
Mk_gk01	100×15	3766	0.2655	0.3173	0.0783	**0.185874**	**0.282793**	**0.048179**
Mk_gk02	100×25	3958	0.0758	0.1983	0.1194	**0**	**0.198332**	**0.109141**
Mk_gk03	150×25	5650	0.1239	0.2018	0.0998	**0.070796**	**0.19469**	**0.063426**
Mk_gk04	150×50	5764	**0.0867**	**0.3348**	0.2916	0.19084	0.37127	**0.104762**
Mk_gk05	200×25	7557	0.1323	0.2772	0.215	**0.052931**	**0.122403**	**0.042474**
Mk_gk06	200×50	7672	0.2998	0.4438	0.2339	**0**	**0.290667**	**0.146561**
Mk_gk07	500×25	19215	0.5204	0.7203	0.824	**0.041634**	**0.096279**	**0.034293**
Mk_gk08	500×50	18801	0.6117	0.768	1.1028	**0.09574**	**0.206904**	**0.083153**
	Average		0.2646	0.4077	0.3706	**0.0797**	**0.2204**	**0.079**

Table 7: Comparisons of NBHS2 with HEDA2

Problem	$n \times m$	Best known	HEDA2			NBHS2		
			Min.Gap	Ave.Gap	Var.Gap	Min.Gap	Ave.Gap	Var.Gap
Mk_gk01	100×15	3766	0.6107	0.936	0.6555	**0.185874**	**0.293415**	**0.08721**
Mk_gk02	100×25	3958	0.758	0.979	0.4039	**0.176857**	**0.30192**	**0.08093**
Mk_gk03	150×25	5650	0.9381	1.1531	0.7878	**0.159292**	**0.444248**	**0.120439**
Mk_gk04	150×50	5764	1.093	1.2673	0.7069	**0.121443**	**0.40857**	**0.204463**
Mk_gk05	200×25	7557	1.2439	1.496	0.7483	**0.158793**	**0.39169**	**0.111906**
Mk_gk06	200×50	7672	1.121	1.4436	1.1573	**0.195516**	**0.421011**	**0.173081**
Mk_gk07	500×25	19215	1.7018	1.846	0.9873	**0.150924**	**0.232891**	**0.063714**
Mk_gk08	500×50	18801	1.6595	1.726	0.288	**0.356364**	**0.534014**	**0.088908**
Mk_gk09	1500×25	58085	2.4585	2.5461	0.6813	**0.175605**	**0.258156**	**0.059196**
Mk_gk10	1500×50	57292	2.0177	2.117	0.7807	**0.378761**	**0.532448**	**0.059884**
Mk_gk11	2500×100	95231	1.7043	1.7348	0.2184	**0.454684**	**0.634615**	**0.089418**
	Average		1.3915	1.5677	0.6741	**0.2286**	**0.40482**	**0.1036**

boldface.

Table 6 reveals that NBHS1 dominates the HEDA1 algorithm for 7 out 8 problems except Mk_gk04. HEDA1 obtains the better Min.Gap and Ave.Gap but NBHS1 has better Var.Gap for Mk_gk04. For Mk_gk02 and Mk_gk06, the Min.Gap values are equal to 0 for NBHS1, that is, NBHS1 found out the corresponding best known solutions. This confirms the strong exploration ability of NBHS1 on finding better solutions. As for the Var.Gap, the results of NBHS1 are better than that of the HEDA1 algorithm in all instances. Every Var.Gap value is smaller than 0.2 and most of them are smaller than 0.1. The comparison of Var.Gap values shows that NBHS1 has stronger stability than HEDA1. Focusing on the average of Ave.Gap values, the average for HEDA1 is nearly two times of that for NBHS1. In other words, the corresponding improvement made by the proposed NBHS1 approach on HEDA1 is nearly 50%.

A few observations can be made from Table 7. NBHS2 obtains better results in term of Min.Gap, Ave.Gap and Var.Gap for all the instances and it outperforms the HEDA2 with significant advantages. The strong ability of NBHS2 on finding better solutions is evident according to the Min.Gap values achieved by NBHS2. In addition, the smallest Ave.Gap value on Mk_gk01 by HEDA2 is close to 1.5 times of the largest Ave.Gap value on Mk_gk11 by NBHS2 and the average for HEDA2 is nearly 4 times that for NBHS2.

NBHS2 is much more robust than HEDA2 since the largest Var.Gap value on Mk_gk04 for NBHS2 is still smaller than the smallest Var.Gap value on Mk_gk11 for HEDA2 and the average of Var.Gap values for NBHS2 is hardly one-sixth of that for HEDA2. It is known that the MKP becomes more difficult as the dimension increases. The Ave.Gap values get worse in HEDA2 as the number of items grows, but there is no apparent trend in NBHS2. This shows that the dimension size has little impact on the performance of NBHS2. This evidence is likely due to the repair operator for NBHS2 which is independent of problem dimensionality. In sum, NBHS2 is more powerful than HEDA2 and it is desirable to use this heuristic for multidimensional knapsack problems, especially for the instances with large number of decision variables.

4.5. Algorithm complexity analysis and discussions

As discussed in the last few sections, NBHS1 and NBHS2 have showed great superiority over those state-of-the-art methods especially on large-scale instances. Note that the MKP is a well-known NP-hard problem. For traditional mathematical programming methods, it becomes more difficult or even impossible to solve within the available time when the dimension is large. Thanks to the specific heuristic procedure proposed in this paper, the shortcoming has been overcome and MKP with large dimension can be solved in an acceptable period of time. Computational complexity of the proposed NBHS algorithm is analyzed for evaluating its efficiency in this section.

Since the computational time depends on the runtime environment including hardware, programming language and coding style, it is not always a reliable measure when using algorithms to optimize problems. Hence, the number of basic operations required is usually served as the criterion to evaluate the computational complexity of algorithms. As shown in the computational procedure, NBHS consists of six main processes: initialization, memory consideration, pitch adjustment, re-initialization, repair operator and update operator. As mentioned earlier, the problem dimension is n, the number of constraints is m, the harmony memory size is HMS, the harmony memory considering rate is HMCR, the maximum generation number is NI and the slackness ratio is S.

It should be noted that the cost on repair operator is not constant and the computational complexity is analyzed in the worst scenario briefly. The worst computational complexity to repair an infeasible solution is

$$T_o = (1-S)\cdot n \cdot t_o,$$

where t_o represents the calculation time to repair one variable value.

For the initialization, the total time is calculated as

$$T_{in} = HMS \cdot (n \cdot t_{in} + T_o),$$

where t_{in} represents the calculation time to initialize each decision variable by the random method.

For the memory consideration, the total time is calculated as

$$T_{mc} = NI \cdot HMCR \cdot n \cdot t_{mc},$$

where t_{mc} represents the calculation time to get and round off the mean value.

For the pitch adjustment, the total time is calculated as

$$T_{pa} \approx NI \cdot HMCR \cdot n \cdot \left(\frac{2HMS}{HMS-1} + \frac{1}{3}\right) \cdot t_{in}.$$

For the re-initialization, the total time is calculated as

$$T_{ri} = NI \cdot n \cdot (1-HMCR) \cdot t_{in}.$$

For the repair process, the total time is calculated as

$$T_{ro} = NI \cdot T_o.$$

For the update operator, the total time is calculated as

$$T_{se} = NI \cdot t_{se},$$

where t_{se} represents the calculation time on selection and contains the time for evaluating the fitness values and constraint values.

Therefore, the total computational time of NBHS can be roughly calculated according to the following equation:

$$\begin{aligned} &T_{NBHS} \\ &\quad \approx T_{in} + T_{mc} + T_{pa} + T_{ri} + T_{ro} + T_{se} \\ &\quad = NI \cdot t_{se} + HMS \cdot (1-S) \cdot n \cdot t_o + n \cdot \Big\{ HMS \cdot t_{in} + \\ &\quad NI \cdot \Big[(1-S) \cdot t_o + HMCR \cdot \Big(\tfrac{t_{in}(7HMS-1)}{3(HMS-1)} + t_{mc} - t_{in} \Big) \Big] \Big\}. \end{aligned}$$

It should be mentioned that the cost of determining the surrogate multipliers in NBHS1 and the relative mean resource occupation in NBHS2 are not considered. The big O notation can exclude coefficients and lower order terms and thus it is commonly used to express the time complexity of an algorithm. Since HMS and n are far less than NI and t_o is far larger than t_{in}, t_{mc} and t_{se}, the total computational time of NBHS is

$$T_{NBHS} = O\left[n \cdot NI \cdot (1-S)\right].$$

Thus, the proposed NBHS algorithm has a linear asymptotic time complexity as the dimension increases. The problem dimension, the slackness ratio and the maximum generation number are the main determinants of the computational time. This can also be indicated by the CPU times for simulation experiments in Table 8 and Table 9.

Table 9: Average CPU computational times (s) for instances in Test set 1

Problem	n	m	s_j	NBHS1	NBHS2
Mk_gk01	100	15	0.5	12.54	13.92
Mk_gk02	100	25	0.5	12.8	14.07
Mk_gk03	150	25	0.5	15.8	17.91
Mk_gk04	150	50	0.5	17.17	18.8
Mk_gk05	200	25	0.5	19.63	21.38
Mk_gk06	200	50	0.5	20.14	23.11
Mk_gk07	500	25	0.5	37.29	41.68
Mk_gk08	500	50	0.5	39.48	42.47
Mk_gk09	1500	25	0.5	101.08	113.35
Mk_gk10	1500	50	0.5	104.01	110.2
Mk_gk11	2500	100	0.5	193.72	193.44

For all instances in Test set 1, the average computational times over 30 runs have been computed in Fig. 8. It can be concluded that the average computational times are approximately proportional to the $1-S$ values given m and n. Meanwhile, when we fix m and S, the average computational times are approximately proportional to the n values. A smaller slackness ratio or a larger problem dimensionality requires more computational efforts. These conclusions support the computational complexity result analyzed above. Furthermore, unlike other methods, the increase of m values only results in a small increase of computational times for constant n and S values. This may be induced by the evaluation cost of fitness values and constraints values in the repair process of infeasible solutions. The times for Test set 2 are given in Table 9 as well and result of the computational complexity analysis is confirmed again.

Based on the experiment results of Test set 1 and Test set 2, it can be concluded that the proposed NBHS algorithm has stronger search capability and can find better solutions than other methods. Through the computational complexity analysis, NBHS is considered to be an algorithm with linear asymptotic time complexity. The actual CPU computational times supports this conclusion. In sum, NBHS can be thought as a competitive approach for solving the multidimensional knapsack problems.

Some specific characters which enable NBHS to get promising performance are listed as follows:

(1) Unlike traditional mathematical programming methods, MKPs are regarded as decision making problems here. That is, the value selection from 0 and 1 for each variable individually. This makes NBHS possess a linear asymptotic time complexity and thus reduces the difficulty especially in the large-scale instances.

(2) The selective probability of each value, rather than the exact variable values, is concerned in the search process. This consideration is more reasonable and effective for 0-1 problems than other meta-heuristic algorithms.

(3) Based on new heuristic information, a repair operator on infeasible solutions is proposed to improve the search effi-

Table 8: Average CPU computational times (s) for instances in Test set 1

m	n	S_j	NBHS1	NBHS2	m	n	S_j	NBHS1	NBHS2	m	n	S_j	NBHS1	NBHS2
		0.25	23.10	23.73			0.25	24.78	25.48			0.25	27.22	27.23
5	100	0.5	14.82	15.23	10	100	0.50	15.88	16.20	30	100	0.50	17.82	17.98
		0.75	7.86	8.00			0.75	8.35	8.58			0.75	9.54	9.36
		0.25	49.79	50.30			0.25	54.18	54.93			0.25	58.53	59.40
5	250	0.5	30.60	31.77	10	250	0.50	32.94	35.46	30	250	0.50	36.40	38.43
		0.75	14.82	15.55			0.75	15.86	17.37			0.75	17.85	19.00
		0.25	90.58	96.70			0.25	98.13	106.74			0.25	106.53	115.49
5	500	0.5	54.56	58.76	10	500	0.50	59.08	71.65	30	500	0.50	64.60	74.26
		0.75	25.93	29.56			0.75	27.68	33.20			0.75	30.74	37.90

ciency. Through repair, these solutions are no longer infeasible and better solutions would be obtained.

(4) The concept of relative mean resource occupation is used to eliminate the difficulty in determining the surrogate multipliers for large-scale MKPs. This makes it possible to solve MKPs with any dimensions and constraints.

(5) With respect to the existing HS variants, the ingenious pitch adjustment scheme proposed in NBHS not only avoids the selection of both PAR and bw values but also considers the internal property of those harmonies stored in the HM. The new pitch adjustment rule without any parameter alleviates the parameter selection procedure and makes NBHS more convenient to be used in various fields.

Thanks to these proposed specific procedures, NBHS has a linear asymptotic time complexity and can solve any MKP in rational time.

5. Conclusion

In this paper, a new binary harmony search (NBHS) algorithm is proposed as an extension to the existing algorithms for effectively tackling large-scale 0-1 multidimensional knapsack problems. Since the classical HS algorithm is designed to solve continuous problems, the framework of HS needs to be adjusted to account for the characteristics of 0-1 multidimensional knapsack problems. In the proposed NBHS, the following adjustments have been made. First, a binary encoding is utilized. Second, a probability distribution based improvisation scheme which uses mean harmony and an ingenious pitch adjustment is used to generate a new candidate harmony. Third, a dynamic parameter adjustment scheme of HMCR is employed to better adapt to the evolution of the search process. Fourth, a new simple but effective repair operator derived from specific knowledge of the MKP is presented to guarantee the feasibility of harmonies stored in the harmony memory.

The extensive numerical experiments on two sets of high-dimensional 0-1 multidimensional knapsack problems reveal that the proposed NBHS algorithm outperforms all other state-of-the-art algorithms considered. In sum, NBHS is an effective and promising alternative for tackling 0-1 multidimensional knapsack problems in scientific and engineering fields.

[1] Zanakis SH. Heuristic 0-1 linear programming: An experimental comparison of three methods. Manage Sci 1977; 24(1): 91-104.

[2] Bertsimas D, Demir R. An approximate dynamic programming approach to multidimensional knapsack problems. Manage Sci 2002; 48(4): 550-565.

[3] Boyer V, Elkihel M, El Baz D. Heuristics for the 0-1 multidimensional knapsack problem. Eur J Oper Res 2009; 199(3): 658-664.

[4] Vimont Y, Boussier S, Vasquez M. Reduced costs propagation in an efficient implicit enumeration for the 0-1 multidimensional knapsack problem. J Comb Optim 2008; 15(2): 165-178.

[5] Li VC, Liang YC, Chang HF. Solving the multidimensional knapsack problems with generalized upper bound constraints by the adaptive memory projection method. Comput Oper Res 2012; 39(9): 2111-2121.

[6] Chu PC, Beasley JE. A genetic algorithm for the multidimensional knapsack problem. J Heuristics 1998; 4(1): 63-86.

[7] Leung SC, Zhang D, Zhou C, Wu T. A hybrid simulated annealing metaheuristic algorithm for the two-dimensional knapsack packing problem. Comput Oper Res 2012; 39(1): 64-73.

[8] Hanafi S, Freville A. An efficient tabu search approach for the 0-1 multidimensional knapsack problem. Eur J Oper Res 1998; 106(2): 659-675.

[9] Kong M, Tian P, Kao Y. A new ant colony optimization algorithm for the multidimensional knapsack problem. Comput Oper Res 2008; 35(8): 2672-2683.

[10] Chen WN, Zhang J, Chung HSH, Zhong WL, Wu WG, Shi YH. A novel set-based particle swarm optimization method for discrete optimization problems. IEEE Trans Evol Comput 2010; 14(2): 278-300.

[11] Wang L, Wang SY, Xu Y. An effective hybrid EDA-based algorithm for solving multidimensional knapsack problem. Expert Syst Appl 2012; 39(5): 5593-5599.

[12] Geem ZW, Kim JH, Loganathan GV. A new heuristic optimization algorithm: harmony search. Simul 2001; 76(2): 60-68.

[13] Mahdavi M, Fesanghary M, Damangir E. An improved harmony search algorithm for solving optimization problems. Appl Math Comput 2007; 188(2): 1567-1579.

[14] Omran MGH, Mahdavi M. Global-best harmony search. Appl Math Comput 2008; 198(2): 643-656.

[15] Alatas B. Chaotic harmony search algorithms. Appl Math Comput 2010; 216(9): 2687-2699.

[16] Askarzadeh A, Rezazadeh A. An innovative global harmony search algorithm for parameter identification of a PEM fuel cell model. IEEE Trans Ind Electron 2012; 59(9): 3473-3480.

[17] Askarzadeh A, Rezazadeh A. Parameter identification for solar cell models using harmony search-based algorithms. Sol Energy 2012; 86(11): 3241-3249.

[18] Lee KS, Geem ZW. A new structural optimization method based on the harmony search algorithm. Comput Struct 2004; 82(9): 781-798.

[19] Hasançebi O, Erdal F, Saka MP. Adaptive harmony search method for structural optimization. J Struct Eng 2009; 136(4): 419-431.

[20] Zou D, Gao L, Wu J, Li S, Li Y. A novel global harmony search algorithm for reliability problems. Comput Ind Eng 2010; 58(2): 307-316.

[21] Zou D, Gao L, Li S, Wu J. An effective global harmony search algorithm for reliability problems. Expert Syst Appl 2011; 38(4): 4642-4648.

[22] Coelho LDS, Mariani VC. An improved harmony search algorithm for power economic load dispatch. Energy Conv Manag 2009; 50(10): 2522-2526.

[23] Khorram E, Jaberipour M. Harmony search algorithm for solving combined heat and power economic dispatch problems. Energy Conv Manag 2011; 52(2): 1550-1554.

[24] Yuan Y, Xu H, Yang J. A hybrid harmony search algorithm for the flexible job shop scheduling problem. Appl Soft Comput 2013; 13(7): 3259-3272.

[25] Afkousi-Paqaleh M, Rashidinejad M, Pourakbari-Kasmaei M. An imple-

mentation of harmony search algorithm to unit commitment problem. Electr Eng 2010; 92(6): 215-225.
[26] Kulluk S, Ozbakir L, Baykasoglu A. Training neural networks with harmony search algorithms for classification problems. Eng Appl Artif Intell 2012; 25(1): 11-19.
[27] Srinivasa Rao R, Narasimham SVL, Ramalinga Raju M, Srinivasa Rao A. Optimal network reconfiguration of large-scale distribution system using harmony search algorithm. IEEE Trans Power Syst 2011; 26(3): 1080-1088.
[28] Geem ZW. Harmony search in water pump switching problem, in: First International Conference on Computing, Networking and Communications, ICNC 2005, Springer Berlin, Heidelberg, 2005, p. 751-760.
[29] Greblicki J, Kotowski J. Analysis of the properties of the harmony search algorithm carried out on the one dimensional binary knapsack problem, in: 12th International Conference on Computer Aided Systems Theory, EUROCAST 2009, Springer Verlag, 2009, p. 697-704.
[30] Wang L, Xu Y, Mao Y, Fei M. A discrete harmony search algorithm, in: International Conference on Life System Modeling and Simulation, LSMS 2010, Springer Verlag, 2010, p. 37-43.
[31] Wang L, Yang R, Xu Y, Niu Q, Pardalos PM, Fei M. An improved adaptive binary harmony search algorithm. Inf Sci 2013; 232: 58-87.
[32] Wang L, Yang R, Pardalos PM, Qian L, Fei M. An adaptive fuzzy controller based on harmony search and its application to power plant control. Int J Elec Power 2013; 53: 272-278.
[33] Zou D, Gao L, Li S, Wu J. Solving 0-1 knapsack problem by a novel global harmony search algorithm. Appl Soft Comput 2011; 11(2): 1556-1564.
[34] Kaveh A, Ahangaran M. Discrete cost optimization of composite floor system using social harmony search model. Appl Soft Comput 2012; 12(1): 372-381.
[35] Lee KS, Geem ZW, Lee S, Bae K. The harmony search heuristic algorithm for discrete structural optimization. Eng Optimiz 2005; 37(7): 663-684.
[36] Gao K, Pan Q, Li J. Discrete harmony search algorithm for the no-wait flow shop scheduling problem with total flow time criterion. Int J Adv Manuf Technol 2011; 56(5-8): 683-692.
[37] Geem ZW. Novel derivative of harmony search algorithm for discrete design variables. Appl Math Comput 2008; 199(1): 223-230.
[38] Layeb A. A hybrid quantum inspired harmony search algorithm for 0-1 optimization problems. J Comput Appl Math 2013; 253: 14-25.
[39] Moh'd Alia O, Mandava R. The variants of the harmony search algorithm: an overview. Artif Intell Rev 2011; 36(1): 49-68.
[40] Pirkul H. A heuristic solution procedure for the multiconstraint zero-one knapsack problem. Nav Res Logist 1987; 34(2): 161-172.
[41] Pirkul H, Narasimhan S. Efficient algorithms for the multiconstraint general knapsack problem. IIE Trans 1986; 18: 195-203.
[42] Akçay Y, Li H, Xu SH. Greedy algorithm for the general multidimensional knapsack problem. Ann Oper Res 2007; 150(1): 17-29.
[43] Hill RR, Kun Cho Y, Moore JT. Problem reduction heuristic for the 0-1 multidimensional knapsack problem. Comput Oper Res 2012; 39(1): 19-26.
[44] Yoon Y, Kim YH, Moon BR. A theoretical and empirical investigation on the Lagrangian capacities of the 0-1 multidimensional knapsack problem. Eur J Oper Res 2012; 218(2): 366-376.
[45] Yoon Y, Kim YH. A memetic Lagrangian heuristic for the 0-1 multidimensional knapsack problem. Discrete Dyn Nat Soc 2013; 2013: 1-10.