Noname manuscript No.
(will be inserted by the editor)

Compressed Representation for Higher-Level Meme Space Evolution: A Case Study on Big Knapsack Problems

Liang Feng · Abhishek Gupta · Yew-Soon Ong

Received: date / Accepted: date

Abstract In the last decades, a plethora of dedicated heuristic and meta-heuristic algorithms have been crafted to solve complex optimization problems. However, it is noted that the majority of these algorithms are restricted to instances of small to medium size only. In today's world, with the rapid growth in communication and computation technologies, massive volumes of data are generated and stored daily, making it vital to explore learning and optimization techniques that can handle 'big' problems. In this paper, we take an important step in the aforementioned direction by proposing a novel, theoretically motivated compressed representation with high-level meme evolution for big optimization. In contrast to existing heuristics and meta-heuristics, which work directly on the solution space, the proposed meme evolution operates on a high-level meme space. In particular, taking knapsack problem as the case study, a meme, in the present case, represents a knowledge-block as an instruction for solving the knapsack problem. Since the size of the meme, as defined in this paper, is not strongly sensitive to the number of items in the underlying knapsack problem, the search in meme space provides a compressed form of optimization. In order to verify the effectiveness of the proposed approach we carry out a variety of numerical experiments with problem sizes ranging from the small (100 items) to the very large (10000 items). The results provide strong encouragement for further exploration, in order to establish meme evolution as the gold standard in big optimization.

Liang Feng
College of Computer Science, Chongqing University, China.
E-mail: liangf@cqu.edu.cn

Abhishek Gupta, and Yew-Soon Ong
Data Science and Artificial Intelligence Research Center (DSAIR), School of Computer Science and Engineering, Nanyang Technological University, Singapore.
E-mail: {abhishekg, asysong}@ntu.edu.sg

Keywords Meme Evolution · Big Knapsack Problem · Dimension Reduction · Compressed Optimization

1 Introduction

In the past decades, by taking inspiration from nature or genetics, many heuristic and meta-heuristic algorithms have been proposed in the literature [1,2]. Due to their flexibility and ease of use, these algorithms have been applied to solve the difficult nonlinear, multi-modal, and discrete NP-hard problems, and have enjoyed significant successes in obtaining optimal or near-optimal solutions on a plethora of complex real-world applications. However, it is worth noting that majority of these are restricted to solving problems of small to medium size only. This is however expected due to the curse of dimensionality, which implies that the size of the search space grows exponentially with the dimension of the traditional problem solution.

Today, the advancements in digital sensors, communication systems, computing power, and storage infrastructures, have led to massive volumes of data generation and collection within organizations [3]. The real-world complex optimization problems, as stated previously, are therefore expected to encounter large-scale volume of data to analyze and optimize [4]. However, heuristic and meta-heuristic algorithms are expected to lack the required scalability. This is primarily because the search process in these algorithms is carried out directly on the solution space of the problem. As prescribed by the curse of dimensionality, the corresponding search complexity increases exponentially with the problem size. Clearly, new learning and optimization techniques that can efficiently overcome the obstacles associated with big dimensionality to handle large-scale problems are urgently needed. The discovery of such techniques will drive significant advances in science and engineering, and improve

overall quality of life, as much can be understood about the world around us simply by appropriately harnessing data. In this paper, we take a first step towards the aforementioned goal by deriving inspiration from the rapidly growing field of memetic computation, which is briefly described next.

Recently, the new science of memetics in cultural evolution, which represents the mind-universe analogue to genetics in biological evolution, has proliferated the fields of biology, cognition, psychology, etc. [5–8]. Like genes that serve as "instructions for building proteins" in genetics, memes are then "instructions for carrying out behavior, stored in brains" in memetics. Drawing from its sociological origins, a meme, in computational intelligence, is generally defined as a basic unit of information, i.e., a building block of higher-order knowledge, encoded in various possible computational representations for the purpose of improved problem-solving cite [8]. Notably, memes can either be utilized in a task-specific context, or may even be viewed as a computational entity housing generalizable domain knowledge. Based on this definition, memetic computation has then been defined as a paradigm that uses the notion of meme(s) as units of information encoded in computational representations for the purpose of problem-solving [8]. Significant research output has followed the expanding popularity of the concept, in both the natural and computational sciences. For instance, Situngkir presented a structured analysis of culture by means of memetics, where meme was regarded as the smallest unit of information [9]. Heylighen *et al.* discussed the replication, spread, and reproduction operators of memes in cultural evolution [10]. Further, Meuth *et al.* demonstrated the potential of meme learning for more efficient problem solving [11]. In this work, meme has been defined as the useful segments of the optimized problem solutions. Thus the direct reuse of these memes from small to complex problems could enhance the evolutionary search process. More recently, Feng *et al.* presented a study towards intelligent evolutionary optimization of problems through the transfer of structured knowledge in the form of memes learned from previous problem-solving experiences [12]. In these works, memes have been defined as building blocks of meaningful information, which represent recurring real-world patterns for problem-solving. Of particular interest to the present study is the observation from [12] that memes can be encoded in a significantly more succinct manner as compared to genetic representations in the original problem (see Fig. 1). Moreover, they can be designed such that their volume is not strongly sensitive to that of the underlying problem, providing a potentially compressed representation for big optimization.

Following this cue, in this paper, we propose a novel compressed representation with high-level meme evolution for big optimization. In particular, taking the classical knapsack problem (KP) as our case study, a suitable description

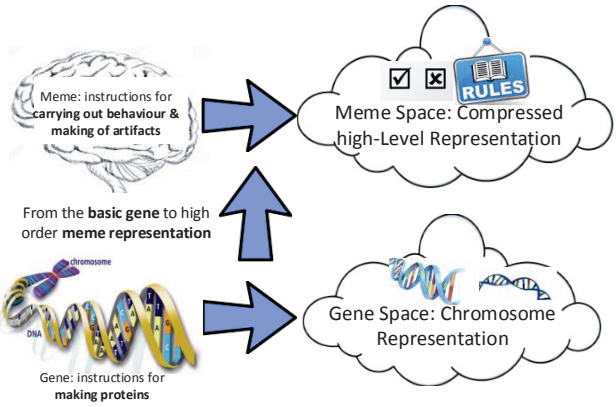


Fig. 1 The compressed high-level meme space versus original gene space.

of memes in the context of KP is first conceived such that a compressed description of the problem can be formulated. Subsequently, instead of following the traditional steps of genetic evolution, the proposed paradigm evolves high-level solution representations (or memes) of KP instances. It is worth noting that the meme defined in this paper is a higher level representation of the problem with respect to the traditional genetic representation. It is not the representation of the configurations of evolutionary operators and parameters for a particular solver that proposed for adaptive algorithm design in the literature. To the best of our knowledge, this is the first work that presents meme evolution as a search methodology in compressed space for the purpose of optimizing large-scale problems. In order to evaluate the performance of the proposed method, comprehensive numerical experiments are performed on KP instances of varying size, ranging from the small (100 items) to the very large (10000 items).

The rest of this paper is organized as follows. Section 2 presents the problem statement of the KP and discusses existing methods for solving it. A subsection describing meme-centric computing for search is also provided therein. The proposed compressed optimization via meme evolution paradigm for big KPs together with the theoretical rationale behind our proposition is detailed in Section 3. Section 4 contains empirical studies which verify the effectiveness of the proposed approach. Lastly, concluding remarks and directions for future research are drawn in Section 5.

2 Preliminaries

This section begins with a brief introduction to the KP. Subsequently, the literature review and a discussion on the drawbacks of solving KPs by directly searching the solution space of the problem are presented. Further, a discussion on meme-centric computing for search, which serves as the motivation behind the proposed method, is provided.

2.1 The Knapsack Problem

The knapsack problem (KP) is among the most fundamental problems in combinatorial optimization and has drawn the attention of researchers for several decades. The main reason behind the widespread interest in the problem is that it is simple to describe and has a wide range of notable applications in the real-world, such as, in project selection, investment decision-making, optimum resource distribution, etc. [13]. Theoretically, KPs have been proven to be NP-hard, which can be described as follows: given a set of N objects, each object (or items) i ($i = 1, \ldots, N$) is associated with a profit v_i and a weight w_i. The problem asks for filling the knapsack with a subset of the items such that the total profit is maximized and the total weight does not exceed a given capacity W. The mathematical formulation of the problem is stated as:

$$\begin{aligned} \max_{\mathbf{x}} \ & \sum_{i=1}^{N} v_i x_i \\ \text{s.t.} \ & \sum_{i=1}^{N} w_i x_i \leq W \\ & x_i \in \{0,1\}, 1 \leq i \leq N \end{aligned} \tag{1}$$

where the binary decision variables x_i are used to indicate whether item i is included in the knapsack or not. Without loss of generality, it may be assumed that all profits and weights are positive, that all weights are smaller than the capacity W so each item fits into the knapsack, and that the total weight of the items exceeds W to ensure a nontrivial problem.

2.2 Solving KPs by Searching on the Problem Space

Knapsack problems have been extensively studied due to their many practical applications in economic and industrial contexts [14]. In the literature, exact approaches such as branch-and-bound [15] and dynamic programming [16] have been proposed to find the global optimum by enumerating the possible solutions for KPs with small search spaces. On the other hand, many heuristic or meta-heuristic methods have also been proposed to handle small to medium KPs via an iterative stochastic search. Particular algorithms include simulated annealing [17], genetic algorithm [18], ant colony optimization [19], particle swarm optimization [20,21], differential evolution [22], Quantum-inspired Evolutionary Algorithm [23], and more recently, competition algorithm [24], harmony search [25], etc. A key feature of these algorithms is that they iteratively guide and modify the operations of subordinate heuristics to produce high-quality solutions, thereby demonstrating effective search capabilities for solving KPs. An interesting commonality among the algorithms is that they directly search the solution space of the problem. In other words, the solution representations of these methods are defined in the original search space itself, in the form of vectors of binary variables. More recently, Zhang *et al.* presented a novel way to design a P system for directly obtaining the approximate solutions for KPs [26]. However, besides the evolutionary search, the design of a spiking neural network is required.

It is clear that the length of the binary vectors equals the number of items involved in the given KP instance, i.e., the problem size of the KP. Remember, that an item can either be placed in the knapsack (encoded as 1) or not (encoded as 0). Accordingly, given N items, the number of possible solutions of the KP instance is 2^N. This implies that as N increases the size of the search space amplifies exponentially; thereby quickly making an algorithm which relies on navigating the original search space computationally intractable. It is for this reason that most existing algorithms are restricted to solving KPs of small to medium size only. In order to meet the requirements of the emerging big-dimensionality, powerful new methods are urgently needed.

3 Compressed Optimization by Meme Evolution for Big KPs

In this section, we present the compressed representation with meme evolution paradigm for big KPs. The method performs search in the high-level meme space, and is therefore capable of providing a compressed optimization process. As depicted in Fig. 2, the proposed paradigm consists of four basic components, namely, *meme initialization*, *meme reproduction*, *meme evaluation*, and *meme selection*.

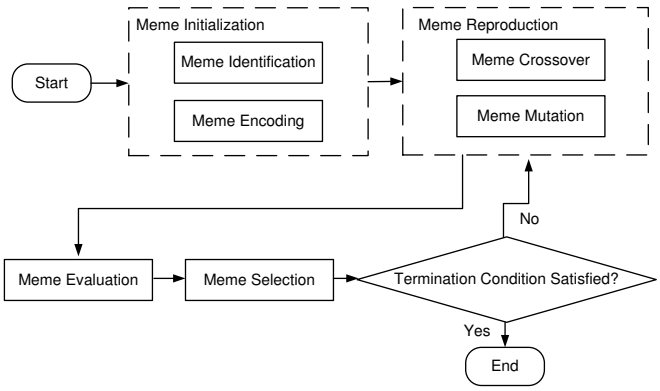


Fig. 2 Outline of the proposed meme evolution paradigm.

Meme initialization is the process of generating a population of memes which kick-start the evolutionary search process. This initialization can be performed randomly in much the same manner as chromosome initialization in standard genetic algorithms. Effective meme initialization must

however consider two important aspects, these being, *meme identification* and *meme encoding*. For any encountered problem, one must first identify an appropriate high-level representation or knowledge building-block, in the form of a computationally encoded meme, which allows subsequent evolution to be carried out efficiently.

Next, meme reproduction simulates the transmission of memes from parents to their offspring. Such transmission occurs via two means: *meme crossover* and *meme mutation*. Here, meme crossover serves to generate offspring memes through a direct recombination of the parent memes. Meme mutation, through a process analogous to genetic mutation, then introduces small random perturbations to the generated offspring with the aim of promoting diversity among future generations of memes.

Subsequently, during meme evaluation, offspring memes are first decoded into feasible solution vectors in the original solution space of the problem. The fitness values of the memes are then deduced based on the obtained solution vectors.

Finally, through meme selection, which mimics the process of 'natural selection' in biological evolution, memes with higher fitness values are encouraged to survive into future generations and contribute more offspring than their peers.

In what follows, the design specifics of the proposed meme initialization, meme reproduction, meme evaluation, and meme selection steps shall be presented in the context of KPs.

3.1 Meme Initialization for KPs

As stated in the aforementioned, the process of initializing memes begins with the identification of appropriate knowledge building-blocks of the underlying problem, that can be expressed efficiently in some computationally encoded form.

Based on the description in Section 2.1, it is possible to view the KP as a two class clustering problem. The first class representing items that are to be selected and placed in the knapsack, while the other representing items that are not to be selected. For the sake of illustration, consider the conceptual representation of the KP in Fig. 3(a). The figure depicts a two-dimensional random feature space in which each item is represented by a unique point marked by a circle. The 'optimum' grouping of the items into the two classes is depicted by the dashed partitioning, where the cluster of items that must be selected is denoted by the "+" symbol, and the cluster of items that must be rejected is denoted by the "-" symbol. Note that by an n-dimensional random feature space it is meant that each item in the KP is assigned an n-dimensional vector (the choice of n being left to the user) of randomly generated features. Each feature is considered to be a independently and identically distributed (i.i.d) number uniformly sampled between 0 and 1. For instance, in the two-dimensional case, an item could be assigned the random vector $(0.23, 0.57)$, which represents the location of the item in a new two-dimensional feature space. This random representation scheme has been employed in the proposed paradigm to describe each KP item (find the theoretical rationale behind this strategy in Section 3.5). Assigned vectors are considered fixed properties of their respective items throughout the evolutionary process[1].

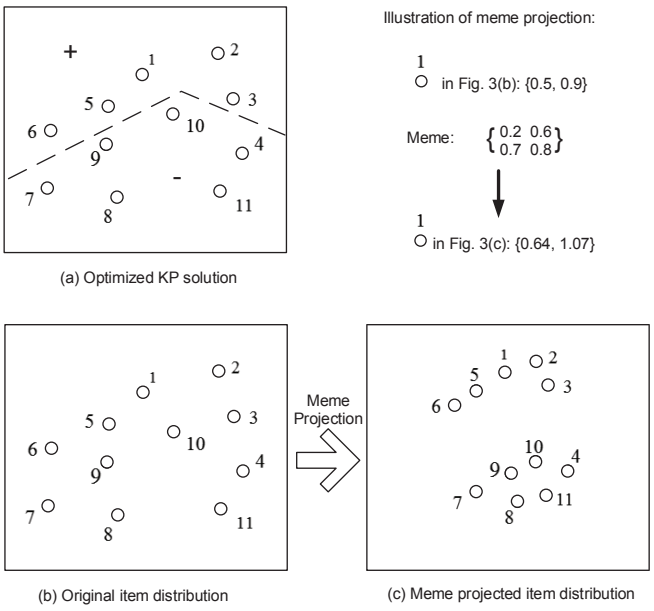


Fig. 3 Meme as a linear transformation of knapsack items. Each circle denotes one item; $+$ and $-$ in Fig. 3(a) represent the items that are either selected or not, in the optimum solution.

Following from the definition of memes as instructions for carrying out behavior stored in the brain, memes for the solution of KPs are designed to prescribe a series of steps that ensure an optimum grouping of the items. Note, from Fig. 3(b), that standard unsupervised clustering algorithms, such as K-Means, will generally fail to identify optimum clusters in the original random feature space. For this reason, in the context of KPs, we define memes as computational entities in the form of transformation matrices, which act on the (random) feature vector of the items so as to project them into a new space where the pairwise distances between the items is differently scaled. It is considered that the space of all possible transformation matrices constitutes a com-

[1] As the random features of items are fixed throughout the search, the task of finding the optimal KP solution becomes to search for the optimal meme which can provide the optimal clustering on the items giving the desired KP solution. The important properties of the items, i.e., profit and weight, are used to determine the fitness of a meme for guiding the search direction of meme evolution.

pressed meme space, in which the search is carried out for an optimal meme. Given a candidate meme, the subsequent clustering of items in the transformed space guides us to obtaining a particular candidate solution of the underlying KP. In particular, in the transformed space, items belonging to the same cluster shall map close to one another, while items belonging to different clusters shall map further apart. Analogously, the clustering may also be thought of as identifying a linear separating hyperplane dividing the points in the transformed space into two distinct classes. This notion is illustrated through Fig. 3(b) and Fig. 3(c), wherein the original distribution of items is projected into a new space via a meme-based operator. As can be observed, if an optimal meme has been obtained, the optimum grouping can perhaps be more easily deduced from the transformed distribution of the items (for e.g., via a K-Means clustering algorithm).

In order to have a simple description of the meme-based operator, such that it can be efficiently evolved in subsequent steps of the algorithm, the memes are simply encoded in the form of linear transformation matrices. In order to understand the workings of memes in this form, consider a pair of items $\mathbf{x}_a$ and $\mathbf{x}_b$ with randomly generated n-dimensional feature vectors $(x_{a1}, \ldots, x_{an})^T$ and $(x_{b1}, \ldots, x_{bn})^T$, respectively. The meme-based transformations of the feature vectors (e.g., from Fig. 3(b) to Fig. 3(c)) are then given by $\mathbf{M} \cdot \mathbf{x}_a$ and $\mathbf{M} \cdot \mathbf{x}_b$, where $\mathbf{M}$ is the meme. Consequently, the scaled distance between them in the transformed space is,

$$d_M(\mathbf{x}_a, \mathbf{x}_b) = \sqrt{(\mathbf{M}\mathbf{x}_a - \mathbf{M}\mathbf{x}_b)^T(\mathbf{M}\mathbf{x}_a - \mathbf{M}\mathbf{x}_b)} \quad (2)$$

where T denotes the transpose of a matrix or vector. The encoded meme $\mathbf{M}$ is an $m \times n$ matrix, where m is the dimensionality of the transformed space. Note that m, like n, is also user-defined. Since both these parameters are found to have no strong relationship with the number of items in the underlying KP, the requirements for a compressed representation scheme are satisfied.

Having identified a suitable computational encoding for the meme, a population of $\mathbf{M}$'s must be initialized to start the evolutionary process. This can be achieved through simple random generation, i.e., considering $\mathbf{M}$ to be an $m \times n$ matrix, each element of this matrix is randomly sampled from a uniform distribution over the range $[-1, 1]$.

3.2 Meme Reproduction for KPs

To generate the offspring memes from parents, we consider the following *meme crossover* operator:

$$\mathbf{M}_c = c_1\mathbf{M}_{p1} + c_2\mathbf{M}_{p2} \quad (3)$$

where $\mathbf{M}_{p1}$ and $\mathbf{M}_{p2}$ are the parent memes selected. c_1 and c_2 are randomly generated coefficients from the range $[0, 1]$. The methods generally employed in genetic search for parent selection can readily be replicated for choosing parent memes. For instance, the binary tournament selection approach is used in this study.

In order to introduce innovation and maintain diversity within the population of memes, *meme mutation* is carried out by adding a random perturbation matrix to the meme as follows:

$$\mathbf{M}_c^{'} = \mathbf{M}_c + \mathbf{R} \quad (4)$$

where $\mathbf{R}$ is an $m \times n$ matrix with each element randomly generated from a Gaussian distribution with mean μ and standard deviation σ. In this study μ and σ are set to 0 and 0.1, respectively. Further, similar to genetic search, *meme mutation* is performed only with a small probability, so as to prevent the algorithm from degenerating into a random search.

3.3 Meme Evaluation and Selection for KPs

A given meme $\mathbf{M}$ is evaluated by translating it into a feasible binary solution vector of the KP. This is achieved by first operating $\mathbf{M}$ on the random feature vectors of each item to project them into a new space (i.e., $\mathbf{x}' = \mathbf{M} \cdot \mathbf{x}$). A simple unsupervised clustering scheme, such as the K-Means algorithm, is then applied on the collection of transformed vectors to identify the two most natural clusters in the transformed space[2]. Based on some rule, the items in one of the clusters are assumed to be selected for placement in the knapsack, while items in the other are rejected. In the present study, the larger cluster (with more items) is always chosen for inclusion[3]. In case the selected cluster violates the capacity constraint of the knapsack, we apply Dantzig's greedy approximation algorithm [27] as a corrective (local refinement) step, in which items are considered for deletion in order of increasing *efficiency*, i.e., in terms of their profit to weight ratio. The aforementioned series of steps eventually leads to a feasible binary solution corresponding to a particular $\mathbf{M}$, the objective value (or fitness value) of which is simply calculated by summing the profits of items chosen for placement in the knapsack. A graphic illustration of the complete decoding procedure is provided in Fig. 4.

Further, the meme selection step is based on the fitness values of the memes, and follows a simple elitist strategy.

[2] The time complexity of K-means is $O(k * n)$, where k is the number of clusters and n is the number of date points to be clustered.

[3] Without loss of generality, other elaborated decoding process can also be applied here, e.g., choose the cluster with more efficient items.

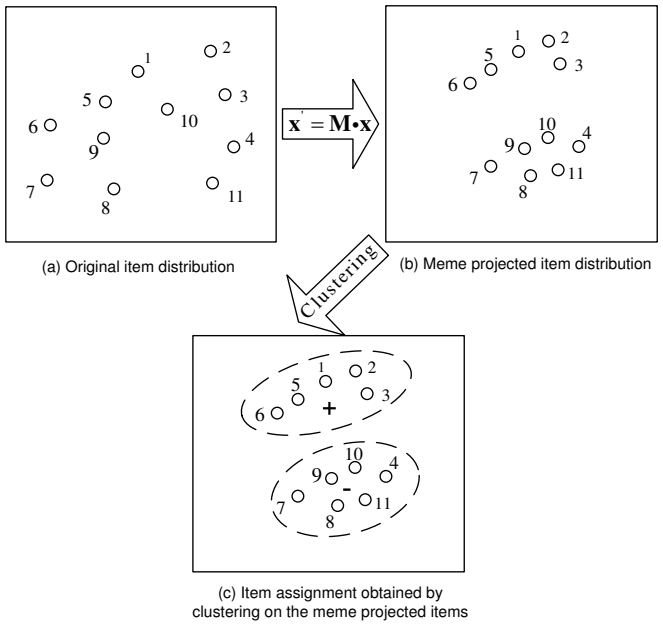


Fig. 4 Illustration of the proposed meme decoding for KPs.

Algorithm 1: Pseudo code of the proposed meme evolution paradigm for solving KPs

Begin:
For a given KP instance:
Generate *pop* memes randomly for the meme population.
Evaluate the entire meme population and obtain the fitness of each meme. /*use *meme decoding* as depicted in Fig. 4 */
while *Stop condition is not satisfied* **do**
    Perform *meme crossover* with tournament selection.
    Perform *meme mutation* with a probability pm.
    Perform *meme selection* with the proposed *meme decoding* scheme and elitist survival strategy.
End

3.4 Summary of the Proposed Meme Evolution Paradigm for KPs

The pseudo-code of the proposed meme evolution paradigm is outlined in Alg. 1. For an encountered KP instance, a population of memes, each of size $m \times n$ (m and n being user-defined), is randomly generated. Subsequently, the fitness value of each meme is obtained by the proposed meme decoding procedure. Next, meme crossover and probabilistic meme mutation are performed to generated offspring memes. Like for the case of genetic algorithms, the probability of meme mutation (i.e., pm) is set to a small value. In this study, pm is fixed at 0.2 for all experiments. Lastly, the offspring and parent memes are concatenated and ranked, in order to determine the fittest memes that survive to the next generation. The aforementioned steps are repeated until the fitness values converge, or some stopping criterion is met.

3.5 Discussion on the Theoretical Rationale of the Proposed Approach

In this section, we elaborate upon the theoretical foundations of the proposed approach. In particular, we aim to shed light on the following questions:

- - Instead of using the original features, i.e., the profit and weight, why are randomly generated features employed to represent a KP item?
- - How does the evolution of the meme (transformation matrix $\mathbf{M}$) operating on the randomly generated feature space facilitate the solution of the KP?

In the literature, a KP item is typically characterized by two features, namely, its weight (w) and profit (v). However, restricting our clustering (or classification) based approach to operate in only two dimensions heavily limits its capacity to identify an optimal grouping of items into selected (1) and rejected (0) classes. The difficulties pertaining to optimum clustering are further aggravated in the commonly encountered case that the profits and weights are linearly dependent, i.e., $v = \alpha \times w$, α being a positive proportionality constant, as the effective dimensionality reduces to one.

The first motivation for considering the generation of completely random features for each item in a KP instance stems from the simple observation above. One of the salient features of our proposition is that *the dimensionality of the synthetically generated random features can be arbitrarily configured by the user*. As a result, one is free to choose arbitrarily large feature dimensionality, which makes the corresponding KP items progressively more linearly separable in higher-dimensions. This claim can be substantiated by the following theorem.

Theorem 1 *If each (N) items in a KP is assigned an n-dimensional random feature vector, such that the features are $i.i.d$ and drawn from a uniform distribution, then, the probability (P) of there existing a linear hyperplane that optimally separates the items into selected (1) and rejected (0) groups strictly increases with n as:*

$$P_n = 2^{-(N-1)} \cdot \sum_{i=0}^{n} C_i^{N-1} \tag{5}$$

Proof Let the n-dimensional random feature vectors for each of the N KP items be contained in a (random) matrix $\mathbf{F}$ of size $N \times n$. From Cover's theorem [28], it trivially follows that if all submatrices of $\mathbf{F}$ of size $n \times n$ are of full rank (i.e., the rows/columns are linearly independent), then, the probability that there exists a linear hyperplane optimally separating the items is given by Eq. 5. Accordingly, it

only remains to be shown that any submatrix of $\mathbf{F}$ of size $n \times n$ is indeed of full rank with probability 1. To this end, note that if a square submatrix is not of full rank, implies that it is singular. However, the set of singular matrices in $\mathbb{R}^{n^2}$ has Lebesgue measure zero, since it is the zero set of a polynomial (i.e. the determinant function) and polynomial functions are Lebesgue measurable [29]. Moreover, as the uniform distribution (from which the features are sampled) is absolutely continuous with respect to the Lebesgue measure, the probability that a submatrix is singular (and therefore not of full rank) is 0. In other words, the probability that the submatrices are of full rank is 1. ■

As discussed in Section 3.1, a meme is defined in this paper as a linear transformation matrix that scales the distance between items (in the random feature space) in a manner analogous to identifying a linear separating hyperplane dividing the items into two distinct classes. According to the Theorem, the existence of an optimal separating hyperplane becomes increasingly more likely as the dimensionality of the random feature space is enlarged. Interestingly, Cover's theorem, which forms the crux of our proof above, also acts as the key motivation behind kernel methods in machine learning. However, unlike kernel methods, the item features in the present study are generated randomly (completely independent of the original features of the items). The reason is that in machine learning the effort is to learn a mapping between the original (input) features and output labels; thus, the input features must be preserved in some sense. On the other hand, the solution of the KP merely necessitates finding an optimal clustering (grouping) of the items, regardless of the feature space in which the clustering is carried out. Therefore, there is no pressing need to preserve the original features of the items.

Further, observe that if we choose $n = N-1$, as per Eq. 5 an optimal separating hyperplane is guaranteed to exist. This can be seen by the following calculation:

$$P_{n=N-1} = 2^{-(N-1)} \cdot \sum_{i=0}^{N-1} C_i^{N-1} = 1 \tag{6}$$

Nevertheless, in practice, we contend that the KP items need not be projected into an $N-1$ dimensional random feature space - in fact, this is undesirable as the size of the corresponding meme $\mathbf{M}$ grows larger as well. In fact, despite the dimensionality implied by the theoretical analysis to guarantee optimal clustering, satisfactory results can already be achieved by using significantly fewer random featuresłdue to the positive interplay of meme evolution with the corrective local refinements (as shall be shown in the next section). This observation is what truly makes the compressed representation practically viable.

4 Empirical Study

We evaluate the efficacy of the compressed representation with meme evolution paradigm through comprehensive numerical testing on a variety of KP instances, ranging in size from the small (100 items) to the very large (10000 items).

4.1 Experiment Setup

We randomly generate KP instances by considering different relationships between the profits of the items and their weights. The instances are generated according to [13, 27, 23].

Since the difficulty of a KP is greatly dependent on the correlation between profit and weight values, three categories of value correlations are considered herein while testing the proposed algorithm; these categories are: *strongly correlated*, *weakly correlated*, and *uncorrelated*. The process for generating the corresponding instances is as follows:

- *strong correlated*: $w(i) = rand(1, u)$ and $v(i) = w(i) + r$.
- *weakly correlated*: $w(i) = rand(1, u)$ and $v(i) = w(i) + rand(-u, u)$ (if $v(i) \leq 0$, this $v(i)$ will be ignored and the calculation is repeated until $v(i) > 0$).
- *uncorrelated*: $w(i) = rand(1, u)$ and $v(i) = rand(1, u)$.

where $rand(a, b)$ implies the generation of a uniform random value between a and b; u, r are predefined parameters. As reported in [27], problems with higher correlation between profits and weights are expected to be harder to solve.

Further, for each category of value correlation, two types of knapsack capacities are considered. One with restrictive capacity, and the other with average capacity. For the restrictive variant, the capacity is given by $2u$, where u, as introduced above, is the maximum possible weight of a single item. In the case of the average knapsack, the capacity is given by $0.5 \sum_{i=1}^{N} w(i)$, where N is the number of items. Clearly, the optimal solution for the "restrictive" type KP will prescribe very few items for selection, while the optimal solution for an "average" type KP will generally prescribe about half the items for inclusion.

Both small and big KP instances are investigated in this paper, so as to study the scalability and robustness of the proposed method. For small KPs, problems with 100 and 200 items are generated. u and r values are set to 10 and 5, respectively. For big KPs, problems of size 2000, 5000, and 10000 items are studied, while u and r are set to 100 and 50, respectively.

In order to compare the performance of our method against a baseline, we consider an advanced KP solver, namely Quantum-Inspired EA (QEA) [23], which is a type of univariate estimation of distribution algorithm [30], and two

popular variants of the genetic algorithm that have been recommended in [13,27,23] as being superior for solving problems with average and restrictive knapsack capacities. The first variant is a penalty function based approach in which if a solution violates the capacity constraint of the knapsack, its objective value is penalized by the following logarithmic penalty function:

$$Pen(s) = log_2(1 + \rho \cdot (\sum_{i=1}^{N} \mathbf{x}(i) \cdot w(i) - W))$$

where $\mathbf{x}$ denotes a binary KP solution that has violated the capacity constraint. ρ is the maximum efficiency considering all items in the KP instance (efficiency was defined in Section 3.3). The second variant is a genetic algorithm hybridized with a corrective step. Any solution violating the capacity constraint is now repaired according to Dantzig's greedy approximation algorithm, much like in the case of meme evolution. Since in all our experiments the genetic algorithm with a logarithmic penalty function is consistently outperformed by the one employing a corrective step, the latter is selected as the preferred baseline throughout this study.

In terms of the parameter settings used for the meme evolution paradigm, the values of m and n are both set to 10 in all the initial experiments. Thus, each item in a KP is represented by a 10 dimensional vector, each element of which is randomly generated between 0 and 1[4]. Accordingly, the memes $\mathbf{M}$ are described as a 10×10 square matrix. However, a further analysis of the effects of varying m and n is carried out and discussed in subsection 4.4. In addition, parameter settings of both, the QEA and the genetic solver, are according to the published paper [23]. Further, population sizes for the QEA, the genetic solver, and meme-based solver are configured to 100 individuals. The stopping criterion for all the algorithms is set to 100 generations and 200 generations for small and big KPs, respectively (unless otherwise mentioned). Finally, as has been mentioned earlier, the well-known K-Means unsupervised clustering algorithm is employed in the meme decoding process of the proposed meme evolution paradigm.

4.2 Performance on Small KPs

Table 1 presents the performance of the proposed algorithm, QEA and the genetic solver on small KP instances. Therein, "*B.Profit*", "*Ave.Profit*" and "*Std.Profit*" indicate the best profit, averaged profit, and standard deviation obtained by the corresponding algorithm across 30 independent runs, respectively. In addition, the upper bound, or exact global optimum, of each small KP instance is obtained by a dynamic programming solver. Superior performances of the meme evolution paradigm have been highlighted in bold.

As can be observed, the QEA and genetic solver perform rather well on the small KP instances. In particular, they find the global optimal on 9 and 8 out of 12 instances, respectively. However, it is most important to note that the performance of the meme evolution approach is still superior. It achieved competitive performance in terms of both "*B.Profit*" and "*Ave.Profit*" for the KP instances with 100 items. Moreover, for KP instances with 200 items, where the search space becomes slightly bigger and more complex, the proposed method finds solutions that are superior to both, the QEA and the genetic solver, on three instances, namely, "Ave.S.200", "Ave.W.200", and "Ave.U.200".

Further, in order to assess the efficiency of the proposed approach, the representative search convergence traces of the meme evolution paradigm, QEA, and the genetic solver are depicted in Fig. 5. As is clear, meme evolution leads to significantly faster convergence than the other two solvers. Since the three algorithms search on different spaces (with the QEA and genetic algorithm navigating the original solution space, while the proposed approach searches on a high-order meme space), the faster search performance achieved by our method indicates that searching on the compressed, high-order meme space is more efficient.

4.3 Performance on Big KPs

The empirical results obtained by the meme evolution paradigm, QEA and the genetic solver over 30 independent runs on big KP instances, ranging from 2000 to 10000 items in size, are summarized in Table 2. The superior performances of the proposed algorithm have been highlighted in bold. In particular, "*AP.SR%*" denotes the relative improvement in total profit achieved by the proposed approach, as compared to the QEA. It is computed as:

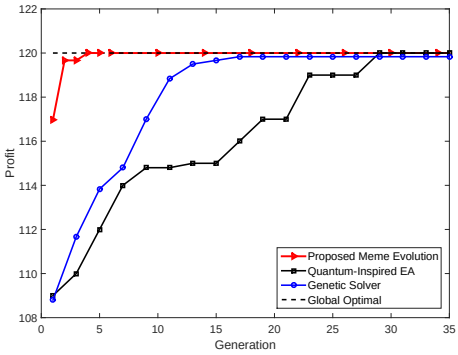
$$\textit{AP.SR\%} = \frac{(Ave.Profit_{memetic} - Ave.Profit_{QEA})}{Ave.Profit_{QEA}} \times 100\%$$

Diff.BP gives the differences between the best profits (i.e., "*B.Profit*") obtained by the QEA or genetic solver and the proposed approach. It is given by:

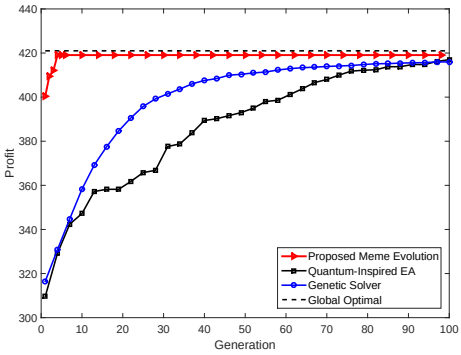
$$\textit{Diff.BP} = B.Profit_{QEA/genetic} - B.Profit_{memetic}$$

It can be observed from Table 2 that the proposed meme evolution paradigm achieved notably superior overall performance as compared to both the QEA and genetic solver. In particular, meme evolution improved the average profit (i.e., "*Ave.Profit*") on 16 out of the 18 KP instances considered, achieving a maximum of 37.36% improvement compared to the QEA. Further, in terms of "*B.Profit*", the proposed approach obtained better quality solutions on 14 out

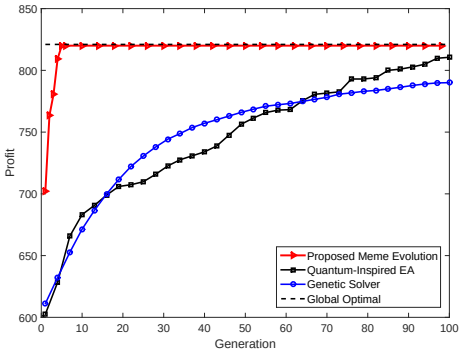
[4] For each KP instance, the feature values are randomly generated before the running of meme evolution. These features are fixed for all the independent runs on this KP instance.



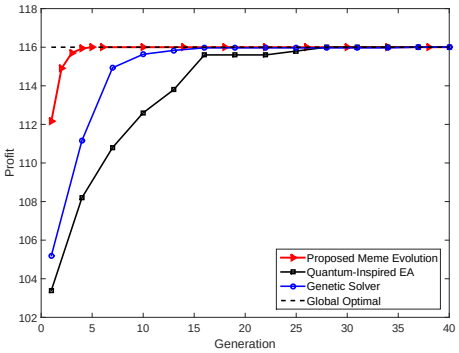
(a) Instance Res.S.100



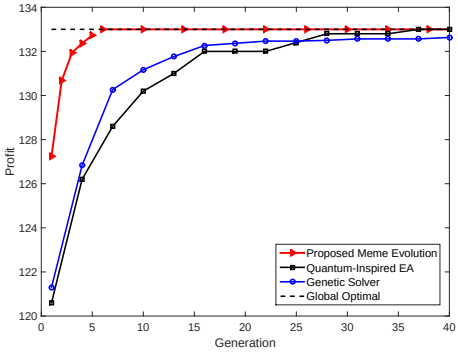
(b) Instance Ave.U.100



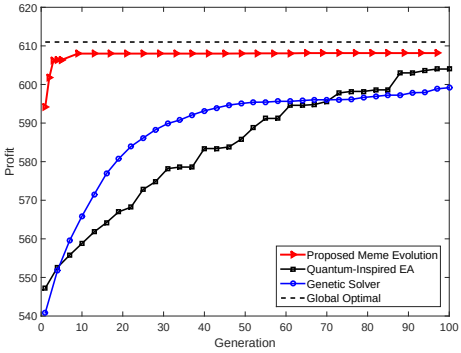
(c) Instance Ave.U.200



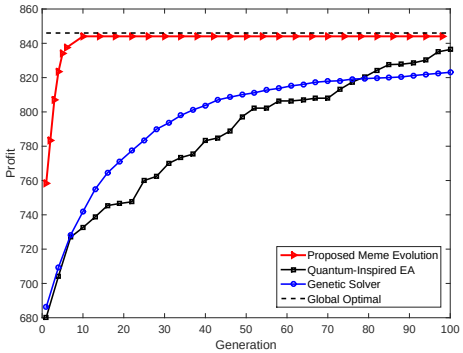
(d) Instance Res.U.100



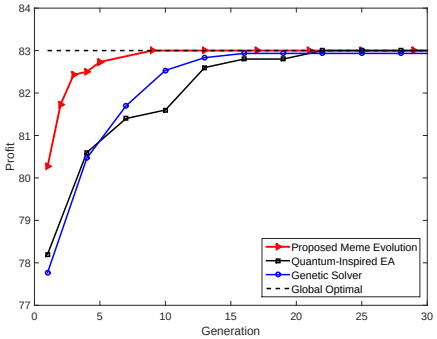
(e) Instance Res.U.200



(f) Instance Ave.S.100



(g) Instance Ave.W.200



(h) Instance Res.W.200

Fig. 5 Averaged convergence graphs of the proposed meme evolution, Quantum-Inspired EA, and genetic solver on small KP instances. "*Res*" and "*Ave*" denote the restrictive and average type of KPs, respectively. "*S*", "*W*" and "*U*" denote the *strongly correlated*, *weakly correlated*, and *uncorrelated* KP instances, respectively. 100 and 200 are the KP sizes.

Table 1 Averaged results of the proposed meme evolution, Quantum-Inspired EA, and the genetic solver on small KP instances. "*Res*" and "*Ave*" denote the restrictive and average type of KPs, respectively. "*S*", "*W*" and "*U*" denote the *strongly correlated*, *weakly correlated*, and *uncorrelated* KP instances, respectively. 100 and 200 are the KP sizes.

Data Set		Meme Evolution			Quantum-Inspired EA			Genetic Solver		
	Global Optimal	*B.Profit*	*Ave.Profit*	*Std.Profit*	*B.Profit*	*Ave.Profit*	*Std.Profit*	*B.Profit*	*Ave.Profit*	*Std.Profit*
Res.S.100	120.00	120.00	120.00	0.00	120.00	120.00	0.00	120.00	120.00	0.00
Res.W.100	74.00	74.00	72.13	0.43	74.00	**74.00**	0.00	74.00	**74.00**	0.00
Res.U.100	116.00	116.00	116.00	0.00	116.00	116.00	116.00	116.00	116.00	0.00
Ave.S.100	611.00	**611.00**	**608.13**	0.57	606.00	604.00	2.74	606.00	599.17	5.14
Ave.W.100	380.00	380.00	376.40	1.13	380.00	**379.00**	1.00	380.00	375.67	3.72
Ave.U.100	421.00	421.00	**419.07**	0.37	421.00	417.00	2.12	421.00	415.97	2.92
Res.S.200	120.00	120.00	120.00	0	120.00	120.00	0	120.00	120.00	0
Res.W.200	83.00	83.00	**83.00**	0.00	83.00	**83.00**	0.00	83.00	82.97	0.18
Res.U.200	133.00	133.00	**133.00**	0.00	133.00	**133.00**	0.00	133.00	132.73	0.52
Ave.S.200	1210.00	**1207.00**	**1205.10**	0.40	1180.00	1174.00	4.69	1190.00	1170.73	9.44
Ave.W.200	846.00	**846.00**	**844.07**	0.36	846.00	836.40	7.09	833.00	823.10	6.42
Ave.U.200	821.00	**820.00**	**820.00**	0.00	815.00	810.60	4.28	802.00	790.03	6.54

of the 18 KP instances. On the largest problem, the improvement achieved in terms of the best profit reaches a significantly large value of 117173.22. It is worth noting that the improvements brought about by the meme evolution paradigm are more significant on the "average" type KPs than on the "restrictive" type KPs. This is due to the limited knapsack capacities of "restrictive" type KPs, which leads to corresponding optimized solutions with only a few items selected for placement in the knapsack. Accordingly, the feasible search space is smaller and easier to navigate as compared to "average" type KPs.

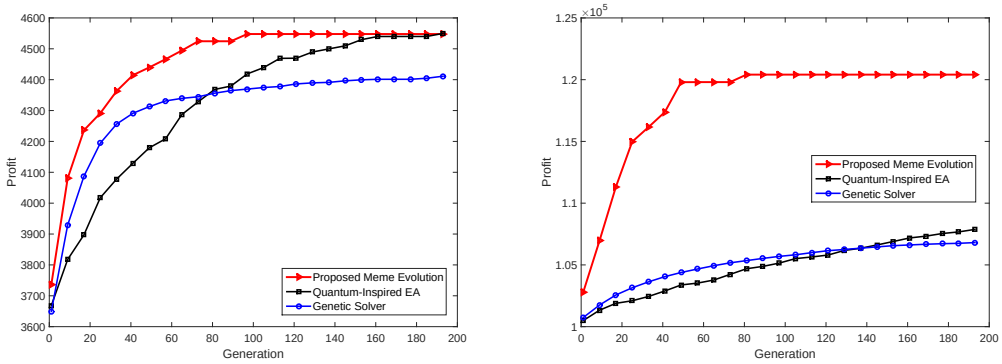
Next, the representative search convergence traces of the proposed method, QEA, and genetic solver are presented in Fig. 6. As can be observed, superior search performance has been achieved by the proposed method on all the big KP instances. In the case of "restrictive" type KPs (e.g., "Res.S.2000", "Res.S.5000", etc.), the QEA achieved similar profit as the proposed meme evolution, albeit with slower convergence speed. The genetic algorithm is also competitive with the proposed method over the first 50 generations. Subsequently however, genetic search is clearly outperformed. For the more complex "average" type KPs, the proposed approach demonstrates superior performance throughout the entire evolutionary search process.

Overall, it is observed that the convergence rates of both, the QEA and the genetic search, are much slower in comparison to the proposed meme evolution paradigm as the existing methods tend to get trapped at local optima. Meme evolution, by searching on an entirely different space, is seen to effectively overcome this obstacle, thereby establishing its superiority in the domain of big KP optimization.

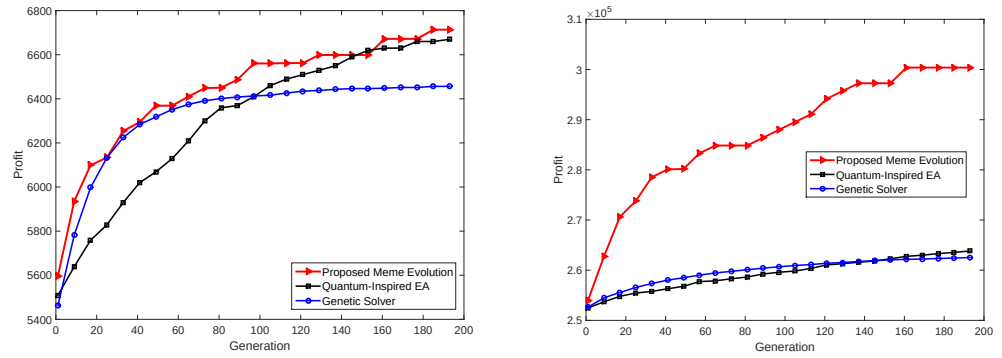
4.4 Scalability Assessment of the Meme Evolution Paradigm

In this subsection, we carry out a multifaceted scalability study of the meme evolution paradigm. We first refer to Fig. 7, which assesses the robustness of the method under increasing problem size. Remember that for all instances the meme $\mathbf{M}$ is a 10×10 square matrix. The figure plots the ratio between the average profits obtained (i.e., "*Ave.Profit*" in Tables 1 and Tables 2) and the best known (or best found) profit values for small to big KPs. As can be observed, the performance of both, the QEA and the genetic solver deteriorate monotonically with larger problems; while on the other hand, the meme evolution paradigm continues to perform at a superior level throughout.

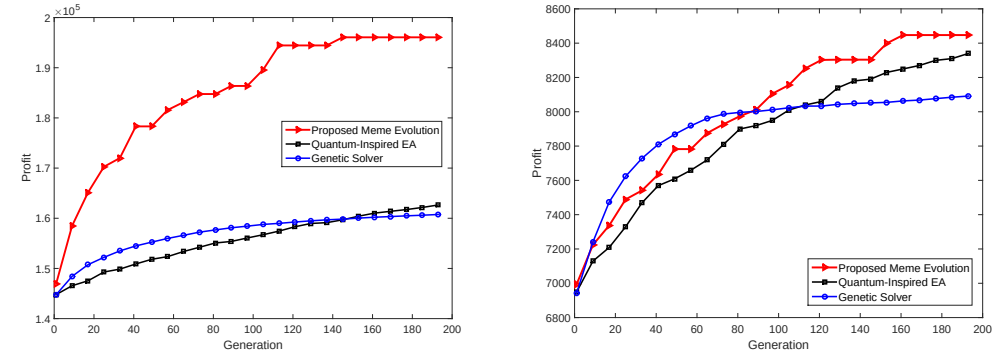
Subsequently, we consider the efficiency of the proposed method in comparison to the QEA. The genetic solver is not considered herein as we find the QEA to be superior, as indicated by previous empirical studies. A measure of efficiency in evolutionary computation is the number of generations it takes for an algorithm to converge to a near optimal solution. Typically, with increasing problem size, the required number of generations is expected to increase. In order to compare the performance of the meme evolution paradigm against the QEA in the above context, we carry out the following experiments: run the QEA (a) for 10000 generations for a KP with 2000 items, (b) for 20000 generations for a KP with 5000 items, and (c) for 40000 generations for a KP with 10000 items. Keep in mind that the meme-based solver was only run for 200 generations for all the above instances. The intention behind these experiments was to identify the number of generations required by the QEA to match the performance of meme evolution. In Fig. 8, "Case 1" represents the scenario in which the QEA is run for 200 generations, while "Case 2" represents experimental results with the aforementioned generation counts. It can be observed that the profit obtained by the QEA is significantly improved for "Case 2", i.e., when the population is allowed to evolve for numerous generations. However, it is worth noting that despite the increased computational effort, its performance remains inferior to meme evolution. In fact, its performance continues



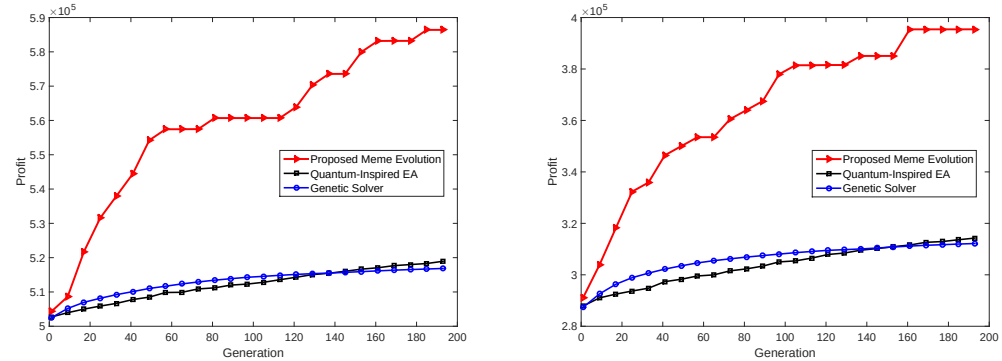
(a) Instance Res.S.2000 (b) Instance Ave.S.2000



(c) Instance Res.S.5000 (d) Instance Ave.S.5000



(e) Instance Ave.W.5000 (f) Instance Res.S.10000

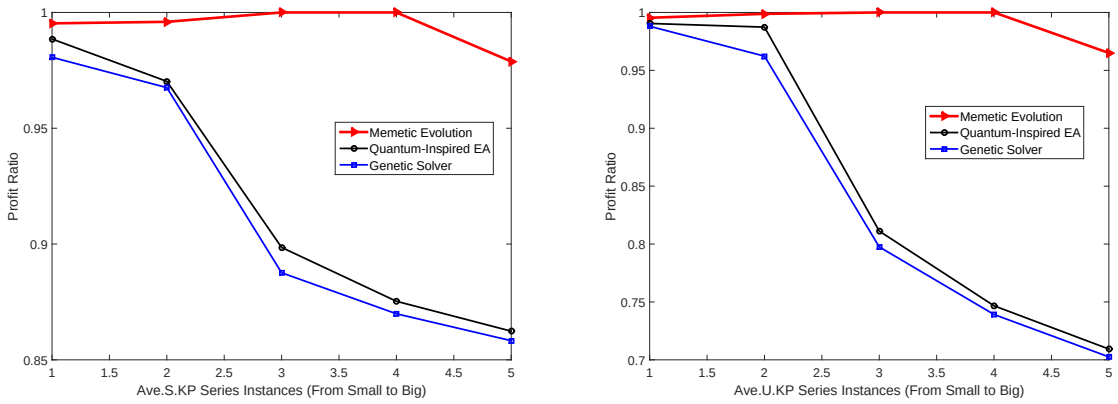


(g) Instance Ave.S.10000 (h) Instance Ave.W.10000

Fig. 6 Averaged convergence graphs of the proposed meme evolution, Quantum-Inspired EA, and genetic solver on big KP instances. "*S*", "*W*" and "*U*" denote the *strongly correlated*, *weakly correlated*, and *uncorrelated* KP instances, respectively. 2000, 5000 and 10000 are the KP sizes.

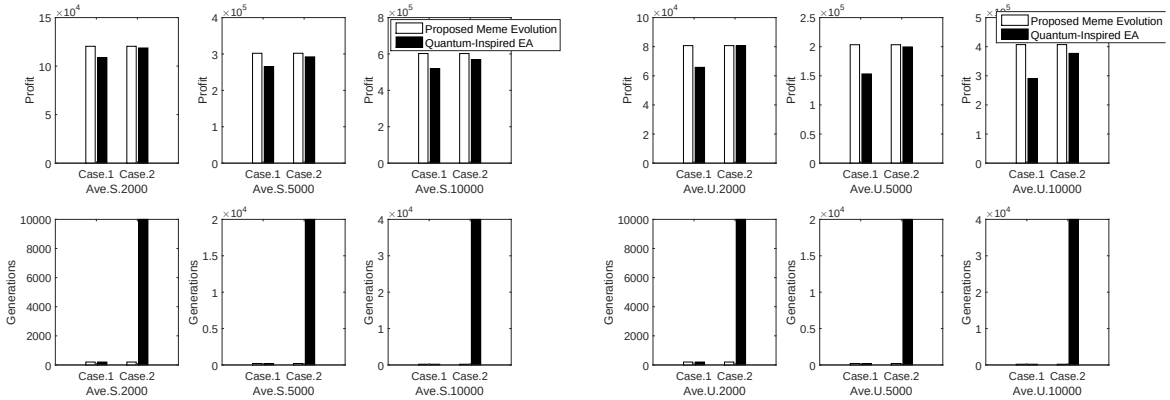
Table 2 Averaged results of the proposed meme evolution, Quantum-Inspired EA, and genetic solver on big KP instances."*Res*" and "*Ave*" denote the restrictive and average type of KPs, respectively. "*S*", "*W*" and "*U*" denote the *strongly correlated*, *weakly correlated*, and *uncorrelated* KP instances, respectively. 2000, 5000 and 10000 are the KP sizes.

Data Set	Proposed Method			Quantum-Inspired EA			Genetic Solver		
	B.Profit	*Ave.Profit*	*AP.SR%*	*B.Profit*	*Ave.Profit*	*Diff.BP*	*B.Profit*	*Ave.Profit*	*Diff.BP*
Res.S.2000	4547.83	**4547.83**	0%	4547.83	**4547.83**	0	**4549.85**	4417.92	+2.02
Res.W.2000	2521.72	**2521.72**	0.14%	2521.72	2518.13	0	2507.93	2489.72	−13.79
Res.U.2000	4944.58	**4944.58**	0.24%	**4954.22**	4932.67	+9.64	4951.76	4917.84	+7.18
Ave.S.2000	**120409.24**	**120409.24**	11.29%	108630.61	108191.48	−11778.63	107800.94	106874.25	−12608.30
Ave.W.2000	**77776.27**	**77776.27**	15.40%	67527.26	67399.53	−10249.01	67902.46	66508.58	−9873.81
Ave.U.2000	**80730.83**	**80730.83**	23.28%	65807.98	65487.47	−14922.85	65665.07	64384.19	−15065.76
Res.S.5000	**6748.55**	**6748.55**	1.18%	6699.97	6669.97	−48.58	6600.00	6456.48	−148.55
Res.W.5000	**4227.78**	**4227.78**	0.12%	**4227.78**	4222.84	0	4200.36	4136.34	−91.44
Res.U.5000	8016.09	**8016.09**	0.22%	8016.09	7998.57	0	7974.48	7852.42	−41.61
Ave.S.5000	**301937.72**	**301937.72**	14.25%	264874.63	264283.10	−37063.09	264120.86	262651.01	−37816.86
Ave.W.5000	**197667.87**	**197667.87**	21.20%	164520.38	163090.26	−33147.49	163496.12	160855.29	−34171.75
Ave.U.5000	**203382.67**	**203382.67**	33.93%	153222.49	151858.26	−50160.18	153316.19	150329.12	−50066.48
Res.S.10000	**8498.80**	**8498.80**	1.67%	8399.97	8359.59	−98.83	8249.42	8093.15	−249.38
Res.W.10000	**5278.70**	5136.23	−0.13%	5276.14	5142.99	−2.56	5195.38	**5146.94**	−83.32
Res.U.10000	**11177.88**	**10921.36**	0.16%	11116.82	10903.89	−61.06	11032.98	10866.92	−144.90
Ave.S.10000	**602326.69**	**589563.89**	13.50%	519974.18	519434.24	−82352.51	518991.73	516958.40	−83334.96
Ave.W.10000	**398897.01**	**398897.01**	26.66%	315826.80	314937.28	−83070.21	315246.71	312442.35	−83650.30
Ave.U.10000	**407200.36**	**392969.03**	37.36%	290672.14	288834.60	−116528.22	290027.14	286080.68	−117173.22



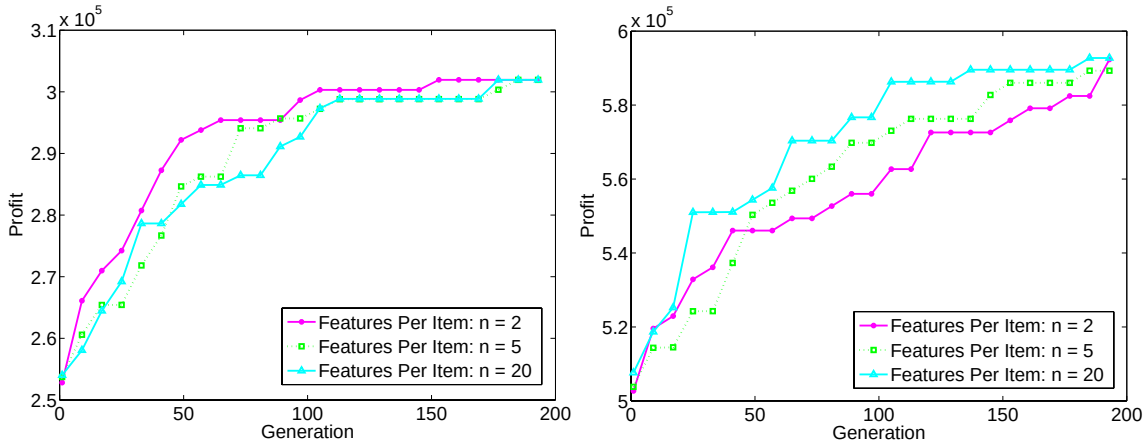
(a) Ave.S.KP Series Instances (b) Ave.U.KP Series Instances

Fig. 7 Robustness comparison of the meme evolution paradigm versus both, the Quantum-Inspired EA and the genetic solver, for the *Ave.S.KP* and *Ave.U.KP* series of instances.



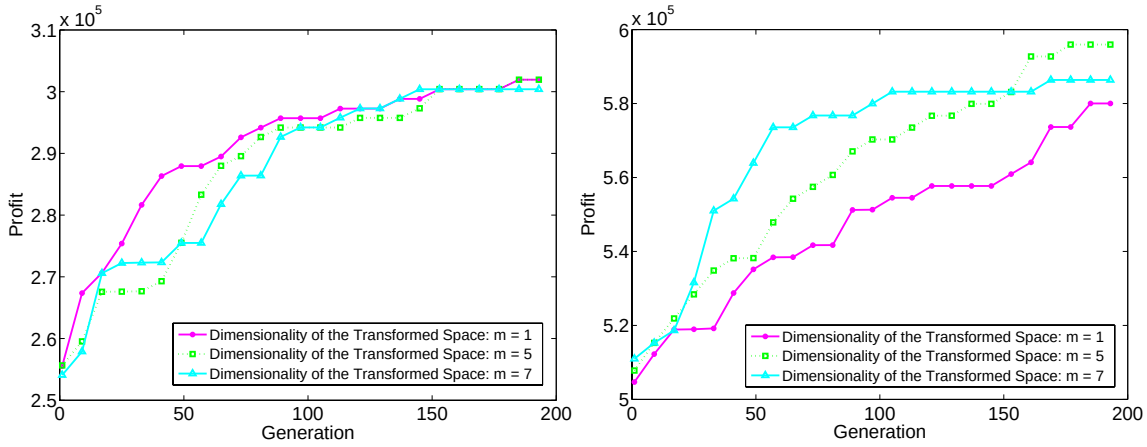
(a) Ave.S.KP Series Instances (b) Ave.U.KP Series Instances

Fig. 8 Efficiency of the proposed method against Quantum-Inspired EA for the *Ave.S.KP* and *Ave.U.KP* series of instances.



(a) Instance Ave.S.5000 (b) Instance Ave.S.10000

Fig. 9 Averaged convergence graphs of the proposed approach on instances "Ave.S.5000" and "Ave.S.10000" with different feature vector dimensionality.



(a) Instance Ave.S.5000 (b) Instance Ave.S.10000

Fig. 10 Averaged convergence graphs of the proposed approach on instances "Ave.S.5000" and "Ave.S.10000" with different dimensionality of the transformed space.

to deteriorate monotonically (with problem size) relative to meme evolution.

Finally, we attempt to study the effects of the user-defined meme and random feature space dimensionality parameters (i.e., m and n) while solving problems of large size. Note that it has already been verified through previous experimental results, wherein $\mathbf{M}$ was always a 10×10 matrix, that the required meme dimensionality is not very sensitive to the size of the underlying KP. Nevertheless, in order to provide deeper insight, a parametric study is carried out herein. The big KPs, i.e., "Ave.S.5000" and "Ave.S.10000", are used for our investigation. In terms of the dimensionality of the random feature vectors (i.e., n), three settings are studied, namely, $n = 2$, $n = 5$, and $n = 20$. For each of the above cases the meme $\mathbf{M}$ is assumed to be a square matrix. Subsequently, while studying the effects of the dimensionality of the transformed space (i.e., m), the following parameter settings are considered: $m = 1$, $m = 5$, and $m = 7$. The corresponding n is set to 10 throughout.

The results obtained under varying n and m are presented in Figs. 9 and 10, respectively, wherein the averaged convergence graphs over 30 independent runs are depicted. From Fig. 9(a), for instance "Ave.S.5000", the convergence trace is marginally faster for $n = 2$, as compared to $n = 5$ or 20. This can be explained by the observation that the small size of the corresponding meme provides a more succinct search process. Extending this logic, the comparatively slower convergence for $n = 20$ is also explained. However, it is interesting to note from Fig. 9(b) that while considering the largest instance (i.e., "Ave.S.10000") the trends observed earlier get reversed. We explain this effect by arguing that for large and complex search spaces, small values of n provide a very restrictive search process. The phenomenon is in fact contended to be analogous to the loss of critical

information during excessive data compression (lossy compression). In contrast, large values of n allow sufficiently diverse search, thereby enabling the algorithm to effectively locate optimum solutions.

Further, in Fig. 10, while studying the effects of m, it is noted that the trends observed in Fig. 9 are repeated. Thereby reinforcing the validity of our arguments presented above. In summary, we find that varying the meme and random feature space dimensionality can indeed be useful towards fine-tuning the performance of the meme evolution paradigm. However, it is also revealed that in many cases the effects may not be too significant, as a single $\mathbf{M}$ (10×10) has already been shown to work well on a variety of problem sizes.

5 Conclusion

In this paper, to handle big optimization problems, in contrast to traditional search paradigms which operate directly on the problem space, we have proposed a novel compressed representation for optimization search on the high-order meme space. In contrast to the search in problem space, the volume of the proposed high-order representation is not strongly sensitive to that of the underlying problem. In particular, using knapsack problem as the case study, we have designed the compressed, higher-order meme representation for KP solutions in the form of a linear transformation matrix. Further, we have proposed an algorithm which provided a form of compressed optimization by searching on the high-order meme level in the context of KPs.

In order to verify the efficacy of our proposed approach, comprehensive empirical studies have been conducted on six categories of problems with sizes ranging from the small to the very large. The superior performances observed showed that the exploration of high-order meme space is a promising direction for tackling big optimization problems when compared to traditional search methods.

In the future, we hope to generalize the methodology to a larger variety of big combinatorial optimization problems, such as, the generalized assignment problem, the quadratic assignment problem, vehicle and arc routing problems, feature selection, etc.

Acknowledgment

This work is partially supported under the National Natural Science Foundation of China (Grant No. 61603064), the Frontier Interdisciplinary Research Fund for the Central Universities (Grant No. 106112017CDJQJ188828), Chongqing application foundation and research in cutting edge technologies (cstc2017jcyjAX0319) and the Data Science and Artificial Intelligence Center (DSAIR) at the Nanyang Technological University.

No Conflict of Interest.

References

1. D. Whitley. A genetic algorithm tutorial. *Statistics and computing*, 4(2):65–85, 1994.
2. J. Kennedy. Particle swarm optimization. *Encyclopedia of machine learning*, pages 760–766, 2011.
3. Y. Zhai, Y.-S. Ong, and I. W. Tsang. The emerging 'big dimensionality'. *IEEE Computational Intelligence Magazine*, 9(3):154–160, 2014.
4. M. Ferguson. Architecting a big data platform for analytics. *A whitepaper prepared for IBM, Armonk, New York*, 2012.
5. D. Liu, K. C. Tan, C. K. Goh, and W. K. Ho. A multiobjective memetic algorithm based on particle swarm optimization. *IEEE Transactions on Systems, Man and Cybernetics - Part B*, 37(1):42–50, 2007.
6. K. Tang, Y. Mei, and X. Yao. Memetic algorithm with extended neighborhood search for capacitated arc routing problems. *IEEE Transactions on Evolutionary Computation*, 13(5):1159–1166, 2009.
7. F. Neri, C. Cotta, and P. Moscato. *Handbook of Memetic Algorithms*. Studies in Computational Intelligence. Springer, 2011.
8. X. S. Chen, Y. S. Ong, M. H. Lim, and K. C. Tan. A multi-facet survey on memetic computation. *IEEE Transactions on Evolutionary Computation, In Press*, 2011.
9. H. Situngkir. On selfish memes: culture as complex adaptive system. *Journal of Social Complexity*, 2(1):20–32, 2004.
10. F. Heylighen and K. Chielens. Cultural evolution and memetics. *In Encyclopedia of Complexity and System Science, ed. B. Meyers. Springer*.
11. R. Meuth, M. H. Lim, Y. S. Ong, and D. Wunsch. A proposition on memes and meta-memes in computing for higher-order learning. *Memetic Computing*, 1:85–100, 2009.
12. L. Feng, Y.-S. Ong, , M.-H. Lim, and Ivor W. H. Tsang. Memetic search with inter-domain learning: A realization between cvrp and carp. *IEEE Transactions on Evolutionary Computation*, 2014.
13. H. Kellerer, U. Pferschy, and D. Pisinger. Knapsack problems. *Springer*, 2004.
14. F. Kellerer, U. Pferschy, and D. Pisinger. *Knapsack problems*. Germany:Springer, 2004.
15. S. Martello and P. Toth. , upper bounds and algorithms for hard 0-1 knapsack problems. *Operations Research*, 45:768–778, 1997.
16. R. Andonov, V. Poirriez, and S.V. Rajopadhye. Unbounded knapsack problem: dynamic programming revisited. *European Journal of Operational Research*, 123:394–407, 2000.
17. A. Liu, J. Wang, G. Han, S. Wang, and J. Wen. Improved simulated annealing algorithm solving for 0/1 knapsack problem. *in: Proceedings of the Sixth international Conference on Intelligent Systems Design and Applications*, 02:1159–1164, 2006.
18. J. F. Zhao, T. Huang, F. Pang, and Y. Liu. Genetic algorithm based on greedy strategy in the 0-1 knapsack problem. *International Conference on Genetic and Evolutionary Computing*, pages 105–107, 2009.
19. H. X. Shi. Solution to 0/1 knapsack problem based on improved ant colony algorithm. *International Conference on Information Acquisition*, pages 1062–1066, 2006.
20. Z. K. Li and N. Li. A novel multi-mutation binary particle swarm optimization for 0/1 knapsack problem. *Proceedings of the 21st annual international conference on Chinese control and decision conference*, pages 3042–3047, 2009.

21. J. C. Bansal and K. Deep. A modified binary particle swarm optimization for knapsack problems. *Applied Mathematics and Computation*, 218(22):11042–11061, 2012.
22. P. Chen, J. Li, and Z. Liu. Solving 0-1 knapsack problems by a discrete binary version of differential evolution. *in: Second International Symposium on Intelligent Information Technology Application*, 2:513–516, 2008.
23. K. H. Han and J. H. Kim. Quantum-inspired evolutionary algorithm for a class of combinatorial optimization. *IEEE Transactions on Evolutionary Computation*, 6:580–593, 2002.
24. N. Moosavian. Soccer league competition algorithm for solving knapsack problems. *Swarm and Evolutionary Computation*, 20(0):14 – 22, 2015.
25. X. Kong, L. Gao, H. Ouyang, and S. Li. A simplified binary harmony search algorithm for large scale 01 knapsack problems. *Expert Systems with Applications*, 42(12):5337 – 5355, 2015.
26. G. Zhang, H. Rong, F. Neri, and M. J. Pérez-Jiménez. An optimization spiking neural p system for approximately solving combinatorial optimization problems. *International Journal of Neural Systems*, 24(05):1440006, 2014.
27. Z. Michalewicz and J. Arabas. Genetic algorithms for the 0/1 knapsack problem. In ZbigniewW. Ras and Maria Zemankova, editors, *Methodologies for Intelligent Systems*, volume 869 of *Lecture Notes in Computer Science*, pages 134–143. Springer Berlin Heidelberg, 1994.
28. T. M. Cover. Linear separability. *Open Problems in Communication and Computation*, pages 156–157, 1987.
29. Z. Wang, M. Zoghi, F. Hutter, D. Matheson, and N. De Freitas. Bayesian optimization in high dimensions via random embeddings. *International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1778–1784, 2013.
30. C. Patvardhan, S. Bansal, and A. Srivastav. Quantum-inspired evolutionary algorithm for difficult knapsack problems. *Memetic Computing*, 2(7):135–155, 2015.