Guided Genetic Algorithm for the Multidimensional Knapsack Problem

Pradumn
Information Technology Department
ABV-IIITM, Gwalior, India
imt_2019075@iiitm.ac.in

Utkarsh Agnihotri
Information Technology Department
ABV-IIITM, Gwalior, India
imt_2019109@iiitm.ac.in

Ashish Nehra
Information Technology Department
ABV-IIITM, Gwalior, India
imt_2019022@iiitm.ac.in

***Abstract*—Genetic Algorithm (GA) has emerged as a powerful method for solving a wide range of combinatorial optimisation problems in many fields. This paper presents a hybrid heuristic approach named Guided Genetic Algorithm (GGA) for solving the Multidimensional Knapsack Problem (MKP). GGA is a two-step memetic algorithm composed of a data pre-analysis and a modified GA. The pre-analysis of the problem data is performed using an efficiency-based method to extract useful information. This prior knowledge is integrated as a guide in a GA at two stages: to generate the initial population and to evaluate the produced offspring by the fitness function. Extensive experimentation was carried out to examine GGA on the MKP. We added Elitism to our research work that improved the quality of results obtained on running the GGA. And Produced better results in the same number of generations as that of without using Elitism.**

***Index Terms*—Genetic algorithm, Hybrid heuristic, Memetic algorithm, Multidimensional knapsack problem, Core concept, Guided genetic algorithm**

I. Introduction

The Genetic Algorithm (GA) was introduced over forty years ago and continues to be widely used in numerous research applications as a stochastic method for solving combinatorial optimization problems, particularly NP-hard problems. While exact methods often fail to find good solutions in a reasonable time, GA has been successfully applied to many real-world problems and traditional combinatorial problems [4], [21], [31]. Inspired by the biological evolution of living species, GA aims to improve the quality of successive generations by applying various genetic operators, such as crossover and mutation, to a randomly generated initial population of individuals. However, as GA is a stochastic process, there is no guarantee of optimality; only a large number of generations and individuals can increase confidence in the solution obtained [29]. Over the past few decades, several variants of GA have been proposed to improve its performance and accelerate its convergence in finding an optimal solution, such as modifying the GA operators or evolutionary behavior [9]. This paper focuses on the Guided GA concept and presents a memetic algorithm, named GGA, for the Multidimensional Knapsack Problem (MKP) that exploits prior knowledge about the problem data to drive the GA's evolutionary process towards promising areas of the solution space [23, 27]. GGA is inspired by two main concepts: the Proximate Optimality concept, which assumes that the best solutions have a similar structure, and the Core Concept for the MKP, which provides a mathematical model for ordering the items in the MKP based on a compromise between their weights (costs) and values. The CCMKP output is used at two stages of the evolutionary process: the initialisation and evaluation (fitness function) stages. The paper is structured as follows: Sect. 2 provides a definition of MKP and the Core concept for MKP (CCMKP). Section 3 gives an overview of the literature review related to the GGA. The proposed algorithm GGA is introduced in Sect. 4. The experimental setup and the parameters tuning are given in Sect. 5. Section 6 presents the conducted experiments and the obtained results. Conclusions and final remarks are drawn in Sect. 7.

Identify applicable funding agency here. If none, delete this.

II. The multidimensional knapsack problem

This section presents the MKP mathematical model adopted in this work and provides a quick overview on the Core concept for MKP [24].

A. Problem definition

The MKP is composed of n items and a knapsack with m different capacities $[c_i]$ (i ε 1,..., m). Each item j (j ε 1,..., n) has a weight $[w_i]$ j on each capacity i of the knapsack and a value $[p_j]$. The goal is to pack the items in the knapsack so as to maximise the overall value without exceeding the capacities of the knapsack. The MKP model can be represented by the following integer program:

$$\text{Maximise}: \sum_{j=1}^{n} p_j x_j$$

$$\text{Subject to}: \sum_{j=1}^{n} w_{ij} x_j \leq c_i \quad i \in \{1 \dots m\}$$

$$\text{where } x_j \in \{0, 1\} \quad j \in \{1 \dots n\}$$

A feasible solution X for MKP represents the selected items to be packed. A decision variable x_j represent the item j and is binary where x_j = 1 means that item j is packed, and x_j = 0 means that item j is not packed in the knapsack. w_{ij} represents the weight of item j on constraint i.

B. The core concept for MKP

The application of the Core concept for resolving combinatorial and linear programming issues was first limited to knapsack problems by Balas and Zemel [7], and later, it was broadened to include the MKP by Puchinger et al. [24]. The CCMKP computes the effectiveness (score) of each variable (item) in the MKP, which reflects the anticipated increase in value by including a variable in the final solution. High efficiency indicates that a variable is likely to be included in the optimal solution, while low efficiency indicates that it is unlikely to appear in the optimal or nearly optimal solutions. Variables with similar efficiency are categorized as the Core. After sorting variables in descending order based on their CCMKP efficiency, they are separated into three groups. Variables with high efficiency are set to 1, while those with low efficiency are set to 0, and those with similar efficiency form the Core. Consequently, the Core concept allows the original problem to be reduced to the Core problem. The Core concept is based on an efficiency measure function. The aim is to assign an efficiency value to each variable, according to its significance in producing the optimal solution, in such a way to promote those having the high values and low weights. Several efficiency measures have been used as approximations of the efficiency function, for example, simple efficiency (e_j^{simple}) [13], scaled efficiency (e_j^{scaled}), Senju Toyoda (e_j^{st}) [28] and general efficiency ($e_j^{general}$) [19] as shown in Equations.

$$e_j^{\text{general}} = \frac{p_j}{\sum_{i=1}^{m} r_i w_{ij}}$$

$$r_i = \frac{\sum_{j=1}^{n} w_{ij} - c_i}{\sum_{j=1}^{n} w_{ij}}$$

where e_j : efficiency of item j; p_j : value of item j; w_{ij} : weight of item j on constraint i; c_i : capacity of knapsack on constraint i and r_i : utility ratio.

III. Related works

In literature, various methods related to the guided GA concept have been proposed for a wide range of applications. For example, Yang and Jat [34] used a memory called MEM to guide the GA process for solving the Course Timetabling Problem. MEM is a limited-size list of room and time slot pairs that are integrated into the crossover operator of the guided GA. In [2], an external structure was used to guide GA, while in [10], GA was augmented with an approximate probabilistic model to guide the crossover and mutation operators. The probabilistic model estimated the quality of candidate solutions and evaluated their fitness, which enabled the operators to generate more promising solutions. For specific problem characteristics, the ProDiGen GA [32] used precision, simplicity, and completeness values to guide the optimization process. A slowdown-guided GA [16] estimated the execution slowdown of tasks to guide the GA search process, while guided mutation [35] used a probability model to generate offspring in promising areas. The guided crossover operator in [25] selected the second parent based on a metric called Mutual fitness, and one offspring was generated by crossing the parents. Although these GA variants incorporate guidance methods specific to the addressed problems, they do not propose a formal way to extract the guidance information or integrate it into the optimization process. Some approaches use genetic operators for partial guidance. The cited works demonstrate the versatility of guided GA in optimization and the diversity of guidance methods employed.

IV. The guided genetic algorithm GGA

The algorithm in this paper is motivated by the observation that in many optimisation real-world problems, some prior information about the components/patterns that are likely to appear in the good solutions could be known as shown in [1]. For example, in MKP, it is possible using linear relaxation or the optimal fractional solution [12], to predict some of the items that are likely or unlikely to appear in the good solutions. This study proposes a method for using such prior information as an additional guide for the GA evolutionary process for the MKP problem. By guide, we mean any structure external to GA, which maintains its original composition and is used to drive its search process. This can be through a subset of operators, in order to accelerate the search process and improve the speed of convergence. This section aims to describe the GGA components.

A. Chromosome design

The population is composed of a finite number of chromosomes. A chromosome represents a feasible solution to the problem (MKP). As mentioned before, the target in the MKP is to define the subset of items that maximises the total profit. The GGA chromosome consists in the set of items to be added to the knapsack. GGA uses the binary encoding scheme, where each gene represents whether the item is selected or not. The allele value 1 represents that the item is selected and the allele value 0 represents that it is not. Each of the items are coded either with 0 or 1. A chromosome is formed only by the number of items that it contains. (Fig. 1) shows a chromosome.

B. Guiding information

The guiding information is based on the work by [24]. The items are sorted decreasingly according to their statistical efficiencies e_j based on the value and the weight (e_j^{simple}, e_j^{scaled}, e_j^{st} or $e_j^{general}$). In simple words, the items are sorted based on how likely each item is to appear in high performing individuals, the items at the top of this list are likely to be selected while the items at the bottom are unlikely to appear in good solutions. However, it is important to note here that this list is just an estimate and not a predefined part of the solution. It should also be noted that the greedy heuristic [27], as it is only based on the efficiency sorting, is not an effective solution for the strongly correlated problem instances of MKP [18].

The sorting operation allows favouring items that have a good compromise (i.e. efficiency) between the average value

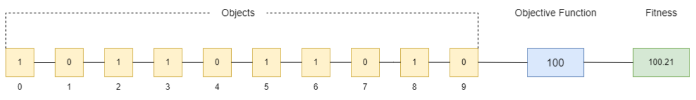

Fig. 1. An example of the chromosome design. The objects (items) packed in the knapsack are represented as binary representation, a one on a particular array index represents that the item denoted by that index is selected and a 0 represents that that particular item is not selected. The objective function value is calculated by summing the values of all the objects selected while the fitness is calculated according to the fitness function.

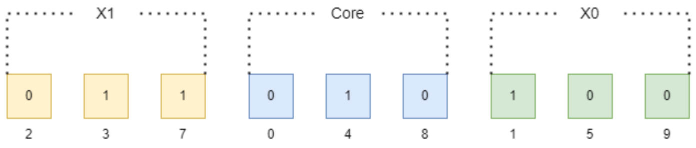

Fig. 2. An example of the guide construction. The objects (items) are sorted according to the efficiency e_j

and the overall weight. The efficiency of an item is high if its value is high while its required global capacity is low. The sorted items are split into three sets (Fig. 2) where the value of each variable is assigned as follows:

- X_1 : x_j = 1 The variables have the best efficiency e_j . These variables are most likely to build the best solutions even the optimal solution.

- Core : x_j = ? The efficiency values e_j of these items are medium, therefore, it is difficult to predict with confidence whether or not some may appear in the optimal solutions.

- X_0 : x_j = 0 The variables have a very low efficiency e_j, in other words, the value is low or the weight is large or both.

The guide is represented by the variables of $X_1 \cup$ Core $\cup$ X_0. The sizes of X_1, Core and X_0 are determined as follows: Construct a feasible solution by adding the items in order. The item that makes the solution infeasible represents the centre of Core. The size of each part of the guide depends on the size of Core. Setting the size of Core defines the size of the other parts.

C. Initial Population

The GGA algorithm uses a special initialisation process which allows the GA to make use of the prior information available about the items, and in the same time generates a diverse initial population to ensure exploration of the search space. A chromosome is generated from the items of X_1 completed by items generated randomly. In each chromosome, X_1 is integrated with a probability α. If α is set to zero this means that all the items in each individual are selected randomly, while α = 1 means that each individual in the initial population contains all the items of X_1. This method allows having an initial population of good quality by integrating X_1 and ensures the diversification by adding the rest randomly.

D. Fitness evaluation

In GGA, the objective function is different than the fitness function f(j). The first is the value of a solution relative to MKP problem. It is evaluated according to, while the fitness function is defined in a way to guide the search process of GGA. Different formulations of the fitness function are examined by introducing the efficiency e_j , X_1 and X_0.

A) The fitness function is calculated in the same way as the objective function:

$$f(j) = \sum_{j=0}^{n} p_j x_j$$

B) The efficiency e_j is introduced in its evaluation. Every generation, the fitness value of each chromosome is calculated. The fitness formula allows giving more chance to the chromosome that has a high efficiency to be selected more than the others.

$$f(j) = \sum_{j=0}^{n} e_j p_j x_j$$

C) X_1 and X_0 are introduced in the fitness measuring; the first as a reward and the second as a penalty. The aim is to reward (respectively penalise) each chromosome according to its similarity with X_1 (respectively X_0). Thus allows, at the same time, increasing the chance for the good chromosome to be selected and decreasing it for the bad one.

$$f(j) = \sum_{j=0}^{n} p_j x_j + \text{ reward } - \text{ penalty}$$

Where reward = $s_1 \star p_z$, penalty = $s_0 \star p_z$, s_1 and s_0 represent the similarity rate with X_1 and X_0 respectively, and p_z is a significant percentage of the average objective function of the generation (in the experiments p_z = 0.1 is used).

D) The fitness uses the similarity of the chromosome with X_1 as follows:

$$f(j) = (1 + s_1) \sum_{j=0}^{n} p_j x_j$$

E. Genetic operators

GGA uses standard genetic crossover and mutation operator with as mutation probability of pm. Tournament selection of size 5 is used as the selection method, the random single point method is applied for crossover. For mutation, the mutation by random multiple point bit flip is applied with the probability pm. Finally some of the best chromosomes called as elites of each generation constituting some percentage of top performing chromosomes are send directly to the next generation without applying any of the genetic operator on them. The pseudo-code of the GGA is illustrated in Fig. 3.

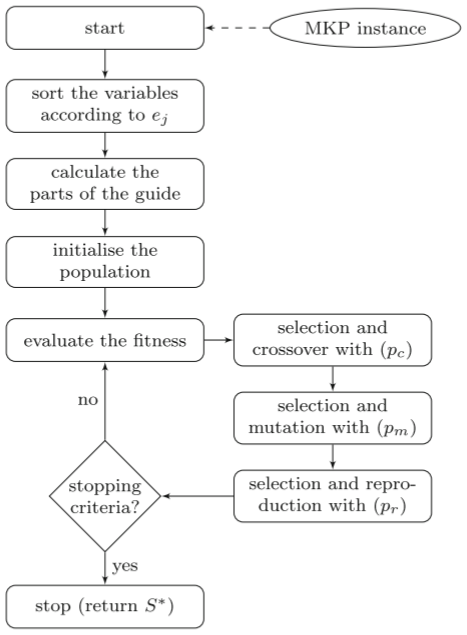
start
MKP instance
sort the variables according to e_j
calculate the parts of the guide
initialise the population
evaluate the fitness
selection and crossover with (p_c)
selection and mutation with (p_m)
selection and reproduction with (p_r)
stopping criteria?
no
yes
stop (return S^*)

Fig. 3. Flowchart of the GGA optimisation process

V. Experimental setup and parameters tuning

This section explains the experimental setup and presents the parameter tuning analysis for the GGA algorithm. It is important to note that the concept of guide can be applied to any problem if an effective method for sorting the problem variables exists. For an experimental purpose, and because the chosen sorting method concerns MKP, it is natural to use data from this problem. The test platform is a Lenevo laptop with 8GB RAM capacity and an Intel Core (TM) i5-9300 H 2.5 Ghz CPU. The C++ language is used to implement the approach. The table shown below describes the various parameters that are used for the simulation of the guided genetic algorithm. And are then set to the experimentally examined values that provides the optimal results.

TABLE I
Parameters of the GGA used to perform the experiments

Parameter	Description	Value
p_s	Population size	100
n_g	Number of generations	10000
p_m	Mutation probability	0.2
α	X1 integration rate on the initial population	0.787
s_t	Tournament Size	5
elt	Percentage of elite chromosomes per generation	0.05
sc	Scale value for Jaccard Similarity	100

VI. Results

We have considered 5 datasets for the examination of the results of GGA using elitism as shown in the Table 2. The table displays the results of GGA on the datasets named as set1- set 5. The Testcase column represents the test case taken. The Optimum Value column describes the real answer of the corresponding set problem. The Before Elitism and After Elitism columns represents the results of GGA without using Elitism and with using Elistism respectively on the corresponding dataset.

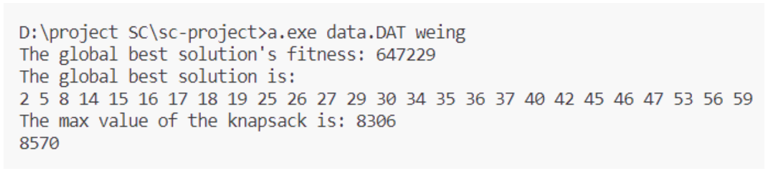
```
D:\project SC\sc-project>a.exe data.DAT weing
The global best solution's fitness: 647229
The global best solution is:
2 5 8 14 15 16 17 18 19 25 26 27 29 30 34 35 36 37 40 42 45 46 47 53 56 59
The max value of the knapsack is: 8306
8570
```

Fig. 4. Results of GGA on the set-1 dataset

TABLE II
Results

Testcase	Optimum Value	Before Elitism	After Elitism
set1.DAT(m=30, n=60)	8570	8138	8264
set2.DAT(m=5, n=90)	11191	10447	10433
set3.DAT(m=10, n=20)	2139	1871	1871
set4.DAT(m=2, n=105)	1095445	844073	919632
set5.DAT(m=5, n=80)	10220	9226	9563

VII. Conclusion

The purpose of the current study was to introduce a memetic algorithm named Guided Genetic Algorithm (GGA) for solving the Multidimensional Knapsack Problem (MKP) and applying elitism on it to improve its performance. GGA analyses the problem data based on a greedy algorithm. Useful information about the items of the MKP are extracted and integrated in the initialisation and evaluation operators of a GA. The research has shown that adding the guidance has significantly improved the performance of the GA and accelerated its speed of convergence. And addint the elitism improes its performance even further. This study confirmed that GGA has a real impact on GA performance. One of the more significant findings of this study is that prior-knowledge about the data of a combinatorial optimisation problem could be significantly helpful to accelerate its solving. The future work intends to extend and evaluate GGA on other combinatorial optimisation problems.

References

[1] Rezoug, A., Bader-El-Den, M. Boughaci, D. Guided genetic algorithm for the multidimensional knapsack problem. Memetic Comp. 10, 29–42 (2018)
[2] Acan A, Tekol Y (2003) Chromosome reuse in genetic algorithms. In: Genetic and evolutionary computation conference, Springer, pp 695–705
[3] Akçay Y, Li H, Xu SH (2007) Greedy algorithm for the general multidimensional knapsack problem. Ann Oper Res 150(1):17
[4] Bader-El-Den M, Gaber M (2012) Garf: towards self-optimised random forests. In: International conference on neural information processing, Springer, pp 506–515

[5] Bader-El-Den M, Poli R (2007) Generating sat local-search heuristics using a gp hyper- euristic framework. In: International conference on artificial evolution (Evolution Artificielle), Springer, pp 37–49

[6] Bader-El-Den M, Poli R, Fatima S (2009) Evolving timetabling heuristics using a grammar-based genetic programming hyperheuristic framework. Memet Comput 1(3):205–219

[7] Balas E, Zemel E (1980) An algorithm for large zero-one knapsack problems. Oper Res 28(5):1130–1154

[8] Baroni MDV, Varejão FM (2015) A shuffled complex evolution algorithm for the multidimensional knapsack problem. In: Iberoamerican congress on pattern recognition, Springer, pp 768–775

[9] Castro F, Gelbukh A, González M (2013) Advances in Soft Computing and Its Applications. In: 12th Mexican international conference, MICAI 2013, Mexico City, Mexico, November 24-30, 2013, proceedings. No. pt. 2 in lecture notes in computer science, Springer Berlin Heidelberg, https://books.google.com.om/books?id=WgC6BQAAQBAJ

[10] Chen SH, Chang PC, Cheng T, Zhang Q (2012) A self-guided genetic algorithm for permutation flowshop scheduling problems. Comput Oper Res 39(7):1450–1457

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

[12] Dantzig GB (1957) Discrete-variable extremum problems. Oper Res 5(2):266–288

[13] Dobson G (1982) Worst-case analysis of greedy heuristics for integer programming with nonnegative data. Math Oper Res 7(4):515–531

[14] Drake JH, Özcan E, Burke EK (2015) Modified choice function heuristic selection for the multidimensional knapsack problem. In: Genetic and evolutionary computing, Springer, pp 225–234

[15] . Fatima S, Bader-El-Den M (2010) Co-evolutionary hyper-heuristic method for auction based scheduling. In: Evolutionary computation (CEC), 2010 IEEE congress on, IEEE, pp 1–8

[16] Gabaldon E, Lerida JL, Guirado F, Planes J (2014) Slowdownguided genetic algorithm for job scheduling in federated environments. In: International conference on nature of computation and communication, Springer, pp 181–190

[17] Hill RR, Cho YK, Moore JT (2012) Problem reduction heuristic for the 0–1 multidimensional knapsack problem. Comput Oper Res 39(1):19–26

[18] Huston S, Puchinger J, Stuckey P (2008) The core concept for 0/1 integer programming. In: Proceedings of the fourteenth symposium on Computing: the Australasian theory-Volume 77, Australian Computer Society, Inc., pp 39–47

[19] Kellerer H, Pferschy U, Pisinger D (2004) Introduction to NPcompleteness of knapsack roblems. Springer, Berlin, Heidelberg, pp 483–493

[20] Magazine M, Oguz O (1984) A heuristic algorithm for the multidimensional zero-one knapsack problem. Eur J Oper Res 16(3):319–326

[21] Perry T, Bader-El-Den M, Cooper S (2015) Imbalanced classification using genetically optimized cost sensitive classifiers. In: Evolutionary computation (CEC), 2015 IEEE congress on, IEEE, pp 680–687

[22] Pirkul H (1987) A heuristic solution procedure for the multiconstraint zero? one knapsack problem. Naval Res Logist 34(2):161–172

[23] Poli R, Koza J (2014) Genetic programming. In: Search methodologies, Springer, pp 143–185

[24] Puchinger J, Raidl GR, Pferschy U (2006) The core concept for the multidimensional knapsack problem. Springer, Berlin

[25] Rasheed K (1999) Guided crossover: a new operator for genetic algorithm based optimization. In: Evolutionary computation, 1999. CEC 99. Proceedings of the 1999 congress on, IEEE, vol 2, pp 1535–1541

[26] Rezoug A, Boughaci D (2016) A self-adaptive harmony search combined with a stochastic local search for the 0–1 multidimensional knapsack problem. Int J Bio Inspired Comput 8(4):234–239

[27] Rezoug A, Boughaci D, Bader-El-Den M (2015) Memetic algorithm for solving the 0-1 multidimensional knapsack problem. In: Portuguese conference on artificial intelligence, Springer, pp 298–304

[28] Senju S, Toyoda Y (1968) An approach to linear programming with 0-1 variables. Management Science pp B196–B207

[29] Snášel V, Dvorsk ‘y J, Ochodková E, Krömer P, Platoš J, Abraham A (2010) Evolving quasigroups by genetic algorithms. In: Proceedings of DATESO, Citeseer, pp 108–117

[30] Sudholt D (2015) Parallel evolutionary algorithms. In: Springer handbook of computational intelligence, Springer, pp 929–959

[31] Tang KS, Man KF, Kwong S, He Q (1996) Genetic algorithms and their applications. IEEE Signal Process Mag 13(6):22–37

[32] Vázquez-Barreiros B, Mucientes M, Lama M (2014) A genetic algorithm for process discovery guided by completeness, precision and simplicity. In: International conference on business process management, Springer, pp 118–133

[33] Volgenant A, Zoon J (1990) An improved heuristic for multidimensional 0–1 knapsack problems. J Oper Res Soc 41(10):963–970

[34] Yang S, Jat SN (2011) Genetic algorithms with guided and local search strategies for university course timetabling. IEEE Trans Syst Man Cybern Part C (Appl Rev) 41(1):93–106

[35] Zhang Q, Sun J, Tsang E (2005) An evolutionary algorithm with guided mutation for the maximum clique problem. IEEE Trans Evolut Comput 9(2):192–200