Knowledge-Based Genetic Algorithm for the 0-1 Multidimensional Knapsack Problem

Abdellah Rezoug
Department of Computer Science
University Mhamed Bougara of Boumerdes
Boumerdes, Algeria
Email: abdellah.rezoug@gmail.com

Mohamed Bader-El-Den
School of Computing
University of Portsmouth
Portsmouth, United Kingdom
Email: mohamed.bader@port.ac.uk

Dalila Boughaci
Department of Computer Science
University of Science and Technology
Houari Boumediene, Algiers, Algeria
Email : dalila_info@yahoo.fr

***Abstract*—This paper presents an improved version of Genetic Algorithm (GA) to solve the 0-1 Multidimensional Knapsack Problem (MKP01), which is a well-known NP-hard combinatorial optimisation problem. In combinatorial optimisation problems, the best solutions have usually a common partial structure. For MKP01, this structure contains the items with a high values and low weights. The proposed algorithm called Genetic Algorithm Guided by Pretreatment information (GAGP) calculates these items and uses this information to guide the search process. Therefore, GAGP is divided into two steps, in the first, a greedy algorithm based on the efficiency of each item determines the subset of items that are likely to appear in the best solutions. In the second, this knowledge is utilised to guide the GA process. Strategies to generate the initial population and calculate the fitness function of the GA are proposed based on the pretreatment information. Also, an operator to update the efficiency of each item is suggested. The pretreatment information has been investigated using the CPLEX deterministic optimiser. In addition, GAGP has been examined on the most used MKP01 data-sets, and compared to several other approaches. The obtained results showed that the pretreatment succeeded to extract the most part of the important information. It has been shown, that GAGP is a simple but very competitive solution.**

I. Introduction

The 0-1 Multidimensional Knapsack Problem (MKP01) is composed of N items and a knapsack with m different capacities c_i where $i \in \{1, \ldots, m\}$. Each item j where $j \in \{1, \ldots, n\}$ has a profit p_j and can take w_{ij} of the capacity i of the knapsack. The goal is to pack the items in the knapsack so as to maximise the profits of items without exceeding the capacities of the knapsack. The MKP01 can be represented as the following integer program:

$$\textbf{Maximise}: \quad \sum_{j=1}^{n} p_j x_j \tag{1}$$

$$\textbf{Subject to}: \sum_{j=1}^{n} w_{ij} x_j \leq c_i \qquad i \in \{1 \ldots m\} \tag{2}$$

$$x_j \in \{0, 1\} \qquad j \in \{1 \ldots n\} \tag{3}$$

During the past few decades several variants of GA have been proposed, all of them aim to increase the performance of GA and boost its convergence. Most of these ideas are either based on changing the GA operators such as: crossover and mutation (e.g. one-point, two-point, cut and splice, three parents, uniform, flip bit, Boundary, non-uniform, uniform, etc.), or based on modifying the GA's evolutionary behaviour, such as: Distributed GA [1], Hybrid GA ([2], [3]), Parallel GA [4],Adaptive GA [5], Genetic Programming [6], etc. An extended overview of the GA variations is available in [7]. This work focuses on on the GA versions that implement the concept of guide.

The concept of *proximate optimality* suggests that, in most cases, the best solutions have a similar structure. [8] presented a primal greedy gradient algorithm for the MKP01 that establishes a decreasing sort of the items such as the most priority is given to those most likely to form the best solution. The sort is calculated according to an *efficiency measurement* that try to find the compromise between the profit and the weight. Latter [9] applied the principal of *efficiency measurement* in addition to the *core concept* for reducing the size of the problem data to only the most relevant items. On the other hand, the process of GA is stochastic, this leads to an important useless work.

The aim in this paper is to reinforce the GA process using the useful information about the items. To this purpose, the Genetic Algorithm Guided by Pretreatment information (GAGP) is proposed. Firstly, GAGP applies the primal greedy with the core concept decomposition to extract a useful information about the subset of important items. Secondly, specific population initialisation, fitness function and update efficiency measurement operators augment a standard GA by exploiting the pretreatment information. In GAGP, an important rang of solutions are avoided and the process does not consider the non relevant solutions.

The paper is structured as follows: Section II gives an overview of the literature review related to the GA with guidance. The proposed algorithm GAGP is introduced in Section III. Section IV presents the conducted experiments and the obtained results. The conclusions and final remarks are drawn in Section V.

II. Related Works

There are several methods related to the guided GA concept in the literature, that has been applied to a wide range of applications. For solving the Course Timetabling Problem, the approaches by ([10], [11]) use a memory denoted MEM to record useful information to guide the GA process and improve its performance. MEM is a list of limited size, in which a list of room and time slot pairs is recorded. This information is integrated into the crossover operator of the proposed guided GA. Other researchers used an external structure to guide GA such as ([12], [13]). Another approach for guiding the GA is through the use of approximate probabilistic models. Also, GA and other evolutionary algorithms have been developed for a wide range of problem where the problem domain information is embedded in the algorithm [14], [15], [16], [17].

In ([18], [19]) The GA is augmented with an approximate probabilistic model to guide the crossover and mutation operators. The probabilistic model is used to estimate the quality of candidate solutions generated by the traditional crossover and mutation operators. It also evaluates the quality of candidate solutions. This estimation enables the crossover and mutation operators to generate more promising solutions.

A subset of the genetic operators is guided. The proximate optimality principle assumes that good solutions have a similar structure. Based on this principle, the guided mutation proposed by [20] uses a probability model inspired by estimation of distribution algorithms EDA mutation operator. The generated offspring by this operator is constructed based on the best parent so far and a dynamic probability model and a probability β. This allows conducting the searching process in promising areas.

A guided crossover operator has been proposed by [21]. The crossover operator works by using guidance from all members of the GA population to select a direction for exploration. The first parent is selected by the selection operator. To select the second parent, a metric named $Mutual\_fitness$ is calculated for all the other chromosomes. The chromosome which has the maximum value is selected. One offspring is generated by crossing the parents in a point chosen randomly such that the offspring resulting is the best.

The guidance methods in these GA variants are specific to the addressed problems, they do not propose a formal way to extract the guidance information or are integrated to the optimisation process. Some approaches incorporate a partial guidance using genetic operators.

III. Genetic Algorithm Guided by pretreatment information for The MKP01

The algorithm in this paper is motivated by the observation that in may optimisation real-world problem, we may have some prior information about the components/patterns that are likely to appear in the good solutions. For example, in MKP01, it is possible using linear relaxation or the "optimal fractional solution" ([22], [23]) 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 MKP01 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 GAGP components.

A. Chromosome design

The population is composed of a finite number of chromosomes. A chromosome represents a feasible solution to the problem (MKP01). As mentioned before, the target in the MKP01 is to define the subset of items that maximises the total profit. The GAGP chromosome consists of the set of the items to be added to the knapsack. GAGP uses the integer representation, where each gene presents an item ID. The items are coded as integer numbers. A chromosome is formed only by the number of items that it contains. This representation allows reducing the size of the processed data (Fig. 1).

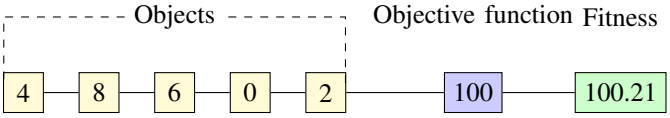

Fig. 1. Example of the the chromosome design.

B. Pretreatment

The guiding information is based on the work by [9]. The items are sorted in decreasing order according to a statistical efficiency e_j based on the profit and the cost. In simple words, the items are sorted based on how likely each item to appear in high performing individuals, the item at the top of this list are the items that are likely to be selected while the items at the bottom of the list are the items that are unlikely 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 be noted also that the Greedy heuristic [8] is only based on the efficiency sorting is not an effective solution for the strongly correlated problem instances of the MKP01 [24]. The efficiency is measured according to Eq. 4 as presented in [8]:

$$e_j = \frac{p_j}{\sum_{j=1}^{m} w_{ij}(\sum_{l=1}^{n} w_{il} - c_i)} \tag{4}$$

The sorting operation allows favouring items that have a good compromise (i.e. efficiency) between the average profit and overall capacity. The efficiency of an item is high if its profit is high while its required global capacity is low. The sorted items are split into three sets 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 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 profit is low or the capacity is large or both.

The guide is represented by the items 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 the 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 $Core$. Set the size of $Core$ defines the size of the other parts.

C. Guided Genetic Algorithm Optimisation

In the pretreatment step, GAGP classifies the items into three subsets, a subset of the items that are likely to be packed (i.e. X_1), a subset of the items that are unlikely to be packed (i.e. X_0) and a third subset of the items that are slightly similar with medium score cost/profit (i.e. $Core$). The items of X_1 and $Core$ are integrated in the optimisation step of GAGP. Therefore, the population is initialised using the items of X_1 with a probability α. Also, the fitness of each chromosome is evaluated according to its objective function value and the overall efficiency of its items. Then, the selection operator chose items for crossings and mutations according to their fitness values.

1) *Initial population.* GAGP 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 $\alpha = 1$ means that each individual in the initial population contains all the items in 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.
2) *Fitness evaluation.* Besides the population initialisation, the guidance by the pretreatment information is integrated in the GA by this operator. The fitness function $f(j)$ is evaluated according to Eq. 1. The efficiency e_j is introduced in its evaluation according to Eq. 5. Each generation, the fitness of each individual is measured by multiplying the overall efficiency and profit of its items. If an individual is composed mainly of items included in X_1 or $Core$, then, the overall efficiency of its items is high. Therefore, its fitness function will be high. Similarly, if an individual contains mainly items included in X_0, then, its fitness will be low. That means that individuals having a high similarity rate with X_1 or $Core$ are favoured to be selected in the next generations of the evolution process. The fitness formula allows giving more chance to the chromosome that has a high efficiency to be selected more than the others.

$$f(j) = \sum_{j=0}^{n} e_j p_j x_j \tag{5}$$

3) *Genetic operators.* GAGP uses standard genetic crossover and mutation operators. A tournament selection of size 5 is used as the selection method, and the random single point method is applied with a probability p_c as a crossover method. For the mutation operator, the random multiple point bit flip with the probability p_m is adopted. And finally, a reproduction operator copies a subset of individuals with the probability p_r such as $p_c + p_m + p_r = 1$.
4) *Update efficiency.* The Sorting efficiency is not always efficient especially for the problems with strong correlation. A step of efficiency update is proposed that aims to make a perturbation in the items efficiencies. Two items j, j', $j \in X_1$ and are selected and their efficiency is permuted. Rather than maintaining the same guidance, the search process diversify the guide with the items in X_1 and $Core$. This modification has an impact on the fitness evaluation and so on the whole process of GA.
5) *Stopping condition* The process of optimisation is repeated until a specific number of iteration is reached.

The algorithm could be described by the following pseudo-code (Algorithm 1).

Algorithm 1 The GAGP pseudo-code.

Require: MKP01 instance
Ensure: a feasible solution S
1: calculate the efficiency e_j for each variable
2: sort the items according to the efficiency measurement
3: calculate X_1, $Core$ and X_0 of the guide
4: initialise the population pop with X_1 and α
5: **for** $ctr = 1$ to ng **do**
6: evaluate the fitness for each chromosome in pop according to the fitness equation
7: crossover with (pc)
8: mutation with (pm)
9: reproduction with (pr)
10: select randomly items j, j' such as $j \in X_1$, $j' \in Core$ and permute their efficiencies
11: **end for**
12: return the best solution S^*.

IV. Experimental Results

The experiments aims to compare the proposed GAGP with the state-of-art results reported in the literature (Section IV-G). For an experimental purpose, and because the chosen sorting method concerns MKP01, it is natural to use data from this problem. The test platform is a Toshiba laptop with 4GB RAM capacity and an Intel Core (TM) i5-4200 M 2.5 Ghz CPU. The Java language is used to implement the approach.

A. Test data-sets

The data utilised to undertake the tests are composed of 270 MKP01 instances. This data is divided into 9 classes, each contains 30 instances. The number of items ranges in $\{5, 10, 30\}$ while the number of constraints ranges in $\{100, 250, 500\}$. The data was proposed in [25] and are available on-line on the OR-Library[1]. The optimal solution for most of the instances are known, while it is still unknown for some difficult instances.

B. Algorithm parameters

The parameters of a heuristic approach may determine its effectiveness. The GAGP contains many parameters and to determine their values many experiments have been conducted. Table I summaries the preferred values of the parameters for GAGP:

TABLE I
THE PARAMETERS VALUES OF GAGP

parameter	description	value
ng	number of generation	500
ps	population size	500
pc	crossover probability	0.2
pm	mutation probability	0.7
pr	reproduction probability	0.1
α	rate of X_1 integration on the initial population	0.9
δ	$Core$ size	$0.15n$
st	selection tour	10
mp	number of mutation points	3
pp	number of permutation points	1

C. Core size

This experiment aim to determine the best size of the *Core*. Also, it allows to determine whether enlarging the size I_1 is likely or not to include more items of the optimal solutions. The experiment consists in applying the CPLEX solver the *Core* of the 270 instances, and measuring the Average Distance From the Optimum ($A.D.F.O$) (the optimum is the best-known solution of each instance). this experiment allows also to determine the position of the best items after sorting is performed.

The results reported in Table II summary the obtained $A.D.F.O$ with $\delta = \{10, 0.1n, 0.15n, 0.2n\}$ with n is the number of items. The results indicated that the larger is δ the better is the $A.D.F.O$. Also, it revealed that the optimal solution, after the sort, was probably gathered in a subset of 20% of the items. This could means, in our case to include more items from the *Core* in I_1.

D. Impact of Seed on the GAGP

A part of the chromosomes of the initial population (IP) is generated randomly. A value of $Seed$ is defined to have the same quality of IP in each run. In order to evaluate its effect on the GA-Guided, five values are examined. Fig. 2 shows that $Seed$ has no great influence on the result. The results also shows that $Seed$ = 4, 6 or 8 gives the best solutions.

[1]http://people.brunel.ac.uk/~mastjjb/jeb/orlib/files/

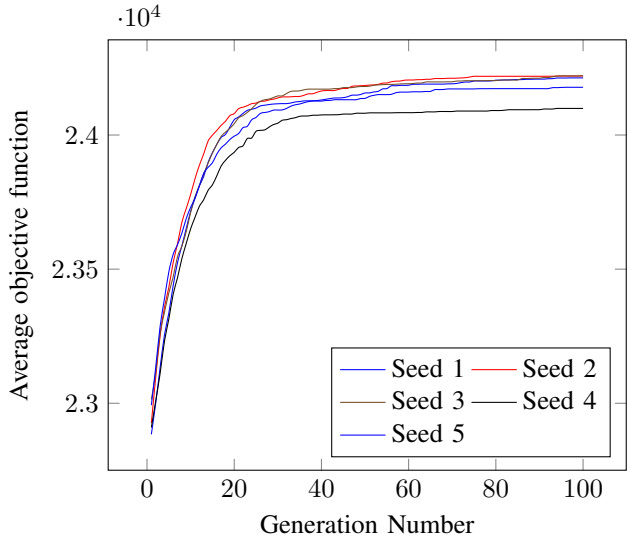

Fig. 2. Comparing the GAGP convergence with five values of $Seed$

E. Analytical study of the guidance

This analytical study compares between the items of the optimal solution and the two main parts of the guide (i.e. X_1 and $Core$). In addition, it compares between the optimal solution and the solution obtained by GAGP. Therefore, this study aims to understand how significant is the sort and measure its effective impact on the GA. Also, it aims to find out if the GAGP does effectively follow the guide. The results allow to know what drops the GA in the wrong solution, whether the guide or the optimisation process itself.

The composition of the optimal solution S^*, calculated using the deterministic CPLEX Optimizer 12.5, is compared to a feasible solution S obtained by GAGP. Also, the items of the X_1 and $Core$ and the placement of the items of the S^* in the three parts of the guide are given (where +, * and - corresponds to item in X_1, in $Core$ and in X_0 respectively). The first four instances OR5x100.0.25_1-4 are used to conduct this analysis. Finally, the Average Distance From the Optimum $A.D.F.O$ of the solution calculated by GAGP is given. The obtained results of the comparison are reported in Table III.

A percentage of $75-90\%$ of the items in S^* are included in X_1 or $Core$. Similarly, S contained $75-90\%$ of the items of S^*. Almost the same items initially contained in X_1 and $Core$ are maintained in S. Some (3 to 7) items of the excluded part X_0 appear in S^* at the same time some were introduced in S by the mutation operator. In the first three instances, at most one item form X_1 has not been contained in S^*. GAGP could be more effective by introducing a better mutation operator. The efficiency measurement function would be more effective if $Core$ contained a slightly more items of X_0. Most items of $Core$ were components of S^*, that supports the efficiency

TABLE II
AVERAGE DISTANCE FROM THE OPTIMUM $A.D.F.O$ OF THE CPLEX APPLICATION ON THE *Core* FOR DIFFERENT SIZE δ ON ALL THE DATA.

	α	$\delta = 10$	$\delta = 0.1n$	$\delta = 0.15n$	$\delta = 0.2n$
$t = 5$					
5	0.25	14.446	7.589	3.830	**1.355**
	0.5	5.605	2.885	1.555	**0.637**
	0.75	1.968	0.826	0.440	**0.158**
10	0.25	9.777	5.784	3.239	**1.679**
	0.5	4.555	2.780	1.598	**1.055**
	0.75	2.127	1.234	0.793	**0.393**
30	0.25	6.316	4.234	2.817	**1.672**
	0.5	4.664	3.280	2.264	**1.646**
	0.75	2.111	1.319	0.832	**0.508**
$t = 10$					
5	0.25	14.446	7.589	3.830	**1.354**
	0.5	5.605	2.885	1.555	**0.637**
	0.75	1.968	0.826	0.440	**0.158**
10	0.25	9.777	5.784	3.239	**1.677**
	0.5	4.555	2.780	1.597	**1.052**
	0.75	2.127	1.234	0.792	**0.392**
30	0.25	6.316	4.234	2.817	**1.662**
	0.5	4.664	3.280	2.264	**1.642**
	0.75	2.111	1.319	0.831	**0.507**
$t = 50$					
5	0.25	14.446	7.589	3.830	**1.350**
	0.5	5.605	2.885	1.554	**0.636**
	0.75	1.968	0.826	0.439	**0.158**
10	0.25	9.777	5.784	3.239	**1.672**
	0.5	4.555	2.780	1.596	**1.049**
	0.75	2.127	1.234	0.792	**0.391**
30	0.25	6.316	4.234	2.817	**1.661**
	0.5	4.664	3.280	2.264	**1.637**
	0.75	2.111	1.319	0.831	**0.503**
$t = 100$					
5	0.25	14.446	7.589	3.830	**1.350**
	0.5	5.605	2.885	1.554	**0.636**
	0.75	1.968	0.826	0.439	**0.158**
10	0.25	9.777	5.784	3.239	**1.669**
	0.5	4.555	2.780	1.596	**1.048**
	0.75	2.127	1.234	0.792	**0.391**
30	0.25	6.316	4.234	2.817	**1.661**
	0.5	4.664	3.280	2.264	**1.637**
	0.75	2.111	1.319	0.831	**0.503**

update operator proposed in GAGP.

F. Performance of GAGP compared to GA

A comparison between GAGP and a standard version of GA was conducted to measure the contribution of the pre-analysis information on the convergence of GA. GAGP and GA both were executed 30 times on some instances. The obtained objective function values of each algorithm was recorded. The average objective function of both approaches is compared in Fig. 3. The lower and upper whiskers show the worst and best results achieved from 30 independent run times. The box shows the lower and upper quartiles, while the line in the middle box shows the median value. The results indicated that GAGP outperformed GA throughout the evolutionary process.

G. Comparison with the literature

As with most optimisation problems, MKP01 heuristics could be classified in two groups: the first is*constructive* heuristics, that aim to construct a solution. The second is *improvement* heuristics which aim to improve a given initial solution normally generated first by a *constructive* heuristic. The proposed method is considered as a *constructive* heuristic. However, in order to demonstrate the performance of the proposed method, the performance of the GAGP is compared with both *constructive* and *improvement* approaches. The following is short description of the methods (*constructive* and *improvement*) used in the comparison presented in this section. GAGP is compared with the standard GA algorithm and other

TABLE III

ANALYTICAL COMPARISON OF GAGP FEASIBLE SOLUTION COMPOSITION TO THE COMPOSITION OF THE OPTIMAL SOLUTION OBTAINED BY CPLEX AND THE COMPOSITION OF THE PARTS OF THE GUIDANCE INFORMATION USING THE OR5X100.0.25 1-4 INSTANCES. S : ITEMS OF THE SOLUTION OBTAINED BY GAGP. S* : ITEMS OF THE OPTIMAL SOLUTION OBTAINED BY CPLEX. G : GROUP OF S* ITEM IN THE GUIDE (ITEM IS : + $\in X_1$, - $\in X_0$ OR * $\in Core$). $A.D.F.O$: AVERAGE DISTANCE FORM THE OPTIMUM IN % OF THE GAGP SOLUTION

S	S*	G	X_1	S	S*	G	X_1	S	S*	G	X_1	S	S*	G	X_1
1	1	+	**1**	1	3	+	**3**	7	4	+	**4**	**0**	0	*	**3**
3	3	*	4	**3**	10	+	**10**	10	11	+	**11**	**1**	1	*	**5**
6	6	+	**6**	**10**	18	*	**27**	**11**	13	+	**13**	**3**	3	+	**11**
8	8	+	**8**	**20**	20	*	**28**	**13**	18	+	**18**	**5**	5	+	12
10	10	-	**23**	**27**	27	+	**34**	**18**	19	-	**21**	6	8	-	**22**
12	18	*	**26**	**34**	28	+	**45**	**21**	21	+	**28**	**8**	11	+	**27**
17	23	+	**31**	**36**	34	+	**49**	**28**	26	-	**37**	**11**	13	-	28
23	25	-	**43**	39	36	-	**56**	**32**	28	+	**44**	12	22	+	**30**
26	26	+	**49**	**41**	41	*	**57**	**34**	32	*	**48**	**22**	24	-	**34**
28	28	*	**56**	**42**	42	*	**61**	36	34	-	**55**	**24**	27	+	**35**
29	29	*	**62**	**45**	45	+	**62**	**37**	37	+	69	26	30	+	**42**
31	31	+	**65**	**48**	48	*	**73**	**42**	42	*	**72**	**27**	34	+	**53**
41	43	+	**68**	**49**	49	+	**90**	**44**	44	+	**74**	**30**	35	+	**63**
43	49	+	**76**	**53**	53	-	**91**	**48**	48	+	**84**	**34**	42	+	69
49	56	+	**78**	**56**	56	+	**93**	**51**	51	-	**87**	**35**	49	*	70
56	61	-	**85**	**57**	57	+	**95**	53	55	+	**92**	**42**	53	+	**78**
62	62	+	**92**	**58**	58	-	**99**	**55**	59	*		**49**	54	-	**86**
65	65	+		**61**	61	+		**59**	60	-		**53**	55	-	**94**
68	68	+	Core	**62**	62	+	Core	**60**	64	*	Core	**54**	56	*	
76	70	-	**3**	70	64	*	**18**	**64**	72	+	10	**55**	58	*	Core
78	73	*	15	**73**	73	+	**20**	**72**	74	+	**32**	**56**	61	*	**0**
84	76	+	**18**	**74**	74	*	**41**	**74**	78	*	36	**61**	63	+	**1**
85	78	+	**28**	**81**	81	*	**42**	**78**	79	-	**42**	**63**	74	-	15
91	84	-	**29**	**88**	88	*	**48**	**79**	84	+	**59**	68	78	+	**49**
92	85	+	34	**90**	90	+	**64**	**84**	87	+	**64**	70	79	-	**56**
94	91	-	66	**91**	91	+	**74**	**87**	92	+	**78**	76	86	+	**58**
95	92	+	**73**	92	93	+	**81**	**92**	93	*	**93**	**79**	94	+	**61**
98	95	-	81	**93**	95	+	**88**	**93**	96	*	**96**	**86**	95	+	65
	98	+	**98**	**95**	99	+	92	**96**	99	*	**99**	**95**			68
				99											**95**
$A.D.F.O$ = 0.82				$A.D.F.O$ = 0.51				$A.D.F.O$ = 0.23				$A.D.F.O$ = 0.50			

state-of-the-art optimisation methods reported in the literature. GAGP is compared to the following constructive approaches : PECH (Primal Effective Capacity Heuristic) [26]; MAG [27]; VZ [28]; PIR (Pirkul 1987) and SCE (Shuffled Complex Evolution) [29]. GAGP is also compared to the following improvement approaches : CB [25]; NR (P) (New Reduction (Pirkul)) [30] and MCF (Modified Choice Function - Late Acceptance Strategy) [31]. The comparison is shown in Table IV. The approaches are compared in terms of $A.D.F.O$ and all the instances of the Chu&Beasley data are included. The overall best $A.D.F.O$ are mentioned in bold and star whereas the best $A.D.F.O$ per category of heuristic is mentioned in bold only. As shown in table IV, GAGP is competitive with both construction and improvement methods and has managed to outperform both group of methods on a few instances.

V. CONCLUSION

This paper aims to present a modified version of GA. Extracted information about the variables likely to appear in the best solutions are used to guide the search process of GA. The approach called Genetic Algorithm Guided by Pretreatment information (GAGP) begins by analysing the problem data using a gradient greedy sorting method which sorts the variables according to an efficiency value expressed by profit and cost. These information are used to drive the GA search process by its integration in the generation of the initial population and for measuring the fitness function. Some experiments were conducted using a set of well-known MKP01 data. It has been shown that the information improves the performance of GA. The pretreatment allows to reduce the size of the problem to only the most relevant space of solutions, this allows the search process to avoid the areas of worst solutions. In addition, the results obtained in the resolution of MKP01 are competitive. As prospects for the next step, we expect to apply the method to other optimisation problems in different domains e.g. classification [32], [33], [34] and domain specific scheduling [35] optomisation [36].

REFERENCES

[1] H. Adeli and S. Kumar, "Distributed genetic algorithm for structural optimization," *Journal of Aerospace Engineering*, vol. 8, no. 3, pp. 156–163, 1995.

[2] A. Rezoug, D. Boughaci, and M. Bader-El-Den, "Memetic algorithm for solving the 0-1 multidimensional knapsack problem," in *Portuguese Conference on Artificial Intelligence*. Springer, 2015, pp. 298–304.

[3] V. Valls, F. Ballestin, and S. Quintanilla, "A hybrid genetic algorithm for the resource-constrained project scheduling problem," *European Journal of Operational Research*, vol. 185, no. 2, pp. 495–508, 2008.

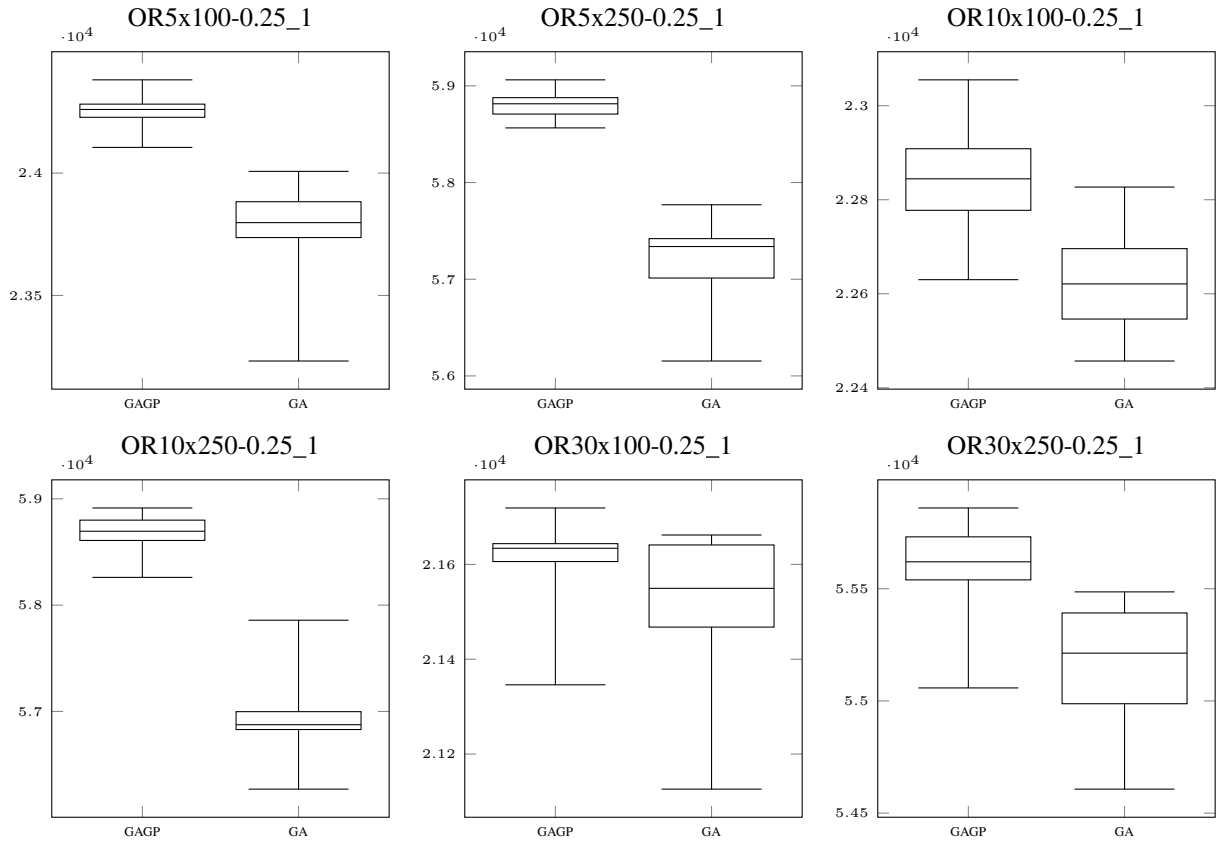

Fig. 3. The objective function values rang obtained by GAGP compared with GA within 30 run

TABLE IV
COMPARISON OF RESULTS OBTAINED BY GAGP WITH GA, CONSTRUCTIVE AND IMPROVEMENT HEURISTICS

			Constructive							Improvement		
n	m	α	GAGP	GA	PECH	MAG	VZ	PIR	SCE	CB	NR(P)	MCF
5	100	0.25	**0.35***	2.17	7.3	13.6	10.3	1.6	3.5	0.99	0.94	1.09
		0.50	**0.48**	0.86	3.4	6.7	6.9	0.77	2.6	0.45	**0.44***	0.57
		0.75	**0.21***	0.42	2.02	5.1	5.6	0.48	1.1	0.32	**0.22**	0.38
	250	0.25	0.58	4.03	7.1	6.6	5.8	**0.53**	4.3	**0.23***	0.46	0.41
		0.50	0.36	1.15	3.2	5.2	4.4	**0.24**	3.3	**0.12***	0.17	0.22
		0.75	0.23	0.58	1.8	3.5	3.5	**0.16**	1.5	**0.08***	0.1	0.14
	500	0.25	0.51	4.27	6.4	4.9	4.1	**0.22**	4.6	1.56	**0.15***	0.21
		0.50	0.36	1.45	3.4	2.9	2.5	**0.08**	3.6	0.79	**0.06***	0.1
		0.75	0.22	0.65	1.7	2.3	2.41	**0.06**	1.8	0.48	**0.03***	0.06
10	100	0.25	**1.0**	2.40	8.2	15.8	15.5	3.4	6.8	**0.09***	2.05	1.87
		0.50	**0.53**	1.53	3.7	10.4	10.7	1.8	5.1	**0.04***	0.81	0.95
		0.75	**0.27**	0.53	1.8	6.1	5.67	1.1	2.4	**0.03***	0.44	0.53
	250	0.25	**0.75**	3.56	5.8	11.7	10.5	1.1	6.9	**0.51***	0.88	0.79
		0.50	0.48	1.35	2.5	6.8	5.9	**0.57**	5.4	**0.25***	0.39	0.41
		0.75	**0.27**	0.66	1.5	4.4	3.7	0.33	2.8	**0.15***	0.19	0.24
	500	0.25	0.71	3.61	5.1	8.8	7.9	**0.52**	6.8	**0.24***	0.34	0.44
		0.50	0.4	1.44	2.4	5.7	4.1	**0.22**	5.8	**0.11***	0.14	0.2
		0.75	0.29	0.71	1.2	3.6	2.9	**0.14**	3.4	**0.07***	0.1	0.13
30	100	0.25	**1.56***	2.27	6.8	17.3	17.2	9.1	8.6	2.91	2.24	3.61
		0.50	**1.07***	1.72	3.2	11.8	10.1	3.51	6.6	1.34	1.32	1.6
		0.75	**0.36***	0.78	1.9	6.58	5.9	2.03	3.6	0.83	0.8	0.97
	250	0.25	**1.66**	3.20	4.8	13.5	12.4	3.7	8.3	**1.19***	1.27	1.75
		0.50	**1.0**	1.46	2.1	8.6	7.1	1.5	6.9	**0.53***	0.75	0.79
		0.75	**0.5**	0.73	1.2	4.4	3.9	0.84	3.8	**0.31***	0.38	0.43
	500	0.25	4.07	3.50	3.7	9.8	9.6	**1.89**	8.6	**0.61***	0.89	1.05
		0.50	2.14	1.45	1.7	7.1	5.7	**0.73**	7.4	**0.26***	0.36	0.44
		0.75	0.51	0.69	0.9	3.7	3.5	**0.48**	4	**0.17***	0.23	0.27

[4] D. Sudholt, "Parallel evolutionary algorithms," in *Springer Handbook of Computational Intelligence*. Springer, 2015, pp. 929–959.

[5] M. Alavidoost, M. Tarimoradi, and M. F. Zarandi, "Fuzzy adaptive genetic algorithm for multi-objective assembly line balancing problems," *Applied Soft Computing*, vol. 34, pp. 655–677, 2015.

[6] R. Poli and J. Koza, "Genetic programming," in *Search Methodologies*. Springer, 2014, pp. 143–185.

[7] F. Castro, A. Gelbukh, and M. González, *Advances in Soft Computing and Its Applications: 12th Mexican International Conference, MICAI 2013, Mexico City, Mexico, November 24-30, 2013, Proceedings*, ser. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2013, no. pt. 2. [Online]. Available: https://books.google.com.om/books?id=WgC6BQAAQBAJ

[8] S. Senju and Y. Toyoda, "An approach to linear programming with 0-1 variables," *Management Science*, pp. B196–B207, 1968.

[9] J. Puchinger, G. R. Raidl, and U. Pferschy, *The Core Concept for the Multidimensional Knapsack Problem*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 195–208.

[10] S. N. Jat and S. Yang, "A guided search genetic algorithm for the university course timetabling problem," pp. 180–191, 2009.

[11] S. Yang and S. N. Jat, "Genetic algorithms with guided and local search strategies for university course timetabling," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 41, no. 1, pp. 93–106, 2011.

[12] A. Acan and Y. Tekol, "Chromosome reuse in genetic algorithms," in *Genetic and Evolutionary Computation Conference*. Springer, 2003, pp. 695–705.

[13] S. Louis and G. Li, "Augmenting genetic algorithms with memory to solve traveling salesman problems," in *Proceedings of the Joint Conference on Information Sciences*, 1997, pp. 108–111.

[14] M. Bader-El-Den and R. Poli, "Generating sat local-search heuristics using a gp hyper-heuristic framework," in *International Conference on Artificial Evolution (Evolution Artificielle)*. Springer, 2007, pp. 37–49.

[15] M. Bader-El-Den, R. Poli, and S. Fatima, "Evolving timetabling heuristics using a grammar-based genetic programming hyper-heuristic framework," *Memetic Computing*, vol. 1, no. 3, pp. 205–219, 2009.

[16] B. Aziz, M. Bader, and C. Hippolyte, "Search-based sql injection attacks testing using genetic programming," in *European Conference on Genetic Programming*. Springer, 2016, pp. 183–198.

[17] M. M. Gaber and M. B. Bader-El-Den, "Optimisation of ensemble classifiers using genetic algorithm." in *KES*, 2012, pp. 39–48.

[18] S.-H. Chen, P.-C. Chang, T. Cheng, and Q. Zhang, "A self-guided genetic algorithm for permutation flowshop scheduling problems," *Computers & Operations Research*, vol. 39, no. 7, pp. 1450–1457, 2012.

[19] S.-H. Chen, P.-C. Chang, Q. Zhang, and C.-B. Wang, "A guided memetic algorithm with probabilistic models," *International Journal of Innovative Computing, Information and Control*, vol. 5, no. 12, pp. 4753–4764, 2009.

[20] Q. Zhang, J. Sun, and E. Tsang, "An evolutionary algorithm with guided mutation for the maximum clique problem," *IEEE Transactions on Evolutionary Computation*, vol. 9, no. 2, pp. 192–200, 2005.

[21] K. Rasheed, "Guided crossover: A new operator for genetic algorithm based optimization," in *Evolutionary Computation, 1999. CEC 99. Proceedings of the 1999 Congress on*, vol. 2. IEEE, 1999, pp. 1535–1541.

[22] G. B. Dantzig, "Discrete-variable extremum problems," *Operations research*, vol. 5, no. 2, pp. 266–288, 1957.

[23] W. Shih, "A branch and bound method for the multiconstraint zero-one knapsack problem," *Journal of the Operational Research Society*, vol. 30, no. 4, pp. 369–378, 1979.

[24] S. Huston, J. Puchinger, and P. Stuckey, "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., 2008, pp. 39–47.

[25] P. C. Chu and J. E. Beasley, "A genetic algorithm for the multidimensional knapsack problem," *Journal of heuristics*, vol. 4, no. 1, pp. 63–86, 1998.

[26] Y. Akçay, H. Li, and S. H. Xu, "Greedy algorithm for the general multidimensional knapsack problem," *Annals of Operations Research*, vol. 150, no. 1, p. 17, 2007.

[27] M. Magazine and O. Oguz, "A heuristic algorithm for the multidimensional zero-one knapsack problem," *European Journal of Operational Research*, vol. 16, no. 3, pp. 319–326, 1984.

[28] A. Volgenant and J. Zoon, "An improved heuristic for multidimensional 0-1 knapsack problems," *Journal of the Operational Research Society*, vol. 41, no. 10, pp. 963–970, 1990.

[29] M. D. V. Baroni and F. M. Varejão, "A shuffled complex evolution algorithm for the multidimensional knapsack problem," in *Iberoamerican Congress on Pattern Recognition*. Springer, 2015, pp. 768–775.

[30] R. R. Hill, Y. K. Cho, and J. T. Moore, "Problem reduction heuristic for the 0–1 multidimensional knapsack problem," *Computers & Operations Research*, vol. 39, no. 1, pp. 19–26, 2012.

[31] J. H. Drake, E. Özcan, and E. K. Burke, "Modified choice function heuristic selection for the multidimensional knapsack problem," in *Genetic and Evolutionary Computing*. Springer, 2015, pp. 225–234.

[32] T. Perry, M. Bader-El-Den, and S. Cooper, "Imbalanced classification using genetically optimized cost sensitive classifiers," in *Evolutionary Computation (CEC), 2015 IEEE Congress on*. IEEE, 2015, pp. 680–687.

[33] M. Bader-El-Den and M. Gaber, "Garf: towards self-optimised random forests," in *International Conference on Neural Information Processing*. Springer, 2012, pp. 506–515.

[34] M. Bader-El-Den, E. Teitei, and M. Adda, "Hierarchical classification for dealing with the class imbalance problem," in *Neural Networks (IJCNN), 2016 International Joint Conference on*. IEEE, 2016, pp. 3584–3591.

[35] T. El-Boghdadly, M. Bader-El-Den, and D. Jones, "Evolving local search heuristics for the integrated berth allocation and quay crane assignment problem," in *Evolutionary Computation (CEC), 2016 IEEE Congress on*. IEEE, 2016, pp. 2880–2887.

[36] M. Bader-El-Den and T. Perry, "Mathematical function optimization using a novel algorithm based on newtonian field theory," in *Evolutionary Computation (CEC), 2016 IEEE Congress on*. IEEE, 2016, pp. 5154–5161.