Improvements to Penalty-Based Evolutionary Algorithms for the Multi-Dimensional Knapsack Problem Using a Gene-Based Adaptive Mutation Approach

Şima Uyar
Istanbul Technical University
Computer Engineering Department
Maslak 34469 Istanbul, Turkey
etaner@cs.itu.edu.tr

Gülşen Eryiğit
Istanbul Technical University
Computer Engineering Department
Maslak 34469 Istanbul, Turkey
gulsen@cs.itu.edu.tr

ABSTRACT

Knapsack problems are among the most common problems in literature tackled with evolutionary algorithms (EA). Their major advantage lies in the fact that they are relatively simple to implement while they allow generalizations for a wide range of real world problems. The multi-dimensional knapsack problem (MKP), which belongs to the class of NP-complete combinatorial optimization problems, is one of the variations of the knapsack problem. The MKP has a wide range of real world applications such as cargo loading, selecting projects to fund, budget management, cutting stock, etc. The MKP has been studied quite extensively in the EA community. Due to the constrained nature of the problem, constraint handling techniques gain great importance in the performance of the proposed EA approaches. In this study, the applicability of a generational EA that uses a penalty-based constraint handling technique and a gene locus based, asymmetric, adaptive mutation scheme is explored for the MKP. The effects of the parameters of the explored approach is determined through tests. Further experiments, using large MKP instances from commonly used benchmarks available through the Internet are performed. Comparison tables are given for the performance of the explored approach and other good performing EAs found in literature for the MKP. Results show that performance improves greatly when compared with other penalty-based techniques, but the explored approach is still not the best performer among all. However, unlike the explored technique, the EAs using the other constraint handling techniques require a great amount of extra computational effort and need heuristic information specific to the optimization problem. Based on these observations, and the fact that the performance difference between the explored scheme and the better performers is not too high, research on improving the explored approach is still in progress.

GECCO'05 June 25–29, 2005, Washington, DC, USA.

Categories and Subject Descriptors

I.2.8 [**Computing Methodologies**]: Artificial Intelligence—*Problem Solving, Control Methods, and Search*

General Terms

Algorithms

Keywords

Genetic algorithms, multi-dimensional knapsack problem, adaptive mutation, penalty-based constraint handling

1. INTRODUCTION

Knapsack problems [14] are among the most common problems in literature tackled with evolutionary algorithms (EA) [17]. Their major advantage lies in the fact that they are relatively simple to implement while they allow generalizations for a wide range of real world problems. It is noted in [14] that as a result of the analysis of the requests made to the Stony Brook Algorithm Repository [22], the conclusion [21] is that the knapsack algorithm implementations are among the third needed implementations in the repository. There are many different variations of the knapsack problems. The multi-dimensional knapsack problem (MKP), which belongs to the class of NP-complete combinatorial optimization problems, is one of these variations. The MKP has a wide range of real world applications such as cargo loading, selecting projects to fund, budget management, cutting stock, etc. The MKP has been studied quite extensively in the EA community and there are numerous publications on the applications of EAs to the MKP [4, 6, 7, 8, 9, 15, 19, 20].

The MKP is a constrained optimization problem. Different constraint handling techniques have been proposed and analyzed in literature for this problem. These will be explored in more detail in Section 2. Each of these techniques have their advantages and disadvantages. Modifications to improve their performance have also been developed. To ensure the proper evaluation of the proposed approaches, several benchmark instances have been developed and they are available on the Internet. Most papers report their results on these benchmarks, which makes comparisons easier.

The aim of this study is to explore the properties of the MKP and explore and improve an EA that uses a penalty-

based constraint handling technique. Penalty based techniques that have been proposed in earlier works are not very successful [6, 9, 13], mainly due to the presence of infeasible individuals. The technique explored in this paper uses an adaptive mutation scheme which fastens convergence to the optimum. When the population converges on an optimum, diversity is increased to let the search for other optima continue. This new EA approach, GBAM+, is a modification of the previously proposed mutation adaptation algorithm GBAM (Gene Based Adaptive Mutation) in [23, 24]. Tests are performed using the larger benchmark MKP instances on the Internet. Results are compared with others reported in literature for these specific instances.

The improvements obtained in the penalty based techniques is very promising and the performance approaches those reported for other constraint handling techniques. However there is still more work to do to outperform the approaches that currently provide the best results. The main drawback of those approaches is the extra work and the problem specific heuristic information they require in order to provide problem specific improvements. GBAM+ also uses heuristic information to improve the search process, however this information is obtained directly by observing the population and is not specific to a problem class or instance. These will be discussed in greater detail in later sections.

The rest of the paper is organized as follows: Section 2 gives an overview of the existing EA approaches for the MKP, Section 3 introduces the modified GBAM+ approach, Section 4 details the experiments that have been performed and provides results and discussions and Section 5 concludes the paper and provides possible future work directions.

2. RELATED WORK ON THE MKP

The MKP belongs to a class of NP-complete optimization problems with constraints. The MKP can be defined as in Eq. 1.

$$maximize \sum_{j=1}^{n} p_j \,.\, x_j$$
$$subject\, to \sum_{j=1}^{n} w_{ij} \,.\, x_j \leq C_i \quad i = 1, 2, ..., m \tag{1}$$

where n is the number of items, m is the number of knapsacks, $x_j \in \{0, 1\}$ shows whether jth item is included in the subset or not, p_j shows the profit of the jth item, w_{ij} shows the weight of the jth item in the ith knapsack and C_i is the capacity constraint of the ith knapsack.

There has been quite a lot of work done for the solution of the MKP. A good survey on the knapsack problems, including the MKP, and all the different solution techniques (both standard techniques, heuristics (e.g. in [18]) and hybrids (e.g. in [25])) is given in [14]. In [6], a detailed survey of the problem, of the different EA approaches proposed, of their weaknesses, strengths and performance comparisons is provided.

As noted in [6] and [9], the presence of constraints introduces the problem of infeasible parts in the search space. In [6], the techniques to handle constraints are categorized into three main groups. Some of these try to directly handle the constraints while others use different representations or decodings to transform the constrained problem into an unconstrained problem. The main constraint handling techniques are:

- Direct search in the complete search space,
- Direct search in the feasible search space,
- Indirect search in the feasible search space.

Direct search means that the candidate solutions produced by the search algorithm are actually a part of the original search space, whereas indirect search means that the candidate solutions belong to a transformed search space and are mapped back into the original search space.

Direct search in the complete search space techniques include the penalty-based constraint handling methods where the algorithm is allowed to search in the whole search space. In these techniques, the fitness of all individuals are determined according to a defined objective function, however the infeasible individuals have their fitness values penalized through a penalty function. For these techniques, design of good penalty functions plays an important role in the performance of the different approaches [6, 9, 15].

In the direct search in the feasible search space group of approaches, specialized techniques to ensure the feasibility of all solutions are used. The fitness of all individuals are evaluated using the objective function. Repair algorithms, where infeasible individuals are repaired into feasible individuals, belong to this group [4, 19].

Indirect search in the feasible search space approaches include decoder-based techniques where the search space is transformed and the new search space consists only of feasible individuals. Search continues in the transformed search space and found solutions are mapped back into the original search space [7, 12, 13, 20].

In penalty based techniques, the fitness of infeasible individuals are penalized. Good design of penalty calculation methods is crucial to the performance of an EA. In [6, 9] different penalty techniques are compared. It is shown in [6, 9] that the solution of the MKP lies close to the feasible boundary. Based on these observations and the results of the comparisons, a good penalty technique

- should penalize the infeasible individuals by an amount that is proportional to their constraint violations,
- should guarantee that the fitness of the best infeasible solution always remains lower than the fitness of the worst feasible solution in the population,
- should guide the population towards the feasible boundary.

A penalty method that satisfies these requirements and provides better performance than the previously proposed techniques is given in [6, 9].

Repair algorithms aim to take infeasible individuals and transform them back into feasible individuals. Chu and Beasley [4] propose such a repair technique. Raidl [19] takes this one step further and uses a specialized initialization technique which ensures diversity as well as the property that all individuals are feasible. He also uses a repair technique and applies a local improvement operator to the individuals. Both in [4] and [19], very good results are reported.

The major drawback of these techniques is the extra computational cost involved in repairing infeasible individuals and applying local improvements to solutions.

Decoder based techniques and their properties have been studied mainly in [6, 7, 8, 13, 20]. One of the most successful decoding techniques proposed in literature for the MKP is the weight coding approach given in [20]. In the weight coding approach, solution candidates are represented by real valued weight vectors. The decoding phase consists of two steps. In the first step, the original problem is modified by a biasing technique using the weight values. In the second step, the real valued vector is decoded into the phenotype in the binary space using a decoder heuristic. Through the use of specialized decoder heuristics, feasibility of the solutions is guaranteed. The decoded solution is then evaluated using the original fitness function. In [20], several weight coding techniques and decoding heuristics are introduced and compared. These techniques also suffer from extra computational efforts introduced through the biasing and decoding phases. Moreover they heavily rely on heuristic information about the specific optimization problem.

Most of these studies use commonly available benchmark MKP instances downloadable from the Internet. Smaller instances can be found at [11] maintained by Heitkoetter. Larger instances, engineered to conform to specific properties of good MKP benchmarks, are artificially generated by Chu and Beasley [2] and can be found at the OR-Library[3]. More recently, larger instances of the MKP have been generated and made available for download at [10].

3. THE MODIFIED GBAM+ ALGORITHM

The original GBAM algorithm [23, 24] uses an asymmetric, adaptive mutation mechanism on each loci on the chromosomes. The mutation adaptation strategy increases or decreases the mutation rate for each locus on the chromosome based on feedback obtained from the current population. This mechanism of GBAM is classified as adaptive according to the categorization in [5].

The main motivation behind GBAM is the fact that the setting of the mutation rate is one of the most sensitive parameters of an EA. Traditionally, the best values are determined using a trial-and-error method. This method does not guarantee an optimal setting as well as being very time consuming. However GBAM starts from an initial mutation rate and allows the adaptation based on feedback obtained from the search process. This allows more flexibility by specifying a mutation value in a range rather than fixing it before the actual run. Different from other known mutation adaptation strategies, GBAM has a separate mutation rate value for each locus. An adaptive approach for adjusting mutation rates for the gene locus based on the feedback obtained by observing the relative success or failure of the individuals in the population is used. Since the mutation rates at each locus depend mainly on whether the individuals with a specific allele value for that locus is successful or not, GBAM is more suited to problems in which the representation is binary.

In GBAM, there are two different mutation rates defined for each locus: p_{m1} for those genes that are "1" and p_{m0} for those that are "0". In the reproduction phase, the appropriate mutation rate is applied based on the gene allele value. Initially all of the mutation rates are set to an initial value in the specified boundaries. Then for each generation, the mutation probabilities p_{m1} and p_{m0} for each locus are updated based on feedback taken from the relative success or failures of those individuals having a "1" or "0" at that locus.

For a maximization problem, the update rule for the two mutation rate values for one gene location can be seen in Eq. 2. This update rule is applied separately for each locus.

$$p^{+}_{m0} = \begin{cases} p_{m0} + \gamma, & \frac{S_{avg}}{P_{avg}} > 1 \\ p_{m0} - \gamma, & \frac{S_{avg}}{P_{avg}} \leq 1 \end{cases}$$
$$p^{+}_{m1} = \begin{cases} p_{m1} - \gamma, & \frac{S_{avg}}{P_{avg}} > 1 \\ p_{m0} + \gamma, & \frac{S_{avg}}{P_{avg}} \leq 1 \end{cases} \quad (2)$$

where the p_{mi} value corresponds to the rate of mutation which is applied when the gene value is equal to "i" in the corresponding gene location, γ is the mutation update value, S_{avg} is the average fitness of the individuals with an allele "1" for the corresponding gene location and P_{avg} is the average fitness of the population. It follows from Eq. 2 that for a maximization problem, if the ratio of S_{avg} to P_{avg} is greater than 1, the allele value "1" for the corresponding locus is assumed to generate more successful results. Therefore, a decrease in p_{m1} and an increase in p_{m0} for the corresponding locus are implemented and vice versa if the S_{avg} to P_{avg} ratio is less than 1. As a result of the updates at each generation, p_{mi} values are allowed to change within the limits defined by lower and upper bounds. If an update causes a mutation rate to exceed the limits, it is set to the corresponding boundary value. This mutation adaptation mechanism is similar to the basic ideas used in simple estimation distribution algorithms [16], especially the population based incremental learning (PBIL) technique proposed in [1].

GBAM is shown in [24] to perform well compared to other parameter adaptation techniques. It is tested on a very simple, unimodal One-Max problem to show how it effectively decreases the first hitting time to the optimum [23]. The second set of tests are performed on the 4-Peaks problem [1] which has a partially deceptive fitness function. GBAM also performs well on various instances of this problem with differing degrees of hardness. The final tests are conducted using relatively easy instances of the MKP, namely the Weing-7 and Weish-30 datasets obtained from [11], with promisingly good results. These results motivate a further investigation into the suitability of the technique to harder instances of the MKP, which are obtained from the OR-Library [2, 3]. These test problems are generated by Chu and Beasley for their study in [4]. The basic outline of the GBAM+ algorithm used in this study is similar to the ones proposed and discussed in [23, 24], but due to the nature of the MKP, some modifications need to be made. These modifications are discussed in the following sub-sections. The flow of the algorithm is given in Fig. 1 and the lines containing the modifications are marked in bold letters.

It is shown in [6, 9] that EA approaches using a penalty-based constraint handing technique suffer from the feasibility problem, where the population may drift towards the infeasible region. Using appropriate penalty terms to guide the population back towards the feasible boundary helps to overcome this problem. The penalty calculation technique proposed and shown to perform best in [6, 9] is used in this study when calculating the penalty term for infeasible indi-

```
initialize population;
initialize mutation arrays;
do
{
  calculate fitnesses;
  update mutation arrays;
  if (proportion_feasibles < propFeas)
    reset mutation arrays;
  if (population converged)
  {
    re-init percent of population;
    reset mutation arrays;
  }
  select parents;
  cross-over pairs;
  apply mutation;
  form next generation;
} while not end-of-generations;
```

Figure 1: Outline of the GBAM+ algorithm

viduals. This penalty term is calculated using Equation 3.

$$penalty = \frac{1 + p_{max}}{w_{min}} * max(CV) \quad (3)$$

where p_{max} is the maximum profit value of all the items, w_{min} is the minimum weight of all the items in all the knapsacks, $max(CV)$ is the maximum constraint violation value for all the knapsacks.

3.1 Controlling the Number of Feasible Individuals

GBAM+ is an adaptive technique which uses the heuristic information provided by the individuals of the current population to increase or decrease the mutation rates as explained in Section 3. However for the MKP, not all individuals are feasible.

For the mutation update of GBAM+ to work properly, only the feasible individuals should be used in the adaptation. If the number of feasible individuals in the population drops to very low levels, there is no longer enough heuristic information in the population to guide the adjustment of the mutation rates.

In this new implementation of GBAM+, which is geared specifically to be used in constrained problems, if the proportion of feasible individuals in the population drops below a predefined threshold, all mutation array values are reset to the initial values. The setting of the actual value of the threshold is explored in Section 4.

3.2 Convergence Control and Re-Initialization

In [24], it is noted that even though the fast convergence rates introduced by GBAM are considered useful, once the whole population converges, necessary measures need to be taken to re-introduce diversity. A very simple diversity introduction scheme is used in that study and further analysis of better techniques is left as a future work.

In GBAM+, similar to the approach taken in [24], a percentage of the population is changed when the population converges. Gene locus convergence is said to occur when 95% of the genes at that locus has the same allele. And based on this, population convergence occurs when all loci have converged.

In this study, after convergence, a percentage of the population is randomly re-initialized and all mutation array values are reset to their initial values. The individuals in the population which are replaced by the new random individuals are also selected randomly. This step of the algorithm is carried out when the population converges which means that most of the population is similar, so it does not make much difference which individuals are selected for replacement.

This type of random re-initialization introduces diversity into the converged population. The setting of the actual percentage value is explored in Section 4.

4. EXPERIMENTS

The experiments performed for this study consist of two main parts. The first part and the second part are composed of 6 and 1 tests respectively.

In the first part, the setting of the GBAM+ parameters, which are seen to be effective on performance are explored and tests are performed to justify the actual settings of these parameter values. These parameters are:

- threshold for the proportion of feasible individuals in the population (*propFeas*)
- the ratio of the population to re-initialize when the population converges (*reinitRatio*)
- mutation rate update step length (*mutUpdate*)
- initialization method to generate the first population of the EA
- re-initialization method used when the population converges
- the number of generations per run

As explained in previous sections, GBAM+ needs to observe the search process in order to update the mutation arrays. However, it is better to guide the search process using the heuristic information gained from feasible solutions. In GBAM+, when the proportion of feasible individuals in the population drops below a threshold given by *propFeas*, the mutation arrays are reset to the initial value.

It is also explained in the previous section that when the population converges, it is no longer possible to search other areas in the search space. When convergence occurs, in GBAM+, a percentage of the population is re-initialized. The *reinitRatio* parameter determines this percentage.

It is left as later work in [23, 24] to study the effect of the mutation update step size which is used as the increment or decrement value when the mutation values are increased or decreased. The parameter *mutUpdate* corresponds to the parameter γ in Eq. 2.

The importance of the initial population especially for EAs using a penalty-based constraint handling technique is discussed in [6] in great detail. If the initial population is generated randomly, especially for instances with low tightness ratios, this will give an initial population which consists mostly of infeasible individuals. It is hard for most penalty-based EAs to guide the search back into the feasible region. A clever initialization routine, C^*, which uses problem specific information, is proposed in [6]. Initialization techniques are tested with GBAM+ before further experiments are performed.

A similar issue can be considered when re-initializing a percentage of the population when convergence occurs. Since

Table 1: Basic properties of the EA used for all experiments

Representation	Binary
Population size	250
Parent selection	Binary tournaments
Replacement	Generational
Elitism	Best replaces worst
Duplicate elimination	None
Initial mutation value	$1/ChromosomeLength$
Upper mutation bound	0.5
Lower mutation bound	0.0001
Cross-over	Uniform
Cross-over rate per pair	1.0

Table 2: Parameter settings for the first set of experiments

Parameter	Setting
propFeas	*0.25*
reinitRatio	*0.75*
mutUpdate	1/2L
initialization	clever
re-initialization	clever
generations per run	20000

the same initialization technique that is used for creating the initial population is used for re-initialization after convergence, similar tests are performed before further experimentation.

EA performance usually depends on the number of generations it is allowed to continue for each run. The generation count parameter becomes important when investigating the performance of the EAs.

In the second part of the experiments, performance of GBAM+ is explored using the larger instances of the MKP benchmarks found in literature. In this part, parameter settings are used as determined as a result of the set of experiments in the first part.

The chosen values for all the above mentioned parameters used in the experiments will be given after their effects are explored in the first set of the experiments. The rest of the parameters of GBAM+ are determined empirically and are set as given in Table 1 for all the tests in this study.

4.1 Experiments to Determine the Effects of Different Parameters

In this testing stage, effects of specific GBAM+ parameters given above are explored. For this purpose, tests are performed on one of the hardest instances of the Chu-Beasley benchmark [3]. The first problem of the set which has 500 items, 30 knapsacks and a tightness ratio of 0.25 is selected. Results are given averaged over 30 runs of the algorithm for each test. Performance comparisons are given based on average fitness values obtained over 30 runs. In all the tests, the same settings, except for the parameter setting explored at each test, are used. These are given in Table 2.

Test 1

The first test is to explore the effect of the threshold for the feasible individual proportion (*propFeas*). Tests are performed with $propFeas$ values between 0.0 and 1.0 with increments at 0.125. The results are given in Fig. 2. The plot shows that when the threshold is set to 0, which means that there is no control on the number of feasible individuals in the population, the performance is very bad. This is an expected outcome since the heuristic information used in guiding mutation rates in GBAM+ depends on the feasible individuals. Increasing this threshold also increases the performance. As can be seen best performance occurs around 0.25. Based on this observation, $propFeas = 0.25$ is used in the second set of experiments.

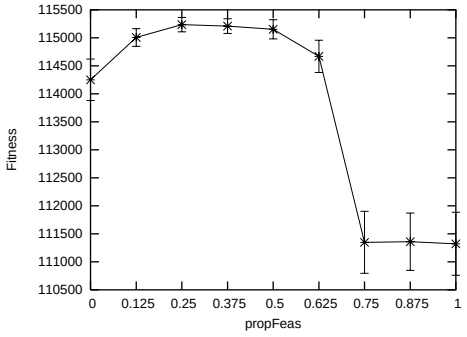

Figure 2: Effect of propFeas ratio

Test 2

The second test is to explore the effect of the ratio of the population to re-initialize when the population converges (*reinitRatio*). Tests are performed with $reinitRatio$ values between 0.0 and 1.0 with increments at 0.125. The results are given in Fig. 3. The results show that the best performance occurs around 0.75. Based on this observation, $reinitRatio = 0.75$ is used in the second set of experiments.

Test 3

The third test is to explore the effect of the mutation rate update step length (*mutUpdate*). As mentioned before, this corresponds to the γ parameter in Eq. 2. Tests are performed with several levels of values for the $mutUpdate$, and

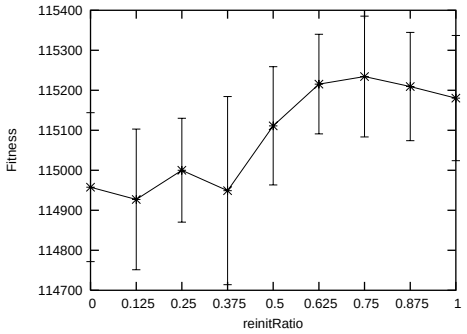

Figure 3: Effect of reinitRatio

Table 3: Effect of mutUpdate

mutUpdate	Average Fitness	Stdev
1/L	113464.13	572.05
1/2L	115196.57	148.75
1/3L	115166.97	132.92
1/4L	115151.90	126.79
1/5L	115129.90	190.08
1/10L	115090.67	177.44
1/15L	115000.57	173.61
1/20L	114982.87	192.22
1/30L	114998.23	187.65
1/40L	114830.40	196.77

Table 4: Effect of the initialization method

Initialization Method	Average Fitness	Stdev
Random	115274.67	211.60
Clever (C^*)	115196.57	148.75

the results are given in Table 3. The results show that the best performance occurs when the *mutUpdate* value is high and drops when the value gets smaller. However, when the *mutUpdate* value is set equal to the initial mutation rate, on the first adaptation when a mutation value is decreased, it gets set to the lower bound immediately. This allows for no time to explore the optimal mutation rate for the loci. This decreases the performance. The results show that the best performance occurs between $1/2L$ and $1/5L$. Based on these observations, $mutUpdate = 1/2L$ is used in the second set of experiments.

Test 4

The fourth test is to explore the effect of the initialization technique. The C^* clever initialization technique [6] is compared with a pure random initialization approach in this test. The results are given in Table 4. The clever initialization technique requires extra computational effort and is slower than a pure random initialization, and the performance of the random initialization method is slightly better on the average. So for the second set of experiments, random initialization is used to generate the first population of individuals.

Test 5

The fifth test is to explore the effect of the convergence control mechanism. All tested cases given in Table 5, except for the first case denoted as "None", includes some form of a convergence control. In "No Re-Init", there is no re-initialization of individuals and only the mutation array values are reset. In "Random Re-Init", a percentage of the population is re-initialized with new random individuals. In "Clever Re-Init", a percentage of the population is re-initialized with new individuals generated using the C^* clever initialization technique [6]. In the last two methods, mutation array values are also reset to the initial value. The results are given in Table 5. It can be seen that some form of convergence control and re-initialization is required when the population converges. As in the previous test, on the average, random re-initialization seems to be slightly better than the clever re-initialization method and again clever re-initialization is more computationally costly. Based on these observations, for the second set of experiments, re-initialization will also be performed randomly when the population converges.

Table 5: Effect of the convergence control technique

Convergence Control	Average Fitness	Stdev
None	114847.90	204.86
No Re-Init	114925.53	188.01
Random Re-Init	115224.60	152.26
Clever Re-Init	115196.57	148.75

Table 6: Effect of the no. of generations in a run

Generations	Average Fitness	Stdev
10000	115039.07	260.67
15000	115164.40	143.42
20000	115196.57	148.75
25000	115269.27	150.07
30000	115265.43	124.76

Test 6

The final test in this phase is to explore the effect of the number of generations the EA is allowed per run. Tests are performed using 10000, 15000, 20000, 25000, 30000 generations and the results are given in Table 6. It is seen that the performance improves as the number of generations increases. However no improvement can be seen after 25000 generations. Based on this observation, 25000 generations is used in the rest of the experiments.

Summary of Parameter Settings

Based on the results of the tests in this experiment phase, the settings for the tests in the second set of experiments are used as follows:

- random initialization of the initial population,
- random re-initialization with $reinitRatio = 0.75$ after convergence and resetting of mutation arrays,
- 25000 generations per run ,
- $propFeas = 0.25$,
- $mutUpdate = 1/2L$.

4.2 Experiments with Large MKP Instances

For this second part of the experiments, 90 of the largest Chu-Beasley problem instances [3] are selected. The properties of the selected problems are given in Table 7.

Performance comparisons are made based on the average %gap [4, 6, 19, 20]. The %gap is calculated as in Eq. 4.

$$\%gap = 100 * (opt^{LP} - opt^{EA})/opt^{LP} \quad (4)$$

where $opt^{LP}$is the LP relaxed optimum and opt^{EA} is the best value found by the EA. The LP relaxed optimum values which are given at [3] are used in the tests. Resulting %gap

Table 7: Properties of the test problems used

Items	KSacks	Tightness Ratio	Problem Count
500	5	0.25	10
		0.5	10
		0.75	10
500	10	0.25	10
		0.5	10
		0.75	10
500	30	0.25	10
		0.5	10
		0.75	10

Table 8: %Gap comparisons

PR	*tr*	PN	PD	G	H1	H2	CB
500-5	0.25	4.50	1.11	0.59	0.12	0.13	0.09
	0.50	2.36	0.51	0.21	0.05	0.06	0.04
	0.75	1.16	0.31	0.09	0.03	0.04	0.03
500-10	0.25	4.87	1.62	0.60	0.33	0.39	0.24
	0.50	2.49	0.74	0.27	0.15	0.20	0.11
	0.75	1.36	0.42	0.15	0.09	0.13	0.07
500-30	0.25	4.91	2.17	0.97	0.79	1.00	0.61
	0.50	2.57	0.97	0.43	0.34	0.45	0.26
	0.75	1.46	0.53	0.28	0.20	0.33	0.17
Avg. gap		**2.85**	**0.93**	**0.40**	**0.23**	**0.30**	**0.18**

values are given in Table 8, along with results reported in literature for the same problems.

In Table 8, PR denotes the problem setting given by the $(itemCount, knapsackCount)$ pair, *tr* denotes the tightness ratio, PN denotes the EA approach that uses a penalty technique and no duplicate elimination [6], PD denotes the EA approach that uses a penalty technique and duplicate elimination [6], G denotes the GBAM+ approach used in this paper, H1 denotes the weight coded GA technique with decoding heuristic 1 [20], H2 denotes the weight coded GA technique with decoding heuristic 2 [20] and CB denotes the repair based EA proposed by Chu and Beasley [4].

All %gap values for the algorithms in Table 8, other than GBAM+, are taken from the reported values in the corresponding tables given in [6], [20] and [4]. In Table 8, the %gap values are given in decreasing order from left to right, so the better performing approaches are located at the rightmost columns.

Discussion of Results

From the results in Table 8 it can be seen that previously proposed penalty based techniques do not perform well. Similar conclusions are also given in [6, 9, 13]. As stated in [6, 9], the optimal solutions to the MKP lie close to the feasible boundary. However, when using penalty based techniques, only a few of the individuals in the population are able to go towards this boundary and it is not usually sufficient to locate good solutions [6]. However, GBAM+ works in such a way that by using heuristic performance information obtained directly from the ongoing search process, the whole population is guided towards where the better performing individuals are going. So this aspect of GBAM+ helps the population to converge on locations closer to the feasible boundary where good solutions are located.

It can be argued that GBAM+ requires more fitness evaluations than all other approaches. Even though the reported tests are performed using 25000 generations per run for maximal performance, the same tests have also been run with 20000 generations and the obtained results are similar and do not change the ordering of the algorithms given in Table 8. These results are not reported here due to space limitations. Even though GBAM+ requires more fitness evaluations, the other techniques require extra computational calculations. Techniques such as clever initialization methods, calculation of surrogate multipliers, LP or Lagrangian relaxation mechanisms, decoding heuristics, local search heuristics and repair algorithms are all computationally very costly. Moreover, penalty based techniques are much easier to implement than decoder based methods or repair approaches which usually contain the above mentioned extra techniques.

Besides, all of the above mentioned techniques used for improving the performance of the better performing EA approaches require using special heuristic information specific to the optimization problem being solved. GBAM+ also improves its performance through the use of heuristic information, however, the heuristic information used in GBAM+ is obtained directly from the ongoing search process, so it is not as problem specific as the others. Thus GBAM+ is more applicable to a wider range of problem classes.

Another observation that can be made from the results in Table 8 is that the difference in performance of GBAM+ and the best performers decreases for harder problems. Especially for the problems with 500 items and 30 knapsacks, the %gaps values are very close to those of the better performing algorithms.

So on the whole, the results show that the GBAM+ mechanism improves the performance of penalty based techniques and achieves a performance which comes close to the best performers for some of the problems. There is still room for improvement and further experiments and research is being carried out.

5. CONCLUSION AND FUTURE WORK

In this study, the gene based adaptive mutation rate adjustment approach (GBAM) proposed in [23, 24] is adapted (GBAM+) to be used with the multi-dimensional knapsack problem (MKP). The MKP is an NP-complete, constrained optimization problem. The parameter settings and the performance of GBAM are explored through experiments run on hard instances chosen from the benchmark created by Chu and Beasley [4] and available from the OR-Library [3]. Due to time and space limitations, only the results for the hardest 90 problems from the benchmark are reported in this study. However, experiments are being performed with the smaller instances found in the OR-Library in addition to even larger problem instances which can be downloaded from [10].

The results of the experiments show that the explored technique greatly improves the performance of penalty based techniques and approaches the performance of the other techniques for most of the problems. There is still more work to do to fine tune the approach and make it able to outperform or at least match the performance of the others, This is a desirable outcome because penalty-based approaches,

- require less computation costs,
- are easier to implement,
- require much less problem specific information,
- are more general

than decoder based approaches or repair algorithms which usually incorporate techniques such as special representations and their corresponding special operators, clever initialization methods, calculation of surrogate multipliers, LP or Lagrangian relaxation mechanisms, decoding heuristics, local search heuristics and repair algorithms. The results obtained in this study are quite encouraging and promote further study.

The experiments and ideas explored in this study are part of an ongoing project. Another branch of the project deals with analyzing the dynamics of the fitness landscapes created by using this mutation adaptation scheme. Also the effects of this mutation adaptation scheme is being explored in problems where there is strong linkage.

6. REFERENCES

[1] Baluja S., Caruana. R., "Removing the Genetics from the Standard Genetic Algorithm", in Proceedings of the 12th International Conference on Machine Learning, Morgan Kaufmann, pp. 38-46, 1995.

[2] Beasley J. E., "OR-Library: distributing test problems by electronic mail", Journal of the Operational Research Society, Vol. 41:11, pp1069-1072, 1990.

[3] Beasley J. E., OR Library, "*http://www.brunel.ac.uk/depts/ma/research/jeb/info.html*", 2004.

[4] Chu P. C., Beasley J. E., "A Genetic Algorithm for the Multidimensional Knapsack Problem", Journal of Heuristics, Vol. 4, pp. 63-86, Kluwer, 1998.

[5] Eiben A. E., Hinterding R., Michalewicz Z., "Parameter Control in Evolutionary Algorithms", IEEE Transactions on Evolutionary Computation, Vol, 3:2, pp. 124-141, 1999.

[6] Gottlieb J., Evolutionary Algorithms for Constrained Optimization Problems, PhD Thesis, Technical University of Clausthal - Germany, 1999.

[7] Gottlieb J., Raidl G. R., "Characterizing Locality in Decoder-Based EAs for the Multidimensional Knapsack Problem", in Proceedings of Artificial Evolution (EA99), 1999.

[8] Gottlieb J., Raidl G. R., "The Effects of Locality on the Dynamics of Decoder-Based Evolutionary Search", in Proceedings of the Genetic and Evolutionary Computation Conference (GECCO), 2000.

[9] Gottlieb J., "On the Feasibility Problem of Penalty-Based Evolutionary Algorithms for Knapsack Problems", in Applications of Evolutionary Computing: Proceedings of EvoWorkshops 2001: EvoCOP, 2001.

[10] Hearin Center for Enterprise Science, "*http://hces.bus.olemiss.edu/tools.html*", 2004.

[11] Heitkoetter J., SAC-94 Suite of 0/1 Multiple-Knapsack Problems, "*http://elib.zib.de/pub/Packages/mp-testdata/ip/sac94-suite/*", 2004.

[12] Hinterding R., "Mapping, Order-Independent Genes and The Knapsack Problem", in Proceedings of the 1st IEEE International Conference on Evolutionary Computation, 1994.

[13] Hinterding R., "Representation, Constraint Satisfaction and The Knapsack Problem", in Proceedings of the IEEE International Conference on Evolutionary Computation, 1999.

[14] Kellerer H., Pferschy U., Pisinger D., *Knapsack Problems*, Springer, 2004.

[15] Khuri S., Baeck T., Heitkoetter J., "The Zero/One Multiple Knapsack Problem and Genetic Algorithms", in *Proceedings of the ACM Symposium on Applied Computation*, ACM Press, 1994.

[16] Larranaga P., Lozano J. A., Estimation of Distribution Algorithms: A New Tool for Evolutionary Computation, Kluwer, 2001.

[17] Michalewicz Z., Genetic Algorithms + Data Structures = Evolution Programs, Springer, 1999.

[18] Pirkul H., "A Heuristic Solution Procedure for the Multiconstraint Zero-One Knapsack Problem", Naval Research Logistics, Vol. 34, pp. 161-172, 1987.

[19] Raidl G. R., "An Improved Genetic Algorithm for the Multiconstrained 0-1 Knapsack Problem", in Proceedings of the 5th IEEE International Conference on Evolutionary Computation, 1998.

[20] Raidl G. R., "Weight-Codings in a Genetic Algorithm for the Multiconstraint Knapsack Problem". in Proceedings of the IEEE Congress on Evolutionary Computation (CEC99), 1999.

[21] Skiena S. S., "Who is Interested in Algorithms and Why? - Lessons from the Stony Brook Algorithms Repository", 2nd Workshop on Algorithm Engineering (WAE'98), pp. 204-212, 1998.

[22] Skiena S. S., The Stony Brook Algorithms Repository, "*http://www.cs.sunysb.edu/~algorith/*", 2001.

[23] Uyar A. S., Eryigit G., Sariel S., "An Adaptive Mutation Scheme in Genetic Algorithms Fastening Convergence to the Optimum", in *Proceedings of 3rd APIS: Asian Pacific International Symposium on Information Technologies*, 2004.

[24] Uyar A. S., Sariel S., Eryigit G., "A Gene Based Adaptive Mutation Strategy for Genetic Algorithms", in *Proceedings of GECCO 2004: Genetic and Evolutionary Computation Conference*, 2004.

[25] Vasquez M., Hao J. K., "A Hybrid Approach for the 0-1 Multidimensional Knapsack Problem", in Proceedings of the IJCAI, 2001.