Weight-Codings in a Genetic Algorithm for the Multiconstraint Knapsack Problem

Günther R. Raidl
Institute of Computer Graphics
Vienna University of Technology
Karlsplatz 13/1861, 1040 Vienna, Austria
raidl@apm.tuwien.ac.at

Abstract- This paper presents different variants of weight-coding in a genetic algorithm (GA) for solving the multiconstraint knapsack problem (MKP). In this coding, a chromosome is a vector of weights associated with the items of the MKP. The phenotype is obtained by using the weights to generate a modified version of the original problem and applying a decoding heuristic to it. Four techniques of biasing the original problem with weights are discussed. Two well working decoding heuristics, one based on the surrogate relaxation and the other one based on the Lagrangian relaxation, are introduced.

The different weight-coding variants are experimentally compared to each other using a steady-state GA. Furthermore, the influence of the biasing strength, a strategy parameter of the codings, is investigated. In general, the GA found solutions being substantially better than those obtained by applying heuristics to the MKP directly.

1 Introduction

Weight-Coding is a solution encoding technique for genetic algorithms (GAs) that already proved to be well suited for different combinatorial optimization problems. The basic idea is to represent a candidate solution by a vector of numerical weight values w_j ($j = 1, \ldots, n$). A two-step process is used to decode such a chromosome into a phenotypic solution: First, the original problem P is temporarily modified to P' by biasing problem parameters with the weights w_j. Secondly, a problem-specific non-evolutionary *decoding heuristic* is used to actually generate a solution for P'. This solution is finally interpreted and evaluated for the original (unmodified) problem P.

In a weight-coded GA, classical recombination and mutation operators can be used to generate new chromosomes. Feasibility of all generated candidate solutions can be guaranteed if a suitable decoding heuristic is used. In contrast to many other techniques that map vectors of numerical values to feasible solutions of constrained combinatorial optimization problems, weight-coding usually provides strong locality: Similar chromosomes normally map to similar phenotypical solutions, and recombination can therefore produce offsprings inheriting much of the parental phenotypical structure.

Weight-codings have already been successfully used for a variety of problems, such as the optimum communications spanning tree problem [16], the rectilinear Steiner tree problem [9], the 3-satisfiability problem [5], the minimum weight triangulation problem [2], the traveling salesperson problem [10, 11], and the multiple container packing problem [19].

This paper presents different variants of weight-coding in a GA for the *multiconstraint knapsack problem* (MKP). The next section provides a definition of the MKP and a short overview of prior approaches to solve it. Section 3 describes four different biasing methods and two suitable decoding heuristics. A steady-state GA, which formed the basis for a number of experiments, is described in Sec. 4, and Sec. 5 documents obtained results. These results indicate that a weight-coded GA is a robust and effective technique for finding high-quality solutions to the MKP, provided that a suitable decoding heuristic and biasing technique is used. Average results are comparable to those of two previously presented, highly effective hybrid GAs for the MKP (Chu and Beasley [3, 4], Raidl [18]).

2 The Multiconstraint Knapsack Problem

The MKP is a classical, NP-complete combinatorial optimization problem with applications in various fields such as economics. A set of n items and a set of m resources are given. Each item j ($j = 1, \ldots, n$) has assigned a profit p_j and for each resource i ($i = 1, \ldots, m$) a resource consumption value $r_{i,j}$. The problem is to identify a subset of all items that leads to the highest possible total profit and does not exceed given resource limits b_i. Formally, the MKP can be stated as follows:

$$\text{maximize} \qquad f = \sum_{j=1}^{n} p_j x_j, \tag{1}$$

$$\text{subject to} \qquad \sum_{j=1}^{n} r_{i,j} x_j \le b_i, \quad i = 1, \ldots, m, \tag{2}$$

$$x_j \in \{0, 1\}, \quad j = 1, \ldots, n$$

$$\text{with} \qquad p_j > 0, \quad r_{i,j} \ge 0, \quad b_i \ge 0.$$

The variables searched for are the x_j. If item j is element of the subset, x_j is set to 1, otherwise to 0. Equation 1 represents the total profit of selected items and Eq. 2 the m resource constraints. Note that all p_j, $r_{i,j}$, and b_i are always positive (or zero).

Because of the NP-completeness of the MKP, exhaustive search algorithms such as branch-and-bound that lead to globally optimal solutions are in general too time-consuming and can only be applied to very small problems. Note that much research concerning knapsack problems deals with the simpler uni-dimensional knapsack problem with $m = 1$. For this special case, effective approximation algorithms have been presented in the past [14]. Several heuristics were also presented for the general MKP, such as those from Pirkul [17], Magazine and Oguz [13], and Volgenant and Zoon [22]. But unfortunately, the effectiveness of these heuristics is very limited if they are applied to MKPs where both m and n are large. See [3, 4] for a comprehensive review on exact and heuristic algorithms.

In the last years, GAs have shown to be well suited for finding high-quality solutions to also larger knapsack problems, see [3, 4, 6, 7, 8, 15, 18, 20, 21]. In [18], Raidl observed that these GA approaches can be divided into two categories according to the solution encoding techniques. Some algorithms use *direct encoding*, meaning that a chromosome of the GA contains a gene for each item indicating directly if the item is supposed to be packed into the knapsack. In this case infeasible solutions must be handled by using a repair algorithm or adding a penalty term to the objective function. On the other hand, some GAs use *order-based encoding* in which a chromosome contains a permutation of all items. The actual solution is obtained by applying a first-fit algorithm: In the order given by the permutation, one item after the other is inserted into the initially empty knapsack as long as it does not violate a capacity constraint. When applying order-based encoding, special recombination and mutation operators must be used to generate new chromosomes that contain valid permutations again. Note that the efficiency of some GAs for the MKP could be enhanced considerably by hybridizing them, i.e. by including some local improvement operator, heuristic repair operator, and/or heuristic initialization procedure, see [3, 4, 6, 18].

3 Weight-Codings for the MKP

Weight-coding seems to be an interesting new approach to the MKP since it eliminates the necessity of an explicit repair algorithm, a penalization of infeasible solutions, or special recombination and mutation operators. Furthermore, a weight-coded GA is already a hybrid approach since it includes the problem specific heuristic decoding function.

3.1 Biasing the Original Problem

In the proposed weight-coded GA for the MKP, a candidate solution is represented by a vector $(w_1, w_2, \ldots, w_n)$ of weights. Weight w_j is associated with item j. Different biasing techniques can be used for obtaining the modified (biased) problem P' to which the decoding heuristic will be applied. Furthermore, the weights w_j may be initialized and mutated in different ways. The methods that are examined in this work are described in the following.

(B1) Addition of uniformly distributed weights to profits:

$$p'_j = p_j + w_j, \quad w_j = \mathcal{R}(0, \gamma\overline{p}). \tag{3}$$

Biased profits p'_j are obtained by adding associated weights w_j to the original profits. During initialization and mutation, weights w_j are set to uniformly distributed random numbers (denoted by $\mathcal{R}(0, \gamma\overline{p})$) in the range from 0 to the average original profit $\overline{p} = (\sum_{j=1}^{n} p_j)/n$ multiplied by a *biasing strength* γ. Negative weights are not allowed to avoid problems with profits that may otherwise become negative. The biasing strength γ is a strategy parameter which therefore does not depend on absolute values of profits.

(B2) Addition of relative, uniformly distributed weights to profits:

$$p'_j = p_j + w_j, \quad w_j = \mathcal{R}(0, \gamma p_j). \tag{4}$$

Weights are now set to random values in ranges proportional to the actual profits p_j. γ is again the biasing strength. An advantage of this technique over B1 is that the median biased problem corresponds to the original problem since the problem structure does not change if profits are multiplied by the same constant value. This biasing technique is therefore "symmetrical".

(B3) Multiplication of profits with logarithmically distributed weights:

$$p'_j = p_j w_j, \quad w_j = (1+\gamma)^{\mathcal{R}(-1,1)}. \tag{5}$$

Original profits are now multiplied by weights that are logarithmically distributed in the range $[1/(1+\gamma), 1+\gamma]$. The median value of this distribution is 1. Therefore, median biased profits correspond to original profits.

(B4) Multiplication of profits with log-normally distributed weights:

$$p'_j = p_j w_j, \quad w_j = (1+\gamma)^{\mathcal{N}(0,1)}. \tag{6}$$

In contrast to B3, a log-normal distribution is used for initializing and mutating weights. $\mathcal{N}(0, 1)$ denotes a normally distributed random number with mean 0 and standard deviation 1. This gives the advantage that small changes of profits are made with higher probabilities, but large changes are also possible. Again, median biased profits correspond to original profits.

3.2 Decoding Heuristics

The following two heuristics are proposed as decoding heuristics for obtaining the phenotypical solution to a biased problem. For simplicity, we assume that the resource coefficients $r_{i,j}$ are normalized during a preprocessing step:

$$\begin{array}{ll} r_{i,j} \leftarrow r_{i,j}/b_i & \text{for } i = 1, \ldots, m,\ j = 1, \ldots, n, \\ b_i \leftarrow 1 & \text{for } i = 1, \ldots, m. \end{array}$$

```
function Heuristic-1:
determine a_i for i = 1,...,m by solving the LP-relaxed
                MKP and taking the dual variables;
μ_j ← Σ_{i=1}^m a_i r_{i,j} for j = 1,...,n;
u_j ← p_j/μ_j for j = 1,...,n;
x_j ← 0 for j = 1,...,n;
R_i ← 0 for i = 1,...,m;
for all j sorted according to decreasing u_j do
    if R_i + r_{i,j} ≤ 1 for all i = 1,...,m then
        x_j ← 1;
        R_i ← R_i + r_{i,j} for i = 1,...,m;
return (x_1, x_2, ..., x_n);
```

Figure 1: *The surrogate relaxation based heuristic H1*

```
function Heuristic-2:
λ_i ← 0 for i = 1,...,m;
x_j ← 1 for j = 1,...,n;
R_i ← Σ_{j=1}^n r_{i,j} for i = 1,...,m;
while not(R_i ≤ 1 for all i = 1,...,m) do
    determine resource I for which R_I = max{R_i};
    for all items j with x_j = 1 do
        if r_{I,j} > 0 then
            δ_j ← (p_j − Σ_{i=1}^m λ_i r_{i,j}) / r_{I,j};
        else
            δ_j ← ∞;
    determine item J for which δ_J = min{δ_j | x_j = 1};
    λ_I ← λ_I + δ_J;
    x_J ← 0;
    R_i ← R_i − r_{i,J} for i = 1,...,m;
for all items j with x_j = 0 sorted according to
                                decreasing p_j do
    if R_i + r_{i,j} ≤ 1 for all i = 1,...,m then
        x_j ← 1;
        R_i ← R_i + r_{i,j} for i = 1,...,m;
return (x_1, x_2, ..., x_n);
```

Figure 2: *The Lagrangian relaxation based heuristic H2*

(H1) The surrogate relaxation based heuristic:

In [17], Pirkul presents a heuristic for the MKP which makes use of surrogate duality. The m resource constraints (Eq. 2) are transformed into a single constraint using surrogate multipliers a_i ($i = 1, \ldots, m$):

$$\sum_{j=1}^{n} \left(\sum_{i=1}^{m} a_i r_{i,j} \right) x_j \le \sum_{i=1}^{m} a_i. \tag{7}$$

Assuming suitable surrogate multipliers a_i are known, a feasible solution to the MKP can be obtained in the following greedy way: First, all items are sorted in decreasing order of *profit/pseudo-resource consumption ratios* $u_j = p_j/\mu_j$ with $\mu_j = \sum_{i=1}^{m} a_i r_{i,j}$. Then, the items are processed in this order, and each item which would not violate any of the m resource constraints is packed into the knapsack, i.e. x_j is set to 1. See Fig. 1 for a more detailed pseudo-code.

Pirkul [17] suggests several methods to derive the surrogate multipliers a_i. One of the simplest methods to obtain reasonably good multipliers is to solve the *linear programming* (LP) relaxed MKP in which the variables x_j may get arbitrary values from the interval $[0,1]$ and to use the values of the dual variables as the surrogate multipliers. In other words, a_i is set to the shadow price of the i-th constraint in the LP relaxed MKP.

To keep the computational effort of decoding a chromosome in a weight-coded GA with this heuristic small, the surrogate multipliers a_i are determined only once for the original problem data in a preprocessing step. Furthermore, also the pseudo-resource consumptions μ_j can be predetermined. During the chromosome decoding step the heuristic starts with the computation of the ratios u_j for the biased profits p'_j. The computational effort for decoding a chromosome in this way is only $O(n \,\mathrm{ld}\, n)$ for sorting the items according to actual profit/pseudo-resource consumption ratios plus $O(nm)$ for packing the knapsack and checking the constraints during each step.

(H2) The Lagrangian relaxation based heuristic:

In [13], Magazine and Oguz present a heuristic for the MKP which uses the Lagrangian relaxation of the MKP. All m resource constraints (Eq. 2) are incorporated into the maximization goal (Eq. 1) by subtracting resource consumptions multiplied by *Lagrange multipliers* λ_i ($i = 1, \ldots, m$, $\lambda_i \ge 0$) from the total profit:

$$\text{maximize} \quad f^{\mathrm{LR}} = \sum_{j=1}^{n} p_j x_j - \sum_{i=1}^{m} \lambda_i \sum_{j=1}^{n} r_{i,j} x_j. \tag{8}$$

Assuming the Lagrange multipliers λ_i are known, this maximization problem (without further constraints) can be solved easily, since x_j must simply be set to 1 if and only if

$$\sum_{j=1}^{n} p_j - \sum_{i=1}^{m} \lambda_i r_{i,j} > 0. \tag{9}$$

The difficulty is to find values for the Lagrange multipliers such that this optimal $\vec{x} = (x_1, x_2, \ldots, x_n)$ for Eq. 8 is a feasible solution for the MKP and also satisfies

$$\sum_{i=1}^{m} \lambda_i \left(1 - \sum_{j=1}^{n} r_{i,j} x_j \right) = 0, \tag{10}$$

in which case $\vec{x}$ is optimal for the MKP.

Magazine and Oguz [13] suggest the following heuristic procedure for obtaining good (but usually suboptimal) values for λ_i and simultaneously deriving $\vec{x}$. See also Fig. 2 for a more detailed pseudo-code.

Initially, all Lagrange multipliers λ_i are set to 0, and all x_j are set to 1. Although Eq. 9 is satisfied, this is in general not a feasible solution for the MKP. Next, all actual resource consumptions R_i are determined, and the most violated constraint I is identified. The corresponding multiplier λ_I is then increased as much as necessary to violate Eq. 9 for just one variable x_J. x_J is set to 0, and resource consumptions R_i are updated. This step is repeated until the solution has become feasible. A final local improvement step checks if any zero-variable can be set to 1 without violating any constraint.

Basically, the computational effort for this procedure is $O(n^2m)$, but it can be improved to $O(n(n+m))$ if the net profits $p_j - \sum_{i=1}^{m} \lambda_i r_{i,j}$ are saved and adjusted each time after changing the multiplier λ_I. But nevertheless, this decoding heuristic is computationally clearly more expensive than H1.

Note that only a relatively small part of all possible feasible solutions is covered by the search space of a weight-coded GA using one of the proposed decoding heuristics. Generally it is essential that most good solutions and especially the global optima are covered. In other words, only poor solutions should be omitted. In case of the presented biasing techniques and decoding heuristics, we can guarantee for any feasible solution $\vec{x}$ that either $\vec{x}$ itself or a better solution containing all items selected in $\vec{x}$ plus some others is covered if the biasing strength γ is large enough. The advantage of both heuristics is that they produce only meaningful solutions lying on the boundary of the feasible region of all possible solutions where also the global optima are located. Note that also Gottlieb [6] observed that it is crucial for any EA for MKP to emphasize search on this boundary. The practical influence of different values for γ is investigated in Sec. 5.

4 A weight-coded GA for the MKP

The described weight-coding variants have been incorporated into a traditional steady-state GA with binary tournament selection. Within a chromosome, weights w_j are directly stored as real valued genes. Initial solutions are generated by assigning each weight a random value within the range or with the distribution specific to the used biasing technique.

In early experiments, uniform crossover proved to behave slightly better than one- or two-point crossover. The mutation operator modifies a weight by resetting it to a new random value. New candidate solutions are generated by always performing crossover and applying mutation with a probability of $3/n$ per gene. A smaller probability for performing mutation or recombination increases the danger of premature convergence; a much larger probability for mutation degrades performance.

As already observed in previous GAs for similar combinatorial optimization problems [3, 4, 11, 18, 19], it proved again to be essential to disallow duplicates in the population. This is accomplished by using a replacement scheme that only accepts new solutions different from all others in the population. The test for equality is efficiently performed on phenotype level using a hash table. If a new solution is not a duplicate, it always replaces the solution with the worst fitness.

Preliminary experiments indicated that a population size of 100 works well with problems of different sizes and properties. Each GA run terminated when 100,000 solutions had been evaluated without finding a new best solution. This criterion ensures sufficient convergence in practice.

5 Experimental Comparison

Standard MKP test data proposed by Chu and Beasley [3, 4] and publically available from OR-Library[1] [1] were used to practically examine the GA with the different biasing techniques and two decoding heuristics. These test data contain 10 problem instances for each combination of $m \in \{5, 10, 30\}$, $n \in \{100, 250, 500\}$, and $\alpha \in \{0.25, 0.5, 0.75\}$ with $\alpha = b_i / \sum_{j=1}^{n} r_{i,j}$ being the *tightness ratio*. Since the optimal solution values for most of these problems are not known, the quality of a solution is measured by the percentage gap of the objective value f with respect to the optimal value of the LP-relaxed problem $f^{\mathrm{LP}}_{\max}$: *%-gap* $= 100(f^{\mathrm{LP}}_{\max} - f)/f^{\mathrm{LP}}_{\max}$.

First of all, test runs were performed with the aim to compare biasing techniques B1 to B4 for both decoding heuristics H1 and H2 and examine the influence of different biasing strengths γ in the range from 0.01 to 100. Results of runs for 10 medium sized problem instances with $m = 10$, $n = 250$, and $\alpha = 0.5$ were averaged. Tables 1 and 2 and Fig. 3 show *%-gap*s of best-of-run solutions and the numbers of evaluations needed to find them.

In general, it can be seen that all four biasing techniques work well for both decoding heuristics if the biasing strength γ is larger than or equal to some *working bound* $\gamma_{\min}$ (e.g. for H1 with B1: $\gamma_{\min} \approx 0.02$). If γ lies below this bound, the GA's search space is too narrow; chromosomes are not able to represent some promising solutions. Note that Julstrom observed a similar robustness of the biasing strength above a certain lower bound in a weight-coded GA for the traveling salesperson problem [12].

Although differences are small, biasing techniques B2, B3, and B4 perform better than B1. A reason for this slightly poorer behavior of B1 seems to be that B1 distorts the original problem by asymmetrically biasing it: The median modified problem does not correspond to the original problem. For both decoding heuristics, the biasing techniques which multiply profits by logarithmically or log-normally distributed weights (B3 and B4) lead to the best results with the smallest *%-gap*s if the biasing strength γ is chosen only a bit larger than $\gamma_{\min}$. For larger γ the *%-gap* increases and differences between the four biasing techniques become insignificant.

Regarding the number of evaluations, no significant differences could be observed between the biasing techniques. Up to $\gamma \approx 2$, there is the general trend that smaller biasing strengths lead to faster convergence. Obviously, a reason for this is the narrower search space when γ is smaller.

Considering this observations and also that B4 has the

[1] http://mscmga.ms.ic.ac.uk/info.html

Table 1: Average results for a weight-coded GA using decoding heuristic H1, biasing techniques B1 to B4, and different biasing strengths γ. All values are average values obtained from runs for 10 different problems with $m = 10$, $n = 250$, and $\alpha = 0.5$.

H1	B1		B2		B3		B4	
γ	*%-gap*	*Evals*	*%-gap*	*Evals*	*%-gap*	*Evals*	*%-gap*	*Evals*
0.01	0.465	21800	0.408	3880	0.349	20410	0.314	21080
0.02	0.372	24170	0.322	13490	0.294	9770	0.277	35060
0.05	0.301	31880	0.292	45780	0.267	24430	0.276	52350
0.10	0.297	22270	0.273	32420	0.273	39640	0.275	82340
0.15	0.286	30930	0.288	19490	0.291	69960	0.300	79390
0.2	0.311	78990	0.281	47850	0.292	98280	0.343	95650
0.3	0.311	56830	0.301	50160	0.304	108530	0.344	118030
0.5	0.319	91250	0.303	84890	0.353	122360	0.321	151720
0.7	0.332	154680	0.309	75640	0.323	92600	0.334	150240
1.0	0.320	151610	0.338	93160	0.340	115550	0.382	143510
1.5	0.366	125370	0.335	124370	0.348	147200	0.356	122920
2	0.330	169900	0.322	176440	0.322	164750	0.363	161690
5	0.366	164120	0.344	123760	0.362	132610	0.374	134970
10	0.362	199200	0.352	111650	0.391	158370	0.367	147830
20	0.371	193400	0.337	178700	0.350	157830	0.356	159400
50	0.390	155540	0.369	163740	0.358	139850	0.360	121500
100	0.372	145590	0.326	247290	0.382	115590	0.368	197460

Table 2: Average results for a weight-coded GA using decoding heuristic H2, biasing techniques B1 to B4, and different biasing strengths γ. All values are average values obtained from runs for 10 different problems with $m = 10$, $n = 250$, and $\alpha = 0.5$.

H2	B1		B2		B3		B4	
γ	*%-gap*	*Evals*	*%-gap*	*Evals*	*%-gap*	*Evals*	*%-gap*	*Evals*
0.01	4.525	24670	4.472	31230	4.152	36580	3.453	61690
0.02	4.181	70210	4.550	23590	3.543	18550	2.559	108370
0.05	3.474	49920	4.175	29800	2.364	89790	0.969	151890
0.10	2.614	85760	3.372	52600	1.199	121180	0.316	128030
0.15	1.978	94720	2.923	73820	0.632	89231	0.292	122030
0.2	1.454	120600	2.501	118480	0.371	87820	0.321	86580
0.3	0.347	132740	1.936	101230	0.321	97143	0.307	81360
0.5	0.368	126300	1.345	112840	0.311	143740	0.325	131600
0.7	0.338	130230	0.372	109220	0.323	121353	0.337	127622
1.0	0.329	147290	0.333	104100	0.335	143250	0.354	141000
1.5	0.326	147910	0.343	143960	0.347	102314	0.334	104870
2	0.345	164910	0.352	70980	0.352	90610	0.320	150230
5	0.349	104930	0.335	114900	0.317	180450	0.325	183210
10	0.342	141910	0.348	111500	0.329	160550	0.341	152390
20	0.355	172160	0.327	137090	0.350	157670	0.352	147230
50	0.353	193280	0.323	112710	0.355	149910	0.336	173210
100	0.356	118360	0.345	120420	0.342	160232	0.351	130520

smallest working bounds $\gamma_{\min}$ for both decoding heuristics, B4 with $\gamma \approx 0.05$ for H1 and with $\gamma \approx 0.2$ for H2 seem to be the best choices for at least the used test problems. But note that for decoding heuristic H1 biasing technique B3 performes similarly well.

Some experiments regarding the comparison of the four biasing techniques were also made with smaller and larger problem instances of Chu's test problem set. The obtained results were very similar to those documented here. Also the working bounds $\gamma_{\min}$ and therefore the optimal value for γ did not differ substantially. But nevertheless, note that the optimal value for γ depends on the distributions of profits

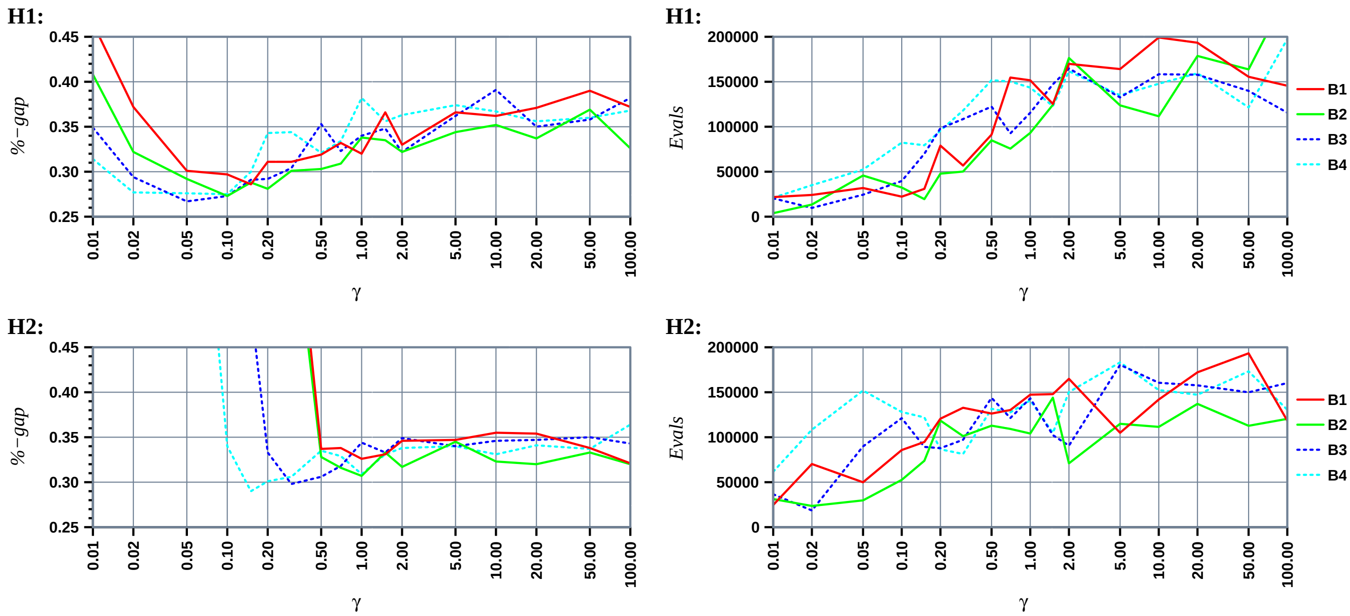

Figure 3: Average results for a weight-coded GA using decoding heuristic H1 and H2, biasing techniques B1 to B4, and different biasing strengths γ.

p_j and resource consumption values $r_{i,j}$ (but not on absolute values).

Using B4 with $\gamma = 0.05$ for H1 and $\gamma = 0.2$ for H2, large scale tests were performed for all of Chu's test problems. Table 3 shows average *%-gap*s of the solutions obtained when applying heuristics H1 and H2 directly (without any GA) and average results of the GA runs. Note that heuristic H1 led always to better solutions than H2. Furthermore, the solutions found by both GA variants are in all cases substantially better than those obtained by the heuristics solely.

Although the total average *%-gap*s for the GA with H1 and H2 as decoding heuristic do not differ much (H1: 0.59, H2: 0.65), the GA with H1 is the clear winner: The GA with H2 found only slightly better solutions for some small problems with few constraints ($m = 5$, $n = 100$). Furthermore, significantly different are the number of evaluations needed to find these solutions. In average the GA with H1 needed only half the number of evaluations of the GA with H2. But even more different are the associated computing times (measured on a Pentium II PC). Because of the larger computational complexity of H2, the GA with this decoding heuristic is especially for the large problems up to a factor 20 slower. In general, these results indicate clearly that H1 should be prefered over H2 as decoding heuristic (very small problems might be an exception).

The obtained results, especially those for H1, also compare well to the results of the hybrid GAs proposed by Chu and Beasley [3, 4], Raidl [18], and Gottlieb [6]. For most problems, they report slightly smaller *%-gap*s, but on the other hand more evaluations were performed per run. Further tests using the same numbers of evaluations would be necessary to make a fair comparison.

6 Conclusions and Future Work

This paper has described different variants of a novel coding of solutions for the MKP. Each chromosome is a vector of weights associated with items. A phenotype is obtained by using the weights to generate a modified version of the original problem and applying a decoding heuristic to it. Both presented decoding heuristics work well, but the surrogate relaxation based method (H1) is in general preferable because of the smaller computational effort and the slightly better resulting solutions. The solutions obtained by the weight-coded GA variants were in all cases substantially better than those found by the heuristics alone.

Four different biasing techniques were presented and experimentally compared to each other. Although they all work well if the biasing strength is larger than a certain working bound, the method of multiplying profits by log-normally distributed weights exhibits small advantages. If the biasing strength is chosen to be only a bit larger than this working bound, the best results are usually achieved, and the number of evaluations needed by the GA to converge to good solutions is significantly smaller. Note that the results obtained for the different biasing techniques may also be of interest for weight-coded GAs addressing other combinatorial optimization problems.

An open question is how an optimal biasing strength can be found in general. Beside the derivation of some heuristic

Table 3: Average results of tests on 270 problem instances with varying m, n, and α: Shown are *%-gaps* obtained by applying heuristics H1 and H2 directly and *%-gaps* of best-of-run solutions with needed evaluations $Evals$ and CPU times t obtained by weight-coded GAs using decoding heuristics H1 and H2. Biasing technique B4 with $\gamma = 0.05$ for H1 and $\gamma = 0.2$ for H2 was used. All values are average values determined from runs for 10 different problems.

m	n	α	H1	H2	GA with H1			GA with H2		
			%-gap	*%-gap*	*%-gap*	$Evals$	$t[s]$	*%-gap*	$Evals$	$t[s]$
5	100	0.250	2.840	9.056	1.007	6370	4.4	0.989	22700	31.5
		0.500	1.397	4.852	0.453	20350	14.0	0.455	20630	24.0
		0.750	0.950	3.833	0.319	4520	3.1	0.318	10430	9.1
		Avg.	1.729	5.914	0.593	10413	7.2	0.587	17920	21.5
5	250	0.250	1.026	4.839	0.256	47910	58.3	0.273	101000	588.3
		0.500	0.530	3.969	0.127	53860	66.0	0.132	82630	368.0
		0.750	0.309	2.811	0.080	29710	36.5	0.087	67630	184.0
		Avg.	0.622	3.873	0.154	43827	53.6	0.164	83753	380.1
5	500	0.250	0.454	3.851	0.115	60860	136.2	0.126	199250	4198.1
		0.500	0.217	2.536	0.053	105830	238.2	0.057	146760	2288.8
		0.750	0.137	2.017	0.032	62180	140.7	0.037	114690	998.0
		Avg.	0.269	2.802	0.067	76290	171.7	0.073	153567	2495.0
10	100	0.250	3.708	12.627	1.624	41322	29.6	1.707	50930	75.8
		0.500	2.478	8.580	0.803	30560	22.3	0.827	42630	53.1
		0.750	1.279	4.914	0.493	23380	17.1	0.519	40810	38.9
		Avg.	2.488	8.707	0.973	31754	23.0	1.018	44790	55.9
10	250	0.250	1.754	9.812	0.589	79500	101.3	0.664	89340	544.4
		0.500	0.801	5.788	0.276	52350	69.2	0.311	86580	407.0
		0.750	0.528	3.711	0.161	33640	44.7	0.188	91430	266.6
		Avg.	1.028	6.437	0.342	55667	72.1	0.388	89117	406.0
10	500	0.250	0.822	7.802	0.332	105390	246.9	0.385	129620	2838.9
		0.500	0.403	5.216	0.150	42400	102.0	0.196	157080	2580.3
		0.750	0.287	3.237	0.085	76920	189.4	0.126	149460	1379.0
		Avg.	0.504	5.418	0.189	74903	179.4	0.236	145387	2266.1
30	100	0.250	11.087	14.568	3.067	8070	6.1	3.075	28960	49.4
		0.500	4.339	10.403	1.376	30740	24.4	1.478	46960	68.9
		0.750	2.345	5.751	0.848	18280	15.1	0.942	68170	77.0
		Avg.	5.924	10.241	1.764	19030	15.2	1.832	48030	65.1
30	250	0.250	3.811	12.031	1.382	49710	69.1	1.615	143210	965.0
		0.500	1.739	7.910	0.609	68840	102.3	0.706	89710	479.2
		0.750	1.224	3.972	0.348	36820	58.4	0.449	101740	347.6
		Avg.	2.258	7.971	0.780	51790	76.6	0.923	111553	597.3
30	500	0.250	2.217	8.955	0.785	133840	343.3	0.995	169330	4129.5
		0.500	1.030	6.505	0.336	71650	199.9	0.447	200850	3737.3
		0.750	0.524	3.452	0.195	85320	256.0	0.331	214830	2302.5
		Avg.	1.257	6.304	0.439	96937	266.4	0.591	195003	3389.8
Total Average:			1.786	6.407	0.589	51179	96.1	0.646	98791.11	1075.19

formula, *self adaption* (i.e. the biasing strength is optimized by the GA itself) might be a promising approach. Furthermore, there remain several other ways to bias the original problem, and also other MKP heuristics may be suitable decoding heuristics.

Bibliography

[1] J. E. Beasley: *Obtaining Test Problems via Internet*, Journal of Global Optimization 8, pp. 429–433, 1996.

[2] K. Capp, B. Julstrom: *A Weight-Coded Genetic Algorithm for the Minimum Weight Triangulation Problem*, in Proc. of the 1998 ACM Symposium on Applied Computing, ACM Press, pp. 327–331, 1998.

[3] P. C. Chu: *A Genetic Algorithm Approach for Combinatorial Optimization Problems*, Ph.D. thesis at The Management School, Imperial College of Science, London, 1997.

[4] P. C. Chu, J. E. Beasley: *A Genetic Algorithm for the Multidimensional Knapsack Problem*, Journal of Heuristics 4, pp. 63–86, 1998.

[5] A. E. Eiben, J. K. Van der Hauw: *Solving 3-SAT by GAs adapting constraint weights*, in Proc. of the 1997 IEEE Int. Conf. on Evolutionary Computation, Indianapolis, IN, pp. 81–86, 1997.

[6] J. Gottlieb: *Evolutionary Algorithms for Multidimensional Knapsack Problems: The Relevance of the Boundary of the Feasible Region*, submitted to the Genetic and Evolutionary Computation Conf., Orlando, FL, 1999.

[7] R. Hinterding: *Mapping, Order-independent Genes and the Knapsack Problem*, in Proc. of the 1st IEEE Int. Conf. on Evolutionary Computation, Orlando, FL, pp. 13–17, 1994.

[8] S. Khuri, T. Bäck, J. Heitkötter: *The Zero/One Multiple Knapsack Problem and Genetic Algorithms*, in Proc. of the 1994 ACM Symposium on Applied Computing, ACM Press, pp. 188–193, 1994.

[9] B. Julstrom: *Representing Rectilinear Steiner Trees in Genetic Algorithms*, in Proc. of the 1996 ACM Symposium on Applied Computing, ACM Press, pp. 245–250, 1996.

[10] B. Julstrom: *Comparing Decoding Algorithms in a Weight-Coded GA for TSP*, in Proc. of the 1998 ACM Symposium on Applied Computing, ACM Press, pp. 313–317, 1998.

[11] B. Julstrom: *Insertion Decoding Algorithms and Initial Tours in a Weight-Coded GA for TSP*, in Proc. of the 3rd Genetic Programming Conf., Madison, Wisconsin, pp. 528–534, 1998.

[12] B. Julstrom: *The Maximum Weight Parameter in a Weight-Coded GA for TSP*, in Proc. of the 3rd Genetic Programming Conf., late breaking paper, Madison, Wisconsin, pp. 101–105, 1998.

[13] M. J. Magazine, O. Oguz: *A Heuristic Algorithm for the Multidimensional Zero–One Knapsack Problem*, European Journal of Operational Research 16, pp. 319–326, 1984.

[14] S. Martello, P. Toth: *Knapsack Problems: Algorithms and Computer Implementations*, J. Wiley & Sons, 1990.

[15] A. L. Olsen: *Penalty Functions and the Knapsack Problem*, in Proc. of the 1st IEEE Int. Conf. on Evolutionary Computation, Orlando, FL, pp. 559–564, 1994.

[16] C. C. Palmer, A. Kershenbaum: *Representing Trees in Genetic Algorithms*, in Proc. of the 1st IEEE Int. Conf. on Evolutionary Computation, Orlando, FL, pp. 379–384, 1994.

[17] H. Pirkul: *A Heuristic Solution Procedure for the Multiconstrained Zero-One Knapsack Problem*, Naval Research Logistics 34, pp. 161–172, 1987.

[18] G. R. Raidl: *An Improved Genetic Algorithm for the Multiconstrained 0-1 Knapsack Problem*, in Proc. of the 5th IEEE Int. Conf. on Evolutionary Computation, Anchorage, Alaska, pp. 207–211, 1998.

[19] G. R. Raidl: *A Weight-Coded Genetic Algorithm for the Multiple Container Packing Problem*, to appear in Proc. of the 14th ACM Symposium on Applied Computing, San Antonio, TX, 1999.

[20] G. Rudolph, J. Sprave: *Significance of Locality and Selection Pressure in the Grand Deluge Evolutionary Algorithm*, in Proc. of the Int. Conf. on Parallel Problem Solving from Nature IV, pp. 686–694, 1996.

[21] J. Thiel, S. Voss: *Some Experiences on Solving Multiconstraint Zero-One Knapsack Problems with Genetic Algorithms*, INFOR 32, pp. 226–242, 1994.

[22] A. Volgenant, J. A. Zoon: *An Improved Heuristic for Multidimensional 0–1 Knapsack Problems*, Journal of the Operational Research Society 41, pp. 963–970, 1990.