TECHNISCHE UNIVERSITÄT WIEN
Institut für Computergraphik und Algorithmen

Empirical Analysis of Locality, Heritability and Heuristic Bias

Günther R. Raidl and Jens Gottlieb

Forschungsbericht / Technical Report

TR–186–1–04–05

18. May 2004

Favoritenstraße 9-11 / E186, A-1040 Wien, Austria
Tel. +43 (1) 58801-18601, Fax +43 (1) 58801-18699
www.cg.tuwien.ac.at

Empirical Analysis of Locality, Heritability and Heuristic Bias in Evolutionary Algorithms: A Case Study for the Multidimensional Knapsack Problem

Günther R. Raidl raidl@ads.tuwien.ac.at
Institute of Computer Graphics and Algorithms, Vienna University of Technology
Favoritenstr. 9–11/1861, 1040 Vienna, Austria

Jens Gottlieb jens.gottlieb@sap.com
SAP AG
Neurottstr. 16, 69190 Walldorf, Germany

Abstract

Five different representations and associated variation operators are studied in the context of a steady-state evolutionary algorithm (EA) for the multidimensional knapsack problem. Four of them are indirect decoder-based techniques, and the fifth is a direct encoding including heuristic initialization, repair, and local improvement. The complex decoders and the local improvement and repair strategies make it practically impossible to completely analyze such EAs in a fully theoretical way. After comparing the general performance of the EA variants on two benchmark suites, we present a hands-on approach for empirically analyzing important aspects of initialization, mutation, and crossover in an isolated fashion. Static, inexpensive measurements based on randomly created solutions are performed in order to quantify and visualize specific properties with respect to heuristic bias, locality, and heritability. These tests shed light onto the complex behavior of such EAs and point out reasons for good or bad performance. In addition, the proposed measures are also examined during actual EA runs, which gives further insight into dynamic aspects of evolutionary search and verifies the validity of the isolated static measurements. All measurements are described in a general way, allowing for an easy adaption to other representations and combinatorial problems.

Keywords

Representation, locality, heritability, heuristic bias, evolutionary algorithm, multidimensional knapsack problem.

1 Introduction

Evolutionary algorithms (EAs) have been applied successfully to a great variety of combinatorial optimization problems. While their performance is affected by general parameter choices like population size, the used selection scheme, or the application probabilities of variation operators, the success mainly depends on problem-specific decisions concerning the representation of solution candidates and the variation operators used. In the design of an EA, these decisions are usually based on intuition and experience, rather than on detailed formal and empirical analysis of alternative representations and operators.

We use a formal model to characterize the interplay of representation and variation operators. The operators work in the *search space*, which is typically called genotype space in case of decoder-based EAs. The search space is mapped to the *phenotype space*, the set of all solution candidates for the problem at hand. Our model allows to quantify important aspects of evolutionary search, which are essential for good performance. Here, we mainly focus on three aspects: *locality*, *heritability*, and *heuristic bias*.

Locality means that small steps in the search space, like those typically performed by mutation operators, cause small phenotypic changes. Strong locality allows evolutionary search to explore the phenotype space in a meaningful way since variation and selection cause exploitation of the neighborhoods of promising phenotypes. Intuitively, we expect these neighborhoods to contain phenotypes of high quality, too, since in most practical problems minor phenotypic differences typically cause minor fitness differences. Weak locality prevents evolutionary search from a meaningful exploration of the phenotype space because small variations often cause strong phenotypic changes. In the worst case, the search behaves like random search in the phenotype space, which is usually ineffective.

Heritability refers to the ability of crossover operators to produce children that combine meaningful features of their parents. Each property of an offspring should stem from at least one of its parents, and crossover should preserve properties appearing in all parents. This allows the exploitation of successful common substructures in the parents. Obviously, a perfect preservation of parental phenotypic properties can be achieved by simply copying the child from one parent. This, however, is not desired since it contradicts another role of crossover: It should mix the parents' phenotypic properties in a creative way, i.e. new phenotypes should be introduced into the search process. All these aspects are frequently associated with the role of crossover in successful evolutionary search, and we capture them with the general term heritability. Using crossover without sufficient heritability is questionable and often hinders the search process rather than supporting it.

Heuristic bias concerns the mapping from search space to phenotype space. Evolutionary search explores a search space, which is defined by the representation and the variation operators. This search space is often only indirectly connected to the phenotype space. The efficacy of the search process is strongly influenced by the mapping between these spaces. Hence, using some heuristic in this mapping yields a certain distribution of phenotypes, which can help to increase performance if the distribution is biased towards phenotypes of higher fitness. We refer to this effect as heuristic bias.

Our approach analyzes locality, heritability, and heuristic bias in detail and documents their effects on evolutionary search. We use the notions of distances in the search space and the phenotype space for describing the reachability between elements of the search space and the similarity among phenotypes, respectively. They allow to formalize locality and heritability by introducing specific measures. Heuristic bias

is characterized by the fitness distribution obtained by randomly sampling the search space. Based on these concepts, the main ingredients of evolutionary search can be analyzed in an isolated, static fashion, or their interplay can be examined dynamically during actual EA runs. The static analysis is computationally efficient and can predict the real search dynamics to a large extent. Therefore, it provides a solid basis for deciding which representation and variation operators are suitable for a given problem.

The applicability of our approach is demonstrated on the *multidimensional knapsack problem* (MKP), a well-known NP-hard problem for which several EA-based approaches have been proposed in the literature. We introduce and compare five different EAs for this problem, and perform static and dynamic analyses explaining the success or failure of these algorithms, respectively. Although empirical results are presented for the MKP only, we remark that the proposed approach is general and can easily be adapted to other problems by defining appropriate phenotypic distance metrics.

This paper unifies previous work (Raidl and Gottlieb, 1999; Gottlieb and Raidl, 1999, 2000) on this topic and adds significant new results in every aspect. The analysis does now include a variant of the currently most successful EA for the MKP, which uses a direct representation, repairing, and local improvement. Furthermore, we consider here the random-key representation that has been successfully applied in several problem domains like e.g. network design (Rothlauf et al., 2002) or, more generally, permutation optimization (Bosman and Thierens, 2002). Whereas locality and heritability have already been studied before, heuristic bias has not been explicitly considered. Empirical results are presented for a standard benchmark suite with instances containing up to 500 items (Chu and Beasley, 1998) and more recent benchmarks with up to 2500 items, which were first used by Vasquez and Hao (2001).

We proceed by introducing the MKP in Section 2, followed by a review of evolutionary algorithms for it in Section 3. A comparison of selected EAs is presented in Section 4, which focuses on general aspects like solution quality and duplicate ratio. Basic concepts such as spaces and associated distances are introduced in Section 5. Related previous work is reviewed in Section 6. Then, the considered EAs for the MKP are analyzed in detail, in order to explain the results from Section 4: Sections 7 and 8 focus on static aspects like heuristic bias and locality and heritability, respectively. Section 9 concentrates on dynamic aspects of evolutionary search, thereby cross-checking the results of the static analyses and gaining new insights into search dynamics. Conclusions are given in Section 10.

2 The Multidimensional Knapsack Problem

In the multidimensional knapsack problem, the objective is to determine a subset of $n > 0$ items, which yields maximum profit and does not exceed the capacities of $m > 0$ resources. Formally, the problem is stated as

$$\text{maximize} \quad g(x) = \sum_{j=1}^{n} p_j x_j \tag{1}$$

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

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

where $p_j > 0$ is the profit of item j, $c_i > 0$ the capacity of resource i, and $r_{ij} \geq 0$ the resource consumption of item j w.r.t. resource i. The decision variables $x = (x_1, \ldots, x_n)$

specify for each item j whether it is selected ($x_j = 1$) or not ($x_j = 0$).

Many practical problems can be stated as MKP, like e.g. cargo loading, project selection, or resource allocation in computer networks. There are lots of theoretical and empirical studies for numerous variants of knapsack problems (Martello and Toth, 1990; Kellerer et al., 2004). The *unidimensional knapsack problem*, a special case of the MKP with $m = 1$, is only weakly NP-hard and solvable in pseudo-polynomial time. However, the general case $m > 1$ is strongly NP-hard (Garey and Johnson, 1979) and exact techniques are in practice only applicable to instances of small to moderate size.

The MKP belongs to the general class of covering and packing problems, which are structurally equivalent in the sense that the global optima are located on the boundaries of the feasible regions (Gottlieb, 1999). In case of the MKP, the boundary contains the feasible solutions which cannot be improved by inserting more items without violating resource capacities. We refer to (Chu and Beasley, 1998; Kellerer et al., 2004) for an overview of heuristic and exact algorithms for the MKP, and continue with a survey on evolutionary algorithms for it.

3 Evolutionary Algorithms for the Multidimensional Knapsack Problem

3.1 Overview

The choice of an appropriate constraint-handling technique is an important issue when solving the MKP by an evolutionary algorithm. The most promising approaches are based on heuristic decoders (Raidl, 1999) or repair algorithms combined with local improvement (Chu and Beasley, 1998; Raidl, 1998). A comparison of standard constraint-handling techniques reveals that the success of evolutionary algorithms for the MKP strongly depends on their ability to restrict or at least strongly focus the search on the boundary of the feasible region (Gottlieb, 1999).

Here we consider another critical issue: The choice of the representation and variation operators. Five classical representations are considered in the following, which all produce only solutions on the boundary. Four representations are indirect, i.e. new search spaces are introduced that are mapped to the phenotype space by means of decoding procedures. As fifth representation, we consider the direct encoding via bit strings, which is the most natural representation for the MKP. We present variation operators for each representation as well as the decoding procedures for the indirect representations.

3.2 Permutation Representation (PE)

The permutation representation is typically used for sequencing tasks as they appear in scheduling and routing problems, but it has also been applied to the unidimensional knapsack problem (Hinterding, 1994) and the MKP (Thiel and Voss, 1994; Raidl, 1998). The approach considers permutations of all items, $\pi : \{1, \ldots, n\} \rightarrow \{1, \ldots, n\}$, denoted by $\pi = (\pi_1, \ldots, \pi_n)$. A *first-fit* algorithm is used to decode such a permutation into a feasible solution. It starts with the feasible solution $x = (0, \ldots, 0)$ and considers each item in the order determined by the permutation π. Each corresponding decision variable x_{π_j}, $j = 1, \ldots, n$, is increased from 0 to 1 if the inclusion of item π_j does not violate any capacity constraint. The whole procedure runs in time $O(m \cdot n)$, since m capacity checks must be performed per item.

Standard permutation operators can be used, and in particular uniform order based crossover and swap mutation were reported to yield good results (Hinterding, 1994; Gottlieb, 1999). In general, this representation turned out to perform relatively well, although inferior to the approaches described in Sections 3.5 and 3.6.

3.3 Ordinal Representation (OR)

Among other problems, the ordinal representation was studied for the traveling salesperson problem (Grefenstette et al., 1985), the unidimensional knapsack problem (Michalewicz and Arabas, 1994), and the MKP (Gottlieb and Raidl, 1999). A solution is represented by a vector $v = (v_1, \ldots, v_n)$ with $v_k \in \{1, \ldots, n-k+1\}$ for $k \in \{1, \ldots, n\}$. The vector is mapped to a permutation π of the items $\{1, \ldots, n\}$, which is further decoded to a feasible solution via the first-fit heuristic described in Section 3.2.

The mapping of v to a permutation $\pi = (\pi_1, \ldots, \pi_n)$ uses an ordered list initialized with all the items: $L = (L_1, \ldots, L_n) = (1, \ldots, n)$. Vector v is traversed from its first to its last position. Each entry v_k specifies a position in L; the referenced element L_{v_k} is removed from L and represents π_k. As an example, assume $v = (3, 1, 2, 1)$. Initially, the ordered list $L = (1, 2, 3, 4)$. Vector v is decoded by successively removing the elements 3, 1, 4, 2 from L yielding the permutation $\pi = (3, 1, 4, 2)$. The mapping from v to π takes time $O(n \cdot \log n)$ when implementing L as a balanced search tree in which the number of elements is stored for each subtree in the corresponding root node. The total decoding time for obtaining the represented solution is then $O(n \cdot (m + \log n))$.

This representation allows the use of standard variation operators. Typically, the mutation operator randomly chooses a position $k \in \{1, \ldots, n\}$ and then uniformly draws a new value for v_k from $\{1, \ldots, n-k+1\}$. Any classical crossover like uniform or multi-point crossover can be used. Previous studies employed one-point crossover.

The mapping from v to π is a one-to-one mapping, and the search spaces of the ordinal representation and the permutation representation are therefore equally large. A closer look at the ordinal representation's decoding procedure reveals that a change of one position of v can have dramatic effects on the decoded solution because each item selection modifies list L and may influence all following item selections. Although bad results for knapsack problems were already reported (Michalewicz and Arabas, 1994; Gottlieb and Raidl, 1999), this representation serves as a good example for weak locality and is primarily considered here for that reason.

3.4 Random-Key Representation (RK)

The random-key approach is based on real-valued vectors $w = (w_1, \ldots, w_n)$, where each item j is assigned a weight $w_j \in [0, 1]$. Inspired by Bean's work on random keys (Bean, 1994), Hinterding (1999) used such a representation for the unidimensional knapsack problem. The decoder sorts all items according to their weights, which yields a permutation $\pi = (\pi_1, \ldots, \pi_n)$ with $w_{\pi_j} \leq w_{\pi_{j+1}}$ for $j \in \{1, \ldots, n-1\}$. Again, this permutation is decoded via the first-fit heuristic already used for the permutation representation. The sorting of weights requires time $O(n \cdot \log n)$, yielding $O(n \cdot (m + \log n))$ as total decoding time.

Standard operators like two-point crossover and Gaussian mutation were used by Hinterding (1999), who reported this approach as being inferior to the permutation representation. In our experiments we employ uniform crossover and positional mutation, i.e., one weight is chosen randomly and reinitialized to a new random value.

3.5 Weight-Biased Representation (WB)

Weight-biasing is a general technique that has already been used successfully for a variety of combinatorial optimization problems; Julstrom (1997) gives a survey on several applications. The general principle is as follows. A solution is represented by a vector of real-valued weights. For obtaining the phenotype such a weight vector represents, a two-step process is used: First, the original problem Π is temporarily modified to Π'

by biasing certain problem parameters according to the weights. Secondly, a problem-specific heuristic is used to derive a solution for Π'. This solution is interpreted and evaluated for the original (unbiased) problem Π.

It is relatively easy to adapt this general technique to many combinatorial optimization problems which may even include complex constraints. Classical positional crossover and mutation operators can be applied, and, when using a suitable biasing scheme and decoding heuristic, only feasible candidate solutions are created.

Cotta and Troya (1998) and Raidl (1999) studied the weight-biased representation for the MKP. Different decoding heuristics and techniques for biasing the original problem have been investigated. The following log-normal distributed multiplicative biasing scheme and surrogate relaxation based heuristic were found to often work best (Raidl, 1999).

Let $w = (w_1, \ldots, w_n)$ be the weight vector representing a candidate solution; weight w_j is associated with item j of the MKP. Initialization sets each weight to a log-normally distributed random value:

$$w_j = (1+\gamma)^{\mathcal{N}(0,1)}, \quad j = 1, \ldots, n\,. \tag{4}$$

$\mathcal{N}(0,1)$ denotes a normally distributed random number with mean 0 and standard deviation 1, and $\gamma > 0$ is a strategy parameter that controls the average intensity of biasing. The original MKP instance is biased by multiplying each item's profit p_j with the associated weight:

$$p'_j = p_j w_j, \quad j = 1, \ldots, n\,. \tag{5}$$

Thus, the larger the strategy parameter γ, the stronger is the expected modification of profits and the biasing leading away from the solution the decoding heuristic would create for the original, unbiased problem. Since the resource consumption values r_{ij} and resource limits c_i are not modified, each feasible solution for the biased problem is also feasible for the original problem.

The heuristic Raidl (1999) suggests for decoding has originally been proposed by Pirkul (1987) and makes use of the surrogate duality. The m resource constraints (2) are collapsed into a single constraint using surrogate multipliers a_i, $i = 1, \ldots, m$:

$$\sum_{j=1}^{n} \left(\sum_{i=1}^{m} a_i r_{ij} \right) x_j \leq \sum_{i=1}^{m} a_i c_i\,. \tag{6}$$

Suitable surrogate multipliers a_i are obtained by solving the linear programming (LP) relaxation of the MKP, in which the variables x_j may get real values from $[0, 1]$. The values of the dual variables are then used as surrogate multipliers, i.e. a_i is set to the shadow price of the i-th constraint in the LP-relaxed MKP.

Pirkul's heuristic starts with the "empty" solution $x = (0, \ldots, 0)$ and sorts all items according to decreasing *pseudo-utility ratio*

$$u_j = \frac{p'_j}{\sum_{i=1}^{m} a_i r_{ij}}, \tag{7}$$

the ratio of profit and pseudo-resource consumption; a higher pseudo-utility ratio heuristically indicates that an item is more efficient, while a low ratio reflects low profit in combination with high resource consumption. Then, the first-fit strategy already used as decoder in the permutation representation is applied; all items are traversed in

the predetermined order and each item's variable x_j is set to 1 if no resource constraint is violated.

To keep the computational effort of decoding a weight vector reasonably small, the surrogate multipliers a_i and the resulting pseudo-resource consumptions are determined only once for the original problem in a preprocessing step. The computational effort of the decoder is then only $O(n \cdot \log n)$ for sorting the items plus $O(n \cdot m)$ for the first-fit strategy, yielding $O(n \cdot (m + \log n))$ in total.

As variation operators, uniform crossover and positional mutation, which chooses one weight w_j randomly and reinitializes it by Equation (4), are applied.

3.6 Direct Representation (DI)

Chu and Beasley (1998) proposed an EA based on a direct representation of solutions by characteristic bit vectors. Uniform crossover and classical bit-wise mutation are used, which may cause constraint violations. Infeasible candidate solutions are immediately repaired by iteratively removing items until all constraints are satisfied. Furthermore, each solution is locally improved by inserting new items that do not cause constraint violations. Both phases, repairing and local improvement, are guided by a heuristic ordering of the items.

Here, we use a variant of the original proposal. Initialization is guided by the LP relaxation of the MKP, as suggested by Raidl (1998) and refined by Gottlieb (1999). More specific, let $x^{LP} = (x_1^{LP}, \ldots, x_n^{LP}) \in [0,1]^n$ be the optimal solution of the LP relaxation, which is calculated during preprocessing. In a first phase, a subset of "elite" items is selected by including each item j with probability x_j^{LP}. The first-fit heuristic is then applied to a random permutation of this restricted set. In a second phase, all remaining items are randomly ordered and also processed by the first-fit heuristic in order to obtain a solution on the boundary of the feasible region.

Repairing and local improvement use a heuristic ordering of the items according to their pseudo-utility ratios defined in Equation (7). Again, the dual variables from the solution of the LP relaxation are used as surrogate multipliers a_i.

The repair algorithm removes the least promising items first in order to obtain a feasible solution of high quality. Thus, the repair phase processes the included items ordered by increasing pseudo-utility ratio, removing items until all constraints are satisfied.

Local improvement considers all items not appearing in the solution in decreasing pseudo-utility ratio order and includes an item if no constraints are violated. Thus, the most promising items are included first in order to increase profit as much as possible.

Due to the employed initialization, repair, and local improvement methods, a candidate solution is always located on the boundary of the feasible region. Assuming the LP relaxation is solved during preprocessing, initialization, repairing, and local improvement require $O(m \cdot n)$ time. This configuration represents the most effective evolutionary algorithm for the MKP we are aware of.

3.7 Other Representations

Other representations were proposed, which do not focus the search so strongly on the boundary of the feasible region. These approaches are discarded from further consideration in our empirical analysis, but, for the sake of completeness, we briefly introduce them in the following.

Variable-length representation Besides his proposal to use permutations for the unidimensional knapsack problem, Hinterding (1994) also introduced a variable-length

representation resembling a selection of items fitting into the knapsack. He calls this representation a direct encoding; however, many representations of the same solution exist, since the items are stored in a particular order. The employed injection crossover and in particular the mutation operator make explicit use of this item ordering when using the first-fit algorithm.

The variable-length representation is inferior to the permutation representation on the larger instances examined (Hinterding, 1994), which might be caused by the fact that the permutation-based decoder produces only solutions on the boundary of the feasible region, while for the variable-length representation mutation and crossover may produce feasible candidates to which further items could be added. The used initialization routine is equivalent to producing a random permutation and applying the permutation decoder.

Genetic programming Bruhn and Geyer-Schulz (2002) investigated genetic programming over context-free languages with linear constraints and presented results on variants of the MKP. A candidate solution is represented by a derivation tree produced by the grammar. Each leaf node selects an item, and each node contains the used and free capacity of the sub-tree it represents, which allows for checking the capacity constraints in each sub-tree. The suggested initialization routine, which produces a derivation tree by iteratively applying a randomly chosen rule of the grammar, is computationally expensive since infeasible solutions are simply rejected; i.e. the creation of one feasible individual in the initial population can be preceded by the creation of many infeasible candidates. Neither the initialization nor the variation operators are guaranteed to produce feasible solutions on the boundary. The authors claim their approach being superior to a penalty-based EA using a variable-length representation comparable to that of Hinterding (1994). We believe the inferior performance of the penalty-based EA in their study is primarily caused by its suboptimal configuration: The initialization routine produces feasible solutions far away from the boundary, i.e. many additional items can in general be added without violating capacity constraints, and the employed mutation operator is only able to replace an item by another, but cannot include additional items.

4 Empirical Comparison

This section empirically compares the different representations and associated variation operators with respect to their general performance. The following sections analyse the effects of heuristic bias, locality, and heritability – which altogether influence performance substantially – in more detail in order to gain a better understanding of *why* some approach works well or not.

4.1 Experimental Setup

We compare the five representations and associated operators listed in Table 1 and introduced in the last section in a common steady-state evolutionary algorithm framework. In the following, we refer to the individual EA variants also by the applied representation's name.

The common framework is straightforward and has been used in previous studies on EAs for the MKP by Chu and Beasley (1998), Raidl (1998), Gottlieb (1999) and Levenhagen et al. (2001). In each iteration, two parents are selected from the population by binary tournaments with replacement. One offspring is created by applying recombination and mutation. The offspring replaces the worst solution in the population, with

Table 1: Considered representations and their operators.

Representation	Permutation (PE)	Ordinal (OR)	Random-Key (RK)	Weight-Biased (WB)	Direct (DI)
Initialization	random	random	random	random (log-normal)	heuristically
Crossover	uniform order based	one-point	uniform	uniform	uniform
Mutation	swap two positions	one position (uniform)	one position (uniform)	one position (log-normal)	flip each pos. with prob. $1/n$

one exception: If the new offspring represents the same phenotype as another candidate in the population, the new offspring is discarded. This duplicate elimination is a simple but effective technique for counteracting premature convergence (Raidl and Gottlieb, 1999).

The population size is 100, and each run was terminated after 1 000 000 created solution candidates; rejected duplicates were not counted. This stopping criterion allows each considered EA variant to reasonably converge on all instances. Thus, when prolonging the runs, further improvements are typically only tiny. In many cases substantially fewer iterations would also have been sufficient to obtain the final solution.

For the weight-biased representation, multiplication of profits by log-normally distributed weights with the biasing strategy parameter $\gamma = 0.05$ (see Section 3.5) was applied, as recommended by Raidl (1999).

A standard test suite of MKP benchmark instances introduced by Chu and Beasley (1998) and available from the OR-Library[1] (Beasley, 1996) is used. This test suite contains 10 instances for each combination of $m \in \{5, 10, 30\}$ constraints, $n \in \{100, 250, 500\}$ items, and tightness ratio $\alpha \in \{0.25, 0.5, 0.75\}$. Each problem has been generated randomly such that $c_i = \alpha \cdot \sum_{j=1}^{n} r_{ij}$ for all $i = 1, \ldots, m$. Here, we focus on the first instances with $\alpha = 0.5$ for each combination of m and n only, and call them CB1 to CB9.

In addition, we also use another MKP benchmark suite[2], which was first referenced by Vasquez and Hao (2001) and originally provided by Glover and Kochenberger. These instances, called GK01 to GK11, range from 100 to 2500 items and from 15 to 100 constraints.

4.2 Solution Quality

For a solution x, the quality is measured by the *gap*

$$\frac{g(x^{LP}) - g(x)}{g(x^{LP})}$$

between the optimum x^{LP} of the LP-relaxed problem and x. Table 2 displays average gaps of the final solutions and their standard deviations obtained from 30 runs per problem instance. Results are also given for a random search (RS) approach, which creates random permutations of all items and decodes them by the first-fit strategy

[1] `http://mscmga.ms.ic.ac.uk/info.html`
[2] `http://hces.bus.olemiss.edu/tools.html`

Table 2: Average gaps of best solutions and their standard deviations.

instance			gap [%] (and standard deviation)					
name	m	n	PE	OR	RK	WB	DI	RS
CB1	5	100	0.425 (0.000)	0.745 (0.210)	0.425 (0.000)	0.425 (0.000)	0.425 (0.000)	7.554 (0.412)
CB2	5	250	0.120 (0.012)	1.321 (0.346)	0.115 (0.009)	0.106 (0.007)	0.106 (0.006)	9.651 (0.375)
CB3	5	500	0.081 (0.016)	2.382 (0.657)	0.065 (0.010)	0.042 (0.008)	0.038 (0.003)	11.393 (0.169)
CB4	10	100	0.762 (0.001)	1.013 (0.163)	0.762 (0.003)	0.761 (0.000)	0.762 (0.003)	8.330 (0.401)
CB5	10	250	0.295 (0.033)	1.498 (0.225)	0.277 (0.021)	0.249 (0.017)	0.261 (0.008)	10.294 (0.303)
CB6	10	500	0.225 (0.040)	2.815 (0.462)	0.200 (0.029)	0.131 (0.014)	0.112 (0.007)	11.628 (0.240)
CB7	30	100	1.372 (0.134)	1.800 (0.182)	1.338 (0.123)	1.319 (0.093)	1.336 (0.091)	9.323 (0.399)
CB8	30	250	0.608 (0.048)	2.076 (0.346)	0.611 (0.072)	0.535 (0.031)	0.519 (0.013)	11.809 (0.266)
CB9	30	500	0.429 (0.058)	3.267 (0.442)	0.376 (0.037)	0.306 (0.024)	0.288 (0.012)	12.965 (0.245)
GK01	15	100	0.377 (0.068)	0.683 (0.098)	0.384 (0.080)	0.308 (0.077)	0.270 (0.028)	3.344 (0.170)
GK02	25	100	0.503 (0.062)	0.959 (0.144)	0.521 (0.068)	0.481 (0.045)	0.460 (0.007)	3.293 (0.182)
GK03	25	150	0.517 (0.060)	1.002 (0.140)	0.531 (0.077)	0.452 (0.042)	0.366 (0.007)	3.237 (0.118)
GK04	50	150	0.712 (0.090)	1.164 (0.143)	0.748 (0.098)	0.669 (0.081)	0.528 (0.021)	3.133 (0.095)
GK05	25	200	0.462 (0.072)	1.124 (0.153)	0.552 (0.118)	0.397 (0.046)	0.294 (0.004)	3.424 (0.068)
GK06	50	200	0.703 (0.070)	1.236 (0.141)	0.751 (0.108)	0.611 (0.060)	0.429 (0.018)	2.943 (0.092)
GK07	25	500	0.523 (0.088)	1.468 (0.092)	0.651 (0.087)	0.382 (0.082)	0.093 (0.004)	3.566 (0.059)
GK08	50	500	0.749 (0.086)	1.517 (0.109)	0.835 (0.125)	0.534 (0.066)	0.166 (0.006)	2.840 (0.051)
GK09	25	1 500	0.890 (0.075)	2.312 (0.113)	1.064 (0.133)	0.558 (0.042)	0.029 (0.001)	3.524 (0.030)
GK10	50	1 500	1.101 (0.065)	1.883 (0.076)	1.177 (0.082)	0.727 (0.070)	0.052 (0.003)	2.702 (0.021)
GK11	100	2 500	1.237 (0.060)	1.677 (0.056)	1.246 (0.067)	0.867 (0.061)	0.052 (0.002)	2.115 (0.035)
average			0.605 (0.057)	1.597 (0.215)	0.631 (0.068)	0.493 (0.043)	0.329 (0.012)	6.353 (0.187)

used for the permutation representation. The best of 1 000 000 such random samples is considered the final solution of a RS run, and the results of 30 runs were averaged.

All EAs performed on all instances substantially better than random search. The EAs with the direct (DI) and weight-biased (WB) representations returned in general the best results, followed by the permutation representation (PE), random-keys (RK), and finally the ordinal representation (OR).

Especially on the largest instances GK07 to GK11, DI clearly outperformed all other approaches. Heuristic initialization, local improvement, and heuristic repair obviously play an important role. For the smaller instances, differences in the gaps of DI and WB are only small. Due to the small standard deviations these differences are nevertheless significant in most cases: t-tests with an error level of 1% reveal that DI is better than all other representations – in particular also WB – on all instances but CB1, CB2, CB4, CB5, and CB7. Only on CB5, WB was significantly better than DI.

Among the evolutionary algorithms, OR performed consistently worst (on 0.1% error level). On CB1 – the smallest instance – all approaches except OR and random search always identified the known optimal solution.

Differences between PE and RK are small and not consistent. On some instances from Chu and Beasley (CB3, CB5, CB6, and CB9), RK obtained significantly smaller gaps on a 1% error level, whereas on some instances from Glover and Kochenberger (GK05 and GK07 to GK10), PE performed better. Observed differences were insignificant in the other cases.

4.3 Convergence and Speed

Figure 1 shows the gap of the best solution per iteration for exemplary runs of each EA variant and random search on problem instance CB5. Most remarkably, WB and DI already started with relatively low gaps in the initial populations, which is due to the heuristic bias in the decoding (WB) and the heuristic initialization (DI), respectively. All other EAs started with random solutions on the boundary of the feasible region, and therefore their gaps were initially nearly identical and much higher. PE and RK converged similarly to reasonably small final gaps. Among the EAs, OR converged slowest and yielded the worst final solution. However, even OR performed substantially better than random search.

Averaged over the 30 runs for instance CB5, DI converged fastest and found its

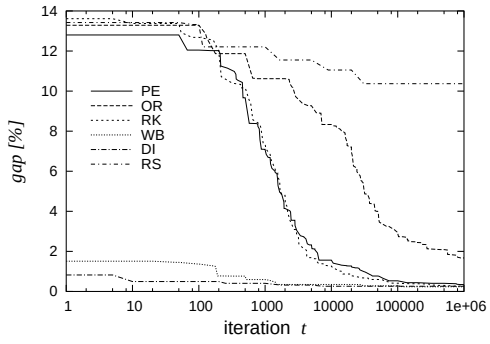

Figure 1: Qualities of so-far best solutions plotted over the iterations for typical runs of the different EA variants and random search on problem instance CB5.

best solution after 179 742 iterations, followed by WB (444 070 iterations), RK (646 632 iterations), PE (689 398 iterations), and finally OR (886 232 iterations). Random search identified its best solution on average in iteration 502 133, which is close to the expected value 500 000.

In our implementation of the EAs, the CPU times on a Pentium-III/500MHz PC for a single run varied from some seconds for the smallest instance CB1 up to about seven hours for the largest instance GK11. DI was generally fastest (on average 4 329 seconds for GK11); PE, RK, and WB needed roughly the same time (≈16 000 seconds), and OR was slowest (25 126 seconds).

4.4 Duplicate Ratio

Newly created offsprings are rejected when they are phenotypically identical to a solution already contained in the population. Creation of these duplicates causes additional computational costs and does not contribute to the progress of the search. These additional computational costs are measured by the *duplicate ratio*, which is the ratio of rejected duplicates among all solutions created.

Table 3 lists the average duplicate ratios for each EA variant and instance. We observe that the high CPU time reported for OR in the previous section is mainly caused by the creation of many duplicates. While OR's duplicate ratio ranges from 29.3% to 40.1%, this ratio is substantially smaller for all the other representations. PE produces on average the least number of duplicates (4.0%), followed by RK (4.7%), WB (5.8%),

Table 3: Average duplicate ratios.

instance			duplicate ratio [%]				
name	m	n	PE	OR	RK	WB	DI
CB1	5	100	4.0	30.5	5.7	10.8	14.2
CB2	5	250	2.5	29.3	3.0	4.8	8.3
CB3	5	500	2.6	30.3	2.9	3.8	8.2
CB4	10	100	2.7	32.9	3.6	6.3	8.7
CB5	10	250	2.8	31.9	3.3	4.6	6.9
CB6	10	500	3.0	32.9	3.5	4.1	7.0
CB7	30	100	3.6	35.5	4.7	6.0	9.7
CB8	30	250	3.8	35.2	4.3	6.2	8.3
CB9	30	500	4.8	35.7	5.4	6.8	10.1
GK01	15	100	3.0	34.0	3.8	4.6	5.6
GK02	25	100	3.1	34.4	3.8	4.7	5.8
GK03	25	150	3.5	33.8	4.3	4.9	9.3
GK04	50	150	3.8	34.9	4.8	5.4	5.1
GK05	25	200	4.3	34.9	5.5	6.5	7.8
GK06	50	200	4.4	35.4	5.3	5.4	7.9
GK07	25	500	4.9	36.0	5.6	5.8	7.1
GK08	50	500	5.0	36.5	5.6	5.8	6.3
GK09	25	1 500	6.1	38.8	6.5	6.7	7.6
GK10	50	1 500	5.9	38.7	6.4	6.4	9.4
GK11	100	2 500	5.8	40.1	6.4	6.6	6.9
average			4.0	34.6	4.7	5.8	8.0

and DI (8.0%).

For PE, OR, RK, and partly WB, we observe the trend that the duplicate ratio increases with the problem size. The reason is that with increasing n, the number of solutions on the boundary of the feasible region does not grow as quickly as the size of the search space. For larger n, on average more elements of the search space are mapped to one phenotype. The smallest instance CB1 is an exception: All EAs except OR find the (known) global optimum usually early during a run. CB1 is therefore a "simple" instance in the sense that there are only few local optima, good local optima are located close to each other, or the global optimum has a large basin of attraction. These conditions cause a fast convergence and the loss of diversity, which is expressed by the high duplicate ratio. A reason for the generally higher ratios of WB and DI in comparison to PE and OR is the heuristic bias of the former representations, which supports the creation of more similar candidate solutions.

5 Basic Concepts

5.1 Spaces

Evolutionary search can be characterized by three spaces, which are shown in Figure 2: the *search space* S, the *phenotype space* P, and the *fitness space* F. In each space, similarity among the elements can be characterized by the notion of distance. The fitness space, which reflects solution quality, is typically scalar and ordered.

In case of decoder-based EAs, a candidate solution is represented by its genotype which is mapped to its phenotype by a decoder. Variation operators work in the genotype space, the set of all possible genotypes, and the EA searches for genotypes mapped to phenotypes of high fitness. Therefore, it is reasonable to use the term *search space* for the genotype space. Here, the search space is obviously different from the phenotype space.

In case of EAs with a direct representation, where variation operators directly work on phenotypes, the search space consists of the same elements as the phenotype space. The distinction between search space and phenotype space is nevertheless important, since different meanings and definitions of distance may be associated to them.

For the MKP, $P = \{0,1\}^n$ and $F = \mathbb{R}_0^+$, and the search space depends on the representation and associated variation operators. Considering for example the permutation representation, the search space S is the set of all permutations of $\{1,\ldots,n\}$.

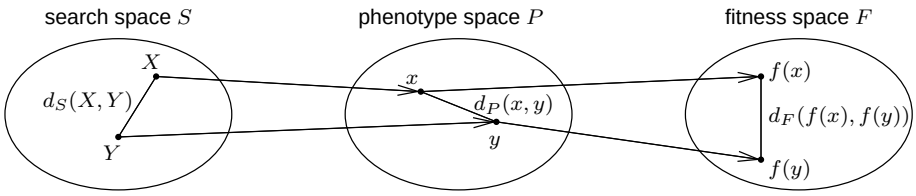

Figure 2: Spaces and associated distances.

5.2 Distances

For judging the similarity or dissimilarity of two elements from a certain space Z, a *distance metric* is usually defined (Ronald, 1997a). In general, a function $d_Z : Z \times Z \to \mathbb{R}_0^+$ is called a distance metric if it satisfies the conditions of non-negativity (8), symmetry

(9), and the triangle-inequality (10)

$$d_Z(x,y) \geq 0 \text{ and } d_Z(x,y) = 0 \leftrightarrow x = y\,, \tag{8}$$
$$d_Z(x,y) = d_Z(y,x)\,, \text{ and} \tag{9}$$
$$d_Z(x,y) \leq d_Z(x,z) + d_Z(z,y) \tag{10}$$

for all $x, y, z \in Z$. In the following, we discuss distances for the three spaces, the search space, the phenotype space, and the fitness space.

Phenotypic distance d_P This distance is defined on the phenotype space P and captures the semantic difference of candidate solutions. It is independent of the used representation, but depends on the problem to be solved.

Definition 5.1 *We quantify the distance of two arbitrary phenotypic solutions $x, y \in P$ as the total number of elementary phenotypic properties in which they differ. For the MKP, elementary properties are the items packed into the knapsack, and the phenotypic distance is the Hamming distance*

$$d_P(x,y) := \sum_{j=1}^{n} |x_j - y_j|\,. \tag{11}$$

Obviously, d_P satisfies the metric conditions (8–10), and $d_P(x,y) \leq n$ for all $x, y \in P$. For many discrete optimization problems, such as the MKP, a phenotypic distance can be defined in a straightforward way. E.g. in case of the traveling salesperson problem, the total number of different edges would be an appropriate measure since edges can be seen as the most important phenotypic properties. Ronald (1997a, 1998) proposed several distance measures for order-based problems.

Search distance d_S The distance for the search space S is supposed to reflect how easily or likely one element can be reached from the other via the variation operator(s). In previous work, $d_S(X,Y)$ was often defined as the minimum number of mutations necessary to transform one element $X \in S$ into another $Y \in S$. Such a definition is suitable, for example, when considering bit strings and a mutation operator that always changes exactly one bit. Also in case of a permutation representation and swap mutation, the minimum number of necessary operations can be efficiently determined. However, sometimes the task of determining this minimum number turns out to be difficult. For example, calculating the minimum number of inversions to transform one permutation into another is NP-hard (Caprara, 1999).

Beside the computational difficulty, this definition of a search distance is not always meaningful. In classical genetic algorithms, each bit is mutated with a certain, low probability. Although only few bits are modified in the expected case, each element of the search space can be reached from each other with a probability greater than zero. Thus, the search distance would be one for all pairs of different elements, which is obviously meaningless.

In other cases, genes do not receive all possible value equally likely; consider the weight-biased representation and its mutation operator based on a log-normal distribution. In such a situation, the minimum number of operations to transform one element into another does not adequately reflect reachability, because different transitions in the search space have different probabilities of being performed.

Alternatively, we may turn to a definition of the search distance which relies on the probability with which one element of the search space will be transformed into the

other. Such definitions, however, often have difficulties in fulfilling the metric conditions. Furthermore, the computation is in practice often difficult or even impossible, in particular when more complex variation operators including repair or local improvement mechanisms with Lamarckian write-back are applied. The MKP's direct representation illustrates this situation.

Due to the described difficulties, we cannot explicitly define meaningful search distances in a consistent way for all the MKP representations we consider here. Instead, we only make the following implicit definition, which is sufficient for our further investigations.

Definition 5.2 *When applying $k \geq 0$ successive mutations (plus repair and local improvement in case of the direct encoding) to a search space element $X \in S$, each element $Y \in S$ is created with a certain probability $P_X^k(Y) \in [0, 1]$. We assume to have a distance function d_S, for which*

$$\sum_{Y \in S} P_X^k(Y) \cdot d_S(X, Y) = k\,. \tag{12}$$

In other words, the average distance from X to an infinite set of search space elements sampled according to the probability distribution P_X^k is assumed to be k. A finite (but large) set of search space elements created in this way approximates the exact case, and thus, its average distance is said to be approximately k.

Note that the above assumption may lead to asymmetric distance functions, which do not fulfill the metric conditions. However, this just reflects the actual situation that the probability for performing a certain transition is not always identical to the probability of the reverse transition.

Fitness distance d_F The definition of a distance on the fitness space F satisfying the metric conditions (8–10) is straightforward:

$$d_F(f(x), f(y)) := |f(x) - f(y)|. \tag{13}$$

We note that we give this definition here only for the reason of completeness, since our methodology focuses on the search distance d_S and the phenotypic distance d_P; the fitness distance d_F is actually not needed in our analysis.

6 Related Work

There are many important properties an evolutionary algorithm usually must fulfill in order to be effective in practice, see e.g. Liepins and Vose (1990). Here, we focus on locality, heritability, and heuristic bias, which we perceive as some of the most essential problem-dependent ingredients for success.

Literature contains some empirical studies about these properties and relationships between the search space, the phenotype space, and the fitness space. In the following, we give an overview of this related work and emphasize differences to our approach.

Manderick et al. (1991) studied correlation coefficients for the fitness values of solutions before and after variation. They conclude that a strong dependency usually exists between these coefficients and performance. Their approach characterizes locality by the relation between the search space and the fitness space, and is therefore generally applicable. Our approach is based on phenotypic distances instead of fitness differences. We feel this is a more accurate basis when the aim is to analyze the internal behavior of an EA, since the phenotypic distance reflects structural differences of

phenotypes, whereas the fitness difference considers only the (scalar) solution quality associated to the phenotypes. As a consequence, a phenotypic distance of zero implies identical fitness, but identical fitness does not imply phenotypic similarity. Anyway, the correlation coefficients proposed by Manderick et al. (1991) may also be interesting measures for our purpose if the locality between phenotype space and fitness space is weak or if the definition of a phenotypic distance is not straightforward.

Jones and Forrest (1995) suggested the fitness distance correlation as a measure of search difficulty, based on the intuition that the fitness should resemble the search distance from a global optimum. This measure correctly predicted difficulty for some problems, but it is not directly applicable to problems with unknown global optima. The search distance to global optima can be approximated, but this may lead to incorrect predictions, as demonstrated by Altenberg (1997).

Fitness landscapes describe the relation between search space and fitness (Jones, 1995), the environment in which local search and evolutionary algorithms are working. Although properties of fitness landscapes – like e.g. the number of local optima or the sizes of their basins of attraction – give insight into potential search dynamics, the total outcome of search remains difficult to predict, even for easy problems (Reeves, 1999). The choice of the representation and operators for a given problem could be based on the difficulties of the corresponding fitness landscapes. However, we are not aware of reliable and generally applicable measures of difficulty for fitness landscapes. For the usefulness of fitness landscapes and the fitness distance correlation in the design of memetic algorithms, see (Merz and Freisleben, 1999).

A locality concept based on the explicit relation between genotypes and phenotypes was suggested by Sendhoff et al. (1997). They used probabilistic measures for mutation operators and claimed that small genotypic changes should imply small phenotypic changes. Their approach was investigated on continuous parameter optimization and structure optimization problems, respectively.

Rothlauf (2002) used genotypic and phenotypic distances to study the impact of the representation on the search complexity, particularly from the perspective of building blocks. Weak locality between genotypes and phenotypes indicates that the representation induces a complexity differing from the original problem's complexity, in terms of the relation between the phenotype space and the fitness. Furthermore, locality in the neighborhood of global optima appeared to be essential for the success of evolutionary search.

Another aspect of evolutionary search, which is related to locality, heritability and heuristic bias, is the redundancy of a representation. It can be defined as the average number of genotypes mapped to the same phenotype. Although redundancy affects the search process (Rothlauf and Goldberg, 2003), its usefulness was disputed (Ronald, 1997b; Knowles and Watson, 2002). In particular, our previous study (Raidl and Gottlieb, 1999) revealed that decoder-based EAs for the MKP suffer significantly from redundancy. The best option was to reduce the effects of redundancy by removing phenotypic duplicates. Therefore, we do not study redundancy here in more detail, but only make the following observations.

Among the representations we consider for the MKP, DI has the lowest redundancy. The permutation representation and the ordinal representation have identical redundancy, which is significantly higher than the redundancy of DI. WB and RK have the highest redundancy due to their real-valued genes.

7 Heuristic Bias

Evolutionary algorithms explore the phenotype space by iteratively applying variation and selection, where the former is responsible for generating new phenotypes and the latter yields a focus on better phenotypes. In addition to selection, which typically works problem-independently by only considering fitness values, there are often various kinds of bias that may support (or hinder) evolutionary search in finding better solutions more quickly.

In the unbiased case, each element of the phenotype space has the same probability of being represented when either randomly choosing an element of the search space or when applying the variation operators to randomly chosen parents. In such EAs, selection is the only force driving the search towards specific phenotypes of high quality.

If some phenotypes have higher probabilities to be created when sampling the search space without any selection pressure, we call this a *bias* towards these phenotypes; other phenotypes are then created with lower probability.

Such bias can be helpful if it favors solutions near optimal solutions. The sampling by evolutionary search – and in particular random search – may benefit from such bias because the expected average fitness of created solutions is higher than without bias. Such kind of bias can be induced by heuristics in the mapping from the search space to the phenotype space or by problem-specific variation operators. Therefore we refer to it as *heuristic bias*.

Usually, heuristic bias has the side effect of reducing diversity. Therefore, it is typically used together with methods that ensure a certain level of diversity, like e.g. phenotypic duplicate elimination.

Whereas heuristic bias can support evolutionary search, bias towards phenotypes with low fitness or towards local optima located far away from global optima with respect to the search distance can obviously be counteractive. Therefore, introducing bias into the search process must be done carefully and in a balanced way.

7.1 Discussion

Only a simple EA applying a direct representation without repair and local improvement but a penalty function for considering constraint violations would be unbiased. All the EA variants we consider here make use of different kinds of heuristic bias. We can order the the approaches by their degree of heuristic bias.

7.1.1 Permutation, Ordinal, and Random-Key Representation

These approaches generate only solutions on the boundary of the feasible region. This fact can be considered as a strong bias, since all other feasible and infeasible solutions have probability zero of being created. The solutions on the boundary have positive probabilities, but these are not identical. This additional unintended bias towards some solutions is due to the items' individual resource consumptions and the first-fit heuristic used to decode permutations. Consider the example of $n = 3$ items, $m = 1$ resource with capacity $c_1 = 2$, and the items' resource consumptions $r_{11} = 2$, $r_{12} = 1$, and $r_{13} = 1$. Among all permutations of the items $\{1, 2, 3\}$, the permutations $(1, 2, 3)$ and $(1, 3, 2)$ are both decoded into the phenotype $(1, 0, 0)$, and the four remaining permutations $(2, 1, 3)$, $(2, 3, 1)$, $(3, 1, 2)$, and $(3, 2, 1)$ are decoded into the phenotype $(0, 1, 1)$. Thus, the phenotype $(0, 1, 1)$ is twice as likely as $(1, 0, 0)$. Note that this effect is caused by the resource constraints only and is independent of the profits of the items. In general, these representations are therefore not particularly biased towards fitter pheno-

types on the boundary.

7.1.2 Weight-Biased Representation

This representation only produces phenotypes on the boundary of the feasible region, too. The initialization, based on a log-normal distribution of the weights, and the sophisticated decoding procedure together provide a strong heuristic bias, since locally optimal solutions near the solution obtained by Pirkul's surrogate duality heuristic for the original problem are favored. The effects of different values for the biasing parameter γ in the weight's initialization function (4) have been studied by Raidl (1999). It was observed that the weight-biased approach works well for a wide range of values for γ, as long as γ is larger than some minimum working bound. Obviously, if γ is too small, the solution created by Pirkul's heuristic for the unbiased problem is favored too much and evolutionary search gets trapped at or near it quickly. For a larger γ, the heuristic bias is smaller and the EA will converge slower but behave robust. If $\gamma \to \infty$, the bias towards the solution of Pirkul's heuristic for the original problem diminishes, and the total heuristic bias is similar to the approaches discussed in Section 7.1.1.

7.1.3 Direct Representation

EAs with a direct representation in which infeasible solutions created by the initialization or variation operators are immediately repaired have the obvious bias towards feasible solutions. If local improvement is applied as well, the heuristic bias is even stronger since only solutions on the boundary of the feasible region are created.

The idea of introducing heuristic bias by local optimization is the key factor for the success of memetic algorithms (Moscato, 1999). Since this technique reduces the actually considered parts of the phenotype space dramatically, care must be taken to prevent the search from getting trapped at poor local optima too easily.

In case of the direct representation for the MKP, the heuristic ordering in which items are considered during repair and local improvement introduces a further level of heuristic bias. This bias favors solutions on the boundary, which are similar to the LP-relaxed solution.

Last but not least, the special initialization of DI yields another heuristic bias, comparable to repairing and local optimization.

7.2 Comparison

The previous paragraphs discussed the heuristic bias of evolutionary algorithms for the MKP in a qualitative way. Now, we analyze the selected approaches empirically in more detail and quantify heuristic bias. ¿From Chu and Beasley's benchmarks with $n = 250$ items and $m = 10$ constraints, we consider the first instance for each tightness ratio $\alpha \in \{0.25, 0.5, 0.75\}$. Note that the instance with $\alpha = 0.5$ is the one named CB5 in Section 4.

For each representation and each of the three instances, 1 000 000 phenotypes were randomly created by the initialization methods. Two different variants are considered in case of DI: In the first one, repair and local improvement were applied to random bit strings (random initialization), while in the second variant, solutions were created by the special heuristic initialization function described in Section 3.6. This allows to separately investigate the heuristic bias of repair combined with local improvement on the one side and the heuristic initialization on the other side.

Figure 3 shows the frequencies with which the solutions' gaps fall into the 0.1%-wide intervals $[0, 0.1\%)$, $[0.1\%, 0.2\%)$, etc. The main tendencies are the same for all three tightness ratios. The gaps are always approximately Gaussian distributed. Mean gaps

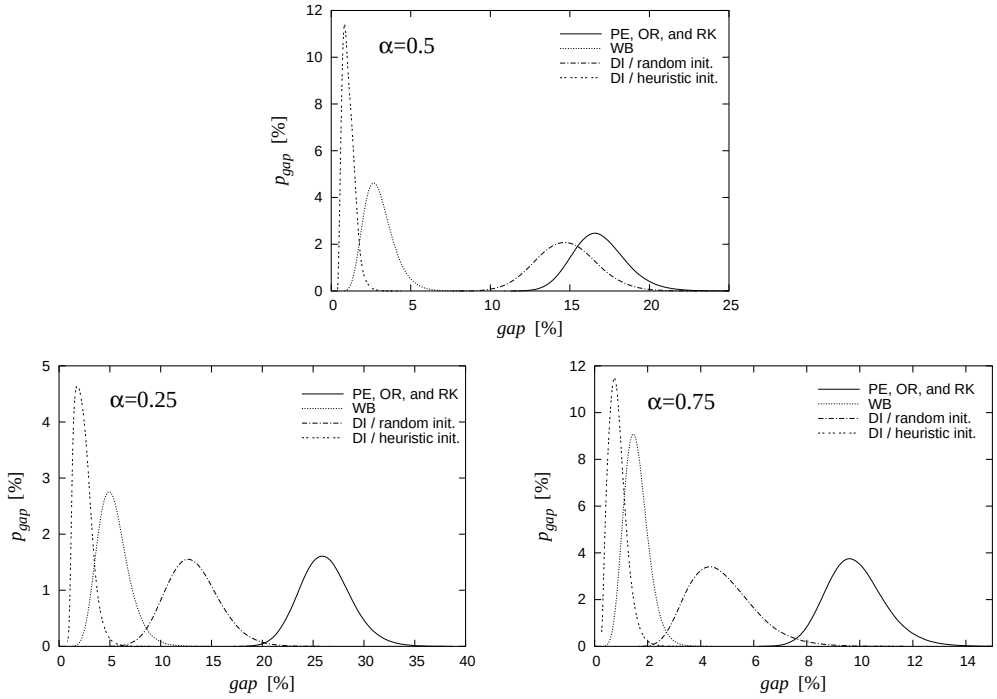

Figure 3: Frequencies of gaps from random solutions on instances with tightness ratios $\alpha \in \{0.25, 0.5, 0.75\}$ to fall into intervals of 0.1% width.

and standard deviations are subsumed in Table 4. PE, OR, and RK exhibit identical distributions with the largest mean values and standard deviations. This is not surprising since these three representations use the same first-fit heuristic in their decoders and have no further heuristic bias. The heuristic initialization of DI exhibits the smallest mean gaps and standard deviations, which indicates its strongest heuristic bias. The combination of DI's random initialization with repair and local improvement exhibits particularly less heuristic bias than DI's heuristic initialization and WB's decoder.

For each representation, the mean gaps of random solutions differ significantly for the three tightness ratios. However, we may not generally conclude that the heuristic bias is stronger for instances with larger tightness ratios. Usually, the solution of the LP-relaxed MKP, on which the definition of the gap relies, represents a tighter bound to the discrete optimum for problem instances where the tightness ratio is large.

Therefore, instead of comparing absolute values of gaps, we should consider relative differences among the different representations and initializations. For PE, OR, RK, WB, and DI's heuristic initialization, the relations between the mean gaps, respectively the standard deviations, remain roughly the same over the three tightness ratios. We conclude that in these cases, the heuristic bias does not strongly depend on the tightness ratio. In contrast, DI's random initialization behaves significantly different on the three instances; the reason is explained in the following.

We first consider the instance with tightness ratio $\alpha = 0.25$. Feasible solutions usually contain only relatively few – about $n/4$ – items due to the low resource capacities. However, in DI's random initialization, 50% of the variables x_j are expected to be ini-

Table 4: Observed mean gaps [%] and corresponding standard deviations σ for random solutions on instances with tightness ratios $\alpha \in \{0.25, 0.5, 0.75\}$.

Representation	$\alpha = 0.25$		$\alpha = 0.5$		$\alpha = 0.75$	
	gap [%]	σ	gap [%]	σ	gap [%]	σ
PE	26.21	2.50	16.86	1.67	9.85	1.10
OR	26.21	2.51	16.86	1.67	9.85	1.11
RK	26.21	2.51	16.86	1.67	9.84	1.10
WB	5.35	1.52	3.00	0.91	1.57	0.45
DI / rand. init.	13.12	2.56	14.77	1.89	4.79	1.22
DI / heur. init.	2.29	0.81	1.07	0.37	0.81	0.34

tially set to 1. The heuristic repair operator has to reset many of them to 0 in order to make the solution feasible. Then, local improvement will typically find only few variables that can be set to 1. Repair therefore dominates the construction of solutions in this case, and it is mainly responsible for the significantly higher heuristic bias of DI's random initialization in comparison to PE, OR, and RK.

In case of $\alpha = 0.75$, the resource capacities are large, and about $3/4$ of all variables are expected to be set to 1 in feasible, locally optimal solutions. Starting from a random selection of items, repair has usually almost nothing to do, but local improvement includes many additional items. Therefore, the heuristic bias comes mainly from local improvement.

When $\alpha = 0.5$, about $n/2$ variables x_j are expected to be set to 1 in a feasible, locally optimal solution. Since this corresponds to the expected number of variables initially set to 1 by DI's random initialization, both, repair and local improvement make usually only small changes. The total heuristic bias is in this case therefore smallest, and the empirically observed mean gap is only a bit smaller than that of PE, OR, and RK. This small difference, however, is nevertheless caused by repairing and local improvement.

To conclude, the stronger heuristic bias of DI and WB is a significant condition for the high performance and quick convergence of the EAs based on these representations. The heuristic bias of DI lies mainly in its special heuristic initialization and only secondly in the repair and local improvement operators, whose impacts depend on the tightness ratio. Nevertheless, repairing and local improvement contribute to the overall heuristic bias of DI, too.

Obviously, heuristic bias does not solely determine performance. In particular, it does not explain the performance differences between PE, OR, and RK observed in Section 4.2. In the next section, we will focus on further important properties, namely locality and heritability of the variation operators.

8 Locality and Heritability

In order to gain insight into the locality and heritability properties a representation with its variation operators provides, several measures and empirical techniques to obtain estimations are discussed in this section. These measures were originally introduced in (Gottlieb and Raidl, 1999). Later, they were adapted to the fixed charge transportation problem (Gottlieb and Eckert, 2000; Eckert and Gottlieb, 2002).

Here and in the remaining parts of this article, problem instance CB5 ($n = 250$, $m = 10$, $\alpha = 0.5$) is used in all empirical investigations, but the general trends remain

also valid for other instances of different size and tightness ratios, as more extensive tests have confirmed.

8.1 Mutation Innovation MI

Mutation operators work in the search space, but their semantic effect can only be analyzed in the phenotype space, which contains structural information about solution candidates. Therefore, we characterize the effect of mutation by the distance between the involved phenotypes. Let $X \in S$ be an element from the search space and $X^m \in S$ the resulting element after applying mutation. In case of the direct representation, we consider repair and local improvement as part of the mutation operator; furthermore, the original solution X is assumed to be feasible and to lie on the boundary, and $x, x^m \in P$ shall be the phenotypes represented by X and X^m, respectively.

Definition 8.1 *The mutation innovation is the phenotypic distance between solution x and the mutated solution x^m,*

$$MI := d_P(x, x^m)\,. \tag{14}$$

MI is a random variable that describes how much "innovation" is introduced into a solution by mutation. Its distribution immediately reflects several important aspects concerning locality.

We investigated the mutation innovation of the five representations on problem instance CB5 empirically by randomly creating 100 000 elements of the search space and applying mutation to all of them. Figure 4 shows a histogram for the resulting distributions of the mutation innovation.

First, we consider the case $MI = 0$, occurring with probability $P(MI = 0)$, in which mutation does not affect the phenotype at all. Large values for $P(MI = 0)$ indicate that either mutation often does not make moves in the search space, or many different elements of the search space map to the same phenotype. The latter possibility reflects a high degree of redundancy or strong heuristic bias.

In all five considered representations, the probabilities $P(MI = 0)$ are relatively high, mainly because the search space is mapped to phenotypes lying on the boundary of the feasible region only. As might be expected, the stronger heuristic bias of WB and DI results in even higher values for these two representations. In general, the high values for $P(MI = 0)$ demonstrate the importance of phenotypic duplicate elimination for maintaining a minimum diversity in the population.

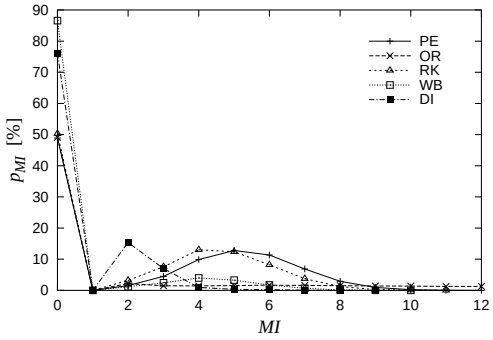

Figure 4: Histograms for the empirical distributions of mutation innovation MI.

There is no possibility that $MI = 1$ in any of the five representations, since $d_P(x, y) \geq 2$ for any pair of distinct solutions $x, y \in P$ lying on the boundary of the feasible region.

When mutation actually modifies the phenotype, thus, $MI > 0$, the distributions of MI become roughly binomial and it makes sense to calculate mean values as estimations for the conditional expected values $E(MI|MI > 0)$ and standard deviations $\sigma(MI|MI > 0)$. In general, a small $E(MI|MI > 0)$ indicates high locality – a single mutation changes the phenotype only slightly – and should therefore be aspired. Large expected values signalize that highly different solutions are frequently generated, locality is weak, and hence the search of the EA tends towards random search.

Table 5 lists empirically obtained values for the considered representations. OR yields substantially higher values for $E(MI|MI > 0)$ and $\sigma(MI|MI > 0)$ than the other representations: 27.83 items, about 11% of all $n = 250$ items, must be expected to change when mutation creates a new, non-duplicate solution. Thus, locality is weak in OR's mutation. All other representations exhibit relatively low expected values. With $E(MI|MI > 0) = 2.46$, which is remarkably close to the lower bound 2 on the distance between two phenotypes on the boundary, DI provides the highest locality, followed by WB, RK and PE with mean values around 5.

In the EA, mutation is – together with crossover, which is the topic of the next section – iteratively applied. How does the distribution of mutation innovation change when $k > 1$ successive mutations are applied? We investigated this case again empirically on instance CB5 by randomly creating 100 000 solutions and mutating each of them 1 000 times. In each sequence of mutations, the phenotypic distance between the original solution and the one created after $k \in \{1, 2, 3, 4, 8, 16, 32, 64, 128, 256, 512, 1\,000\}$ mutations were calculated, i.e. the mutation innovations over k mutations, denoted by MI^k.

As might be expected when considering the mutations to be independent, the probability that the process leaves the phenotype unchanged drops approximately exponentially with increasing k according to the formula

$$P(MI^k = 0) \approx P(MI = 0)^k \,. \tag{15}$$

Figure 5 plots empirically obtained mean values $E(MI^k|MI^k > 0)$ and standard deviations $\sigma(MI^k|MI^k > 0)$ over the number of mutations k. OR exhibits much higher mean values and standard deviations than the other representations as long as k is less than 100. Again, this is a strong indication for OR's weak locality with respect to mutation. For large k, the mean values of PE, OR, and – to slightly lesser degrees DI and RK – approach values around $n/2 = 125$, which is approximately the expected phenotypic distance of two random phenotypes. WB approaches a significantly lower value around 35 due to its strong heuristic bias. In particular, we observe here a significant difference to DI, which is also strongly heuristically biased by its repair and local

Table 5: Characteristic values for the mutation innovation MI.

Measure	PE	OR	RK	WB	DI
$P(MI = 0)$ [%]	48.90	49.12	50.32	86.57	76.09
$E(MI\|MI > 0)$	5.31	27.83	4.66	4.35	2.46
$\sigma(MI\|MI > 0)$	1.56	23.14	1.46	1.35	0.74

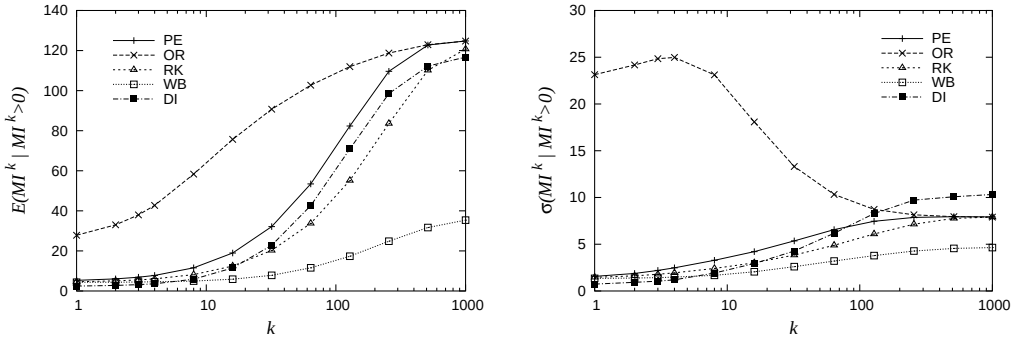

Figure 5: Empirical estimations for $E(MI^k|MI^k > 0)$ and $\sigma(MI^k|MI^k > 0)$ plotted over the number of mutations k.

improvement, but where iterated mutation is also able to create solutions much farther away. This can be an advantage for DI, since it reduces the probability for getting trapped at a local optimum.

8.2 Crossover Innovation CI

The ability of crossover to mix the parents' phenotypic properties in a creative way can be quantified by distances in the phenotype space. In binary crossover, an offspring $X^c \in S$ is created from two parents $X^{p1}, X^{p2} \in S$. Let $x^c, x^{p1}, x^{p2} \in P$ be the corresponding phenotypes. For the direct representation, we again assume that X^{p1} and X^{p2} are feasible and lie on the boundary and repair and local improvement of the offspring is performed after crossover. We adopt the concept of the mutation innovation for crossover in the following way.

Definition 8.2 *The crossover innovation CI is the phenotypic distance between an offspring and its phenotypically closer parent:*

$$CI := \min(d_P(x^c, x^{p1}), d_P(x^c, x^{p2}))\,. \tag{16}$$

The parents X^{p1} and X^{p2}, respectively x^{p1} and x^{p2}, can be interpreted as random variables, and x^c and CI are then dependent random variables.

Obviously, CI is 0 if either $x^c = x^{p1}$ or $x^c = x^{p2}$, thus, if crossover is not able to create a new solution.

In general, the expected value of CI strongly depends on the distance between the parents $d_S(X^{p1}, X^{p2})$, respectively $d_P(x^{p1}, x^{p2})$. Larger parental distances will usually induce larger crossover innovations. The practical meaning is that in an EA, crossover will be able to create more different solutions when the population's diversity is high than when its solutions are similar.

We empirically investigated the crossover innovation for the five representations on randomly created pairs of parent solutions. To consider the dependency on the parental distance in a reasonable way, only each X^{p1} is created independently at random, and the second parent X^{p2} is derived from X^{p1} via $k > 0$ consecutively applied mutations. In case of DI, X^{p2} was also repaired and locally improved. Furthermore, the duplicate elimination of the EA framework was taken into account by discarding all pairs of phenotypically identical parents; thus, $d_P(x^{p1}, x^{p2}) > 0$ always holds. The expected phenotypic distance between the parents is therefore

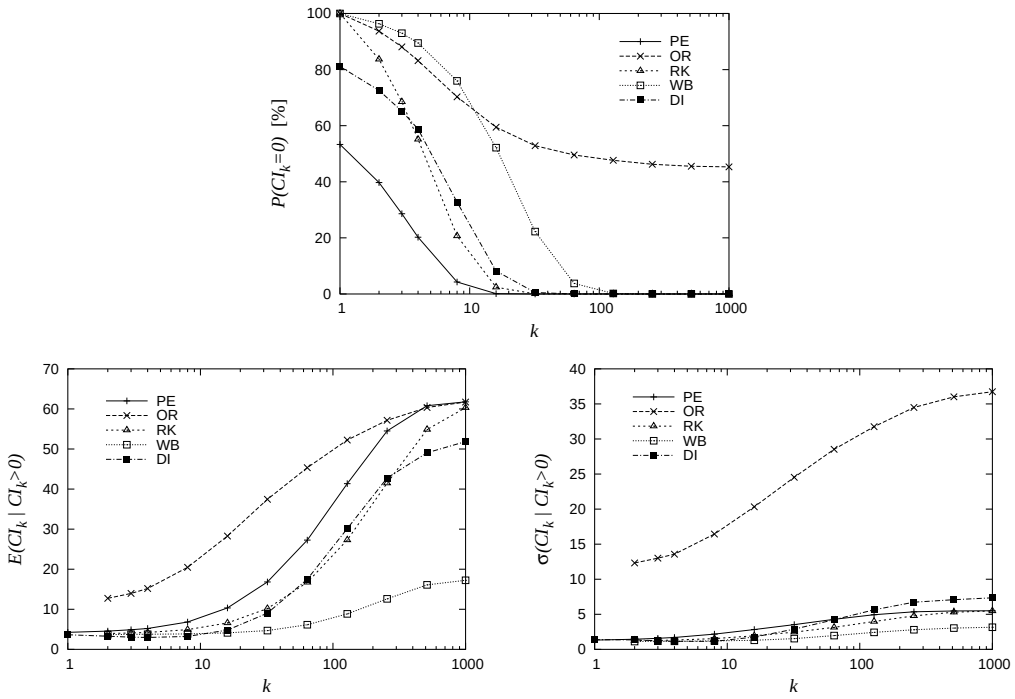

Figure 6: Crossover innovation: Empirically observed probabilities for $CI_k = 0$ and estimated expectations $E(CI_k|CI_k > 0)$ with standard deviations $\sigma(CI_k|CI_k > 0)$.

$E(d_P(x^{p1}, x^{p2})) = E(MI^k|MI^k > 0)$, as it was shown in Figure 5.

In the following, we denote the crossover innovation for the case where the second parent was derived from the first via k mutations by CI_k. 100 000 pairs of parents were created for each $k \in \{1, 2, 3, 4, 8, 16, 32, 64, 128, 256, 512, 1000\}$. Figure 6 plots the observed probabilities for $CI_k = 0$ and the empirically estimated conditional expectations $E(CI_k|CI_k > 0)$ with standard deviations $\sigma(CI_k|CI_k > 0)$ over k.

If $k = 1$, the one-point crossover of OR and the uniform crossover of RK and WB are unable to produce new solutions; thus $P(CI_1 = 0) = 100\%$, and no values therefore exist for $E(CI_1|CI_1 > 0)$ and $\sigma(CI_1|CI_1 > 0)$. In case of PE and $k = 1$, swap mutation is used to derive the second parent, and in combination with uniform order based crossover, solutions different to the parents can be created with probability $P(CI_1 > 0) \approx 47\%$. DI uses uniform crossover, as RK and WB do, but it is able to create new solutions for $k = 1$ with probability $P(CI_1 > 0) \approx 20\%$. This is possible since repair and local improvement are applied and mutation changes each bit with a certain probability.

The generally relatively high probabilities $P(CI_k = 0)$ for small k emphasize the importance of taking care of the population diversity in the EA, as the considered EA framework does by discarding phenotypically identical solutions. In general, $P(CI_k = 0)$ decreases with increasing k. Whereas it approaches 0 in case of PE, RK, WB and DI, it remains relatively high ($> 45\%$) for OR. The reason is OR's one-point crossover which frequently exchanges only genes in the rear part of the genotype that have no effect during decoding. In OR, the phenotype is mainly determined by the genes in the

front. The large duplicate ratios of OR observed in Section 4.4 can now be explained by the high probabilities of both, $CI_k = 0$ and $MI = 0$.

The plots for $E(CI_k|CI_k > 0)$ and $\sigma(CI_k|CI_k > 0)$ are indicators for locality of crossover. $E(CI_k|CI_k > 0)$ should be small for small k and become larger for increasing k. In more detail, the distance of the offspring to both of its parents should usually not be larger than the distance between the parents, i.e. $E(CI_k|CI_k > 0) \leq E(MI^k|MI^k > 0)$ should hold for any k.

This condition holds for all considered representations. Thus, CI_k gives no indication for weak locality of any of our representations concerning crossover. OR's $E(CI_k|CI_k > 0)$ is particularly larger than those of the other representations, but this follows from the large phenotypic distances of the parents ($E(MI^k|MI^k > 0)$) due to the weak locality of mutation. WB's $E(CI_k|CI_k > 0)$ is very low, which coincides with its low mutation innovation displayed in Figure 5. The reason is the strong heuristic bias of WB.

8.3 Crossover Loss CL

A meaningful crossover operator should be able to create offsprings mostly out of substructures from the parental phenotypes. The following measure quantifies how strongly this condition is violated.

Definition 8.3 *The crossover loss CL is the total size of phenotypic substructures (the amount of phenotypic properties) in an offspring x^c that is not inherited from either of the parents x^{p1} and x^{p2} but newly introduced.*

For the MKP this means

$$CL := \sum_{j=1}^{n} \delta(x_j^c, x_j^{p1}, x_j^{p2}) \tag{17}$$

$$\text{with} \quad \delta(x_j^c, x_j^{p1}, x_j^{p2}) = \begin{cases} 0 & \text{if } x_j^c = x_j^{p1} \text{ or } x_j^c = x_j^{p2}, \\ 1 & \text{otherwise.} \end{cases} \tag{18}$$

Based on the definition of the phenotypic distance metric, this can also be written as

$$CL := \frac{1}{2}(d_P(x^c, x^{p1}) + d_P(x^c, x^{p2}) - d_P(x^{p1}, x^{p2}))\,. \tag{19}$$

Note that $CI = 0$ implies $CL = 0$, but not vice versa. We preclude the case where crossover is not able to create a new solution and consider the expected value $E(CL|CI > 0)$ and its standard deviation $\sigma(CL|CI > 0)$.

Empirical estimations were calculated based on the experiments described in the previous section. CL_k denotes the crossover loss when the second parent is created out of the first, randomly generated one by k mutations. Figure 7 shows these estimated $E(CL_k|CI_k > 0)$ and $\sigma(CL_k|CI_k > 0)$ plotted over k for the five representations.

In general, DI exhibits the smallest expected crossover loss and standard deviation, and thus, the strongest heritability. The corresponding values of WB, RK, and PE are slightly larger, especially for $k \geq 10$, but still reasonable. OR exhibits substantially larger values for $E(CL_k|CI_k > 0)$, and they increase significantly with increasing k. The standard deviation $\sigma(CL_k|CI_k > 0)$ follows this trend. We conclude that OR's heritability is particularly poor when the parents are very different. The reason is again the strong dependency of the meaning of each gene on all its preceding genes. In case of $k = 1\,000$, about 28 variables receive on average new values that are not inherited from

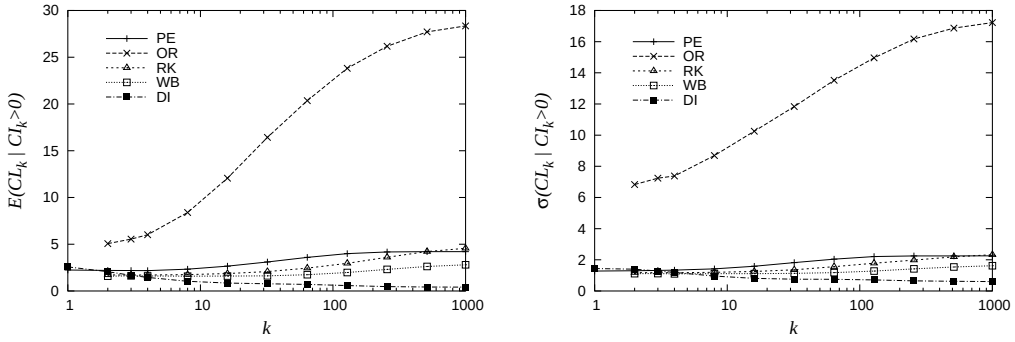

Figure 7: Crossover loss: Empirically estimated $E(CL_k|CI_k > 0)$ and $\sigma(CL_k|CI_k > 0)$.

the parent solutions. These are $\approx 11\%$ of all 250 variables, and they account for about the half of the differences between the offspring and its closer parent, i.e. $E(CL_k|CI_k > 0) \approx 1/2 \cdot E(CI_k|CI_k > 0)$.

9 Search Dynamics

The previous section investigated the representations in combination with their variation operators and decoders, respectively repair and local improvement. That analysis revealed strong indicators for various conditions that affect performance. All these *static* investigations were done without performing complete runs of evolutionary algorithms. Obviously, the iterated interplay of selection, variation operators, and replacement strategy in the EA exhibits its own dynamics. Therefore, this section studies the complete *dynamic* case and confirms that the results obtained by the static measurements remain valid and accurate.

In evolutionary search, the behavior of the variation operators is expected to depend on the current population's properties. Therefore, we observed the measures introduced in the previous section also during actual runs of the EA. At each iteration t of such a run, the phenotypic distance of parents selected for crossover $PD_t = d_P^t(x^{p1}, x^{p2})$, the crossover innovation CI_t, the crossover loss CL_t, and the mutation innovation MI_t were calculated.

The descriptive values $E(PD_t)$, $P(MI_t = 0)$, $E(MI_t|MI_t > 0)$, $P(CI_t = 0)$, $E(CI_t|CI_t > 0)$, $\sigma(CI_t|CI_t > 0)$, $E(CL_t|CI_t > 0)$, and in addition the probability that the created offspring is a duplicate $P(dup_t)$ were estimated in dependence of t. We divided the whole run into consecutive observation intervals and calculated the proportions, respectively conditional mean values and standard deviation for each interval.

The steady-state framework with the properties described in Section 4 was used, and again, we consider instance CB5 as representative example. Since the population dynamics usually change faster in early phases of a run, the size of the observation intervals is small at the beginning and increases with t. We start with intervals of size 10 and multiply this size by 10 after iterations 100, 1 000, 10 000, and 100 000. In order to further increase the confidence of the results, 20 independent runs were performed, and all sample values of the corresponding intervals were merged.

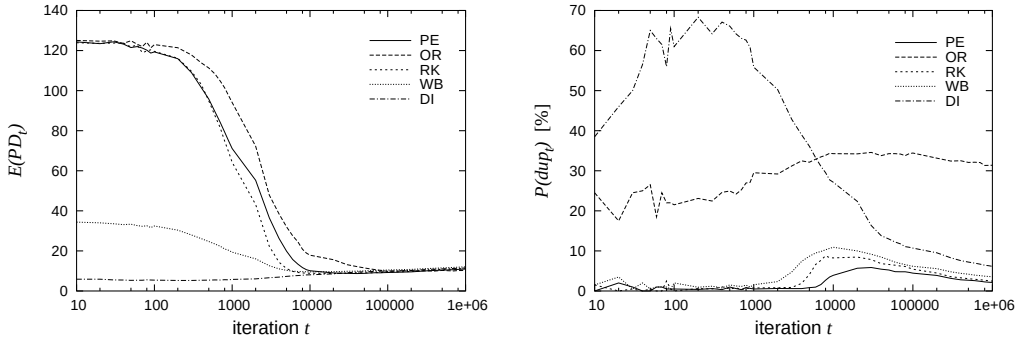

Figure 8: Dynamic investigations: Average phenotypic distance of crossover's parents PD_t and estimated probability of creating duplicates $P(dup_t)$.

9.1 Mean Distance of Crossover Parents and Probability of Duplicates

Figure 8 shows for all five representations the dynamics of the phenotypic mean distance of crossover parents $E(PD_t)$ and the probability of duplicates $P(dup_t)$ plotted over t. $E(PD_t)$ is a good indicator for the diversity in the population. PE, OR, and RK start with large values, thus, high diversity. We already know that the initial diversity is significantly smaller for WB and in particular DI due to their strong heuristic bias, and the initial $E(PD_t)$ is therefore also much smaller in these cases. After about 10 000 iterations, the diversity has decreased substantially in all other representations as well. The curves of PE, RK, WB, and DI meet at the same value of about 9.5. Obviously, the duplicate elimination scheme avoids significantly smaller average parent distances and, thus, the total loss of diversity.

A surprise may be the behavior after this point, which we call *post-convergence diversity increase*: $E(PD_t)$ increases in all cases to a final value of ≈ 12. The reason for this behavior is that the population has already converged in highly fit regions of the search space during the first phase of a run; the neighborhoods of identified local optima have been searched well, and the best solutions of these regions are contained in the population. Due to the used replacement strategy, new solutions will only remain in the population if they are at least as good as the other population members and phenotypically different. This implies that such accepted new solutions usually do not lie in the already well searched regions. Thus, these solutions are typically farther away from the rest of the population, and diversity is increased.

The population's diversity is also reflected by the observed probabilities of creating duplicates $P(dup_t)$. In case of PE, RK, and WB, this probability is low at the beginning and increases after some thousand iterations. Due to the post-convergence diversity increase, $P(dup_t)$ decreases during the last phase of a run. The duplicate ratio of OR is generally much larger (as already observed in Section 4), but otherwise behaves similarly. Considering DI, $P(dup_t)$ is particularly high at the beginning due to the strong heuristic bias of initialization, but decreases strongly after about thousand iterations.

9.2 Mutation Innovation

Figure 9 shows the dynamics of mutation innovation. The measures $P(MI_t = 0)$ and $E(MI_t|MI_t > 0)$ do not change significantly during a run and closely correspond to the values obtained in the static measurements, compare Table 5. This is quite intuitive

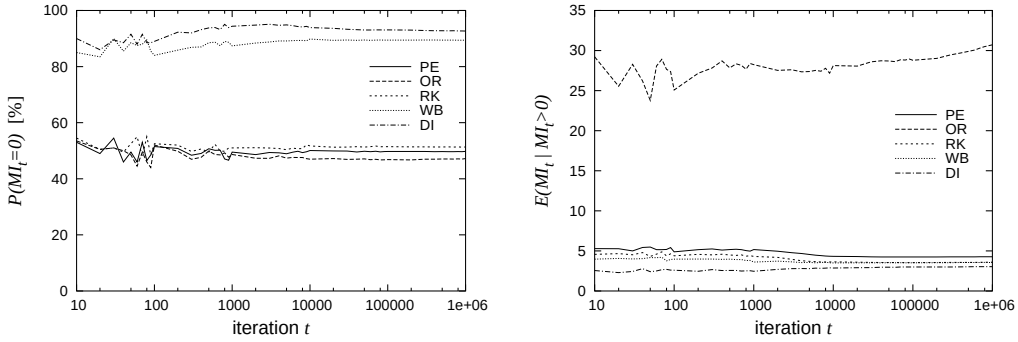

Figure 9: Dynamic investigations: Mutation innovation MI_t.

and proves that mutation innovation does not depend on the population dynamics and in particular the population diversity.

9.3 Crossover Innovation and Crossover Loss

Figure 10 shows dynamic results for crossover. The probability of creating an offspring that is identical to one of its parents $P(CI_t = 0)$ is always low for PE, RK, and WB, and relatively high for OR. In case of DI, $P(CI_t = 0)$ is high at the beginning of a run due to DI's low initial diversity, but it decreases after about thousand iterations. For all representations except DI, the conditional mean value $E(CI_t|CI_t > 0)$ is relatively large at the beginning of a run and decreases to ≈ 5. The smaller initial values of WB and in particular DI are due to their heuristic biases and the lower diversity. The conditional mean crossover loss $E(CL_t|CI_t > 0)$ is small in all cases except OR, which exhibits relatively large values during the first few thousand iterations because of the poor heritability of its crossover.

9.4 Comparison with Static Analysis

The observations and differences in the search dynamics among the representations are not surprising, since they conform to the results of the previous static measurements. During the static measurements for crossover, the degree of population diversity was modeled by deriving the second parent from the first via k mutations. Figure 5 showed the conditional mean mutation innovation $E(MI^k|MI^k > 0)$ for such a sequence of k mutations, and thus, the phenotypic mean distances of crossover parents $E(PD_k)$. We use $E(MI^k|MI^k > 0)$ to find values for k that induce approximately the same phenotypic mean parent distances as observed during the dynamic measurements.

At the beginning of an EA run, initial solutions are created independently at random. With the exception of DI, the highest average parent distance $E(PD_t)$ is provided at this point. In our static investigations, $k = 1\,000$ yields the best approximation of this case. For PE, OR, RK, and WB, all dynamically observed measures $P(CI_t = 0)$, $E(CI_t|CI_t > 0)$, $\sigma(CI_t|CI_t > 0)$, and $E(CL_t|CI_t > 0)$ with $t = 10$ are approximately equal to the corresponding static values with $k = 1\,000$. DI is an exception since it applies heuristic initialization. It starts with a particularly lower parent distance $E(PD_t) \approx 5$, which is approximately reached in the static measurements with $k = 10$. With this value, DI's static measures for crossover innovation and crossover loss also almost correspond with its dynamic ones.

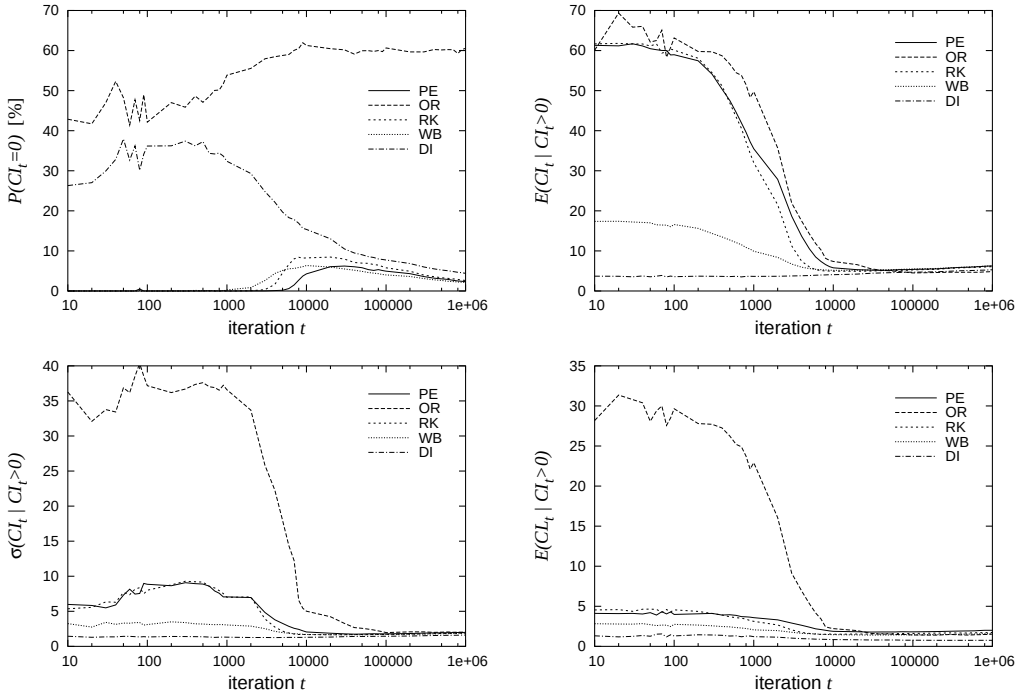

Figure 10: Dynamic investigations: Crossover innovation CI_t and crossover loss CL_t.

Next, we consider as example iteration $t = 2\,000$ of the dynamic measurements. The observed values for the phenotypic parent distance are reached in the static measures when $k \approx 64$ for PE, $k \approx 16$ for OR, $k \approx 100$ for RK and WB, and $k \approx 10$ for DI. Again, static measures with these k-values almost correspond with the dynamic measures at $t = 2\,000$.

This approximate conformity of dynamic and static results holds consistently for any iteration. There is only one restriction regarding OR: Due to the poor locality of OR's mutation, the static measurements were not able to model the situation of very low diversity. The average phenotypic distance of crossover parents in the static measurements is never less than the expected innovation of a single mutation $E(MI|MI > 0) \approx 27.83$. However, in the dynamic measurements $E(PD_t)$ becomes smaller than this value after about 6 000 iterations.

10 Conclusions

We have compared five different representations and associated variation operators for the MKP. Four of them are decoder-based, and the fifth is a direct encoding including local improvement and repair. The basic ideas of these representations have already been applied to a variety of other combinatorial optimization problems, too. The complex decoders, respectively local improvement and repair strategies, of these representations make it practically impossible to analyze an EA applying these representations in a fully theoretical way.

This article proposed a hands-on approach to study properties of the representation and its variation operators that substantially influence the performance of such

EAs. Inexpensive static measurements based on randomly created solutions were applied to gain insight into the behavior of decoder, initialization, repair, local improvement, mutation, and crossover. The key properties we focused on are heuristic bias, locality, and heritability. Based on a phenotypic distance metric, measures were introduced for mutation innovation, crossover innovation, and crossover loss in order to quantify and "visualize" different aspects of mutation and crossover. The distribution of fitness values was used to characterize heuristic bias.

In addition to the static measurements, observations during actual EA runs were performed. More information about dynamic aspects, mainly concerning diversity, could be gathered, and it was shown that the results of the isolated static measurements remain valid in the dynamic case.

From the five considered MKP representations, DI exhibits the highest heuristic bias and strongest locality and heritability. Its diversity is generally low, but the variation operators in combination with the phenotypic duplicate elimination of the used EA framework guarantee the necessary amount of innovation for making progress in the search process. This combination of good properties explains why DI achieves the best performance among the considered representations. The weaker effectivity of the other considered EAs can be mainly explained by their weaker locality and heritability (in particular OR) and their weaker heuristic bias.

Specific results were mainly shown for a single problem instance. For other instances, absolute values of results are obviously different. However, extensive tests on differently structured instances have consistently shown the same qualitative trends for the five considered representations.

The introduced measures quantify essential aspects of evolutionary search, which are often used intuitively without formalization. In general, the described measurements represent a framework that can aid the design process of EAs. Performing the static measurements on candidates of representations and operators allows for a relatively inexpensive pre-evaluation without performing extensive complete EA runs. In this way, particular strengths or weaknesses can be noticed early, and the general performance of an EA can be predicted to some degree. The decision of which representation and combination of operators one should finally use is supported.

The proposed approach can easily be applied to various other combinatorial optimization problems. Basically, only an appropriate phenotype distance metric is needed to analyze representations and variation operators for other problems.

Acknowledgments

The first author's work is supported by the Austrian Science Fund (FWF) under the grant P16263–N04.

References

Altenberg, L. (1997). Fitness distance correlation analysis: An instructive counterexample. In Bäck (1997), pages 57–64.

Angeline, P. J. et al., editors (1999). *Proceedings of the 1999 IEEE Congress on Evolutionary Computation*. IEEE Press.

Bäck, T., editor (1997). *Proceedings of the 7th International Conference on Genetic Algorithms*. Morgan Kaufmann.

Bean, J. C. (1994). Genetic algorithms and random keys for sequencing and optimization. *ORSA Journal on Computing*, 6(2):154–160.

Beasley, J. E. (1996). Obtaining test problems via internet. *Journal of Global Optimization*, 8:429–433.

Bosman, P. A. N. and Thierens, D. (2002). Permutation optimization by iterated estimation of random keys marginal product factorizations. In Merelo et al. (2002), pages 331–340.

Bruhn, P. and Geyer-Schulz, A. (2002). Genetic programming over context-free languages with linear constraints for the knapsack problem: First results. *Evolutionary Computation*, 10(1):51–74.

Caprara, A. (1999). Sorting permutations by reversals and Eulerian cycle decompositions. *SIAM Journal on Discrete Mathematics*, 12:91–110.

Chu, P. C. and Beasley, J. E. (1998). A genetic algorithm for the multidimensional knapsack problem. *Journal of Heuristics*, 4:63–86.

Corne, D., Dorigo, M., and Glover, F. (1999). *New Ideas in Optimization*. McGraw-Hill, Berkshire, England.

Cotta, C. and Troya, J. M. (1998). A hybrid genetic algorithm for the 0–1 multiple knapsack problem. In *Proceedings of the 3rd International Conference on Artificial Neural Nets and Genetic Algorithms*, pages 250–254.

Eberhart, R. et al., editors (1997). *Proceedings of the 1997 IEEE International Conference on Evolutionary Computation*. IEEE Press.

Eckert, C. and Gottlieb, J. (2002). Direct representation and variation operators for the fixed charge transportation problem. In Merelo et al. (2002), pages 77–87.

Eshelman, L. J., editor (1995). *Proceedings of the 6th International Conference on Genetic Algorithms*. Morgan Kaufmann.

Fogel, D. B., editor (1998). *Proceedings of the 1998 IEEE International Conference on Evolutionary Computation*. IEEE Press.

Fonlupt, C. et al., editors (1999). *Artificial Evolution: Fourth European Conference*, volume 1829 of *LNCS*. Springer.

Garey, M. R. and Johnson, D. S. (1979). *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, New York.

Gottlieb, J. (1999). *Evolutionary Algorithms for Constrained Optimization Problems*. PhD thesis, Department of Computer Science, Technical University of Clausthal.

Gottlieb, J. and Eckert, C. (2000). A comparison of two representations for the fixed charge transportation problem. In Schoenauer, M. et al., editors, *Parallel Problem Solving from Nature – PPSN VI*, volume 1917 of *LNCS*, pages 345–354. Springer.

Gottlieb, J. and Raidl, G. R. (1999). Characterizing locality in decoder-based EAs for the multidimensional knapsack problem. In Fonlupt et al. (1999), pages 38–52.

Gottlieb, J. and Raidl, G. R. (2000). The effects of locality on the dynamics of decoder-based evolutionary search. In Whitley, D. et al., editors, *Proceedings of the 2000 Genetic and Evolutionary Computation Conference*, pages 283–290. Morgan Kaufman.

Grefenstette, J. J., Gopal, R., Rosmaita, B. J., and Gucht, D. V. (1985). Genetic algorithms for the traveling salesman problem. In Grefenstette, J. J., editor, *Proceedings of the First International Conference on Genetic Algorithms*, pages 160–168. Lawrence Erlbaum.

Hinterding, R. (1994). Mapping, order-independent genes and the knapsack problem. In Schaffer, D., Schwefel, H.-P., and Fogel, D. B., editors, *Proceedings of the First IEEE Conference on Evolutionary Computation*, pages 13–17. IEEE Press.

Hinterding, R. (1999). Representation, constraint satisfaction and the knapsack problem. In Angeline et al. (1999), pages 1286–1292.

Jones, T. (1995). *Evolutionary Algorithms, Fitness Landscapes and Search*. PhD thesis, University of New Mexico, Albuquerque, New Mexico.

Jones, T. and Forrest, S. (1995). Fitness distance correlation as a measure of problem difficulty for genetic algorithms. In Eshelman (1995), pages 184–192.

Julstrom, B. A. (1997). Strings of weights as chromosomes in genetic algorithms for combinatorial problems. In Alander, J. T., editor, *Proceedings of the Third Nordic Workshop on Genetic Algorithms and their Applications*, pages 33–48.

Kellerer, H., Pferschy, U., and Pisinger, D. (2004). *Knapsack Problems*. Springer.

Knowles, J. D. and Watson, R. A. (2002). On the utility of redundant encodings in mutation-based evolutionary search. In Merelo et al. (2002), pages 88–98.

Levenhagen, J., Bortfeldt, A., and Gehring, H. (2001). Path tracing in genetic algorithms applied to the multiconstrained knapsack problem. In Boers, E. J. W. et al., editors, *Applications of Evolutionary Computing – EvoWorkshops 2001*, volume 2037 of *LNCS*, pages 40–49. Springer.

Liepins, G. and Vose, M. (1990). Representation issues in genetic algorithms. *Journal of Experimental and Theoretical Artificial Intelligence*, 2:101–115.

Manderick, B., de Weger, M., and Spiessens, P. (1991). The genetic algorithm and the structure of the fitness landscape. In Belew, R. K. and Booker, L. B., editors, *Proceedings of the 4th International Conference on Genetic Algorithms*, pages 143–150. Morgan Kaufmann.

Martello, S. and Toth, P. (1990). *Knapsack Problems: Algorithms and Computer Implementations*. J. Wiley & Sons.

Merelo, J. J. et al., editors (2002). *Parallel Problem Solving from Nature – PPSN VII*, volume 2439 of *LNCS*. Springer.

Merz, P. and Freisleben, B. (1999). Fitness landscapes and memetic algorithm design. In Corne et al. (1999), pages 245–260.

Michalewicz, Z. and Arabas, J. (1994). Genetic algorithms for the 0/1 knapsack problem. In Raś, Z. W. and Zemankova, Z., editors, *Proceedings of the 8th International Symposium on Methodologies for Intelligent Systems*, volume 869 of *LNAI*, pages 134–143. Springer.

Moscato, P. (1999). Memetic algorithms: A short introduction. In Corne et al. (1999), pages 219–234.

Pirkul, H. (1987). A heuristic solution procedure for the multiconstrained zero-one knapsack problem. *Naval Research Logistics*, 34:161–172.

Raidl, G. R. (1998). An improved genetic algorithm for the multiconstrained 0–1 knapsack problem. In Fogel (1998), pages 207–211.

Raidl, G. R. (1999). Weight-codings in a genetic algorithm for the multiconstraint knapsack problem. In Angeline et al. (1999), pages 596–603.

Raidl, G. R. and Gottlieb, J. (1999). On the importance of phenotypic duplicate elimination in decoder-based evolutionary algorithms. In Brave, S. and Wu, A. S., editors, *Late Breaking Papers at the 1999 Genetic and Evolutionary Computation Conference*, pages 204–211, Orlando, FL.

Reeves, C. (1999). Fitness landscapes and evolutionary algorithms. In Fonlupt et al. (1999), pages 3–20.

Ronald, S. (1997a). Distance functions for order-based encodings. In Eberhart et al. (1997), pages 49–54.

Ronald, S. (1997b). Robust encodings in genetic algorithms. In Dasgupta, D. and Michalewicz, Z., editors, *Evolutionary Algorithms in Engineering Applications*, pages 29–44. Springer.

Ronald, S. (1998). More distance functions for order-based encodings. In Fogel (1998), pages 558–563.

Rothlauf, F. (2002). *Representations for Genetic and Evolutionary Algorithms*. Physica, Heidelberg.

Rothlauf, F. and Goldberg, D. E. (2003). Redundant representations in evolutionary computation. *Evolutionary Computation*, 11(4):381–415.

Rothlauf, F., Goldberg, D. E., and Heinzl, A. (2002). Network random keys – a tree representation scheme for genetic and evolutionary algorithms. *Evolutionary Computation*, 10(1):75–97.

Sendhoff, B., Kreutz, M., and von Seelen, W. (1997). A condition for the genotype-phenotype mapping: Causality. In Bäck (1997), pages 73–80.

Thiel, J. and Voss, S. (1994). Some experiences on solving multiconstraint zero-one knapsack problems with genetic algorithms. *INFOR*, 32:226–242.

Vasquez, M. and Hao, J.-K. (2001). A hybrid approach for the 0–1 multidimensional knapsack problem. In *Proceedings of the 17th International Joint Conference on Artificial Intelligence*, pages 328–333.