Towards an Analysis of Dynamic Environments

Jürgen Branke
Institute AIFB
University of Karlsruhe
76128 Karlsruhe, Germany
branke@aifb.uni-karlsruhe.de

Erdem Salihoğlu
Istanbul Technical University
Informatics Institute
Maslak 34469 Istanbul, Turkey
salihoglu@be.itu.edu.tr

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

ABSTRACT

Although the interest in nature-inspired optimization of dynamic problems has been growing constantly over the past decade, very little has been done to analyze and characterize a changing fitness landscape. However, it would be very helpful for algorithm development to have a better understanding of the nature of fitness changes in dynamic real-world problems. In this paper, we propose a number of measures that can be used to analyze and characterize the dynamism in a problem changing over time. Additionally, we introduce a new dynamic multi-dimensional knapsack problem as a close-to-real-world test problem.

Categories and Subject Descriptors: I.2.8 Heuristic Algorithms

General Terms: Algorithms.

Keywords: evolutionary algorithm, dynamic environment, characterizing dynamism

1. INTRODUCTION

Many real-world optimization problems are changing over time, requiring repeated re-optimization, and a continuous tracking of the changing optimum. Examples include scheduling, where new jobs arrive over time, or vehicle routing, with new requests arriving over time. Evolutionary algorithms (EAs) have shown some promise for this domain, and the number of publications in the field rose steadily over the past decade. For an overview on EA approaches to dynamic environments see e.g. [7, 8, 32].

Although many of the approaches have shown to work well, most of them have been applied to simple artificial benchmark problems only, and little has been done to characterize and understand the nature of a change in real-world problems.

When trying to characterize static problems, an important concept is that of a fitness landscape. A fitness landscape is an intuitive model of what situations a search algorithm might encounter, and depends not only on the problem or fitness function, but also on the definition of a neighborhood structure on the search space, which, in turn, depends on the representation and search operators, i.e., the algorithm. The fitness landscape model allows to use intuitive descriptions like local optima, peaks and valleys, or ruggedness.

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

The aim of this paper is to take the idea of a fitness landscape one step further, and to analyze changes of the fitness landscape due to changes of the underlying problem instance. For that purpose, we propose a number of measures that may be helpful to understand what aspects of the fitness landscape change, and most importantly, what information can be carried over from one stage of the problem to the next. The latter is particularly important when designing algorithms for dynamic optimization problems, because it is the information that carries over from one stage to the next that can be exploited and that may result in algorithms which perform better than simple restart whenever the problem changes.

The paper is structured as follows. Some related work is reviewed in Section 2. The proposed measures to characterize change are defined in Section 3. In Section 4, we define the dynamic benchmark problem used in this study. Experimental results are reported in Section 5. The paper concludes with a summary and ideas for future work.

2. RELATED WORK

Fitness landscapes of static problems have been examined e.g. in [18, 23, 28]. An alternative attempt to characterize a search space is the concept of epistasis, see e.g. [10].

There have been previous attempts to classify and categorize dynamic environments, without necessarily attempting to actually measure the characteristics. Branke [5, 7] uses the following main criteria to characterize changing environments:

- *Frequency of change:* How often does the environment change. For optimization purposes, that usually means how many fitness evaluations can be performed between changes.
- *Severity of change:* This is usually characterized by the distance from the old to the new optimum.
- *Predictability of change:* Are the changes purely random or do they follow a pattern that could be learned.
- *Cycle length / cycle accuracy:* This defines, for cycling environments, how long it takes until the environment returns to a previous state, and how accurately it will return to it.

De Jong [15] proposes a different classification of problems:

- *Alternating problems*
- *Problems with changing morphology*, i.e., the fitness landscape changes according to certain topological rules
- *Drifting landscapes*, and
- *Abrupt and discontinuous problems*.

Weicker [32] aims at establishing a mapping between different types of dynamic optimization problems, techniques and performance measures. To achieve this, he proposes a mathematical framework for classifying and comparing dynamic problems. Within the framework, the dynamics of a problem are formally defined through defining different changes (rotations, fitness rescalings and coordinate transformations) for components of a decomposable fitness function. However, as the author notes himself, the defined properties like severity, predictability and homogeneity can only be determined for a specific domain of artificial benchmark problems.

While the above mentioned criteria have been used as inspiration for designing artificial benchmark problems, to our knowledge, no one has ever attempted to characterize and measure the dynamism of the fitness landscape of a given dynamic real-world problem.

Many authors have proposed simple artificial benchmark problems that allow to exactly determine the problem dynamics. Examples include the dynamic bit-matching problem [29], the moving sphere [1], or a multi-peak environment where peak-heights change over time [30]. One of the most prominent artificial benchmark problems is the moving peaks problem that has been proposed concurrently and independently in [6, 20]. It consists of a set of peaks changing over time in location, height, and width. Richter [24] has recently modified this benchmark, allowing to influence the degree of chaos in the changes as measured by the Lyapunov exponent. Although these artificial benchmark problems are quite popular, it is not clear how closely their dynamics resemble the dynamics of real-world problems.

One of the few real-world dynamic benchmark problems are job shop scheduling (e.g., [4]) and a greenhouse simulator [31]. The latter, however, proved to be not very challenging to evolutionary algorithms as it features extremely simple search landscapes.

In this paper, we are using a dynamic multi-dimensional knapsack problem where profits, weights and capacities may change over time (see Section 4).

There are also a number of theoretical investigations on how the path of evolution is influenced by moving or oscillating fitness landscapes [2, 17, 25, 26, 27, 33].

3. CHARACTERIZING CHANGE

In this section, we propose a number of measures to characterize the nature of a changing landscape, and discuss their advantages and disadvantages. Most of these measures are based on a large number of samples taken from the search space, which are evaluated after every change. To improve the estimates, we choose the samples by stratified sampling. Some measures also require the knowledge of the location of the global optimum, and are thus only applicable if this can be determined.

If the measure involves a distance between solutions, this depends on the underlying representation and search operators. In the experiments in Section 5, we look at real-valued and binary encodings, and use Euclidean distance and Hamming distance, respectively. Where local hill-climbing (LHC) is required, we use steepest descent hill climbing with bit flip neighborhood for binary representation and fixed step size of 1/5 the range of the search space in one variable in the real-valued domain.

Change severity: The distance between the optimum before the change and the optimum after the change has been deemed important in virtually all previous categorizations (cf. Section 2) and is usually termed "change severity". We agree that this is an important measure and list it here for completeness. Furthermore, for reasons of comparison, we normalize the change severity by dividing the distance by the maximal distance between any two points of the search space. Note that it is possible that there is more than one optimal solution, in which case we simply pick one for each environment to calculate the change severity[1]. If the change severity is low, it means that it is likely that the new optimum is somewhere close to the old optimum, and approaches like hypermutation [9], which keep the population and just add some diversity to revive exploration seem promising. On the other hand, if the change severity is large, such approaches are bound to fail.

Estimated change severity: For most real-world problems, it will not be possible to determine the optimum, and thus the change severity can not be determined. Therefore, it would be nice to have an estimator for the change severity that works without knowing the optimum. Here, we test the idea to estimate the change severity by looking at the distance between the best sample before and after the change.

Fitness correlation: This measure looks at the correlation coefficient of the fitnesses of all samples before and after a change. If the correlation is high, previous good solutions will remain to be good also in the new environment, and strategies with a memory of good solutions may be helpful. We calculate correlation coefficients over fitness values and ranks.

LHC fitness correlation: The samples we use are basically random points in the search space. However, usually we assume that the least our algorithm can do quickly is local hill-climbing. Therefore we propose here to measure the correlation of fitness after LHC before and after the change. That is, we look at local optima only (after LHC), and ask whether a similar fitness level can be reached again after a change by LHC. If the correlation is high, this indicates that simple local hill-climbing after a change may be sufficient to obtain new good quality solutions.

Fitness change correlation of similar points: Here we would like to know whether similar (in terms of distance in search space) points experience a similar fitness change. If the correlation is high, it means that whole areas in the search space change coherently, i.e., the structure of the landscape doesn't change too much. We measure the correlation with distance d in the following way: for each of n random points, we pick a random point with distance d, and observe the correlation of fitness *changes* between these n pairs.

Estimated value of last-stage local optima: Instead of starting from scratch after a change of the environment, many approaches attempt to re-use information from the environment's previous stage, e.g., by keeping the old population. With this measure, we want to determine the benefit of keeping information about good solutions from the previous stage. To this end, we would like to compare the fitness that can be obtained by LHC from one of the previous stage's best k local optima to the fitness that is obtained when starting from a random sample. Unfortunately, this measure requires to know the best k local optima, which is generally impossible to compute for any practical problem. Therefore, we report in this paper only on the estimated value of last-stage local optima. In this case, we compare the fitness obtained by performing LHC from one of the previous stage's best samples to the fitness obtained by LHC from a random sample.

Value of past optima: Instead of only using information from the environment's last stage, one may also look back for several stages. Consequently, several EA approaches to dynamic optimization problems introduce a memory of optima found in previous stages of the environment, hoping that these might help to discover the new optimum after a change. This is of course particularly promising in cyclic environments, where the optimum returns to

[1]We use a multi-dimensional knapsack problem with real-valued weights and profits in Section 5, for which the existence of several global optima is rather unlikely.

previous locations, or close to previous locations. For the proposed measure, we keep the optima of the m previous stages and check how much the solution found by LHC from a previous optimum improves compared to starting LHC from a random solution. Again, if there are more than one global optimum, we store a random one in the memory, assuming that an EA would also focus on a single optimum.

Estimated value of past optima: If the true optimum of a problem is unknown, as in most practical applications, we would like to have a measure that also works without knowing the optimum. Here, we replace the optimum in our previous measure by the point obtained from LHC on the best sample.

4. THE DYNAMIC MULTI-DIMENSIONAL KNAPSACK PROBLEM

Knapsack problems [16] are commonly used combinatorial benchmark problems to test the performance of EAs. There are many different variations of knapsack problems. The multi-dimensional knapsack problem (MKP) is one of these variations and belongs to the class of NP-complete combinatorial optimization. The MKP has a wide range of real world applications such as cargo loading, selecting projects to fund, budget management, cutting stock, etc. This aspect of the problem makes it a good choice to use as the example problem in this study. The MKP can be formalized as follows.

$$maximize \sum_{j=1}^{n} p_j \cdot x_j \tag{1}$$

subject to

$$\sum_{j=1}^{n} r_{ij} \cdot x_j \leq c_i, \quad i = 1,2,...,m \tag{2}$$

where n is the number of items, m is the number of resources, $x_j \in \{0,1\}$ shows whether item j is included in the subset or not, p_j shows the profit of item j, r_{ij} shows the resource consumption of item j for resource i and c_i is the capacity constraint of resource i.

For the MKP, several different genetic representations and genetic operators have been proposed. A detailed comparison can be found in [13]. Because we want to also examine the effect of the representation on the fitness landscape, in this study, we look at not only one but two successful approaches: a direct binary encoding using penalties for constraint handling, and a decoder-based real-valued encoding. These two are discussed in the following in more detail.

4.1 Binary representation

This is a direct encoding where each bit of a bit string corresponds to an item, and the bits indicate whether an item should be included in the knapsack or not [13, 14]. Because this does not guarantee feasibility, infeasible solutions are penalized

$$fitness(x) = f(x) - penalty(x) \tag{3}$$

In [13, 14], different penalty functions are compared, and the following one is recommended:

$$penalty(x) = \frac{p_{max}+1}{r_{min}} * max\{CV(x,i) \mid i \in I\}, \quad x \in U \tag{4}$$

$$p_{max} = max\{p_i \mid i \in I\} \tag{5}$$

$$r_{min} = min\{r_{ij} \mid i \in I, j \in J\} \tag{6}$$

$$CV(x,i) = max(0, \sum_{j \in J} r_{ij} \cdot x_j - c_i) \tag{7}$$

where p_{max} is the largest profit value calculated as in Eq. 5, r_{min} is the minimum resource consumption calculated as in Eq. 6 and $CV(x,i)$ is the maximum constraint violation for the ith constraint c_i calculated as in Eq. 7. It should be noted that $r_{min} \neq 0$ must be ensured.

4.2 Weight Coding

A successful example for the decoder based techniques is the weight-coding (WC) approach [22]. In the weight-coding technique, a candidate solution consists of a vector of real-valued genes (biases) associated with each item of the MKP. To obtain the corresponding phenotype, first the original problem P is transformed (biased) into a modified problem P' by multiplying the original profits of each item with the corresponding bias. Then, a fast heuristic is used to find a solution to P', and finally, the resulting solution (items to be placed in knapsack) is evaluated based on the original problem. Raidl [22] discusses two possible decoding heuristics, one based on a surrogate relaxation technique and the other based on Lagrangian relaxation. The one using the surrogate relaxation method is preferred due to its lower computational requirements. The surrogate relaxation method [21] simplifies the original problem by transforming all constraints into a single one as follows:

$$\sum_{j=1}^{n} \left(\sum_{i=1}^{m} a_i \cdot r_{ij} \right) x_j \leq \sum_{i=1}^{m} c_i \tag{8}$$

where a_i is the surrogate multiplier for the ith constraint, r_{ij} is the resource coefficient. Surrogate multipliers are determined by solving the relaxed MKP (i.e., variables x_i can take any value $\in [0,1]$) by linear programming (LP), and using the values of the dual variables as surrogate multipliers. Then, to obtain a heuristic solution to the MKP, the *profit/pseudo-resource consumption ratios* denoted as u_j are calculated as given in Eq. 9.

$$u_j = \frac{p_j}{\sum_{i=1}^{m} a_i r_{ij}} \tag{9}$$

The items are then sorted in decreasing order based on their u_j values. Using this ordering, items are added to the solution one at a time if none of the constraints are violated. More details regarding this method may be found in [22].

To keep computation costs low, in [22] the surrogate multiplier values a_i are determined only once for the original problem at the beginning. As a result, the decoding step starts with the computation of the u_j values based on the biased profits.

Biases are restricted to $[(1+\gamma)^{-1}, (1+\gamma)^{1}]$, where γ is called the biasing strength and determines the size of the covered search space. In the following experiments, we set $\gamma = 1$.

Weight-codings have the advantage of introducing heuristic bias and restricting the search to the feasible part of the search space.

4.3 The Dynamic MKP

Dynamic instances of knapsack problems have been proposed before. However, they usually consider only one dimension, and the only dynamic An instance of a MKP depends on the values of the profits p_j, resource consumptions r_{ij} and the resource constraints c_i. While dynamic instances of knapsack problems have been proposed before, they usually consider only one dimension, and the only dynamic aspect is a cyclic change of the resource constraint (see e.g. [12, 19]).

In the dynamic MKP we implemented for this study, for every change, the profits, resource consumptions and the constraints are

multiplied by a normally distributed random variable as follows:

$$p_j^+ = p_j * (1 + N(0, \sigma_p))$$
$$r_{ij}^+ = r_{ij} * (1 + N(0, \sigma_r)) \quad (10)$$
$$c_i^+ = c_i * (1 + N(0, \sigma_c))$$

The standard deviation of the normally distributed random variable used for the changes has been set to $\sigma_p = \sigma_r = \sigma_c = 0.05$ so that changes are of moderate severity. Each profit p_j, resource consumption r_{ij} and constraint c_i is restricted to an interval as determined in Eq. 11.

$$\begin{array}{c} 0.8 * p_j \leq p_j \leq 1.2 * p_j \\ 0.8 * r_{ij} \leq r_{ij} \leq 1.2 * r_{ij} \\ 0.8 * c_i \leq c_i \leq 1.2 * c_i \end{array} \quad (11)$$

If any of the changes causes any of the lower or upper bounds to be exceeded, the value is bounced back from the bounds and set to a corresponding value within the allowed boundaries.

In the experiments below, the dynamic multi-dimensional knapsack problems are based mostly on the problem PB5 that can be obtained from [3]. Some results are also based on the instances HP1 and WEISH02, which can also be obtained at [3].

5. ANALYSIS OF LANDSCAPE CHANGES

In this section, we calculate the measures proposed above for several instances of the dynamic MKP. Unless stated otherwise, we use 32768 samples to calculate the measures for the binary encoding, and 5000 samples to calculate the measures for WC[2]. Optimal solutions were calculated by running a MIP solver from the glpk package [11].

5.1 Change severity

Figure 1 (a) shows the actual and estimated change severity over 20 environmental changes for the binary representation. The actual normalized change severity lies between 0.1 and 0.4 with an average of 0.235, i.e., a jump from the old to the new optimum requires to flip an average of 23.5% of the bits. That is less than 50% which would have been required if the new landscape were completely unrelated to the old one, but probably still more than what simple strategies like hill-climbing or hypermutation can handle. The estimated change severity depends on the samples selected, which is why in Figure 1 (a) mean over 20 runs plus/minus standard deviation is plotted. For the 20 runs, the seed of the environment change generation is kept the same which means all runs are performed on the same set of environment changes with different samples from the search space. On average over the 20 runs, the measure predicts a normalized change severity of 0.279, which is reasonably close to the true value of 0.235. However, the standard deviation is rather large, and the correlation between estimated and true change severity is quite low, which means that the estimator is not too helpful in determining the change severity of a single change. In the example from Figure 1 (a), we took 32768 samples, which amounts to approximately 3.1% of the search space. Obviously, the accuracy of the measure could be improved by increasing the number of samples (with perfect match when the complete search space is sampled).

Figure 1 (b) shows the same plot for the WC encoding. Since it is not obvious how to transfer an optimal solution back into weights, we can't plot the true severity for this encoding. With an average normalized change severity of 0.345, the estimated change severity is significantly higher, but still below random movements. Also, the estimates have a significantly lower standard deviation.

[2]Since WC only covers the feasible search space, much smaller sample numbers are necessary to get a representative set.

5.2 Fitness correlation

Table 1 shows the fitness correlation of the samples before and after a change. As can be seen, for the binary encoding correlation is extremely high (0.98-0.99). Because we wondered whether that is only due to the strong penalty for infeasible solutions and thus a categorization into feasible and infeasible solutions, we additionally tested the correlation of fitness ranks in the population. However, that turned out to be almost as high (0.94-0.99). Therefore, in this environment, good solutions are likely to be found where good solutions have been in the previous stage. Surprisingly, when looking at fitnesses after LHC, correlation is significantly lower. So far, we have no intuitive explanation for this.

For the WC encoding, correlation is much lower (0.53-0.6), but not affected by LHC. The latter is probably due to the many plateaus existing in this fitness landscape.

Figure 2 shows how the fitness correlation depends on the time lag. As is to be expected, the correlation decreases with increasing time lag. Decrease is stronger for WC than for binary encoding. Additional tests (not shown) have demonstrated that fitness correlation for the WC encoding increases with the biasing strength.

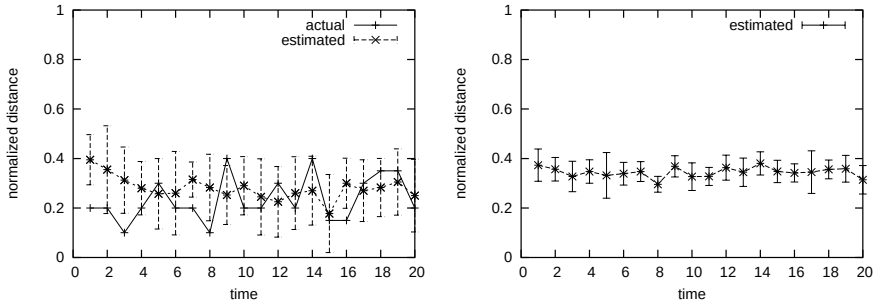
5.3 Fitness change correlation of similar points

The fitness change correlation measures how strongly correlated the fitness *change* of an individual is with a random individual with normalized distance d. The results for binary encoding and WC encoding are shown in Figure 3. For binary encoding, correlation of close neighbors is rather high with a correlation coefficient of 0.75 for $d = 0.05$. For weight-biased encoding, correlation is much less pronounced. As is to be expected, in both encodings, correlation decreases with distance. For normalized distances greater than 0.5, it even turns negative. Such information could perhaps be exploited algorithmically, but so far, this has not been attempted.

5.4 Value of last-stage local optima

Running LHC from a random solution yields an average fitness of approximately 1642 for the binary encoding and 1789 for the WC encoding. [3] This difference is clearly due to the heuristic bias that is part of the WC encoding. The average optimal value (i.e., the upper bound) is 2047.36. Running LHC from one of the k best samples yields significantly better solutions than starting from random solutions. Figure 4 shows the solution quality that can be obtained by doing LHC starting from the k-th best sample from the previous stage, for binary encoding (a) and for WC encoding (b). As would be expected, the less good the solution in the previous stage (i.e., the larger k), the smaller the solution quality usually obtained after the change. But even starting from the 50-th sample from the previous stage yields a much better solution than starting from a random sample. As for starting from random solutions, WC yields better solutions also when starting from last stage's good solutions. Also, degradation over k is much slower when the WC coding is used, which means that the quality of the previously best found solution is not as important with WC. Nevertheless, the general trends are also observable for WC.

[3]Note that because we only simulated 20 environmental changes, and we want to use information from up to the previous 10 stages in the next section, the values in this and the next section are averaged only over the environments 10-20.

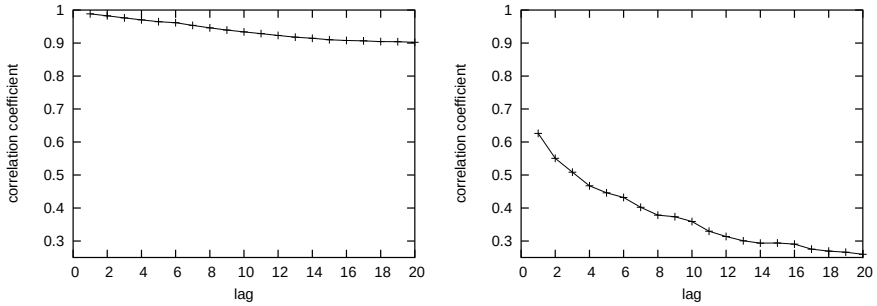


(a) Binary encoding (b) WC encoding

Figure 1: Change severity over 20 consecutive changes.

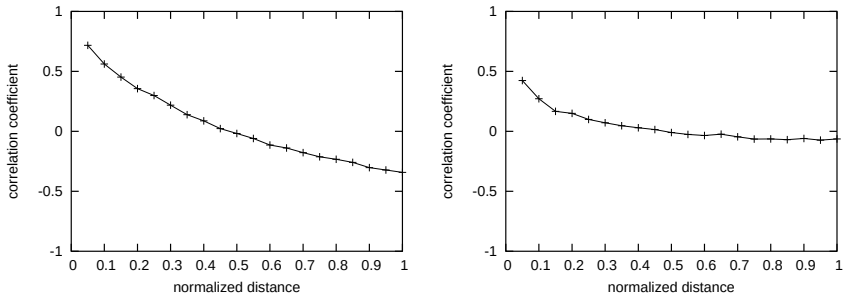
Table 1: Fitness correlations and rank correlations for different test instances, with and without LHC, and binary as well as WC encoding. Average and standard deviation are given.

		binary			WC		
		PB5	HP1	WEISH02	PB5	HP1	WEISH02
Fitness	avg	0.99	0.98	0.99	0.60	0.53	0.56
	std	0.009	0.013	0.008	0.092	0.128	0.159
Fitness	avg	0.76	0.77	0.90	0.64	0.51	0.53
after LHC	std	0.058	0.101	0.027	0.084	0.198	0.187
Rank	avg	0.98	0.94	0.99			
	std	0.010	0.0281	0.009			
Rank	avg	0.75	0.76	0.90			
after LHC	std	0.061	0.100	0.029			



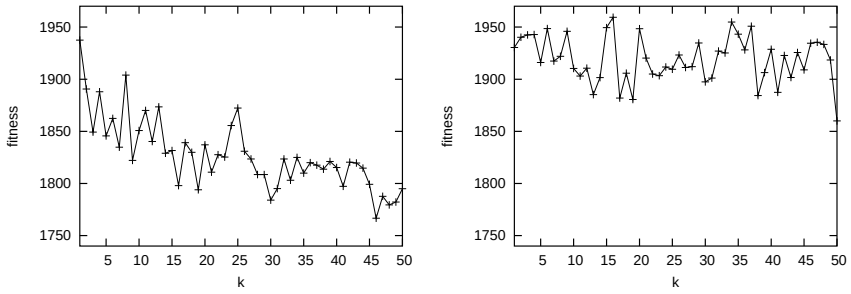
(a) Binary encoding (b) WC encoding

Figure 2: Fitness correlation depending on time lag, for problem PB5.



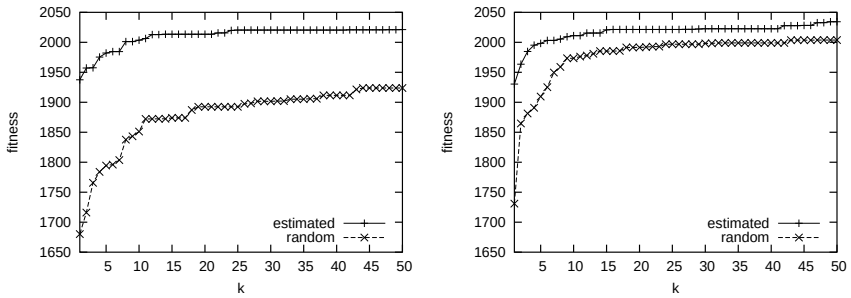
(a) Binary encoding (b) WC encoding

Figure 3: Fitness change correlation between random individuals of normalized distance d.



(a) Binary encoding (b) WC encoding

Figure 4: Estimated value of good solutions from previous stage.



(a) Binary encoding (b) WC encoding

Figure 5: Combined value of the k best solutions from previous stage.

The benefit of using several (in our case the k best) previous samples is shown in Figure 5. As is to be expected, the quality of the best found solution increases with the number of random starting points for LHC. For previously good samples, the effects could be too strongly correlated (e.g., because they all lie in the same region of the search space), so that adding additional starting points might not be beneficial. And indeed, the benefit of increasing the number of previously good samples is smaller than the benefit of increasing the number of random points. In fact, for WC, starting from more than 15 previous solutions has almost no effect. However, starting from previously good samples is always better than starting from random samples. Also, note how closely the quality approaches the optimum of 2047.36 in case of WC. Therefore, we conclude that in a dynamic environment, it is beneficial to maintain a diverse set of high-quality solutions from the previous stage.

5.5 Value of past optima

Instead of just keeping information from the last stage, it may be helpful to use a memory, storing the best solutions found in the m previous stages. Again, the results need to be compared to what can be obtained by starting from random solutions, namely the already quoted fitness of approximately 1642 for the binary encoding and 1789 for the WC encoding. As for the last measure, let us first look at the value of single solutions. Clearly, Figure 6 shows that starting LHC from a local optimum of previous stages again significantly improves performance. As is to be expected, there is a general tendency that with increasing age of the previous optimum, its benefit shrinks. However, the degradation is surprisingly slow. Despite the fact that the environment was not designed to be cyclic, starting LHC from an optimum 10 environmental stages ago performs much better than starting from a random point. For the binary encoding, we have additionally plotted the value of starting from the previous environment's true optima (as opposed to the best solution found). As expected, the value of the true optima is generally higher than the value of the perceived optima, but the estimated value seems to be a reasonable approximation.

Now let us again turn at the accumulated value of several previous optima. As can be seen in Figure 7, additional optima add value, although the additional benefit becomes rather small if the additional optima are from more than 5 environmental stages ago, in particular for the WC encoding.

Comparing the value of previous optima with the value of last-stage good solutions, both seem to have comparable benefit (cf. results for $k < 10$ with $m < 10$). Only for larger numbers, one might detect a slight advantage for using samples from only the previous stage, which may be due to the fact that the information from many stages ago is too outdated to be helpful

6. CONCLUSION

In this paper, we have proposed a number of measures with the aim to characterize fitness landscape changes of a dynamic optimization problem. These measures include change severity, fitness correlations, and the value of previous good solutions. Furthermore, we have applied these measures to several instances of a multi-dimensional knapsack problem, and two different representations. We have shown that the proposed measures allow interesting observations.

We are aware that these measures are only a first step towards the analysis of dynamic landscapes. In the future, we will use these measures to compare and possibly classify different real-world problems. Also, we will use them to examine artificial benchmark problems and check whether benchmark problems have similar characteristics to real-world problems. Finally, it would be most valuable if based on the problem characteristics the suitability of different algorithmic approaches could be predicted.

7. REFERENCES

[1] P. J. Angeline, D. B. Fogel, and L. J. Fogel. A comparison of self-adaptation methods for finite state machines in dynamic environments. In L. J. Fogel et al., editor, *Conference on Evolutionary Programming*, pages 441–449, 1996.

[2] D. V. Arnold and H.-G. Beyer. Random Dynamics Optimum Tracking with Evolution Strategies. In J.J. Merelo, P. Adamidis, H.-G. Beyer, J.L. Fernández-Villacañas, and H.-P. Schwefel, editors, *Parallel Problem Solving from Nature*, pages 3–12, Heidelberg, 2002. Springer.

[3] J. E. Beasley. Or-library. online, www.brunel.ac.uk/depts/ma/research/jeb/orlib/mknapinfo.html.

[4] C. Bierwirth and D. C. Mattfeld. Production scheduling and rescheduling with genetic algorithms. *Evolutionary Computation*, 7(1):1–18, 1999.

[5] J. Branke. Evolutionary algorithms for dynamic optimization problems - a survey. Technical Report 387, Insitute AIFB, University of Karlsruhe, February 1999.

[6] J. Branke. Memory enhanced evolutionary algorithms for changing optimization problems. In *Congress on Evolutionary Computation*, volume 3, pages 1875–1882. IEEE, 1999.

[7] J. Branke. *Evolutionary Optimization in Dynamic Environments*. Kluwer, 2001.

[8] J. Branke and H. Schmeck. Designing evolutionary algorithms for dynamic optimization problems. *Theory and application of evolutionary computation: recent trends*, pages 239–262, 2002. S. Tsutsui and A. Ghosh, editors.

[9] H. G. Cobb. An investigation into the use of hypermutation as an adaptive operator in genetic algorithms having continuous, time-dependent nonstationary environments. Technical Report AIC-90-001, Naval Research Laboratory, Washington, USA, 1990.

[10] Y. Davidor. Epistasis variance: suitability of a representation to genetic algorithms. *Complex Systems*, 4:369–383, 1990.

[11] GLPK. GNU linear programming kit. online, http://www.gnu.org/software/glpk/glpk.html.

[12] D. E. Goldberg and R. E. Smith. Nonstationary function optimization using genetic algorithms with dominance and diploidy. In J. J. Grefenstette, editor, *International Conference on Genetic Algorithms*, pages 59–68. Lawrence Erlbaum Associates, 1987.

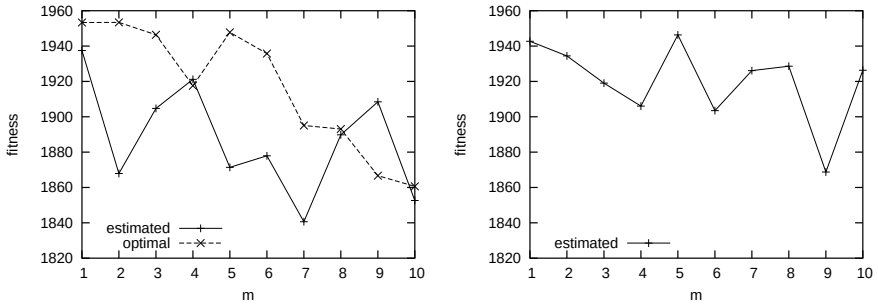
[13] J. Gottlieb. *Evolutionary Algorithms for Combinatorial Optimization Problems*. Phd, Mathematisch-Naturwissenschaftlichen Fakultaet der Technischen Universitaet Clausthal, December 1999.

[14] J. Gottlieb. On the feasibility problem of penalty-based evolutionary algorithms for knapsack problems. In E.J.W. Boers et al., editor, *Applications of Evolutionary Computing*, volume 2037 of *LNCS*, pages 50–60. Springer, 2001.

[15] K. A. De Jong. Evolving in a changing world. In *International Symposium on Foundations of Intelligent Systems*, pages 512–519. Springer, 1999.

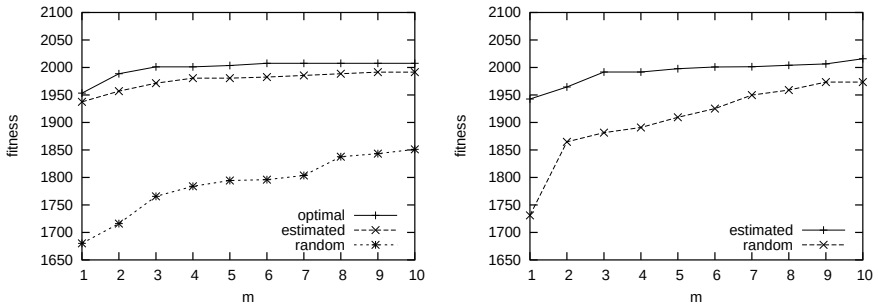
[16] H. Kellerer, U. Pferschy, and D. Pisinger. *Knapsack Problems*. Springer, 2004.

[17] Y. Li and C. O. Wilke. Digital evolution in time-dependent fitness landscapes. *Artificial Live*, 10:123–134, 2004.



(a) Binary encoding (b) WC encoding

Figure 6: Value of best solution from m-th previous stage.



(a) Binary encoding (b) WC encoding

Figure 7: Combined value of best solutions from m previous stages.

[18] P. Merz. Advanced fitness landscape analysis and the performance of memetic algorithms. *Evolutionary Computation*, 12(3):303 – 325, 2004.

[19] N. Mori, H. Kita, and Y. Nishikawa. Adaptation to a changing environment by means of the thermodynamical genetic algorithm. In *Parallel Problem Solving from Nature*, pages 513–522. Springer, 1996.

[20] R. W. Morrison and K. A. DeJong. A test problem generator for non-stationary environments. In *Congress on Evolutionary Computation*, volume 3, pages 2047–2053. IEEE, 1999.

[21] H. Pirkul. A heuristic solution procedure for the multiconstraint zero-one knapsack problem. *Naval Research Logistics*, 34:161–172, 1987.

[22] G. R. Raidl. Weight-codings in a genetic algorithm for the multiconstraint knapsack problem. In *Congress on Evolutionary Computation*, pages 596–603. IEEE, 1999.

[23] C. Reidys and P. Stadler. Combinatorial landscapes. *SIAM Review*, 44:3–54, 2002.

[24] H. Richter. Behavior of evolutionary algorithms in chaotically changing fitness landscapes. In Xin Yao et. al., editor, *Proceedings of Parallel Problem Solving from Nature VIII*, volume 3242 of *LNCS*, pages 111–121. Springer, 2004.

[25] C. Ronnewinkel, C.O. Wilke, and T. Martinetz. Genetic algorithms in time-dependent environments. In *Theoretical Aspects of Evolutionary Computing*. Springer, 2001.

[26] J. E. Rowe. Finding attractors for periodic fitness functions. In W. Banzhaf et al., editor, *Genetic and Evolutionary Computation Conference*, pages 557–563. Morgan Kaufmann, 1999.

[27] J. E. Rowe. Cyclic attractors and quasispecies adaptability. In L. Kallel, B. Naudts, and A. Rogers, editors, *Theoretical Aspects of Evolutionary Computing*, pages 251–259. Springer, 2001.

[28] P. F. Stadler. Towards a theory of landscapes. In R. Lopez-Pena et al., editor, *Complex Systems and Binary Networks*, pages 77–163. Springer, 1995.

[29] S. A. Stanhope and J. M. Daida. Genetic algorithm fitness dynamics in a changing environment. In *Congress on Evolutionary Computation*, volume 3, pages 1851–1858. IEEE, 1999.

[30] K. Trojanowski and Z. Michalewicz. Searching for optima in non-stationary environments. In *Congress on Evolutionary Computation*, volume 3, pages 1843–1850. IEEE, 1999.

[31] R. K. Ursem, T. Krink, M. T. Jensen, and Z. Michalewicz. Analysis and modeling of control tasks in dynamic systems. *IEEE Trans. Evolutionary Computation*, 6(4):378–389, 2002.

[32] K. Weicker. *Evolutionary Algorithms and Dynamic Optimization Problems*. Der Andere Verlag, 2003.

[33] C. O. Wilke. *Evolutionary Dynamics in Time-Dependent Environments*. Shaker Verlag, 1999.