Crossover Operators for Multiobjective k-Subset Selection

Thorsten Meinl
Thorsten.Meinl@uni-konstanz.de

Michael R. Berthold
Michael.Berthold@uni-konstanz.de

Nycomed Chair for Bioinformatics and Information Mining
University of Konstanz
Box 712
78457 Konstanz, Germany

ABSTRACT

Genetic algorithms are often applied to combinatorial optimization problems, the most popular one probably being the traveling salesperson problem. In contrast to permutations used for TSP, the selection of a subset from a larger set has so far gained surprisingly little interest. One intriguing example of this type of problems occurs in diversity selection for virtual high throughput screening, where k molecules need to be selected from a set of n while optimizing certain constraints. In this paper we present a novel representation for k-subsets and several genetic operators for it.

Categories and Subject Descriptors: I.2.8 [Artificial Intelligence]: Problem Solving, Control Methods, and Search — Heuristic methods

General Terms: Algorithms

Keywords: Crossover, Genetic Algorithm, Multiobjective Optimization, Combination, Subset, Diversity Selection

1. INTRODUCTION

Selecting k items out of a set of n is a common problem. In this article we concentrate on an application in the Life Sciences, more precisely in the early drug discovery process. Before pharma companies start automated tests with several hundreds of thousands of molecules in so-called *HTS* (High Throughput Screening), they usually plan to buy or synthesize a few thousand new molecules specifically for the current target. However, this set of compounds must satisfy two main criteria:

1. They should be as active as possible on the target.
2. The set of molecules should be as diverse as possible.

These two objectives usually contradict each other, because very similar molecules often show the same or comparable activities. Therefore this leads to a classical multiobjective optimization problem: Select k molecules out of a set of n at the same time maximizing the activity and minimzing the pairwise similarities (or maximizing diversity).

Given the fact that investigating all possible k-subsets out of n molecules quickly becomes computationally very expensive (the binomial coefficient $\binom{n}{k} = \frac{n!}{k!(n-k)!}$ computes the number of possibilities), it is infeasible to check them all except for very small values of k and/or n.

GECCO'09, July 8–12, 2009, Montréal Québec, Canada.
ACM 978-1-60558-325-9/09/07.

Clearly, genetic algorithms are a good method for heuristic combinatorial optimization problems such as the one discussed here. Interestingly, the community has done a lot of research in finding suitable genetic operators and representations for *permutations*, but to our knowledge there exists only one single publication about operators for combinations [1], yet. In this paper we introduce two novel crossover operators for combinations of fixed size (k-subsets) together with a common mutation operator and apply all three on the problem of diversity selection from a set of molecules.

2. CROSSOVER OPERATORS FOR COMBINATIONS

Before crossover and mutation operators are designed, a chromosome encoding needs to be defined. Typically, for genetic algorithms bit strings are used but there also exist integer or even real number encodings. In [1] bit strings are used, but we decided for an integer valued encoding, which we motivate below.

2.1 Chromosomes with two-point crossover

A combination can easily be encoded in a k-length integer array that holds the selected items' numbers, e.g. `[2,4,5]`. This representation allows for an easier access to the selected elements than a bit string because in a bit string, the set bits need to be searched for. Additionally, combinations *with repetitions* can easily be evolved with only minor modifications to the operators. The fixed-sized array also ensures that a chromosome always consists of k selected elements, but special care has to be taken that no element appears twice.

The mutation operator works by selecting a random element j in the integer array and a random number $r < n$ – repeating this while r has already been selected before – and replacing j with r.

The crossover operator works similar to normal two-point crossover. First, two random points are selected and for the two offspring the numbers in between the two points are taken. The remaining entries are filled up with elements from the other individual *but only if a number has not been selected for the individual yet.* This can be checked by maintaining a bit set of selected elements. Entries still missing after this second step are consecutively filled up from the other individual – they are even taken from the middle interval.

2.2 Chromosomes with uniform crossover

For some problems, such as our problem of diversity selec-

tion, uniform crossover works better or at least offers faster convergence. Therefore we also implemented an adaption of uniform crossover for combinations in the integer array representation. For reasons of efficiency we decided to work with sorted arrays this time, as this makes it very easy to detect duplicate entries. Of course, the sorted order needs to be maintained throughout all operations. Therefore, the mutation operator is modified slightly. Instead of scanning the whole array for the new number r, this can more easily be done in logarithmic time with binary search. After no duplicate has been found, r needs to be inserted at the right position (which is already known from the binary search) in order to keep the array sorted. The elements between the removed number and the insertion position of r are shifted accordingly and r is inserted at the free place.

Implementing pseudo-uniform crossover with a sorted array is now straight-forward. In principle every second element is exchanged between the two parent individuals, but again, this may introduce duplicate entries. However, using the two sorted arrays of both parent individuals makes it easy to create a sorted combined array of length $2 * k$ containing all numbers from both individuals. Note, that a number can appear at most twice in this new array and if that is the case, both occurrences are next to each other. The two offspring are now created by taking all odd elements for the first child and all even elements for the second.

3. EXPERIMENTS

In order to evaluate the three operators' behaviour, we used them for finding Pareto-optimal solutions for the problem described in section 1. In addition to a real-world dataset we also used synthetic data to check that the results are not overly specific to the chosen dataset. The underlying MO genetic algorithm we used was NSGA-II[2].

The first dataset we used is publicly available from BindingDB.org (`http://www.bindingdb.org`) consisting of 1,376 molecules that have been tested for their activity against the CDK-2 protein. The dataset contains the molecules' activities as IC_{50} values and their 2D structure, from which we computed their pairwise similarities. In our experiments we set the subset size to 137 (10% of the database), resulting in a search space of about 1.8×10^{193} possible solutions. Initially the population size was set to 300 and mutation rate to 1%. Each experiment was carried out 10 times.

Figure 1 shows in the top part the hypervolume [3] for the three different operators, averaged over 10 runs. It is clearly visible that uniform crossover converges much faster than the other two operators. However, it is outperformed in the end, but only to a small amount which is mainly due to "border" solutions in the corners of the search space (which in our application are of less interest than solutions in the center part of the front).

We also performed several other experiments with more individuals, higher mutation rates and different values of k but the results were comparable to the presented ones, therefore we omit their presentation here.

In order to verify that the operators' behaviour is not an artifact of the chosen dataset, we also generated synthetic datasets consisting of datapoints randomly distributed between 0 and 1 in the plane. Then three "activity spots" were assigned which get an activity value of 1. All other points get an activity that exponentially decreases with increasing distance to the two chosen points.

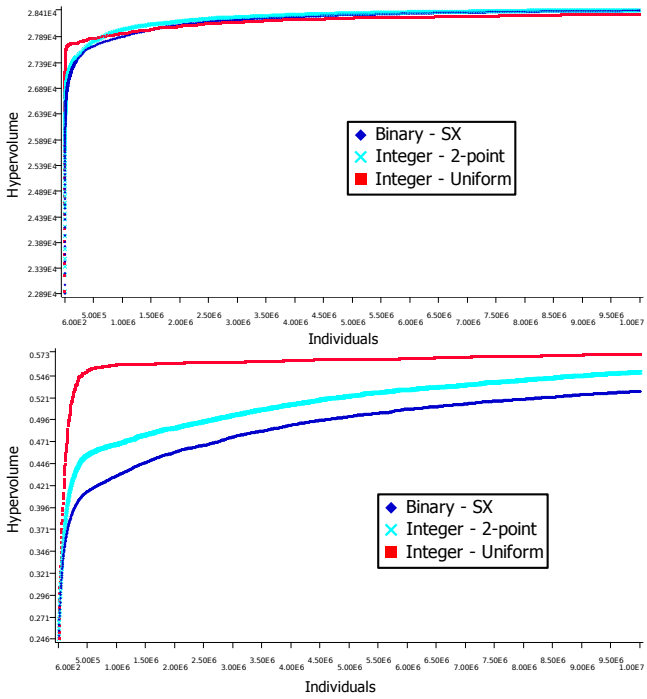

Figure 1: The hypervolume indicators for up to $10,000,000$ individuals for the molecule (top) and synthetic dataset (bottom). Uniform crossover converges much faster but is outperformed by two-point crossover in late generations on the molecular dataset.

The experiments were carried out in the same way: k was set to 10% of the whole dataset, population size was 300, mutation rate 1%, and each test was run 10 times. The lower part of Figure 1 shows the average value of the hypervolume indicator for up to $1,000,000$ generated individuals for a dataset with $10,000$ random points. The results for other datasets are comparable, therefore we omit their exact presentation.

On the synthetic dataset uniform crossover performs even better than two-point crossover. Interestingly even the integer two-point crossover is superior to the binary SX operator.

4. REFERENCES

[1] J.-S. Chen and J.-L. Hou. A Combination Genetic Algorithm with Applications on Portfolio Optimization. In *Advances in Applied Artificial Intelligence*, volume 4031 of *Lecture Notes in Computer Science*, pages 197–206. Springer, Berlin, Germany, 2006.

[2] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, 6:182–197, 2002.

[3] E. Zitzler, J. Knowles, and L. Thiele. Quality Assessment of Pareto Set Approximations. In *Multiobjective Optimization - Interactive and Evolutionary Approaches*, volume 5252 of *Lecture Notes in Computer Science*, pages 373–404. Springer, Berlin, Germany, 2008.