The Impact of Problem Features on NSGA-II and MOEA/D Performance

By: Lars Folkersma (Student number: 5543800)
Supervisor: Dirk Thierens
Second Examiner: Silja Renooij

Thesis for the master Artificial Intelligence, Utrecht Unversity

June 25, 2020

Abstract

The multi-objective evolutionary algorithms NSGA-II and MOEA/D are often compared using test problems which combine a number of factors to create a difficult problem. Instead, we study the effect of a variety of factors directly by using problems in which the presence of these factors can be specified, with the purpose of studying the effect of each factor individually. The factors we study are the shape of the objective space and the uniformity of the distribution of solutions therein, the number of objectives and the correlation between them, and the separability of variables. We do some additional experiments to better understand some of our results, especially the phenomena we see when we vary the correlation between objectives at a high number of objectives. Lastly, we investigate the possibility of predicting algorithm performance based on attributes which were measured rather than specified.

NSGA-II is non-domination-based while MOEA/D is decomposition-based, which together with the fact that they are both frequently used makes for an especially interesting comparison that may contribute to a further understanding of the differences between their designs, as well as to the ability to choose the best algorithm for any given problem.

Chapter 1

Introduction

The **multi-objective evolutionary algorithms (MOEAs)** NSGA-II and MOEA/D are both frequently used algorithms within the domain of **evolutionary multi-objective optimization**. However, they differ significantly in their approach to solving multi-objective optimization problems. NSGA-II is non-domination-based, while MOEA/D is decomposition-based [1, p. 264].

The literature clearly shows that depending on the test problem, any variant of NSGA-II or MOEA/D that we are using in these experiments can be the best of these algorithms [3]. Within the literature there are various test problems on which NSGA-II and MOEA/D are compared. However, within these test problems a variety of factors contributing to problem difficulty often come together, usually in an attempt to resemble real-world problems. (For example: DTLZ [13], WFG [18]). Instead, this research takes a number of factors from the literature for which it has been suggested that they are significant factors in determining the relative performance between these algorithms, and for them are crafted a number of test problems in which the presence of these factors is tunable, to study their effect on the relative performance between NSGA-II and MOEA/D directly. The goal is to contribute to the ability to choose the right algorithm for the job, as well as to increase our understanding of the implication of the design differences between NSGA-II and MOEA/D.

Within the literature the **shape of the Pareto front** and the **separability of variables** are described as being significant for the performance of certain decomposition-based algorithms [3, p. 188]. Another paper shows that the **number of objectives** and the **correlation between objectives** is also an important factor [1]. On top of that we introduce another factor of study namely the **uniformity of the distribution of solutions**, which seems a logical choice as we will see. We use an existing test problem (ρMNK-landscapes) and craft others in which the presence of these factors is tunable, to study their effect on the relative performance between NSGA-II and MOEA/D directly.

With this research we hope to contribute to a big and active field of research: By January 2017, there were over 8900 publications on evolutionary multi-objective optimization[1], and it has already been applied to engineering problems, scheduling problems, economic and finance problems, automatic cell planning problems and traveling salesman problems[9, p. 33], showing both a wide applicability of this kind of algorithm, as well as a very significant amount of established research.

We will start with an extensive explanation of evolutionary multi-objective optimization, as well as a number of related factors such as Pareto dominance and hypervolume. We shall describe the workings of NSGA-II and MOEA/D in detail, including a very short review of related literature.

We will review some of the differences in performance between these algorithms on test problems as found in the literature, and how that leads us to choosing the factors for investigation that we in fact choose.

We perform a number of experiments in the style described earlier, based on ρMNK-problems (which will be described in an earlier chapter) and variations on disk-quadrant problems which are constructed analogously to the DTLZ problems (a test suite we will also review in an earlier chapter).

Having performed a number of experiments that measure the effect of factors which were deliberately put into the problem, with part of the motivation being the ability to find the best algorithm for the job, our research begs the question: Can we, given a problem, quickly analyze it in order to choose the best tool for the job, without using the information that was used to create the problem? Thus we measure the effect of a number of measured features of test problems on the relative performance of NSGA-II and MOEA/D.

1.1 Relevance of Evolutionary Algorithms in Artificial Intelligence

Arguably, evolutionary algorithms are themselves part of AI, being a form of soft computing, which refers to "computational techniques designed to deal with imprecision, uncertainty, approximation and partial truths"[28, p. 22]. Evolutionary algorithms also appear in the form of evolutionary programming, where an evolutionary approach is used to evolve a program or a neural network[28, p. 92].

The field of evolutionary algorithms also overlaps with various other forms of AI. For example, evolutionary algorithms can be used for reinforcement learning [29]. They are "considered to be one of the most successful search techniques for complex problems", and have been used for things such as knowledge extraction in a database and learning of controllers in robotics [30, p. 1-2].

[1]Counting 'journal papers' and 'conference papers' in the repository maintained by Dr. Carlos A. Coello Coello, which can be found at `http://delta.cs.cinvestav.mx/~ccoello/EMOO/`

Chapter 2

Evolutionary Multi-objective Optimization

2.1 Multi-objective Optimization Problems

To explain evolutionary multi-objective optimization, we should first explain multi-objective optimization in general. As the name suggests, we have some kind of problem in which we try to perform well at multiple objectives at the same time. The following is a more exact definition:

Definition 2.1.1. A **multi-objective optimization problem** consists of [1, p. 265](notation ours):

- A **decision space** or **solution space** Ω of admissible solutions
- An **objective space** $\mathbb{R}^M$ in which the quality of these solutions is expressed, where M is the number of objectives.
- **Objective functions** or **fitness functions** relating the decision space to the objective space, that is to say M functions $f_i : \Omega \to \mathbb{R}, i = 1, \ldots, M$. Sometimes the functions will be seen as components of one function $f : \Omega \to \mathbb{R}^M$, simply called the objective function.

When we want to perform well at multiple objectives at the same time, it becomes a problem in itself to define which solutions is better than another. For example, when optimizing on two objectives, solution A may perform better at the first objective, and solution B may perform better at the second objective. Were there some kind of function that could decide how to balance these objectives, we could of course just optimize that instead but that would mean we don't really have a multi-objective problem, so we shall assume we do not have such a function.

Therefore, rather than trying to find 'the solution' or even 'a solution' to the problem, we instead try to find an entire set of solutions. This begs the question which solutions we should include in the solution set. The answer is rather simple: We want a variety of solutions, all of which are potentially useful. That is to say, all solutions for which there is no other solution which is superior in every way. This leads us to the definition of concepts such as Pareto dominance.

Definition 2.1.2. Given some multi-objective fitness function f, whose components correspond to objectives, a solution x **Pareto dominates** a solution y if

$$\forall i \in \{1, \ldots, M\} : f_i(x) \geq f_i(y)$$

and

$$\exists i : f_i(x) > f_i(y)$$

[1, p. 265] (This formulation is for a maximization problem; it can be used the other way around in a minimization problem, where it is instead about having a smaller value rather than a greater value.)

Definition 2.1.3. A solution is **Pareto optimal** if it is not Pareto dominated by any other solution.[1, p. 265]

Definition 2.1.4. The set of all Pareto optimal solutions is referred to as the **Pareto set**, and the image (under the fitness function) of this set in the objective space is called the **Pareto front**[1, p. 265].

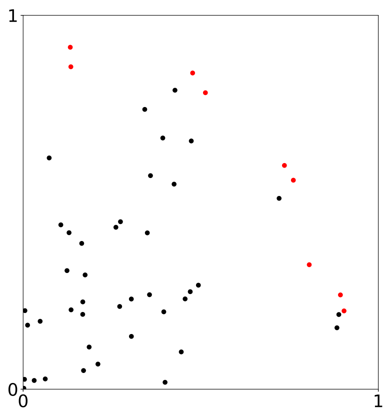


Figure 2.1: Example set of solutions to a 2-dimensional multi-objective optimization problem. Pareto optimal solutions in red (together they form the Pareto front), other solutions in black.

Every multi-objective optimization problem, as previously defined, of course has a Pareto front. In general, and in this document in particular, the goal of a multi-objective optimization algorithm is to find, or at least approach this Pareto front.

2.1.1 Examples of multi-objective optimization problems

There is a large variety of multi-objective optimization problems, both specific problems and entire families of problems. We will present a few problems here which are used for benchmarking in the literature, as well as some problems which we will use later on.

DTLZ

The DTLZ (named after its authors, Deb, Thiele, Laumanns and Zitzler) test problem suite was developed with the following intended features[13, p. 4]:

- Controllable hindrance in converging to the Pareto front as well as in diversification
- Scalable number of decision variables
- Any number of objectives
- Simple to construct
- Easily comprehensible Pareto-front
- Exhibiting difficulties similar to those in most real-world problems

The decision variables in DTLZ problems are real numbers between 0 and 1, the number of which can be varied (though there are some constraints). The problems are all rather specific constructions with varying attributes. For example, DTLZ1 has a simple Pareto front described by the linear hyper-plane

$$\sum_{i=1}^{M} f_i = 0.5$$

(which in the 3-dimensional variant is the triangle connecting $(0.5; 0; 0)$, $(0; 0.5; 0)$ and $(0; 0; 0.5)$, for example) . DTLZ2 has as its Pareto-front the first octant of the unit sphere. DTLZ3 is similar, but modified to deliberately introduce extra local optima, and DLTZ4 is also similar but with a very dense distribution of optima around the 0-plane for any objective. Other variants test the ability to converge to a curve (and a harder variant of this), to converge to a disconnected set of Pareto-optimal regions, to converge to a Pareto front which is a combination of a straight line and a hyper-plane, and lastly there is the problem DLTZ9 where the density of solutions gets thinner towards the Pareto-optimal region[13, p. 19-28].

For some of the problems the Pareto front has a rather simple form, such as the hyperplane or octant of the unit sphere as we have just seen. However, some of these problems (like DTLZ7-9) cannot be represented in such a simple form[3, p .172].

WFG

The WFG (Walking Fish Group) problems are based on a broader toolkit for designing test problems. Using this toolkit a number of specific test problems were proposed, WFG1-WFG9. These problems supposedly (according to their own authors) offer a more comprehensive set of challenges than for example DTLZ[17, p. 497]. Some of these have a Pareto front that is easy to describe, such as WFG4-9 where the Pareto-front is given by

$$\sum_{i=1}^{M} f_i = 1 \land \forall i : f_i \geq 0$$

but just as with DTLZ not all Pareto-fronts can be easily represented[3, p .172].

The WFG problems vary over a number of possible attributes, such as[17, p. 494,495,498]:

- Having a convex or concave Pareto-front
- Having a connected or disconnected Pareto front
- Having a degenerate front or not (where the dimensionality of the Pareto front is smaller than $M-1$ where M is the number of objectives)
- Having uni-modal and multi-modal test problems (uni-modal is where an objective function has only a single optimum)

Also, the test problems are designed to have mostly inseparable decision variables [17, p. 485]; that is to say that the optima of any decision variable viewed individually depend on the value of other decision variables; decision variables can't be optimized independently of the other variables[17, p. 479].

Knapsack

A knapsack problem is a problem where there is some set of items which have profits and weights attributed to them. The objective is to find the subset of those items with the highest total profit while their total weight does not exceed some limited set capacity. One might imagine wanting to put as much value as possible in a knapsack while still being able to carry it. Because we are in the business of multi-objective optimization rather than single-objective optimization, we use a multi-objective variant.

Definition 2.1.5. A **multi-objective knapsack problem** is a problem with the following attributes[1, p. 267]:

- There is some number of objectives, M, and some number of items N.
- There are profits a_{ij} and weights b_{ij} attributed to the items, and there are maximum capacities c_i (where i varies over $1, \ldots, M$ and j over $1, \ldots, N$). For some i the capacity may be chosen to be infinite to not add an extra capacity constraint.
- Solutions consist of binary vectors of length N, one bit for each item to encode whether that item is included in the solution or not. The space of admissible solutions consists of those solutions that satisfy

  $$\sum_{j=1}^{n} b_{ij} x_j \leq c_i$$

  for all $i \in \{1, \ldots, M\}$.
- The objective functions are given by

  $$f_i(x) = \sum_{j=1}^{N} a_{ij} x_j$$

  for all $i \in \{1, \ldots, M\}$.

MNK landscapes and ρMNK-Landscapes

MNK-landscapes and ρMNK-landscapes are randomly generated problems with a variety of tunable parameters.

Definition 2.1.6. An **NK-landscape** is a single-objective problem that has as its solutions all binary vectors of some given length N. The various numbers in the vector, or genes, have some number K of genes associated with them. These are either chosen randomly, or as the next K genes, wrapping around to the start of the vector when the end of the binary vector is reached. For every gene there is a function defining a score based on that gene and its associated genes, being a function that maps $\{0, 1\}^{K+1}$ (the gene and its K associated genes) to a random number on $[0, 1)$. Averaging these components gives the objective value for the entire solution[20, p. 120].

Definition 2.1.7. A **MNK-landscape** is a multi-objective problem that has M separate objectives, each defined as in the case of the NK-landscape[20, p. 120].

Definition 2.1.8. A **ρMNK-landscape** is a MNK-landscape where the objectives are correlated with some correlation ρ. [20, p. 120] defines a precise way to generate ρMNK-landscapes that follow a supplied correlation coefficient between the objectives quite exactly, which could be useful to analyze the effect of correlated objectives.

2.1.2 Measuring performance

Convergence measure (CM), inverted generational distance (IGD) and hypervolume (HV) are widely-used performance metrics[8, p. 6]. Among these, hypervolume is of particular interest, as it is the performance measure we will use in our experiments.

Definition 2.1.9. Hypervolume as a metric for the performance of a multi-objective optimizer, measures the space that is (weakly) Pareto-dominated by a set of solutions[22, p. 86][23, p. 5].

Hypervolume "can give a comprehensive assessment in terms of convergence and diversity"[8, p. 6], and can be said to have "nicer mathematical properties than other performance indicators"[22, p. 1].

2.2 Multi-objective Evolutionary Algorithms

What we are generally trying to achieve in multi-objective optimization, is to find the Pareto set. A multi-objective evolutionary algorithm (MOEA) is a method that attempts to find, or at least approach this set. There is a wide variety of evolutionary approaches, but what they have in common is the following:

Definition 2.2.1. In an **evolutionary algorithm** or **genetic algorithm** we maintain a set, often called a **population**, of solutions to a problem, on which evolutionary operators operate to generate solutions, in an attempt to find solutions with higher fitness. There are three major evolutionary operators [25, p. 24]:

- **Mutation**, where a random (often small) change is made to a solution
- **Recombination** or **cross-over**, where new solutions are created by recombining multiple existing solutions.
- **Selection**, some mechanism by which certain solutions are allowed to be added to the population and other solutions are being removed from the population, based on their fitness (which is defined as part of the problem).

Evolutionary algorithms are a logical choice for multi-objective optimization, because as a consequence of their population-based nature, EAs can be used to approximate the entire Pareto set/front in a single run (in which one population is subject to evolutionary operators for some number of fitness evaluations, running time, or until some other stop condition is achieved)[9, p. 32][4, p. 712][1, p. 264].

2.2.1 NSGA-II

NSGA-II is a MOEA using **non-dominated sorting**, where solutions are sorted into **non-dominated fronts**. The first non-dominated front is the Pareto set of the current population, the second non-dominated front is the Pareto set of the current population when the first non-dominated front is left-out, and so forth[10, p. 183,184]. This way of sorting the population is the main way of assessing the relative fitness of solutions in NSGA-II[1, p. 267]. We refer to the solutions in the first non-dominated front as being of **non-domination rank** 1, those in the second front as being of non-domination rank 2, and so forth[10, p. 184].

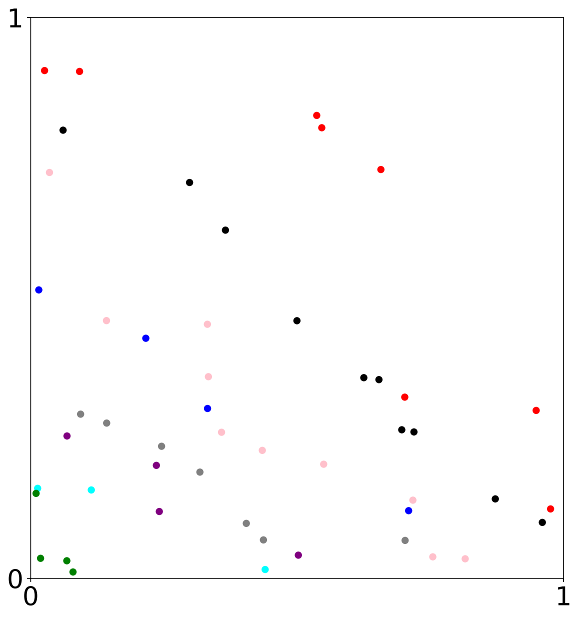


Figure 2.2: Example set of solutions to a 2-dimensional multi-objective optimization problem. Each non-dominated front in it's own color. For example: The first non-dominated front is red, the second front is black, the last (8th) Pareto front is in green.

Within the fronts that the population is sorted into, **crowding distance** is measured, which is used as a score to maintain diversity within the population. The solutions which have the lowest or highest value on any of the objectives will have infinite crowding distance. All other solutions will have a crowding distance which scales with the distance to their closest neighbors, measured individually along all objectives and then added to get a single number[10, p. 185].

NSGA-II uses a combination of **binary tournament selection** (where pairs of solutions are selected from the population and the better one is chosen), crossover, and mutation to generate a new population from the current population of size μ, which is then merged with the current population, after which the best μ solutions are chosen to end up with the next generation. The first non-dominated front takes precedent over the second, the second over the third, and so forth, and when a selection has to be made within one single front, the individuals with the highest crowding distance are chosen[1, p. 186].

NSGA-II was introduced in 2002 as an algorithm that attempted to alleviate the three difficulties for which MOEAs using non-dominated sorting were mainly criticized[10, p. 182]:

- High computational complexity, more specifically being computationally expensive for large population sizes
- The lack of **elitism**, where good solutions get the opportunity to survive for many generations
- The need for specifying a parameter (specifically for ensuring diversity, the so-called sharing parameter), considered undesirable

NSGA-II is "[arguably] one of the most popular Pareto dominance based MOEAs" (April 2009)[7, p. 285] and "the most frequently-used Pareto dominance based EMO algorithm in the literature" (April 2015)[1, p. 267].

Non-dominated sorting

A naive approach to non-dominated sorting would be to iterate through all possible pairs of solutions and check whether one dominates the other, while keeping track of the solutions that so far are non-dominated, then mark all the undominated ones with their non-domination rank, temporarily remove them from the population, and start over with the rest of the population until the entire population has been sorted. This however has a complexity of $O(MN^3)$[10, p. 184].

The approach which we will be using has a complexity of $O(MN^2)$[10, p. 184]. While going through all possible pairs of solutions, one keeps track for every solution which solutions it dominates as well as a counter of how many solutions it is dominated by. To get the first non-dominated front, one goes through the population to find all solutions with their counter at 0. To find the other fronts, one simply iterates through all the lists of dominated solutions that are kept for the previous front, and for all these solutions decrements their counter. If any counter reaches 0, it is part of the next front (as apparently everything that dominated it was in a previous front).

Both approaches are taken from [10, p. 184].

Crowding distance

The crowding distance is another measure for the fitness of solutions within NSGA-II, one that is used for solutions within the same Pareto front. The idea is to quantify the significance of a solution to maintaining diversity. Solutions which are on the extremes of any objective will have infinite crowding distance. Other solutions will have a crowding distance relating roughly to how close a solution is to its nearest neighbors.

One starts by initializing the crowding distance for every solution at 0. Then one iterates through the objectives, and for each objective one sorts the solution according to that objective. The first and last solution get their crowding distance set to infinity, and for all other solutions one adds the difference between the next and the previous solution.

Together with non-dominated sorting we can define the crowded-comparison operator, which will combine the measures of non-domination rank and crowding distance into one general measure that can be used on any solutions within an NSGA-II population.

Definition 2.2.2. The **crowded-comparison operator** is defined as follows [10, p. 185]:

- If two solutions differ in non-domination rank, then the one with the lower non-domination rank dominates the other.
- If two solutions are of the same non-domination rank, then the one with the higher crowding distance dominates the other.

Now we can recap how NSGA-II works. One starts with some randomly generated population and assigns non-domination ranks as well as crowding distances. Then, until some stopping criterion has been met, one repeats the following steps:

1. Using (binary tournament) selection, crossover and mutation, generate a new population and combine it with the current population.
2. Assign non-domination ranks as well as crowding distances.
3. Create the new generation by adding as many non-dominated fronts as fit, starting at the first, according to the population size.
4. If this leaves room in the next generation (because even though it is smaller than the population size permits, adding the next non-dominated front would make it greater than permitted) fill the remaining space with the solutions from the next front that have the highest crowding distance.

2.2.2 MOEA/D

MOEA/D is a **decomposition-based** EMO algorithm, where a multi-objective problem is decomposed into a number of single-objective problems. In MOEA/D one starts with a number of **weight vectors** which attribute weights to the various objectives. Using a **scalarizing function** the weight vectors are turned into mappings from the solution space to a single real number, defining the objective for that weight vector[1, p. 268].

The weights for the various objectives are selected from $\{0, \frac{1}{H}, \frac{2}{H}, \ldots, 1\}$, where H is freely specified integer. The weight vectors are chosen to be all vectors that can be formed using these numbers such that the weights sum up to one. As such, the number of weight vectors scales with H [1, p. 286]. There are multiple ways to define a scalarizing function based on the weights, more on that will be explained later.

In MOEA/D every weight vector has one and only one solution attributed to it. Thus the H parameter together with the number of dimensions to the problem determines the population size[1, p. 268,270].

The weight vectors will have a number of **neighbors** associated with them. The exact number of neighbors can be specified using a parameter, the **neighborhood size**. Which vectors are neighbors to which is decided based on the Euclidean distance between them. Vectors that are closer to each other are preferred to be neighbors over vectors that are distant from each other. Within these neighborhoods new solutions are generated and compared to the currently held solutions as candidates for replacing them[1, p. 268].

The basic assumption of MOEA/D is that a uniformity of weight vectors will naturally lead to a diversity of Pareto optimal solutions. When the Pareto front is close to the hyperplane $\sum_{i=1}^{M} f_i = 1$ in objective space it is thought that MOEA/D can obtain uniformly distributed Pareto optimal solutions, but when the Pareto front is complex, this basic assumption of MOEA/D may be violated[21, p. 234]. There are a number of variations on MOEA/D to be found in the literature that attempt to use adaptive weight adjustment to adapt to the shape of a more complex Pareto front, such as [21].

It is possible that MOEA/D will find unique solutions for each weight vector, but this is not always the case. "The number of obtained non-dominated solutions by MOEA/D is often much smaller than the number of weight vectors"[3, p. 170] because a single good solution can be shared by multiple weight vectors and not all solutions are always non-dominated[3, p. 170], thus the population may contain solutions that are not included in the approximation of the Pareto front.

Neighborhood size

There are some obvious difficulties identified with the strategy employed by MOEA/D when it comes to the size of the neighborhoods. For example, a solution could be generated within a neighborhood, which is an improvement for one of the weight vectors which is not in that neighborhood. It seems like this could easily be remedied by using a large neighborhood size, but this introduces the problem that the diversity of solutions may rapidly decrease, because when a solution is found that is good for a large number of vectors, it will replace the current solution of all these vectors thus vastly reducing the diversity[3, p. 173-

174]. There is no obvious best choice for the neighborhood size either, as in [1, p. 276] the best specification for neighborhood size varied between 5, 10 and 50, depending on both the problem that was being solved and the performance indicator used.

However, in practice it seems like a minor detail. Experiments show that MOEA/D is not very sensitive to the neighborhood size, except in that it should not be exceptionally small. A neighborhood size of around 10 is generally found to be effective, though this may be a consequence of the specific problem and other settings of the algorithm[4, p. 727-729][1, p. 276].

Scalarizing functions

In MOEA/D there are three methods to use the weights to create concrete scalarizing functions (in the original paper, [4, p. 713-715], at least. The method is generic enough that one could use an entirely different scalarizing function instead).

- **Weighted sum**, where the output of the scalarizing function is just the weighted sum of performance on the various objectives, weighted exactly as described by the weight vector. One problem with the weighted sum approach is that in the case of non-concave Pareto fronts not every Pareto optimal solution can be found[4, p. 713].

- **Weighted Tchebycheff**, which uses some reference point z^* in addition to the weight vector w and defines the function

  $$f^{TE}(x|w, z^*) = \max_{i=1,2,\ldots,k} \{w_i * |z_i^* - f_i(x)|\}$$

  which is to be minimized. In [4, p. 713] the reference point was defined as

  $$z_i^* = \max_{x \in \Omega} \{f_i(x)\}$$

  where Ω is the union of all populations so far. (According to [1, p. 268] though, it was scaled by a factor of 1.1, which is also what they use). In other words, it tries to minimize the maximum distance over all objectives in respect to a value 10% higher than their overall maximum value ever recorded in a population, where these distances are weighed as specified in the weight vector.

- **Penalty-based boundary intersection (PBI)**. In this case the following function is to be minimized:

  $$f^{PBI}(x|w, z^*) = d_1 + \theta d_2$$

  where

  $$d_1 = \frac{||(z^* - f(x))^T w||}{||w||}$$

and

$$d_2 = ||f(x) - (z^* - d_1 \frac{w}{||w||})||$$

where θ is some user defined parameter, which in [4, p. 724] was 5 (though they acknowledge that finding the right value may "require a bit more human effort"[4, p. 724]) and in [1] varied over 0.01, 0.1, 1 and 5. w refers to the weight vector. There is some reference point z^* corresponding to the topmost boundary, just as with weighted Tchebycheff. The idea is that from this reference point lines emanate in various directions corresponding to the weight vector, along which the solutions are scored according to how close they get to the reference point (d_1). Because forcing the solutions to be on the line would create an equality constraint, which is undesirable, deviation from the line is instead allowed but penalized (d_2)[4, p. 714].

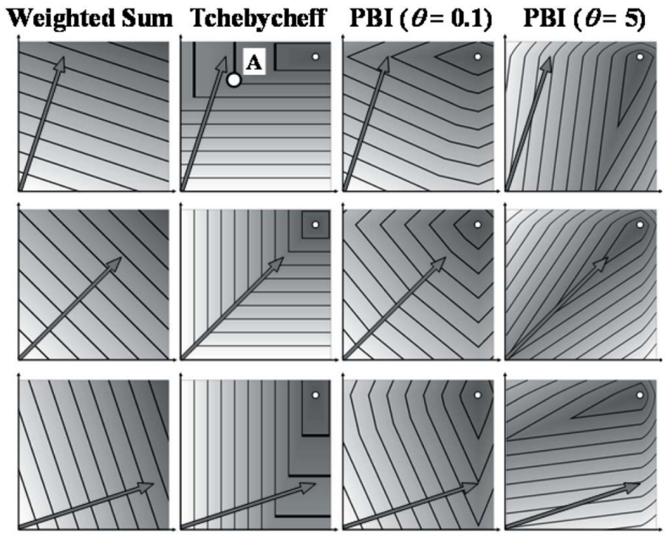


Figure 2.3: Contour lines illustrating the effect of the various scalarizing functions. The weight vector is shown as an arrow. Source: [1, p. 268]

The choice of scalarizing function can be very significant. For example, in one experiment in [1, p. 274] (the 10-500 multidimensional knapsack problem, performance measured in hypervolume relative to the origin) MOEA/D using Tchebycheff was found to perform only 87.7% as good compared to when the weighted sum scalarizing function was used. In the exact same context, PBI was found to be between 99.7% and 61.9% as good, depending on the choice of θ, where $\theta = 0.01$ was found to perform best and $\theta = 5$ was found to perform worst. However, there is no objectively best scalarizing function. Simply by varying the number of dimensions of the problem and keeping everything else the same, either the Tchebycheff, weighted sum or PBI was found to perform the best. Within PBI, the best value of θ was either 5 or 0.01 in this particular set of experiments.

Now we can recap exactly how MOEA/D works:

One starts be generating all possible weight vectors based on the number of dimensions M and the parameter H, and attributing random solutions to them. Then, while the stopping criterion has not been met, one repeats the following instructions: For each weight vector, select two solutions from its neighborhood, and apply crossover and mutation to generate a new solution. Using the scalarizing function, compare this to the current solution attributed to each weight vector in the neighborhood and if it scores better, replace the current solution with it[1, p. 267-268][4, p. 713-716].

Variants

Many variants on MOEA/D have been suggested. For example, it has been suggested that the various sub-problems of MOEA/D may have different computational difficulties and therefore it would be reasonable to assign different amounts of computational resources to different problems, leading to the idea of MOEA/D-DRA, with dynamical resource allocation[5, p. 2].

As previously stated, the basic assumption of MOEA/D is that it can obtain a diversity of Pareto optimal solutions by having uniformity of the weight vectors, which works well if the Pareto front is close to the hyper-plane given by $\sum_{i=1}^{m} f_i = 1$. This assumption may be violated, leading to a situation where several sub-problems of MOEA/D will have the same optimal solution, and as such computing effort is wasted on trying to optimize two identical problems. It is also possible that at the same time, there are non-dominated solutions distributed in a narrow gap in one of the objectives, which don't correspond nicely to any weight vector. In an attempt to fix this problem MOEA/D-AWA was introduced, which among other things does adaptive weight adjustment, where sub-problems are deleted from crowded parts of the Pareto front and new sub-problems introduced into other parts of the Pareto front. Experimentally it was shown that this approach indeed helps MOEA/D to obtain better uniformity in cases where the Pareto front has a sharp peak and a low tail, and it also performs better on low-dimensional Pareto fronts in many-objective problems. On problems with a discontinuous Pareto front the adaptive weight adjustment helps to recognize discontinuous parts of the Pareto front and reduce the computational effort spent on these parts[21, p. 234].

Though NSGA-III is arguably a successor of NSGA-II rather than MOEA/D (having been created by the same author and sharing the NSGA name), NSGA-III uses an approach similar to MOEA/D in that it explicitly searches in a number of different directions which are uniformly distributed, except NSGA-III defines multiple reference points rather than search directions. NSGA-III uses this mainly to maintain diversity on top of the non-dominated sorting approach which is also seen in MOEA/D. NSGA-III is explicitly designed to be able to handle problems with four or more objectives. Experimentally it outperformed MOEA/D on various but not all test problems which were chosen from the DTLZ and WFG problems[11].

Another variant is MOEA/D-EGO, which uses Gaussian stochastic process models on top of the standard MOEA/D. That is to say that the objective functions are based on a Gaussian stochastic process. It is aimed at the situation where the objective function is computationally very expensive and reduces the actual calculation of the objective functions to only the initial population, and a limited number of the generated solutions when the model deems these to be valuable to evaluate for model improvement. MOEA/D-EGO was experimentally not compared to either MOEA/D or NSGA-II, and as such there are no real results relevant to this document. However, it is perhaps interesting to note that it offers the possibility of some parallelism over its competitors, by virtue of the MOEA/D-based design[6, p. 2].

And then there is MOEA/D-ACO, which attempts to combine MOEA/D with ant colony optimization. Knowledge about the problem is discovered during the search and represented as a pheromone matrix in which the value for any solution component is based on an empirical measurement of how likely this component is present in a promising solution, and some problem-specific information is generated before the problem to create a heuristic information matrix. In MOEA/D-ACO every weight-vector has its own associated ant which uses a pheromone matrix and heuristic information matrix to probabilistically generate new solutions. Each ant has its own heuristic information matrix, which for example contains information about the price-to-weight ratio of items in a knapsack problem. The pheromone matrices are shared to some extent, with weight vectors having been grouped and the ants of each group sharing a pheromone matrix. In their experimental results they found that MOEA/D-ACO outperformed MOEA/D-GA (yet another variant, MOEA/D with local search) in the context of a knapsack problem[27, p. 1845-1846,1849].

Though it is hard to make an exact statement about such differences, it seems like in the literature there are a lot more variants of MOEA/D than for example of NSGA-II. Though no hard conclusions can be drawn from this, it might give us to wonder whether MOEA/D perhaps lends itself especially well to being adapted with extra mechanisms to optimize it for certain specialized purposes. Or a perhaps more interesting conclusion could be that within MOEA/D there is a lot of room for improvement.

Chapter 3

Comparing NSGA-II and MOEA/D

3.1 Problem-dependence

Perhaps a good central point of departure is to notice that **claims of superiority of either NSGA-II or MOEA/D have to be specific to the problem that the algorithms are being tested on**.

This point is most clearly illustrated when NSGA-II and MOEA/D (and some other algorithms as well) are compared on a large variety of test problems, such as WFG and DLTZ problems and variations on them. Which one is better purely depends on the problem. The same variation can be observed within MOEA/D. Tchebycheff, PBI and weighted sum are all the best approach on some problem and the worst approach on some other problem (within the scope of the algorithms here named)[3, p. 177-178,180-182]. As such a simple comparison of NSGA-II or MOEA/D in terms of which is better, is not of much use. However, it might instead be interesting to research what the determining factors are for whether NSGA-II or MOEA/D will perform better.

A few factors were identified as contributing to the high-performance of recently proposed weight vector-based algorithms on certain problems [3, p. 188]. (Note: the paper cited here probably referred to more recent algorithms of this sort, but as these factors mainly have to do with a decompositional approach using uniformly distributed weight-vectors, one would expect these factors to be significant for MOEA/D as well):

- **The triangular shape of the Pareto front** which can be said to be similar in shape to the distribution of weight vectors
- **A small Pareto front** in comparison to the size of the feasible region in objective-space.
- **Separable decision variables**, allowing for easy convergence and diversification and thereby making it possible to focus on the uniformity of

obtained solutions over the Pareto front.

Apart from varying the shape of the Pareto front, we might also look at **varying the density of points in the Pareto front** in certain areas, as the identified reason that the shape would matter is its similarity to the shape of the distribution of weight vectors. This distribution is not just of a certain shape, but also uniform, like the distribution of weight vectors.

Experimental results have shown that the performance of NSGA-II relative to MOEA/D improves when objectives become more correlated, and when there is a large number of dimensions (in particular 6 or more) NSGA-II outperforms some or all forms of MOEA/D[1, p. 279]. Thus it seems that when the objectives are highly correlated, the search ability of NSGA-II does not severely degrade when the number of objectives increases while that of MOEA/D does. Problems like multi-objective knapsack, and variations on the DLTZ and WFG test problems have been used repeatedly to show that Pareto-Dominance based algorithms face difficulties when the number of objectives increases [3, p. 170]. This indicates that the **correlation between objectives** may be of interest, as well as the **number of objectives**.

However, the size of the Pareto front relative to the decision space was already identified as a potential factor of interest and within ρMNK-landscapes it has been shown that the ratio of the number of Pareto optimal solutions compared to the size of the search space varies with the correlation between objectives. This ratio goes down as objectives become more correlated[19]. Thus, even though the correlation between objectives may be a significant factor to study it may be hard to separate this from the factor of relative size of the Pareto front.

In another experiment, it was found that NSGA-II outperforms some algorithms designed specially for **many-objective optimization** (loosely defined as having 4 or more objectives) on some problems with relatively low dimensions (DTLZ7 and WFG8 are given as examples). MOEA/D's search ability is found to work very well on some problems (such as DTLZ2, DTLZ3 and WFG1) but encounters great difficulties on others (such as DTLZ7 and WFG8). It is also claimed that "MOEA/D appears to be more competitive in relatively low-dimensional problems"[8, p. 13]. This further reinforces the idea that that the relative performance of algorithms is highly problem-dependent and it contributes to another of the identified factors for research: The number of objectives.

3.2 Problems with current test suites

In the test suites we have seen so far, we have seen very little attempt to isolate particular factors that make a problem difficult. Instead, we see attempts in providing a test suite that tries to reflect the sort of difficulties that one encounters in a real-world problem, such as DTLZ[13, p. 3], or which mostly combines a number of complexities to create test problems such as WFG[18, p. 14].

What we will try to do instead, is to isolate certain difficulties as much as possible, such that we can make a harder claim about what exactly an algorithm is good or bad at.

3.3 Constructing test-problems

Out of the test-problems seen so far, ρMNK-landscapes may be of particular interest. Within ρMNK-landscapes the number of objectives and the correlation can be exactly specified. These are the parameters ρ and M.

Another one of the potential factors, that of separable decision variables, is also one that could be researched using ρMNK-landscapes, as the parameter K in this context has the exact role of varying how many variables influence each other.

As stated before the size of the Pareto front relative to the feasible region varies with the correlation between objectives, which can be specified within ρMNK-landscapes. However, we might be interested in finding the effect of one of these factors apart from the other in which case ρMNK-landscapes might not offer a straightforward approach.

The uniformity and shape of the Pareto front could perhaps be modified simply by a transformation on the coordinates of the objective space.

In addition to this, the paper that introduced DTLZ problems also offers some ways to design test problems. Especially interesting is the Bottom-Up Approach where one simply chooses a Pareto front which one describes as a parametric surface involving $M - 1$ variables where M is the number of objectives of the problem. One can then build an objective space by adding another variable in the parametric equations scaling the values up or down. Then one constructs the decision space by mapping the decision variables to the parameters of the parametric surface. The exact way in which this is done can be varied (with having multiple (local) optima, being linear or not, etc.) and as such the difficulty in converging to the Pareto front can be regulated[13, p. 7-11].

3.4 Research Question

A number of factors that may influence the relative performance of NSGA-II and MOEA/D have been identified, namely:

- The separability of decision variables.
- The number of objectives.
- The correlation between objectives.
- The shape of the Pareto front (especially its similarity to the shape of distribution of weight vectors).
- The uniformity of distribution of solutions within the Pareto front.

- The size of the Pareto front compared to the size of the feasible region in objective-space.

The main research question would be: What is the influence of these factors on the relative performance between NSGA-II and MOEA/D experimentally? Which we can research on an individual basis, resulting in 6 sub-questions.

This could be done by designing test problems for these factors in which they are the predominant factor, or in which they can be varied. (For example, a test problem in which we can set the exact correlation between objectives could be used to measure the effect of this factor on the relative performance between NSGA-II and MOEA/D). There are a number of ways to measure the performance of MOEAs, of which the hypervolume of the Pareto-dominated part of the domain seems to be the best choice according to the literature. However, the drawback of hypervolume is that it may be expensive to compute, so exceptions could be made to this.

Partially this would reproduce some similar research in new contexts (for example, in [1] the effect of correlation and number of objectives is measured, but in the context of knapsack-problems only, while we would use ρMNK-landscapes), but it would also include some new experiments for the other factors.

As MOEA/D and NSGA-II are rather different in their approach, one being decomposition-based and the other non-domination-based, these results may also lead to further insight in the relative strengths and weaknesses of such approaches more broadly.

Lastly such results may inspire further experiments to investigate why the observed phenomena are there. One could investigate to what extent the relative performance of these algorithms correlates with for example their ability to maintain a diverse population. The exact measurements would depend on the experimental results when testing the effect of the aforementioned factors.

Chapter 4

Experiments with Specified Attributes

4.1 Experiments using ρMNK-Landscapes

Our main focus in these experiments is to investigate the various factors in as much isolation as possible.

For the most part this can be done, but the size of the Pareto front is an exception. The dimensionality as well as the correlation between objectives for a large part determine the size of the Pareto front. Therefore we will leave this factor out.

4.1.1 Methodology

In general, we will use a running time of 30000 evaluations per algorithm, and repeat the experiments 10 times (for each repetition of an experiment a new landscape will be generated). All listed scores are based on the average of the 10 repetitions, and expressed as a percentage of the best performer, so the best result always has a score of 100.

The population size will be kept at (or as close as possible to) 50. This is to ensure that it's feasible to calculate the hypervolume of the Pareto front, as that is the performance metric we'll be using. This limit of feasibility is purely a consequence of the speed of the computer on which the algorithm will be performed.

We use uniform crossover with a swapping probability of $\frac{1}{2}$, and a mutation rate of one over twice the number of variables, which for the most part will end up being $\frac{1}{40}$.

Additionally, to test for statistical significance we use the Mann-Whitney U test, and call a difference significant if $p < 0.05$.

We will show visualizations of the results with most of the experiments. The visualizations within one figure are all the result of the exact same problem.

(The problem is randomly generated, but the same problem is used for each algorithm in the visualizations; as opposed to generating a separate random problem for each algorithm).

4.1.2 Separability

To measure the effect of the correlation between objectives we use ρMNK-landscapes and vary the ρ parameter, as described earlier.

We have some number of decision variables (N), which all have an attributed fitness function which also involves a number of variables after it (K), mapping every possible input to some random number between 0 and 1. These fitness functions can be averaged to get the fitness of the entire solution. The dimensionality of the functions can be varied (M) and the random values can be generated in a way which creates some desired correlation (ρ) between the overall fitness functions.

When we are interested in the effect of separability of decision variables, we can simply vary the number K. Which values are possible is limited by two factors: It's mathematically impossible to have $K \geq N$, and the amount of memory needed to store the functions scales exponentially in K (because for every gene and every dimension of the problem, we store 2^{K+1} values).

As we are not trying to measure the effect of the number of dimensions, we will simply use $M = 2$ because it is easiest to visualize and allows the most control over the population size, and $\rho = 0$ because we want to keep its effect as neutral as possible. For K and N we find experimentally that for reasons of feasibility we can let K go up to about 16 when the solution size N is 20. So we will vary K over 0, 1, 2, 4, 8 and 16.

Results

K	NSGA-II	WS	TCH	PBI-5	PBI-0.1
0	100	99.823	99.991	99.965	99.703
1	99.455	99.778	100	99.393	98.599
2	99.783	99.374	99.767	100	98.808
4	100	96.447	98.644	99.173	96.391
8	100	97.427	98.556	98.944	95.328
16	100	93.908	92.767	94.480	92.172

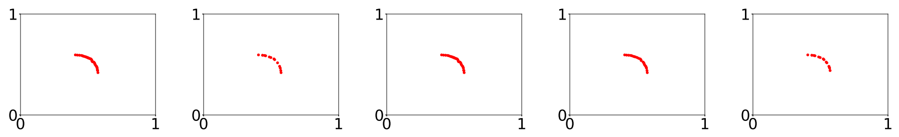


Figure 4.1: NSGA-II, MOEA/D (WS), MOEA/D (TCH), MOEA/D (PBI-5), MOEA/D (PBI-0.1), $K = 0$

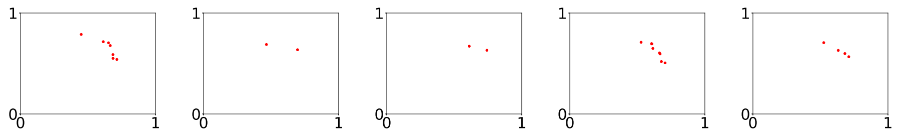


Figure 4.2: NSGA-II, MOEA/D (WS), MOEA/D (TCH), MOEA/D (PBI-5), MOEA/D (PBI-0.1), $K = 16$

Discussion

NSGA-II seems like a clear winner as it is always within 99.45% of the best score, and the MOEA/D variants seem to drop off relative to it as K gets higher. For $K = 16$, the difference between NSGA-II and every other method is found to be statistically significant.

Thus it seems that indeed, as it becomes harder to separate variables, NSGA-II seems to do relatively better. Though it must be said that this is at rather high values of K because at $K > 10$ all variables are connected in some way.

To explain this result we might hypothesize the following: As the number of variables that are connected to each other increases, the fitness correlation between solutions and their offspring under either mutation or recombination should become lower. Suppose we were to mutate one bit in a solution. If $K = 0$ the effect on the fitness is at most $\frac{1}{N}$ in general and $\frac{1}{20}$ in our case, as the fitness function is the average of each variable's associated fitness function, which is based only on itself. But in the case where $K = 16$, this is $\frac{17}{20}$ (as a solution appears in the fitness contribution of 16 other genes on top of its own contribution). Thus, it may occur more frequently in MOEA/D that a good solution is generated (good in the sense of being close to the Pareto front) but is rejected because it wasn't good within the neighborhood where it was generated, when K is higher. NSGA-II just cares about the distance to the Pareto front, so it should have no such problems. A simple way to test this would be to see whether it makes a difference to set the neighborhood size to the size of the population, and re-run the experiment.

K	NSGA-II	WS	TCH	PBI-5	PBI-0.1
0	100	99.884	99.991	99.921	99.636
1	99.851	99.633	99.558	100	99.140
2	99.829	99.599	100	99.935	99.342
4	99.383	95.710	100	97.526	97.723
8	100	99.297	98.526	98.426	97.705
16	100	98.403	97.063	97.351	95.812

Within these results there is less variation than in the first result, so it seems like the phenomena just hypothesized is playing a role. The differences between NSGA-II and MOEA/D under $K = 16$ are only statistically significant for PBI-0.1 and TCH. However, it does not remove the variation to the point of there being no clear difference depending on K.

A problem with using NK-Landscapes is that changing the K parameter could affect the shape of the Pareto front, as well as how solutions are distributed within the Pareto front. This limits our ability to explain all phenomena in terms of separability and its consequences.

One might also wonder whether the combination of $N = 20$ and $K = 16$ doesn't make the problem so complex as to be indistinguishable from randomness. Flipping one variable would (more or less randomly) affect the value of 16 variables before it, as their fitness contributions all depend on the next $K = 16$ variables.

To test whether there is any significant difference, a problem was specified where solutions are also bit-strings of length 20, but every bit-string has its own completely random fitness value (between 0 and 1) for every objective. We found the following result:

K	NSGA-II	WS	TCH	PBI-5	PBI-0.1
0	100	99.336	99.602	99.563	98.646

The relative order between the algorithms is still intact, however, we see much smaller differences. This indicates that there is still some structure present in the ρMNK problems of the sort we tested that algorithms can take more or less advantage of. It must be said though, that in this case the difference between NSGA-II and all MOEA/D variants is again statistically significant.

4.1.3 Number of objectives

To study the effect of the number of objectives we also use ρMNK-landscapes. We will keep $p = 0$ and $N = 20$ and we set $K = 4$, which is a somewhat arbitrary choice. M we can vary, but we will have to keep some practicalities in mind here. When the dimensionality gets higher we know from theory that the proportion of solutions in the Pareto-front gets higher. We have only the computational power to compute hypervolumes of Pareto-fronts of about 50 solutions, and MOEA/D is limited in it's choice of population size depending on M. For $M = 2$, the allowed population sizes for MOEA/D are all integers above 2, but for example for $M = 20$, the smallest allowed population sizes are 20 and 210, where 210 might already be problematically high as it could produce a Pareto-front with significantly more solutions than 50. Unless we want to have a MOEA/D in which every search direction is given by a unit vector, we can handle at most $M = 9$. And thus we will vary M over $2, 4, 6, 8, 9$.

The population size will be given by whatever the highest population size allowed for MOEA/D of at most 55 is (the exact number will vary depending on M). NSGA-II will use the same population size.

Results

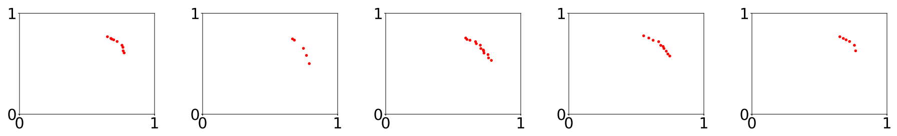


Figure 4.3: NSGA-II, MOEA/D (WS), MOEA/D (TCH), MOEA/D (PBI-5), MOEA/D (PBI-0.1), $M = 2$

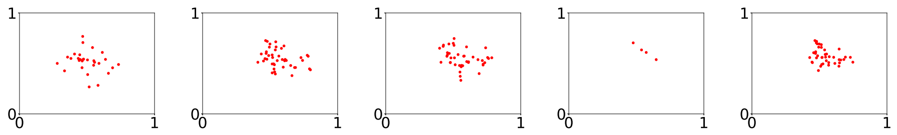


Figure 4.4: NSGA-II, MOEA/D (WS), MOEA/D (TCH), MOEA/D (PBI-5), MOEA/D (PBI-0.1), $M = 9$; only the first two objectives

M	NSGA-II	WS	TCH	PBI-5	PBI-0.1
2	99.531	96.830	100	98.630	97.236
4	87.564	100	99.207	89.206	98.800
6	38.140	94.430	95.902	75.498	100
8	62.548	93.933	91.607	63.928	100
9	34.635	94.029	92.862	58.444	100

Results

We see a great amount of variation in results, not just between NSGA-II and MOEA/D, but between all variants. It seems that as the number of objectives increases, all variants of MOEA/D except PBI with $\theta = 5$ become clearly better than NSGA-II. The difference between NSGA-II and all other algorithms is statistically significant, as is the difference between PBI-5 and all other algorithms, at 9 dimensions. The same goes for 8 dimensions, except between NSGA-II and PBI-5.

It's interesting to see that the algorithm that is clearly the best at accommodating a large number of objectives, PBI with $\theta = 0.1$, is incidentally the algorithm most like (in terms of design) the algorithm which within MOEA/D is clearly the worst: PBI with $\theta = 5$. The only difference is in the θ parameter, which regulates for the various directions in which the search takes place how strongly deviation from the exact direction is to be penalized. One would almost suggest that it is important to not care too much about the direction of the search, but if that was the only factor to explain the result, it would be inexplicable why NSGA-II, which doesn't care about directions of search except for wanting results not to be too close to each other, performs so badly.

The performance of NSGA-II can be explained quite easily. At 6 objectives, it starts occurring quite frequently that 100% of the solutions within a generation, including those generated by mutation and/or crossover, are non-dominated. At 8 objectives this happens after 10 generations already, and the number of non-dominated solutions typically doesn't go below 90%. As the selection mechanism within NSGA-II is non-dominated sorting, this means that there is very low selection pressure to go towards the Pareto front, and in some generations (when 100% of the solutions are non-dominated) no selection pressure at all to do so. However, for MOEA/D there is a variety of fitness functions (corresponding to their weight vectors), which all maintain their own selection pressure.

What's left to explain is why the value of θ has such a big impact as the number of objectives increases. The difference that θ makes, is that it regulates how high the penalty should be for deviating from the exact direction set by the weight vector. Its effect is linear; for $\theta = 5$ the penalty is 50 times as high as it is for $\theta = 0.1$. What may explain the behavior is that as the number of objectives increases, solutions generally are less close to the directions set by the weight vectors, as there are more parameters to spread over but there is the same number of solutions (as this is regulated purely by N, which was the same in all situations). Under a low penalty for this, such as $\theta = 0.1$, this does not do much, but under $\theta = 5$, it could have the effect of rejecting lots of good solutions (good as in being close to the Pareto front) because they weren't close enough to any weight vector. We will later use a continuous test problem, in which equally good solutions exist in all directions. It will be interesting to see if the difference goes away.

4.1.4 Correlation between objectives

To research the effect of the correlation between objectives we can again do something similar as with the previous experiments. We can use ρMNK landscapes where we fix $M = 2$, $N = 20$, $K = 4$. The correlation, ρ, will be varied over -0.9, -0.5, 0, 0.5, 0.9.

Results

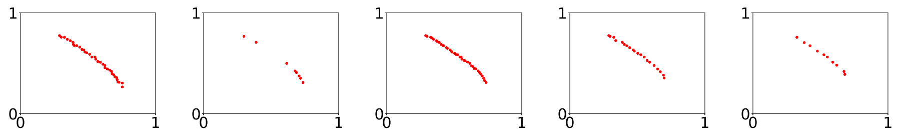


Figure 4.5: NSGA-II, MOEA/D (WS), MOEA/D (TCH), MOEA/D (PBI-5), MOEA/D (PBI-0.1), $p = -0.9$

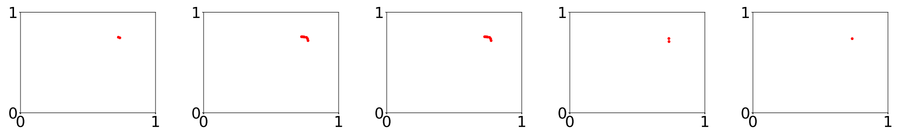


Figure 4.6: NSGA-II, MOEA/D (WS), MOEA/D (TCH), MOEA/D (PBI-5), MOEA/D (PBI-0.1), $p = 0.9$

ρ	NSGA-II	WS	TCH	PBI-5	PBI-0.1
-0.9	100	95.056	99.004	99.340	97.932
-0.5	99.471	98.278	100	99.321	97.783
0	99.823	100	96.511	99.242	98.786
0.5	100	99.326	98.703	98.660	96.106
0.9	98.968	98.467	100	99.475	99.305

Discussion

The results seem to be characterized mostly by somewhat of a lack of clear patterns.

This contradicts what one would expect based on [1, p. 279], where there was a clear trend that the relative performance of NSGA-II increases when the correlation increases. However, in that paper the experiment was performed on problems of 4, 6, 8 and 10 dimensions. It could be the case that the effect of correlated objectives depends on the number of objectives. Another reason could be that the effect is somehow specific to the shape of the problem, or the specific way that 'correlated problems' are constructed in [1], namely as weighted averages of uncorrelated problems[1, p. 267].

Therefore, it's worth attempting to see if we get clearer trends in the relative performance of the algorithms at a high number of dimensions. If not, we may have to conclude that correlation in itself does not have a clear effect, and it's dependent on the test problem and its construction.

The following results were created using 9 objectives (different values of p have been chosen because we cannot generate a problem with all negatively correlated objectives when there are more than 2).

ρ	NSGA-II	WS	TCH	PBI-5	PBI-0.1
0	41.591	92.498	91.459	55.903	100
0.25	81.251	98.358	100	52.071	95.965
0.5	99.197	97.166	100	62.155	97.250
0.75	97.113	99.479	100	90.654	99.182
0.9	99.774	100	87.899	73.014	92.556
0.95	100	90.718	88.662	72.659	77.351

Here it does seem like the same effect is present as described in [1, p. 279]. NSGA-II starts catching up to the other algorithms at higher correlations and overtaking them at extremes. At $p = 0.9$ the difference between it and other algorithms isn't statistically significant anymore. This is a great difference from what normally happens at $M = 9$.

This is a potentially interesting phenomenon for further research. We have the following hypothesis for why we get the results that we do: The higher the correlation between objectives, the smaller the Pareto front will be compared to the rest of the search space, while the more objectives, the more solutions are part of the Pareto front. We have already seen that as the number of objectives increases, NSGA-II has trouble in maintaining any selection pressure as there are fewer Pareto-dominance relations. Perhaps the increased correlation between objectives undoes some of this 'damage' to NSGA-II. We will later get back to this in detail (chapter 5).

4.1.5 Shape of the Pareto front

Specifically, we are interested in the effect of the shape of the Pareto front when it doesn't nicely match the distribution of weight vectors that MOEA/D uses. ρMNK-Landscapes themselves do not offer a simple way to regulate the shape of the Pareto front, but we can slightly modify the problem to allow for this. We introduce an extra parameter, $s \in [0, 1]$, which regulates how much the shape is transformed. For $s = 0$, there is no transformation, and for $s = 1$, the transformation is extreme. We simply transform the solutions of the ρMNK-Problem to polar coordinates and then change the argument of the solutions to be closer to either the point $\frac{\pi}{8}$ or $\frac{3\pi}{8}$, whatever they're closest to, by multiplying their distance to this argument by $1 - s$. This creates two increasingly smaller (as s goes to 1) pockets of solutions, as far as their directions from the origin go.

Results

s	NSGA-II	WS	TCH	PBI-5	PBI-0.1
0	99.928	100	99.285	99.792	99.174
0.2	100	99.446	99.005	99.199	98.100
0.4	98.435	99.689	100	96.888	99.397
0.6	100	97.417	96.757	96.828	98.389
0.8	98.346	98.442	100	96.696	98.648

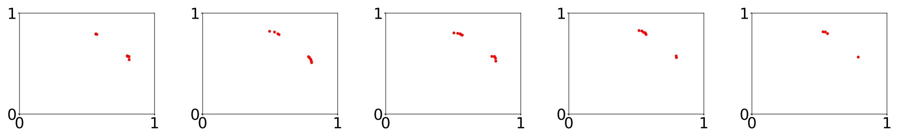


Figure 4.7: NSGA-II, MOEA/D (WS), MOEA/D (TCH), MOEA/D (PBI-5), MOEA/D (PBI-0.1), $s = 0.4$

Discussion

We see no clear trends in these results. There also aren't that many relations of statistical significance, as at no point is any algorithm different from all others in a statistically significant way.

4.1.6 Uniformity

Similar to the previous factor, we can explore the effect of uniformity by translating the argument of solutions when viewed as polar coordinates. We can introduce a new parameter, $u \in [0, 1]$, and modify the argument, as seen on a scale of 0 to 1 rather than 0 to $\frac{\pi}{2}$ to the power of u. At $u = 1$, this has no effect, at $u = 0$, all solutions will be concentrated on the axis of one objective, and in between, for example at $u = 0.5$ we will have an effect that, except for solutions with an argument of $\frac{\pi}{2}$, their argument will decrease, and the higher the argument, the more it decreases, which will make the solutions more concentrated at the bottom and more sparse at the top, relatively speaking. Of course this also deforms the space of solutions slightly (at least in effect), as with a high probability there won't be any solutions with an argument extremely close to $\frac{\pi}{2}$, and so there will be a space of increasing size where no solutions are found.

Results

u	NSGA-II	WS	TCH	PBI-5	PBI-0.1
1	99.083	99.317	100	98.948	98.378
0.8	97.630	98.706	100	98.240	95.118
0.6	97.953	99.197	100	99.402	98.362
0.4	99.296	99.141	100	98.935	97.535
0.2	98.882	100	99.706	97.475	97.446

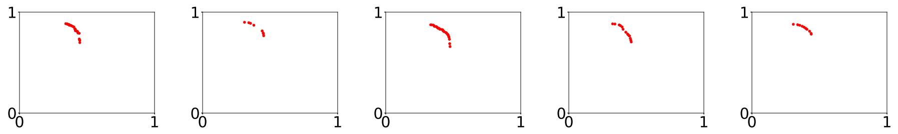


Figure 4.8: NSGA-II, MOEA/D (WS), MOEA/D (TCH), MOEA/D (PBI-5), MOEA/D (PBI-0.1), $u = 0.4$

Discussion

As with the shape, there is no algorithm that's different from all others in a statistically significant way at any point, and there don't seem to be any clear trends.

4.2 Continuous unit-disk quadrant problems

To not limit our use to just discrete problems, we formulate another problem to test the algorithms on. The feasible objective space will be given by the top-right quadrant of the unit-disk, the Pareto front will be the top-right quadrant of the unit-circle, and the search space will simply be the trivial parametrization of this space. (This approach is based on that of [13, p. 7,11].) In mathematical terms the problem is:

$$f_1(r,t) = r * cos(t\frac{\pi}{2})$$

$$f_2(r,t) = r * sin(t\frac{\pi}{2})$$

Where:

$$r, t \in [0,1]^M$$

We can also expand this to a multi-dimensional variant:

$$f_1(r, t_1, \dots, t_{M-1}) = r * cos(t_1) * cos(t_2) * \cdots * cos(t_{M-1})$$

$$f_2(r, t_1, \ldots, t_{M-1}) = r * cos(t_1) * cos(t_2) * \cdots * sin(t_{M-1})$$
$$f_3(r, t_1, \ldots, t_{M-1}) = r * cos(t_1) * cos(t_2) * \cdots * sin(t_{M-2})$$
$$\ldots$$
$$f_{M-1}(r, t_1, \ldots, t_{M-1}) = r * cos(t_1) * sin(t_2)$$
$$f_M(r, t_1, \ldots, t_{M-1}) = r * sin(t_1)$$

For the two-dimensional variant, we can easily apply the same transformations as previously to vary the uniformity with which solutions are distributed, as well as the shape of the Pareto front. However, we now have the advantage that we know the exact shape, so we aren't merely distorting it in a way we can describe, it is also trivial to see what the new objective space looks like. And when it comes to varying the density, we no longer distort the shape of the solution space, as the transformation maps the entire unit-disk quadrant to itself bijectively, except when $u = 0$.

However, it also comes with a disadvantage. There is no clear way to create a correlation between objectives, and our options to make the variables less separable are very limited. We could let the value of r distort θ, or the other way around, or we would have to introduce more variables.

For the disk-quadrant-based problems, we need another form of crossover and mutation, as these are problems with continuous variables. We will use simulated binary crossover[14] (this source does not give an explicit algorithm; our implementation was instead based on [16, p. 4]) with $\eta_c = 15$ and polynomial mutation[15, p. 2] with $\eta_m = 20$, as was used in [13, p. 20].

Also, because this problem is not that difficult to solve, we use a far smaller number of fitness evaluations as our stopping criterion: 400 fitness evaluations.

4.2.1 Shape of the Pareto front

To use this in the context of the shape of the Pareto front, we can use the same transformation as with ρMNK-landscapes.

Results

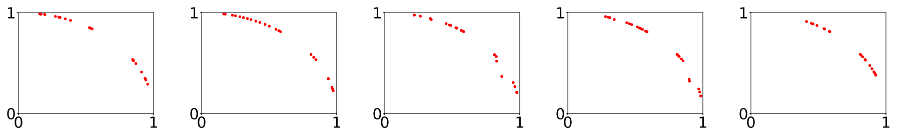


Figure 4.9: NSGA-II, MOEA/D (WS), MOEA/D (TCH), MOEA/D (PBI-5), MOEA/D (PBI-0.1), $s = 0.4$

s	NSGA-II	WS	TCH	PBI-5	PBI-0.1
0	100	97.952	99.396	97.829	98.233
0.2	100	97.841	97.942	98.236	98.210
0.4	100	96.947	98.405	97.604	98.502
0.6	100	97.763	98.988	96.546	98.467
0.8	100	97.734	99.402	96.578	98.752
1	100	99.261	99.874	99.013	99.937

Discussion

The data seems to converge as the shape variable gets smaller. Perhaps the problem is just too easy at that point, allowing all algorithms to solve it to near perfection and thus without a lot of differentiation. The significance tests shows no significant differences at $s = 1$. We do see a slight dip for some of the algorithms in the mid-range of values. For s between 0.2 and 0.8 the difference between NSGA-II and all other algorithms is statistically significant, and thus we may conclude that indeed, NSGA-II is significantly better than all of MOEA/D when the shape of the Pareto front is limited to a significantly smaller subset of all possible directions, except for when it consists only of a few points.

4.2.2 The separability of decision variables

As stated before, there is not a lot of room for testing this without modifying the problem, but there is one simple test we can do. Rather than defining:

$$f_1 = r * cos(t\frac{\pi}{2})$$

$$f_2 = r * sin(t\frac{\pi}{2})$$

We let the optimal value for r depend on t.

Define $d = 1 - \frac{|r-t|}{max(t,1-t)}$, that is to say, d is the difference between r and t relative to the greatest possible distance between them. We now have a problem with the exact same Pareto shape, comparable in difficulty, but instead of the optimum value of r being 1, it will be t.

$$f_1 = d * cos(t\frac{\pi}{2})$$

$$f_2 = d * sin(t\frac{\pi}{2})$$

Results

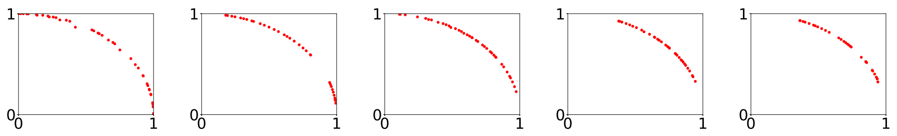


Figure 4.10: NSGA-II, MOEA/D (WS), MOEA/D (TCH), MOEA/D (PBI-5), MOEA/D (PBI-0.1), separable

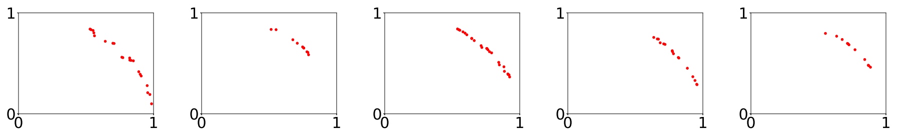


Figure 4.11: NSGA-II, MOEA/D (WS), MOEA/D (TCH), MOEA/D (PBI-5), MOEA/D (PBI-0.1), inseparable

separable	NSGA-II	WS	TCH	PBI-5	PBI-0.1
True	99.913	99.969	100	98.717	99.129
False	98.591	94.388	100	98.446	95.646

Discussion

When we go to an inseparable problem, we see an increase in the relative performance of MOEA/D (TCH) increases, and that of all the others decreases. Sadly we can not see any larger patterns because we don't have any variants that are "in-between". However, the difference between MOEA/D (TCH) and other algorithms is statistically significant in the inseparable case.

4.2.3 The number of objectives

We will use the multi-dimensional variant that was described earlier.

Results

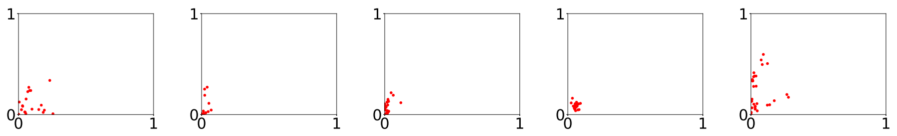


Figure 4.12: NSGA-II, MOEA/D (WS), MOEA/D (TCH), MOEA/D (PBI-5), MOEA/D (PBI-0.1), $M = 8$

M	NSGA-II	WS	TCH	PBI-5	PBI-0.1
2	99.916	100	99.791	98.697	99.096
4	61.604	76.532	97.561	94.342	100
6	18.705	44.639	70.335	82.074	100
8	11.650	49.900	77.660	54.008	100

We see very clearly that MOEA/D is very well equipped to handle the increase in number of dimensions for this test case. From being not significantly better than other algorithms (except for PBI-5), it becomes clearly (and statistically significantly) better than all other algorithms. At 6 and 8 objectives, all differences are statistically significant as well.

4.2.4 The uniformity

We will use the same technique as was used in the context of ρMNK-landscapes.

Results

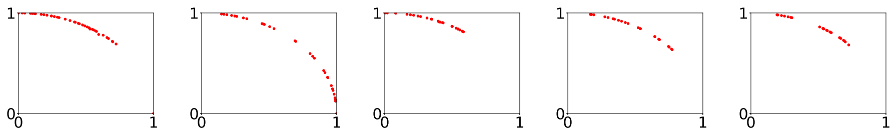


Figure 4.13: NSGA-II, MOEA/D (WS), MOEA/D (TCH), MOEA/D (PBI-5), MOEA/D (PBI-0.1), $u = 0.2$

u	NSGA-II	WS	TCH	PBI-5	PBI-0.1
1	99.671	100	99.350	98.497	98.731
0.8	100	99.546	99.141	97.745	98.410
0.6	99.251	100	99.355	97.369	97.885
0.4	100	99.288	96.970	95.046	96.284
0.2	100	94.329	85.363	84.475	85.814

Discussion

We see a clear pattern of NSGA-II improving relative to all other algorithms as the problem is being deformed to reduce the uniformity. At $s = 0.4$ and $s = 0.2$ the difference between NSGA-II and other algorithms is statistically significant. (As is the difference between WS an other algorithms, at $s = 0.2$.

Chapter 5

Many-objective high-correlation problems

We see no interesting patterns when it comes to the correlation between objectives in the standard case. However, we do reproduce a result (and quite clearly so) previously found by [1], that for some reason the correlation between objectives has an enormous positive effect on the relative performance of NSGA-II relative to all other variants, when the number of objectives was high (at $m = 9$, specifically). This suggests it may be interesting to investigate the interplay between the correlation between objectives and the number of objectives in their influence on relative performance.

The following is a hypothesis for what may cause this behavior. For all algorithms we are using in this research, handling a high number of variables is a challenge. For NSGA-II however, the challenge is specifically that there is no real selection pressure because there are very few Pareto dominance relations (as the number of objectives increases, the probability that of two random solutions one Pareto dominates the other becomes progressively smaller). However, when we increase the correlation between objectives, this has the opposite effect. As the correlation between objectives increases, the probability that of two random solutions one Pareto dominates the other becomes progressively greater. Because MOEA/D doesn't make direct use of Pareto-dominance relations, this counteraction of the difficulty may still be there, but should be less strong.

To research whether this hypothesis is true, we want to find how the tendency for Pareto-dominance relations varies over various specifications of the correlation strength, and whether this can explain how the performance of NSGA-II varies with correlation strength.

We will measure both the hypervolume for a wide variety of correlation strengths between variables and the probability that out of randomly generated solutions one dominates the other, and measure correlations, as well as using a linear regression to see the significance of the latter when the former is controlled for. We will do the same experiment for MOEA/D.

The relationship between the correlation between variables (ρ) and the probability for a Pareto-dominance relation between randomly generated solutions was found to be 0.98 ($p = 3.4 * 10^{-73}$, measured using 100 solutions over $\rho = 0, 0.01, \ldots, 0.99$, $M = 9$, $K = 4$, $N = 20$). So the value of ρ seems to almost fully determine the probability for Pareto-dominance relations within this context.

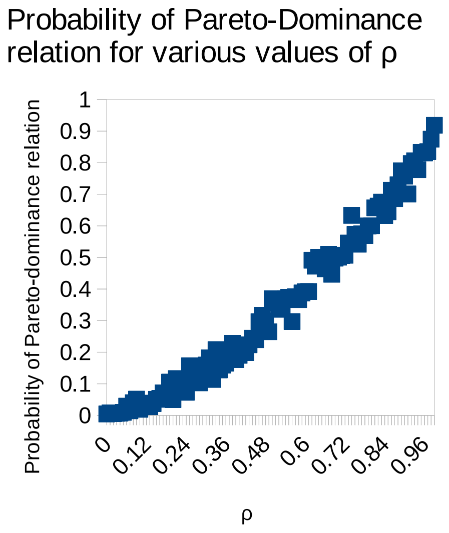
Using the same experiment, but now measuring the hypervolume that our algorithms reach after 30000 fitness evaluations, we measure the following correlations between the probability of Pareto-dominance relations and hypervolume:

Algorithm	Correlation	p-value
NSGA-II	0.72	$2.0 * 10^{-17}$
TCH	0.63	$3.2 * 10^{-12}$
PBI-5	0.70	$5.6 * 10^{-16}$
PBI-0.1	0.58	$1.7 * 10^{-10}$
WS	0.64	$5.7 * 10^{-13}$

We can further analyze the link between probability of Pareto-dominance relations and hypervolume by using a linear regression. This gives the following results:

Algorithm	Intercept	Slope	R^2
NSGA-II	0.032	0.091	0.52
TCH	0.045	0.068	0.39
PBI-5	0.023	0.083	0.49
PBI-0.1	0.044	0.062	0.34
WS	0.046	0.046	0.41

This gives us some quite interesting information which seems to confirm our hypotheses. We see that R^2 is the biggest for NSGA-II, which confirms our hypothesis that the effect of the probability for Pareto dominance relations is the biggest on NSGA-II. (It is however quite close to that of PBI-5, so it is perhaps more of an indication than conclusive evidence. Also an R^2 of 0.52 is not that high and tells us that though there is predictive power in this model, another model may describe the data even better). Furthermore we see that the slope is highest for NSGA-II, which indicates that for NSGA-II indeed the gain in hypervolume as the probability of Pareto-dominance increases is the highest.



Purely based on a visualization a reasonable hypothesis seems to be that the relation between the probability of Pareto-dominance and hypervolume for NSGA-II is actually that the probability of Pareto-dominance is proportional to the square root of the hypervolume. To see how reasonable this hypothesis is, we do the same analyses, but now using the square of the hypervolume rather than the hypervolume itself. What we find however, is that the strength of the correlation, the p-value of the correlation, and the R^2 of the linear regression all get worse.

Correlations:

Algorithm	Correlation	p-value
NSGA-II	0.57	$1.5 * 10^{-9}$
TCH	0.50	$1.6 * 10^{-7}$
PBI-5	0.50	$1.3 * 10^{-7}$
PBI-0.1	0.45	$2.5 * 10^{-6}$
WS	0.49	$2.8 * 10^{-7}$

Linear regression:

Algorithm	Intercept	Slope	R^2
NSGA-II	0.00058	0.013	0.31
TCH	0.0016	0.011	0.25
PBI-5	-0.00036	0.012	0.25
PBI-0.1	0.0015	0.010	0.20
WS	0.0017	0.011	0.24

Chapter 6

Experiments with Measured Attributes

In this research we have identified a number of potential factors that influence the relative performance of algorithms and we have seen that our attempts to vary the presence of these factors have indeed lead to various significant differences in relative performance.

However, these factors have been specified as parameters in problem generation or definition. It would perhaps be useful to not just look at parameters that can be specified, but also to measurable features of problems. In a real-life scenario, one may not know the shape of the Pareto front ahead of time, nor the uniformity of the way the solutions are distributed within the feasible part of the objective space.

Take the following scenario: We have an actual multi-objective optimization problem to calculate good solutions for a problem we encounter for an application in industry. Running the algorithm would take long and be expensive, so we would like to know ahead of time which algorithm can be expected to be the best performer. We could take the factors hitherto investigated, but how would we actually use those? We know for which values of K within a ρMNK-problem you would rather use NSGA-II instead of MOEA/D, but this real world problem perhaps isn't easily compared to ρMNK-problems. We could make an educated guess, but not much more than that.

To gain **predictive power**, it would be good to not limit ourselves to specified parameters, but to also use **measured attributes** that don't use information about how the problem is constructed. Of course it also comes at a cost. Specified parameters are known exactly, while the same can of course not necessarily be said of measured attributes (being measured based on a sample which may or may not be representative). This may severely limit the predictive power of these attributes, which is exactly contrary to what we were trying to achieve.

One example of a measurable attribute is the **correlation between the**

fitness of parents and their offspring, under both crossover and mutation.

Another measurable attribute is the tendency for Pareto-dominance, just like in the previous chapters. What is **the probability for there to be a Pareto-dominance relationship between two randomly generated solutions**?

We have also seen that the **shape of the objective space** of admissible solutions can matter. It's hard to know exactly what this shape looks like from a sample, but we can try to gauge it in the following way: We generate weight vectors like we do for MOEA/D, using $H = 10$. Then we attribute solutions to the vectors based according to which vector gives it the lowest d_2 term as calculated for the scalarizing function PBI. We then take a measure similar to the Gini-impurity of this distribution. Let W be the set of the counters for the vectors in no particular order and $|X|$ the population size, then this impurity measure is defined as: $1 - \sum_{w \in W} (\frac{w}{|X|})^2$. (The Gini impurity is defined as $i_{Gini}(x) = \sum_{i=1}^{M} \frac{x_i}{||x||_1}(1 - \frac{x_i}{||x||_1})$, where x is a vector[31].)

Lastly, of course we can simply measure the **correlation between the various objectives** on our randomly generated sample.

We will run 250 random ρMNK-problems, including our extra u and s settings. We will have $M = 2$, $N = 20$, $K \in \{0, \dots, 8\}$, $\rho \in [0, 0.9]$. Because a slight amount of u or s can already mean a lot of distortion (no matter the specific value of s, if it is greater than zero it splits the feasible objective space into two distinct components), instead of varying them randomly over a wide range of values, u is randomly chosen to be either 1 or 0.5 and s is randomly chosen to be either 0 or 0.5.

We will analyze the results in two ways. Firstly we will look at whether these measured statistics have any information by looking at the strength and p-value of the correlation between any statistic and any algorithm's fitness, and secondly we will use all statistics together to, separately for every algorithm, run a linear regression to see the predictive capabilities in the form of the R^2 value.

All p values were smaller than 10^{-4}, so these have been left out. The following table contains correlation strengths except for the last row.

Statistic	NSGA-II	WS	TCH	PBI-5	PBI-0.1
Objective correlation	0.51	0.52	0.52	0.51	0.54
Impurity	0.40	0.38	0.38	0.39	0.41
Mutation fitness correlation	0.26	0.27	0.29	0.28	0.25
Crossover fitness correlation	0.29	0.27	0.28	0.27	0.26
Pareto domination probability	0.48	0.48	0.48	0.47	0.50
Linear regression R^2	0.36	0.37	0.37	0.36	0.39

We see that the correlation strength is somewhat promising, going as high as 0.54. This shows that measured statistics can (depending on the statistic as they also went as low as 0.25) say something about the expected result of an algorithm. Sadly the factors do not combine into a particularly good predictor, though a R^2 of around 0.4 isn't terrible.

It could very well be that the problem is simply that the relations between the statistics and the fitness are not linear. Perhaps by using a slightly more advanced model we could do much better. If anything, this research shows promise, but is far from conclusive.

Chapter 7

Conclusion & Discussion

7.1 Experiments with specified attributes

Experiments showed that as variables become less separable, NSGA-II starts to outperform MOEA/D. It seems like this can be partially explained by virtue of the lower fitness correlation, as increasing the neighborhood size to include the entire population at all times does away with some of the effect. The results also suggest that even when the separability of variables becomes extreme, there is still a structure present in the problem which any algorithm may or may not be more apt to exploit. Furthermore, experiments showed NSGA-II outperforming MOEA/D as the uniformity of the distribution of solutions decreased.

Experiments also showed that that MOEA/D tends to outperform NSGA-II when the number of objectives increases.

Along several of the experiments, the specific scalarizing function chosen for MOEA/D was significant, and even the choice of parameter within it. For example, in our experiments where we varied the number of dimensions, PBI-5 got as low as 54% of the performance of PBI-0.1.

Furthermore, we find that under a high number (9, in this document) of objectives, as the correlation between objectives increases, NSGA-II starts outperforming MOEA/D which is an effect previously observed in the context of multi-objective knapsack problems.

Surprisingly, we did not find a significant effect to changing the shape of the feasible part of our objective space (and by implication the Pareto front). It may be that the shape only really becomes a problem in combination with other factors, or perhaps our problem was too easy. Higher dimensions offer the possibility for more complex shapes of which it may be interesting to research whether the shape can be the determining factor in the relative performance of algorithms.

Factor	NSGA-II	WS	TCH	PBI-5	PBI-0.1
Inseparability	+	-	-	-	-
Number of objectives	- -	-	-	- -	++
Correlation between objectives	.	.	.	.	.
. . . at high number of objectives (9)	++	-	- -	- -	-
Shape	.	.	.	.	.
Uniformity	++	- -	- -	- -	- -

Table 7.1: Our interpretation of results: As the factor increases, what the algorithm tends toward; Legend: (++) for "clear best choice", (+) for "arguably best choice", (.) for "doesn't matter or unclear", "(-) for "arguably inferior choice", (- -) for "clear inferior choice".

7.2 Many-objective high-correlation problems

Based on what we found in our experiments with specified attributes we did some extra work in investigating the cause of the behavior that under a high number of objectives a higher correlation between them makes NSGA-II work better.

We find some interesting results, showing the correlation between the probability of Pareto-dominance relations and hypervolume can be as high as 0.72 and is at it's highest for NSGA-II. We find that this probability has an explanatory power for the hypervolume, namely $R^2 = 0.52$ under a linear regression while having both the highest R^2 and the highest slope (with the probability of Pareto-dominance being the independent variable).

This tells us that the probability of Pareto-dominance relations, which is almost fully determined by the correlation between variables ρ (as the correlation between ρ and the probability for Pareto-dominance relations is 0.98 with $p < 10^{-72}$) has a significant influence on the hypervolume the varying algorithms achieve, and favors NSGA-II. However, with an $R^2 = 0.52$ it would also be too strong a claim to say that the effect has been totally explained away. It is hard to say to what extent we have explained the effect, as there is of course a random element in these algorithms and problems and thus an extremely high R^2 might simply be impossible.

7.3 Experiments with Measured Attributes

We have performed a number of experiments where we measured a variety of attributes on a population sample. We found that these attributes have some explanatory power, though a small amount, with $R^2 = 0.36$ in a linear regression. We also found that correlations between these attributes and hypervolume were not very high, as they varied between 0.26 and 0.51, but they were all significant with $p < 10^{-4}$.

Our work in this domain was small and exploratory, as it was not the main subject of this thesis, but it shows some promise to be further investigated.

7.4 Future work

As the factors we chose for our experiments on ρMNK and circle-quadrant problems were somewhat arbitrary, one could always expand on the list of factors whose effect could be researched.

For the most part our research was satisfactory. However, when it comes to the matching of the shape of the Pareto front and the distribution of the weight factors, the effects were not at all as strong as expected. All works are necessarily limited in scope and we have not chosen to pursue more complex, higher-dimensional shapes, but this may very well be interesting for future research. In the somewhat simple problems we explored the factor of shape did not seem to have much of an effect, even though the literature suggests that it does. The obvious questions that remains are: When does it and when does it not matter? And how much?

On the topic of researching the cause of the effect that under a high number of objectives, increasing the correlation between objectives favors NSGA-II, we did find some results that seem to explain the phenomenon. However, our predictive model only had an R^2 of 0.52 and the difference in slope was not that big between the various algorithms. Our research strongly suggests an explanation, but some other methods (perhaps a different model than linear regression) may be required to make our explanation even more convincing.

When it comes to experiments with measured attributes, we showed that finding significant correlations and models with some predictive power is possible. However, the results were quite limited. Because this was not the main subject of this thesis we have not done a lot of work in further exploring the possibilities of predicting performance using measured attributes, but it could very well lead to interesting results to try other measures or predictive models. This may be especially interesting because it would have a clear real-world application if good results could be produced, namely to make the choice of optimal algorithm for a real-world problem easier.

Bibliography

[1] H. Ishibuchi, N. Akedo, Y. Nojima, Behavior of Multiobjective Evolutionary Algorithms on Many-Objective Knapsack Problems, IEEE Transactions on Evolutionary Computation (2015)

[2] H. Ishibuchi, N. Akedo, H. Ohyanagi, Y. Nojima, Behavior of EMO Algorithms on Many-Objective Optimization Problems with Correlated Objectives, Proc. of 2011 IEEE Congress on Evolutionary Computation (2011)

[3] H. Ishibuchi, Y. Setoguchi, H. Masuda, Y. Nojima, Performance of Decomposition-Based Many-Objective Algorithms Strongly Depends on Pareto Front Shapes, IEEE Transactions on Evolutionary Computation, Vol 21, No. 2 (2017)

[4] Q. Zhang, H. Li, MOEA/D: A multiobjective evolutionary algorithm based on decomposition, IEEE Transactions on Evolutionary Computation (2007)

[5] Q. Zhang, W. Liu, H. Li, The Performance of a New Version of MOEA/D on CEC09 Unconstrained MOP Test Instances, 2009 IEEE Congress on Evolutionary Computation (2009)

[6] Q. Zhang, W. Liu, E. Tsang, B. Virginas, Expensive Multiobjective Optimization by MOEA/D with Gaussian Process Model, IEEE Transactions on Evolutionary Computatation, Vol 14. Issue 3 (2010)

[7] H. Li, Q. Zhang, Multiobjective Optimization Problems With Complicated Pareto Sets, MOEA/D and NSGA-II , IEEE Transactions on Evolutionary Computation, Vol 13, No 2. (2009)

[8] M. Li, S. Yang, X. Liu, R. Shen, A Comparative Study on Evolutionary Algorithms for Many-Objective Optimization, International Conference on Evolutionary Multi-Criterion Optimization (2013)

[9] A Zhou, B. Qu, H. Li, S. Zhao, P.N. Sugunthan, Q. Zhang, Multiobjective evolutionary algorithms: A survey of the state of the art, Swarm and Evolutionary Computation 1 (2011) (Elsevier)

[10] K. Deb, A. Pratap, S. Agarwal, T. Meyarivan, A Fast and Elitist Multiobjective Genetic Algorithm: NSGA-II, IEEE Transactions on Evolutionary Computation, Vol 6, No. 2 (2002)

[11] K. Deb, H. Jain, An Evolutionary Many-Objective Optimization Algorithm Using Reference-Point-Based Nondominated Sorting Approach, Part I: Solving Problems With Box Constraints, IEEE Transactions on Evolutionary Computation, Vol 18, No. 4 (2014)

[12] K. Deb, H. Jain, An Evolutionary Many-Objective Optimization Algorithm Using Reference-Point Based Nondominated Sorting Approach, Part II: Handling Constraints and Extending to an Adaptive Approach, IEEE Transactions on Evolutionary Computation, Vol 18, No. 4 (2014)

[13] K. Deb, L. Thiele, M. Laumanns, E. Zitzler, Scalable Test Problems for Evolutionary Multi-Objective Optimization, A. Abraham, L. Jain, R. Golberg (eds) Evolutionary Multiobjective Optimization, Advanced Information and Knowledge Processing, Springer, London (2005)

[14] K. Deb, R.B. Agrawal, Simulated Binary Crossover for Continuous Search Space, Complex Systems 9, (2000)

[15] K. Deb, D. Deb, Analyzing Mutation Schemes for Real-Parameter Genetic Algorithms, International Journal of Artificial Intelligence and Soft Computing (2014)

[16] K. Deb, H. Beyer, Self-Adaptive Genetic Algorithms with Simulated Binary Crossover, Evolutionary computation. 9. 197-221, (2001)

[17] S. Huband, P. Hingston, A Review of Multi-objective Test Problems and a Scalable Test Problem Toolkit, IEEE Transactions on Evolutionary Computation Vol 10, No. 5 (2006)

[18] S. Huband, L. Barone, L. While, P. Hingston, A Scalable Multi-objective Test Problem Toolkit, A Scalable Multi-objective Test Problem Toolkit. ECU Publications. 3410. 280-295 (2005)

[19] S. Verel, A. Liefooghe, L. Jourdan, C. Dhaenens, On the Structure of Multiobjective Combinatioral Search Space: MNK-Landscapes with Correlated Objectives, European Journal of Operational Research 227 (2013)

[20] S. Verel, A. Liefooghe, L. Jourdan, C. Dhaenens, Analyzing the Effect of Objective Correlation on the Efficient Set of MNK-Landscapes, Learning and Intelligent Optimization Conference (LION 5) (2011)

[21] Y. Qi, J. Sun, J. Wu, MOEA/D with adaptive weight adjustment, Evolutionary Computation (2013)

[22] L. While, L. Bradstreet, L. Barone, A Fast Way of Calculating Exact Hypervolumes, IEEE Transactions on Evolutionary Computation, Vol 16, No. 4 (2012)

[23] E. Zitzler, L. Thiele, M. Laumanns, C.M. Fonseca, V.G. da Fonseca, Performance Assessment of Multiobjective Optimizers: An Analysis and Review, IEEE Transactions on Evolutionary Computation, Vol 7, No. 2 (2003)

[24] W. Peng, Q. Zhang, H. Li, Comparison between MOEA/D and NSGA-II on the multi-objective travelling salesman problem, In: CK. Goh, YS. Ong, K.C. Tan (eds) Multi-Objective Memetic Algorithms. Studies in Computational Intelligence, vol 171. Springer (2009)

[25] C.A. Coello Coello, G.B. Lamont, D.A. Van Veldhuizen, Evolutionary Algorithms for Solving Multi-Objective Problems, 2nd edition, Springer Science+Business Media, LLC (2007)

[26] P Chang, Q. Zhang, S. Chen, J. Lin, MOEA/D for flowshop scheduling problems, 2008 IEEE Congress on Evolutionary Computation (2008)

[27] L. Ke, Q. Zhang, R. Battiti, MOEA/D-ACO: A Multiobjective Evolutionary Algorithm Using Decomposition and Ant Colony, IEEE Transactions on Cybernetics, Vol. 43, No. 6 (2013)

[28] K. Frankish, W.M. Ramsey, S. Franklin, K. Arkoudas, S. Bringsjord, W.S. Robinson, M.A. Boden. R. Sun, R.D. Beer, D. Danks, M. Vincze, S. Wachsmuth, G. Sagerer, E. Amir, Y. Wilks, E. Alonso, M. Scheutz, P. Husbands, M. Bedau, N. Bostrom, E. Yudkowsky, The Cambridge Handbook of Artificial Intelligence, Cambridge University Press (2014)

[29] D.E. Moriarty, A.C. Schultz, J.J. Grefenstette, Evolutionary Algorithms for Reinforcement Learning, Journal of Artificial Intelligence Research 11 (1999)

[30] J. Alcalá-Fez, L. Sánchez, S. Garcia, M.J. del Jesus, S. Ventura, J.M. Garell, J. Otero, C. Romero, J. Bacardit, V.M. Rivas, J.C. Fernández, F. Herrera, Keel: A Software Tool to Assess Evolutionary Algorithms for Data Mining Problems, Soft Computing 13(3), Springer Verlag (2009)

[31] E.S. Laber, M. Molinaro, F de A.M. Pereira, Binary Partitions with Approximate Minimum Impurity, Proceedings of the 35th International conference on Machine Learning (2018)