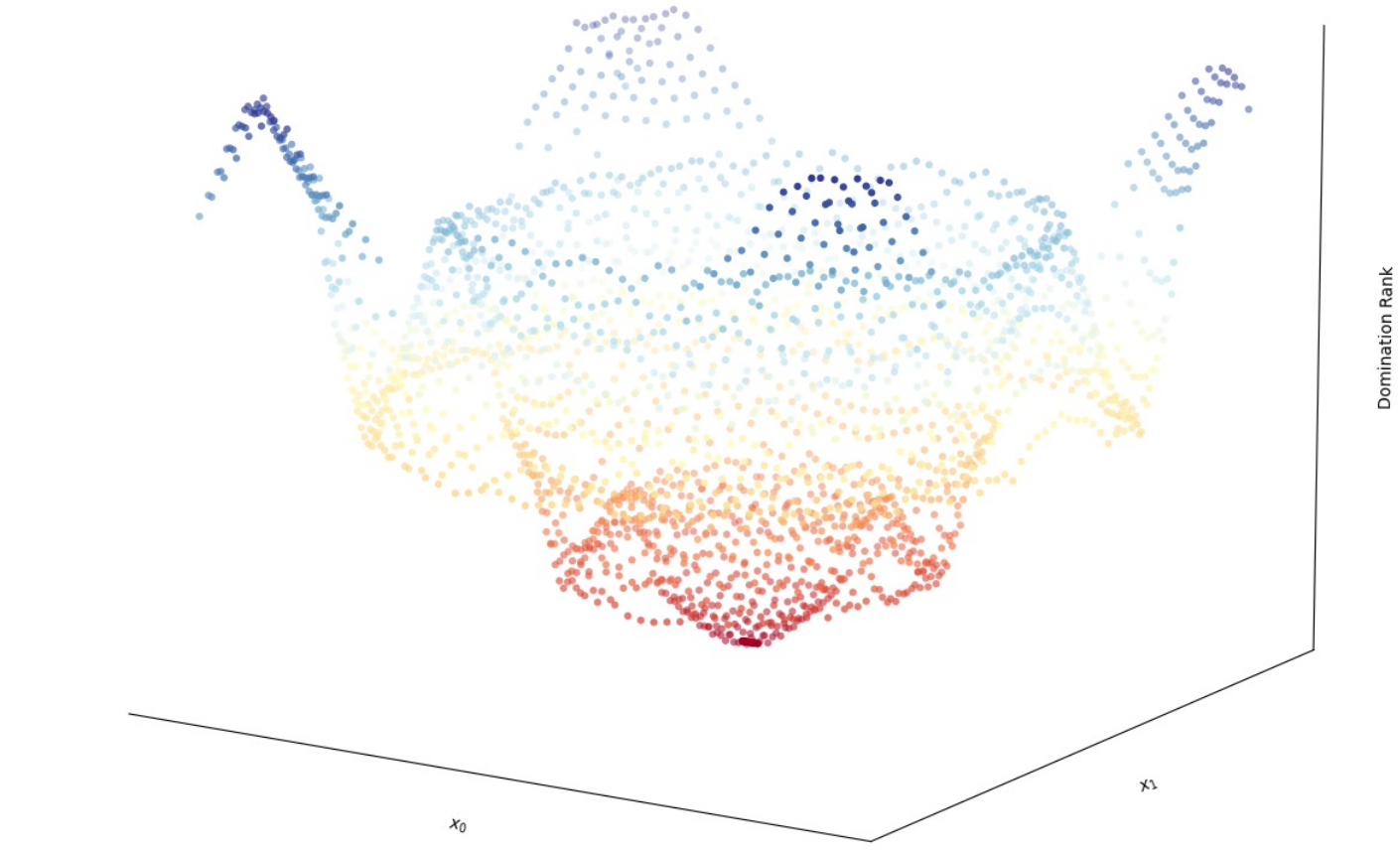
Hybridizing Hypervolume based Evolutionary Algorithms and Gradient Descent by Dynamic Resource Allocation

D. Ha



TU Delft

Hybridizing Hypervolume based Evolutionary Algorithms and Gradient Descent by Dynamic Resource Allocation

by

D. Ha

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on Friday April 29, 2022 at 17:00 AM.

Student number:	4367480	
Project duration:	December 10, 2021 – April 29, 2022	
Thesis committee:	Prof. dr. P. Bosman,	TU Delft, supervisor
	Dr. S. Picek,	TU Delft
	Dr. T. Deist,	External Expert

An electronic version of this thesis is available at http://repository.tudelft.nl/.

TU Delft

Preface

This documents describes the master thesis of Damy Ha, which was done under daily supervision of Dr. Timo Deist and overall supervision of Prof. Dr. Peter A.N. Bosman. This work has been submitted to the seventeenth International Conference on Parallel Problem Solving from Nature (PPSN XVII) in the form of a conference paper. Support was granted in the form of an internship by Centrum Wiskunde & Informatica (CWI), which is financed by NWO.

In this document we look at different algorithms to solve real-valued multi-objective problems. Specifically, we look at an uncrowded hypervolume-based (UHV) evolutionary algorithm (EA) and an UHV-based gradient algorithm. EAs are well-known to be highly suited for multi-objective (MO) optimization. Recently however, a gradient-based UHV algorithm, known as UHV-ADAM, was shown to be more efficient than (UHV-based) EAs if few local optima are present. If many local optima are present, EAs generally remain more efficient. Combining the two techniques could exploit synergies, especially if problems have many local optima, i.e., the EA could be leveraged to avoid local optima while the efficiency of gradient algorithms could speed up convergence to the Pareto set. It is a priori however not clear what would be the best way to make such a combination. In this work, therefore, we study the use of a dynamic resource allocation scheme to create hybrid UHV-based algorithms. On several bi-objective benchmarks, we find that the hybrid algorithms produce similar or better results than the EA or gradient-based algorithm alone, even when finite differences are used to approximate gradients. The implementation of the hybrid algorithm is available at https://github.com/damyha/uncrowded-hypervolume.

The members of the thesis committee are: Prof. dr. P. A.N. Bosman, Dr. S. Picek and Dr. T. Deist.

D. Ha
Delft, August 2021

Contents

1 Introduction

In real-valued single objective optimization, a commonly used and well-studied type of algorithms are gradient algorithms. Gradients indicate in which direction the most improvement can be obtained locally. By iteratively following the gradients, an optimum can be efficiently found. This optimum however, can turn out to be a local optimum, i.e. an optimum that is not globally optimal. Usually however, the global optimum is what is desired. Evolutionary algorithms (EAs) are another class of optimization algorithms. EAs, in general, are known for their control mechanisms in avoiding local optima. A key mechanism for this lies in that multiple solutions are kept in a population. Search space information is iteratively inferred from the population and used to guide the search towards an optimum.

In this work, we focus on real-valued multi-objective (MO) optimization, where multiple objectives need to be optimized. Usually however, the objectives contradict each other, resulting in that there is not a single solution that is optimal for all objectives. As a consequence, MO-optimization algorithms usually try to find a diverse set of Pareto optimal solutions, usually as efficiently as possible.

Evolutionary algorithms (EAs)(e.g. [1], [2], [3]) are well-known to be highly suited for MO optimization [4]. However, in real-valued MO optimization, classic domination-based approaches (see section 2.2.1) lose selection pressure when approaching the Pareto set [5], [6]. Indicator-based approaches, such as optimizing the hypervolume (HV) [7] or the extended uncrowded hypervolume (UHV) [8], [9] can overcome this issue and ensure that individual solutions converge to the Pareto set. Recently, a gradient-based UHV algorithm known as UHV-ADAM [10] was shown to be more efficient than (UHV-based) EAs if few local optima are present. If many local optima are present however, EAs generally remain more efficient. Combining the two techniques could exploit synergies, especially on problems with many local optima, i.e., the EA could be leveraged to avoid local optima while the gradient algorithms could be leveraged to efficiently converge to the Pareto set. It is however unknown a priori, how the techniques should be combined to get the best results. Attempts in the literature however have been successful at creating efficient MO hybrid algorithms.

In [3] a hybrid algorithm was proposed that probabilistically executes different variation operators of EAs. Gradient algorithms however have not been integrated into their work. In [11] a domination-based EA was combined with gradient-based algorithms that exploit either the gradient of a single objective or a combination thereof that corresponds to maximum improvement in a multi-objective sense. Furthermore, resources are dynamically assigned to the gradient algorithms via a resource allocation scheme (RAS). A HV-based hybrid algorithm was introduced in [12], which combines both an EA and gradient algorithm that aim to maximize the HV. In contrast to [11] however, [12] executes the gradient algorithm after the EA is finished. Supplementing the EA during evolution however might be of key value.

In this work, we study the potential of unifying the convergence properties of UHV-based MO algorithms with a hybrid interleaving optimization scheme. Specifically, we formulate a new UHV-based hybrid algorithm and show that the hybrid algorithm is capable of performing better than the worst of the original algorithms or in some cases better than both algorithms. For this, we combine a UHV-based EA called UHV-GOMEA [9] with UHV-ADAM [10] via an extended RAS of [11]. The resulting hybrid algorithm is consequently UHV-based. The UHV distinguishes itself in that a set of solutions is optimized instead of individual solutions. Concretely, this means that the UHV-based hybrid (and EA) employ populations of solution sets, not individual solutions, where each individual solution set

is optimized towards the Pareto set. In this work, we furthermore empirically determine the hybrid's architecture using a similar set of benchmarks as in [9] and [10]. We then compare the final algorithm with its component algorithms, UHV-GOMEA and UHV-ADAM, and other UHV-based algorithms on the Walking Fish Group (WFG) benchmark set. This report starts off with a literature study in chapter 2. Chapter 2 contains basic information on real-valued MO problems and lays the foundation of the research questions. The hybridization method is then described in chapter 3. This chapter explains the mechanics of the algorithms that are used for the hybridization, the modification done to those algorithms such that hybridization is achievable and ends with the hybridization scheme. The experiments to investigate the architecture of the hybrid are then shown in chapter 4. After the experiments, the discussion can be found in chapter 5 followed by the conclusion in chapter 6.

2

Background

This chapter introduces the terminology used in this work to describe real-valued multi-objective (MO) optimization. It starts off by defining an MO optimization problem, followed by a brief summary of existing MO algorithms. Finally, a research question and its sub questions are defined.

2.1. Multi-objective optimization

In real-valued single objective (SO) optimization, a single objective needs to be optimized. Solutions are iteratively selected from a search space and are evaluated using the objective function until a termination condition is satisfied, e.g. when a solution is found that satisfies a desired objective value or when a computation budget has been depleted. Multi-objective (MO) optimization, differs from SO optimization in that multiple objectives need to be optimized simultaneously. Generally, the objectives contradict each other. As a consequence, a single solution usually does not exists that is optimal in all objectives. Instead, a set of Pareto optimal solutions, called the Pareto set, exists. These Pareto optimal solutions are optimal in that other solutions can not be found that are better in all objectives. Only other Pareto optimal trade offs can be found. The goal of MO optimization is usually to find a diverse and representative subset of this Pareto set. This set of solutions can then be given to a decision maker that picks one of the trade off solutions.

2.1.1. Defining a multi-objective problem

In this work, an SO optimization problem is defined as a single function $f : \mathcal{X} \to \mathbb{R}$ which needs to be minimized, where $\mathcal{X} \subseteq \mathbb{R}^n$ is an n-dimensional search space. In an MO optimization problem, m objective functions need to be minimized. Let $\mathbf{f} : \mathcal{X} \to \mathbb{R}^m$, with $\mathbf{f} = [f_0, ..., f_{m-1}]$ be an m-dimensional vector of objective functions. In this work, we only focus on bi-objective problems ($m = 2$). The values that vector $\mathbf{f}$ can take is called the objective space. A solution $\mathbf{x}$ of search space $\mathcal{X}$, where $\mathbf{x} = [x_0, ..., x_{n-1}]$, will be called an MO-solution. Assessing an MO-solution $\mathbf{x}$ with $\mathbf{f}$ maps the solution from the search space to the objective space. To asses the quality of an MO-solution, Pareto optimality is often used. The terminology associated with Pareto optimality will be listed below. Figure 2.1 furthermore illustrates the important concepts of Pareto optimality relevant to this work, for an arbitrary bi-objective function in a 2D search space.

- `Weak dominance`: A solution $\mathbf{x_0} \in \mathcal{X}$ is said to weakly dominate solution $\mathbf{x_1} \in \mathcal{X}$, written as $\mathbf{x_0} \preceq \mathbf{x_1}$ if and only if $f_i(\mathbf{x_o}) \leq f_i(\mathbf{x_1}), \forall i \in \{1, ..., m\}$
- `Dominance`: A solution $\mathbf{x_0}$ is said to dominate solution $\mathbf{x_1}$, written as $\mathbf{x_0} \prec \mathbf{x_1}$ when it is weakly dominated and $\exists i \in \{1, ..., m\} : f_i(\mathbf{x_o}) < f_i(\mathbf{x_1})$.
- `Pareto optimal solution`: A solution is Pareto optimal when it is not dominated by any solution in $\mathcal{X}$.
- `Pareto set`: Pareto set $\mathcal{A}^\star$ is the set of all Pareto optimal solutions: $\mathcal{A}^\star = \{\mathbf{x_0} \in \mathcal{X} : \nexists \mathbf{x_1} \in \mathcal{X} : \mathbf{x_1} \prec \mathbf{x_0}\}$.

- `Pareto front`: The Pareto front is the vectors of objective values obtained by applying the objective functions on all Pareto optimal solutions: $\{\mathbf{f}(\mathbf{x}) : \mathbf{x} \in \mathcal{A}^\star\}$.

Using Pareto optimality, the goal of MO-algorithms is usually to find a set $\mathbb{S} \subset \mathcal{X}$ of a manageable number of (near-)Pareto-optimal MO-solutions, preferably also as efficiently as possible.

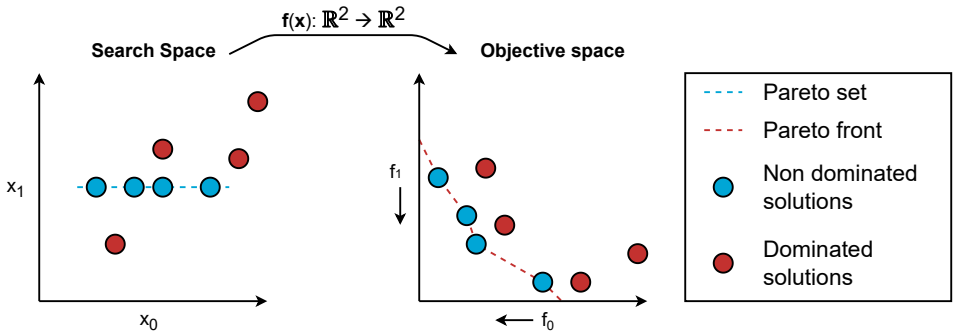


Figure 2.1: Pareto optimality shown for an arbitrary bi-objective optimization problem in a 2D search space. The search space and objective space are displayed on the left and right respectively.

2.1.2. Multi modal problems

Depending on which objective functions a MO-optimization problem contains, different objective spaces can be constructed. In this work, two types of problems are examined. Uni-modal problems are defined as problems where the objective functions are uni-modal, i.e. the functions only have one global optima and no other local optima exists. Multi-modal problems in this work, will be referred to as problems where at least one objective function has local optima. Multi-modal problems differ from uni-modal problems, in that MO-optimization algorithms can get stuck in an local optimum. This possibly prevents an MO-algorithm from fully converging to the Pareto set.

2.2. Approaches to solve multi-objective problems

In order to solve multi-objective problems, in this work we will look at two types of algorithms, namely: evolutionary algorithms (EAs) and gradient algorithms. We will also look at two different approaches, namely: domination-based and indicator-based.

2.2.1. Domination-based evolutionary approach on multi-objective problems

Evolutionary algorithms (EAs) have shown to be effective at solving real-valued MO-optimization problems [4]. EAs, which use natural evolution as a metaphor for constructing randomized search algorithms, are heuristic algorithms. They usually maintain a collection of solutions, referred to as a “population“, and use the population to infer information from the search space. In general, an EA follows the cyclic pattern displayed in figure 2.2. The EA starts off by initializing a population of solutions in the search space. From the population, parents are selected to produce offspring. The offspring are evaluated, and depending on the selection strategy, used to replace members of the old population. The process repeats until a termination condition is satisfied, e.g. when the computational budget has expended or when a solution has been found that satisfies certain criteria. Determining which parents are selected for producing offspring is an important design aspect of an EA. Premature members of a population can make the gene pool too homogeneous, resulting in convergence to local optima. Choosing less fit parents decreases the probability of ending up in local optima but usually also slows down convergence. How offspring is created from parent solutions is an another important design decision.

Well known real-valued MO EAs (e.g. [1], [13]) are domination-based. Domination-based EAs employ domination-based fitness selection schemes to push the individual MO-solutions to the Pareto set. Usually, if a solution is found that dominates a solution of the population, the dominated solution is replaced by this newly found non-dominated solution. Domination-based strategies however, often lose selection pressure when approaching the Pareto set [5], [6]. When the population size is much smaller than the Pareto set, at some point in time, the population of an EA is likely to only consist of non-dominated solutions. From this population of non-dominated solutions and the non-dominated

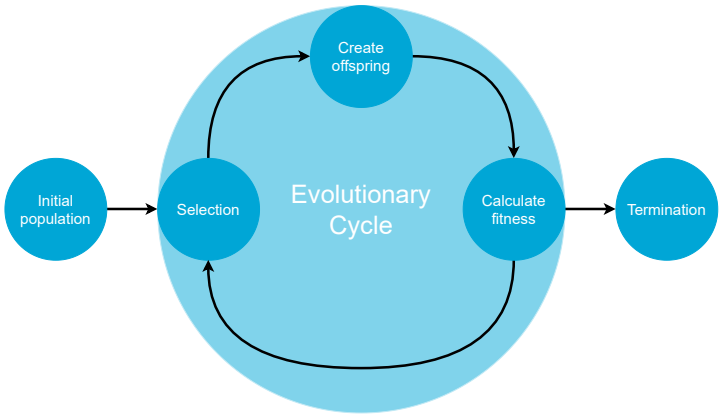


Figure 2.2: General evolutionary cycle of an EA

offspring solutions that it creates, solutions must be selected for the population of the next generation. All non-dominated solutions are considered equally good in terms of Pareto optimality. As such, it is not clear which non-dominated solutions must be brought over to the next population and which must be discarded. This results in the loss of selection pressure. Losing selection pressure not only causes the selection to turn into a random selection, but also causes the search to turn into a random search. Discarding non-dominated solutions furthermore causes loss of information of the search space. Loss of information is detrimental to the search, as solutions can deteriorate in terms of quality as the search progresses.

Figure 2.3 illustrates an example in objective space where solutions deteriorate in terms of Pareto efficiency. In this example, a search process is initiated in generation 0, where the population size is 4. In generation 1, offspring is produced and 4 solutions must be selected for the next population. In this example, 2 parents consistently generate dominated solutions that are discarded, causing the parent solutions to always be selected for the next generation. We are however, more interested in the other 2 non-dominated parent and non-dominated offspring pairs in “Situation A“ and “Situation B“. To fill up the 2 remaining spots of the next generation’s population, we are free to pick any combination of the non-dominated solutions. We decide, for an arbitrary reason, to pick the non-dominated offspring solution in situation A, and the non-dominated parent solution in situation B. In generation 2, we once more select 2 solutions to our liking. In situation A, we once again choose the non-dominated offspring solution. In situation A however, it is not evident that this decision leads to the loss of quality. The offspring’s grand parent solution (of generation 0) dominates the current offspring solution, however it was earlier discarded. In situation B, we choose the offspring solution, as it clearly dominates the parent solution, and thus leads to an improvement of the front. These two situations lead to a simultaneous front degradation and front improvement. In practice, this effect occurs regularly once most solutions of a population are non-dominated, and effectively prevents the solutions from fully converging to the Pareto set.

2.2.2. UHV-based approach on multi-objective problems

Indicator-based approaches, which for example optimize the hypervolume (HV) [14] or uncrowded hypervolume (UHV) [9], have shown promising results to overcome the stagnation that occurs in domination-based approaches and ensures that individual solutions converge to the Pareto set. Indicator-based MO algorithms use indicator functions to asses and assign a numeric value to a solution or more commonly a set of limited number of solutions p, as is the case with the HV and UHV indicator functions. Assessing a set of solutions allows these indicator functions to combine proximity and diversity into a numeric value, effectively redefining an MO optimization problem into an SO optimization problem. In this work, we focus on UHV-based MO algorithms. The UHV is based on the well-known HV indicator function, but additionally applies pressure on dominated solutions to improve.

The UHV, which is calculated over a set $\mathbb{S}$ consisting of p solutions, measures the HV, i.e. the area in objective space enclosed by the non-dominated solutions of $\mathbb{S}$ and a predetermined reference point $\mathbf{r} = (r_0, r_1)$, and punishes the dominated solutions of $\mathbb{S}$ via the uncrowded distance (ud) as displayed in equation 2.1. To calculate the HV, let $\mathcal{A}$ be the approximation set that contains all non-dominated

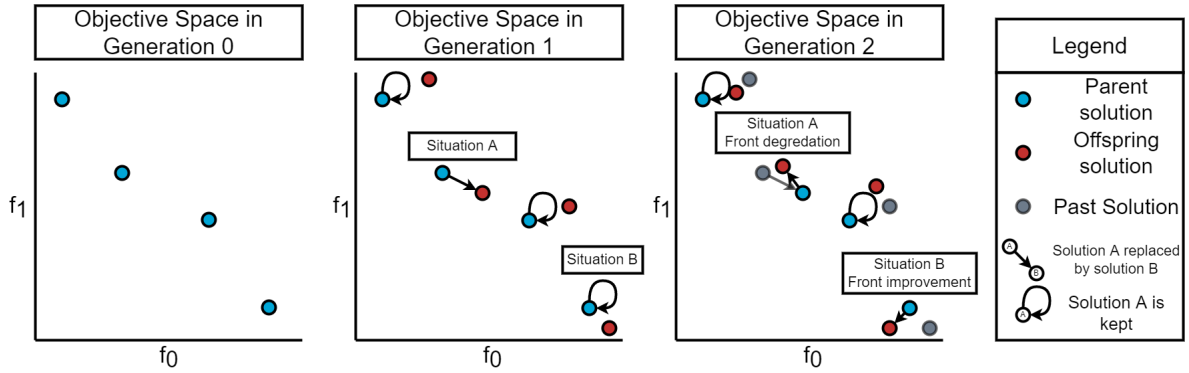


Figure 2.3: An example of domination-based EAs and the consequences of losing selection pressure for a population size of 4. In each generation, 4 solutions must be selected for the next generation's population. In situation A and B, front degradation and front improvement are observed over time.

solutions of $\mathbb{S}$. $\mathcal{A}$ then forms an approximation boundary $\partial\mathbf{f}(\mathbb{S})$ in objective space, as is shown in figure 2.4. The HV is the region encapsulated between approximation boundary $\partial\mathbf{f}(\mathbb{S})$ and reference point $\mathbf{r}$. The aforementioned uncrowded distance $\text{ud}(\mathbf{x}, \mathbb{S})$ is the closest Euclidean distance between MO-solution $\mathbf{x}$'s objective values $\mathbf{f}(\mathbf{x})$ and the approximation boundary $\partial\mathbf{f}(\mathbb{S})$. By definition, $\text{ud}(\mathbf{x}, \mathbb{S})$ is zero for a non-dominated solution. In equation 2.1, the uncrowded distances are taken to the power of m (number of objectives), to get to the same unit as the HV [9]. The sum of uncrowded distances is furthermore scaled by $\frac{1}{|\mathbb{S}|}$ to reduce the undesired effect of increasing the uncrowded distances when the HV improves.

The goal of UHV-based algorithms is to maximize the UHV, as maximization leads directly to the minimization of the original objective functions as well improving the diversity [15].

$$\text{UHV}(\mathbb{S}) = \text{HV}(\mathbb{S}) - \frac{1}{|\mathbb{S}|}\sum_{\mathbf{x}\in\mathbb{S}} \text{ud}(\mathbf{x}, \mathbb{S})^m \tag{2.1}$$

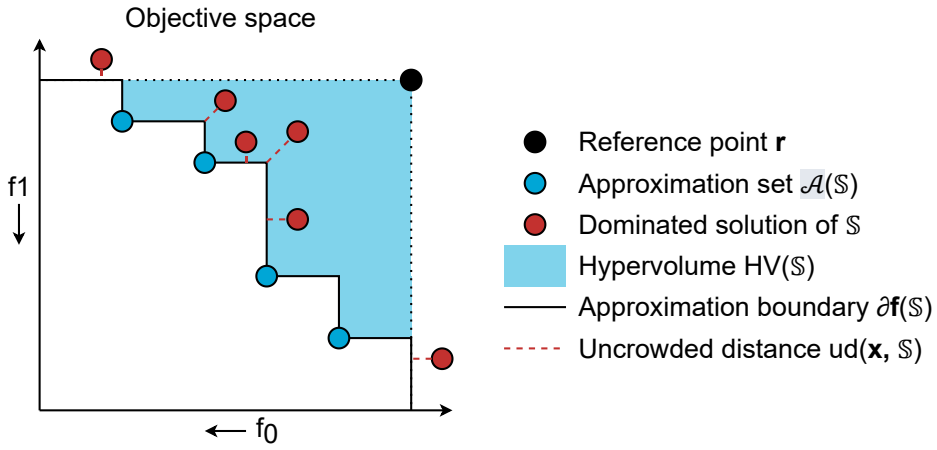


Figure 2.4: Illustration of the uncrowded hypervolume in an arbitrary bi-objective minimization problem.

2.2.3. UHV-based hybrid approach on multi-objective problems

In this work, we select two recently introduced UHV-based algorithms as candidates for hybridization, namely: evolutionary algorithm UHV-GOMEA [9] and gradient algorithm UHV-ADAM [10]. UHV-GOMEA has been selected, as in [9] it was shown that UHV-GOMEA is able to overcome stagnation, which domination-based EAs suffer from. Furthermore, UHV-GOMEA is part of the Gene-pool Optimal Mixing (GOM) family of algorithms, which have shown great success in various optimization problems.

UHV-ADAM has been selected as gradient algorithm, not only because it is also UHV-based and thus limits the potential number of compatibility issues between UHV-GOMEA and UHV-ADAM, but also because it has shown superior performance with respect to UHV-GOMEA in [10], when MO problems have few local optima and, especially when the gradients are analytically calculated at no additional cost.

We will combine both algorithms by extending the resource allocation scheme (RAS) of [11]. The resulting hybrid algorithm is consequently UHV-based. The UHV is used to optimize a set of solutions towards the Pareto set, instead of individual solutions. Concretely, this means that the UHV-based hybrid that will be constructed and UHV-GOMEA both employ a population of solution sets, not a population of individual solutions as usually is the case with domination-based EAs such as [1], [13]. Each solution set is optimized towards the Pareto set, where at termination, the best solution set is selected from the population. For UHV-ADAM, only a single solution set is optimized. The internal mechanics of UHV-ADAM and UHV-GOMEA will be discussed in section 3.1 and 3.2 respectively.

2.3. Research question

In this work the main research question is: “Using the redefinition of multi-objective problems to single objective problems via the uncrowded hypervolume distance, can hybrid algorithms be constructed that are more efficient in solving multi-objective problems compared to pure evolutionary algorithms or gradient algorithms? “

To answer the research questions the following sub questions must be asked:

- How should the architecture of the hybrid, and in particular a hybrid that uses a dynamic resource allocation scheme, look like ?
- How does the empirically determined hybrid algorithm compare to its component algorithms?

3

Hybridization

This chapter describes the hybridization method of combining evolutionary algorithm UHV-GOMEA with gradient algorithm UHV-ADAM. This chapter starts off by highlighting the mechanics of the individual algorithms and the changes made to the original algorithms to facilitate the hybridization. The hybridization method is then described.

3.1. Gradient algorithm UHV-ADAM

3.1.1. Mechanisms of UHV-ADAM

UHV-ADAM [10] is based on the stochastic gradient descent algorithm ADAM [16]. It optimizes a single solution set $\mathbb{S}$ of p number of MO-solutions, which it parameterizes as ϕ^0 such that $\phi^0 = [\mathbf{x_0}, ..., \mathbf{x_{p-1}}] \in \mathbb{R}^{p \cdot n}$. Let $\mathbf{F}(\phi^0)$ be the operator that assesses the objective functions for every MO-solution in ϕ^0, as displayed in equation 3.1. UHV-ADAM starts by randomly initializing the MO-solutions of ϕ^0 and evaluates the objective values ($f_0(\mathbf{x_i})$, $f_1(\mathbf{x_i})$) and objective gradients ($\nabla f_0(\mathbf{x_i})$, $\nabla f_1(\mathbf{x_i})$) for every MO-solution $\mathbf{x_i}$ ($i = 0, \cdots, p-1$) of solution set ϕ^0. Using the objective values and objective function gradients, the gradient of the UHV indicator: $\nabla\text{UHV}(\phi^0)$ is calculated. $\nabla\text{UHV}(\phi^0)$ indicates how MO-solutions in the search space must move to (locally) obtain the most UHV gain. In section 3.1.2, the derivation of the UHV gradient is summarized for the interested reader.

UHV-ADAM determines the direction in which solutions are moved in the next iteration via a variance-corrected weighted average of $\nabla\text{UHV}(\phi^0)$ dubbed ζ. How far the solutions are moved is determined by γ and the variance correction. γ is determined by a shrinking scheme which reduces γ by 1 percent if no UHV improvement was found. The initial γ is computed by taking 1% of the average initialization range. This initialization method will be used later to reinitialize UHV-ADAM within the hybrid algorithm. UHV-ADAM repeats the process of calculating the UHV gradient and moving the solutions until all computation resources, e.g., a time or function evaluation budget, have been spent or a desired UHV value has been reached. The pseudo code of UHV-ADAM is shown in algorithm 1.

$$\mathbf{F}(\phi) = \begin{bmatrix} \mathbf{f(x_0)} \\ ... \\ \mathbf{f(x_{p-1})} \end{bmatrix} = \begin{bmatrix} f_0(\mathbf{x_0}) & ... & f_{m-1}(\mathbf{x_0}) \\ \vdots & \ddots & \vdots \\ f_0(\mathbf{x_{p-1}}) & ... & f_{m-1}(\mathbf{x_{p-1}}) \end{bmatrix}, \text{where } \phi = \begin{bmatrix} \mathbf{x_0} \\ ... \\ \mathbf{x_{p-1}} \end{bmatrix} \tag{3.1}$$

3.1.2. Derivation of the UHV gradient

In section 3.1.1, the UHV gradient, i.e. $\nabla\text{UHV}(\phi^0)$, was calculated as it indicates how MO-solutions in the search space must move to (locally) obtain the most UHV gain. Consequently, an expression needs to be determined for $\nabla\text{UHV}(\phi^0)$. We will summarize the derivation of [10] and [17], by starting off with the formalization of the UHV gradient in equation 3.2.

$$\nabla\text{UHV}(\mathbf{F}(\phi)) = \frac{\partial\text{UHV}(\mathbf{F}(\phi))}{\partial\phi} = \left[\frac{\partial\text{UHV}(\mathbf{F}(\phi))}{\partial\mathbf{x_0}}, ..., \frac{\partial\text{UHV}(\mathbf{F}(\phi))}{\partial\mathbf{x_{p-1}}}\right] \tag{3.2}$$

Algorithm 1 Pseudo code of UHV-ADAM

function UHV-ADAM
  $t \leftarrow 0$
  $\phi^0(t) \leftarrow$ initializeSingleSolutionSet()
  $\gamma(t) \leftarrow$ setInitialStepSize($\mathcal{X}_{\text{init}}$)
  $\mathbf{F}(\phi^0(t))$, $\nabla\mathbf{F}(\phi^0(t)) \leftarrow$ evaluate($\phi^0(t)$) // Evaluate fitness and gradients

  while ¬terminate **do**
    $\nabla\text{UHV}(\phi^0(t)) \leftarrow$ calculateUHVGradient($\mathbf{F}(\phi^0(t))$, $\nabla\mathbf{F}(\phi^0(t))$)
    $\zeta(t) \leftarrow$ updateWeightedAverage($\nabla\text{UHV}(\phi^0(t))$)
    $\phi^0(t+1) \leftarrow \phi^0(t) + \gamma(t)\zeta(t)$ // Update solution set
    $\mathbf{F}(\phi^0(t+1))$, $\nabla\mathbf{F}(\phi^0(t+1)) \leftarrow$ evaluate($\phi^0(t)$) // Evaluate fitness and gradients

    if $\text{UHV}(\phi^0(t+1)) < \text{UHV}(\phi^0(t))$ **then** // Update step size
      $\gamma(t+1) \leftarrow 0.99 \cdot \gamma(t)$
    else
      $\gamma(t+1) \leftarrow \gamma(t)$
    $t \leftarrow t+1$
  return ϕ^0 // Return the solution set

In section 2.2.2, the UHV was shown to be calculated in objective space. The MO-solutions of the solution set however, move in the search space. To bridge the two different spaces, the chain rule is applied on the UHV indicator, that is: first derive the UHV with respect to the objective space followed by a derivative of the objective space with respect to the search space. Equation 3.3 shows the chain rule for a single MO-solution $\mathbf{x_i}$.

$$\frac{\partial \text{UHV}(\mathbf{F}(\phi))}{\partial \mathbf{x_i}} = \frac{\partial \text{UHV}(\mathbf{F}(\phi))}{\partial \mathbf{F}(\phi)} \frac{\partial \mathbf{F}(\phi)}{\partial \mathbf{x_i}} = \sum_{j=0}^{p-1} \frac{\partial \text{UHV}(\mathbf{F}(\phi))}{\partial \mathbf{f}(\mathbf{x_j})} \frac{\partial \mathbf{f}(\mathbf{x_j})}{\partial \mathbf{x_i}} \tag{3.3}$$

In equation 3.3, $\frac{\partial \mathbf{f}(\mathbf{x_j})}{\partial \mathbf{x_i}}$ can be observed, which is the objective gradient, i.e. how the objective value changes with respect to the MO-solutions in the search space. To make it even more clear, this is the gradient that is analytically calculated or must be estimated. The objective gradient of MO-solution j is calculated irrespective of MO-solution i. Consequently, in equation 3.3, $\frac{\partial \mathbf{f}(\mathbf{x_j})}{\partial \mathbf{x_i}} = \mathbf{0}$ $\forall i \neq j$. For $i = j$, we obtain $\frac{\partial \mathbf{f}(\mathbf{x_i})}{\partial \mathbf{x_i}} = [\nabla f_0(\mathbf{x_i}), ..., \nabla f_{m-1}(\mathbf{x_i})]$, which is a concatenation of all objective gradients with respect to solution $\mathbf{x_i}$. As we only consider bi-objective problems, $\frac{\partial \mathbf{f}(\mathbf{x_i})}{\partial \mathbf{x_i}}$ can be further reduced to $[\nabla f_0(\mathbf{x_i}), \nabla f_1(\mathbf{x_i})]$. Furthermore, equation 3.3 can be reduced to equation 3.4.

$$\frac{\partial \text{UHV}(\mathbf{F}(\phi))}{\partial \mathbf{x_i}} = \frac{1}{W} \frac{\partial \text{UHV}(\mathbf{F}(\phi))}{\partial f_0(\mathbf{x_i})} \nabla f_0(\mathbf{x_i}) + \frac{1}{W} \frac{\partial \text{UHV}(\mathbf{F}(\phi))}{\partial f_1(\mathbf{x_i})} \nabla f_1(\mathbf{x_i}) \tag{3.4}$$

Equation 3.4 shows that the UHV is directly influenced by the objective gradients. An imbalance in the size of the objective gradients can bias the UHV gradient towards one of the objectives. For this reason, the objective gradients are normalized by $W = \left\| \left[\frac{\partial \text{UHV}}{\partial f_0(\mathbf{x_i})}, \frac{\partial \text{UHV}}{\partial f_0(\mathbf{x_i})} \right] \right\|$, transforming equation 3.4 to equation 3.5.

$$\frac{\partial \text{UHV}(\mathbf{F}(\phi))}{\partial \mathbf{x_i}} = \frac{\partial \text{UHV}(\mathbf{F}(\phi))}{\partial f_0(\mathbf{x_i})} \nabla f_0(\mathbf{x_i}) + \frac{\partial \text{UHV}(\mathbf{F}(\phi))}{\partial f_1(\mathbf{x_i})} \nabla f_1(\mathbf{x_i}) \tag{3.5}$$

The final step is to unpack the UHV into the hypervolume (HV) and uncrowded distance (ud). Equation 3.6 shows the unpacking for a single objective function, where k is the objective number ($k \in \{0, 1\}$). $\frac{\partial \text{HV}(\mathbf{F}(\phi))}{\partial f_k(\mathbf{x_i})}$ is the HV improvement with respect to objective function k. $\frac{\partial \text{UHV}(\mathbf{F}(\phi))}{\partial f_1(\mathbf{x_i})}$ is the improvement of the uncrowded distance with respect to objective function k.

$$\frac{\partial \text{UHV}(\mathbf{F}(\phi))}{\partial f_k(\mathbf{x_i})} = \frac{\partial \text{HV}(\mathbf{F}(\phi))}{\partial f_k(\mathbf{x_i})} - \frac{\partial \text{UD}(\mathbf{F}(\phi))}{\partial f_k(\mathbf{x_i})} \tag{3.6}$$

When a solution is dominated, $\frac{\partial \text{HV}(\mathbf{F}(\phi))}{\partial f_k(\mathbf{x_i})}$ does not contribute to the HV and can be set to 0. If it is non-dominated $\frac{\partial \text{HV}(\mathbf{F}(\phi))}{\partial f_k(\mathbf{x_i})}$, can be directly determined from the approximation boundary $\partial\mathbf{f}(\mathbb{S})$, as is shown in figure 3.1 for $\mathbf{x}_2$.

For $\frac{\partial \text{UD}(\mathbf{F}(\phi))}{\partial f_k(\mathbf{x_i})}$, when a solution is non-dominated, it influences where the approximation front is located and thus potentially can worsen the uncrowded distance. An improvement in HV however, is always more desirable than a worsening uncrowded distance. For this reason, the uncrowded distance is set to 0 for non-dominated solutions. If an MO-solution is dominated however, $\frac{\partial \text{UD}(\mathbf{F}(\phi))}{\partial f_k(\mathbf{x_i})}$ can be determined as it is the horizontal and vertical distance to the the closest point on the approximation boundary, as is shown in figure 3.1 for $\mathbf{x}_4$.

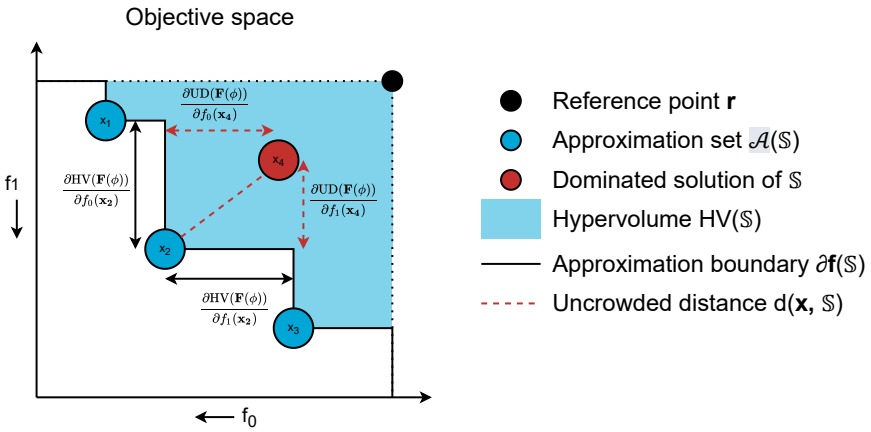


Figure 3.1: Components of the hypervolume and uncrowded distance gradients.

3.1.3. Changes made to UHV-ADAM

In this work UHV-ADAM has been extended such that the single-solution set solving algorithm is compatible with population-based UHV-GOMEA. To this end, UHV-ADAM steps are applied to population members after a generation of UHV-GOMEA. UHV-ADAM instances are assigned to each solution set of the population, allowing the weighted moving average and γ to be tuned accurately and differently to the environment of each solution set in the population. UHV-ADAM instances are reset every time the variation operator of UHV-GOMEA is applied to prevent γ and the moving averages of UHV-ADAM instances to become inaccurate if UHV-GOMEA makes big leaps in the search space. Resetting the UHV-ADAM instances comes at the cost of warming up the moving averages again as well as redetermining γ. γ is re-estimated by creating the tightest box that contains all MO-solutions of the population and to take 1 percent of the average box width. Finally, a RAS will be used to adaptively determine which algorithm (ADAM or GOMEA) should be used more during a run. After determining the resource distribution, the resources assigned to UHV-ADAM must be distributed over the population members. The distribution methods will be introduced in section 3.3.3, after the resource allocation scheme is introduced.

3.2. Evolutionary algorithm UHV-GOMEA

3.2.1. Mechanisms of UHV-GOMEA

UHV-GOMEA [9] is a recently introduced UHV-based EA that leverages strengths of the single-objective model-based EA known as RV-GOMEA [2]. UHV-GOMEA is part of the Gene-pool Optimal Mixing (GOM) family of algorithms, known for its use of the GOM operator and the modeling of linkage via a Family Of Subsets (FOS). Both mechanisms will be explained in section 3.2.2.

UHV-GOMEA starts off by randomly initializing and evaluating a population of N solution sets: $\phi = [\phi^0, \cdots, \phi^{N-1}]$, where each individual ϕ^i $(i = 0, \cdots, N-1)$ has p MO-solutions. Gradient information is not used nor calculated. UHV-GOMEA then selects the best 35% of the solution sets with the highest UHV value as parents. A variation operator is applied on the parents to create new offspring solution sets, offspring is evaluated and statistics are calculated to tune the variation operator for the next generation. This process is repeated until termination. The pseudo code is shown in algorithm 2.

Algorithm 2 The basic structure of UHV-GOMEA

```
1:  function UHV-GOMEA
2:      t ← 0
3:      φ(t) ← initializePopulation()
4:      F(φ(t)) ← evaluate(φ(t))                          // Evaluate population
5:
6:      while ¬terminate do
7:          parentSet ← selectParents(φ(t), F(φ(t)))
8:          for F_j ∈ F do                                // Determine parameters for offspring
9:              N(μ̂_{F_j}(t), Σ_{F_j}(t)) ← estimateAndAdaptGaussian(parentSet)
10:         φ(t+1) ← createNewSolutionSets()
11:         t ← t + 1
12:     return φ                                          // Return the solution set
```

3.2.2. GOMEA

The variation operator of UHV-GOMEA is based on the GOM operator of RV-GOMEA. The GOM operator is used together with a Family Of Subsets (FOS) linkage model. The FOS allows dependencies between problem variables to be captured. Normally a FOS, denoted by $\mathcal{F}$, is a subset of the powerset of all problem variables. However as UHV-GOMEA optimizes multiple MO-solutions, the powerset is taken over the parameterization of the solution set, i.e. $\mathcal{P}([0, 1, ..., p \cdot n - 1])$. Each subset $\mathcal{F}_j$ of $\mathcal{F}$ describes the indices of the solution set that are deemed dependent. Although any linkage model can be constructed, for the hybrid, we only focus on the marginal product linkage model (Lm) as UHV-GOMEA has shown superior performance with Lm in most experiments of [9]. The Linkage Tree (Lt) FOS however, will also be used, but only by UHV-GOMEA when it acts as a reference in the WFG benchmark.

Before any linkage model is created in UHV-GOMEA, UHV-GOMEA first greedily rearranges the MO-solutions of each solution set at the start of a generation. MO-solutions of each solution set are sorted such that all i'th MO-solution $\mathbf{x_i}$ $(i = 0, ..., p-1)$ is in the same region of the approximation front. For this, the objective space means m_i are calculated, where m_i measures the mean objective values of the set of all i'th MO-solution, i.e. $\mathbf{m}_i = \frac{1}{N}\sum_{j=1}^{N} \mathbf{f}(\mathbf{x}_i^j)$. Each MO-solution of every solution set is then greedily re-ordered by iteratively finding the (next-)nearest MO-solution-mean pair. At initialization, this greedy sorting process will not directly lead to sorted solution sets, however as time progresses solutions will be correctly ordered.

After sorting the MO-solutions of each solution set, the aforementioned linkage models can be constructed. Lm groups all solution set variables pertaining to $\mathbf{x_i}$ $(i = 0, ..., p-1)$ into sets. These sets are part of the Lm FOS. Figure 3.2 illustrates the relationship between $\mathbf{x_i}$ and $\mathcal{F}_i$ for Lm. Lt creates multiple levels of linkage, by hierarchically clustering the indices of the solution set. It does this in two stages. Figure 3.3 will be used as an example where a solution set consists of 4 MO-solutions with a problem dimensionality of 3. At the bottom of figure 3.3, the individual indices of the solution set are represented as squares, indexed from 0 to 11. Subsets $[0, 1, 2]$, $[3, 4, 5]$, $[6, 7, 8]$ and $[9, 10, 11]$ represent MO-solutions $\mathbf{x}_0$, $\mathbf{x}_1$, $\mathbf{x}_2$ and $\mathbf{x}_3$ respectively. In the first stage of constructing the Lt, a hierarchical tree is created of the indices pertaining to an MO-solution. For this, UPGMA [18] is used, where the two nearest subsets are merged iteratively based on a metric, until the original MO-solution is reconstructed or until a subset size larger than $0.35N - 1$ is reached. In the example, this means that in the first stage we merge solution indices $0, 1, 2$, according to the UPGMA process, until set $[0, 1, 2]$ is obtained. The metric used to merge subsets is based on joint mutual information (MI). For the MI, we collect all the problem parameter values of the population pertaining to the indices of a FOS subset.

The MI is then calculated as $\text{MI}_{ij} = \log\left(\sqrt{\frac{1}{1-(\frac{\hat{\Sigma}_{ij}}{\hat{\sigma}_i\hat{\sigma}_j})^2}}\right)$, where $\hat{\sigma}_j$ is the estimated standard deviation of all problem parameter values pertaining to index i and $\hat{\Sigma}_{ij}$ is the estimated covariance between indices i and j. For a set of indices X, where $X = a \cup b$, the MI between X and another set of indices Y is calculated as $MI_{XY} = \frac{|a|}{|X|}\text{MI}_{aY} + \frac{|b|}{X}\text{MI}_{bY}$.

After the tree is created for the MO-solutions, UHV-GOMEA continues to the second phase, where it merges the indices of the solution sets pertaining to the individual MO-solutions. It re-uses UPGMA to do this, however the Euclidean distance between the mean objective value m_i is used instead of mutual information. In the second phase, Lt still limits the subset size to $0.35N - 1$ as dependencies of larger subsets can not be estimated. After the second phase is done, Lt contains all merged subsets and original problem variables indices, except the FOS subsets that are larger than the subset size limit. An example where everything is included is shown on the right side of figure 3.3. Lt is reconstructed every generation as the MO-solutions of the population change over time.

Using any linkage model that is eventually created, the GOM operator is finally applied on all population members such that new solutions are created. The GOM operator mixes a selected parent solution with a donor solution, by changing the parent's problem variables with that of a donor solution according to the indices of a selected FOS. Specifically, for each FOS element which contains a set of indices, a multivariate joint Gaussian distribution is estimated over all values of the population pertaining to the indices of the selected FOS subset. From this distribution, new solutions are sampled and injected into the parent solution. If the UHV of the parent improves, the change is kept. Otherwise, the change is rejected. All FOS subsets of the FOS are applied, on all parents of the population.

Before sampling happens though, the distribution parameters are slightly changed. The "anticipated mean shift" feature anticipates where the mean of the Gaussian should move based on the historic values. The "adaptive variance scaling" feature enlarges the variance of the distribution to counter the variance diminishing effects of selection.

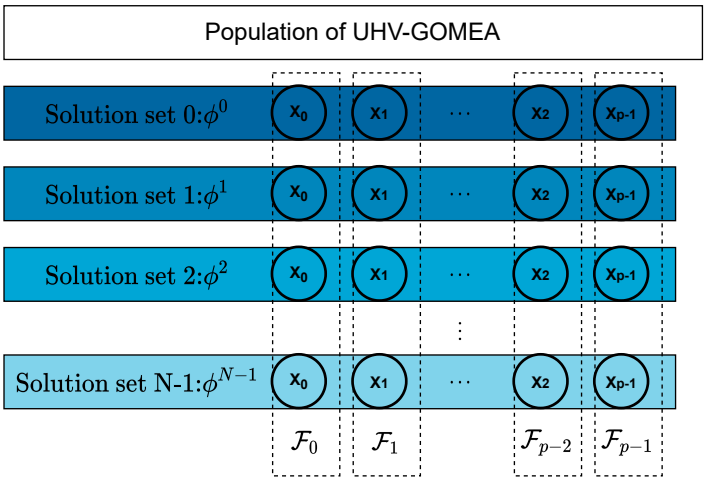


Figure 3.2: Marginal linkage (Lm) of UHV-GOMEA. Each circle represents an MO-solution that is greedily sorted, such that all i'th MO-solution ($i = 1, \cdots, p-1$) is in the same part of the approximation front. The solution sets of the population are displayed horizontally. The linkage subsets are shown vertically.

3.3. The hybridization

The hybrid created in this work is based on [11], where a resource allocation scheme (RAS) was used to dynamically assign resources to gradient algorithms. The RAS of [11] is extended in this work, to accommodate the modified UHV-ADAM. Furthermore, only UHV-GOMEA and the modified UHV-ADAM are hybridized.

The hybrid algorithm starts off similar to UHV-GOMEA, by initializing a population of N solution sets. Each solution set is assigned a UHV-ADAM instance, which is initialized accordingly. The hybrid algorithm executes the variation operator of UHV-GOMEA and the update operator of UHV-ADAM

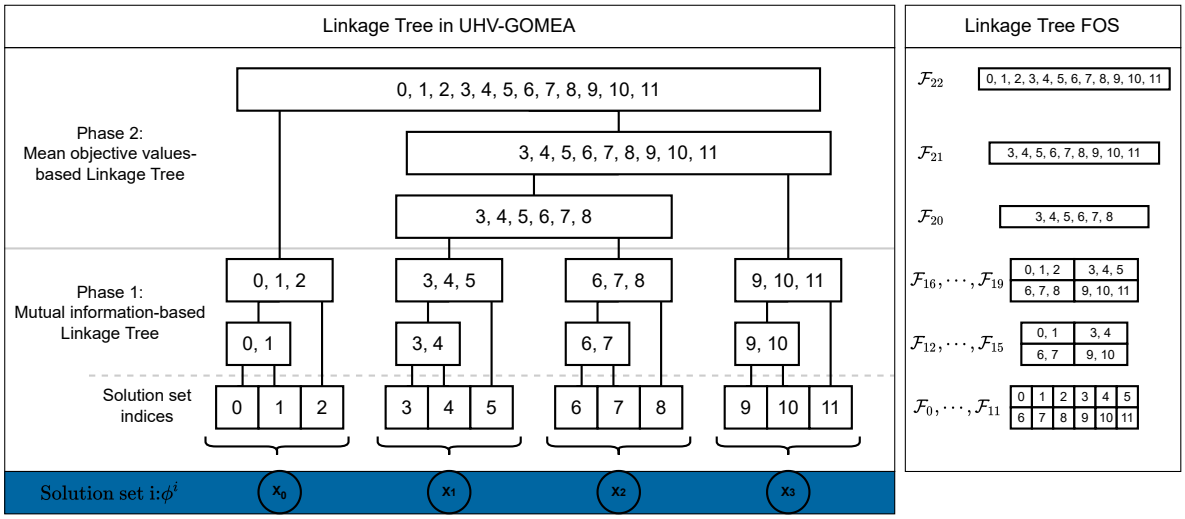


Figure 3.3: The two phased construction of a linkage tree (Lt) in UHV-GOMEA. UHV-GOMEA first constructs a mutual information-based tree for the MO-solutions. In the second phase, it constructs subsets of MO-solutions using the Euclidean distance between the mean of the objective values. On the right side of the image, all the FOS subsets are displayed.

sequentially. UHV-GOMEA's operator is always executed once per generation, while the extended RAS determines the number of UHV-ADAM operations.

3.3.1. The resource allocation scheme

Let the actual number of evaluations and improvements found in generation t by optimizer $o \in \{\text{GOMEA}, \text{ADAM}\}$ be $E_o(t)$ and $I_o(t)$ respectively. An evaluation occurs when one MO-solution $\mathbf{x_i}$ is evaluated. What entails an improvement will be discussed later in section 3.3.2. Let the number of evaluations and improvements to be considered for redistribution be $\mathcal{E}_o(t)$ and $\mathcal{I}_o(t)$ respectively. For UHV-ADAM, only the values of the current generation are of interest, that is: $\mathcal{E}_{\text{ADAM}}(t) = E_{\text{ADAM}}(t)$ and $\mathcal{I}_{\text{ADAM}}(t) = I_{\text{ADAM}}(t)$. The number of evaluations and improvements to be considered for UHV-GOMEA is a sum of values of previous generations, that is: $\mathcal{E}_{\text{GOMEA}}(t) = \sum_{t'=t_{min}}^{t} E_{\text{GOMEA}}(t)$ and $\mathcal{I}_{\text{GOMEA}}(t) = \sum_{t'=t_{min}}^{t} I_{\text{GOMEA}}(t)$, where $t_{min} \geq 0$ and t_{min} is chosen as large as possible such that $\mathcal{E}_{\text{GOMEA}}(t) \geq \mathcal{E}_{\text{ADAM}}(t)$ still holds. Loosely speaking, by imposing the aforementioned criteria on t_{min}, UHV-GOMEA, at the minimum, matches the resources spent by UHV-ADAM. Figure 3.4 illustrates two arbitrary examples. UHV-GOMEA includes past values for two reasons: it makes the comparison between the gradient algorithm and EA fairer and also allows the number of gradient algorithm calls to grow [19].

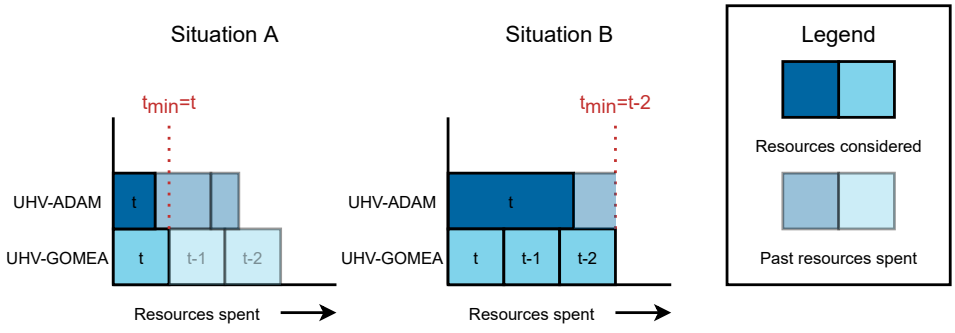


Figure 3.4: Resources considered by UHV-GOMEA and UHV-ADAM in two arbitrary situations. The resources considered by UHV-GOMEA is at minimum the number of resources considered by UHV-ADAM.

Following [19], the EA's variation operator is executed once per generation while the number of executions of the gradient algorithms are related to the respective reward they receive. The reward, displayed in equation 3.7, is the efficiency of finding improvements. The reward is 0 if $\mathcal{E}_o(t) = 0$.

$$\mathcal{R}_o(t) = \frac{\mathcal{I}_o(t)}{\mathcal{E}_o(t)} \tag{3.7}$$

Let the evaluations to be redistributed to UHV-ADAM be $\mathcal{E}_{\text{ADAM}}^{\text{Redistributed}}(t)$. $\mathcal{E}_{\text{ADAM}}^{\text{Redistributed}}(t)$ is the ratio of UHV-ADAM's contribution to the total reward times the total sum of evaluations to be considered in generation t as shown in equation 3.8.

$$\mathcal{E}_{\text{ADAM}}^{\text{Redistributed}}(t) = \frac{\mathcal{R}_{\text{ADAM}}(t)}{\sum_{o'} \mathcal{R}_{o'}(t)} \sum_{o'} \mathcal{E}_{o'}(t) \tag{3.8}$$

To calculate the number of iterations UHV-ADAM can execute with budget $\mathcal{E}_{\text{ADAM}}^{\text{Redistributed}}(t)$, let the number of calls be $\mathcal{C}_{\text{ADAM}}^{\text{Redistributed}}(t)$, where $\mathcal{C}_{\text{ADAM}}^{\text{Redistributed}}(t)$ can be calculated by dividing the resources assigned to UHV-ADAM by the average number of evaluations required per call. The average evaluations per call are estimated using the resources and calls of generation t, resulting in equation 3.9.

$$\mathcal{C}_{\text{ADAM}}^{\text{Redistributed}}(t) = \frac{\mathcal{E}_{\text{ADAM}}^{\text{Redistributed}}(t)}{\frac{\mathcal{E}_{\text{ADAM}}(t)}{\mathcal{C}_{\text{ADAM}}(t)}} = \frac{\mathcal{C}_{\text{ADAM}}(t)}{\mathcal{E}_{\text{ADAM}}(t)} \mathcal{E}_{\text{ADAM}}^{\text{Redistributed}}(t) \tag{3.9}$$

To ensure a smooth decrease in the number of gradient calls, memory decay is implemented in equation 3.10. If the number of calls after redistribution is smaller than the number of calls executed in the current generation, a running average is used to decrease the number of calls. If the number of calls increases, memory decay is not applied in order to stimulate the use of gradient algorithms [19]. The (memory) decay factor η is the is kept to the original value of 0.75 [19].

$$\mathcal{C}_{\text{ADAM}}^{\text{Run}}(t+1) = \begin{cases} \mathcal{C}_{\text{ADAM}}^{\text{Redistributed}}(t), & \text{if } \mathcal{C}_{\text{ADAM}}^{\text{Redistributed}}(t) \geq \mathcal{C}_{\text{ADAM}}^{\text{Run}}(t) \\ \eta^{\text{decay}} \mathcal{C}_{\text{ADAM}}^{\text{Run}}(t) + (1-\eta^{\text{decay}}) \mathcal{C}_{\text{ADAM}}^{\text{Redistributed}}(t), & \text{otherwise} \end{cases} \tag{3.10}$$

The number of UHV-ADAM calls to execute next generation could be set to $\mathcal{C}_{\text{ADAM}}(t+1) = \left\lfloor \mathcal{C}_{\text{ADAM}}^{\text{Run}}(t+1) \right\rfloor$, However, if at some point $\mathcal{C}_{\text{ADAM}}(t) = 0$ holds, UHV-ADAM cannot be activated any more. As UHV-ADAM could become useful again in the future, a waiting scheme is used that makes UHV-ADAM wait $\mathcal{W}_{\text{ADAM}}(t)$ generations. In [19], gradient algorithms are only allowed to be executed at most once per individual per generation. Furthermore, at most (population size) N number of total calls can be executed per generation. Early experiments have shown that executing one UHV-ADAM call per individual does not substantially affect convergence. For this reason, multiple gradient calls can be applied to the same individual. Furthermore, a lower bound is introduced such that if UHV-ADAM is to be executed, it executes at least $\mathcal{C}_{\text{ADAM}}^{\text{min}}$ calls. This ensures that the performance of UHV-ADAM is assessed after it warms up its internal parameters. $\mathcal{C}_{\text{ADAM}}^{\text{min}}$ is set to 10 and has not been further optimized. The cap on total gradient calls is kept and set to N.

The modified waiting scheme is shown in equation 3.11. UHV-ADAM is forced to wait for some generations when $\mathcal{C}_{\text{ADAM}}^{\text{Run}}(t+1) \leq \mathcal{C}_{\text{ADAM}}^{\text{min}}$. Because the extended UHV-ADAM executes a minimum number of calls, $\mathcal{C}_{\text{ADAM}}^{\text{min}}$ has been added to prevent the waiting scheme from triggering too early. The actual number of calls to be executed is shown in equation 3.12, where $\mathcal{C}_{\text{ADAM}}^{\text{min}}$ has also been added to the original equation.

$$\mathcal{W}_o(t+1) = \begin{cases} \left\lfloor \frac{\mathcal{C}_o^{\text{min}}}{\mathcal{C}_o^{\text{Run}}(t+1)} \right\rfloor, & \text{if } \mathcal{W}_o(t) = 0 \\ \mathcal{W}_o(t) - 1, & \text{otherwise} \end{cases} \tag{3.11}$$

$$\mathcal{C}_o(t) = \begin{cases} \mathcal{C}_o^{\text{min}}, & \text{if } \mathcal{W}_o(t-1) = 1 \\ \min(\left\lfloor \mathcal{C}_o^{\text{Run}}(t) \right\rfloor, N), & \text{otherwise} \end{cases} \tag{3.12}$$

3.3.2. Improvement metric

In the resource allocation scheme, any improvement metric can be used to guide the allocation of resources. In this work, five different methods have been investigated. The results of the investigation will be presented in experiment 1. Each metric is measured after the execution of a single call:

1. **Bosman2012***: The number of MO-solutions that are added to an infinite sized elitist archive. This methodology is based on the original metric used in [19].
2. Δ**BestUHV**: Takes the difference between the best found solution's UHV value.
3. Δ**AverageUHV**: Takes the difference between the population's average UHV value.
4. **CountUHVImproved**: Counts the number of times a solution set has improved its UHV value.
5. **CountBestUHVImproved**: Counts the number of times a solution set has been found that improved the best found UHV.

In Bosman2012 [11], an improvement is defined as when a newly created MO-solution is added to an elitist archive. The elitist archive is a finite sized storage space for non-dominated MO-solutions. In Bosman2012*, we use an infinitely large elitist archive to encourage counting MO-solutions that improve the UHV, which otherwise are potentially rejected by a finite sized elitist archive when it is full. In an elitist archive of [9] or [11] that is full, new MO-solutions are only accepted if existing MO-solutions of the archive are dominated. Undesirable rejections are likely to occur in this archive when, for example, MO-solutions are offered that improve the UHV by spreading out without dominating any existing MO-solution of the archive. The infinite sized elitist archive in contrast, accepts all non-dominated solutions. The infinite sized elitist archive however, comes at the cost of additional memory requirements and computation time. This is however, not measured in the experiments. In this regard, Bosman2012* should be considered as a best case scenario.

In this work, indicator function based improvement metrics, as well as counting methods are also investigated. Metrics ΔBestUHV and ΔAverageUHV are the difference between the best found UHV and average population UHV respectively in subsequent generations. CountUHVImproved and CountBestUHVImproved count the number of times the UHV of a solution and that of the best solution have improved respectively.

3.3.3. Distribution method

In the resource allocation scheme, the resources assigned to UHV-ADAM must be distributed over the population members. The following allocation methods have been explored and will be discussed in experiment 3:

- **Best-m-Solutions**: Distribute the resources uniformly to the solution sets that have the m highest UHV values.
- **Best-m%Population**: Distribute the resources uniformly to the solution sets that have the top $m\%$ highest UHV values of the population.
- **All**: Distribute the resources to all solution sets, starting at $\phi^0, \phi^1, \cdots$ until all resources have been distributed.
- **Random**: Distribute the resources randomly, with replacement.

4

Experiments

This chapter describes the experiments that were performed in this work. The setup of the experiments is explained first. The four experiments that were conducted are then shown. In the first experiment, the effects of the improvement metrics of section 3.3.2 are displayed. In the second experiment, the effects of choosing a different gradient algorithm (operators) are shown. In the third experiment, the effects of the choice of different gradient algorithm distribution methods are presented. Finally, the most promising hybrid algorithms in this work are benchmarked on the WFG [20] problem set.

4.1. Setup

4.1.1. Benchmark problems

The multi-objective problems used in the experiments are given in table 4.1, where n is the problem dimensionality. Figure 4.1 further shows a 3D impression of the single-objective functions that are used, where we zoom into the interesting parts of the objective functions. The height of the surface and the heat map both represent the obtained objective values. For this, MO-solutions are sampled from a grid and evaluated. The lower and redder the surface, the fitter an MO-solution is. The blue crosses indicate the optimum of an objective function. Figure 4.2 shows an impression of the bi-objective problems of table 4.1. The height of the surface and the heat map both represent the domination rank calculated by sampling MO-solution from a grid. The domination rank indicates how dominated an MO-solution is. As the desired Pareto set contains all non-dominated solutions, the lower the rank, the better. The domination rank is calculated by determining the non-dominated solutions of the grid, to rank these non-dominated solutions starting from rank 0, to remove these non-dominated MO-solutions from the ranking process and finally to repeat the process of finding, ranking and removing the next non-dominated solutions. In figure 4.2, the blue crosses in the lowest row indicate the non-dominated solutions of the grid.

Table 4.1: The bi-objective benchmark problems selected for the experiments.

#	Problem name	Objectives	Properties
0	Convex bi-sphere	$f_0 = f_{\text{sphere}}(\mathbf{x})$, with $f_{\text{sphere}}(\mathbf{x}) = \sum_{i=0}^{n-1}(x_i)^2$ $f_1 = f_{\text{sphere}}(\mathbf{x} - \mathbf{c_0})$ $\mathbf{c}_0 = [1, 0, \cdots, 0]$	Uni-modal, Decomposable
1	Convex sphere Rosenbrock	$f_0 = \frac{1}{n} f_{\text{sphere}}(\mathbf{x})$ $f_1 = \frac{1}{n-1} f_{\text{ros}}(\mathbf{x})$, with $f_{\text{ros}}(\mathbf{x}) = \sum_{i=0}^{n-2}(100(x_{i+1} - x_i^2)^2 + (1 - x_i)^2)$	Multi-modal, Attraction to f_0
2	Convex sphere Rastrigin	$f_0 = f_{\text{sphere}}(\mathbf{x})$ $f_1 = f_{\text{rast}}(\mathbf{x} - \mathbf{c_2})$, with $f_{\text{rast}}(\mathbf{x}) = An + \sum_{i=0}^{n-1} x_i^2 - A\cos(2\pi x_i)$ $A = 10, \mathbf{c}_2 = [0.5, 0, \cdots, 0]$	Multi-modal
3	Bi-Cosine sphere	$f_0 = f_{\cos}(\mathbf{x})$, with $f_{\cos}(\mathbf{x}) = f_{\text{sphere}}(\mathbf{x})(1 - \beta\cos(2\pi f \lvert\mathbf{x}\rvert))$ $f_1 = f_{\cos}(\mathbf{x} - \mathbf{c_0})$ $\beta = 0.6, f = 0.1$	Multi-modal in f_0 and f_1

Problem 0 (Convex bi-sphere) is uni-modal, objective-wise decomposable [9] and can be quickly solved with gradient algorithms [10]. The Pareto set that is formed by slightly shifting f_1 is visible in

the bottom row of figure 4.2. Problem 1 (Sphere-Rosenbrock) is a low multi-modal problem based on the the Rosenbrock function which has pair-wise dependencies [21]. It is known for pulling algorithms towards the optimum of the more easily solvable Sphere function while potentially getting solutions stuck in a local optimum of the Rosenbrock function. The local optimum however, is not visible in figure 4.1. Problem 1 is scaled to have its Pareto endpoints located at $(1, 0)$ and $(0, 1)$ as was done in [9]. Problem 2 (Sphere-Rastrigin) contains the multi-modal Rastrigin [22] problem, where many local optima are evenly scattered around the solution space. The scattering is visible in figure 4.1 and 4.2. During early experiments, it was observed that Problem 2 can not be easily scaled without retaining its multi-modal properties. Scaling the Rastrigin function by $\frac{1}{n}$ (as was done in Problem 1) for example, makes the problem solvable for UHV-ADAM when n gets larger. Most likely, dividing by n flattens the objectives, allowing UHV-ADAM to ignore any local optima it encounters. Shifting the optima of Rastrigin, such that a disconnected Pareto set is created is also undesirable, as it makes the problem hard to solve with a marginal product linkage model. For these reasons, Problem 2 is not scaled nor shifted into having a disconnected Pareto set. Problem 3, which is designed by myself, is multi-modal in both objectives. The Pareto set of Problem 3 is enveloped by basins, as shown in figure 4.2.

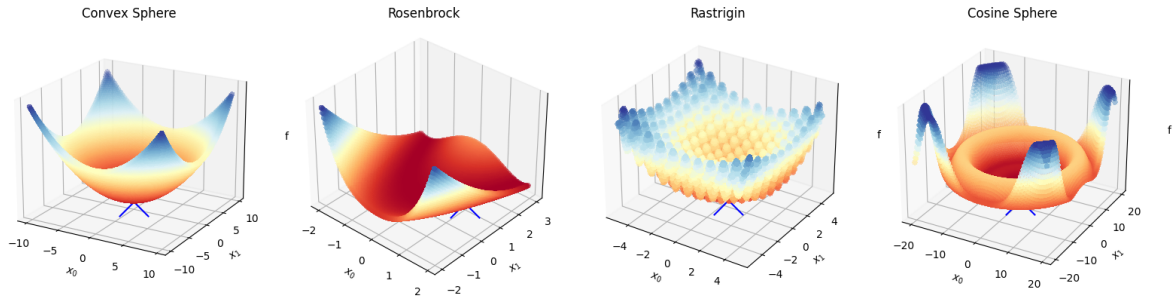


Figure 4.1: Impression of the single-objective problems used in the experiments in a 2D search space. The heat map and height represent the objective values obtained by assessing an objective function on a grid of solutions. The blue crosses indicate the optimum of an objective function.

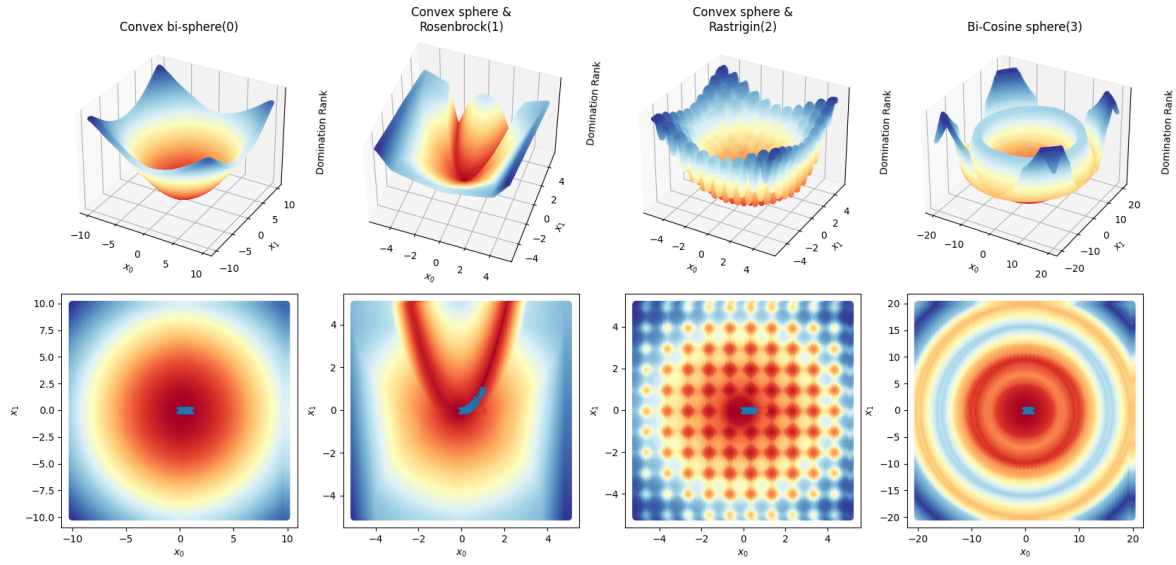


Figure 4.2: Impression of the Bi-objective problems in a 2D search search space, where the height and heat map represent the domination rank. The lower the domination rank, the less dominated an MO-solution is. In this first row, a 3D impressions is given of the domination rank. In the second row, a top-view of the 3D representation is given, where the blue crosses represent the non-dominated solutions of all the MO-solutions used to generate this graph. The non-dominated solutions give an impression where the Pareto set lies.

The algorithms are asymmetrically initialized around the Pareto set for Problem 0, 1 and 3. The initialization ranges are $[-400, 100]^n$ for Problem 0 and 3, $[-20, 10]^n$ for Problem 1. In Problem 2 the

recommended initialization range of $[-5.12, 5.12]^n$ are used. For every problem, the HV reference point is set at $r = (11, 11)$.

4.1.2. Convergence criteria

In this work, all algorithms are run until the computational budget of 10^7 MO-evaluations is expended or until convergence criteria are met. In the first three experiments, the gradients are analytically calculated and come at no cost. In the WFG benchmark, gradients are approximated, which do come at a cost. Regarding the convergence criteria, a target hypervolume (HV) is set per problem and problem dimension, where algorithms that reach the target HV with 6 digits of accuracy, i.e. $\Delta\text{HV}_p < 10^{-6}$, are considered to have converged and successfully solved the problem. The HV is chosen over the UHV, as it is expected that during convergence, all the MO-solutions of a solution set become non-dominated, which reduces the UHV to the HV.

The best found HV is used as the target HV. The best found HV is determined by initializing all algorithms close to the Pareto set, to run each algorithm 30 times with a budget of 10^8 MO-evaluations, after which the best found HV is selected. To produce stable experimental results, it is furthermore confirmed that the 30 best performing runs, amongst all results, all obtain the same HV up to 6 digits of accuracy.

4.2. Experiment 1: Improvement metric

In the resource allocation scheme, the improvement metric quantifies the performance of an optimizer and influences how many resources an optimizer is assigned. In the first experiment, we look at the effect of different improvement metrics. Specifically, we use the problems of table 4.1, and investigate how the implementations behave on different problem dimensionalities. The results are presented in two ways. In the first method, we quantify the performance of each improvement metric via a numerical value. In the second method, we visualize the scalability of the implementations.

For the first method, we tune the implementations on Problem 1 to 3 on the following problem dimensionalities $D = [2, 5, 10, 20, 40, 80]$. Problem 0 is excluded from this method, as tuning the hybrid on such a simple problem is undesirable. For each dimension, we determine the best population N by running the following population sizes $N = [40, 80, 160, 320, 640, 1280]$ 30 times and select the most efficient population size that reaches a median target HV accuracy of $\Delta\text{HV}_p < 10^{-6}$ with a success rate higher than 29 out of 30 runs. Implementations without a single population that meets the success rate threshold are disqualified. The performance score of an implementation is calculated in equation 4.1, which sums the relative performance of the improvement metric imp with respect to the best performing improvement metric amongst all improvement metrics I, over all problems P and problem dimensionalities D.

$$\text{score(imp)} = \sum_{\text{pr} \in \text{P}} \sum_{\text{d} \in \text{D(pr)}} \frac{\text{median(MO-Evaluations(pr, d, imp))}}{\min_{\text{imp}' \in \text{I}} (\text{median(MO-Evaluations(pr, d, imp}'))))} \tag{4.1}$$

In the second method, we reuse the population tuning method and visualize the results of the optimized populations. Specifically, we show the median and interquartile ranges of each optimized population size for each problem dimensionality. Implementations that are disqualified for not reaching the target success rate remain an important source for analysis. For these disqualified implementations, we sort the populations on success rate and MO-evaluations, and determine the statistics on all successful runs.

The base hybrid algorithm used in experiment 1 uses UHV-ADAM as gradient algorithm. Gradient calls are applied on the best 3 solutions with the highest UHV, as in initial experiments this method showed decent results. The number of MO-solutions in a solution set is set to $p = 9$. In the visualization, UHV-GOMEA and UHV-ADAM are also added as a reference.

4.2.1. Results of experiment 1

The effect of the improvement metrics on the required number of MO-evaluations to reach a target accuracy of $\Delta\text{HV}_p < 10^{-6}$, the corresponding success rates (SR) and optimized population sizes are shown in figure 4.3. Table 4.2 shows the scores obtained by the hybrids when using the respective

improvement metrics, according to equation 4.1.

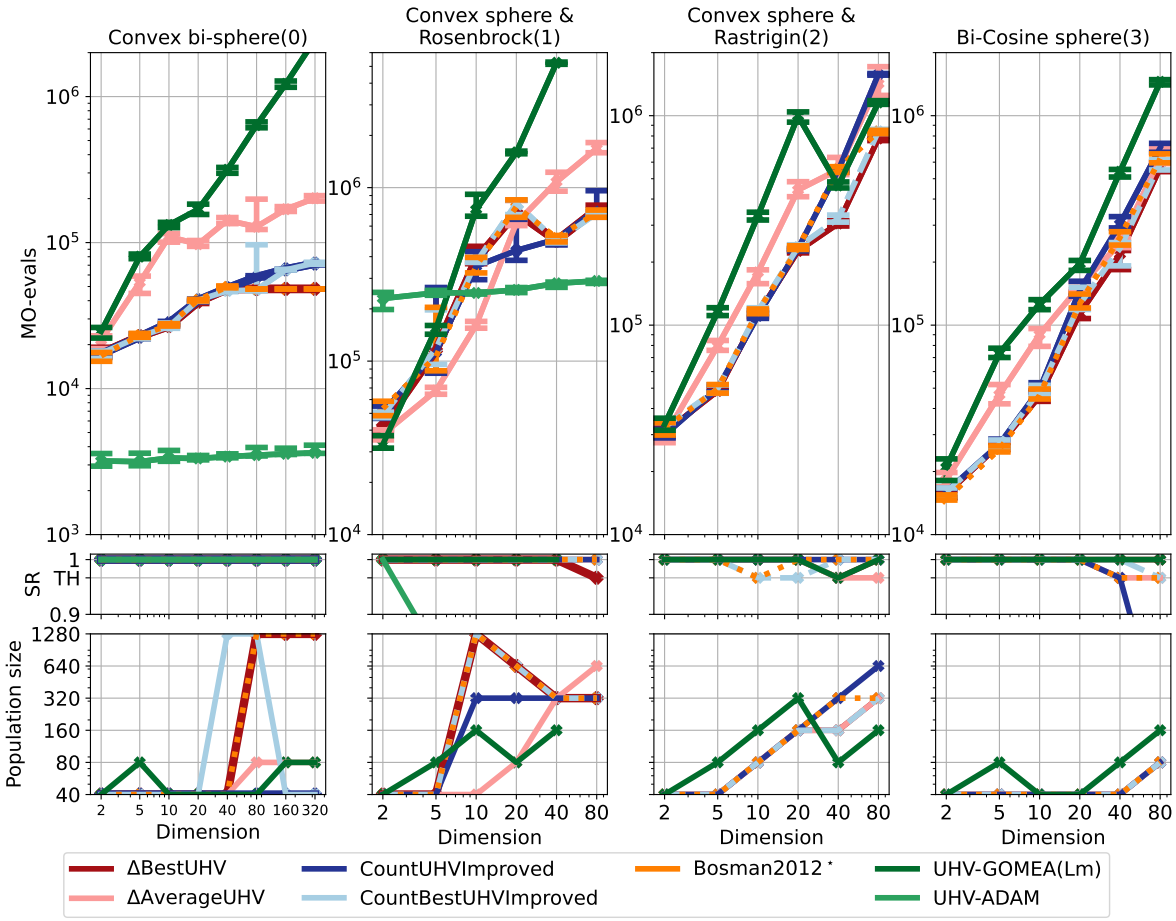


Figure 4.3: The effect of the improvement metrics on the required number of MO-evaluations to reach a target accuracy of $\Delta\text{HV}_p < 10^{-6}$ for various problems where $p = 9$. The success rate (SR) measures the fraction of runs that reach the target accuracy out of 30 runs, where the target threshold (TH) is 29/30 runs. The bottom row shows the optimized population size.

Inspecting UHV-GOMEA in figure 4.3, shows that UHV-GOMEA is able to reach the target HV in all problems except Problem 1, where it fails to find the target HV in dimension 80 due to the computation budget limit. In Problem 2, the problem seems to become easier for UHV-GOMEA after $n = 40$. Analysis shows that reliability is not the main issue, as the dip in the the graph persists when the number of runs is increased from 30 to 90 runs. It is unknown why UHV-GOMEA dips. Inspecting UHV-ADAM shows that the algorithm is able to reach the target HV for Problem 0 reliably and most efficiently. In Problem 1 it can not reliably reach the target HV. Interestingly, in [10], it is able to solve Problem 1 if it is initialized closer to the Pareto set ($[0, 2]^n$). In this experiment, however it gets stuck on local optima. UHV-ADAM fails to reach any target HV in Problem 2 and Problem 3.

Among the improvement metrics in figure 4.3, Bosman2012* and ΔBestUHV (hidden underneath Bosman2012*) are performing the best in Problem 0. Interestingly, these improvement metrics pick large population sizes starting from $n = 80$. CountBestUHVImproved mimics this behavior, but abandons this strategy when $n >= 160$. An analysis of the substantial increase in population is done in section 4.2.2. Figure 4.4 shows the average utilization rate and interquartile ranges for the largest problem dimensions. The utilization rate is the number of calls that are executed, normalized by the maximum number of calls permitted by the resource allocation scheme, which is population size N. Figure 4.4 shows that most improvement metrics at some point have full utilization for Problem 0. This is to be expected as UHV-ADAM is extremely efficient at solving Problem 0. Improvement metric ΔAverageUHV is the only improvement metric that does not reach full utilization; hence it is the least efficient out of all the improvement metrics in figure 4.3. ΔAverageUHV is biased towards UHV-GOMEA

Table 4.2: The scores assigned to each improvement metric. The lower the score, the better. The numbers in bold are the lowest scores of a category.

Problem	ΔBestUHV	ΔAverage UHV	Count UHVImproved	CountBestUHV Improved	Bosman2012*
Convex Sphere & Rosenbrock (1)	9.0	9.0	**8.3**	9.2	9.0
Convex Sphere & Rastrigin (2)	**6.1**	9.8	7.9	6.4	7.1
Bi-Cosine sphere (3)	**6.1**	8.6	7.1	6.4	6.4
Average	**7.1**	9.1	7.8	7.3	7.5

as (the modified) UHV-ADAM is not designed to improve an entire population. A reason for this is because UHV-ADAM is only applied on 3 solutions. Furthermore, concentrated gradient calls suffer more from diminishing returns compared to calls that are evenly spread over the population.

In Problem 1 of figure 4.3, a drop in MO-evaluations by the improvement metrics can be observed. Only CountUHVImproved does not experience a decrease in MO-evaluations. It is unknown why these drops happen. Analysis however once again show that like in Problem 2, reliability is not the issue. The drop in MO-evaluations could be related to UHV-ADAM overtaking UHV-GOMEA in terms of efficiency after $n \geq 10$. Regardless of the observed peculiarities, table 4.2 shows that the scores obtained by the improvement metrics in Problem 1 are rather similar. CountUHVImproved takes the lead as it is able to reach the target HV with a much lower median at $n = 20$, while not performing worse than the other improvement metrics for the other dimensions.

In Problem 2 of figure 4.3, ΔBestUHV and CountBestUHVImproved perform the best, where the former improvement metric takes a slight lead over the latter in table 4.2.

In table 4.2, the results of Problem 3 show that ΔBestUHV is slightly better than CountBestUHVImproved and Bosman2012*. Interestingly, in the normalized call graph (figure 4.4), ΔBestUHV and CountBestUHVImproved both execute gradient calls at the end, while Bosman2012* never enters the waiting state. Although different call strategies are observed, all three improvement metrics obtain similar scores.

Overall, ΔBestUHV has the best average performance according to table 4.2. It is however, not the best improvement metric for all problems, as can be seen from Problem 1. This suggests that the best improvement metric could be problem dependent. ΔBestUHV however, will continue to be used in the experiments that follow, due to having the best average performance.

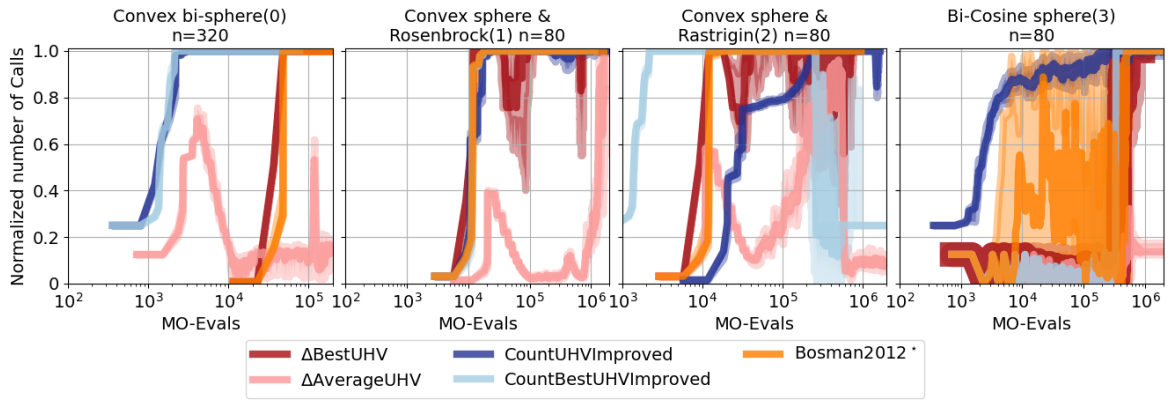


Figure 4.4: The normalized number of calls (utilization rate) per improvement metric for Problem 0 ($n = 320$) and Problem 1 to 3 ($n = 80$). The number of executed calls are normalized by the maximum number of gradient calls allowed by the resource allocation scheme.

4.2.2. Analysis of large population sizes in Problem 0 of experiment 1

In Problem 0 of experiment 1, 3 out of 5 improvement metrics have their associated best population size change from $N = 40$ to $N = 1280$. Understanding the substantial increase in population size provides key insights into the consequences of design choices. Figure 4.3 showed that UHV-ADAM is magnitudes faster than UHV-GOMEA, as the gradients come for free. Intuitively, UHV-GOMEA's contribution to the search should thus be kept at a minimum. Increasing the population size however contradictingly makes UHV-GOMEA calls more expensive.

Figure 4.5 shows the convergence in uncrowded hypervolume (UHV) of improvement metric ΔBestUHV for problem dimension $n = 320$. The convergence, defined as: $\Delta\text{UHV}(\mathcal{A}_p) = \text{HV}(\mathcal{A}_p^\star) - \text{UHV}(\mathcal{A}_p)$, is the difference between the target HV and the UHV of the best found individual. Figure 4.5 shows a single run of every population size used during the optimization, such that individual calls can be observed. The legend of figure 4.5 further shows two colors for each population size. The colors from left to right represent a UHV-GOMEA call and UHV-ADAM call respectively. Region **1** of figure 4.5 illustrates an example of individual calls.

There are numerous reasons why a large population can be desirable for Problem 0. In this analysis two important reasons will be described. Inspecting all population sizes except $N = 1280$ shows that all population sizes show repeating horizontal patterns during convergence. Annotation **2** shows examples of regions that stall. Furthermore, it can be observed that all population sizes need multiple generations to converge. Inspecting $N = 1280$, only two horizontal patterns are visible during convergence. Furthermore, it can be observed that $N = 1280$ only needs 2 generations to converge. The first generation can be observed at annotation **3**, where the first UHV-GOMEA and first 10 UHV-ADAM calls are executed. At annotation **4**, the last generation can be observed. An important reason that makes $N = 1280$ converge the fastest among the optimized population sizes is that stalling does not occur. $N = 1280$ is perceived not to stall because of the convergence criterion. Stalling is likely to occur if the convergence criterion becomes more strict, e.g. when more digits of accuracy are required. The second reason is that a large population size allows more gradient calls to be executed consecutively, as the cap on gradient calls is determined by the population size. Since only 2 generations are executed by $N = 1280$, $N = 1280$ effectively spends less resources warming up the parameters of UHV-ADAM, making its UHV-ADAM calls more efficient compared to other population sizes.

Further investigating why stalls occur reveals potential methods to improve future hybrid algorithms. Region **5** shows a region that stalls. In this region, UHV-ADAM calls are still observed. Contrary to region **5**, for $N = 1280$ at annotation **3**, shows that UHV-ADAM is easily able to improve the UHV. The reason UHV-ADAM is unable to improve the UHV as fast as $N = 1280$ for populations other than $N = 1280$, can be mainly attributed to the step size estimation. The resource allocation scheme resets UHV-ADAM instances after each UHV-GOMEA call to ensure that UHV-ADAM's internal parameters remain tuned to its local search space surroundings. After resetting the parameters, the step size of a UHV-ADAM instance must be re-estimated. Re-estimation happens based on the worst solutions. During convergence, the best solutions quickly move further and further from the worst solutions causing the estimated step sizes to become larger and larger until UHV-ADAM stalls. UHV-GOMEA needs to shrink the gap between the best and worst solutions for UHV-ADAM to become effective again. During this process, UHV-GOMEA is unable to simultaneously improve the best solutions causing UHV-GOMEA to appear as if it is not making any progress. Furthermore, the stalls are worsened by the fact that UHV-ADAM is not entirely stalling. Small improvements are still made by UHV-ADAM, as can be observed in region 6. This causes the resource allocation scheme to assign maximum resources to UHV-ADAM, as UHV-GOMEA makes no progress at all, prolonging the stalls.

In short, the analysis of large population sizes of Problem 0 has shown that substantial stalls occur during convergence. Large population sizes can converge faster than small population sizes if convergence occurs before stalling sets in. Furthermore, stalling can be mainly attributed to the step size estimation method. Improving the step size estimation method could potentially improve convergence.

4.3. Experiment 2: Comparison between gradient algorithms

To investigate the effect of different gradient algorithms, two UHV-based gradient algorithms are constructed. The first algorithm, dubbed PlainGradient, uses the raw UHV gradient without any other mechanism except the step size reduction scheme of UHV-ADAM. The second gradient algorithm, dubbed LineApproximation, uses the same raw UHV gradient, but does an approximation of what a

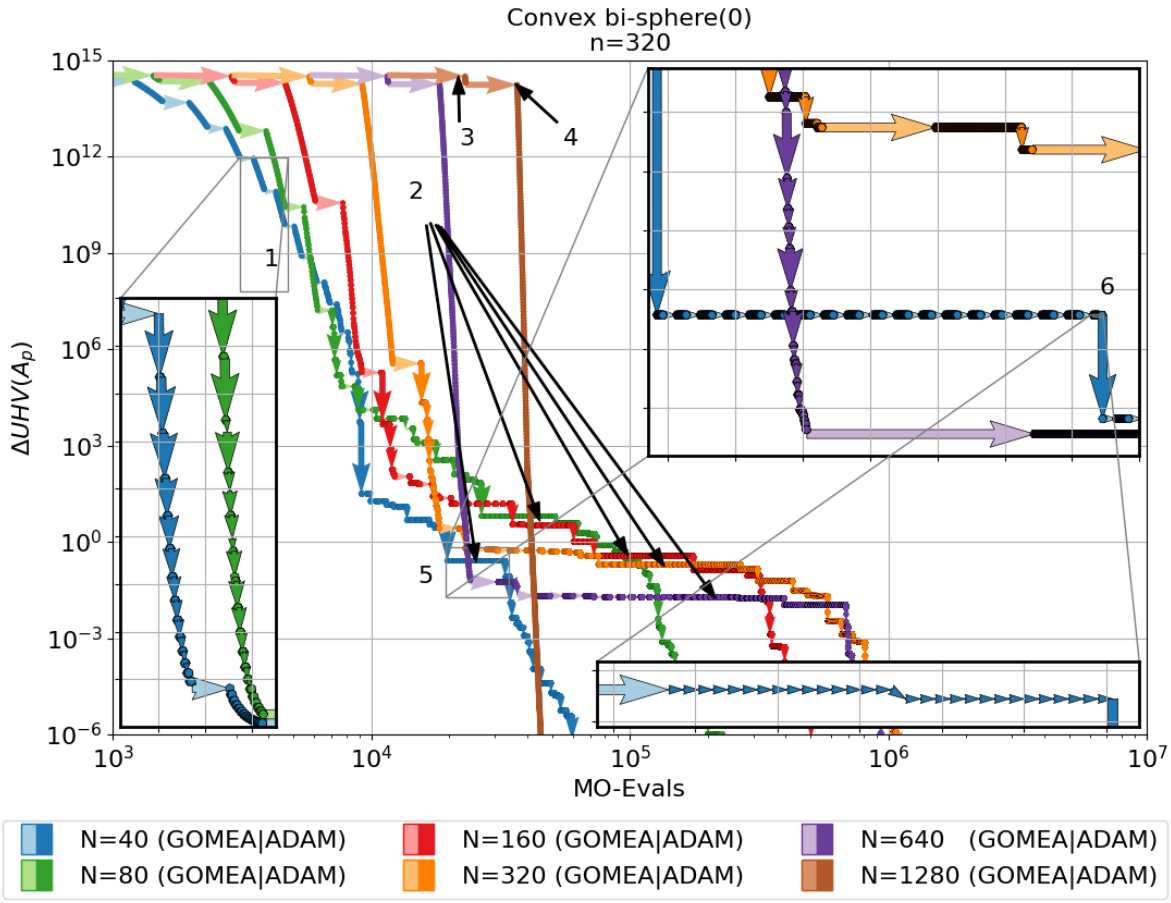


Figure 4.5: Convergence of ΔBestUHV for different population sizes, where $n = 320$ (Problem 0). A single run is displayed for each population size such that each individual call can be observed.

line search algorithm does by sampling at multiple distances. If distance 1 is the sample that would have been obtained by PlainGradient, then LineApproximation samples at distances $[0.01, 0.05, 0.1, 0.5, 0.99, 1, 1.5, 2]$. The sample distance factor that produces the solution set with the highest UHV is picked. The step size reduction scheme of UHV-ADAM is not needed for LineApproximation, as the sample distance that is chosen can be directly used to update γ.

4.3.1. Results of experiment 2

The effect of different gradient algorithms on the required number of MO-evaluations to reach a target accuracy of $\Delta\text{HV}_p < 10^{-6}$ and the corresponding success rate is shown in figure 4.6. The gradient algorithms are optimized on the initial γ estimation factor (percentage of average population range) and γ decay rate (rate of reducing γ when the solution does not improve) when applicable. The optimized parameters are included in appendix A. The optimized population size of experiment 1 is used. Furthermore, all three hybrids use the ΔBestUHV improvement metric.

In Problem 0 of figure 4.6, UHV-ADAM is the worst choice amongst the gradient algorithms. One reason lies with the fact that other (more important) step size parameters of UHV-ADAM are not optimized, e.g. the influence factor of the variance. The step size of the other algorithms are more easily influenced and over optimized for Problem 0 as the step size only depends on the parameters that are currently optimized. In figure 4.6, Problem 1 shows that both Hybrid-PlainGradient and Hybrid-LineApproximation do not converge within budget around $n \geq 20$. A possible hypothesis is that it becomes increasingly more difficult to follow the curvatures of the Rosenbrock problem without any mechanisms suited for the task. UHV-ADAM on the other hand is able to follow the curvature. In Problem 2, Hybrid-ADAM is clearly the most efficient gradient algorithm. In Problem 3, a clear difference can not be seen between Hybrid-ADAM and Hybrid-PlainGradient. Analysis shows that call graphs,

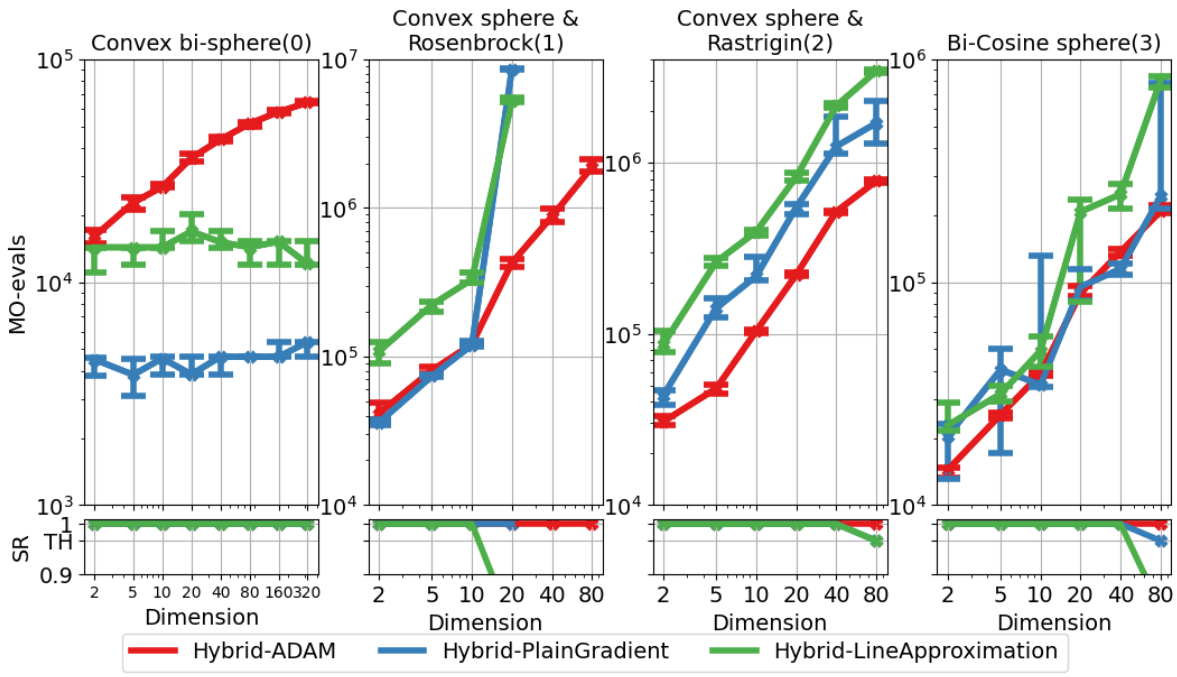


Figure 4.6: The effect of different gradient algorithms on the required number of MO-evaluations to reach a target accuracy of $\Delta HV_p < 10^{-6}$ for various problems where $p = 9$. The success rate (SR) measures the fraction of runs that reach the target accuracy out of 30 runs, where the target threshold is 29 out of 30 runs.

similar to figure 4.4 of experiment 1 were obtained, where the selected improvement metric does not execute many gradient calls. Overall, picking UHV-ADAM, with all its intricate mechanics, seem to be preferable over basic gradient algorithms when more difficult problems are presented.

4.4. Experiment 3: Distribution method of UHV-ADAM

After the resource allocation scheme assigns resources to the modified UHV-ADAM, the resources need to be converted to calls that are applied on the individuals of the population. In the third experiment, the effects of different distribution methods of section 3.1.3 are investigated.

4.4.1. Results of experiment 3

The effect of the choice of method to distribute the resources assigned to the modified UHV-ADAM, on the required number of MO-evaluations to reach a target HV accuracy of $\Delta HV_p < 10^{-6}$ and the corresponding success rate (SR) is shown in figure 4.7. Table 4.3 shows the scores obtained by the distribution methods. Distribution methods that are unable to find a population size that meets the success rate (SR) threshold are disqualified and denoted as: “DQ“. The evaluation budget, population optimization method and solution set size are the same as in experiment 1. The population is retuned as some distribution methods are dependent on the population size. ΔBestUHV is once again the selected improvement metric.

Figure 4.7 shows various implementations of the Best-m-Solutions and Best-m%Population distribution methods. Missing from the legend are the distribution methods that have m values of $[10, 20]$ and $[20(\%), 35(\%), 50(\%)]$ for the Best-m-Solutions and Best-m%Population distribution methods respectively. These distribution methods are not included in figure 4.7 as their performances are equal or worse than distribution methods Best5Solutions and Best10%Solutions respectively.

In Problem 0 of figure 4.7, it should not come as a surprise that BestSolution reaches the target HV the most efficiently. UHV-ADAM, which outperforms all other algorithms in figure 4.3 of experiment 1, only optimizes a single solution set. Selecting a similar distribution method yields similar results. Distribution method Best3Solutions comes in second place.

In Problem 1, BestSolution is overall, once more the most efficient at reaching the target HV. Redistribution methods: ALL, Random, Best5%Population and Best10%Population fail to reach the target

HV within budget or target success rate when $n = 80$ in figure 4.7. The distribution methods that are not disqualified, excluding BestSolution, have scores that are substantially worse than BestSolution in table 4.3. In experiment 1, it was observed that UHV-ADAM is able to efficiently solve Problem 1 if it is initialized away from local optima. For BestSolution, UHV-GOMEA might be able to escape local optima easily in Problem 1, while BestSolution leverages the same principle of UHV-ADAM, i.e. applying calls on a single solution set; hence superior performance is observed. In figure 4.7 however, BestSolution intersect with Best2.5%Population when $n = 80$, raising concerns that BestSolution could perform worse for higher dimensions. Furthermore, it is concerning that the population size becomes smaller when $n \geq 20$. It is unknown why this happens. Interestingly, Best2.5%Population and Best3Solutions, seem to perform better than Best5Solutions or Best10%Population when $n \geq 40$. The opposite is true however when $n \leq 10$. No explanation could be found for this observation.

In Problem 2, Best3Solutions performs the best according to table 4.3. It is followed by BestSolution and Best5Solutions. Interestingly, in previous problems All and Random performed similarly, however in Problem 2, $n = 80$, All is disqualified for failing to reach the target success rate while Random is not.

In Problem 3, Best3Solutions performs the best according to table 4.3. Analysis once again shows that ΔBestUHV, does not execute many gradient calls as was observed in figure 4.4; hence similar performance is observed. An interesting observation is that BestSolution picks a large population at $n = 80$. It is unknown why this happens.

Overall, according to table 4.3, Best3Solutions has a slight lead over BestSolution followed by Best5Solutions and Best2.5%Population. Furthermore, table 4.3 shows that concentrating gradient calls on the best solutions works better than diluting gradient calls over the population.

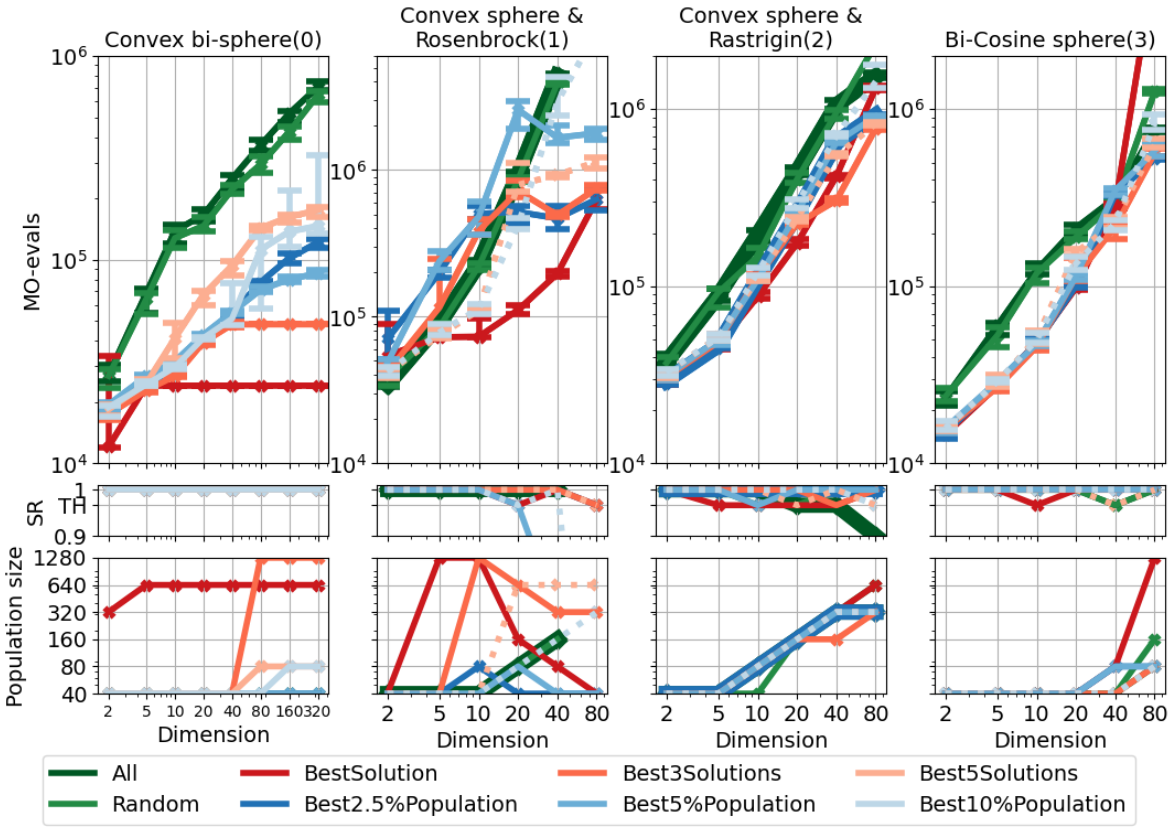


Figure 4.7: The effect of the choice of method to distribute the resources assigned to the modified UHV-ADAM on the required number of MO-evaluations to reach a target accuracy of $\Delta HV_p < 10^{-6}$ for various problems where $p = 9$. The success rate (SR) measures the fraction of runs that reach the target accuracy out of 30 runs, where the target threshold (TH) is 29/30 runs. The bottom row shows the optimized population size.

Table 4.3: The scores assigned to each distribution method. The lower the score, the better. The numbers in bold are the lowest scores of a category. Improvement metrics that do not meet one or more success rate threshold are denoted as disqualified (DQ).

Problem	All	Random	Best Solution	Best3 Solutions	Best5 Solutions	Best2.5% Population	Best5% Population	Best10% Population
Convex sphere & Rosenbrock(1)	DQ	DQ	**6.6**	18.3	17.4	19.7	DQ	DQ
Convex sphere & Rastrigin(2)	DQ	13.4	7.1	**6.7**	7.7	8.1	8.2	9.4
Bi-Cosine sphere(3)	11.6	11.9	17.8	**6.3**	7.1	6.7	7.1	7.2
Average	DQ	DQ	10.5	**10.4**	10.7	11.5	DQ	DQ

4.5. WFG Benchmark

We use the WFG suite [20] as an independent method to benchmark the results of the hybrid algorithm. For detailed characteristics of these 9 benchmark functions, the reader is referred to [20]. WFG1 is a separable problem, with a flat region which can stagnate the search. WFG2 has a uni-modal disconnected convex front. WFG3 is multi-modal and has a linear front. WFG4-9 all have concave fronts, where WFG4 and WFG9 are multi-modal. Following [10] and [9], the benchmark is used in a bi-objective setting with $k_{\text{WFG}} = 4$ position variables and $l_{\text{WFG}} = 20$ distance variables, resulting in $n = 24$ decision variables. The HV reference point $\mathbf{r}$ is set to $r = (11, 11)$. The computation budget is set to 10^7 MO-evaluations for each algorithm, where snapshots are taken at 10^5 and 10^7 MO-evaluations, to get comparable experiments as in [10] (10^5 MO-evaluations) and [9] (10^7 MO-evaluations). For the solution set size, we use $p = 9$.

The algorithms we consider in this experiment include the base algorithms: UHV-GOMEA(Lm), UHV-ADAM, the constructed hybrid algorithms: ΔBestUHV with distribution methods BestSolution and Best3Solutions, as well as other UHV-based algorithms: UHV-GOMEA with a linkage tree (Lt) and UHV-GA-MO [10]. Concerning the hybrid algorithms, Best3Solutions and BestSolution are included as in experiment 3, both distribution methods obtained average scores that were rather similar. Furthermore, concerns were raised that BestSolution might scale worse than Best3Solutions. For the external algorithms, UHV-GA-MO and UHV-GOMEA(Lt) are included to create a similar experimental set-up as in [10] and [9]. UHV-GA-MO is based on the GA-MO scheme [23]. UHV-GA-MO uses the same UHV indicator function as UHV-ADAM, but replaces the modified ADAM gradient ascent scheme with a modified GA-MO gradient ascent scheme. UHV-GOMEA(Lt) is similar to UHV-GOMEA(Lm), but a linkage tree is used to estimate and adapt Gaussians instead of the marginal product linkage model. More on the linkage tree can be found in section 3.2.2.

Each algorithm is executed 30 times. Algorithms that use populations have their population sizes set to 200 following [9], [10]. Gradient-based algorithms use finite difference gradient approximations and will be indicated by the suffix "-FD". Finite difference approximations come at the cost of $(1 + n) \cdot p$ MO-evaluations [10]. Differences of outcomes are compared to the best result and tested for statistical significance up to 4 decimals by a Wilcoxon two-sided rank-sum test with $\alpha = 0.05$.

4.5.1. Results of the WFG Benchmark

Table 4.4 shows the results for 10^5 MO-evaluations and table 4.5 shows the results for 10^7 MO-evaluations. In Table 4.4, substantially different results than [10] are obtained for UHV-ADAM-FD and UHV-GA-MO-FD. The reasons for this is that a bug was detected by me. This work uses the implementations where the bug is resolved.

In Table 4.4, on average the hybrids are performing the best, where Hybrid-BestSolution-FD is performing better than Hybrid-Best3Solutions-FD. With the exception of WFG4, Hybrid-BestSolution-FD never obtains a rank worse than 2. Hybrid-Best3Solutions-FD however, from time to time ranks second to last. Interestingly, in multi-modal problems WFG3, WFG4 and WFG9, Hybrid-BestSolution-FD (and to an extent Hybrid-Best3Solutions-FD) obtains similar or better results than the evolutionary algorithms. This indicates that the gradient techniques are at worst not slowing down convergence, but in some cases can be beneficial in the search. In WFG3 however, Hybrid-BestSolution-FD is performing worse than UHV-ADAM-FD.

Table 4.4: Snapshot taken of the WFG benchmark for 10^5 MO-evaluations. Hypervolume values are shown (mean, $\pm$ standard deviation(rank)). Finite differences (FD) are used for the gradient-based algorithms. Scores in bold are the best or not statistically different from the other bold scores indicated per problem.

Problem	UHV-GOMEA(Lm)	UHV-GOMEA(Lt)	UHV-ADAM-FD	UHV-GA-MO-FD	Hybrid-BestSolution-FD	Hybrid-Best3Solutions-FD
WFG1	85.67±0.29(5)	85.29±0.31(6)	96.97±0.36(3)	96.02±2.35(4)	**97.77**±0.22(1)	97.66±0.22(2)
WFG2	109.22±0.21(2)	109.21±0.26(3)	106.24±5.09(6)	**109.20**±6.79(4)	**109.87**±0.19(1)	108.07±0.94(5)
WFG3	115.01±0.22(4)	115.20±0.17(3)	**116.48**±0.00(1)	114.63±0.37(6)	116.37±0.06(2)	114.90±0.41(5)
WFG4	108.26±1.26(3)	**108.88**±0.56(2)	103.34±3.61(6)	106.72±0.90(5)	**108.03**±2.66(4)	**108.90**±0.77(1)
WFG5	102.57±0.66(6)	102.64±0.83(5)	108.78±1.87(4)	109.30±1.97(3)	**110.97**±0.94(1)	109.66±0.59(2)
WFG6	108.81±0.84(6)	108.99±1.11(5)	**113.73**±0.09(1)	109.43±2.12(4)	**113.70**±0.11(2)	113.51±0.13(3)
WFG7	112.46±0.51(6)	112.70±0.46(5)	**114.35**±0.04(1)	113.47±0.47(4)	114.34±0.03(2)	113.84±0.20(3)
WFG8	109.04±0.26(6)	109.07±0.24(5)	**110.38**±0.83(2)	109.24±1.25(3)	**110.45**±0.86(1)	109.16±0.64(4)
WFG9	108.14±0.53(3)	107.95±0.57(4)	106.70±1.53(5)	102.83±5.15(6)	**108.72**±0.67(1)	108.22±0.54(2)
Rank	4.56(6)	4.22(4)	3.22(3)	4.33(5)	1.67(1)	3.00(2)

Table 4.5: Snapshot taken of the WFG benchmark for 10^7 MO-evaluations.Hypervolume values are shown (mean, $\pm$ standard deviation(rank)). Finite differences (FD) are used for the gradient-based algorithms. Scores in bold are the best or not statistically different from the other bold scores indicated per problem.

Problem	UHV-GOMEA(Lm)	UHV-GOMEA(Lt)	UHV-ADAM-FD	UHV-GA-MO-FD	Hybrid-BestSolution-FD	Hybrid-Best3Solutions-FD
WFG1	94.63±1.73(6)	99.66±2.06(2)	97.32±0.60(4)	96.74±0.60(5)	98.90±0.29(3)	**101.57**±0.49(1)
WFG2	110.13±0.03(4)	110.14±0.03(3)	106.26±5.09(6)	**109.60**±6.68(5)	110.36±1.20(2)	**110.84**±2.04(1)
WFG3	**116.50**±0.00(5)	**116.50**±0.00(1)	**116.50**±0.00(1)	114.78±0.33(6)	**116.50**±0.00(1)	**116.50**±0.00(1)
WFG4	112.75±0.58(4)	113.38±0.32(3)	103.34±3.61(6)	107.21±0.97(5)	113.46±0.35(2)	**114.02**±0.13(1)
WFG5	**112.19**±0.10(5)	**112.22**±0.01(1)	112.21±0.03(4)	111.32±0.68(6)	112.22±0.00(2)	112.22±0.00(3)
WFG6	114.38±0.03(3)	114.38±0.04(4)	113.79±0.10(5)	110.52±2.00(6)	**114.40**±0.00(1)	114.40±0.00(2)
WFG7	**114.40**±0.01(4)	**114.40**±0.00(1)	114.37±0.03(5)	113.88±0.16(6)	114.40±0.00(3)	**114.40**±0.00(1)
WFG8	111.43±0.28(4)	111.54±0.23(3)	110.57±0.81(5)	109.48±1.06(6)	111.70±0.23(2)	**111.82**±0.01(1)
WFG9	111.46±0.16(4)	**111.50**±0.03(2)	107.54±1.10(5)	103.18±5.31(6)	111.49±0.03(3)	**111.51**±0.02(1)
Rank	4.33(4)	2.22(3)	4.56(5)	5.67(6)	2.11(2)	1.33(1)

In Table 4.5, on average the hybrids are once again performing the best. Hybrid-Best3Solutions-FD however is performing better than Hybrid-BestSolution-FD in Table 4.5. This could suggest that Hybrid-BestSolution-FD is more likely to end up with sub-optimal solutions or converges slower when a bigger computation budget is made available. Interestingly, rank-wise, the hybrids never perform worse its component algorithms, i.e. UHV-GOMEA(Lm) and UHV-ADAM-FD. Furthermore, only in WFG3 do the hybrids share a rank with UHV-ADAM. Another observation is that for the multi-modal problems (WFG3, WFG4 and WFG9) the hybrids not only obtain competitive results compared to the EAs, but occasionally find better HVs. In WFG5, WFG6 and WFG7, the hybrids find HVs that are deemed statistically different from the best found HV in four instances, even though similar means and standard deviations are reported. The reason for this lies with the Wilcoxon test, which by default removes runs that have the same HV. For example, in WFG5, 22 out of 30 equal HV pairs were observed when comparing Hybrid-BestSolution-FD and UHV-GOMEA(Lt). In WFG 6 however, Hybrid-Best3Solutions-FD most likely remains statistically different from Hybrid-BestSolution-FD as only 12 equal HV pairs were observed. Overall, the hybrid algorithms produce competitive or better results than its component algorithms and other comparable HV-optimizing algorithms. The performance of the hybrids tested in this experiment however seem to fluctuate depending on the computation budget available.

5

Discussion

A real-valued multi-objective (MO) hybrid algorithm was created by combining two uncrowded hypervolume (UHV) indicator-based algorithms via a dynamic resource allocation scheme. In Experiment 1 it was shown that for UHV optimization, picking an improvement metric is not trivial, as problem dependency has been observed. The results of experiment 1 however, also showed that if the hybrid is tasked to do UHV optimization, on average it benefits most from using the ΔBestUHV improvement metric, followed by CountBestUHVImproved. Both metrics quantify the improvement of the best UHV, while the remaining metrics (Bosman2012, CountUHVImproved, ΔAverageUHV) measure the improvement over all solution sets. This opens the question why resource allocation towards the algorithms which improve fewer solution sets with higher UHV is preferable over the full runtime of the hybrid.

Experiment 1 also showed cases where UHV-GOMEA or the hybrid algorithms of experiment 1, experienced a significant reduction in the number of MO-evaluations needed to solve a problem when the problem dimensionality grows. No explanation could be found for this phenomenon. Analyzing these peculiarities in future work could obtain key insights into the mechanics of the respective algorithms.

One peculiarity of experiment 1, where large population sizes were used by the hybrids to solve the Bi-sphere problem, was analyzed however. Analysis showed that the hybrid algorithms frequently stall during convergence, due to an inaccurately estimated step size of UHV-ADAM. In the current setup, only UHV-GOMEA is able to correct the step size. This “bottleneck“ however exacerbates the longevity of stalls. When UHV-GOMEA is unintentionally improving the step size estimation, it is unable to improve the best solutions simultaneously. UHV-ADAM meanwhile is able to make tiny improvements. As a consequence, the resource allocation scheme allocates a substantial amount of resources to UHV-ADAM while it would be better not to, until an appropriate step size is obtained. Many solutions to reduce the impact of the inaccurate estimate of the step size can be constructed. An interesting approach would be to estimate the step size using the solution on which gradient calls are applied to keep the estimation more local. Preventing the resource allocation scheme from prematurely assigning resources to UHV-ADAM while stalling however, is more difficult to solve. It is difficult to determine whether or not UHV-ADAM is stalling or is actually finding small improvements.

Experiment 1 also provided additional insight in the properties of UHV-ADAM. It confirms that problems with few local optima (e.g. Convex sphere & Rosenbrock) can be solved by UHV-ADAM, while problems with many local optima (e.g. Convex sphere & Rastrigin) are not solvable.

In experiment 2, it was shown that choosing the appropriate gradient algorithm, with the correct mechanics, remains vital to ensure convergence towards the Pareto set. It was also shown that optimizing the correct parameters of the gradient algorithms can have substantial effects on the rate of convergence. In future work, more care needs to be taken to optimize the appropriate parameters of the gradient algorithms.

Experiment 3 has shown that concentrating gradient calls on a select number of solutions is preferred over diluting calls over the entire population. Distributing resources to the solutions with the top 3 highest UHV performed the best on average.

One of the limitations of this work is that the problems that were used to tune the hybrid, all share the commonality of having a connected Pareto set. A connected Pareto set simplifies finding all other Pareto optimal solutions as soon as one solution has been determined. If one of the objectives then

happens to be easily solvable (e.g. Sphere), it potentially creates situations where even algorithms that are not suited to solve multi-modal problems can still find the Pareto set by first solving the easy objective before moving over to the other objective, bypassing any local optimum. Future work should thus consider disconnected Pareto sets.

Another limitation of this work, is that only a single EA, i.e. UHV-GOMEA, has been selected for hybridization. In [9], it was already observed that domination-based EA MO-RV-GOMEA [13] initially performs better than UHV-GOMEA. An even more efficient hybrid algorithm could potentially be created with MO-RV-GOMEA. However, as MO-RV-GOMEA is a domination-based EA, compatibility issues are likely to occur with UHV-based algorithms. Introducing a different EA could furthermore expand the number of experiments, such that the robustness can be tested of the resource allocation scheme.

The final limitation of this work that will be mentioned, is that there is not enough emphasis placed on finite difference approximations. [10] has shown cases where UHV-ADAM converges slower than UHV-GOMEA if finite difference approximations are used. Although this was not observed in the WFG benchmark, in early experiments similar observations were made if UHV-ADAM and UHV-GOMEA were initialized without enclosing the Pareto set. For this reason, more finite difference cases should thus be studied.

Overall, the hybrid that has been created has shown that is able to obtain competitive or better results than its components algorithms, especially when the gradients come for free. The naive lesson to be learned from this thesis is that a variety of algorithms should be hybridized as soon as possible as the potential of hybrids has been shown. However, we must be reminded that in every experiments executed in this thesis, peculiarities were observed. This shows that constructing a hybrid algorithm is not easy. Not only do we need to tune the resource allocation process as is seen in experiment 1, where the improvement metric is problem dependent, we also need to pick the right algorithms, as was shown in experiment 2. I believe that the lesson to be learned from this thesis, is that every algorithm or resource allocation method that is combined into a hybrid algorithm, needs to be picked carefully and requires tuning. Each algorithm brings its own parameters with it. These parameters not only bring another point of failure into an algorithm and thus require careful tuning, but also, the algorithms must be tuned carefully to each other. Nevertheless, the complexity of hybrid algorithms should not halt the development of hybrid algorithms. Hybrids remain a promising addition the spectrum of evolutionary algorithms.

6

Conclusion

In this work, for the first time, a multi-objective optimization algorithm was introduced that hybridizes an uncrowded hypervolume-based (UHV) evolutionary algorithm with a UHV-based gradient algorithm via a dynamic resource allocation scheme (RAS). Experiments used to study the RAS showed that selecting a reward metric for the RAS is not trivial as it was observed that the best metric is problem dependent. Furthermore, the selection and tuning of gradient algorithms remains vital for convergence. Ill-suited gradient algorithms can deceive the resource allocation scheme, causing the hybrid to converge slower than the original algorithms. Experiments also showed that concentrating gradient steps on a select number of solutions of the population, outweighs dispersing gradient steps over the entire population. Implementations of the hybrid algorithm have also been compared to other UHV-based algorithms. It was shown that even if finite difference approximations are used to calculate gradients, it is still able to obtain competitive or better results than the original component algorithms as well as other UHV-based algorithms. We conclude that the resulting hybrid is therefore a promising addition to the existing spectrum of evolutionary algorithms for multi-objective optimization.

A

Experiment 2: Gradient Parameters

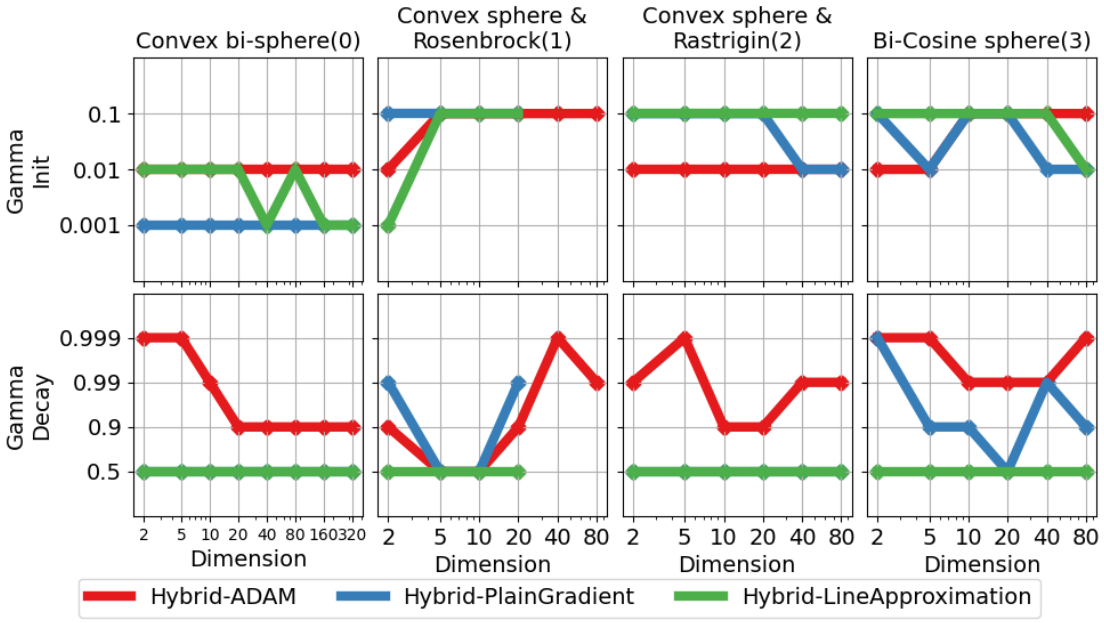


Figure A.1: The initial gamma estimation factor and gamma decay factor of various problems of experiment 2, where $p = 9$. Please note that y-axis is non-linear.

Bibliography

[1] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: Nsga-ii," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 2, pp. 182–197, 2002. doi: 10.1109/4235.996017.

[2] A. Bouter, T. Alderliesten, C. Witteveen, and P. A. N. Bosman, "Exploiting linkage information in real-valued optimization with the real-valued gene-pool optimal mixing evolutionary algorithm," in *Proceedings of the Genetic and Evolutionary Computation Conference*, ser. GECCO '17, Berlin, Germany: Association for Computing Machinery, 2017, pp. 705–712, isbn: 9781450349208. doi: 10.1145/3071178.3071272. [Online]. Available: https://doi.org/10.1145/3071178.3071272.

[3] S. Sharma, J. Blank, K. Deb, and B. K. Panigrahi, "Ensembled crossover based evolutionary algorithm for single and multi-objective optimization," in *2021 IEEE Congress on Evolutionary Computation (CEC)*, 2021, pp. 1439–1446. doi: 10.1109/CEC45853.2021.9504698.

[4] K. Deb and D. Kalyanmoy, *Multi-Objective Optimization Using Evolutionary Algorithms*. USA: John Wiley & Sons, Inc., 2001, isbn: 047187339X.

[5] P. Bosman and D. Thierens, "The balance between proximity and diversity in multiobjective evolutionary algorithms," *IEEE Transactions on Evolutionary Computation*, vol. 7, no. 2, pp. 174–188, 2003. doi: 10.1109/TEVC.2003.810761.

[6] R. Berghammer, T. Friedrich, and F. Neumann, "Convergence of set-based multi-objective optimization, indicators and deteriorative cycles," *Theoretical Computer Science*, vol. 456, pp. 2–17, 2012, issn: 0304-3975. doi: https://doi.org/10.1016/j.tcs.2012.05.036. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0304397512005300.

[7] E. Zitzler and L. Thiele, "Multiobjective evolutionary algorithms: A comparative case study and the strength pareto approach," *IEEE Transactions on Evolutionary Computation*, vol. 3, no. 4, pp. 257–271, 1999. doi: 10.1109/4235.797969.

[8] M. Emmerich, A. Deutz, and N. Beume, "Gradient-based/evolutionary relay hybrid for computing pareto front approximations maximizing the s-metric," in *Hybrid Metaheuristics*, T. Bartz-Beielstein, M. J. Blesa Aguilera, C. Blum, B. Naujoks, A. Roli, G. Rudolph, and M. Sampels, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 140–156, isbn: 978-3-540-75514-2.

[9] S. C. Maree, T. Alderliesten, and P. A. N. Bosman, *Uncrowded hypervolume-based multi-objective optimization with gene-pool optimal mixing*, 2020. arXiv: 2004.05068 [cs.NE].

[10] T. M. Deist, S. C. Maree, T. Alderliesten, and P. A. N. Bosman, *Multi-objective optimization by uncrowded hypervolume gradient ascent*, 2020. arXiv: 2007.04846 [math.OC].

[11] P. A. N. Bosman, "On gradients and hybrid evolutionary algorithms for real-valued multiobjective optimization," *IEEE Transactions on Evolutionary Computation*, vol. 16, no. 1, pp. 51–69, 2012. doi: 10.1109/TEVC.2010.2051445.

[12] V. A. Sosa Hernández, O. Schütze, H. Wang, A. Deutz, and M. Emmerich, "The set-based hypervolume newton method for bi-objective optimization," *IEEE Transactions on Cybernetics*, vol. 50, no. 5, pp. 2186–2196, 2020. doi: 10.1109/TCYB.2018.2885974.

[13] A. Bouter, N. Luong, C. Witteveen, T. Alderliesten, and P. Bosman, "The multi-objective real-valued gene-pool optimal mixing evolutionary algorithm," English, in *GECCO 2017 - Proceedings of the 2017 Genetic and Evolutionary Computation Conference*, GECCO 2017 : Genetic and Evolutionary Computation Conference ; Conference date: 15-07-2017 Through 19-07-2017, Jul. 2017, pp. 537–544. doi: 10.1145/3071178.3071274. [Online]. Available: http://gecco-2017.sigevo.org/index.html/HomePage.

[14] E. Zitzler, L. Thiele, M. Laumanns, C. M. Fonseca, and V. G. da Fonseca, “Performance assessment of multiobjective optimizers: An analysis and review,” *IEEE Transactions on Evolutionary Computation*, vol. 7, no. 2, pp. 117–132, 2003. doi: 10.1109/TEVC.2003.810758.

[15] A. Auger, J. Bader, D. Brockhoff, and E. Zitzler, “Theory of the hypervolume indicator: Optimal μ-distributions and the choice of the reference point,” Jan. 2009.

[16] D. P. Kingma and J. Ba, *Adam: A method for stochastic optimization*, 2017. arXiv: 1412.6980 [cs.LG].

[17] H. Wang, A. Deutz, T. Bäck, and M. Emmerich, “Hypervolume indicator gradient ascent multi-objective optimization,” in *9th International Conference on Evolutionary Multi-Criterion Optimization - Volume 10173*, ser. EMO 2017, Münster, Germany: Springer-Verlag, 2017, pp. 654–669, isbn: 9783319541563. doi: 10.1007/978-3-319-54157-0_44. [Online]. Available: https://doi.org/10.1007/978-3-319-54157-0_44.

[18] I. Gronau and S. Moran, “Optimal implementations of upgma and other common clustering algorithms,” *Information Processing Letters*, vol. 104, no. 6, pp. 205–210, 2007, issn: 0020-0190. doi: https://doi.org/10.1016/j.ipl.2007.07.002. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0020019007001767.

[19] P. Bosman and E. de Jong, “Combining gradient techniques for numerical multi-objective evolutionary optimization,” vol. 1, Jan. 2006, pp. 627–634. doi: 10.1145/1143997.1144111.

[20] S. Huband, L. Barone, L. While, and P. Hingston, “A scalable multi-objective test problem toolkit,” vol. 3410, Jan. 2005, pp. 280–295, isbn: 978-3-540-24983-2. doi: 10.1007/978-3-540-31880-4_20.

[21] P. A. N. Bosman, J. Grahl, and D. Thierens, “Benchmarking parameter-free amalgam on functions with and without noise,” *Evolutionary Computation*, vol. 21, pp. 445–469, 2013.

[22] F. Hoffmeister and T. Bäck, “Genetic algorithms and evolution strategies - similarities and differences,” in *PPSN*, 1990.

[23] H. Wang, A. Deutz, T. Bäck, and M. Emmerich, “Hypervolume indicator gradient ascent multi-objective optimization,” in *9th International Conference on Evolutionary Multi-Criterion Optimization - Volume 10173*, ser. EMO 2017, Münster, Germany: Springer-Verlag, 2017, pp. 654–669, isbn: 9783319541563. doi: 10.1007/978-3-319-54157-0_44. [Online]. Available: https://doi.org/10.1007/978-3-319-54157-0_44.