information

Article

Reinforcement Learning-Based Hybrid Multi-Objective Optimization Algorithm Design

Herbert Palm *,† **and Lorin Arndt** †

Systems Engineering Laboratory, University of Applied Sciences, Lothstrasse 64, 80335 München, Germany
* Correspondence: herbert.palm@hm.edu
† These authors contributed equally to this work.

Abstract: The multi-objective optimization (MOO) of complex systems remains a challenging task in engineering domains. The methodological approach of applying MOO algorithms to simulation-enabled models has established itself as a standard. Despite increasing in computational power, the effectiveness and efficiency of such algorithms, i.e., their ability to identify as many Pareto-optimal solutions as possible with as few simulation samples as possible, plays a decisive role. However, the question of which class of MOO algorithms is most effective or efficient with respect to which class of problems has not yet been resolved. To tackle this performance problem, hybrid optimization algorithms that combine multiple elementary search strategies have been proposed. Despite their potential, no systematic approach for selecting and combining elementary Pareto search strategies has yet been suggested. In this paper, we propose an approach for designing hybrid MOO algorithms that uses reinforcement learning (RL) techniques to train an intelligent agent for dynamically selecting and combining elementary MOO search strategies. We present both the fundamental RL-Based Hybrid MOO (RLhybMOO) methodology and an exemplary implementation applied to mathematical test functions. The results indicate a significant performance gain of intelligent agents over elementary and static hybrid search strategies, highlighting their ability to effectively and efficiently select algorithms.

Keywords: multi-objective optimization; complex systems; Pareto front; hybrid search algorithms; reinforcement learning; intelligent agent

Citation: Palm, H.; Arndt, L. Reinforcement Learning-Based Hybrid Multi-Objective Optimization Algorithm Design. *Information* **2023**, *14*, 299. https://doi.org/10.3390/info14050299

Academic Editor: Katsuhide Fujita

Received: 11 April 2023
Revised: 16 May 2023
Accepted: 18 May 2023
Published: 22 May 2023

1. Introduction

Decision making aims at identifying solutions with optimal target attributes in the space of available solution alternatives. In context of the Cynefin [1] framework, we call a problem's environment (i.e., its habitat, or in Welsh, Cynefin):

- *complicated* when there is a non-trivial number (usually more than two) of known cause–effect relations;
- *complex* when there are at least some relevant cause–effect relations that are not subject to existing experience, or that are not accessible in closed analytical form,

characterizing the problem. Thus, the degree of a problem's complicacy increases with both the dimension of the *design space* (i.e., the domain defining the set of alternative solutions by all degrees of decision freedom) and the number of cause–effect relations (connecting a domain to its co-domain, i.e., the set of achievable objectives). The degree of a problem's complexity, in contrast, increases with its degree of uncertainty about the cause–effect relationships mentioned above [2]. The complicacy and complexity of a problem can both be elevated if decision making is not focused on a single (uni-criterial) objective, but on a multitude of usually conflicting (multi-criterial) objectives. The latter case forms the basis for Multi-Objective Decision Making (MODM) [3,4] or synonymously, Multi-Criteria Decision Making (MCDM) [5], supporting decision makers faced with a multitude of objectives.

Before a multi-objective decision may be concluded, knowledge about conflicting target trade-offs is paramount. Definitions of processes, methods, and algorithms for simultaneously optimizing multiple objectives, and thereby, identifying conflicting target trade-offs, is subject to the multi-objective optimization (MOO) field [6]. For nontrivial MOO problems (i.e., problems without the existence of a single solution that simultaneously optimizes each stated objective), MOO algorithms aim to find solutions that can only be improved with respect to any objective if at least one other objective is worsened at the same time. Mathematically, this basic MOO goal is usually expressed by searching for (design) domain elements d_{opt} that correspond to the global extreme points of the (target) domain in terms of target value minimization:

$$d_{\text{opt}} = \underset{d \in X}{\arg\min}\, f(d) \tag{1}$$

within the design space domain $X \subset \mathbb{R}^d$ being mapped to the (*objective* or *target space*) co-domain T by a $t-$dimensional function

$$f\colon X \to T \subset \mathbb{R}^t. \tag{2}$$

MOO problems can be formulated in the above type of minimization without a loss of generality, since maximizing the objectives, if necessary, is equivalent to minimizing their negatives. While f is analytically known for a complicated problem, it is represented by a black box function in the case of a complex problem that in many cases is only accessible through a computationally expensive DAE (differential-algebraic system of equations)-based simulation model.

In order to deal with simultaneous multiple objectives, MOO extends the scalar valued minimization operation to the identification of a set $P \subset T$ of *non-dominated* solutions $p \in P$ (with P being called the *Pareto front*, containing all *Pareto points*), as defined by the dominance relation between two solutions $p, q \in T$

$$\begin{aligned} p \preccurlyeq q \quad &:\Longleftrightarrow \quad \forall\, i\colon \quad p_i \leq q_i \\ p \prec q \quad &:\Longleftrightarrow \quad \forall\, i\colon \quad p_i < q_i \end{aligned} \tag{3}$$

with the vector components being indicated by $i \in \{1, ..t\}$. The Pareto front of (Pareto-) optimal solutions then may be phrased as

$$P := \{p \in T \mid \nexists\, (q \in T, i \in \{1, .., t\}) : q \preccurlyeq p \wedge q_i \prec p_i\}. \tag{4}$$

In other words, the Pareto front comprises those solutions of a problem that cannot be improved without accepting a deterioration within at least one target dimension.

MOO algorithms focusing on the effective and efficient search of Pareto-optimal points follow an established set of heuristics; for example, multi-objective evolutionary algorithms (MOEAs) [7] (such as particle swarm [8], ant-colony [9], genetic algorithms [10]), or (surrogate) model-based algorithms [11,12]. Within the latter family of heuristics, Multi-Objective Bayesian Optimization (MOBO) [13,14] derived from classical Bayesian approaches play a predominant role through the use of infill criteria that take into account Pareto front identification-related indicators such as the Expected Hyper-Volume Improvement (EHVI) [15].

In order to extend the application scope of MOO algorithms with respect to a wider range of problem classes, hybrid MOO algorithms [16] have been developed. They combine elementary search algorithms and their individual strengths with the goal of overcoming each other's weaknesses. One example of such a hybrid algorithm is the Surrogate Optimization of Computationally Expensive Multi-objective Problems (SOCEMO) [17], which integrates various surrogate model-based optimization techniques to effectively address computationally demanding (or, synonymously, expensive) multi-objective optimization

problems. So far, however, the selection of elementary algorithms and their sequence of application is defined manually rather than following an automated optimization scheme.

Within the paper at hand, we present an algorithm-based process to automatically generate a Reinforcement Learning-Based Hybrid Multi-Objective Optimization (RLhybMOO) intelligent agent inherently meeting an individually definable optimization scheme. The set of underlying elementary algorithms can be selected manually, while the sequence of individual applications results from the reinforcement learning (RL) training process.

The paper is organized as follows: Section 2 introduces the methodological framework by detailing the RLhybMOO intelligent agent training algorithm and the application processes. Section 3 describes an exemplary implementation of the RLhybMOO approach in step-by-step mode, being trained on a mathematical test function, and thereby substantiating a variety of choices under the general RLhybMOO approach. Section 4 presents the results of the trained hybrid MOO intelligent agents, comparing them with those of the individual underlying elementary algorithms. Sections 5 and the Conclusion conclude with a discussion and outlook for future work.

2. RLhybMOO Process Structure and Algorithm

In this section, we present the methodological approach using reinforcement learning (RL) techniques for automatically generating an intelligent hybrid MOO agent that is capable of the state-sensitive sequencing of elementary MOO Pareto front search strategies.

Reinforcement learning methods [18] focus on learning effective decisions within a given set of alternative actions through iterative feedback on the effect of actions taken. Within the RL framework vocabulary, an *agent* selects *actions* that influence the *environment*, resulting in a change in the environment's *state*. The agent in turn receives a *reward* signal from an observer, quantifying the impact values of its actions. By assessing the estimated state of the system and the received rewards, the agent learns to adapt its behavior to optimize a cumulative reward metric. The learning process in RL involves the agent iteratively updating its *control policy*, which represents a mapping of the current state of the environment to the action to be selected. During the training phase, the agent refines its policy by exploring the space of possible actions and exploiting the knowledge gained from past experiences, in accordance with the mathematical principles of its *RL algorithm*. This process allows the agent to improve its decision making and to learn an optimal control policy that maximizes the cumulative reward over time.

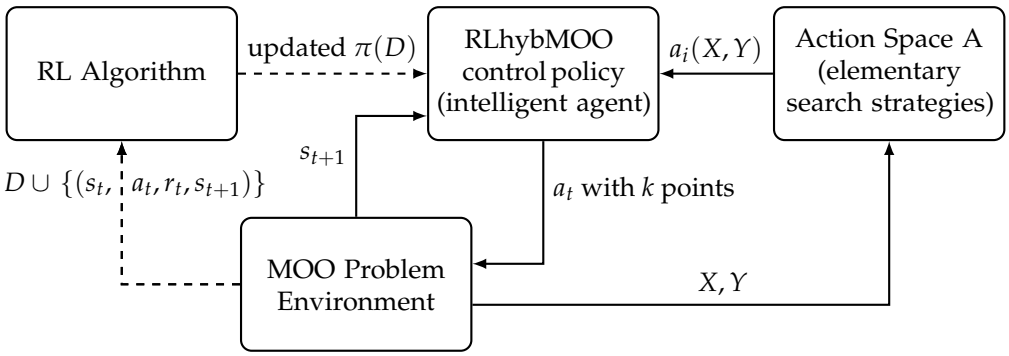


Figure 1. RLhybMOO process structure for training (solid plus dashed lines) and application (solid lines only).

In areas such as cost-optimal planning [19], heuristic estimator selection for satisfying planning [20,21], or dynamic algorithm configuration (DAC) [22,23], the reinforcement learning approach is successfully applied to dynamically respond to problem-specific state changes, finding state-dependent optimal action responses.

Figure 1 shows the structural elements of the process when transferring the RL approach to the task of developing an automated design of a hybrid MOO algorithm. The resulting RLhybMOO policy training algorithm is referenced step-by-step in pseudo-code

form in Algorithm 1. It comprises all of the above mentioned individually required RL elements, as indicated within the structural view of Figure 1:

- The *Action Space* A provides the set of available *actions* to the RLhybMOO control policy as its base of selectable action alternatives. Elements within the Action Space A are predefined by the user. Each element within the *Action Space* represents an elementary MOO sampling strategy $a_i(X, Y) \in A$ that is applicable to the existing problem. Each individual sampling algorithm (also called the *infill* algorithm) yields a dedicated number of sampling points $x_i \in X$ within the design space X (domain) proposed for the next round of (computationally expensive) black-box evaluations to yield a result vector $y_i \in Y$ within the target space Y (co-domain). Usually, the logic of a sampling strategy is based on previous function evaluations $\{(x_i, y_i)\}$ provided by the MOO Problem Environment.

Algorithm 1 RLhybMOO policy training

$A \leftarrow \{a_1, \ldots, a_i\}$ ▷ Define Action set A
$s \leftarrow S(\{(x_i, f(x_i))\})$ ▷ Define State function
$r \leftarrow R(s, a_i)$, $a_i \in A$ ▷ Define Reward function
$\pi \leftarrow \pi_{base}$ ▷ Assign Policy Algorithm base
while $n_{epi} \leq N_{epi}$ **do** ▷ Repeat for N_{epi} episodes
  $X_0 \leftarrow \{x_1, \ldots, x_{N_{init}}\}$ ▷ Set initial DoE
  $Y_0 \leftarrow f(X_0)$ ▷ Simulate at X_0 (expensive)
  $s_0 \leftarrow s_t$ ▷ Initial Environment State s_0
  while $n_{sim} \leq N_{tot}$ **do** ▷ Until abortion criterion
    $a_t \leftarrow \pi(a_i|s_t)$ ▷ Select Policy-based action
    $X_t \leftarrow \{x_1, ..., x_k\}$ ▷ a_t-based sample definition
    $Y_t \leftarrow f(X_t)$ ▷ Simulate at X_t (expensive)
    $X \leftarrow X \cup X_t$ ▷ Extend sample set in domain
    $Y \leftarrow Y \cup Y_t$ ▷ ... and in co-Domain
    $s_{t+1} \leftarrow S(X, Y)$ ▷ Calculate new state
    $r_t \leftarrow R(s_{t+1}, a_t)$ ▷ Calculate reward
    $t \leftarrow t + 1$ ▷ Increase time step
    $n_{sim} \leftarrow n_{sim} + k$ ▷ Update sampling counter
    $D_t \leftarrow \{(a_t, r_t, s_t, s_{t+1})\}$ ▷ Define experience
    $D \leftarrow D \cup D_t$ ▷ Update experience buffer
  end while
  $n_{epi} \leftarrow n_{epi} + 1$ ▷ Update episode counter
  $\pi \leftarrow \pi(D)$ ▷ Update Policy by experience buffer
end while
$\pi_{MOO} \leftarrow \pi$ ▷ Define RLhybMOO Policy

- *MOO Problem Environment*: Black box function evaluation tasks $y_i = f(x_i)$ are answered by the (simulation-capable) model of the MOO Problem Environment. In addition, it provides descriptive *state* features at each time step t, as required for learning the hybrid MOO algorithm control policy. The variables employed to characterize an environment state, referred to as state features, must be observable in a quantifying way, and representative for describing the overall achieved quality of solving the MOO problem. To evaluate the value-adding contribution of the k function evaluations as triggered by an individual action a_i applied at a state s at time t, a suitable *reward* function $r_t = R(s, a_i)$ is provided. Overall, the MOO Problem Environment provides the k function evaluations (X_t, Y_t) requested by action a_t at state s_t, analyzes their value-add by the reward r_t, and calculates the consecutive state s_{t+1} of the progressing optimization process.
  Starting the training or evaluation process requires the definition of several optimization and process-related parameters, specifically the total number of available samples N_{tot}, the number of initial samples N_{init}, and the number of episodes N_{epi} to train

the policy. In the first step (episode) of each optimization process, an initial Latin Hyper Cube Sampling (LHCS) with N_{init} intial training points is executed to receive a first information set (X_0, Y_0) and to calculate the initial state s_0. Subsequent states s_{t+1} of the MOO Problem Environment are handed over to the RLhybMOO policy. In each increasing time step t, the number of evaluated samples n_{sim} is monitored. An individual training phase, and thereby, an episode, ends at the particular time-step when the number of evaluated sample points n_{sim} equals or exceeds the number of available sample points N_{tot}.

- *Control Policy*: Finally, the desired RLhybMOO control policy $\pi(a_i|s_t)$, resulting from the function block of the same name represents a probability density for any action a_i within the action space A as a function of the internal state s_t, including a decision scheme for selecting one of the available actions. The thereby chosen sampling strategy triggers a search at k design points.
- *RL Algorithm*: To start the search for the RLhybMOO control policy, a problem-adequate *RL algorithm* family must be defined. It phrases the mathematical nature of the policy and its base initialization π_{base}. During the training phase (indicated by dashed lines in Figure 1), the selected RL algorithm is fed by individual learning "experiences" $\{(s_t, a_t, r_t, s_{t+1})\}$, as gathered during an iterative episodic training process. The set of these experiences forms the experience memory D as a learning base for the control policy π, repeatedly updated after the completion of every episodic training iteration n_{epi} until the final number of allowed training episodes N_{epi} is reached.

3. Experimental Setup

In this section, a prototypic RLhybMOO design is presented on exemplary basis to further explain the elements of the proposed approach and to demonstrate some of its elementary advantages. The experimental setup for this approach aims to quantify a converging learning behavior via the proposed training process, and allows for a comparison between the thereby gained optimized hybrid MOO algorithmic approach versus its elementary constituents. The quantitative performance evaluation is enabled by the choice of computational cheapness, and to be minimized, the multi-objective Zitzler-Deb-Thiele (ZDT) problem suite functions [24] following the construction:

$$\begin{aligned} &min\, f_1(x) \\ &min\, f_2(x) = g(x)h(f_1(x), g(x)) \end{aligned} \tag{5}$$

The function $g(x)$ is considered as the function of convergence and is for the ZDT functions under consideration, defined as:

$$g(x) = 1 + \frac{9}{n-1}\sum_{i=2}^{d} x_i \tag{6}$$

The ZDT problem suite is of particular value for a number of reasons. Among other aspects, it allows for the formulation of problems with parametrically definable design space dimension d, and provides analytical accessibility to compute the Pareto-optimal points. The RLhybMOO training approach will be conducted on one of these functions (ZDT1), and then additionally bench-marked against established algorithms on other test functions (ZDT2 and ZDT3) of the same ZDT problem suite to analyze its transferability and effectiveness. All essential elements and parameters, as introduced during Section 2, are designed and chosen for emphasizing the demonstrative character of the prototypical application.

Sections 3.1 to 3.4 define the essential elements of the process, while Section 3.5 quantifies the algorithm hyperparameters required for the executable form of the experiments.

3.1. State Description

Learning hybrid MOO algorithm control policies requires descriptive state features containing information about characteristics and dynamics during an ongoing optimization. A variety of observables may be chosen to describe the status s_t of the optimization process at time t. For our example, we select a mix of qualitative and temporal attributes in terms of a five-dimensional state vector with components characterizing or characterized by:

- A solution quality increase with respect to initial sampling,
- A solution quality increase by the last taken action,
- A stagnation counter,
- The number of identified Pareto-optimal solutions,
- The percentage of left over vs. totally available samples.

Quantifying the solution qualities of the MOO algorithm results, we use the hypervolume performance indicator [25]. The hypervolume of a problem solution set measures the size of the sum of all cubes within the target space spanned by all non-dominated (i.e., Pareto-optimal) solutions and some reference point r. During the process, this reference point for the calculation of the hypervolume is set by the worst (e.g., maximum) values in each objective axis from the initial sampling. Maximizing the hypervolume by adding an additional solution point is identical to having identified another Pareto-optimal solution [26], i.e., the hypervolume is a suitable measure of the effectiveness of a sampling strategy. The solution quality at time step t may therefore be characterized by the hypervolume $HV_r(Y_t)$, calculating the hypervolume for the solution set Y_t as existing at time step t, and defined in [26] with respect to the reference point r, as defined in [26].

The first entry in the state vector is expressed by the hypervolume improvement in the current state s_t (at time step t) with respect to its initial value in the initial state s_0 (at time step $t = 0$ after the initial LHCS) relative to the total hypervolume

$$\frac{HV_r(Y(s_t)) - HV_r(Y(s_0))}{HV_r(Y(s_t))}. \tag{7}$$

The second entry in the state vector indicates the quality change achieved by the last action compared to the previous state. We therefore define the hypervolume improvement

$$hvi := \frac{HV_r(Y(s_t)) - HV_r(Y(s_{t-1}))}{HV_r(Y(s_t))} \tag{8}$$

as an appropriate performance indicator.

The stagnation counter, i.e., the third state vector entry, represents the number of time steps that have passed without any solution quality improvement. The metric is intended to encourage exploratory actions if the optimization process stagnates.

As the fourth entry in the state vector, we added the cardinality of Pareto points

$$P = |Y(s_t)| \tag{9}$$

(see Section 1, Equation (4)) of the set $Y(s_t)$ of (black box) function evaluations to enable reactions within the search strategy based on this search efficiency-related performance indicator. Note that the cardinality of $Y(s_t)$ is monotonically growing until it reaches the total number of allowed evaluations N_{tot}.

The fifth state vector entry is defined by the proportion of leftover samples to the total number of available samples n_{sim}/N_{tot} yielding information about the temporal component of the optimization process. This information is especially important for applications in expensive black-box functional optimization, where the number of feasible simulations is limited, and therefore, efficiency should be increased.

3.2. Reward Function

The reward function provides the necessary feedback on the success of chosen actions, based on the available state information by the control strategy. In our example case, the

reward function aims to maximize our solution quality (measured in terms of improvement in hypervolume) with a minimum of computationally intensive function calls. Thus, while the hypervolume merely reflects the effectiveness of the Pareto search, the reward function aims to maximize the efficiency of the targeted hybrid algorithm. This focus may result in a control policy picking at a given state the right action that maximizes the hypervolume improvement with a minimum number of computationally expensive samples. Thus, closely following [27], we propose a reward function that measures the hypervolume improvement between two successive states induced by the previous action a_t in relation to the number k of its returned samples

$$R(s_t, a_t) = C * \frac{hvi}{k} \tag{10}$$

with a scaling constant C. Consequently, the actions that did not improve the hypervolumes of successive states are assigned a reward of zero. The ratio behind this aims to reinforce the search strategy to pick those actions that maximize our objective. Note that not all individual strategies return the same number of new simulation points and thus, a fair reward should be achieved in terms of the algorithms efficiency.

3.3. Action Space

To ensure a successful RL-hybMOO policy search, the predefined selection of the set of available actions (Action Space A) is crucial. For reasons of comparability, we choose in our work one of the few existing and benchmarked hybrid MOO algorithms, SOCEMO [17], using five elementary sampling strategies in a strict and well-defined sequence based on the evaluation of adaptive surrogate models. It is a stated SOCEMO goal to enable efficient sampling with a minimum amount of computationally expensive function evaluations. We therefore picked all five elementary sampling strategies as selectable actions $a_i \in \{1, \ldots, 5\}$ (but not the SOCEMO predefined sequence of their application), where the values are mapped to the adapted elementary sampling strategies described in detail by Müller [17]:

- a_1: Target Value Strategy (ParetoFill),
- a_2: Pareto Points Random Perturbation,
- a_3: Minimum Point Sampling (Single Optimization),
- a_4: Uniform Random Points over the Design Space,
- a_5: Surrogate MOO Genetic Algorithm (NSGA-II).

3.4. Policy Adaptation (RL Algorithm)

To initialize π_{base} and to iteratively adjust a dynamic policy for the sampling strategies, the model-free Soft Actor-Critic (SAC) [28] algorithm is used. This is an off-policy actor-critic Deep Reinforcement Learning algorithm based on the maximum entropy framework [28]. Maximum Entropy RL algorithms attempt to find a policy that compromises between maximizing the reward (exploitation) and the maximum entropy objective (exploration) to successfully complete their task while acting as randomly as possible. The stochastic actor-critic formulation tends to avoid assigning an excessively high probability to any specific action within the range of available actions [28]. This behavior reduces the risk of becoming stuck in a local optimum and provides incentives to explore the action space further, which is a desirable property for our approach as a state-specific representation of decisions about actions affected by stochastic influences. Thus, multiple promising paths are given equal probability mass by the policy. This is beneficial for our approach, since multiple actions can appear attractive in a given state, resulting in multiple near-optimal paths for solving MOO problems. The follow up implementation [29] is used to make SAC compatible with discrete action spaces to fit our MOO Problem Environment, and provides state-of-the-art sample efficiency and robustness in training. This might help for a generalization of this approach when applying to expensive black box functions.

3.5. Parameter Settings

The policy is trained for $N_{epi} = 35,000$ episode iterations. The total number of samples available for the training process is set to $N_{tot} = 100$ with an initial sample population fraction of 40%, i.e., $(N_{init} = 40)$ samples are available for the initial LHCS. To ensure the comparability of results and baseline conditions for each episode and its related learning progress, the training process is conducted with a fixed seed for the initial LHCS.

The test functions are then evaluated over $N_{total} = 50$ with an initial sampling fraction of 60%, i.e., $N_{init} = 30$, and random seeds.

4. Results

All training and test experiments were run on an RTX A5000 GPU, and algorithms were run on an AMD Ryzen 7 5800X CPU with 64 GB of dynamic memory.

For the training of the RLhyMOO control policy, the ZDT1 test problem was used, refining the generic test suite Equations (5) and (6) by

$$\begin{aligned} f_1(x) &= x_1 \\ h(f_1, g) &= 1 - \sqrt{\frac{f_1}{g}} \end{aligned} \tag{11}$$

with $d = 30$ input dimensions. Its analytically well-known optimal set of solutions forms a convex-shaped Pareto front.

Figure 2 shows the mean reward over the total episode during the training process, indicating the span between the maximally and minimally achieved rewards at each episode. The spread decreases over time, with a mean reward remaining almost constant after more than approximately 20,000 training episodes.

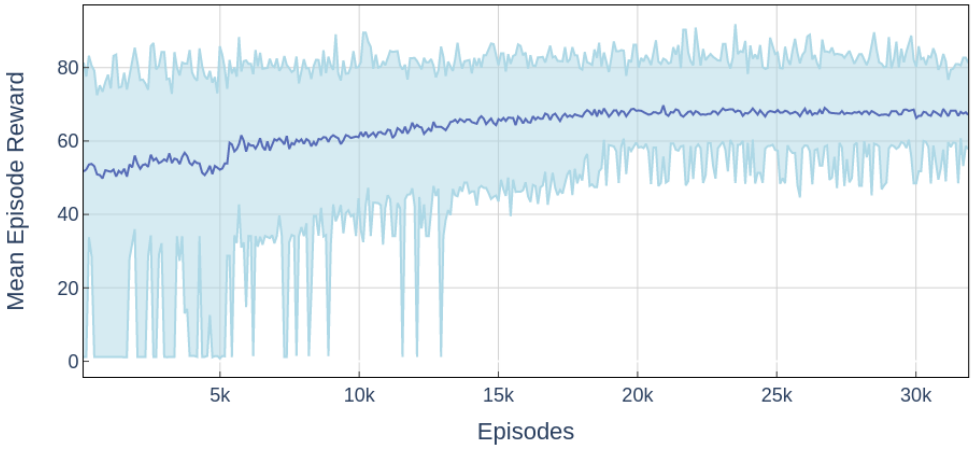


Figure 2. Mean episode reward evolution during training process.

Figure 3 shows how the frequency of samplers used per episode changes during the evolving training process. Sample strategy a_5 is developed to the most frequently used action, and strategies a_1 and a_3 are gaining in importance, while the usage of a_4 slightly decreases, and a_2 are only used sporadically as the number of episodes increases.

To further evaluate the performance of the finally trained policy, it is subsequently applied to ZDT2 with

$$\begin{aligned} f_1(x) &= x_1 \\ h(f_1, g) &= 1 - (\frac{f_1}{g})^2 \end{aligned} \tag{12}$$

and ZDT3 with

$$f_1(x) = x_1$$
$$h(f_1, g) = 1 - \sqrt{\frac{f_1}{g}} - \frac{f_1}{g} * \sin 10 f_1 \pi. \tag{13}$$

These problems are considered as novel from a training perspective, because unlike ZDT1, they have different Pareto front characteristics with a concave shape in ZDT2 and a series of disconnected Pareto fronts in ZDT3. For the benchmark and statistical purposes of evaluation, the number of ZDT1, ZDT2, and ZDT3 input dimensions was varied within the range of $d \in \{2, 3, 5, 10, 15, 20, 25, 30\}$. The trained RLhybMOO policy is benchmarked for an effectiveness comparison against the established algorithms NSGA-II [30], SOCEMO, and the random search LHCS [31], as well as against the individual, model-based elementary sampling strategies used in the Action set A.

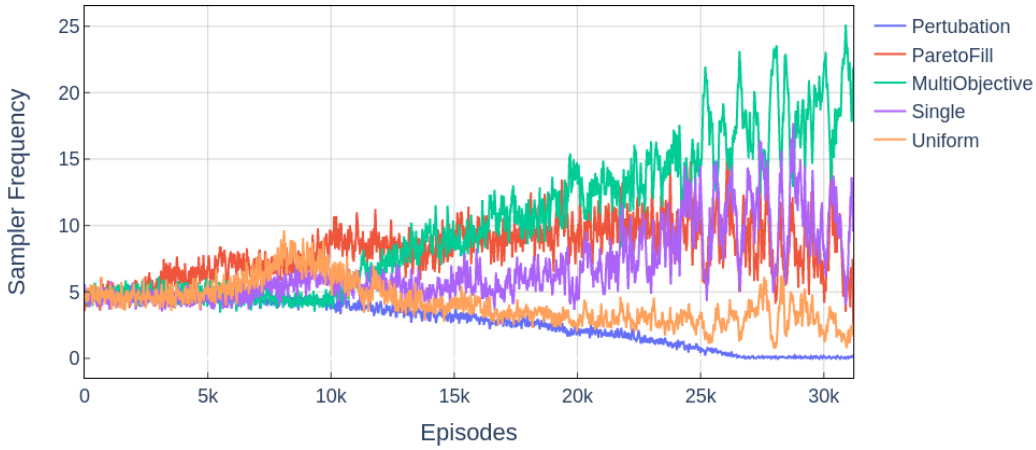
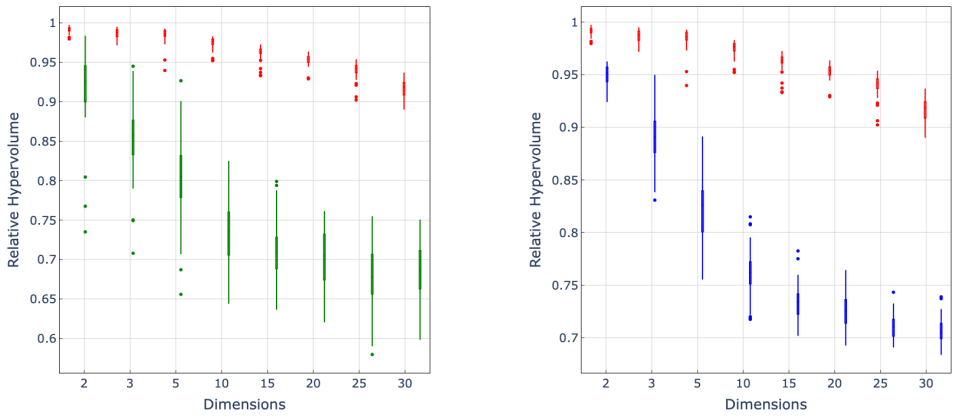


Figure 3. Development of episodic sampler distribution, indicating frequency of elementary algorithm calls within a given episode.

For the application of NSGA-II, the evolutionary parameters *generations* and *population size* are set to $n_{gen} = 5$ and $n_{pop} = 10$. In all cases, the hypervolume is used as a MOO algorithm effectiveness indicator for the ability of the control policy to find Pareto-optimal solutions. Since the analytical form of Pareto fronts and thus, also the maximum achievable hypervolume is known for all test problems, the achieved hypervolume of all algorithm optimization results is determined with the same reference point $r = (2, 10)$ for comparability, according to [26].

Figure 4 indicates the identified relative (with respect to its theoretical maximum) hypervolume of the repeated search results for all described policy tests (in contrast to training) evaluations of the ZDT1 problem. The box plots in Figure 4a,b compare the RLhybMOO relative hypervolume achievement results versus NSGA-II and LHC, respectively. Figure 5 compares the RLhybMOO median performance to that of its underlying individual search strategies as a function of the test problem dimension.

The results for all three test problems are summarized in Table 1, where the listed *Average Performance Median* refers to the average of the median hypervolume over the set of dimensions.



(**a**) (**b**)

Figure 4. Performance comparison of (**a**) RL-hybMOO (red) with NSGA-II (green), and (**b**) RLhybMOO (red) with LHCS (blue), indicated by the relative of the max hypervolume found for ZDT1 over the range of input dimensions.

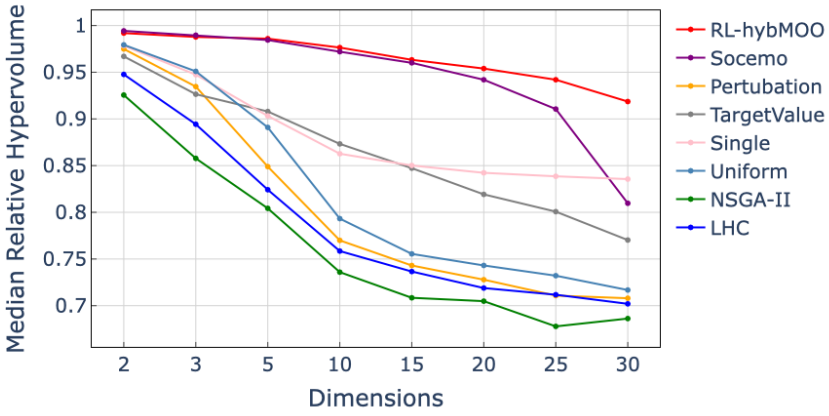


Figure 5. Performance comparison of RLhybMOO median compared to that of its elementary search strategies. Relative of max hypervolume indicated by the relative of the max hypervolume found for ZDT1 over the range of input dimensions.

Table 1. Average Performance Median Comparison.

Test Function	SOCEMO	RLhyb MOO	Target Value	Pertur-bation	Single	Uniform	NSGA II	LHC
ZDT1	94.53%	96.49%	86.40%	80.2%	88.2%	82.0%	86.89%	82.16%
ZDT2	96.23%	96.30%	87.5%	71.92%	85.86%	74.5%	79.18%	73.03%
ZDT3	85.67%	89.23%	87.44%	72.5%	85.84%	75.11%	75.89%	76.26%

5. Discussion

Figure 3 illustrates the progressive learning process over the episodes, towards an increasing efficiency of the control policy. This is obviously achieved by the alternating selection strategy of the elementary search algorithms during the learning process, as shown in Figure 3. The increasing policy search performance is quantified in Figure 2, demonstrating an achieved average reward growth of 30% during the training process,

compared to the starting random decision making policy. This indicates the policies to select elementary search strategies that are more likely to achieve an improvement based on its analyzed problem solving status. Nevertheless, there is still a statistical spread of *min* and *max* total rewards for the episodes, explainable by the factor of randomness in generating evaluation points that each sampling strategy intentionally includes. The two aforementioned observations on the training process indicate that the approach has trained a policy that combines the elementary algorithms based on status analysis in such a way that they contribute as much as possible to the solution quality.

All tested search algorithms in Figure 5 show a decrease in their effectiveness over the number of problem input dimensions. However, our hybrid RL approach outperforms all other (static hybrid and elementary) search strategies over the problem range, up to 30 input dimensions. Even with 30 problem input dimensions, RL-hybMOO still achieves over 90% of the theoretically coverable hypervolume.

The RL-based strategy dominates each of the individual algorithms, indicating the importance of using hybrid algorithms for good performance with a limited number of available samples. Moreover, without the use of a large number of iterations or large population sizes, evolutionary algorithms can hardly solve the optimization problems effectively. In addition, there is a slight improvement over the static alternating (hybrid) sampling approach of SOCEMO in all three test functions, especially with an increasing number of dimensions. However, this has been demonstrated only for one possible instance of the RLhybMOO implementation and the test problem suite. The approach cannot guarantee a generalized improvement for all applications or differently shaped problems.

6. Conclusions and Outlook

This paper introduces a fundamentally novel approach for designing hybrid multi-objective optimization algorithms to solve black-box problems by combining elementary search strategies based on reinforcement learning techniques. We describe the general method and its base algorithm to identify a Reinforcement Learning-Based Hybrid Multi-Objective Optimization (RLhybMOO) policy. The workflow and its performance gain over conventional approaches is demonstrated using a prototype implementation that is trained and evaluated with mathematical test functions. The training process demonstrates the ability for progressive convergent policy learning, and a significant performance gain over the underlying elementary search policies. Not all elementary search strategies contribute equally to the solution process. In this context, the RL-trained policy has demonstrated its ability to prioritize appropriate effective and efficient actions for the Pareto search. A benchmark on mathematical test functions could prove that the RLhybMOO approach is able to outperform elementary and static hybrid search strategies. Some topics within this research, however, remain open for future work, such as applying the approach to a variety of data sets for validation purposes, and expanding the action space through MOBO algorithms. It was found that the description of the state of the solution process, and the associated reward have a significant impact on the adapted policy, and therefore, they deserve further investigation. In addition, the definition of the action space should be extended to include other elementary search algorithms that may affect decision performance. In addition, other RL algorithms and implementation forms should be considered and investigated with respect to real-world, computationally expensive design optimization problems with multiple competing targets.

Author Contributions: Basic idea, L.A.; Article conceptualization, H.P.; methodology, H.P.; software, L.A.; validation, H.P. and L.A.; writing—original draft preparation, H.P. and L.A.; writing—review and editing, L.A. and H.P.; visualization, L.A.; supervision, H.P.; project administration, H.P. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: The data presented in this study are available on request from the corresponding author.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Snowden, D.J.; Boone, M.E. A leader's framework for decision making. *Harv. Bus. Rev.* **2007**, *85*, 68. [PubMed]
2. Renn, O.; Klinke, A.; Van Asselt, M. Coping with complexity, uncertainty and ambiguity in risk governance: A synthesis. *Ambio* **2011**, *40*, 231–246. [CrossRef] [PubMed]
3. Hwang, C.L.; Masud, A.S.M. *Multiple Objective Decision Making—Methods and Applications: A State-of-the-Art Survey*; Springer Science & Business Media: Berlin/Heidelberg, Germany, 2012; Volume 164.
4. Roijers, D.M.; Whiteson, S. Multi-objective decision making. *Synth. Lect. Artif. Intell. Mach. Learn.* **2017**, *11*, 1–129.
5. Aruldoss, M.; Lakshmi, T.M.; Venkatesan, V.P. A survey on multi criteria decision making methods and its applications. *Am. J. Inf. Syst.* **2013**, *1*, 31–43.
6. Deb, K. Multi-objective optimization. In *Search Methodologies*; Springer: Berlin/Heidelberg, Germany, 2014; pp. 403–449.
7. Deb, K., Multi-Objective Evolutionary Algorithms. In *Springer Handbook of Computational Intelligence*; Kacprzyk, J., Pedrycz, W., Eds.; Springer: Berlin/Heidelberg, Germany, 2015; pp. 995–1015. [CrossRef]
8. Sheng, L.; Ibrahim, Z.; Buyamin, S.; Ahmad, A.; Mohd Tumari, M.Z.; Mat Jusof, M.F.; Aziz, N. Multi-Objective Particle Swarm Optimization Algorithms—A Leader Selection Overview. *Int. J. Simul. Syst. Sci. Technol.* **2014**, *15*, 6–19. [CrossRef]
9. Alaya, I.; Solnon, C.; Ghedira, K. Ant Colony Optimization for Multi-Objective Optimization Problems. In Proceedings of the 19th IEEE International Conference on Tools with Artificial Intelligence (ICTAI 2007), Patras, Greece, 29–31 October 2007; Volume 1, pp. 450–457. [CrossRef]
10. Murata, T.; Ishibuchi, H. MOGA: Multi-objective genetic algorithms. In Proceedings of the IEEE International Conference on Evolutionary Computation, Perth, WA, Australia, 29 November–1 December 1995; Volume 1, pp. 289–294.
11. Hale, J.Q.; Zhou, E. A model-based approach to multi-objective optimization. In Proceedings of the 2015 Winter Simulation Conference (WSC), Huntington Beach, CA, USA, 6–9 December 2015; pp. 3599–3609. [CrossRef]
12. Jiang, P.; Zhou, Q.; Shao, X. *Surrogate Model-Based Engineering Design and Optimization*; Springer: Berlin/Heidelberg, Germany, 2020.
13. Betrò, B.; Schoen, F. A stochastic technique for global optimization. *Comput. Math. Appl.* **1991**, *21*, 127–133. [CrossRef]
14. Emmerich, M.; Yang, K.; Deutz, A.; Wang, H.; Fonseca, C. Multicriteria Generalization of Bayesian Global Optimization. In *Advances in Stochastic and Deterministic Global Optimization*; Pardalos, P.M., Zhigljavsky, A., Žilinskas, J., Eds.; Springer International Publishing: Berlin/Heidelberg, Germany, 2016; Chapter A.
15. Emmerich, M.; Klinkenberg, J.W. The computation of the expected improvement in dominated hypervolume of Pareto front approximations. *Rapp. Tech. Leiden Univ.* **2008**, *34*, 3–7.
16. Sindhya, K.; Miettinen, K.; Deb, K. A hybrid framework for evolutionary multi-objective optimization. *IEEE Trans. Evol. Comput.* **2012**, *17*, 495–511. [CrossRef]
17. Müller, J. SOCEMO: Surrogate Optimization of Computationally Expensive Multiobjective Problems. *INFORMS J. Comput.* **2017**, *29*, 581–596. [CrossRef]
18. Gosavi, A. Reinforcement learning: A tutorial survey and recent advances. *INFORMS J. Comput.* **2009**, *21*, 178–192. [CrossRef]
19. Sievers, S.; Katz, M.; Sohrabi, S.; Samulowitz, H.; Ferber, P. Deep Learning for Cost-Optimal Planning: Task-Dependent Planner Selection. *Proc. Aaai Conf. Artif. Intell.* **2019**, *33*, 7715–7723. [CrossRef]
20. Röger, G.; Helmert, M. The More, the Merrier: Combining Heuristic Estimators for Satisficing Planning. *Proc. Int. Conf. Autom. Plan. Sched.* **2010**, *20*, 246–249. [CrossRef]
21. Speck, D.; Biedenkapp, A.; Hutter, F.; Mattmüller, R.; Lindauer, M. Learning Heuristic Selection with Dynamic Algorithm Configuration. In Proceedings of the International Conference on Automated Planning and Scheduling, Nancy, France, 14–19 June 2020. [CrossRef]
22. Biedenkapp, A.; Bozkurt, H.F.; Eimer, T.; Hutter, F.; Lindauer, M.T. Dynamic Algorithm Configuration: Foundation of a New Meta-Algorithmic Framework. In Proceedings of the ECAI, Santiago de Compostela, Spain, 29 August–8 September 2020; IOS Press: Amsterdam, The Netherlands, 2020.
23. Adriaensen, S.; Biedenkapp, A.; Shala, G.; Awad, N.H.; Eimer, T.; Lindauer, M.T.; Hutter, F. Automated Dynamic Algorithm Configuration. *arXiv* **2022**, arXiv:2205.13881.
24. Zitzler, E.; Deb, K.; Thiele, L. Comparison of Multiobjective Evolutionary Algorithms: Empirical Results. *Evol. Comput.* **2000**, *8*, 173–195. [CrossRef] [PubMed]
25. Bader, J.; Zitzler, E. A Hypervolume-Based Optimizer for High-Dimensional Objective Spaces. In *New Developments in Multiple Objective and Goal Programming*; Springer: Berlin/Heidelberg, Germany, 2010; Volume 638, pp. 35–54. [CrossRef]
26. Palm, N.; Landerer, M.; Palm, H. Gaussian Process Regression Based Multi-Objective Bayesian Optimization for Power System Design. *Sustainability* **2022**, *14*, 12777. [CrossRef]
27. Karafotias, G.; Eiben, A.E.; Hoogendoorn, M. Generic Parameter Control with Reinforcement Learning. In Proceedings of the 2014 Annual Conference on Genetic and Evolutionary Computation, Lille, France, 10–14 July 2021; Association for Computing Machinery: New York, NY, USA, 2014; pp. 1319–1326. [CrossRef]
28. Haarnoja, T.; Zhou, A.; Abbeel, P.; Levine, S. Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor. In Proceedings of the International Conference on Machine Learning, Jinan, China, 19–21 May 2018.
29. Christodoulou, P. Soft Actor-Critic for Discrete Action Settings. arXiv **2019**, arXiv:1910.07207.

30. Deb, K.; Pratap, A.; Agarwal, S.; Meyarivan, T. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Trans. Evol. Comput.* **2002**, *6*, 182–197. [CrossRef]
31. Mckay, M.; Beckman, R.; Conover, W. A Comparison of Three Methods for Selecting Vales of Input Variables in the Analysis of Output From a Computer Code. *Technometrics* **1979**, *21*, 239–245. [CrossRef]