AI MDPI

Article

Hypervolume-Based Multi-Objective Optimization Method Applying Deep Reinforcement Learning to the Optimization of Turbine Blade Shape

Kazuo Yonekura *, Ryusei Yamada, Shun Ogawa and Katsuyuki Suzuki

Department of Systems Innovation, The University of Tokyo, Tokyo 113-8654, Japan
* Correspondence: yonekura@struct.t.u-tokyo.ac.jp

Abstract: A multi-objective turbine shape optimization method based on deep reinforcement learning (DRL) is proposed. DRL-based optimization methods are useful for repeating optimization tasks that arise in applications such as the design of turbines and automotive parts. In conventional research, DRL is applied only to single-optimization tasks. In this study, a multi-objective optimization method using improvements in hypervolume is proposed. The proposed method is applied to a benchmark problem and a turbine optimization problem. It succeeded in efficiently solving the problems, and Pareto optimal solutions are obtained.

Keywords: deep reinforcement learning; repeating optimization task; multi-objective optimization; Pareto solutions

check for updates

Citation: Yonekura, K.; Yamada, R.; Ogawa, S.; Suzuki, K. Hypervolume-Based Multi-Objective Optimization Method Applying Deep Reinforcement Learning to the Optimization of Turbine Blade Shape. *AI* **2024**, *5*, 1731–1742. https://doi.org/10.3390/ai5040085

Academic Editor: Gianni D'Angelo

Received: 19 August 2024
Revised: 10 September 2024
Accepted: 13 September 2024
Published: 24 September 2024

1. Introduction

Deep reinforcement learning (DRL) utilizes deep neural networks (DNNs) in reinforcement learning algorithms. DRL is utilized in various industrial fields, mainly for control problems such as robotics [1–3], path planning [4,5], and automobile control [6,7]. DRL is also utilized in mechanical design applications [8], including the designs of airfoil shapes [9–11], turbine blades [12], and flow controls [13,14], as well as involving topology optimization [15].

In mechanical optimization, designers must find solutions to multi-objective optimization problems. In multi-objective optimization, it is necessary to obtain Pareto solutions [16]. Genetic algorithms, including the multi-objective GA (MOGA) [17], non-dominated sorting GA (NSGA) [18], and NSGA-II [19], are among the major methods used to perform multi-objective optimization [20]. Moreover, reinforcement learning has recently been utilized in mechanical design applications because of its high generalization capability. A properly trained RL agent has good generalization capability [9,12,21]. Hence, once trained, an RL agent is applicable to slightly different optimization problems. However, multi-objective DRL (MODRL) methods for mechanical designs have not been extensively studied.

Therefore, this study focuses on MODRL for use in multi-objective optimization problems. Several MODRL methods have been proposed for use in the field of computer science [5,22,23]. The weighted-sum method (WSM) is often used [24–26]. The WSM takes the weighted sum of the objective functions and treats the problem as a single-objective optimization problem. However, in using the WSM to obtain the Pareto front, we have to solve multiple weighted-sum problems by changing the weights. The computational cost of this procedure is high, and a non-convex Pareto front cannot be obtained using this procedure. Another commonly used method is the ε-constrained method [27–29], which uses only one primary objective function and uses the others as constraints.

The hypervolume, which is the volume of the hypercubes generated by the Pareto solutions, is often considered to obtain good Pareto solutions. Moreover, it is often used as both an indicator to measure the goodness of the obtained Pareto solutions and an objective function to be maximized [30–33]. Good Pareto solutions can be obtained by maximizing the hypervolumes.

In DRL, an extensive computational cost is incurred in the model training. Once trained, the agent can adapt to various problems, but if the objective function changes, then the agent must be retrained. Therefore, if DRL is coupled with the WSM or ε-constrained method, then the DRL agent must be trained multiple times to change the weights or constraints. In such cases, the computational costs are extremely high. In contrast, if the hypervolume-based method is coupled with DRL, then only one training phase is required, as the objective function and constraints do not change. Therefore, in this study, we propose a novel MODRL method that couples the hypervolume method with DRL.

The remainder of this paper is organized as follows. In Section 2, we formulate the proposed method by using the hypervolume. In Section 3, we describe the application of the proposed method to a benchmark problem. In Section 4, we formulate and solve the turbine optimization problem by using the proposed method. Finally, in Section 5, we conclude the paper.

2. Hypervolume-Based Multi-Objective Deep Reinforcement Learning

2.1. *Deep Reinforcement Learning for Turbine Blades*

RL obtains the optimal strategy for a Markov decision process (MDP). An MDP is a stochastic process that consists of states s_t, actions a_t, and rewards r_t, where t is the time step. In MDP, the state s_t and action a_t are related only to those values of the previous time step $t-1$. They are not related to the values of the other steps.

The proximal policy optimization (PPO) [34] algorithm was used in this study. PPO uses an actor network and a critic network. The actor network performs actions, whereas the critic network estimates the values of the actions and evaluates the policy of the actor network.

Ref. [12] proposed a DRL-based turbine optimization method that uses computational fluid dynamics (CFD) as an environment. In this method, an agent takes actions to modify turbine shapes. In this study, we coupled this DRL-based optimization method [12] with the hypervolume method.

2.2. *Hypervolumes of Pareto Solutions*

In multi-objective optimization, the solutions are divided into non-dominated solutions (i.e., Pareto solutions) and dominated solutions. We assume that the number of objective functions is d and that all functions are minimized. Let $\boldsymbol{z}^i \in \Re^d$ $(i = 1, 2, \ldots, n)$ be a solution, where i is the suffix and n is the number of solutions. A solution $\boldsymbol{z}^* \in \Re^d$ is the dominant solution if

$$z_k^* > \boldsymbol{z}_k^i,\ \forall k (1 \leq k \leq d)$$

holds. Pareto solutions are defined as non-dominated solutions.

Let $\mathcal{P}_t$ be a set of Pareto solutions at time step t and $\mathcal{P}_t = \{\boldsymbol{p}_1, \boldsymbol{p}_2, \ldots, \boldsymbol{p}_n\}$, where n is the number of Pareto solutions. In the two-dimensional case, $\boldsymbol{p}_i = (x_i, y_i)$. A hypercube $C(\boldsymbol{p}_i)$ of solution $\boldsymbol{p}_i$ is defined as a hypercube with $\boldsymbol{p}_i$ and reference point $\boldsymbol{p}_0$ as its diagonal points. In the two-dimensional case, $C(\boldsymbol{p}_i)$ is defined as

$$C(\boldsymbol{p}_i) = \{(x, y) \mid x_i \leq x \leq x_0,\ y_i \leq y \leq y_0\}.$$

The hypervolume V of the Pareto solutions $\mathcal{P}$ is defined as the hypervolume of the union of sets $C(\boldsymbol{p}_i)$:

$$V(\mathcal{P}) = \| \cup_i C(\boldsymbol{p}_i) \|,$$

where $\| \cdot \|$ denotes the hypervolume.

2.3. Reward Function

The reward function is defined according to three patterns: having the CFD computation fail, obtaining dominated solutions, and successfully finding a Pareto solution. The agent receives penalties in the first two patterns and rewards in the final pattern. An overview of rewards and penalties is provided in Figure 1, where C_1, C_2, C_3, and C_4 are the constants and we let $C_1 = 100$, $C_2 = 10$, $C_3 = 20$, and $C_4 = 50$.

Pattern 1: Failed CFD

If the CFD computation does not converge, a fixed penalty is assigned as a reward.

$$r_t = -C_1. \tag{1}$$

Pattern 2: Dominated solutions

If the CFD computation converges but is a dominated solution, then the agent obtains a penalty, according to the distance between the solution and the Pareto front.

$$r_t = \begin{cases} \log(C_2 - d) - C_3, & \text{if } d < C_2 \\ C_4 & \text{if } d \geq C_2, \end{cases} \tag{2}$$

where d is the distance between the solution and the Pareto front.

Pattern 3: Pareto solutions

If the obtained solution is a Pareto solution, the agent obtains a reward, according to the improvement in the hypervolume. The improvement is defined as follows:

$$r_t = {C_1}^2(V(\mathcal{P}_t) - V(\mathcal{P}_{t-1}))^2. \tag{3}$$

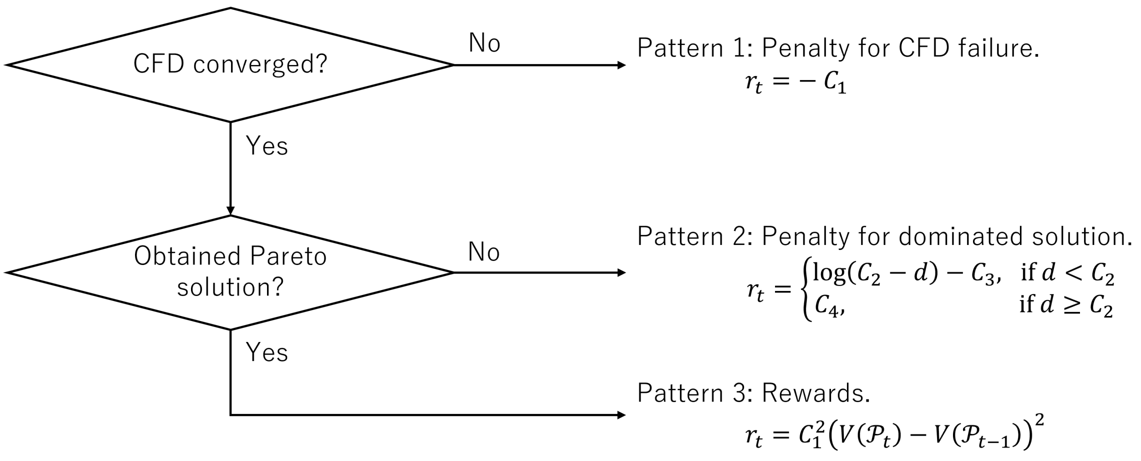

CFD converged?
No
Yes
Pattern 1: Penalty for CFD failure.
$r_t = -C_1$
Obtained Pareto solution?
No
Yes
Pattern 2: Penalty for dominated solution.
$r_t = \begin{cases} \log(C_2 - d) - C_3, & \text{if } d < C_2 \\ C_4, & \text{if } d \geq C_2 \end{cases}$
Pattern 3: Rewards.
$r_t = C_1^2\big(V(\mathcal{P}_t) - V(\mathcal{P}_{t-1})\big)^2$

Figure 1. Rewards and penalties.

2.4. Optimization Algorithm

The optimization algorithm consists of two phases: training and optimization. The agent is trained in the training phase and used in the optimization phase. An overview of the training phase is provided in Figure 2. Each case corresponds to a single-flow condition and consists of 10 episodes, each of which corresponds to one initial solution. A CFD analysis was conducted on the initial shape. Subsequently, the state is input to the agent, and the agent modifies its shape. However, CFD computations sometimes fail because of unsuitable shape modifications. For example, a shape sometimes contains a self-intersection. The termination condition of each episode is the number of shape modifications reaching 50 or the CFD computation failing to converge. The trained agent was used in the optimization phase.

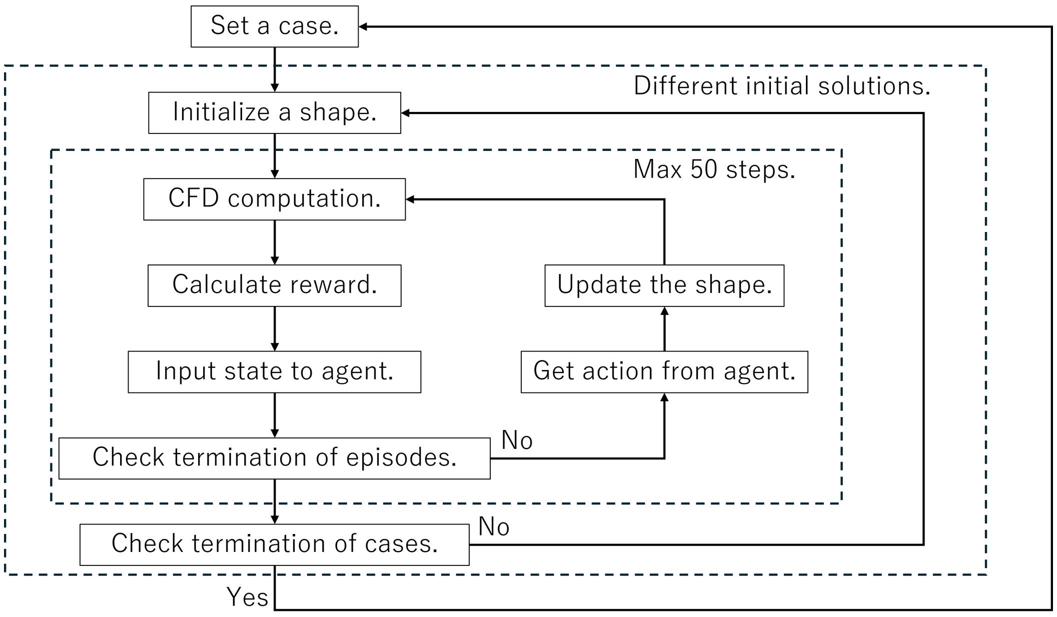


Figure 2. Overview of the DRL-based optimization method.

3. Benchmark Problem

3.1. Problem Definition

To demonstrate the proposed method, we solve the benchmark optimization problem defined in [35]. The benchmark problem is expressed as

$$\text{min.} \quad f_1 = x_1, \tag{4}$$

$$\text{min.} \quad f_2 = gh, \tag{5}$$

$$\text{subject to} \quad g = 1 + 10\frac{\sum_{i=2}^{N} x_i}{N-1}, \tag{6}$$

$$h = \begin{cases} 1 - \left(\frac{f_1}{g}\right)^\alpha, & \text{if } f_1 \le g, \\ 0, & \text{otherwise}, \end{cases} \tag{7}$$

$$x_i \in [-1, 1],\ i = 1, \dots, N. \tag{8}$$

The Pareto front of the problem is analytically solved and written as

$$f_1 = x_1,$$
$$f_2 = 1 - (f_1)^\alpha.$$

The Pareto front is upward concave if $\alpha > 1$, and we use $\alpha = 2$. If we use the reference point $\boldsymbol{p}_0 = (1, 1)$, then the hypervolume is $V(\mathcal{P}) = 1/3$.

3.2. Model Architecture

The state $\boldsymbol{s}_t$ of the model is defined as follows:

$$\boldsymbol{s}_t = (f_1(\boldsymbol{x}_t), f_2(\boldsymbol{x}_t), v(\boldsymbol{x}_t, \mathcal{P}_t)),$$

where $\boldsymbol{v}(\boldsymbol{x}_t, \mathcal{P}_t) \in \Re^2$ denotes the vector from the current solution $\boldsymbol{x}_t$ to the point nearest to the Pareto front. Therefore, the state vector is a four-dimensional vector. The action $\boldsymbol{a}_t \in \Re^2$ is a vector consisting of a continuous variable within the range of $[-0.1, 0.1]$ for each design variable x_1 and x_2.

The architectures of the actor and critic networks are presented in Figure 3. The dropout rate was 0.2. The weights were initialized using HeNormal [36] for the hidden layers and GlorotNormal [37] for the last layer. Table 1 lists the hyperparameters. Default settings were used for hyperparameters except that the learning rate was selected to achieve convergence.

The reward function is the same as that defined in Section 2.3, except for that of "Pattern 1". Since CFD computation is not included in the benchmark problem, "Pattern 1: Failed CFD" (Equation (1)) is not considered.

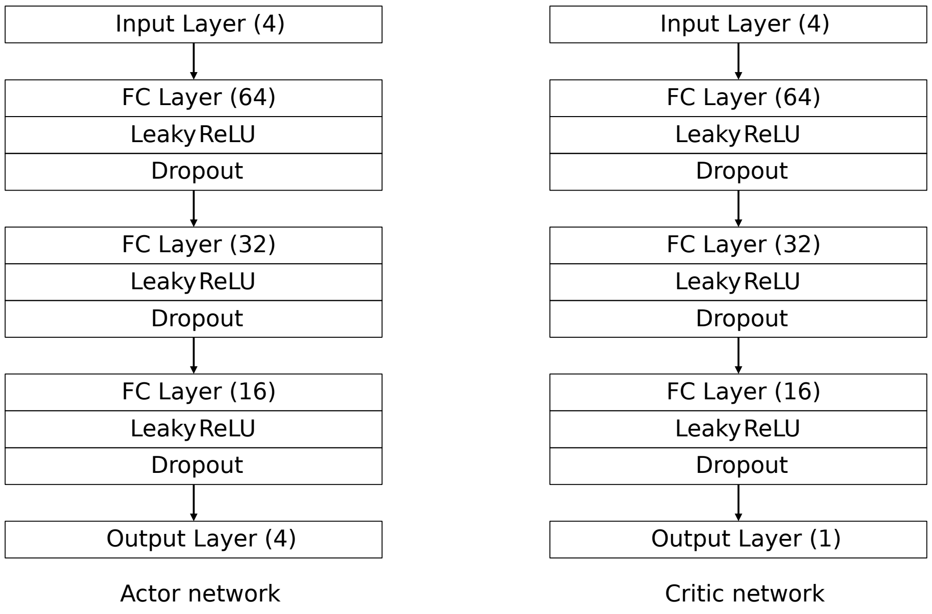


Figure 3. Actor and critic networks for the benchmark problem.

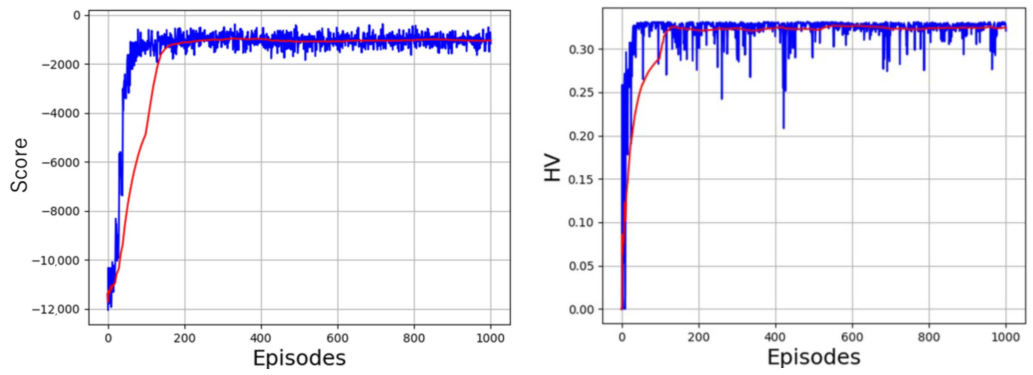
Table 1. Hyperparameters of the benchmark problem.

Hyperparameter	Value	Hyperparameter	Value
Episodes	1000	Batch size	512
Learning rate	10^{-3} to 10^{-5}	Epochs	10
Discount rate γ	0.99	GAE discount rate λ	0.95
Dropout ratio	0.2	Optimization algorithm	Adam

3.3. Results

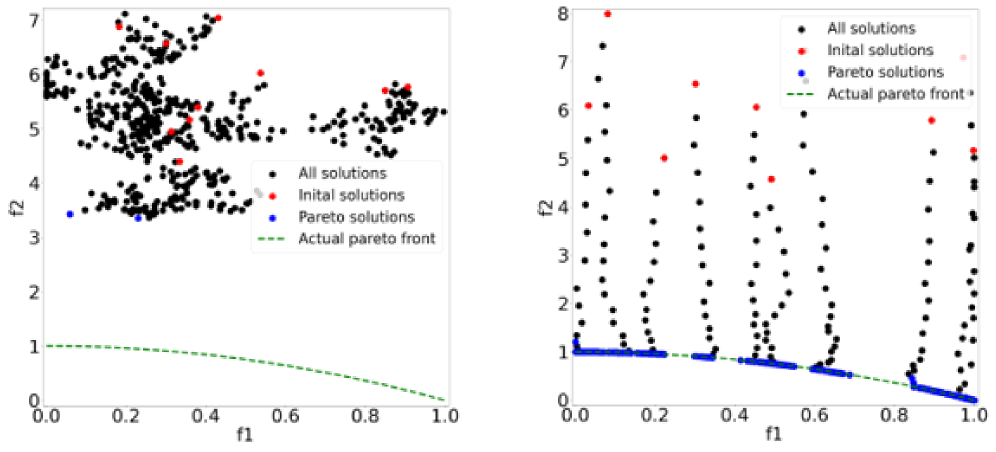
We trained the agent for 1000 episodes, as shown in Table 1. In each episode, the total hypervolume obtained by the agent was calculated, as shown in Figure 4. The moving average of the 10 episodes is also shown in red in Figure 4. The hypervolume after 1000 episodes of training was 0.324, which corresponded to 97.2% of the analytical solution (1/3). Moreover, Figure 4 shows that the score and hypervolume converged within 100 episodes.

The obtained solutions are presented in Figure 5. Without training (episode 0), the agent moved randomly, and Pareto solutions could not be obtained. In episode 1000, the agent succeeded in obtaining Pareto solutions. Although the analytical solution of the Pareto front was upwardly concave, the agent succeeded in obtaining a Pareto front.



(**a**) Score (total reward). (**b**) Hypervolume.

Figure 4. Score and hypervolume of the benchmark problem. (The red line represents the moving average).



(**a**) Episode 0. (**b**) Episode 1000.

Figure 5. Solutions obtained after different episodes.

4. Turbine Optimization Problem

4.1. Problem Definition

We optimized the turbine shape for cascaded turbine blades. We assumed repeated optimization tasks for the problem [12], which is a series of optimization problems whose flow conditions were slightly different from each other. Such optimization tasks are often present in industrial fields. In this study, we assume the flow conditions listed in Table 2. We randomly choose one flow condition from Table 2 and use it in one case. The objective function is to minimize the pressure loss and maximize the torque. The pressure loss is defined as

$$P_{\text{loss}} = \frac{P_{\text{T}_{\text{in}}} - P_{\text{T}_{\text{ex}}}}{P_{\text{T}_{\text{ex}}} - P_{\text{S}_{\text{ex}}}},$$

where $P_{\text{T}_{\text{in}}}$ and $P_{\text{T}_{\text{ex}}}$ are the total pressures at the inlet and outlet boundaries, respectively, and $P_{\text{S}_{\text{ex}}}$ is the static pressure at the outlet boundary. The torque generated by the turbine blade is denoted as F. The optimization problem is formulated as follows:

$$\begin{aligned} &\text{min.} && P_{\text{loss}}, && (9)\\ &\text{min.} && -F, && (10)\\ &\text{subject to} && |\phi_{\text{out}} - \phi_{\text{target}}| \leq 0.5[\text{deg}]. && (11) \end{aligned}$$

The turbine blade shape is defined by adding thickness to the camber line. That is, the upper and lower sides of the blade are defined as $y = y_c + y_t$ and $y = y_c - y_t$, respectively. The camber line is defined as follows

$$y_c = \begin{cases} \frac{m}{p^2}\left(2px - x^2\right), & [0 \leq x \leq p], \\ \frac{m}{(1-p)^2}\left[(1-2p) + 2px - x^2\right], & [p < x \leq 1], \end{cases}$$

where m and p denote the maximum camber and its position, respectively. Subsequently, the shape was rotated according to the origin and scaled such that the chord length became 1 to set a positive stagger angle. The thickness y_t is defined as

$$y_t = \frac{t}{20}\left(0.29690x^{\frac{1}{2}} - 0.12600x - 0.35160^2 + 0.28430x^3 - 0.10150x^4\right),$$

where t is the maximum thickness ratio.

The camber line and thickness distribution are then approximated using B-spline curves. The B-spline curves and control points are presented in Figure 6. Eventually, the blade shape is represented by $y = {y_c}^{\mathrm{B}} + {y_t}^{\mathrm{B}}$ and $y = {y_c}^{\mathrm{B}} - {y_t}^{\mathrm{B}}$, where ${y_c}^{\mathrm{B}}$ and ${y_t}^{\mathrm{B}}$ are the B-spline curves of the camber line and thickness distribution, respectively. The control points of the B-spline curves are used as the design variables.

Table 2. Flow conditions of repeated optimization tasks of turbine blades.

Hyperparameter	Value
Inlet flow angle	0.3–0.5 [rad]
Inlet Mach number	0.2–0.35 [Mach]
Pitch	1.2–1.5 [rad]

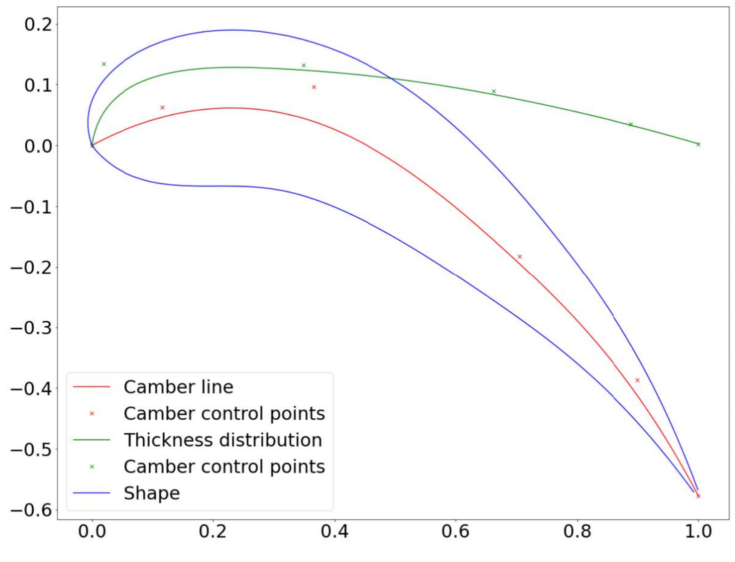


Figure 6. Example of an airfoil shape.

4.2. CFD Computation

The CFD computation is carried out using the MISES software suite, which is a collection of programs for cascade analysis and design [38]. An example computational grid is presented in Figure 7. The pressure distribution is obtained by using MISES. The pressure values are sampled from the pressure distributions on the pressure and suction sides, which are represented by $\boldsymbol{p}_{\mathrm{p}}$ and $\boldsymbol{p}_{\mathrm{s}}$, respectively. In total, 120 samples are collected from each side. The coordinates of the sampled points are represented by $\boldsymbol{s}$, which are the coordinates of the streamline. The pressure values $\boldsymbol{p}_{\mathrm{p}}$ and $\boldsymbol{p}_{\mathrm{s}}$ are normalized using the peak pressure at the leading edge p_0. Hence, $\boldsymbol{p}_\mathrm{p}/p_0$ and $\boldsymbol{p}_\mathrm{s}/p_0$ are used as state variables, as explained in Section 4.3. Figure 8 shows the pressure distributions on the pressure and suction side surfaces.

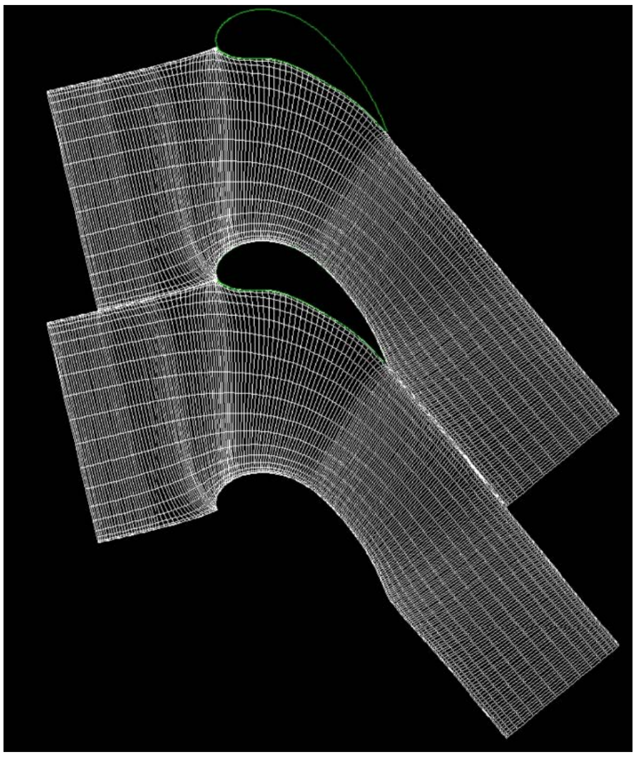


Figure 7. Computation grids for CFD computation.

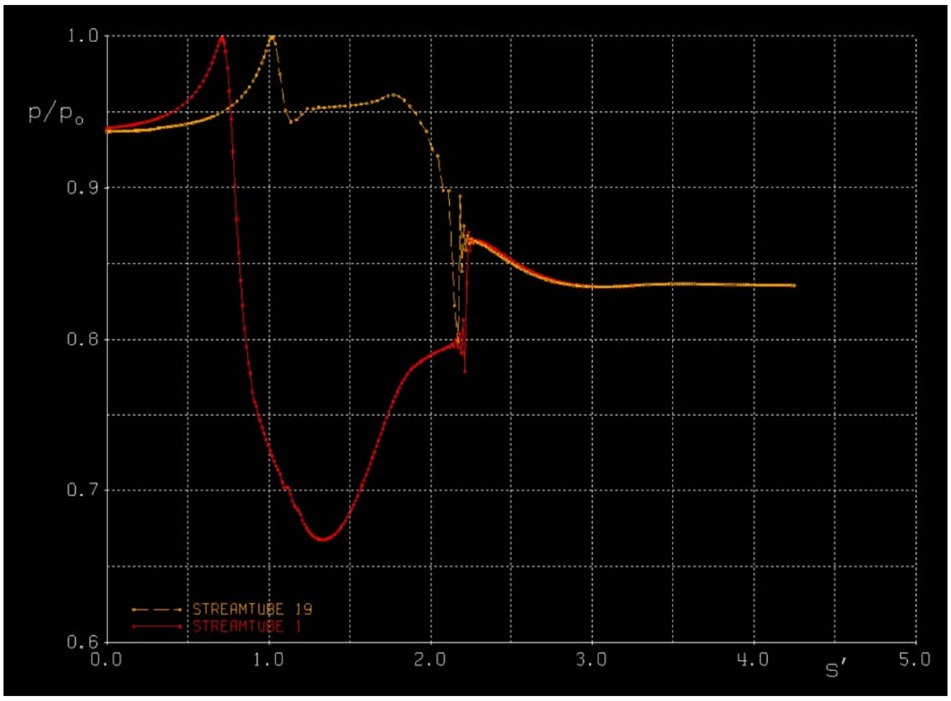


Figure 8. Example of a pressure distribution on the pressure and suction side surfaces.

4.3. Model Architecture

The state variable $\boldsymbol{s}_t$ is constructed using the results of the CFD analysis. The elements of $\boldsymbol{s}_t$ are as follows. In total, the number of dimensions of $\boldsymbol{s}_t$ was 1093.

1. Information of each case:
   - Inlet Mach number,
   - Inlet flow angle,
   - Pitch,
   - Target outlet flow angle.
2. Geometric information:
   - Coordinates of the camber line $(\boldsymbol{x_c}, \boldsymbol{y_c})$,
   - Coordinates of the thickness distribution $(\boldsymbol{x_t}, \boldsymbol{y_t})$,
   - Metal angle of the leading edge.
3. Information of the flow field:
   - Pressure distribution on the pressure side $\boldsymbol{p_p}/p_0$,
   - Pressure distribution on the suction side $\boldsymbol{p_s}/p_0$,
   - Outlet flow angle.
4. Information of the Pareto front:
   - Vector from the current point to the nearest Pareto front.

If the CFD computation failed to converge, information on the flow field could not be obtained. In this case, the following values were used: $\boldsymbol{s} = \boldsymbol{0}$, $\boldsymbol{p}_\mathrm{p}/p_0 = s_\mathrm{p}/p_0 = \boldsymbol{0}$, and an outlet flow angle of 20 [deg].

The action $\boldsymbol{a}_t$ is a vector in $\Re^{17}$ that corresponds to the

1. Difference between the x and y coordinates of the four control points of a camber line,
2. Difference between the x and y coordinates of the four control points of a thickness distribution,
3. Difference of the stagger angle.

The range of $\boldsymbol{a}_t$ was $[-1, 1]$, and $\boldsymbol{a}_t$ was scaled for each design variable. The scale factors for the camber line were 0.003, 0.001 for the thickness distribution, and 0.2 for the stagger angle. As explained in Section 4.1, the turbine shape was generated using the control points and then rotated according to the stagger angle. The shape was then scaled such that the chord line became 1.

Figure 9 illustrates the DNN architectures of the actor and critic networks.

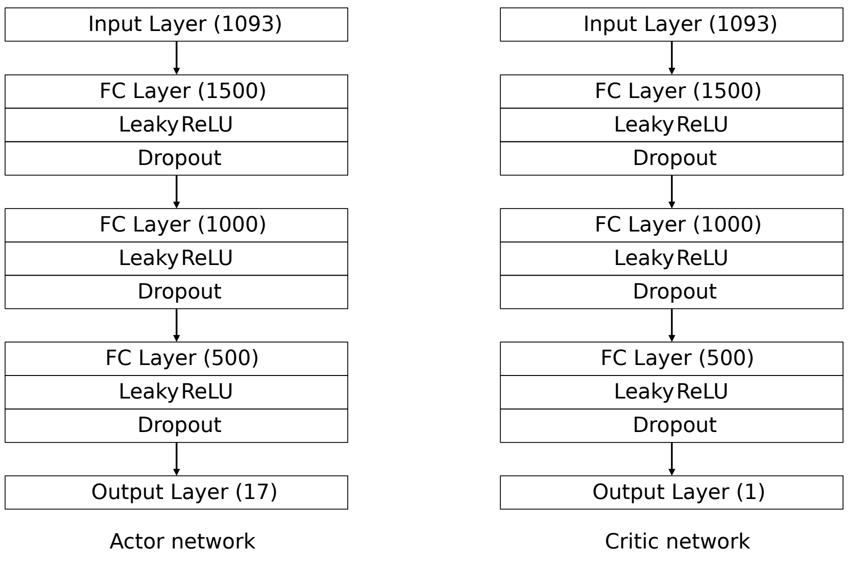


Figure 9. Actor and critic networks for the turbine problem.

4.4. Results

The agent was optimized after training for 1000 episodes. One case was randomly chosen, and 10 initial solutions were used. The results of the objective values are shown in

Figure 10, which indicates that the agent successfully finds Pareto solutions. The hypervolume history is also shown in Figure 11. When a new initial solution is set, the agent succeeds in finding new Pareto solutions. Hence, the hypervolume improves drastically.

The Pareto solutions obtained depend on the initial solution. Therefore, it is necessary to use multiple initial solutions.

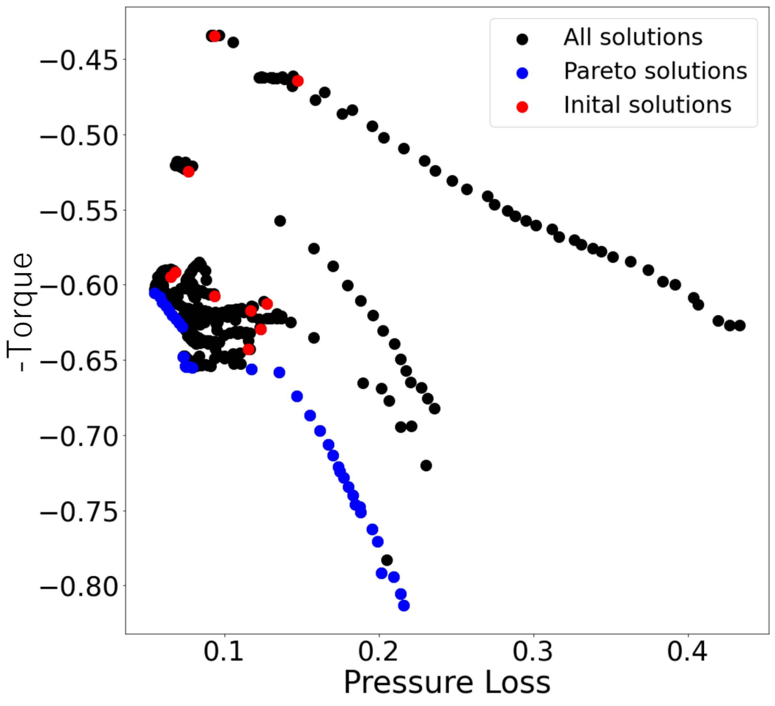


Figure 10. Objective value of the turbine problem.

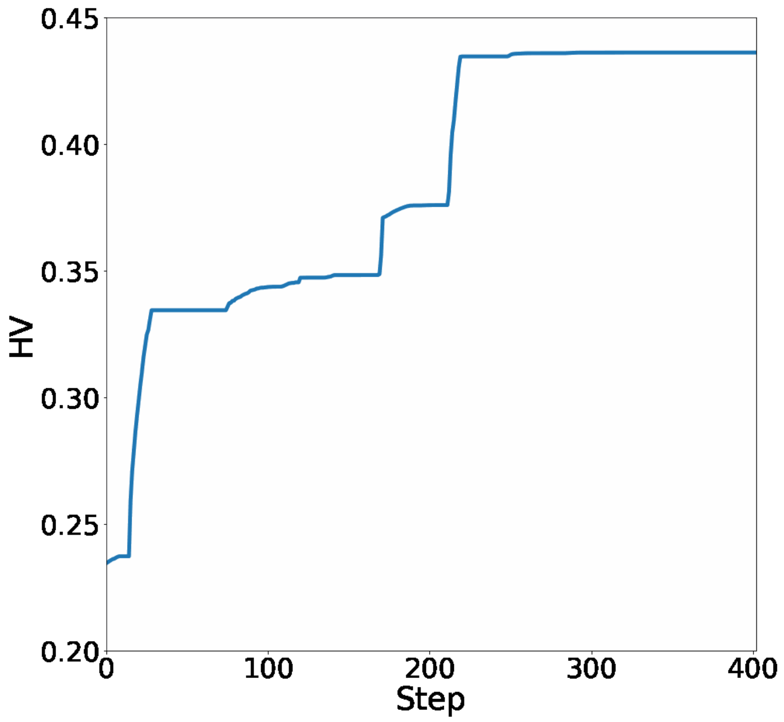


Figure 11. History of the hypervolume (HV) of a turbine problem.

5. Conclusions

A novel multi-objective optimization method using the hypervolume of the Pareto front is proposed for a turbine optimization problem. This method uses the improvement in hypervolumes as a reward. The proposed method was validated using a benchmark problem and then applied to a turbine optimization problem.

In the hypervolume-based method, the objective function and constraint conditions are not changed. Hence, only one DRL agent is required to obtain Pareto solutions. If we use the WSM or ε-constrained method, then different DRL agents must be trained when the weights or constraints are changed. This generalization capability reduces the computational cost. Moreover, because DRL has a rich generalization capability, it can be used for multiple tasks.

Author Contributions: Conceptualization, K.Y.; methodology, K.Y. and R.Y.; software, K.Y. and R.Y.; validation, K.Y. and R.Y.; formal analysis, K.Y. and R.Y.; investigation, K.Y. and R.Y.; resources, K.Y.; data curation, R.Y.; writing—original draft preparation, K.Y.; writing—review and editing, K.Y.; visualization, R.Y.; supervision, K.Y., S.O. and K.S.; project administration, K.Y.; funding acquisition, K.Y. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by JSPS KAKENHI Grant Number JP23K13239.

Informed Consent Statement: Not applicable.

Data Availability Statement: Data are available upon request.

Conflicts of Interest: The first author is a board member of MJOLNIR SPACEWORKS.

References

1. Morales, E.F.; Murrieta-Cid, R.; Becerra, I.; Esquivel-Basaldua, M.A. A survey on deep learning and deep reinforcement learning in robotics with a tutorial on deep reinforcement learning. *Intell. Serv. Robot.* **2021**, *14*, 773–805. [CrossRef]
2. Zhu, K.; Zhang, T. Deep reinforcement learning based mobile robot navigation: A review. *Tsinghua Sci. Technol.* **2021**, *26*, 674–691. [CrossRef]
3. Zhao, W.; Queralta, J.P.; Westerlund, T. Sim-to-Real Transfer in Deep Reinforcement Learning for Robotics: A Survey. In Proceedings of the 2020 IEEE Symposium Series on Computational Intelligence (SSCI), Canberra, ACT, Australia, 1–4 December 2020; pp. 737–744. [CrossRef]
4. Liu, R.; Qu, Z.; Huang, G.; Dong, M.; Wang, T.; Zhang, S.; Liu, A. DRL-UTPS: DRL-Based Trajectory Planning for Unmanned Aerial Vehicles for Data Collection in Dynamic IoT Network. *IEEE Trans. Intell. Veh.* **2023**, *8*, 1204–1218. [CrossRef]
5. Li, K.; Zhang, T.; Wang, R. Deep Reinforcement Learning for Multiobjective Optimization. *IEEE Trans. Cybern.* **2021**, *51*, 3103–3114. [CrossRef] [PubMed]
6. Liao, J.; Liu, T.; Tang, X.; Mu, X.; Huang, B.; Cao, D. Decision-Making Strategy on Highway for Autonomous Vehicles Using Deep Reinforcement Learning. *IEEE Access* **2020**, *8*, 177804–177814. [CrossRef]
7. Wang, H.; Yuan, S.; Guo, M.; Li, X.; Lan, W. A deep reinforcement learning-based approach for autonomous driving in highway on-ramp merge. *Proc. Inst. Mech. Eng. Part D J. Automob. Eng.* **2021**, *235*, 2726–2739. [CrossRef]
8. Ramu, P.; Thananjayan, P.; Acar, E.; Bayrak, G.; Park, J.W.; Lee, I. A survey of machine learning techniques in structural and multidisciplinary optimization. *Struct. Multidiscip. Optim.* **2022**, *65*, 266. [CrossRef]
9. Yonekura, K.; Hattori, H. Framework for design optimization using deep reinforcement learning. *Struct. Multidiscip. Optim.* **2019**, *60*, 1709–1713. [CrossRef]
10. Lou, J.; Chen, R.; Liu, J.; Bao, Y.; You, Y.; Chen, Z. Aerodynamic optimization of airfoil based on deep reinforcement learning. *Phys. Fluids* **2023**, *35*, 037128. [CrossRef]
11. Du, Q.; Liu, T.; Yang, L.; Li, L.; Zhang, D.; Xie, Y. Airfoil design and surrogate modeling for performance prediction based on deep learning method. *Phys. Fluids* **2022**, *34*, 015111. [CrossRef]
12. Yonekura, K.; Hattori, H.; Shikada, S.; Maruyama, K. Turbine blade optimization considering smoothness of the Mach number using deep reinforcement learning. *Inf. Sci.* **2023**, *642*, 119066. [CrossRef]
13. Vignon, C.; Rabault, J.; Vinuesa, R. Recent advances in applying deep reinforcement learning for flow control: Perspectives and future directions. *Phys. Fluids* **2023**, *35*, 031301. [CrossRef]
14. Tang, H.; Rabault, J.; Kuhnle, A.; Wang, Y.; Wang, T. Robust active flow control over a range of Reynolds numbers using an artificial neural network trained through deep reinforcement learning. *Phys. Fluids* **2020**, *32*, 053605. [CrossRef]
15. Jang, S.; Yoo, S.; Kang, N. Generative Design by Reinforcement Learning: Enhancing the Diversity of Topology Optimization Designs. *Comput.-Aided Des.* **2022**, *146*, 103225. [CrossRef]
16. Deb, K. *Multi-Objective Optimization Using Evolutionary Algorithms*; Wiley: Hoboken, NJ, USA, 2001.

17. Murata, T.; Ishibuchi, H. MOGA: Multi-objective genetic algorithms. In Proceedings of the 1995 IEEE International Conference on Evolutionary Computation, Perth, WA, Australia, 29 November–1 December 1995; Volume 1, p. 289. [CrossRef]
18. Srinivas, N.; Deb, K. Muiltiobjective Optimization Using Nondominated Sorting in Genetic Algorithms. *Evol. Comput.* **1994**, *2*, 221–248. [CrossRef]
19. 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]
20. Goldberg, D.E. *Genetic Algorithms in Search, Optimization, and Machine Learning*; Addison-Wesley: Hoboken, NJ, USA, 1989. [CrossRef]
21. Dussauge, T.P.; Sung, W.J.; Fischer, O.J.P.; Mavris, D.N. A reinforcement learning approach to airfoil shape optimization. *Sci. Rep.* **2023**, *13*, 9753. [CrossRef]
22. Keat, E.Y.; Sharef, N.M.; Yaakob, R.; Kasmiran, K.A.; Marlisah, E.; Mustapha, N.; Zolkepli, M. Multiobjective Deep Reinforcement Learning for Recommendation Systems. *IEEE Access* **2022**, *10*, 65011–65027. [CrossRef]
23. Al-Jumaily, A.; Mukaidaisi, M.; Vu, A.; Tchagang, A.; Li, Y. Examining multi-objective deep reinforcement learning frameworks for molecular design. *Biosystems* **2023**, *232*, 104989. [CrossRef]
24. Yang, X.S. Chapter 14—Multi-Objective Optimization. In *Nature-Inspired Optimization Algorithms*; Yang, X.S., Ed.; Elsevier: Oxford, UK, 2014; pp. 197–211. [CrossRef]
25. Kaim, A.; Cord, A.F.; Volk, M. A review of multi-criteria optimization techniques for agricultural land use allocation. *Environ. Model. Softw.* **2018**, *105*, 79–93. [CrossRef]
26. Kalayci, C.B.; Ertenlice, O.; Akbay, M.A. A comprehensive review of deterministic models and applications for mean-variance portfolio optimization. *Expert Syst. Appl.* **2019**, *125*, 345–368. [CrossRef]
27. Mesquita-Cunha, M.; Figueira, J.R.; Barbosa-Póvoa, A.P. New ϵ-constraint methods for multi-objective integer linear programming: A Pareto front representation approach. *Eur. J. Oper. Res.* **2023**, *306*, 286–307. [CrossRef]
28. Yang, Z.; Cai, X.; Fan, Z. Epsilon constrained method for constrained multiobjective optimization problems: Some preliminary results. In Proceedings of the Companion Publication of the 2014 Annual Conference on Genetic and Evolutionary Computation, New Vancouver, BC, Canada, 12–16 July 2014; GECCO Comp'14, pp. 1181–1186. [CrossRef]
29. Becerra, R.L.; Coello, C.A.C. Solving Hard Multiobjective Optimization Problems Using ϵ-Constraint with Cultured Differential Evolution. In Proceedings of the Parallel Problem Solving from Nature—PPSN IX, 9th International Conference, Reykjavik, Iceland, 9–13 September 2006; Runarsson, T.P., Beyer, H.G., Burke, E., Merelo-Guervós, J.J., Whitley, L.D., Yao, X., Eds.; Springer: Berlin/Heidelberg, Germany, 2006; pp. 543–552.
30. Shang, K.; Ishibuchi, H.; He, L.; Pang, L.M. A Survey on the Hypervolume Indicator in Evolutionary Multiobjective Optimization. *IEEE Trans. Evol. Comput.* **2021**, *25*, 1–20. [CrossRef]
31. Auger, A.; Bader, J.; Brockhoff, D.; Zitzler, E. Hypervolume-based multiobjective optimization: Theoretical foundations and practical implications. *Theor. Comput. Sci.* **2012**, *425*, 75–103. [CrossRef]
32. Shang, K.; Ishibuchi, H. A New Hypervolume-Based Evolutionary Algorithm for Many-Objective Optimization. *IEEE Trans. Evol. Comput.* **2020**, *24*, 839–852. [CrossRef]
33. Guerreiro, A.P.; Fonseca, C.M.; Paquete, L. The Hypervolume Indicator: Computational Problems and Algorithms. *ACM Comput. Surv.* **2021**, *54*, 119. [CrossRef]
34. Schulman, J.; Wolski, F.; Dhariwal, P.; Radford, A.; Klimov, O. Proximal Policy Optimization Algorithms. *arXiv* **2017**, arXiv:1707.06347.
35. Deb, K.; Pratap, A.; Meyarivan, T. Constrained Test Problems for Multi-objective Evolutionary Optimization. In Proceedings of the Evolutionary Multi-Criterion Optimization, First International Conference, EMO 2001, Zurich, Switzerland, 7–9 March 2001; Zitzler, E., Thiele, L., Deb, K., Coello Coello, C.A., Corne, D., Eds.; Springer: Berlin/Heidelberg, Germany, 2001; pp. 284–298.
36. He, K.; Zhang, X.; Ren, S.; Sun, J. Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification. In Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV), Santiago, Chile, 7–13 December 2015; pp. 1026–1034. [CrossRef]
37. Glorot, X.; Bengio, Y. Understanding the difficulty of training deep feedforward neural networks. In Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics, Sardinia, Italy, 13–15 May 2010; Teh, Y.W., Titterington, M., Eds.; Proceedings of Machine Learning Research; Volume 9, pp. 249–256.
38. Drela, M.; Youngren, H. *A User's Guide to MISES 2.63*; MIT Aerospace Computational Design Laboratory: Cambrige, MA, USA, 2008.