A Novel Adaptive Weight Selection Algorithm for Multi-Objective Multi-Agent Reinforcement Learning

Kristof Van Moffaert, Tim Brys, Arjun Chandra, Lukas Esterle, Peter R. Lewis and Ann Nowé

***Abstract*— To solve multi-objective problems, multiple reward signals are often scalarized into a single value and further processed using established single-objective problem solving techniques. While the field of multi-objective optimization has made many advances in applying scalarization techniques to obtain good solution trade-offs, the utility of applying these techniques in the *multi-objective multi-agent* learning domain has not yet been thoroughly investigated. Agents learn the value of their decisions by linearly scalarizing their reward signals at the local level, while acceptable system wide behaviour results. However, the non-linear relationship between weighting parameters of the scalarization function and the learned policy makes the discovery of system wide trade-offs time consuming.**

Our first contribution is a thorough analysis of well known scalarization schemes within the multi-objective multi-agent reinforcement learning setup. The analysed approaches intelligently explore the weight-space in order to find a wider range of system trade-offs. In our second contribution, we propose a novel adaptive weight algorithm which interacts with the underlying local multi-objective solvers and allows for a better coverage of the Pareto front. Our third contribution is the experimental validation of our approach by learning bi-objective policies in self-organising smart camera networks. We note that our algorithm (i) explores the objective space faster on many problem instances, (ii) obtained solutions that exhibit a larger hypervolume, while (iii) acquiring a greater spread in the objective space.

I. Introduction

Many optimization problems that need to be solved nowadays are in essence tasks that involve more than one objective. Multi-objective reinforcement learning (MORL) is an extension to reinforcement learning (RL) where the environment provides the agent with multiple feedback signals. Usually in MORL, a weighted linear *scalarization* function is used to translate the original multi-objective problem into a single-objective problem. The weight parameters $w_o \in [0, 1]$ are *preference* factors that identify the relative importance of objective o, with $\sum_{o=1}^{m} w_o = 1$ for m objectives. The goal of MORL is to search the policy space and eventually find policies that provide different trade-offs between the objectives. In a multi-agent environment however, finding system-wide trade-offs is still an open question.

Let us highlight the difficulties by considering the following example. A system engineer wants to optimize packet latency and energy consumption in the routing of his wireless sensor network. In such problems where multiple objectives need to be optimized at the same time, it is not always clear from the problem description (if any) how the objectives influence each other and which parameter values are required to obtain the requested balance. Let us say that the engineer is interested in policies that focus on optimizing the packet latency for 60% and 40% on reducing the energy consumption. Naively, the engineer would set w_1 equal to 0.6 and w_2 to 0.4 for the two objectives. A first difficulty arises as the linear scalarization function has the fundamental limitation that the correspondence between weights and policies is not clear. More precisely, a uniform sampling of weights usually does not result in a uniform sampling of the Pareto optimal set [1], due to different scalings, and the shape of the Pareto front. Secondly, the wireless sensor network, being a cooperative multi-agent infrastructure, introduces an additional level of complexity as the agents can influence each other. The quality of the global solution depends on the local interactions by these agents. These problems require system designers to spend a lot of time on fine-tuning the scalarization weights, in order to find a range of trade-off policies they can then choose from.

The contributions in this paper are threefold. Firstly, we analyse several algorithms that explore the weight space in either a predefined or adaptive manner to obtain trade-off solutions given a limited number of scalarizations to try. Our results highlight their shortcomings in terms of spread in the objective space. Secondly, we propose a new adaptive weight algorithm (AWA) which obtains improved trade-off solutions in terms of hypervolume and spread, when compared to the state-of-the-art. Finally, we apply these algorithms on a multi-objective multi-agent smart camera problem, where the goal of the agents is to cooperate in order to optimize two system-wide conflicting objectives. We are able to improve over the standard weight selection procedures on several quality indicators from the multi-objective optimization field. AWAs guide the multiple agents from the top-down towards executing their individual action selection procedures, giving rise to highly expressive system wide behaviours. This expressiveness can help practitioners with an insightful deployment of agents within real world scenarios that resemble the ones analysed.

This paper is organized as follows. In Section II, we describe necessary concepts such as multi-objective reinforcement learning. In Section III, we empirically evaluate the current methods in a multi-objective multi-agent simulator and highlight their shortcomings. In Section IV, we present our novel algorithm for steering the weights of the underlying

Kristof Van Moffaert, Tim Brys and Ann Nowé are with the Department of Computer Science, Vrije Universiteit Brussel, Brussels, Belgium (email: {kvmoffae,timbrys,anowe}@vub.ac.be). Lukas Esterle is with the Alpen-Adria Universität Klagenfurt and Lakeside Labs (email: lukas.esterle@aau.at). Arjun Chandra is with the University of Oslo (email: chandra@ifi.uio.no) and Peter R. Lewis is with Aston University (email: p.lewis@aston.ac.uk).

optimization algorithm along with experimental results. In Section V, we summarize the paper and form conclusions.

II. Related work

A. Multi-objective reinforcement learning

Multi-objective reinforcement learning (MORL) is an extension to standard reinforcement learning where the environment consists of two or more feedback signals, i.e.

$$\mathbf{R}(s_i, a_i) = [R_1(s_i, a_i), \ldots, R_m(s_i, a_i)] \tag{1}$$

where m is the number of objectives. In MORL, a solution is a policy π, evaluated by its expected return $\mathbf{J}^\pi$, a vector of expected discounted returns for each objective. Thus,

$$\mathbf{J}^\pi \equiv \left[E\left[\sum_{t=0}^{\infty} \gamma^t R_1(s_t, \pi(s_t))\right], \ldots, E\left[\sum_{t=0}^{\infty} \gamma^t R_m(s_t, \pi(s_t))\right]\right] \tag{2}$$

Since the environment now consists of multiple objectives, conflicts can arise when trying to simultaneously optimize the objectives. In such cases, trade-offs between these objectives have to be learnt, resulting in a set of policies. A policy x_1 is said to strictly dominate another policy x_2, i.e. $x_2 \prec x_1$, if performance on each objective by x_1 is not strictly less than the corresponding performance of x_2 and performance in at least one objective is strictly greater. If x_1 improves on x_2 on some objective and x_2 also improves on x_1 on one or more objectives, x_1 and x_2 are said to be incomparable. The set of non-dominated policies is referred to as the *Pareto front*. In [11], a general framework for MORL algorithms was proposed that extends the scalar $\hat{Q}$-values to $\hat{\mathbf{Q}}$-vectors that store a $\hat{Q}$-value for each objective, i.e.

$$\hat{\mathbf{Q}}(s, a) = \left[\hat{Q}_1(s, a), \ldots, \hat{Q}_m(s, a)\right] \tag{3}$$

Current approaches in MORL often use *scalarization* functions [11], [10] to reduce the dimensionality of the underlying multi-objective environment to a single scalar. Scalarization functions often imply that an objective o is associated with a weighted coefficient, which allows the user some control over the nature of the policy found by the system, by placing greater or lesser emphasis on each objective. In a multi-objective environment, this trade-off is parametrized by $w_o \in [0, 1]$ for objective o and $\sum_{o=1}^{m} w_o = 1$. In most cases, a linear combination of the objectives is considered, i.e. $\sum_{o=1}^{m} w_o \cdot \hat{\mathbf{Q}}_o(s, a)$. Usually, only a limited set of scalarizations can be tried and evaluated. Therefore, the goal is to select these scalarizations so that the resulting policies are not only (near) optimal but also spread in the objective space. The spread indicator is important to guarantee that the set of policies is not clustered into particular areas of the objective space, but identifies diverse trade-offs that the user can choose from. In the cooperative multi-agent case, agents must coordinate their local actions in order to maximize system-wide performance, i.e. actions take place at the local agent level, while performance is measured at the global level.

B. Adaptive weight algorithms

The weights of the linear scalarization function, however, only provide minimal guidance to the learning algorithm and only in very limited cases does an even spread of w_o guarantee an even spread on the Pareto front. In general, specifying particular weights does not guarantee that the solutions found are in correspondence. Using the example of Section I, weights 0.6 and 0.4 do not guarantee that the final solution will have a performance focussed 60% on optimizing the packet latency and 40% on energy consumption. Das and Dennis argue [1] that it is not possible to know the specific weights needed to obtain evenly spread solutions on the Pareto front without actually knowing the shape of the Pareto front. Therefore, researchers in the multi-objective optimization field and more precisely local search, have investigated AWAs that alter the weight parameter based on several measures. A recent proposal is *two-phase local search* (TPLS) [8]. TPLS is a powerful algorithmic framework that comprises two phases. In a first phase, a single-objective local search algorithm obtains a high quality[1] solution for one of the objectives, while in the second phase this solution serves as a starting point for a sequence of *scalarizations*, i.e. weights. The goal is to find a set of high quality yet diverse trade-off solutions, by evaluating a sequence of weights. Some TPLS variants use a predefined sequence, while others select the new weight as a function of the solutions already found, and their coverage of the Pareto front. We summarize these existing variants below.

TPLS. The standard procedure defines a sequence for w_1 ranging from 0 to 1 with steps of $\frac{1}{N_{scalar}}$, where N_{scalar} determines the stepsize. The weight for the other objective, i.e. w_2 is calculated by $w_2 = 1 - w_1$ for the bi-objective case. However, although the method performs a uniform sampling of the weight space, there is no guarantee that the resulting solutions will yield a uniform spread in the objective space [1]. Another crucial aspect of TPLS is the sequential order of weights. When TPLS is stopped prematurely, it will not have sampled the latter part of the weight space, potentially leaving a part of the Pareto front uncharted. Hence, TPLS does not produce solutions as good as possible as fast as possible, i.e. it has poor *anytime* behaviour.

RA-TPLS. To overcome these problems, the *Regular Anytime* TPLS algorithm (RA-TPLS) was proposed [2]. The algorithm explores the weight space in a divide-and-conquer manner by progressively exploring finer levels k. RA-TPLS starts at evaluating $w_1 = 0$ and $w_1 = 1$ at level 0. At the next level, when $k = 1$, $w_1 \in \{0.5\}$. At level $k = 2$, $w_1 \in \{0.25, 0.75\}$ and so forth. As the search continues, the coverage of the weight space is refined; at any time, the search effort is (almost) evenly distributed across the possible weight settings.

[1]In multi-objective problems, the quality of a set of solutions is often measured in terms of the hypervolume measure which calculates the volume the Pareto dominating solutions occupy in the objective space for a given reference point [8], [9]

AN-TPLS. The previous two methods generate weights in a predefined manner. However, sometimes the shape of the Pareto front is irregular and the search direction should be adapted by taking into account the actual shape of the Pareto front. *Adaptive Normal* TPLS defines a norm to identify the largest gap in the coverage of the Pareto front [2]. Between all the currently obtained trade-offs, the pair with the largest gap according to the norm specified (Euclidean distance in this variant) is used to calculate the next weight, aiming to fill this largest gap. The new weight w_1 is perpendicular to the line between the objective function f of solutions s_1 and s_2 defining the largest gap in the objective space [2] (assuming s_1 and s_2 are normalized by the smallest and largest currently found objective values):

$$w_1 = \frac{f_2(s_1) - f_2(s_2)}{f_2(s_1) - f_2(s_2) + f_1(s_2) - f_1(s_1)} \tag{4}$$

AN-TPLS-HV. An extension to the standard adaptive TPLS algorithm of above uses an alternative norm to specify a distance measure. The hypervolume measure is employed to measure the size of the gap in the Pareto front [2]. Given two solutions s_1 and s_2, the hypervolume measure calculates the rectangle defined in the objective space:

$$HV(s_1, s_2) = |(f_1(s_1) - f_1(s_2)) \cdot (f_2(s_1) - f_2(s_2))| \tag{5}$$

Although these methods were developed with local search algorithms in mind, they can be adapted to interact with other optimization algorithms, such as reinforcement learning. Note that there exist methods that can efficiently calculate Pareto front policies and their corresponding weights from batch RL data [7]. These algorithms exploit the multi-objective nature of the problem by learning multiple policies at the same time, but these algorithms are only applicable to problems that only involve one agent and where sufficient off-line data is available. In the case of a multi-agent problem the stakes are higher because the problem involves a truly on-line and distributed setting where each agent only executes a single policy at a time. The system also introduces an emergent complexity due to agents having only bounded (local) knowledge. Therefore, the agents would have to cooperate towards the same goal, i.e. attaining a specific system wide reward vector as a product of local coordinated interactions. How these system wide performance should be accomplished in a cooperative multi-agent setting is however left unanswered. To tackle this uncharted research question, we will experimentally validate the previously described adaptive weight schemes in a simulation environment of smart camera networks, called *CamSim*.

III. Experimental validation

Before we proceed to the results, we depict the properties of the existing AWAs in our case study simulation environment and highlight the results. Afterwards, in Section IV, we will propose a novel extension which overcomes much of the limitations of the current AWAs.

A. *CamSim environment*

CamSim [4] simulates a distributed smart camera network. Smart cameras are fully computationally capable devices endowed with a visual sensor, and typically run computer vision algorithms to analyse captured images. Where standard cameras can only provide plain images and videos, smart cameras can pre-process these videos and provide users with aggregated data and logical information, such as the presence or not of an object of interest. Since smart cameras are designed to have a low energy footprint, their processing capabilities are also low. Communication between cameras allows the network as a whole to track objects in a distributed fashion, handing over object tracking responsibilities from camera to camera as objects move through the environment. In one approach [5], cameras exchange object tracking responsibilities through auctions, sending auction invitations to other cameras, who may then bid to buy objects. The cameras use pheromone-based on-line learning to determine which other cameras they trade with most often. This neighbourhood relationship graph (the vision graph), enables them to selectively target their auction invitations and achieve higher levels of efficiency. In [6], six different behavioural strategies were available to cameras, which determined the level of marketing activity they undertook, given the learnt vision graph. Some strategies incurred higher levels of communication overhead but typically obtained higher levels of tracking confidence; other strategies obtained the opposite results. However, the trade-off realised by each strategy was found to be highly scenario dependent; as camera positions varied and object movements differed, the relative benefits of the strategies was greatly influenced.

Although cameras make decisions based on local information, we are primarily interested in performance at the global level. This consists of two network-level objectives:

1) *Tracking confidence*, the achieved tracking confidence during a small time window for each object by the camera tracking that object, summed over all objects. (*Maximize*)
2) *Number of auction invitations*, the number of invitations sent by all cameras as a result of auction initiations, during a small time window, a proxy for communication and processing overhead. (*Minimize*)

The camera agents are single-state independent learners and can choose between six marketing strategies defining each agents behaviour. The scalarized reward r_{total} is camera specific reward and is a weighted-sum of the *utility* reward $r_{utility}$ and the negative *auction invitation* reward $r_{auction}$, given w_0 for the first objective and $(1 - w_o)$ for the second objective:

$$r = w_0 \times r_{utility} + (1 - w_0) \times -r_{auction} \tag{6}$$

The utility reward of a camera i is calculated by

$$r_{utility} = \sum_{j \in O_i} [c_j \cdot v_j \cdot \phi_i(j)] - p + r \tag{7}$$

Here, v_j is a visibility parameter which is determined by the distance and angle of the observed object to the observing camera. The tracking performance is estimated by a confidence value c_j. Both values c_j and v_j are between 0 and 1 as soon as the observed object is within the field of view of a camera, 0 otherwise. $\phi_i : O_i \to 0, 1$ is 1 if camera i attempts to track object j and 0 otherwise. In addition to utility earned by tracking objects, a camera b may make a payment to another camera s in order to "buy" the right to track an object from that camera. This requires that the "selling" camera s already itself owns the object. If an exchange is agreed, then the object is removed from O_s and added to O_b. p denotes the sum of all payments made in trades in that iteration, and r conversely denotes the sum of all payments received [3].

The $r_{auction}$ reward denotes the number of auction invitations sent by this camera at the current time step. Our aim is to minimize the number of auction invitations, but traditionally, RL concerns a maximization problem. Therefore, we want to maximize the negative number of auction invitations in Equation 6. The agents use a softmax action selection strategy with τ equal to 0.2. For more information on the details behind these marketing strategies, we refer to [6].

For the purposes of our evaluation, a scenario comprises a set of cameras with associated positions and orientations, along with a set of objects and their movement paths through the environment. In this paper, we simulate and evaluate configurations within 11 qualitatively different scenarios using the open source CamSim software. We also acquired video feed data from a real smart camera network, which gives us a twelfth scenario. All simulated scenarios are depicted in Figure 1, where a dot represents a camera and the associated triangle represents its field of view.

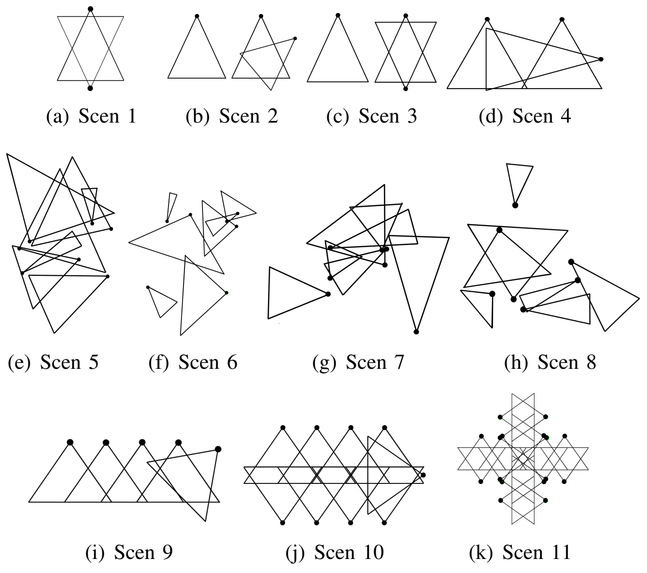


Fig. 1: The scenarios tested with the *CamSim* simulation tool. A dot represents a camera, the associated triangle represents its field of view.

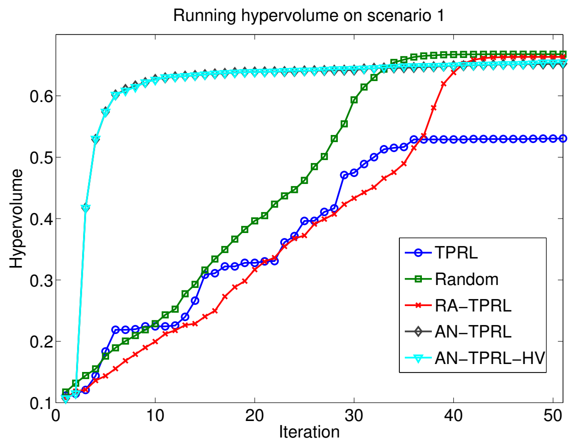


Fig. 2: The hypervolume over time for each of the adaptive weight algorithms on scenario 1.

B. Results I

We will now present the results of applying the adaptive two-phase weight schemes in combination with reinforcement learning agents in CamSim. There are two main side marks. First, note that we do not use local search techniques as in the original TPLS proposals [2] so therefore we refer to these implementations as two-phase reinforcement learning techniques or TPRL. Secondly, in the current setup, the weight parameter used in each iteration of the simulation is the same for all agents. In future work, we will analyse whether it is beneficial to assign different weights to different (sets of) agents to let a *division of labour* emerge.

In Fig. 2 we analyse the *anytime* property of each method, i.e. how fast does the algorithm explore the non-dominated parts of the Pareto front in terms of the hypervolume measure. We focus on scenario 1, but the conclusions generalise to the other scenarios as well. Note that each iteration represents the average value over 10 episodes with a specific scalarization, determined by an AWA. One episode is itself 1000 simulations runs. The uniform distribution in the weight space, i.e. TPRL, is the original AWA used in [6] on this problem. We clearly see that the naive methods such as TPRL, random and RA-TPRL explore the objective space quite slowly in terms of the hypervolume measure. The adaptive methods such as AN-TPRL and AN-TPRL-HV which adapt their weights by considering the 'gap' between solutions in the objective space perform roughly the same on this scenario, i.e. we note that the hypervolume increases rapidly in early stages, while it stagnates after 25 scalarizations. In the end, the performance approaches RA-TPRL and randomly exploring the weight space.

In Fig. 5 (a) to (e), we denote the final Pareto front obtained by each of the methods[2]. We note that some methods are better at dividing the computational effort across the objective space. For example, Fig. 5 (a) is a clear indication that a uniform distribution on the weight space as with *TPRL*

[2]We normalized the values of the solutions for each of the methods and transformed them in order to create a maximization problem for both objectives.

does not guarantee a uniform spread in the objective space. *AN-TPRL* and *AN-TPRL-HV*, in Fig. 5 (d) and (e), adapt the weights in terms of the Euclidean and hypervolume norm, respectively. However, those algorithms focus their resources on particular areas, while leaving other, possibly interesting, trade-off solutions uncharted. In the following section, we highlight the reasons for the limited coverage of the objectives space of these methods.

IV. An improved AWA

The AWAs of Section II-B are experimentally shown to be very successful in a specific application domain. However, we note some properties of the current methods that limit their applicability in other research domains.

First, those AWAs are tailored for stochastic local search algorithms that can be seeded using particular solutions to provide the optimization algorithm with a good initial position in the search space. The AWAs make extensive use of this property to bias the search direction to fruitful and uncharted areas of the Pareto front. However, seeding the reinforcement learning agent in such a way is not possible as most problems tend to be episodic and consist of multiple stages that the agent has to go through.

Secondly, the dichotomic scheme in Eq. 4 uses the segment between two solutions s_1 and s_2 to calculate the new weight. However, the equation does not guarantee that the resulting solution obtained by the calculated weight will lie between the interval of the two parent solutions, i.e. s_1 and s_2. The only assertion that holds is that when the search is seeded from one of these two parent solutions, the resulting solutions will be left of the segment between the two parent solutions of the interval. In case one starts from a random solution, this assertion ceases to hold altogether. These limitations were experimentally highlighted in Section III-B.

The solution that we propose is to combine the properties of RA-TPLS and AN-TPLS, meaning that we use the layered-division of the weight space where deeper layers intensify the search process to particular areas of RA-TPLS together with an adaptive ordering of the elements of the different layers, based on the aspects of the Pareto front currently being explored. Similarly to AN-TPLS we use different norms, such as the Euclidean distance and the hypervolume indicator, to qualify the difference between solutions. By merging both procedures, we combine the best of both worlds, i.e. the layered approach allows us to explore the rough outline of the Pareto front in initial iterations of the algorithm and secondly, we do not rely on the dichotomic scheme in Eq. 4 which does not scale well to general application domains. Subsequently, we no longer require specific seeds that bias the search direction and eventually also the performance of the obtained solutions. We call this algorithm *RA-TPRL-DIST* and *RA-TPRL-HV* with the Euclidean and hypervolume norm, respectively. An outline of the procedure for the bi-objective case is given in Algorithm 3. The algorithms starts by examining the bounds of the Pareto front, i.e. we evaluate the optimization algorithm, in this case $RL()$, with weights 0 and 1. The solutions s_1 and s_2 and their corresponding weights are added to a tree-like data structure that stores the segments of the weight space (line 4). Every iteration, the largest segment, according to a norm (Euclidean distance, hypervolume, …), that was not explored yet is determined (line 6) and the new weight is calculated to be in the middle of that segment (line 7). Subsequently, the optimization algorithm is run with that weight and 2 new segments are added to the tree. i.e. the segment connecting the left solution and the middle solution and another segment from the middle to the right solution. (lines 9 and 10). Eventually, the *Archive* stores the set of multi-objective solution acquired through the run.

```
1:  $s_1 \leftarrow RL(0)$
2:  $s_2 \leftarrow RL(1)$
3:  Add $s_1$, $s_2$ to *Archive*
4:  tree $\leftarrow$ new Tree(new Segment$(0, s_1, 1, s_2)$)
5:  **while** stopping criteria not met **do**
6:      Segment lg $\leftarrow tree.getLargestGap()$
7:      w $\leftarrow \frac{lg.weight_{s_1} + lg.weight_{s_2}}{2}$
8:      $s' \leftarrow RL(w)$
9:      tree.add(new Segment(lg.weight$_{s_1}$, lg.$_{s_1}$, w, $s'$))
10:     tree.add(new Segment(w, $s'$, lg.weight$_{s_2}$, lg.$_{s_2}$))
11:     Add $s'$ to *Archive*
12: **end while**
13: **return** *Archive*
```

Fig. 3: An Improved AWA

We also see opportunities for trying out an alternative norm. The hypervolume measure in AN-TPLS-HV is an interesting indicator that calculates the surface that two solutions occupy in the two-dimensional objective space. Although the hypervolume measure in Eq. 5 works well for calculating the volume of two solutions in the objective space, it is not the common hypervolume formula. An alternative hypervolume calculation takes into account a particular reference point r^* to measure the volume of a given set of solutions, for any number of elements. It is clear that some solutions in the set will have less or more contribution to the final performance because the rectangles (defined by the reference point to each of the solutions) will have significant overlaps. Therefore, it might be interesting to consider the degree of overlap between solutions directly and to minimize this overlap. We call this norm the *overlapping* hypervolume (OHV) measure (Fig. 4). The measure calculates the ratio of the overlap and the unique hypervolume of the two solutions (Eq. 8 and 12).The idea is to order the solutions using this measure in order to intensify the search to segments with the smallest overlap first.

Note that our test environment only considers two objectives, but we believe this algorithm could be generalized to more objectives quite easily. For instance, in Algorithm 3 at line 6, we look for the 'largest gap' in the current Pareto front of already obtained solutions. In the bi-objective environment we used the Euclidean distance, while with three objectives we could use e.g. adaptive triangulation.

$$overlap = |(r_1^* - min(f_1(s_1), f_1(s_2)) \cdot (r_2^* - min(f_2(s_1), f_2(s_2))| \quad (8)$$

$$surf_{s_1} = |(f_1(s_1) - r_1^*) \cdot (f_2(s_1) - r_2^*)| \quad (9)$$

$$surf_{s_2} = |(f_1(s_2) - r_1^*) \cdot (f_2(s_2) - r_2^*)| \quad (10)$$

$$total = surf_{s_1} + surf_{s_2} - overlap \quad (11)$$

$$OHV(s_1, s_2) = \frac{overlap}{total} \quad (12)$$

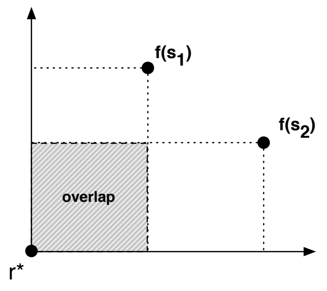


Fig. 4: The overlapping hypervolume (OHV) measures the percentage of overlap between two solutions s_1 and s_2 in the objective space.

A. *Results II*

Next, we analyse and compare the former adaptive weight selection mechanisms of Section II-B and IV on both simulated and real-life scenarios. We focus on the speed at which these methods approximate the Pareto front as measured by several quality indicators, such as the (inverted) generational distance, the generalized spread indicator and the hypervolume distance. The generational distance (GDistance) measures how far the elements in the set of non-dominated vectors, generated by the optimization algorithm, are from those in the Pareto optimal set. The inverted generational distance (IGDistance) calculates how far the elements in the Pareto optimal set are from those in the set of non-dominated vectors found. The generalized spread indicator (GSpread) is a measure of diversity that calculates the extent of spread achieved among the obtained solutions. The hypervolume indicator calculates the surface area between a point of reference and the approximate Pareto front. More information on the quality indicators can be found in [12].

1) Simulated data: First we investigate the *anytime* property of each of the methods. The results in Figures 6 (a) extend the results of Fig. 2 by including our novel methods. We note that the combination with the overlapping hypervolume norm (*AN-TPRL-OHV*) does not manage to improve a lot after the initial iterations and reaches similar performance to *TPRL* on scenario 1. The best combination of speed of learning and final performance was obtained by our *RA-TPRL-HV* and *RA-TPRL-OHV*, while the distance norm (*RA-TPRL-DIST*) learns significantly slower. For scenario 7 in Fig. 6 (b), similar conclusions can be formed except that *AN-TPRL-OHV* is now amongst the best performing algorithms, leaving *TPRL* far behind. In general, the naive methods and *RA-TPRL-DIST* are the slowest learners of the pack, while the adaptive methods reach similar performance in the end. In initial stages, the *AN-TPRL* and *AN-TPRL-HV* methods learn a bit faster, but this difference is negligible.

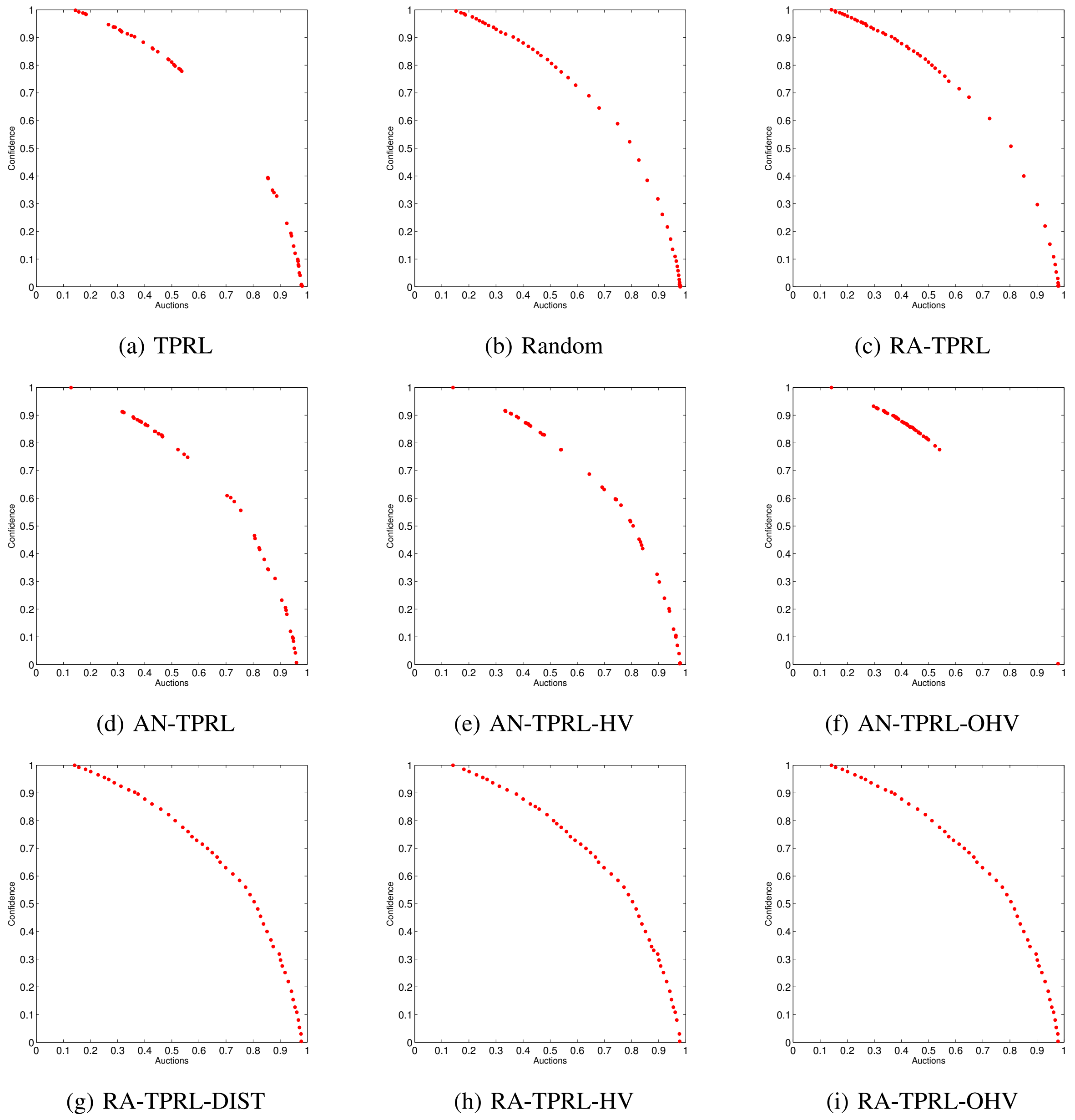
In Fig. 5 the Pareto front after 50 iterations is depicted for each of the methods. We see that TPRL, although providing promising results in previous work on CamSim [6], clearly lacks at exploring every part of the Pareto front as it leaves particular areas uncovered. On the third row of Fig.5, the methods that combine *RA-TPRL* with the distance, hypervolume and overlapping hypervolume norms improve these results in terms of spread in the objective space, while minimizing the gaps by taking into account the limited number of iterations. Other results using quality indicators that quantify these gaps in the Pareto front are presented in Table I. We note that our *RA-TPRL-HV* method obtained the best results in terms of spread in 10 scenarios. The best method in terms of generational (inverted) distance differ a lot for each scenario but in general the differences between the methods are minimal.

2) Real-life data: We continue our evaluation using video feed data from a real smart camera network. This is referred to as the 'real-life' scenario. Figure 7 shows snapshots from each camera at five different points in time. Each camera captured 1780 frames, looped four times to create a total of 7120 frames, each with a resolution of 640×480.

In Fig. 8 the running hypervolume is depicted on the real-life data. Similar performance is obtained for the naive methods but in this example, we clearly see that the *AN-TPRL* methods are stuck at a specific performance level and can not find contributions to their Pareto front. Our *RA-TPRL-HV* and *RA-TPRL-OHV* algorithms are able to improve these latter methods after 5 iterations. At the bottom of Table I, we depict the results of the quality indicators of the different adaptive weight methods. We note that the results are similar for the simulated data, meaning that best performance is obtained by the *RA-TPRL-DIST*, *RA-TPRL-HV* and *RA-TPRL-OHV* methods.

V. Conclusions

In this paper, we draw conclusions on both the algorithmic level and the application-based level with smart camera networks in mind. Firstly, on a conceptual level, we have elaborated on one of the major difficulties of scalarization functions, i.e. determining the weights that specify the emphasis on each of the objectives so as to approximate the Pareto front. Previous work taught us that a uniform distribution in the weight space does not guarantee an even spread in the objective space. Recently, in [2], a series of algorithms were proposed that alter the weight with a *dichotomic* scheme to specify the direction of search. In the experimental section, we have seen that this method does not obtain well-spread results. Therefore, we proposed to combine the regular anytime algorithm which divides the weight

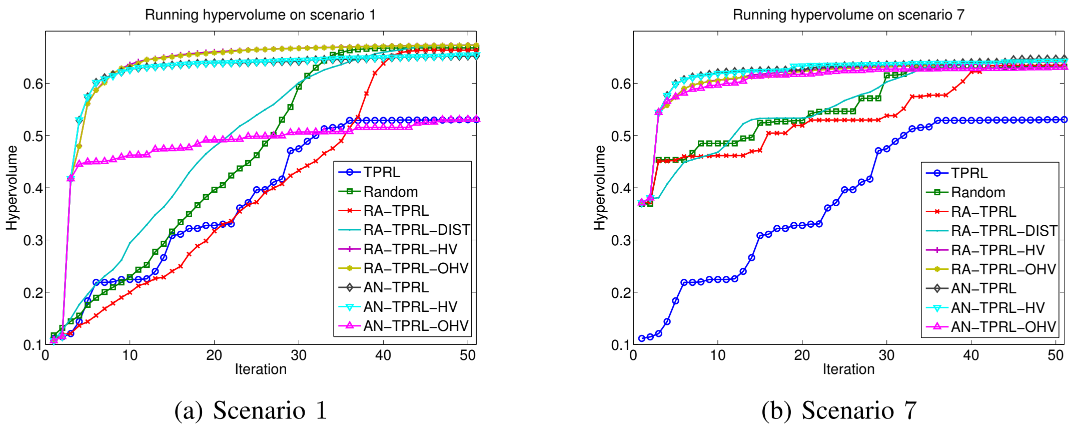


(a) TPRL (b) Random (c) RA-TPRL

(d) AN-TPRL (e) AN-TPRL-HV (f) AN-TPRL-OHV

(g) RA-TPRL-DIST (h) RA-TPRL-HV (i) RA-TPRL-OHV

Fig. 5: The Pareto fronts obtained after 50 iterations for each of the adaptive weight algorithms on Scenario 1.



(a) Scenario 1 (b) Scenario 7

Fig. 6: The hypervolume over time for each of the adaptive weight algorithms on scenarios 1 and 7.

	TPRL	Random	RA-TPRL	RA-TPRL-DIST	RA-TPRL-HV	RA-TPRL-OHV	AN-TPRL	AN-TPRL-HV	AN-TPRL-OHV
Scenario 1									
$GSpread$	0.97501	0.91959	0.95257	0.80777	0.81487	**0.8076**	1.00112	1.00059	1.04571
HV	0.60384	0.66785	0.66347	0.6732	**0.67332**	0.67311	0.65238	0.65446	0.53106
$GDistances$	2.99143	2.98875	2.96057	2.92172	**2.91005**	2.92752	2.92267	2.95842	2.94144
$IGDistance$	14.20181	14.07198	14.07761	14.07234	**14.07109**	14.07234	14.07971	14.07675	14.14254
Scenario 2									
$GSpread$	0.90053	0.92837	0.92312	0.93271	**0.87589**	0.92952	0.95014	1.00981	0.91466
HV	0.66331	0.66783	0.67734	0.67008	0.67916	0.67087	0.6792	**0.68073**	0.65541
$GDistances$	2.83976	2.82884	2.74931	2.9209	2.87788	2.91765	3.02818	3.1312	**2.74639**
$IGDistance$	12.95593	12.95081	**12.90369**	12.95266	12.97541	12.95451	12.96075	12.98492	12.93109
Scenario 3									
$GSpread$	0.89829	0.92254	0.88001	0.90019	**0.85584**	0.88654	0.95599	1.05615	0.97401
HV	0.55002	0.54008	0.54899	0.54764	0.5553	0.54865	**0.58074**	0.53527	0.52907
$GDistances$	3.03947	3.04809	**2.98084**	2.99027	2.9913	2.99857	3.10163	3.20551	3.02713
$IGDistance$	**13.99183**	14.1615	14.03292	14.2017	14.10126	14.18741	14.08196	14.17774	14.05668
Scenario 4									
$GSpread$	0.90403	0.92403	0.92751	0.88937	**0.87291**	0.87961	0.9389	1.02157	0.96905
HV	0.46702	0.45717	0.46915	0.47347	0.47403	**0.47481**	0.46587	0.45524	0.44411
$GDistances$	3.11392	3.11126	**3.06278**	3.08443	3.08056	3.07423	3.15675	3.22976	3.15611
$IGDistance$	14.81707	**14.7147**	14.8793	14.84748	14.89994	14.81748	14.85078	14.85685	14.88275
Scenario 5									
$GSpread$	0.89728	0.93505	0.93387	0.8638	**0.83163**	0.85699	0.9592	1.02245	0.98591
HV	0.7239	0.70845	0.73191	0.73584	**0.73768**	0.73611	0.72231	0.71118	0.61485
$GDistances$	2.8878	2.8815	2.84129	**2.8124**	2.81807	2.83094	3.00157	3.06383	2.94756
$IGDistance$	13.20978	13.28262	**13.16021**	13.16986	**13.16021**	13.17839	13.21716	13.26455	13.31449
Scenario 6									
$GSpread$	0.89975	0.90772	0.90297	0.92098	**0.8757**	0.92804	0.95141	1.04161	0.91413
HV	**0.35623**	0.31604	0.33999	0.32833	0.34209	0.32689	0.34858	0.33307	0.32476
$GDistances$	3.23164	3.4249	**3.15232**	3.27144	3.36656	3.40051	3.49461	3.61006	3.37513
$IGDistance$	14.84423	**14.60966**	14.63511	14.63258	14.63511	14.62701	14.63511	14.63511	14.62822
Scenario 7									
$GSpread$	0.88941	0.91815	0.91261	0.87262	**0.86677**	0.87676	0.92367	1.02515	0.92114
HV	0.6351	0.63149	0.63461	0.64238	0.64151	0.64271	**0.64698**	0.64234	0.63078
$GDistances$	2.91725	2.91555	**2.85605**	2.89079	2.88499	2.8822	2.98927	3.08699	2.94745
$IGDistance$	13.56464	13.60138	13.62932	13.60277	13.60277	13.62932	13.52138	13.56219	**13.50574**
Scenario 8									
$GSpread$	0.89045	0.89952	0.92083	0.87468	**0.8426**	0.85821	0.94294	1.00346	0.90937
HV	0.67929	0.67015	**0.70277**	0.69972	0.69903	0.70077	0.68043	0.6726	0.64769
$GDistances$	2.87852	2.86679	**2.81096**	2.86095	2.87561	2.858	2.99304	3.0593	2.90116
$IGDistance$	13.32639	**13.12466**	13.25595	13.29731	13.2644	13.27501	13.23966	13.25307	13.27036
Scenario 9									
$GSpread$	0.98498	0.97623	0.9948	0.85214	**0.84111**	0.8635	1.01375	1.00589	1.01342
HV	0.77092	0.72027	0.76646	0.77816	**0.77854**	0.77773	0.75198	0.74881	0.71142
$GDistances$	2.86232	2.87269	2.8102	2.76207	**2.755**	2.77794	2.84333	2.90014	2.92109
$IGDistance$	13.12838	13.17884	13.11477	13.11224	**13.11206**	13.11371	13.15331	13.17327	13.18997
Scenario 10									
$GSpread$	0.94538	0.97646	0.95739	0.94302	**0.91827**	0.93815	1.00365	1.06029	1.00073
HV	0.62974	0.63441	0.63944	0.63959	0.63967	0.63957	0.63652	**0.64938**	0.63766
$GDistances$	2.80424	2.79834	**2.7291**	2.73846	2.75131	2.73696	2.95668	3.00213	2.82691
$IGDistance$	12.79585	**12.72557**	12.74433	12.74441	12.74441	12.74441	12.74441	12.74229	12.73365
Scenario 11									
$GSpread$	0.94689	0.96702	0.95472	0.92537	**0.90926**	0.91007	0.98978	1.05081	0.99617
HV	0.56738	0.56372	0.56865	0.57189	0.57219	0.57197	0.5778	**0.58042**	0.56247
$GDistances$	2.94136	2.94263	**2.88296**	2.90703	2.91029	2.90481	2.99781	3.05835	3.00429
$IGDistance$	13.93518	**13.93155**	13.93476	13.93734	13.93801	13.93734	13.99275	13.95814	13.95763
Real-life scenario									
$GSpread$	0.8974	0.90369	0.89542	0.86498	0.85507	**0.85358**	0.94972	1.016	0.95567
HV	0.53053	0.48302	0.52724	0.53287	**0.53401**	0.5332	0.49554	0.44196	0.46728
$GDistances$	3.10149	3.12013	3.09672	**3.05587**	3.05963	3.06153	3.09171	3.17102	3.11861
$IGDistance$	14.9041	15.24093	14.98453	**14.77806**	**14.77806**	**14.77806**	14.96315	15.17256	15.12926

TABLE I: Quality indicators on 12 different scenarios in CamSim

space into levels with increasing degrees of intensification with different norms. These norms determine the order in which the elements on the branches will be evaluated. All the methods were extensively compared on 11 simulated smart camera network scenarios and one real-life setting in the multi-objective multi-agent CamSim framework. We were able to improve previous work in CamSim and our results showed us that our methods can (i) explore the Pareto front faster, (ii) obtain improved solutions in terms of hypervolume and (iii) better spread in the objective space.

Originally, in smart camera networks, there was no global feedback signal that comprises aspects of the Pareto front being explored [6]. In this paper, we have seen that the agents can significantly benefit from a centralized component that steers the search process by analysing the policies obtained so far. Obtaining a wider spread on the system wide Pareto front, as enabled by this component, gives greater expressiveness with which a system could be deployed in scenarios that resemble the ones analysed in simulation. In future work, we will analyse whether it is beneficial for each agent (camera) to use the same weight parameter. Perhaps specifying different weights amongst agents could let groups of agents focus on different objectives. Hence, we could let the agents automatically learn different responsibilities and eventually a division of labour could emerge where agents become experts at particular tasks.

VI. Acknowledgements

Kristof Van Moffaert is supported by the IWT-SBO project PERPETUAL (grant nr. 110041). Tim Brys is funded by a Ph.D grant of the Research Foundation Flanders (FWO). Lukas Esterle, Arjun Chandra and Peter R. Lewis are supported by the EPiCS project under the European Union Seventh Framework Programme (grant nr. 257906). We would also like to thank Jérémie Dubois-Lacoste for his interesting comments and insights on the subject.

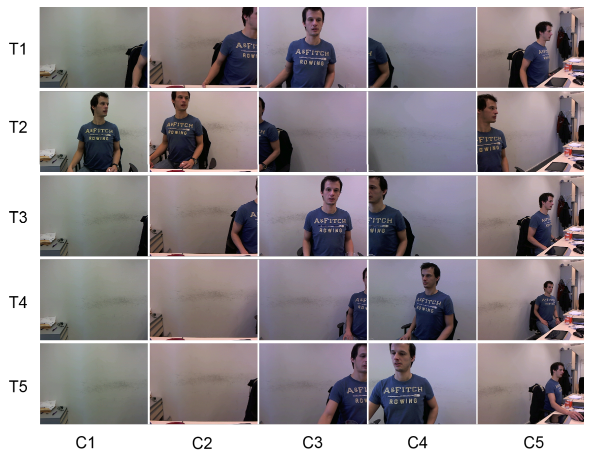


Fig. 7: Shots from five participating cameras tracking a single person.

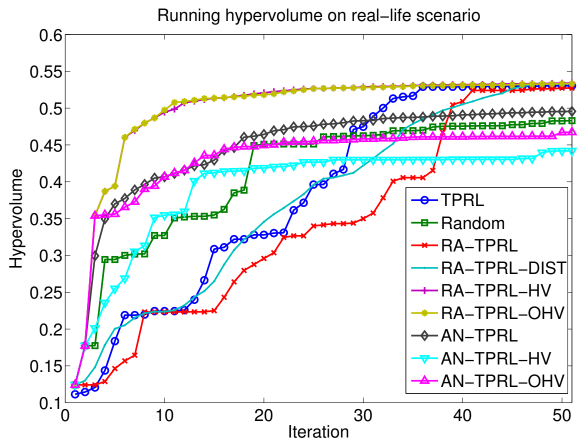


Fig. 8: The hypervolume over time for each of the adaptive weight algorithms on the real-life data.

References

[1] I. Das and J. E. Dennis. A closer look at drawbacks of minimizing weighted sums of objectives for pareto set generation in multicriteria optimization problems. *Structural and Multidisciplinary Optimization*, 14:63–69, 1997.
[2] J. Dubois-Lacoste, M. López-Ibáñez, and T. Stützle. Improving the anytime behavior of two-phase local search. *Annals of Mathematics and Artificial Intelligence*, 61(2):125–154, 2011.
[3] L. Esterle, P. Lewis, M. Bogdanski, B. Rinner, and X. Yao. A Socio-Economic Approach to Online Vision Graph Generation and Handover in Distributed Smart Camera Networks. In *Proceedings of the Fifth ACM/IEEE International Conference on Distributed Smart Cameras*, pages 1–6. IEEE Press, 2011.
[4] L. Esterle, P. R. Lewis, H. Caine, X. Yao, and B. Rinner. CamSim: A distributed smart camera network simulator. In *Proceedings of the 7th IEEE Conference on Self-Adaptive and Self-Organizing Systems Workshops (SASOW)*. IEEE Press, 2013. In press.
[5] L. Esterle, P. R. Lewis, X. Yao, and B. Rinner. Socio-economic vision graph generation and handover in distributed smart camera networks. *ACM Transactions on Sensor Networks*, 10(2), 2014.
[6] P. R. Lewis, L. Esterle, A. Chandra, B. Rinner, and X. Yao. Learning to be different: Heterogeneity and efficiency in distributed smart camera networks. In *Proceedings of the 7th IEEE Conference on Self-Adaptive and Self-Organizing Systems (SASO)*, pages 209–218. IEEE Press, 2013.
[7] D. J. Lizotte, M. Bowling, and S. A. Murphy. Linear fitted-q iteration with multiple reward functions. *Journal of Machine Learning Research*, 13:3253–3295, 2012.
[8] T. Lust and J. Teghem. Two-phase pareto local search for the biobjective traveling salesman problem. *Journal of Heuristics*, 16(3):475–510, 2010.
[9] P. Vamplew, R. Dazeley, A. Berry, R. Issabekov, and E. Dekker. Empirical evaluation methods for multiobjective reinforcement learning algorithms. *Machine Learning*, 84(1-2):51–80, 2010.
[10] K. Van Moffaert, M. M. Drugan, and A. Nowé. Hypervolume-based multi-objective reinforcement learning. *Lecture Notes in Computer Science, Evolutionary Multi-Criterion Optimization (EMO 2013)*, 2013.
[11] K. Van Moffaert, M. M. Drugan, and A. Nowé. Scalarized Multi-Objective Reinforcement Learning: Novel Design Techniques. In *2013 IEEE International Symposium on Approximate Dynamic Programming and Reinforcement Learning*. IEEE, 2013.
[12] D. A. V. Veldhuizen and G. B. Lamont. Multiobjective evolutionary algorithm research: A history and analysis, 1998.