ISSN 2007-9737

Novel Dynamic Decomposition-Based Multi-Objective Evolutionary Algorithm Using Reinforcement Learning Adaptive Operator Selection (DMOEA/D-SL)

José Alfredo Brambila-Hernández[1], Miguel Ángel García-Morales[1], Héctor Joaquín Fraire-Huacuja[1,*], Laura Cruz-Reyes[1], Claudia G. Gómez-Santillán[1], Nelson Rangel-Valdez[1], Héctor José Puga-Soberanes[2], Fausto Balderas[1]

[1] Tecnológico Nacional de México,
Instituto Tecnológico de Ciudad Madero,
Mexico

[2] Tecnológico Nacional de México,
Instituto Tecnológico de León,
Mexico

alfredo.brambila@outlook.com, {soporteclusterlanti, hector.fraire2014}@gmail.com, lauracruzreyes@itcm.edu.mx, {claudia.gs,nelson.rv, fausto.bj}@cdmadero.tecnm.mx, pugahector@yahoo.com

Abstract. Within the multi-objective (static) optimization field, various works related to the adaptive selection of genetic operators can be found. These include multi-armed bandit-based methods and probability-based methods. For dynamic multi-objective optimization, finding this type of work is very difficult. The main characteristic of dynamic multi-objective optimization is that its problems do not remain static over time; on the contrary, its objective functions and constraints change over time. Adaptive operator selection is responsible for selecting the best variation operator at a given time within a multi-objective evolutionary algorithm process. This work proposes incorporating a new adaptive operator selection method into a Dynamic Multi-objective Evolutionary Algorithm Based on Decomposition algorithm, which we call DMOEA/D-SL. This new adaptive operator selection method is based on a reinforcement learning algorithm called State-Action-Reward-State-Action Lambda or SARSA (λ). SARSA Lambda trains an Agent in an environment to make sequential decisions and learn to maximize an accumulated reward over time; in this case, select the best operator at a given moment. Eight dynamic multi-objective benchmark problems have been used to evaluate algorithm performance as test instances. Each problem produces five Pareto fronts. Three metrics were used: Inverted Generational Distance, Generalized Spread, and Hypervolume. The non-parametric statistical test of Wilcoxon was applied with a statistical significance level of 5% to validate the results.

Keywords. Adaptive, operator, selection, dynamic, multi-objective, optimization.

1 Introduction

A definition proposed by Azzouz [1] for a dynamic problem is: “dynamic multi-objective optimization problem (DMOP) is the problem of finding a vector of decision variables which satisfies a set of constraints and optimizes a vector of functions whose scalar values They represent objectives that change over time.” A DMOP can be defined as follows in Equation 1:

$$\begin{gathered}\text{Min or Max: } F(x,t) = \\ \{f_1(x,t), f_2(x,t), \dots, f_m(x,t)\} \\ x \in X^n \\ \text{s.t.} \\ g(x,t) > 0 \\ h(x,t) = 0 \\ x_i^l \le x \le x_i^u, i = 1,2,\dots,n,\end{gathered} \tag{1}$$

where x is the vector of decision variables, n is its size, X^n represents the solution space, f is the set of objectives to be maximized or minimized

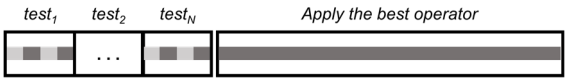

Fig. 1. Test-and-apply structure [7]

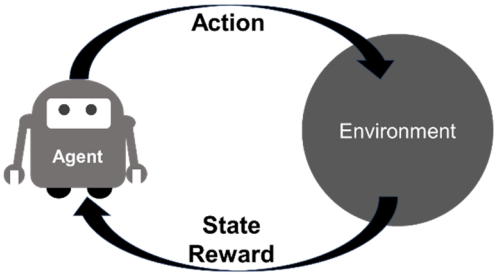

Fig. 2. Interaction Agent-Environment

concerning time, g and h represent the set of inequalities and equalities, respectively, t represents the time or dynamic nature of the problem, and m is the number of objectives.

Algorithm 1 SARSA (λ)

Parameters: α: Learning rate, ε: Policy parameter ε-greedy, γ: Discount factor, λ: Controls the amount of forward and backward learning

Variables: S_t: Current state, S_{t+1}: Next state, A_t: Current action, Reward: Reward calculated for the next state and the current action, QTable: Table of Q values representing the state-action value function

```
Initialize QTable()
foreach episode do
    S_t <- InitialState
    foreach step of episode do
        A_t ← chooseAction(S_t, ε)
        S_{t+1} ← takeAction(A_t)
        A_{t+1} ← chooseAction(S_{t+1}, ε)
        updateTraces(S_t, A_t) //e(S_t, A_t) < −λ × γ * e(S_t, A_t) + 1
        Reward ← getReward(S_{t+1}, A_t)
        Delta ← Reward + γQ(S_{t+1}, A_{t+1}) − Q(S_t, A_t)
        updateQTable(S_t, A_t, Reward, S_{t+1}, δ)
        S_t ← S_{t+1}
        A_t ← A_{t+1}
    end for
end for
```

Ke Li [2] mentions that the recurring problems within single-objective and multi-objective optimization are determining a specific configuration for the control parameters and the selection of the correct genetic operators for each type of problem; this leads us to have different scenarios for solving problems, this becomes more complicated when the person is not an EA expert.

Search operators are of great importance in metaheuristics. Some operators are better for solving a specific problem. On the other hand, the selection and order of use of these operators can affect the performance of an algorithm [3].

In this case, we focus on selecting the correct genetic operator; adaptive operator selection, also called AOS, is used. The AOS is responsible for automatically determining which variation operator to use at a given time within a MOEA process.

Within the state of the art for AOS, the most recent works found correspond to methods based on multi-arm bandits applied to static multi objective optimization algorithms based on decomposition.

In this work, a new adaptive operator selection mechanism is proposed, which is based on the SARSA (λ) reinforcement learning technique. This new AOS mechanism has been incorporated into the DMOEA/D algorithm and compared against the state-of-the-art algorithm, which was also applied to the DMOEA/D algorithm.

2 Background and Related Work

The AOS comprises two main tasks [4]: credit assignment and operator selection. In the first task, the reward or weight of each operator is determined by their performance. The second task selects the best available operator.

Credits can be assigned in different ways depending on our method or algorithm. However, in general, it can be done based on the improvement of the children's fitness concerning the parents, and it can also be done based on a ranking of the operators.

Depending on how the operator selection task works, the AOS is divided into two groups: one encompasses all methods based on a probabilistic approach, and the other encompasses a multi-arm bandit approach (MAB).

Algorithm 2. DMOEA/D-SL

Input: DMOP: Dynamic multi-objective problem,
Pop: Population List,
nPop: Population size,
fileSize: MOEA/D file size,
N: The number of subproblems considered in MOEA/D,
A: Uniform distribution of N weight vectors: $\lambda^1, \ldots, \lambda^N$,
T: The number of weight vectors in the neighborhood of each weight vector,
numStates: Number of states for algorithm SARSA (λ),
numActions: Number of actions / genetic operators,
α: Learning rate,
ε: Policy parameter ε-greedy,
γ: Discount factor,
λ: Controls the amount of forward and backward learning,
nT: Change frequency,
tauT: Severity of change

Output: Front$_f$: Pareto estimate front for each problem change

$EP = \emptyset$

Compute the Euclidean distances between any two weight vectors and then compute the closest weight vectors *T* to each weight vector.

Agent ← SarsaLambdaAlgorithm(numStates, numActions, α, γ, ε, λ)

for $i \leftarrow 1$ **to** N **do**
  $B(i) = \{i_1, \ldots, i_T\}$
  /*where $\lambda^{i_1}, \ldots, \lambda^{i_T}$ are the T closest weight vectors to λ^i */
end for

Generate an initial pop $x^1, \ldots, x^N$ randomly for the specific problem.
$FV^i = F(x^i)$
Initialize $z = (z_1, \ldots, z_m)^N$ for the specific problem
time=0
detectors=GetDetectorList(Pop)
it=0
f=1

while stopping criteria not met **do**
  $S_t \leftarrow$ initialState
  $A_t \leftarrow$ Agent.chooseAction(S_t)
  QVal$_t \leftarrow$ Agent.getQValue(S_t, A_t)

  for $i \leftarrow 1$ **to** $nPop$ **do**
    $y' \leftarrow$ takeAction(A_t)
    for $j \leftarrow 1$ **to** m **do**
      if $z_j < f_j(y')$ **then**
        $z_j \leftarrow f_j(y')$
      end if
    end for

    foreach index $j \in B(i)$ **do**

      if $g^{te}(y'|\lambda^j, z) \leq g^{te}(x^j|\lambda^j, z)$ **then**
        $x^j \leftarrow y'$
        $FV^j \leftarrow F(y')$
        $S_t \leftarrow S_0$
        $S_{t+1} \leftarrow$ nextState()
        $A_{t+1} \leftarrow$ Agent.chooseAction(S_{t+1})
        $QVal_{t+1} \leftarrow$ Agent.getQValue(S_{t+1}, A_{t+1})
        Agent.updateEligibilityTraces(S_t, A_t)
        reward ← Agent.getReward(S_t, A_t)
        $\delta \leftarrow$ reward + $\gamma Q(S_{t+1}, A_{t+1}) - Q(S_t, A_t)$
        Agent.updateQTable(reward, $QVal_t, QVal_{t+1}, \delta$)
        $S_t \leftarrow S_{t+1}$
        $A_t \leftarrow A_{t+1}$
        $QVal_t \leftarrow QVal_{t+1}$

      else if
        $S_t \leftarrow S_1$
        $S_{t+1} \leftarrow$ nextState()
        $A_{t+1} \leftarrow$ Agent.chooseAction(S_{t+1})
        $QVal_{t+1} \leftarrow$ Agent.getQValue(S_{t+1}, A_{t+1})
        Agent.updateEligibilityTraces(S_t, A_t)
        reward ← Agent.getReward(S_t, A_t)
        $\delta \leftarrow$ reward + $\gamma Q(S_{t+1}, A_{t+1}) - Q(S_t, A_t)$
        Agent.updateQTable(reward, $QVal_t, QVal_{t+1}, \delta$)
        $S_t \leftarrow S_{t+1}$
        $A_t \leftarrow A_{t+1}$
        $QVal_t \leftarrow QVal_{t+1}$
      end if
    end foreach
  end for

  Remove from EP all solutions dominated by $F(y')$
  Insert $F(y')$ in EP if there are no solutions in EP that dominate $F(y')$
  Time = $\frac{1}{nT} \times \left\lfloor \frac{1 \times it}{\text{tauT}} \right\rfloor$

  if problemChangeDetection(decetors, time) **then**
    insert EP into Front$_f$
    Initialize $z = (z_1, \ldots, z_m)^N$ for the specific problem
    Evualuate Pop
    Execute changeResponseMechanism()
    $FV^i = F(x^i)$
    Remove from EP all solutions dominated by $F(y')$
    Insert $F(y')$ in EP if there are no solutions in EP that dominate $F(y')$
    $f = f + 1$
  end if

  Agent.reinitializeQTable()
  $it = it + 1$
end while
return Front$_f$

In the multi-arm bandit approach, AOS uses the "multi-arm bandit (MAB) problem paradigm" [2], which considers each operator as an arm of a slot machine, each with an unknown reward probability. These methods seek to maximize the reward accumulated during the process and model these rewards to select the best operator (arm) at each moment.

In our previous work [5], we have applied "the Fitness-Rate-Rank-based Multi-Armed Bandit (FRRMAB)" [2], "Adaptive Operator Selection Based on Dynamic Thompson Sampling (DYTS)" [6], and "Adaptive operator selection with test-and-apply structure for decomposition-based multi-objective optimization (TAOS)" [7] methods to a dynamic version of the MOEA/D algorithm to observe its behavior.

Another of the most recent works in the area is "A novel bicriteria assisted adaptive operator selection (B-AOS) strategy for decomposition-based multi-objective evolutionary algorithms (MOEA/Ds)" proposed in 2021 by Wu Lin [8]. This approach employs two groups of operators; each group includes two genetic operators with different search patterns.

Table 1. Benchmark problems

Problem	Objectives	Change frequency	Change severity
dMOP1 [15]	2	100	10
dMOP2 [15]	2	100	10
FDA1 [16]	2	100	10
FDA3 [16]	2	100	10
DF4 [17]	2	100	10
DF6 [17]	2	100	10
DF10 [17]	3	100	10
DF12 [17]	3	100	10

Table 2. Algorithm parameters

Variables/ Parameters	DMOEA/D-TAOS	DMOEA/D-SL
maxIt	100	100
nPop	100	100
fileSize	100	100
Zeta	0.2	0.2
K	1	--
alpha	--	0.04
gamma	--	0.08
lambda	--	0.07
epsilon	--	0.1

In addition, it uses two criteria, which emphasize convergence and diversity, to help select the appropriate operator. In the probabilistic approach, a probability is attached to each operator, and its selection process is similar to that of a roulette wheel; the operators with the highest probabilities will be those who will have a larger area on the roulette wheel and will be the ones who will have a greater chance of being selected.

The most popular methods within this group are Probability Matching [9] and Adaptive Pursuit [10]. Another work is "Adaptive crossover operator based multi-objective binary genetic algorithm for feature selection in classification" which uses a probability-based AOS within a multi-objective genetic algorithm to solve feature selection problems [11].

The strategies mentioned above are applied to static multi-objective algorithms in the state-of-the-art. In this work, it has been decided to use the most recent strategy, "Adaptive operator selection with test-and-apply structure for decomposition-based multi-objective optimization (TAOS)" [7] and apply it to a dynamic multi-objective algorithm to compare the state-of-the-art strategy with the strategy proposed in this work.

2.1 Adaptive Operator Selection with Test-and-Apply Structure for Decomposition-based Multi-Objective Optimization (TAOS)

In this approach proposed by Lisha Dong in 2022 [6], the whole evolutive process is structured into several continuous sections, each designed to execute testing and application phases.

2.1.1 Test Phase

In the testing phase, each operator is tested in the same environment. The testing phase is divided into N parts, as shown in Figure 1: test_1, …, and test_N. All operators will be evaluated once in order. Therefore, the total number of function evaluations for the testing phase is defined as follows, as shown in Equation 2:

$$FE_{\text{test}} = K \times N, \tag{2}$$

where K denotes the number of operators, and the population size is defined by N. When an operator is applied in the search process, its impact needs to be measured; a successful update indicator (SUI) is introduced to assign credits to the operator. SUI is defined as follows as sampled in Equation 3:

$$SUI = \begin{cases} 1, \text{ if the generated solution is better than at least one solution in the population,} \\ 0, \text{ and otherwise.} \end{cases} \tag{3}$$

A child solution updates the first solution, and the SUI value changes from 0 to 1; if the child continues updating the solutions, the SUI will remain unchanged.

Table 3. Hypervolume (median and IQR values)

Problem	Front	DMOEA/D-TAOS	DMOEA/D-SL
FDA1	1	6.32e-01 6.24e-03	6.31e-01 8.39e-03 ▲
	2	6.33e-01 5.76e-03	6.32e-01 5.35e-03 ▲
	3	6.29e-01 7.83e-03	6.30e-01 8.70e-03 ==
	4	6.33e-01 5.44e-03	6.33e-01 8.34e-03 ▼
	5	6.31e-01 7.67e-03	6.35e-01 5.83e-03 ▼
FDA3	1	6.56e-01 5.97e-03	6.55e-01 4.65e-03 ==
	2	6.86e-01 7.86e-03	6.84e-01 4.33e-03 ==
	3	6.63e-01 3.52e-03	6.61e-01 4.50e-03 ▲
	4	6.51e-01 3.86e-03	6.49e-01 4.44e-03 ▲
	5	6.50e-01 6.20e-03	6.47e-01 8.16e-03 ▲
DMOP1	1	4.28e-01 3.71e-03	4.27e-01 4.36e-03 ▲
	2	4.07e-01 3.13e-03	4.07e-01 2.48e-03 ==
	3	3.88e-01 1.98e-03	3.88e-01 2.34e-03 ▲
	4	3.72e-01 2.27e-03	3.71e-01 2.42e-03 ==
	5	3.58e-01 2.20e-03	3.58e-01 1.96e-03 ==
DMOP2	1	4.28e-01 2.30e-03	4.27e-01 1.93e-03 ▲
	2	4.05e-01 2.61e-03	4.05e-01 3.10e-03 ▲
	3	3.86e-01 2.55e-03	3.86e-01 3.10e-03 ▼
	4	3.69e-01 2.25e-03	3.68e-01 2.53e-03 ▲
	5	3.55e-01 1.49e-03	3.55e-01 3.76e-03 ▼
DF4	1	6.83e-01 4.06e-03	6.80e-01 8.08e-03 ▲
	2	7.34e-01 4.68e-03	7.32e-01 3.60e-03 ▲
	3	7.72e-01 3.46e-03	7.72e-01 3.36e-03 ==
	4	8.04e-01 2.04e-03	8.03e-01 2.91e-03 ▲
	5	8.26e-01 3.16e-03	8.26e-01 3.39e-03 ▲
DF6	1	3.71e-02 6.53e-04	3.70e-02 9.11e-04 ▲
	2	0.00e+00 0.00e+00	0.00e+00 0.00e+00 ==
	3	3.51e-02 1.06e-01	3.81e-02 2.02e-01 ▼
	4	1.06e-01 1.75e-01	1.08e-01 2.97e-01 ▼
	5	2.12e-01 2.13e-01	2.15e-01 3.40e-01 ▼
DF10	1	8.36e-01 6.66e-03	8.36e-01 5.48e-03 ▼
	2	8.30e-01 7.12e-03	8.31e-01 5.09e-03 ▼
	3	8.11e-01 6.06e-03	8.12e-01 5.20e-03 ▼
	4	7.82e-01 5.58e-03	7.82e-01 5.14e-03 ▲
	5	7.45e-01 6.38e-03	7.45e-01 4.35e-03 ▼
DF12	1	3.04e-01 1.18e-02	3.06e-01 2.47e-02 ==
	2	6.53e-01 1.39e-02	6.47e-01 1.71e-02 ▲
	3	6.46e-01 1.16e-02	6.51e-01 1.17e-02 ==
	4	6.53e-01 1.63e-02	6.50e-01 1.90e-02 ▲
	5	6.53e-01 1.57e-02	6.46e-01 2.71e-02 ▲

2.1.1 Apply Phase

When the testing phase has finished, the successful updates count (SUC) is obtained for each operator; this is shown in Equation 4:

$$SUC_{op} = \sum_{i=1} SUI_{op}^{i}, \tag{4}$$

where SUI_{op}^{i} indicates that the operator op is tested in evaluating the ith function of the testing phase. To select the operator, we compare them with each other.

The operator with the SUC with the highest value will be the one selected to be applied in the application phase of that segment. The number of function evaluations for the application phase in each segment is defined in Equation 5:

Table 4. Generalized Spread (median and IQR values)

Problem	Front	DMOEA/D-TAOS	DMOEA/D-SL
FDA1	1	5.16e-01 1.91e-01	4.63e-01 1.90e-01 ▼
	2	5.58e-01 1.45e-01	4.38e-01 1.20e-01 ▼
	3	5.34e-01 1.67e-01	4.68e-01 1.49e-01 ▼
	4	5.27e-01 1.59e-01	4.82e-01 1.50e-01 ▼
	5	5.59e-01 1.40e-01	4.65e-01 1.29e-01 ▼
FDA3	1	6.65e-01 2.03e-01	6.96e-01 1.57e-01 ==
	2	6.22e-01 2.27e-01	5.86e-01 3.57e-01 ▼
	3	8.44e-01 1.47e-01	8.34e-01 1.41e-01 ▼
	4	6.48e-01 7.09e-02	6.37e-01 6.86e-02 ▼
	5	4.70e-01 4.49e-02	4.44e-01 4.73e-02 ▼
DMOP1	1	7.46e-01 5.48e-02	7.33e-01 1.47e-01 ==
	2	7.42e-01 7.13e-02	7.27e-01 6.02e-02 ▼
	3	7.62e-01 1.05e-01	7.45e-01 6.63e-02 ▼
	4	7.53e-01 7.05e-02	7.48e-01 7.32e-02 ▼
	5	7.40e-01 1.17e-01	7.37e-01 6.95e-02 ==
DMOP2	1	7.27e-01 7.21e-02	7.50e-01 9.14e-02 ▲
	2	6.86e-01 8.37e-02	6.73e-01 7.78e-02 ▼
	3	6.92e-01 9.39e-02	7.00e-01 7.33e-02 ==
	4	6.86e-01 5.62e-02	6.93e-01 1.12e-01 ==
	5	6.85e-01 6.99e-02	6.83e-01 6.99e-02 ==
DF4	1	8.38e-01 1.47e-01	8.77e-01 1.78e-01 ==
	2	8.39e-01 1.11e-01	8.56e-01 1.47e-01 ==
	3	1.00e+00 4.66e-02	9.83e-01 6.08e-02 ▼
	4	1.12e+00 3.52e-02	1.10e+00 5.37e-02 ▼
	5	1.16e+00 7.50e-02	1.15e+00 5.48e-02 ▼
DF6	1	7.02e-01 8.48e-02	6.96e-01 8.28e-02 ==
	2	9.41e-03 4.32e-03	9.20e-03 1.12e-02 ==
	3	1.05e-01 4.86e-02	1.10e-01 1.01e-01 ▲
	4	1.07e-01 3.28e-02	1.10e-01 7.07e-02 ▲
	5	1.29e-01 8.68e-03	1.33e-01 3.10e-02 ▲
DF10	1	7.32e-01 1.49e-01	7.61e-01 1.50e-01 ▲
	2	7.78e-01 1.78e-01	7.93e-01 1.11e-01 ==
	3	7.66e-01 1.69e-01	7.73e-01 1.15e-01 ▲
	4	7.54e-01 1.83e-01	7.50e-01 9.83e-02 ▼
	5	7.96e-01 2.02e-01	7.57e-01 1.38e-01 ==
DF12	1	5.14e-01 3.90e-02	4.76e-01 5.83e-02 ▼
	2	4.96e-01 4.41e-02	5.05e-01 3.89e-02 ▲
	3	5.14e-01 3.21e-02	5.07e-01 3.29e-02 ▼
	4	5.12e-01 5.60e-02	5.14e-01 3.97e-02 ▲
	5	5.05e-01 5.91e-02	4.91e-01 5.79e-02 ▼

$$FE_{\text{apply}} = FE_{\text{test}} \times k, \quad (5)$$

where k handles the resources for the testing and application phases, a shorter value of k, i.e., $k < 1$, assigns more resources to the testing phase versus the application phase; a larger value of k makes fewer resources available for the testing phase.

2.2 Reinforcement Learning

Reinforcement learning is “learning how to do and map situations to actions to maximize a numerical reward signal. An Agent needs to be told what actions to take; instead, she must discover which actions yield the most significant reward by attempting them” [12].

Table 5. Inverted generational distance (median and IQR values)

Problem	Front	DMOEA/D-TAOS	DMOEA/D-SL
FDA1	1	7.49e-04 3.08e-04	8.06e-04 2.87e-04 ==
	2	8.25e-04 2.43e-04	7.31e-04 3.24e-04 ==
	3	8.49e-04 2.67e-04	7.99e-04 2.93e-04 ▼
	4	7.71e-04 2.71e-04	7.14e-04 2.70e-04 ▼
	5	8.33e-04 2.82e-04	7.00e-04 2.38e-04 ▼
FDA3	1	6.80e-04 4.16e-04	6.92e-04 3.22e-04 ▲
	2	2.20e-03 1.25e-03	1.97e-03 1.46e-03 ▼
	3	2.93e-03 2.27e-03	2.90e-03 2.30e-03 ▼
	4	3.21e-03 3.31e-03	3.27e-03 2.57e-03 ▲
	5	3.78e-03 2.67e-03	4.57e-03 4.78e-03 ==
DMOP1	1	5.43e-04 1.13e-04	5.19e-04 1.96e-04 ==
	2	5.04e-04 1.43e-04	4.63e-04 9.23e-05 ▼
	3	5.29e-04 2.26e-04	4.98e-04 9.75e-05 ▼
	4	5.13e-04 1.59e-04	5.12e-04 1.40e-04 ▼
	5	5.08e-04 2.18e-04	4.92e-04 1.20e-04 ▼
DMOP2	1	5.34e-04 1.34e-04	5.33e-04 1.64e-04 ▼
	2	4.76e-04 1.54e-04	4.72e-04 1.12e-04 ==
	3	5.18e-04 1.37e-04	5.09e-04 1.26e-04 ==
	4	4.94e-04 1.13e-04	5.13e-04 1.83e-04 ▲
	5	4.79e-04 1.26e-04	4.76e-04 1.53e-04 ▼
DF4	1	2.99e-03 1.43e-03	3.57e-03 1.57e-03 ▲
	2	2.86e-03 1.01e-03	3.07e-03 1.71e-03 ▲
	3	5.49e-03 5.07e-04	5.35e-03 5.72e-04 ▼
	4	8.52e-03 3.20e-04	8.46e-03 3.85e-04 ▼
	5	1.12e-02 1.95e-04	1.12e-02 2.76e-04 ▲
DF6	1	3.86e-04 1.61e-04	4.00e-04 1.73e-04 ▲
	2	2.01e-01 1.92e-01	2.04e-01 2.74e-01 ==
	3	1.51e-02 7.50e-03	1.15e-02 1.07e-02 ▼
	4	1.43e-02 7.36e-03	1.07e-02 1.06e-02 ▼
	5	1.24e-02 6.55e-03	9.28e-03 9.45e-03 ▼
DF10	1	6.62e-03 5.77e-04	6.50e-03 4.09e-04 ▼
	2	6.48e-03 5.11e-04	6.43e-03 2.15e-04 ▼
	3	6.52e-03 5.74e-04	6.50e-03 3.29e-04 ▼
	4	6.62e-03 4.84e-04	6.66e-03 2.71e-04 ==
	5	6.76e-03 3.57e-04	6.81e-03 2.04e-04 ==
DF12	1	2.21e-03 1.43e-04	2.10e-03 3.25e-04 ▼
	2	2.17e-03 2.94e-04	2.24e-03 3.61e-04 ▲
	3	2.21e-03 2.13e-04	2.18e-03 2.05e-04 ▼
	4	1.97e-03 2.39e-04	1.98e-03 1.94e-04 ▲
	5	2.13e-03 2.49e-04	2.09e-03 1.89e-04 ==

In Figure 2, it can be seen that two important components of reinforcement learning are the Agent and the Environment. The Agent is the model that needs to be trained to make decisions.

2.2.1 SARSA (λ)

SARSA (λ) is a reinforcement learning algorithm used to train an Agent in an environment to make sequential decisions and learn to maximize a reward signal accumulated over time. It is a variant of the SARSA algorithm (State-Action-Reward-State-Action), which introduces an additional parameter called "lambda" (λ) to allow a trade-off between forward and backward learning.

The SARSA (λ) algorithm applies the TD (λ) prediction method to state-action pairs rather than

just states, requiring a trace for each pair. Let $E_t\ (s, a)$ be the trace of the state-action pair s, a [9].

Next, algorithm 1 presents the general mechanism of SARSA (λ). The eligibility trace update is defined in Equation 6:

$$e(S_t, A_t) \leftarrow \lambda\gamma e(S_t, A_t) + 1. \quad (6)$$

The calculation of the temporal error is defined in the Equation 7:

$$\delta \leftarrow \text{Reward} + \gamma Q(S_{t+1}, A_{t+1}) - Q(S_t, A_t). \quad (7)$$

The temporal error is used to update the Q values and improve the estimation of the quality of the actions in the different states. The update of the Q values is defined by the Equation 8:

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha\delta e(S_t, A_t), \quad (8)$$

where $e(S_t, A_t)$ is the eligibility trace for the current state-action pair, δ is the temporal error, $Reward$ is the calculated reward, α is the learning rate, ε is the policy parameter $\varepsilon -$ greedy, γ is a discount factor, and λ controls the amount of forward and backward learning.

3 Proposed Algorithm DMOEA/D-SL

This work proposes a new AOS method using a reinforcement learning technique called SARSA (λ) or SARSA Lambda. Furthermore, this new AOS has been integrated into a dynamic MOEA/D algorithm (DMOEA/D).

Two mechanisms proposed by Deb [13] have been added to the MOEA/D algorithm proposed by Zhang and Li in 2007 [14] to make an algorithm capable of working with dynamic problems, a change detection mechanism based on detectors, and the change response mechanism called "A".

We have taken the multi-arm approach and used it as actions, in this case four variants of the differential evolution trader are being used as actions The SARSA Lambda mechanism has been incorporated into the main loop of DMOEA/D. The general structure of this integration is shown in algorithm 2.

SARSA Lambda is initialized with the values corresponding to each of its parameters and is assigned to an object called an Agent; in this step, QTable is also initialized (line 3). For the episode loop, the main DMOEA/D loop is used (line 14), and for the step loop, the loop that runs through the population list (line 18) has been used.

In each turn of the main loop, S_t is initialized in the initial state S_0 (line 15), and the Agent assigns A_t an action based on S_t. To select an action, it uses an ε-greedy policy, where a random number is generated between $[0, 1]$; if the generated value is less than ε, an action (operator) is taken randomly, otherwise best action for the given state is selected (line 16).

We get all the values stored in the QTable for S_t and A_t and assign them to $QVal_t$ (line 17). Once inside the loop that runs through the population, we apply the action (operator) obtained and produce a child called y' (line 19). If y' is better than any of the neighbors of individual i state S_0 is assigned to S_t; otherwise, S_t will be assigned the following state (lines 26-52).

Subsequently, the following action, A_{t+1}, is calculated based on the state S_{t+1}, the value Q_{t+1} is obtained from the QTable for S_{t+1} and A_{t+1}, the eligibility traces are updated with S_t and A_t, the reward for S_t and A_t is calculated, the delta value is calculated, and the QTable is updated. Finally, the state S_t and the current action A_t are updated.

This process is repeated until the stopping criterion has been met. It should also be considered that in each problem change, the QTable must be reinitialized (line 67).

2.2 Actions Pool

Five genetic operators are evaluated. Four different versions of the differential evolution (DE) crossover operator [6] were tested as actions. In addition to each crossover operator, the polynomial mutation operator was also applied:

Action 1: apply DE/rand/1 , $v^i \leftarrow x^i + F \times (x^{r1} - x^{r2})$,

Action 2: apply DE/rand/2 , $v^i \leftarrow x^i + F \times (x^{r1} - x^{r2}) + F \times (x^{r3} - x^{r4})$,

Action 3: apply DE/current-to-rand/1 , $v^i \leftarrow x^i + K \times \left(x^i - x^{r1}\right) + F \times (x^{r2} - x^{r3}) + F \times (x^{r4} - x^{r5})$,

Action 4: apply DE/current-to-rand/2 , $v^i \leftarrow x^i + K \times \left(x^i - x^{r1}\right) + F \times (x^{r2} - x^{r3})$.

The four crossover operators use F=0.5 and CR=1.0, and polynomial mutation with a distribution index of 20 and a 20% of mutation probability.

4 Computational Experiments

Table 1 shows the eight dynamic multi-objective benchmark problems of 2 and 3 objectives used in this experiment. For each algorithm and front of the dynamic multi-objective problem, 30 independent runs were conducted.

The objective of the experimentation is to compare the proposed algorithm against the state-of-the-art algorithm called "Adaptive operator selection with test-and-apply structure for decomposition-based multi-objective optimization (TAOS)" [6]. Table 2 shows the parameters used for each algorithm used. The algorithms were implemented in the Java language.

The values of the parameters for MOEA/D have been taken from state-of-the-art, and the values of the parameters of the SARSA Lambda Agent have been obtained by assigning values arbitrarily by performing multiple experiments to determine the current values.

4.1 Results

The experimentation results are presented below in a table by metric (hypervolume, generalized spread, and inverted generational distance). Wilcoxon non-parametric test was applied with a significance level of 5%.

The first column in the table presents the problem. The second column presents the problem front. Columns three to four present the results of each algorithm. The algorithm in the third column is taken as a reference (MOEA/D-TAOS).

The following symbols are included in the results tables: the symbol ▲ means that there is statistical significance in favor of the reference algorithm, ▼ that there is statistical significance in favor of the algorithm that is compared with the reference algorithm (in favor of the current column), and == means there is no statistical significance.

The cells marked in dark gray represent the winning algorithm in a given problem and the front, and second places are marked in light gray.

4.1.1 Hypervolume

The hypervolume denotes the multidimensional volume of the objective space weakly dominated by an approximation set [5]. In Table 3, the third column is considered as the reference algorithm. As seen in the previous table, in the hypervolume metric, with the Wilcoxon test for the DMOEA/D-SL algorithm, 8 first places are obtained with statistical significance in favor. In comparison, for the DMOEA/D-TAOS algorithm, they obtained 18 first places with statistical significance in their favor.

4.1.2 Generalized Spread

Generalized Spread measures the uniformity and their dispersion of the solutions found [5]. In Table 4, the third column is considered as the reference algorithm. As seen in the previous table, in the generalized spread metric, with the Wilcoxon test for the DMOEA/D-SL algorithm, 20 first places with statistical significance are obtained in favor. In contrast, for the DMOEA/D-TAOS algorithm, 8 first places are obtained with statistical significance in favor.

4.1.3 Inverted Generational Distance

The inverted generation distance provides the average distance between any point in the reference set and its closest point in the approximation set [5]. In Table 5, the third column is considered as the reference algorithm. As seen in the previous table, in the inverted generational distance metric, with the Wilcoxon test for the DMOEA/D-SL algorithm, 21 first places are obtained with statistical significance in favor.

In comparison, TAOS obtains 9 first places with statistical significance in favor. The results obtained for the group of problems presented in this experiment suggest that the proposed algorithm is better in generalized spread and inverted generational distance metrics.

These metrics indicate that the proposed algorithm produces quality solutions with good approximation to the Pareto front and good dispersion. Regarding the hypervolume metric, the clear winner is the state-of-the-art algorithm. The main limitation of using an Agent to make the automatic selection of genetic operators in the

algorithm is the parameter configuration of the Agent; a good parameter configuration can give us quality solutions, but on the other hand, we do use a wrong parameter configuration for the Agent we will get low-quality solutions.

In this work, the parameter values of the SARSA Lambda Agent have been obtained by assigning values arbitrarily by performing multiple experiments to determine the current values. The source code can be downloaded from[1].

5 Conclusions and Future Work

This work proposes a new adaptive operator selection strategy using a reinforcement learning agent. This SARSA Lambda reinforcement learning strategy has been integrated into a dynamic multi-objective decomposition-based algorithm called DMOEA/D-SL. Furthermore, a state-of-the-art adaptive operator selection strategy, in this case, "Adaptive operator selection with test-and-apply structure for decomposition-based multi-objective optimization (TAOS)," has also been integrated into a dynamic multi-objective decomposition-based algorithm.

In the case of the proposed algorithm, the Agent learns to select the best genetic operator at a given moment, even when the definition of the problem changes over time. Extensive experimentation has been performed, and the results have been evaluated with three metrics: hypervolume, generalized spread, and inverted generation distance.

The Wilcoxon test has been applied with a significance level of 5%. The Wilcoxon test suggests that experimentation results are favorable in two metrics for DMOEA/D-SL. These results suggest that the algorithm produces high-quality solutions with a good approximation to the Pareto front and good dispersion.

Future work proposes exploring the parameters of the SARSA Lambda Agent more broadly to achieve better results in the three metrics used. On the other hand, using more than four genetic operators would test the behavior of the strategies used in this work and their performance.

Finally, other reinforcement learning strategies can also be considered to improve the quality of the solutions generated by the algorithm further.

[1] github.com/JAlfredoBrambila/DMOEAD_SL

Acknowledgments

The authors thank CONAHCyT, Mexico, for the support provided through scholarships for postgraduate studies and the National System of Researchers.

References

1. **Azzouz, R., Bechikh, S., Said, L. B. (2016).** Dynamic multi-objective optimization using evolutionary algorithms: A survey. Adaptation, Learning, and Optimization, pp. 31–70. DOI: 10.1007/978-3-319-42978-6_2.
2. **Li, K., Fialho, A., Kwong, S., Zhang, Q. (2014).** Adaptive operator selection with bandits for a multiobjective evolutionary algorithm based on decomposition. IEEE Transactions on Evolutionary Computation, Vol. 18, No. 1, pp. 114–130. DOI: 10.1109/tevc.2013.2239648.
3. **Pei, J., Mei, Y., Liu, J., Yao, X. (2022).** An investigation of adaptive operator selection in solving complex vehicle routing problem. PRICAI 2022: Trends in Artificial Intelligence, pp. 562–573. DOI: 10.1007/978-3-031-20862-1_41.
4. **Fialho, Á., Costa, L. D., Schoenauer, M., Sebag, M. (2010).** Analyzing bandit-based adaptive operator selection mechanisms. Annals of Mathematics and Artificial Intelligence, Vol. 60, No. 1-2, pp. 250–64. DOI: 10.1007/s10472-010-9213-y.
5. **Brambila-Hernández, J. A., García-Morales, M. Á., Fraire-Huacuja, H. J., Angel, A. B. D., Villegas-Huerta, E., Carbajal-López, R. (2023).** Experimental evaluation of adaptive operators selection methods for the dynamic multiobjective evolutionary algorithm based on decomposition (DMOEA/D). Studies in

Computational Intelligence, pp. 307–330. DOI: 10.1007/978-3-031-28999-6_20.

6. **Sun, L., Li, K. (2020).** Adaptive operator selection based on dynamic Thompson sampling for MOEA/D. Lecture Notes in Computer Science, pp. 271—284. DOI: 10.10 07/978-3-030-58115-2_19.
7. **Dong, L., Lin, Q., Zhou, Y., Jiang, J. (2022).** Adaptive operator selection with test-and-apply structure for decomposition-based multi-objective optimization. Swarm and Evolutionary Computation, Vol. 68, pp. 101013. DOI: 10.1016/j.swevo.2021.101013.
8. **Lin, W., Lin, Q., Ji, J., Zhu, Z., Coello, C. A. C., Wong, K. (2021).** Decomposition-based multiobjective optimization with bicriteria assisted adaptive operator selection. Swarm and Evolutionary Computation, Vol. 60, pp. 100790. DOI: 10.1016/j.swevo.2020.100790.
9. **Goldberg, D. E. (1990).** Probability matching, the magnitude of reinforcement, and classifier system bidding. Machine Learning, Vol. 5, pp. 407–425. DOI: 10.1023/A:1022681708029.
10. **Thierens, D. (2005).** An adaptive pursuit strategy for allocating operator probabilities. Proceedings of the 7th annual conference on Genetic and evolutionary computation, pp. 1539–1546. DOI: 10.1145/1068009.1068251.
11. **Xue, Y., Zhu, H., Liang, J., Słowik, A. (2021).** Adaptive crossover operator based multi-objective binary genetic algorithm for feature selection in classification. Knowledge-Based Systems, Vol. 227, pp. 107218. DOI: 10.1016/ j.knosys.2021.107218.
12. **Sutton, R. S., Barto, A. G. (1998).** Reinforcement learning, second edition: An Introduction. London, England: MIT Press.
13. **Deb, K., Bhaskara, U., Rao, N., Karthik, S. (2007).** Dynamic multi-objective optimization and decision-making using modified NSGA-II: A case study on hydro-thermal power scheduling. International conference on evolutionary multi-criterion optimization, pp. 803–817. DOI: 10.1007/978-3-540-70928-2_60.
14. **Zhang, Q., Li, H. (2007).** MOEA/D: A multiobjective evolutionary algorithm based on decomposition. IEEE Transactions on Evolutionary Computation, Vol. 11, No. 6, pp. 712–731. DOI: 10.1109/tevc.2007.892759.
15. **Goh, C., Tan, K. C. (2009).** A competitive-cooperative coevolutionary paradigm for dynamic multiobjective optimization. IEEE Transactions on Evolutionary Computation, Vol. 13, No. 1, pp. 103–127. DOI: 10.1109/ tevc.2008.920671.
16. **Farina, M., Deb, K., Amato, P. (2004).** Dynamic multiobjective optimization problems: test cases, approximations, and applications. IEEE Transactions on Evolutionary Computation, Vol. 8, No. 5, pp. 425–442. DOI: 10.1109/tevc.2004.831456.
17. **Jiang, S., Yang, S., Yao, X., Tan, K. C., Kaiser, M., Krasnogor, N. (2018).** Benchmark problems for CEC2018 competition on dynamic multiobjective optimization. Newcastle University.

Article received on 18/01/2024; accepted on 08/05/2024.
**Corresponding author is Héctor Joaquín Fraire-Huacuja.*