D4.4 Core module of the self-learning nexus engine

Lead: Lluís Echeverria (Eurecat)

Date : 30/06/2024

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 101003881

Project Deliverable

Project Number	Project Acronym	Project Title
101003881	NEXOGENESIS	Facilitating the next generation of effective and intelligent water-related policies, utilizing artificial intelligence and reinforcement learning to assess the water-energy-food-ecosystem (WEFE) nexus

Instrument:	Thematic Priority
H2020 RIA	LC-CLA-14-2020

Title
D4.4 Core module of the self-learning nexus engine

Contractual Delivery Date	Actual Delivery Date
M34: June 2024	M34

Start Date of the project	Duration
01 September 2021	48 months

Organisation name of lead contractor for this deliverable	Document version
EUT	1.0

Dissemination level	Deliverable Type
Public	**Demonstrator**

Authors (organisations)
Lluís Echeverria (EUT), Chaymaa Dkouk (EUT), and Nuria Nievas (EUT)
Reviewers (organisations)
Lydia Vamvakeridou-Lyroudia (KWR)

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 101003881

Abstract

Deliverable D4.4 *Core module of the self-learning nexus engine* is classified as a "Demonstrator." This document accompanies the deliverable and offers detailed explanations about the algorithmic fundaments and technical aspects of the Self-Learning Nexus Assessment Engine (SLNAE) and the NXG DSS. It is intended to complement the digital solutions developed from Task T4.4 *Reinforcement Learning engine.*

The self-learning nexus engine is the core mechanism that supports multi-objective decision-making in the SLNAE tool. The self-learning term refers to the underlying Artificial Intelligence (AI) and Machine Learning (ML) technology, enabling the creation of agents that autonomously learn (i.e. self-learning) optimal policy combinations (i.e. policy packages) to achieve the nexus-related objectives. We propose Multi Objective (Deep) Reinforcement Learning (MODRL) as the foundational family of ML algorithms to implement the NXG Decision Support System (DSS).

The current version of the Self-Learning Nexus Assessment Engine is embedded into the public release of the SLNAE at the following urls: https://slnae-dev.nexogenesis.eu or https://nepat-dev.nexogenesis.eu.

The SLNAE acronym, which is the reference to this tool in the Grant Agreement, has been changed to NEPAT (Nexus Policy Assessment Tool), during the project. This was decided because it was easier to pronounce than SLNAE and also it reflected better for the general audience the content of the tool. In this document, the two acronyms SLNAE and NEPAT are both being used indiscriminately and they refer to the same tool. This has been done on purpose, because SLNAE is mentioned in the Grant Agreement and also because several related actions have started and happened under with the tool named SLNAE, while now, more recent activities are referring to NEPAT and this document is an intermediate report for this tool. By the end of the project, NEPAT will be the name of this tool.

Related Deliverables:
D3.4 Complexity science models implemented for all the Case Studies including explanatory manuals
D3.6 Sensitivity/Uncertainty Analysis Report
D4.1 Self-learning nexus engine specifications and technical design
D4.3. Simulation Policy Framework

Keywords
SLNAE; NEPAT, Nexus Decision Support System, Multi-Objective optimization, Multi-Objective Deep Reinforcement Learning

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 101003881

Abbreviation/Acronyms

CCS	Convex Coverage Set
CS	Case Study
DMORL	Deep Multi-Objective reinforcement learning
DMP	Data Management Plan
DRL	Deep Reinforcement Learning
DSS	Decision Support System
FNRL	Fingerprint networked reinforcement learning
GUI/UI	Graphical User Interface/User Interface
GA	Grant Agreement
ICT	Information and Communication Technologies
IDR	Internal Data Repository
JSON	JavaScript Object Notation
MDP	Markov Decision Process.
ML	Machine Learning
MOMDP	Multi-Objective Markov Decision Process
MOO	Multi-objective optimisation
MORL	Multi-objective reinforcement learning
NEPAT	Nexus Policy Assessment Tool
NXG	Nexogenesis project
PCS	Pareto Coverage Set
POMOMDP	Partially Observable Multi-Objective Markov Decision Process
PQL	Pareto Q learning
R	Reward function
RL	Reinforcement Learning
SDM	System Dynamic Model
SLNAE	Self-Learning Nexus Assessment Engine
SH	Stakeholder
RP	Reference Pathway
WEF	Water-Energy-Food (Nexus)
WEFE	Water-Energy-Food-Ecosystem (Nexus)
WP	Work Package

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 101003881

Contents

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 101003881

Figures

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 101003881

1. Introduction

The WEFE nexus framework highlights the intricate linkages between water, energy, food and ecosystems dominated by complexity and modulated by climatic and socio-economic drivers. For instance, water is essential for agriculture (food production) and energy generation (hydropower and cooling in thermal power plants). Conversely, energy is needed for water extraction, treatment, and distribution, as well as for food production and processing. This interconnectedness creates a web of dependencies where actions in one sector can have significant ripple effects across others.

In this context, decision-making face a multi-objective (MO) problem due to the complex and often conflicting objectives inherent in managing these resources. Addressing the nexus requires balancing the competing demands of each sector while considering their interrelated impacts on sustainability, economic growth, and social well-being.

Instead of aggregating the various nexus objectives into a single scalar signal/objective for planning or learning purposes, we consider them individually. This approach allows the end user to define their own aggregation function, technically known as the utility function, based on their preferences. By doing so, we avoid providing an imperfect solution that would restrict the tool's recommendation scope. We also give more “freedom” to the users to express their preferences.

The self-learning nexus engine is the core mechanism that supports MO decision-making in the SLNAE tool. The self-learning term refers to the underlying Artificial Intelligence (AI) and Machine Learning (ML) technology, enabling the creation of agents that autonomously learn (i.e. self-learning) optimal policy combinations (i.e. policy packages) to achieve the nexus-related objectives. We propose Multi Objective (Deep) Reinforcement Learning (MODRL) as the foundational family of ML algorithms to implement the NXG Decision Support System (DSS).

Each Case Study (CS) represents a unique optimization challenge, formulated as an individual problem to optimise in their own, unique, policy decision space. Additionally, there are three further layers of complexity. First, each CS includes a set of reference scenarios (RCP-SSP combinations), each depicting different potential future conditions. Second, the complexity models implemented by WP3 incorporate randomness in the input data, allowing for both deterministic and stochastic execution modes. Finally, one of the CSs (Inkomati) introduces an additional decision-making dimension: the year when a policy is applied, which we call the ‘dynamic policies’ mode, in contrast to the ‘static mode’ with fixed policies. All these options are available through different implementations of the CSs’ System Dynamic Models (SDMs). Consequently, an agent will be trained for every combination of CS, reference scenario, randomness execution mode, and, for the Inkomati CS, i.e., the policy mode.

This project has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement No 101003881

Deliverable D4.4 *Core module of the self-learning nexus engine* is classified in th GA as a "Demonstrator." This document accompanies the deliverable and offers detailed explanations about the algorithmic fundaments and technical aspects of the Self-Learning Nexus Engine and the NXG DSS. It is intended to complement the digital solutions developed from Task T4.4 *Reinforcement Learning engine.*

The current version of the Self-Learning Nexus Engine is embedded into the public release of the SLNAE at the following urls: https://slnae-dev.nexogenesis.eu or https://nepat-dev.nexogenesis.eu. This version will be further updated with data from each CS.

1.1. Disclaimer

The public version of the SLNAE/NEPAT is currently under development and has been designated as a Beta version (Figure 1). All its modules, including the Self-Learning Nexus Engine and its components or inputs (i.e. SDMs, policies, goals, UI, etc) are undergoing continuous validation and are subject to change. **The results and screenshots in this deliverable are provided solely to demonstrate that the Self-Learning Nexus Engine has been successfully developed and integrated into the SLNAE**.

Following the validation process, in which CSs and SHs will certify the behaviour of the SLNAE/NEPAT modules, the final version of the tool will be deployed. In the upcoming months, the Self-Learning Nexus Engine will be continuously run and adjusted to meet the final requirements of CSs and SHs. This process will be expedited depending on the CS. For the front runners, the Self-Learning Nexus Engine will be ready by August 2024 (M36), including the Inkomati CS. For the followers, it will be ready by December 2024 (M40).

The final version of the SLNAE/NEPAT is expected to be ready by February 2025 (M42) and will be reported in D4.5 *Final version of the self-assessment nexus engine with the corresponding validation* (M42). This final version will include additional secondary functionalities that are not necessary for the success of the upcoming Workshops (WSs). with the CS, to be organised in collaboration with WP1 and WP5.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 101003881

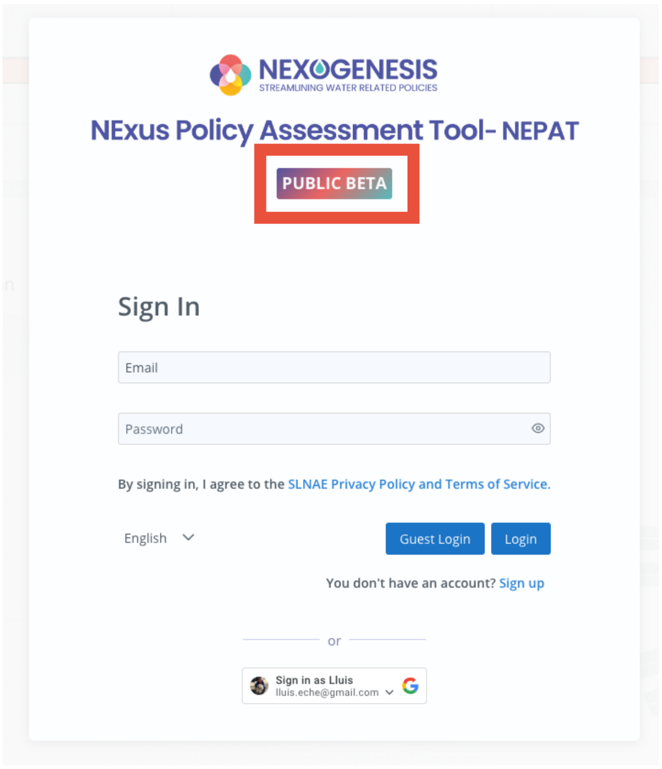

NEXOGENESIS
STREAMLINING WATER RELATED POLICIES
NExus Policy Assessment Tool- NEPAT
PUBLIC BETA
Sign In
Email
Password
By signing in, I agree to the SLNAE Privacy Policy and Terms of Service.
English
Guest Login
Login
You don't have an account? Sign up
or
Sign in as Lluis
lluis.eche@gmail.com

Figure 1. SLNAE UI. Beta version warning.

1.2. Links to the ICT4WATER Cluster

WP4 is considered the 'digital' WP in the Nexogenesis project. Thus, it is the natural link between the project and the ICT4WATER Cluster[1].

The outcomes of the task T4.4 are specially linked to the ICT4WATER Updated Digital Water Action Plan[2] and its 'Intelligent and smart systems', 'Actor engagement and co-creation' and 'Policies' Action Groups[3].

With regards to the actions and activities outlined in the Digital Water Action Plan, this deliverable and the SLNAE/NEPAT tool (developed under WP4 in NXG) contributes to the following (all action numbers refer to the Digital Water Action Plan):

- The 'Intelligent and smart systems' action 2 activities 1 & 2: A multi-optimization decision-making framework (the Self-Learning Nexus Engine), based on Deep Reinforcement Learning, is implemented for decision support.
- The 'Intelligent and smart systems' action 5 activities 1 & 2: Uncertainty is taken into account in the integrated complexity science models.

[1] https://ict4water.eu/
[2] https://ict4water.eu/wp-content/uploads/2023/06/Update-Digital-Water-Action-Plan-V7.pdf
[3] https://ict4water.eu/action-groups/

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 101003881

- The 'Actor engagement and co-creation' action 1 activity 1: A public online tool (the SLNAE) is developed.
- The 'Actor engagement and co-creation' action 2 activities 1 & 2: Stakeholders from all nexus sectors are taken into account during the Nexogenesis co-creation process for the SLNAE design.
- The 'Policies' action 5 activity 1: The SLNAE tool enables policy co-creation.
- The 'Policies' action 6 activity 2: The SLNAE tool enables improved management of governance complexity including uncertainty and other factors.

1.3. NEPAT: a new name for the SLNAE

During the initial period of the project, it was necessary to explain the definition and meaning of the SLNAE to non-technical audiences (e.g. stakeholders) on multiple occasions. The term "Self-Learning" is a technical concept related to AI and ML algorithms involved in its development, which can be difficult to understand and may generate confusion. Additionally, the acronym SLNAE and the full name are not easy to pronounce. Therefore, it was decided with the NXG consortium that a new name was needed.

WP4 led the initiative to define a new name that would be self-explanatory and avoid technical jargon, while incorporating Nexogenesis-related concepts such as "policy assessment" or "impact". Several options were proposed under the cocreation framework and put to a vote, and eventually, the name "Nexogenesis - Nexus Policy Assessment Tool (NEPAT)" was selected as the best option. The new name is simpler, easier to pronounce, and more reflective of the tool's and project's purpose. In order to be consistent with the GA and other official documentation, both names are valid to refer to the SLNAE.

Thus, the SLNAE tool is referred to as either SLNAE or NEPAT indiscriminately.

1.4. Document structure

The document is structured as follows: Section 2 describes the requirements for the Self-Learning Nexus Engine implementation. Section 3 briefly reviews the state of the art in multi-objective optimization and multi-objective reinforcement learning within the nexus domain, highlighting the advantages of using reinforcement learning in the NXG DSS. Section 4 establishes the foundational knowledge and mathematical formulation in the multi-objective optimization domain and introduces the selected MORL algorithms. Section 5 presents the technical design of the NXG decision-making problem and its formalization. Section 6 showcases the initial results of these implementations. Finally, Section 7 demonstrates the current integration of the Self-Learning Nexus Engine into the NEPAT UI, and Section 8 concludes with the findings and outlines the next steps.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 101003881

2. Requirements for the development of the self-learning nexus engine

In order to develop the NEPAT's self-learning nexus engine and DSS (step 3 in Figure 2 - Right), the key following components (Figure 2 - Left) have been previously developed (steps 1 and 2), by other technical WPs, through the co-creation framework (Figure 2 - Right) for Nexus Policy packages identification:

- Policies
- Nexus Indicators
- Nexus Policy Goals, years and targets
- System Dynamics Models (SDM)s

All these components have been provided by each of the five NXG CSs and have been integrated into the NXG Simulation Policy Framework (see deliverable D4.3 *Simulation Policy Framework*). The aim of the DSS is to identify which policy combinations (i.e., policy packages) are suitable for achieving the goals' targets. This objective is further explained and technically formalized later in this document. To measure the impact of each policy package, an SDM is required. Policies have been implemented in the SDMs, so their impact and trade-offs can be quantitatively measured using the Nexus indicators, which are also implemented in the SDMs. A Nexus Goal is linked to one of these indicators, making it measurable.

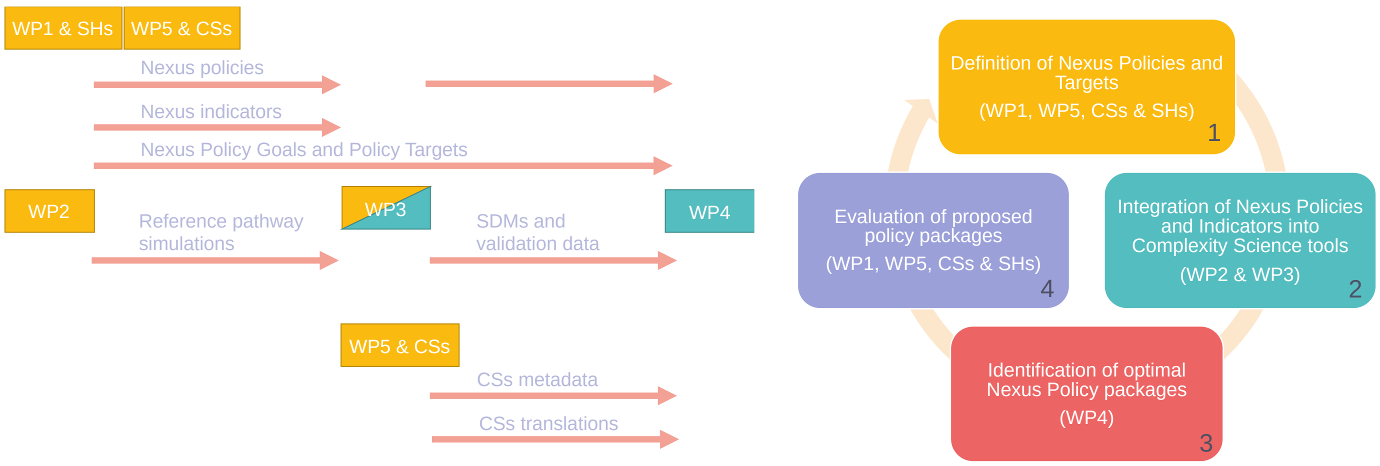

WP1 & SHs
WP5 & CSs
Nexus policies
Nexus indicators
Nexus Policy Goals and Policy Targets
WP2
Reference pathway simulations
WP3
SDMs and validation data
WP4
WP5 & CSs
CSs metadata
CSs translations
Definition of Nexus Policies and Targets
(WP1, WP5, CSs & SHs)
1
Integration of Nexus Policies and Indicators into Complexity Science tools
(WP2 & WP3)
2
Identification of optimal Nexus Policy packages
(WP4)
3
Evaluation of proposed policy packages
(WP1, WP5, CSs & SHs)
4

Figure 2. Left: NXG cross-WP data pipelines in the Internal Data Repository. Right: NXG co-creation framework for Nexus Policy packages identification. Source: D4.1

All these components are available in the Nexogenesis Internal Data Repository or Data Lake (see D4.2 *Data Lake for data sharing*). WP4 collects all these resources and uses them to develop the NEPAT.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 101003881

2.1. Policies

Each CS has proposed an arbitrary number of policies, based on their interest and goals, and the final analysis they want to carry out. Table 1 summarizes the most relevant information, from a decision-making multi-objective point of view, associated with the available policies, as how many of them have been proposed, how many have been finally implemented or its type, among other information.

Table 1. NEPAT available policies

CS	N. of regions	N. of proposed policies	N. of proposed policies at region level	N. of implemented policies	Incompatible policies	Type of policies (Fixed/Dynamic)
Nestos	2+1	10	5	12	No	F
Lielupe	2+1	6	5	24	Yes	F
Jiu	1	7	-	18	Yes	F
Adige	1	12	-	12	No	F
Inkomati	1	11	-	11	No	F/D

Given that some CS are transboundary, there are further considerations regarding policies, because the countries involved in each CS may have/select different policies, applying only to their subregion. Hence, there are two CSs (Nestos and Lielupe) that have split their implementation and offer sub-regions, or sub-basins, in their SDMs. In Lielupe, the regions of Latvia and Lithuania have been separately considered as the system sub-regions. In Nestos, although there are 14 sub-basins identified in the SDMs, policies can only be applied to the sub-regions of Bulgaria or Greece. In these cases, we consider there to be three regions: two sub-regions plus the case study as a whole. Therefore, some policies can be applied at a regional level, allowing decision-making and trade-off analysis at a spatial scale as well.

In some cases, CSs have proposed policies that can be modulated (Figure 3). For example, Policy P5 in the Lielupe CS, targets the reduction of Greenhouse Gas (GHG) emissions. This policy affects the fraction of grasslands to renewables parameter in the SDMs, and it is implemented as a range between 0 and 1, or as a percentage. Initially, we discretize this range into three values: 0.33, 0.66, and 1, resulting in three additional policies. Thus, technically speaking, there may be more implemented policies than those originally proposed. Later, this discretization can be modified based on SHs' feedback. Continuing with this example, these additional policies cannot be applied simultaneously because they modulate the same parameter and correspond to the same real policy. We represent this restriction by marking them as incompatible policies.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 101003881

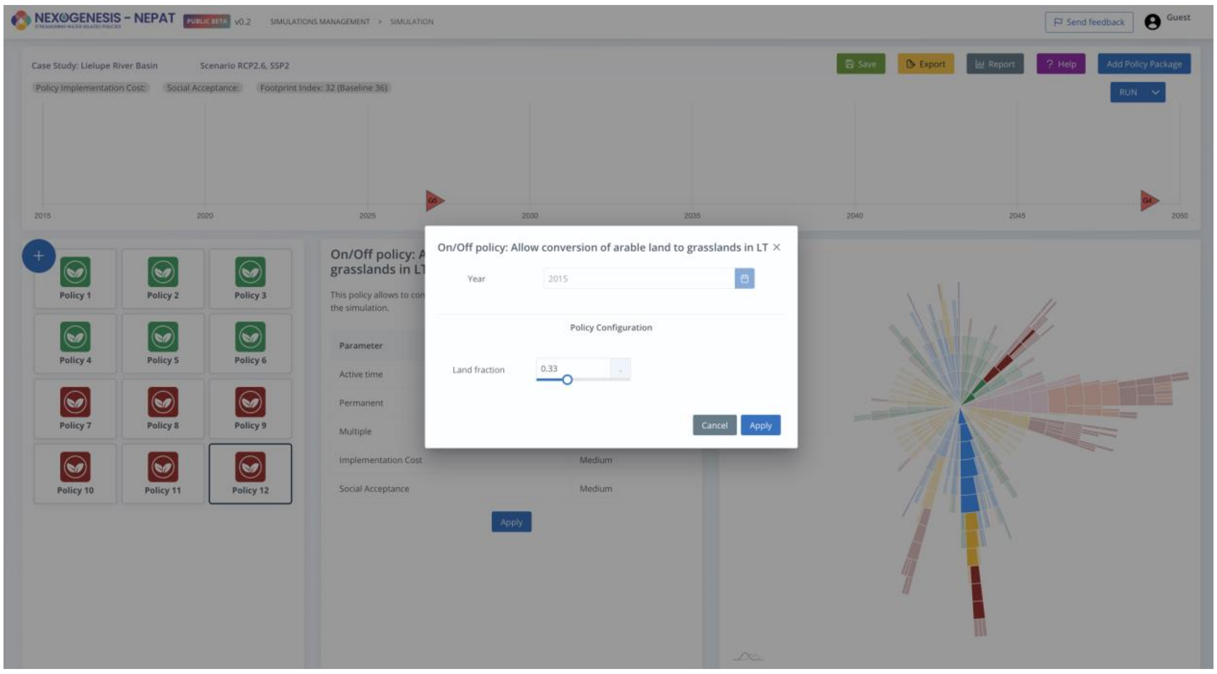


Figure 3. NEPAT UI. Example of Policy modulation

In other cases, like the Nestos CS, there were some policies which were described as appliable to the whole basin, while being implemented in the SDM as two variables (one for each region). In these cases, we split the policy into one for each region.

Finally, there are several ways to classify policies, such as by the direct nexus sector affected, the indirect sectors, or the way the policies function. Here, we classify the proposed policies based on whether they can be applied at any custom time between 2015 and 2050 or not. In the former case, we mark them as ‘Dynamic policies’, enabling decision-making and trade-off analysis at a temporal scale, and in the latter case as ‘Fixed policies’.

Additional information about policies, policy integration, and the policy simulation framework can be found in D4.3 *Simulation Policy Framework*.

2.2. Policy packages

The number of possible policy combinations (i.e. policy packages), considering both the number of implemented policies and their compatibility, can be computed approximately. Table 2 provides an overview of these possible policy packages, illustrating the variety of combinations available based on the given policies. These computations include both valid and invalid policy packages, since the presented algorithms in section 4.3 don’t have action masking implemented, meaning all combinations, whether valid or invalid, can be explored by the agents during the training phase. Hence, these would be the numbers that accurately describe the magnitude of the problem for each case study.

Table 2. CS possible combinations with fixed policies

CS	Policy combinations
Nestos	28,672
Lielupe	218,103,808
Jiu	2,621,440
Adige	28,672
Inkomati	13,312

From all the presented CSs, Inkomati provided SDMs with dynamic policies (Table 1), meaning that they can be configured to start their application in any set time in the simulation. For this case, we have computed the combinations for three scenarios, ordered by the granularity of the policy application time:

- Policy application every 5 years: meaning each policy can be applied in 2015, 2020, and so on up to 2050, providing 7 possible years to choose from.
- Policy application yearly: each policy can be applied in any of the 35 years of the simulation.
- Policy application monthly: each policy can be applied in any of the 420 months of the simulation.

Table 3. Inkomati combinations with dynamic policies

Scenario	Policy combinations
Policy application every 5 years	5.97E+24
Policy application yearly	1.52E+118
Policy application monthly	1.32E+1394

Table 3 shows the number of combinations for each of the considered scenarios. Given the size and complexity of these combination counts, tabular methods are impractical due to their inability to scale to these magnitudes. Therefore, we will employ Deep Reinforcement Learning (Deep RL) algorithms, which are better suited for handling large-scale problems. Although the second and third scenarios are highly detailed and case-specific, possibly with no real nexus application, they exemplify a significant level of complexity. Therefore, they can effectively highlight and demonstrate the advantages of the methodology introduced in the NXG project, particularly in this deliverable.

2.3. Goals

Each CS has proposed an arbitrary number of goals as well, based on its interest and objectives, and the final analysis they want to carry out. Table 4 summarizes the most relevant information, from a decision-making multi-objective point of view, associated with the available goals, as how many of them have been proposed or how many have been finally implemented, among other information.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 101003881

Table 4. NEPAT available goals

CS	N. of proposed goals	N. of proposed goals at region level	N. implemented goals	N. of goals affected by stochastic variables
Nestos	5	5	30	-
Lielupe	4	2	5	1
Jiu	7	-	9	0
Adige	3	-	3	1
Inkomati	14	-	14	9

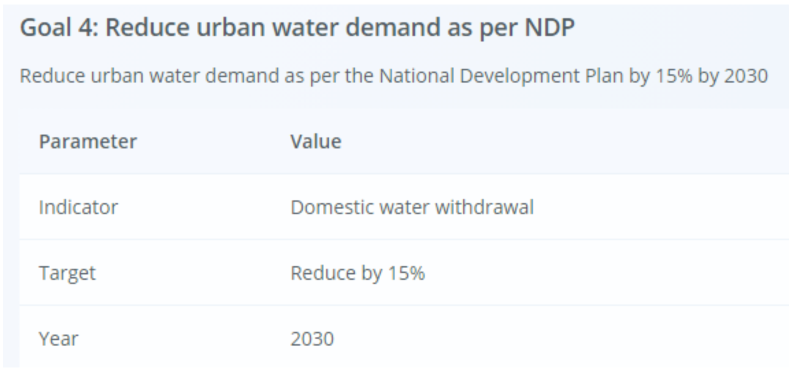
Similarly to the policies: i) goals can also be set at the sub-region level, and ii) there may be more implemented goals than originally proposed. For example, Goal G1 in the Jiu CS aims for an 87% reduction in GHG emissions by 2030 and a 97% reduction by 2050. Two additional goals have been implemented, each corresponding to one of these pairs of years and targets.

Finally, it is important to understand whether the proposed goals are influenced by the underlying data stochasticity implemented in the SDMs for proper consideration.

Additional information about goals, goals integration, and the policy simulation framework can be found in D4.3 *Simulation Policy Framework*.

2.3.1. Goal targets and years

Goals are defined as a three-tuple structure. First, the SDM indicator linked to the goal, which must be used to measure its performance and achievement. Second, the target value (e.g. an absolute amount, a percentage) is used as a threshold to evaluate the achievement of the corresponding goal. And third, a specific year between 2015 and 2050 when the goal achievement must be evaluated.



Goal 4: Reduce urban water demand as per NDP

Reduce urban water demand as per the National Development Plan by 15% by 2030

Parameter	Value
Indicator	Domestic water withdrawal
Target	Reduce by 15%
Year	2030

Figure 4, NEPAT UI goal attributes view

For example, in the Inkomati CS, Goal 4 is defined by the indicator "Domestic water withdrawal," with a target value of 15% to be achieved by a specified year. Figure 4 illustrates how this goal is represented in the NEPAT UI.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 101003881

3. A nexus multi-objective approach

Multi-objective optimization (MOO), also known as multi-criteria or multi-attribute optimization, deals with problems involving more than one objective function to be optimized simultaneously. These problems are ubiquitous in engineering, economics, logistics, and other fields where trade-offs between conflicting objectives need to be made, such as the nexus.

This is a reduced version because these results have been/will be submitted for publication in a peer reviewed journal. For more information please contact the authors - Lluís Echeverria Rovira (Eurecat) lluis.echeverria@eurecat.org.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 101003881

4. Multi-objective RL: problem setting, taxonomy and algorithms

This section first introduces the multi-objective problem setting, extending the single-objective Markov Decision Process (MDP) formalization. Second it presents the taxonomy utilized throughout the rest of the document, and final, it describes the MORL and DMORL algorithms implemented to solve the proposed nexus decision-making problem.

A deep introduction to MDPs and RL can be found in [19]. The following sub-sections are based on [25] and [20].

4.1. Multi-objective problem setting

Extending the MDP formalization presented in D4.1 section *6.2.5.1 MDP formalization*, we introduce the Multi-Objective Markov Decision Process (MOMDP) as the proposed mechanism to support decision-making in the nexus.

A MOMDP is represented by the tuple $\langle S, A, T, \gamma, \mu, R\rangle$, where:

- S is the state space
- A is the action space
- $T : S \times A \times S \rightarrow [0,1]$ is a probabilistic transition function
- $\gamma \in [0, 1)$ is a discount factor
- $\mu : S \rightarrow [0,1]$ is a probability distribution over initial states
- $R : S \times A \times S \rightarrow \mathbb{R}^d$ is a vector-valued reward function, specifying the immediate reward for each of the considered $d \geq 2$ objectives

The key distinction between a single-objective MDP and MOMDP lies in the vector-valued reward function. In a MOMDP, this function provides a numerical feedback signal for each

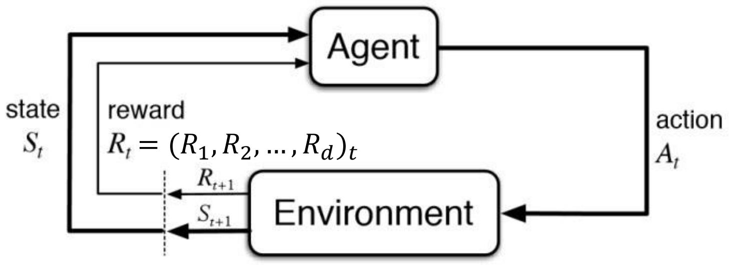


Figure 5. Multi-Objective Reinforcement Learning agent-environment interaction flow

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 101003881

objective under consideration, resulting in a reward vector whose length matches the number of objectives (Figure 5).

This is a reduced version because these results have been/will be submitted for publication in a peer reviewed journal. For more information please contact the authors - Lluís Echeverria Rovira (Eurecat) lluis.echeverria@eurecat.org.

4.2. Multi-objective taxonomy

This section introduces various concepts from the multi-objective optimization domain that will be utilized later in the document. Building on the previously presented term, the utility function, two extensions are defined. These extensions form the basis for the different types of solution sets that can be derived.

4.2.1. A Monotonically increasing utility function

A monotonically increasing utility function, u, follows the rule that if a policy enhances one or more of its objectives without diminishing any others, the scalarized value will also increase.

$$(\forall i : V_i^{\pi} \geq V_i^{\pi\prime}) \wedge (\exists i : V_i^{\pi} > V_i^{\pi\prime}) \Rightarrow u(V^{\pi}) \geq u(V^{\pi\prime})$$

A monotonically increasing utility function can represent both linear (with nonzero positive weights) and non-linear user preferences. Monotonicity in the utility function is a minimal assumption for MORL, as it aligns with the fundamental definition of an objective: we always seek to maximize value in any of the objectives.

4.2.2. A linear utility function

A linear utility function calculates the inner product of a weight vector w and a value vector V^{π}.

$$u(V^{\pi}) = w^T V^{\pi}$$

Each element of the weight vector w specifies how much one unit of value for the corresponding objective contributes to the scalarized value. The elements of the weight vector are all positive real numbers and are constrained to sum to 1.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 101003881

4.2.3. Solution sets

In single-objective RL problems, there exists a unique optimal value V^*, and there can be multiple optimal policies π^* that all achieve this value. The goal in single-objective RL is typically to learn an optimal policy. However, in MOO cases, without any additional information about the user's utility, there can be multiple potentially optimal value vectors V. Therefore, it is necessary to consider different sets of potentially optimal value vectors and policies.

The selection of the solution set is crucial for the efficiency of algorithms used in solving MO problems, as it requires computing all the policies within these sets.

4.2.3.1. The undominated set

The undominated set $U(\Pi)$ consists of the subset of all possible policies Π and their associated value vectors for which there exists a potential utility function u that achieves the maximum scalarized value.

$$U(\Pi) = \{\pi \in \Pi \mid \exists u, \forall \pi' \in \Pi : \ u(V^\pi) \geq u(V^{\pi'})\}$$

However, the undominated set might include redundant policies. These are policies that are optimal for a particular utility function, but there are other policies that are also optimal for the same utility function. In such cases, it is unnecessary to keep all these policies to maintain optimal utility.

4.2.3.2. The coverage set

A set $CS(\Pi)$ is considered a coverage set if it meets two conditions: it must be a subset of $U(\Pi)$, and, for every u in, it includes a policy with the highest scalarized value.

$$CS(\Pi) \subseteq U(\Pi) \ \wedge \ (\forall u, \exists \pi \in \ CS(\Pi), \forall \pi' \in \Pi : \ u(V^\pi) \geq u(V^{\pi'}))$$

4.2.3.3. The Pareto Front

In those cases where the utility function u is any monotonically increasing function, then the Pareto Front (PF) is the undominated set $U(\Pi)$.

$$PF(\Pi) = \left\{\pi \in \Pi \,\middle|\, \nexists \pi' \in \Pi : \ V^{\pi'} \succ_p V^\pi\right\}$$

where $\succ_p$ is the Pareto dominance relation

$$V^\pi \succ_p V^{\pi'} \Leftrightarrow \left(\forall i : \ V_i^\pi \geq V_i^{\pi'}\right) \wedge \left(\exists i : \ V_i^\pi > V_i^{\pi'}\right)$$

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 101003881

In other words, the Pareto front is constituted by all the policies such that there is no other policy with value that is equal of better in all the objectives. A collection of policies whose value functions align with the PF is known as a Pareto Coverage Set (PCS).

4.2.3.4. The convex hull

The convex hull (CH) consists of the subset of Π for which there exists a weight vector w (for a linear utility function) such that the linearly scalarized value is maximized. It is the undominated set for linear utility functions.

$$CH(\Pi) = \{\pi \in \Pi \mid \exists w, \forall \pi' \in \Pi : w^T V^{\pi} \geq w^T V^{\pi'}\}$$

4.2.3.5. The convex coverage set

A set CCS(Π) is a convex coverage set if it is a subset of CH(Π) and if, for every weight vector, it includes a policy whose linearly scalarized value is maximized.

$$CCS(\Pi) \subseteq \mathrm{CH}(\Pi) \wedge \left(\forall \mathrm{w}, \exists \pi \in \mathrm{CCS}(\Pi), \forall \pi' \in \Pi : \mathrm{w}^{\top} \mathrm{V}^{\pi} \geq \mathrm{w}^{\top} V^{\pi'}\right)$$

4.3. Multi-objective Reinforcement learning algorithms

Multi-objective reinforcement learning encompasses a diverse array of methodologies, from bandit problems to deep reinforcement learning architectures. For a thorough review see [12]. A prevalent and widely adopted approach in MORL is extending established single-objective model-free value-based methods like Q-learning [26] to handle multiple objectives simultaneously.

Some examples of this practice include MO Q-Learning, MPMO Q-Learning [27], Pareto Q-Learning [25] and MPQ-learning [28]. These methods are restricted to tabular representations of Q-values, limiting their applicability to more complex problems. For these types of problems, some DRL algorithms have been adapted to multiple objectives. Most of these methods extend the single objective DQN architecture and some examples are Pareto DQN [29] and Envelope Q-learning [30].

There are also some alternatives to the seen value-based approaches, adopting policy search algorithms. Some examples are the Expected Utility Policy Gradient (EUPG) implementation by Roijers et al. [31], the multi-objective categorical Actor-Critic (MOCAC) by Reymond et al. [32] or the multi-objective extension of PPO by Xu et al [33].

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 101003881

For the Self-Learning Nexus Engine, we have considered the tabular method Pareto Q-learning and the DRL algorithm Envelope Q-learning implementations from MORL-Baselines [34] Python library. Considering the number of goals we have (Table 4), we have focused on the algorithms that do not impose explicit limitations on the number of objectives. Finally, we have adapted the available implementations to match the NXG case and its characteristics.

The usage of Pareto Q-learning will allow us to explicitly capture Pareto-optimal solutions, allowing a thorough modelling of the trade-offs between different objectives. Also, since its tabular nature, it will allow us to have a more variate range of recommendations. On the other hand, Envelope Q-learning focuses on efficiently representing the envelope of the Pareto front, simplifying the computational complexity associated with maintaining multiple solutions. This method scales effectively to larger state-action spaces, being the best option for solving the bigger problems presented in section 2.2.

This is a reduced version because these results have been/will be submitted for publication in a peer reviewed journal. For more information please contact the authors - Lluís Echeverria Rovira (Eurecat) lluis.echeverria@eurecat.org.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 101003881

5. The Nexogenesis decision-making problem formalization

[12] and [20] propose six scenarios where a multi-objective approach is required. The NXG decision-making case falls into the decision support scenario, where the user's preferences are unknown or difficult to specify.

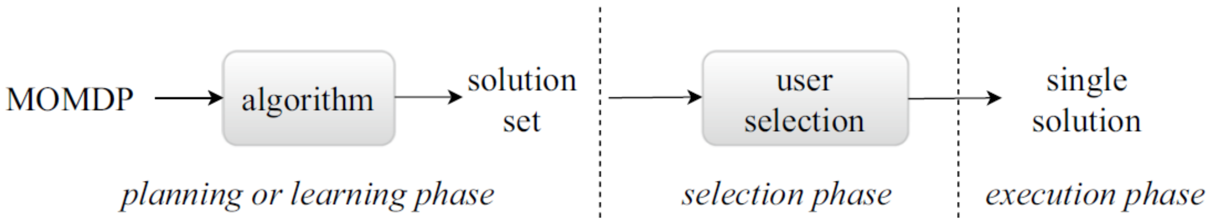


Figure 6. Decision support scenario diagram, extracted from [X]

As seen in Figure 6, in this scenario, since a priori we don't know the user's preferences, we compute a solution set with the Pareto front solutions to be able to respond with an optimal solution regardless of the preferences. Once the user wants a recommendation, he provides its preferences and with those set, the selection phase is run, obtaining the best solution according to the set preferences.

The MOMDP in the NXG decision-making problem is presented as an episodic case where, given an initial situation, the agent must provide the undominated set solutions. Two problem variants are proposed, one considering deterministic environments (i.e. deterministic SDMs), and the other considering stochastic environments. In the first case, the average reference scenario is used. In the second case, available data stochasticity in SDMs is not aggregated, thus a more challenging situation is presented. Further details on how these environments are constructed can be found in D4.3.

Each project CS represents a unique optimization problem. Additionally, there are three further layers of complexity. First, each CS includes a set of reference scenarios (RCP-SSP combinations), each depicting different potential future conditions. Second, the complexity models implemented by WP3 incorporate randomness in the input data, allowing for both deterministic and stochastic execution modes. Finally, one of the CSs (Inkomati) introduces an additional decision-making dimension: the year when a policy is applied, which we call the 'dynamic policies' mode, in contrast to the 'static mode' with fixed policies. All these options are available through different implementations of the CS' System Dynamic Models (SDMs). Consequently, an agent will be trained for every combination of CS, reference scenario, randomness execution mode, and, for the Inkomati CS, the policy mode.

This is a reduced version because these results have been/will be submitted for publication in a peer reviewed journal. For more information please contact the authors - Lluís Echeverria Rovira (Eurecat) lluis.echeverria@eurecat.org.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 101003881

6. Initial results

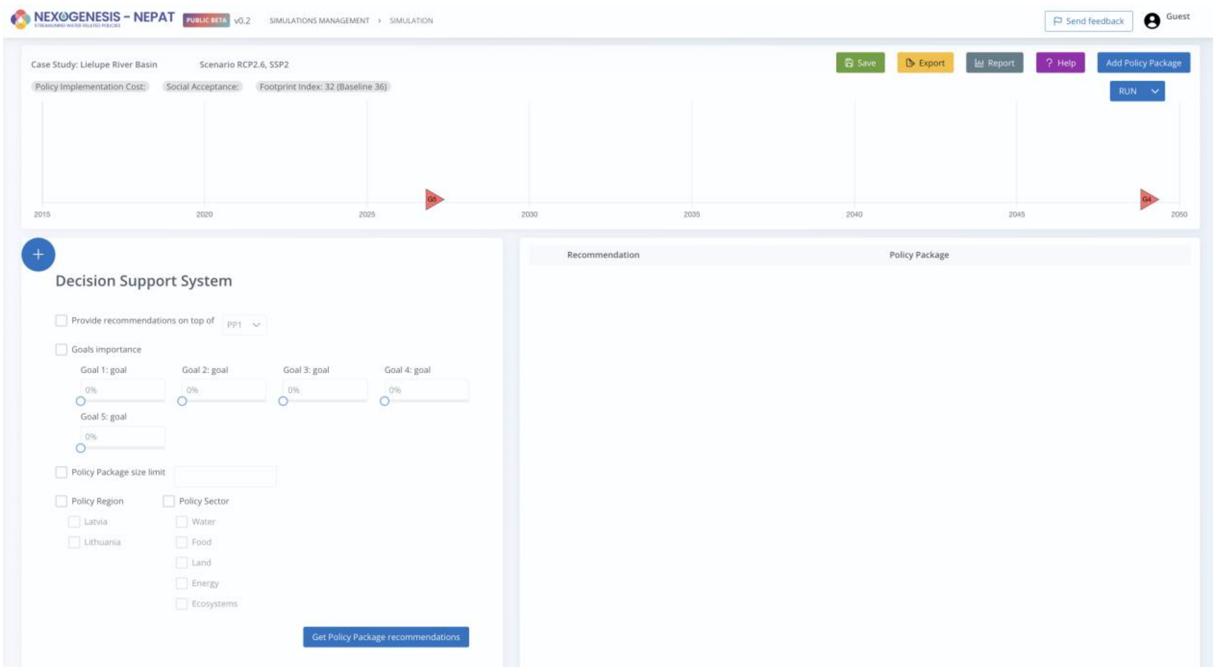
Given the problems presented in section 2.2, we applied the proposed mechanisms to the Lielupe, Jiu and Inkomati CSs with fixed policies using the RCP26-SSP2 scenario and deterministic SDMs. Additionally, we also ran some initial processes for the Inkomati CS with dynamic policies every 5 years. We will continue adjusting the framework, training agents and validating their performance over the coming months.

This is a reduced version because these results have been/will be submitted for publication in a peer reviewed journal. For more information please contact the authors - Lluís Echeverria Rovira (Eurecat) lluis.echeverria@eurecat.org.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 101003881

7. The NEPAT DSS

The trained agents discussed in the previous section have been integrated into NEPAT to validate the recommendation pipeline. A specific view in the NEPAT UI, the Decision Support System view (Figure 7), has been deployed, allowing users to interact with the recommendation service.


Case Study: Lielupe River Basin
Scenario RCP2.6, SSP2
Policy Implementation Cost: Social Acceptance: Footprint Index: 32 (Baseline 36)

Decision Support System

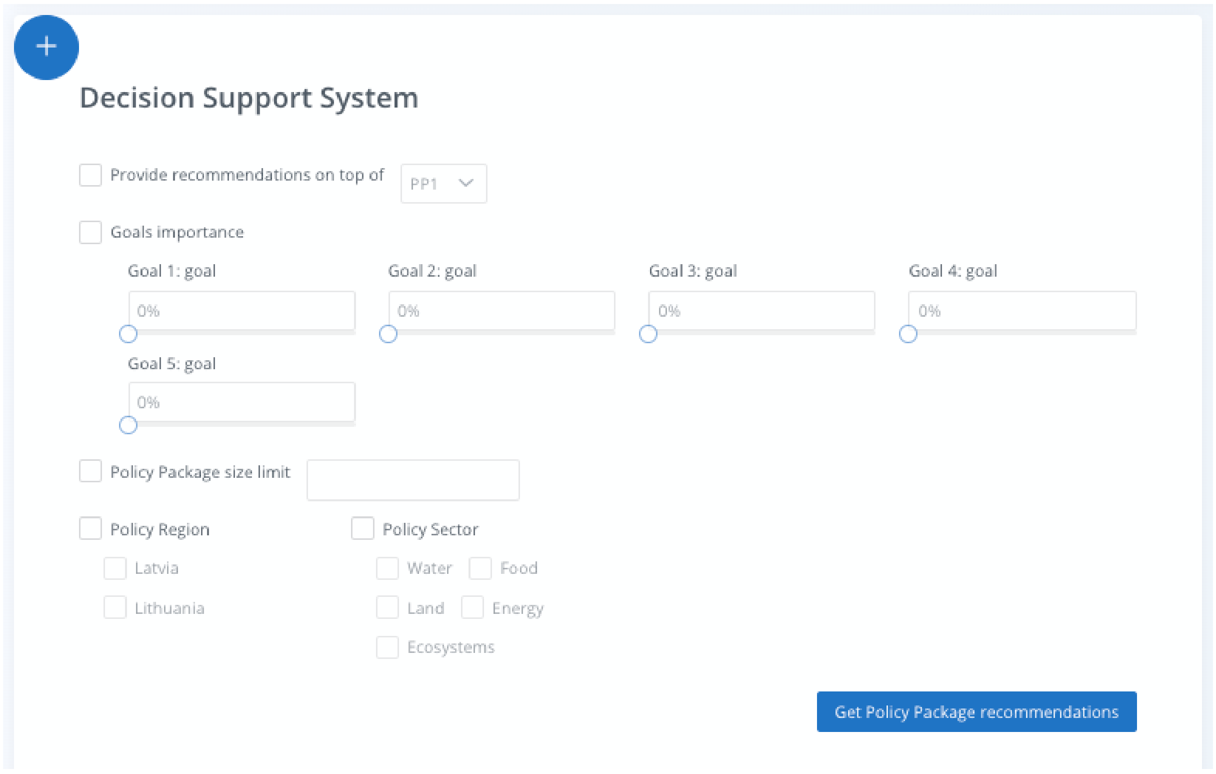
Provide recommendations on top of PP1
Goals importance
Goal 1: goal 0%
Goal 2: goal 0%
Goal 3: goal 0%
Goal 4: goal 0%
Goal 5: goal 0%
Policy Package size limit
Policy Region: Latvia, Lithuania
Policy Sector: Water, Food, Land, Energy, Ecosystems

Recommendation	Policy Package

Figure 7. NEPAT UI. Decision Support System view

The view is divided into two sections. On the left side (Figure 8), the user can determine their preferences to build a customized utility function. Various functionalities are provided in this section. First, the user may choose to obtain recommendations based on a pre-defined policy package. This option sets the initial state (starting policy package) from which the corresponding agent will provide recommendations. Second, and most importantly, the user can define the goals importance by setting different weights using slider mechanisms, one per goal. Next, the user can specify the limit size of the recommended policy package. Finally, as an additional filter, the recommended policies can be limited by sector or region (in those sub-regional contexts) of application.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 101003881


Decision Support System

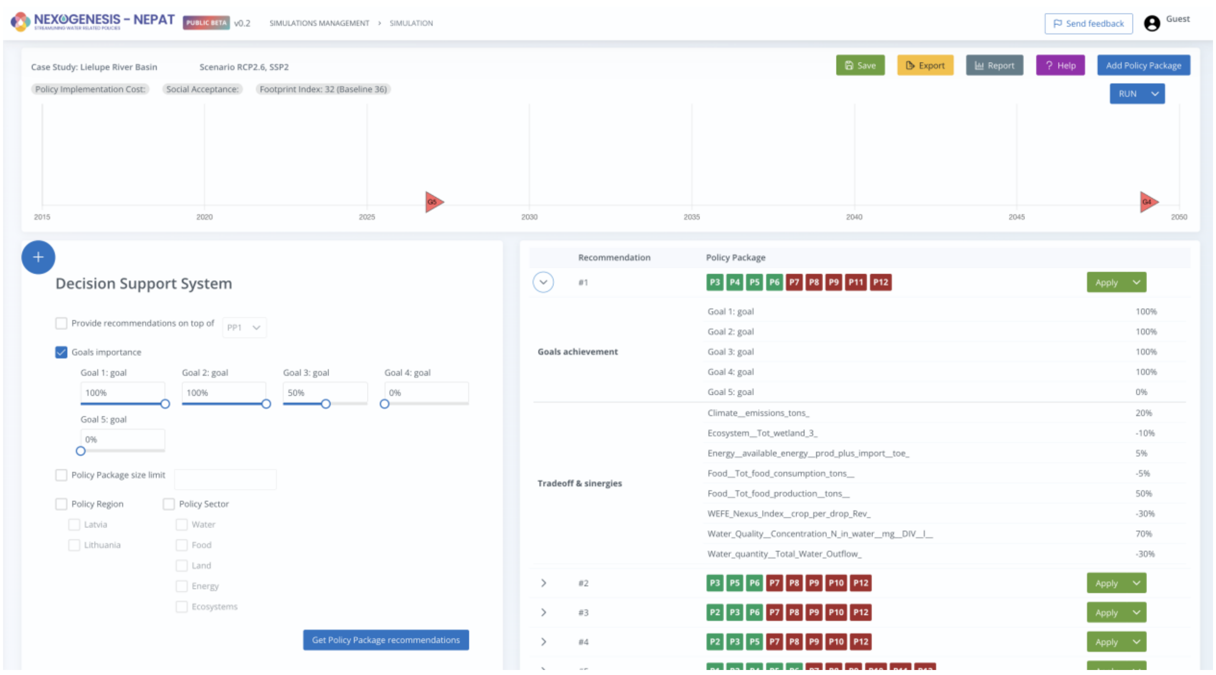
- [ ] Provide recommendations on top of PP1
- [ ] Goals importance

Goal 1: goal	Goal 2: goal	Goal 3: goal	Goal 4: goal	Goal 5: goal
0%	0%	0%	0%	0%

- [ ] Policy Package size limit
- [ ] Policy Region: [ ] Latvia, [ ] Lithuania
- [ ] Policy Sector: [ ] Water, [ ] Food, [ ] Land, [ ] Energy, [ ] Ecosystems

Figure 8. NEPAT UI. Decision Support System view. Zoom on preferences section.

Once the custom utility function is configured, the user must press the “Get Policy Package Recommendation” button. This action will trigger the DSS to obtain the recommendations, which will be listed on the left side of the screen (Figure 9).



Case Study: Lielupe River Basin
Scenario RCP2.6, SSP2
Policy Implementation Cost:
Social Acceptance:
Footprint Index: 32 (Baseline 36)

Decision Support System

- [ ] Provide recommendations on top of PP1
- [x] Goals importance

Goal 1: goal	Goal 2: goal	Goal 3: goal	Goal 4: goal	Goal 5: goal
100%	100%	50%	0%	0%

- [ ] Policy Package size limit
- [ ] Policy Region: [ ] Latvia, [ ] Lithuania
- [ ] Policy Sector: [ ] Water, [ ] Food, [ ] Land, [ ] Energy, [ ] Ecosystems

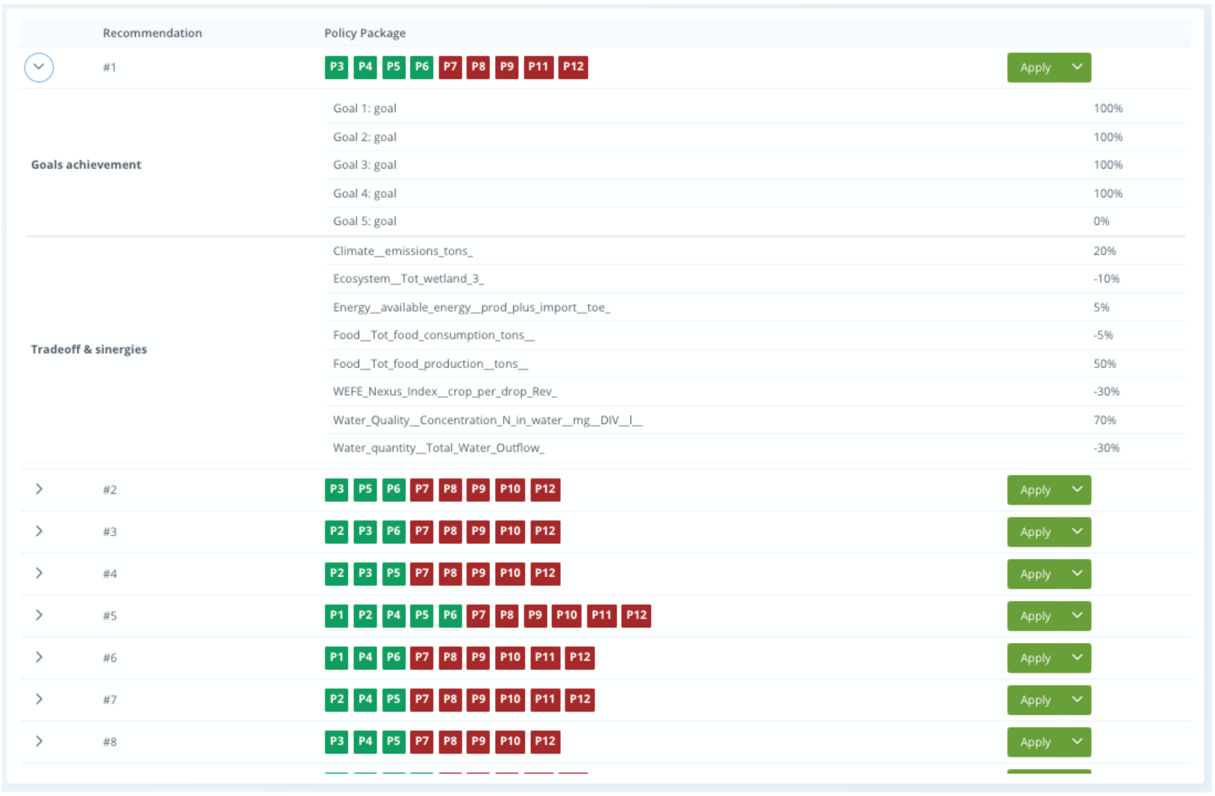
Recommendation	Policy Package		
#1	P3 P4 P5 P6 P7 P8 P9 P11 P12		Apply
Goals achievement	Goal 1: goal	100%	
	Goal 2: goal	100%	
	Goal 3: goal	100%	
	Goal 4: goal	100%	
	Goal 5: goal	0%	
Tradeoff & sinergies	Climate__emissions_tons_	20%	
	Ecosystem__Tot_wetland_3_	-10%	
	Energy__available_energy__prod_plus_import__toe_	5%	
	Food__Tot_food_consumption_tons__	-5%	
	Food__Tot_food_production__tons__	50%	
	WEFE_Nexus_Index__crop_per_drop_Rev_	-30%	
	Water_Quality__Concentration_N_in_water__mg__DIV__l__	70%	
	Water_quantity__Total_Water_Outflow_	-30%	
#2	P3 P5 P6 P7 P8 P9 P10 P12		Apply
#3	P2 P3 P6 P7 P8 P9 P10 P12		Apply
#4	P2 P3 P5 P7 P8 P9 P10 P12		Apply

Figure 9. NEPAT UI. Decision Support System view providing policy package recommendations

The recommended policy packages are listed in a table (Figure 10), which can be expanded to obtain further information about the impact of these policies if applied. First, the achievement of goals is assessed, and second, the status of the nexus footprint indicators is provided. This

This project has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement No 101003881

approach enables a comprehensive overview of the policy package impact across the entire nexus.



	Recommendation	Policy Package	
	#1	P3 P4 P5 P6 P7 P8 P9 P11 P12	Apply
Goals achievement		Goal 1: goal	100%
		Goal 2: goal	100%
		Goal 3: goal	100%
		Goal 4: goal	100%
		Goal 5: goal	0%
Tradeoff & sinergies		Climate__emissions_tons_	20%
		Ecosystem__Tot_wetland_3_	-10%
		Energy__available_energy__prod_plus_import__toe_	5%
		Food__Tot_food_consumption_tons__	-5%
		Food__Tot_food_production__tons__	50%
		WEFE_Nexus_Index__crop_per_drop_Rev_	-30%
		Water_Quality__Concentration_N_in_water__mg__DIV__l__	70%
		Water_quantity__Total_Water_Outflow_	-30%
	#2	P3 P5 P6 P7 P8 P9 P10 P12	Apply
	#3	P2 P3 P6 P7 P8 P9 P10 P12	Apply
	#4	P2 P3 P5 P7 P8 P9 P10 P12	Apply
	#5	P1 P2 P4 P5 P6 P7 P8 P9 P10 P11 P12	Apply
	#6	P1 P4 P6 P7 P8 P9 P10 P11 P12	Apply
	#7	P2 P4 P5 P7 P8 P9 P10 P11 P12	Apply
	#8	P3 P4 P5 P7 P8 P9 P10 P12	Apply

Figure 10. NEPAT UI. Decision Support System view providing policy package recommendations. Zoom on recommendations table.

If the user decides to accept any of the recommendations, the proposed policy package can be directly imported into the Policy Package Builder section by clicking the apply button. This allows the user to continue analyzing the impacts of that policy package through the NEPAT functionalities.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 101003881

8. Conclusions

The AI algorithmic foundations of the Self-Learning Nexus Engine have been formalized, implemented, tested, and initially validated. Additionally, the Decision Support System service has been successfully implemented and deployed, achieving one of the main objectives of task T4.4 and WP4.

The self-learning nexus engine is the core mechanism that supports multi-objective decision-making in the SLNAE/NEPAT tool. The self-learning term refers to the underlying Artificial Intelligence (AI) and Machine Learning (ML) technology, enabling the creation of agents that autonomously learn (i.e. self-learning) optimal policy combinations (i.e. policy packages) to achieve the nexus-related objectives. We discuss and propose Multi Objective (Deep) Reinforcement Learning (MODRL) as the foundational family of ML algorithms to implement the NXG DSS.

Until M40 (end of task T4.4) this framework will be run for all CSs since each of the represents a unique optimization problem. Additionally, there are three further layers of complexity. First, each CS includes a set of reference scenarios (RCP-SSP combinations), each depicting different potential future conditions. Second, the complexity models implemented by WP3 incorporate randomness in the input data, allowing for both deterministic and stochastic execution modes. Finally, one of the CSs (Inkomati) introduces an additional decision-making dimension: the year when a policy is applied, which we call the 'dynamic policies' mode, in contrast to the 'static mode' with fixed policies. All these options are available through different implementations of the CSs' System Dynamic Models (SDMs). Consequently, an agent will be trained for every combination of CS, reference scenario, randomness execution mode, and, for the Inkomati CS, the policy mode, complementing those already presented.

The current version of the Self-Learning Nexus Engine is embedded in the public release of the SLNAE/NEPAT at the following urls: https://slnae-dev.nexogenesis.eu or https://nepat-dev.nexogenesis.eu.

Since the Self-Learning Nexus Engine inputs (i.e. the SDMs, the policies, the goals or the Nexus footprint) are under constant validation, the current published version of the service is designated as a Beta version and is subject to change. The DSS will be ready by August 2024 (M36) for the frontrunners CS (including the Inkomati CS), and for the followers CSs, it will be ready by December 2024 (M40). Final policy package recommendations will be reported in the corresponding WP5 deliverables (D.5.2 to D5.6) (M42).

The final version of the SLNAE is expected to be ready by February 2025 (M42) and will be reported in D4.5 *Final version of the self-assessment nexus engine with the corresponding validation* (M42).

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 101003881

9. References

[1] L. C. Thomas, “Constrained Markov decision processes as multi-objective problems.” University of Manchester. Department of Decision Theory., 1982.

[2] L. Zadeh, “Optimality and non-scalar-valued performance criteria,” *IEEE Trans. Automat. Contr.*, vol. 8, no. 1, pp. 59–60, 1963.

[3] Y. Haimes, “On a bicriterion formulation of the problems of integrated system identification and system optimization,” *IEEE Trans. Syst. Man. Cybern.*, no. 3, pp. 296–297, 1971.

[4] A. Charnes, W. W. Cooper, and R. O. Ferguson, “Optimal estimation of executive compensation by linear programming,” *Manage. Sci.*, vol. 1, no. 2, pp. 138–151, 1955.

[5] C. A. C. Coello, G. T. Pulido, and M. S. Lechuga, “Handling multiple objectives with particle swarm optimization,” *IEEE Trans. Evol. Comput.*, vol. 8, no. 3, pp. 256–279, 2004.

[6] M. Dorigo, M. Birattari, and T. Stutzle, “Ant colony optimization,” *IEEE Comput. Intell. Mag.*, vol. 1, no. 4, pp. 28–39, 2006.

[7] S. Kirkpatrick, C. D. Gelatt Jr, and M. P. Vecchi, “Optimization by simulated annealing,” *Science (80-.).*, vol. 220, no. 4598, pp. 671–680, 1983.

[8] C. M. Fonseca and P. J. Fleming, “Genetic algorithms for multiobjective optimization: formulationdiscussion and generalization.,” in *Icga*, 1993, vol. 93, no. July, pp. 416–423.

[9] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, “A fast and elitist multiobjective genetic algorithm: NSGA-II,” *IEEE Trans. Evol. Comput.*, vol. 6, no. 2, pp. 182–197, 2002.

[10] E. Zitzler, M. Laumanns, and L. Thiele, “SPEA2: Improving the strength Pareto evolutionary algorithm,” *TIK Rep.*, vol. 103, 2001.

[11] Q. Zhang and H. Li, “A multiobjective evolutionary algorithm based on decomposition,” *IEEE Trans. Evol. Comput.*, 2006.

[12] D. M. Roijers, P. Vamplew, S. Whiteson, and R. Dazeley, “A survey of multi-objective sequential decision-making,” *J. Artif. Intell. Res.*, vol. 48, pp. 67–113, 2013.

[13] S. Dhaubanjar, C. Davidsen, and P. Bauer-Gottwein, “Multi-objective optimization for analysis of changing trade-offs in the Nepalese water–energy–food nexus with hydropower development,” *Water*, vol. 9, no. 3, p. 162, 2017.

[14] T.-S. Uen, F.-J. Chang, Y. Zhou, and W.-P. Tsai, “Exploring synergistic benefits of Water-Food-Energy Nexus through multi-objective reservoir optimization schemes,” *Sci. Total Environ.*, vol. 633, pp. 341–351, 2018.

[15] M. Li, Q. Fu, V. P. Singh, D. Liu, and T. Li, “Stochastic multi-objective modeling for optimization of water-food-energy nexus of irrigated agriculture,” *Adv. Water Resour.*, vol. 127, pp. 209–224, 2019.

[16] F. Karamian, A. A. Mirakzadeh, and A. Azari, “Application of multi-objective genetic algorithm for optimal combination of resources to achieve sustainable agriculture based on the water-energy-food nexus framework,” *Sci. Total Environ.*, vol. 860, p. 160419, 2023.

[17] I. Okola, E. O. Omulo, D. O. Ochieng, and G. Ouma, “A comparison of evolutionary algorithms on a Large Scale Many-Objective Problem in Food–Energy–Water Nexus,” *Results Control Optim.*, vol. 10, p. 100195, 2023.

[18] F. Mansour, M. Al-Hindi, M. Abou Najm, and A. Yassine, “Multi-objective optimization for comprehensive water, energy, food nexus modeling,” *Sustain. Prod. Consum.*, vol. 38, pp. 295–311, 2023.

[19] R. S. Sutton and A. G. Barto, “Reinforcement learning: an introduction 2018 complete

This project has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement No 101003881

draft," 2017. doi: 10.1109/TNN.1998.712192.

[20] C. F. Hayes *et al.*, "A practical guide to multi-objective reinforcement learning and planning," *Auton. Agent. Multi. Agent. Syst.*, vol. 36, no. 1, p. 26, 2022.

[21] K. Van Moffaert, M. M. Drugan, and A. Nowé, "Hypervolume-based multi-objective reinforcement learning," in *Evolutionary Multi-Criterion Optimization: 7th International Conference, EMO 2013, Sheffield, UK, March 19-22, 2013. Proceedings 7*, 2013, pp. 352–366.

[22] O. Emamjomehzadeh, R. Kerachian, M. J. Emami-Skardi, and M. Momeni, "Combining urban metabolism and reinforcement learning concepts for sustainable water resources management: A nexus approach," *J. Environ. Manage.*, vol. 329, p. 117046, 2023.

[23] W. Zhang, A. Valencia, and N.-B. Chang, "Fingerprint Networked Reinforcement Learning via Multiagent Modeling for Improving Decision Making in an Urban Food–Energy–Water Nexus," *IEEE Trans. Syst. Man, Cybern. Syst.*, 2023.

[24] R. Wu, R. Wang, J. Hao, Q. Wu, and P. Wang, "Multiobjective multihydropower reservoir operation optimization with transformer-based deep reinforcement learning," *J. Hydrol.*, p. 130904, 2024.

[25] K. Van Moffaert and A. Nowé, "Multi-objective reinforcement learning using sets of pareto dominating policies," *J. Mach. Learn. Res.*, vol. 15, no. 1, pp. 3483–3512, 2014.

[26] C. J. C. H. Watkins and P. Dayan, "Q-learning," *Mach. Learn.*, vol. 8, no. 3–4, pp. 279–292, May 1992, doi: 10.1007/bf00992698.

[27] K. Van Moffaert, M. M. Drugan, and A. Nowé, "Scalarized multi-objective reinforcement learning: Novel design techniques," in *2013 IEEE symposium on adaptive dynamic programming and reinforcement learning (ADPRL)*, 2013, pp. 191–199.

[28] M. Ruiz-Montiel, L. Mandow, and J.-L. Pérez-de-la-Cruz, "A temporal difference method for multi-objective reinforcement learning," *Neurocomputing*, vol. 263, pp. 15–25, 2017.

[29] M. Reymond and A. Nowé, "Pareto-DQN: Approximating the Pareto front in complex multi-objective decision problems," 2019.

[30] R. Yang, X. Sun, and K. Narasimhan, "A generalized algorithm for multi-objective reinforcement learning and policy adaptation," *Adv. Neural Inf. Process. Syst.*, vol. 32, 2019.

[31] D. M. Roijers, D. Steckelmacher, and A. Nowé, "Multi-objective reinforcement learning for the expected utility of the return," in *Proceedings of the Adaptive and Learning Agents workshop at FAIM*, 2018, vol. 2018.

[32] M. Reymond, C. F. Hayes, D. Steckelmacher, D. M. Roijers, and A. Nowé, "Actor-critic multi-objective reinforcement learning for non-linear utility functions," *Auton. Agent. Multi. Agent. Syst.*, vol. 37, no. 2, p. 23, 2023.

[33] J. Xu, Y. Tian, P. Ma, D. Rus, S. Sueda, and W. Matusik, "Prediction-guided multi-objective reinforcement learning for continuous robot control," in *International conference on machine learning*, 2020, pp. 10607–10616.

[34] F. Felten *et al.*, "A toolkit for reliable benchmarking and research in multi-objective reinforcement learning," *Adv. Neural Inf. Process. Syst.*, vol. 36, 2024.

[35] H. Hasselt, "Double Q-learning," *Adv. Neural Inf. Process. Syst.*, vol. 23, 2010.

[36] H. Van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double q-learning," in *Proceedings of the AAAI conference on artificial intelligence*, 2016, vol. 30, no. 1.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 101003881