UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Facultat d'Informàtica de Barcelona

FIB

SEARCH & ANALYSIS OF NEW HEURISTICS FOR SOLVING NP-HARD PROBLEMS WITH DEEP REINFORCEMENT LEARNING

TOMÀS OSARTE SEGURA

Thesis supervisor: SERGIO ÁLVAREZ NAPAGAO (Department of Computer Science)

Degree: Bachelor's Degree in Informatics Engineering (Computing)

Bachelor's thesis

Facultat d'Informàtica de Barcelona (FIB)

Universitat Politècnica de Catalunya (UPC) - BarcelonaTech

Contents

1 Introduction and Contextualization

This type A graduation thesis is situated in the scope of the *Facultat d'Informàtica de Barcelona* (FIB) and pretends to develop a new heuristic with the Deep Reinforcement Learning framework with the ambition to exceed the state of the art.

In the field of computational complexity theory, a problem is called NP-hard if for all other problems which can be solved in non-deterministic polynomial-time, there exists a polynomial-time reduction to the problem, as it is explained in [1]. This type of problems have been recurrently studied in the field of computation and more recently in the last decade, a new approach to find some heuristics to solve them in a short amount of time have been developed using the framework of reinforcement learning. In this thesis, we will address the subset of combinatorial optimization problems known to be NP-hard.

1.1 Basic concept definition

In order to proceed with more dense explanations, there are some basic concepts that should be defined and properly explained.

1.1.1 NP-Hard Problems

NP-hard problems are computational challenges that are notoriously difficult to solve efficiently. They encompass a wide range of optimization, scheduling, and decision problems across various domains. Examples include the Traveling Salesman Problem, the Knapsack Problem, and the Boolean Satisfiability Problem.

These problems are relevant because they capture the complexity of many real-world scenarios and are fundamental in computer science and related fields. While finding exact solutions to NP-hard problems is often impractical, research into approximation algorithms and heuristic methods continues to drive innovation in algorithm design and optimization techniques. Overall, NP-hard problems pose significant computational challenges and inspire ongoing research efforts to develop more efficient solution strategies. For further details, readers are encouraged to consult [1, 2, 3].

1.1.2 Reinforcement Learning

Reinforcement Learning, commonly referred as RL is defined in [4] as an interdisciplinary area of machine learning concerned with how an intelligent agent should choose actions in different states of a dynamic environment in order to maximize the cumulative reward. RL is one of the three basic machine learning paradigms: supervised, unsupervised and reinforcement.

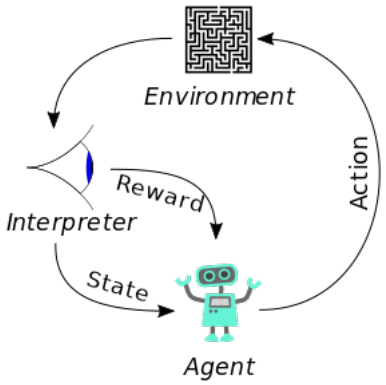

Figure 1: Typical framing of RL. [4]

The field of Reinforcement Learning is expansive, encompassing a wide range of concepts and techniques within machine learning. It is impractical to cover every aspect of RL comprehensively in this section. However, Figure 3, provided by [5], highlights the most essential concepts that serve as foundational knowledge for understanding RL effectively.

Term	Description	Pros	Cons
Model-free RL	The environment is a black box. Agents mostly conduct a trial-and-error procedure to learn on its own. They use rewards to update their decision models	The algorithm does not need a model of the environment	Requires a large amount of samples
Model-based RL	Agents construct a model that simulates the environment and use it to generate future episodes. By using the model, agents can estimate not only actions but also future states	Speed up learning and improve sample efficiency	Having an accurate and useful model is often challenging
Temporal difference learning	Use TD error to estimate the value function. For example, in Q-learning, $Q(s_t, a_t) = Q(s_t, a_t) + \beta(r_t + \gamma \max_a Q(s_{t+1}, a))$	Fast convergence as it does not need to wait until the episode ends	Estimates can be biased
Monte-Carlo method	Estimate the value function by obtaining the average of the same values in different episodes $Q(s_t, a_t) = \lim_{N->\infty} \sum_{i=1}^{N} Q(s_t^i, a_t^i)$	The values are non-biased estimates	Slow convergence and the estimates have high variances. Has to wait until episode ends to do updates
Continuous action space	The number of control actions is continuous	A policy-based method can be used	Cannot use a value-based method
Discrete action space	The number of control actions is discrete and finite	Both policy-based method and value-based method can be used	Intractable if the number of actions is large
Deterministic policy	The policy maps each state to a specific action	Reduce data sampling	Vulnerable to noise and stochastic environments
Stochastic policy	The policy maps each state to a probability distribution over actions	Better exploration	Requires a large amount of samples
On-policy method	Improve the current policy that the agent is using to make decisions	Safer to explore	May be stuck in local minimum solutions
Off-policy method	Learn the optimal policy (while samples are generated by the behavior policy)	Instability. Often used with an experience replay	Might be unsafe because the agent is free to explore
Fully observable environment	All agents can observe the complete states of the environment	Easier to solve than partially observable environments	The number of state can be large
Partially observable environment	Each agent only observes a limited observation of the environment	More practical in real-world applications	More difficult to solve as the agents require to remember past states

Figure 2: RL important concepts. [5]

Furthermore, within the realm of Reinforcement Learning, there exists a diverse array of algorithms,

each offering unique approaches to problem-solving. Our aim is to explore a broad spectrum of these algorithms. A comprehensive review of various RL algorithms can be found in [5], which provides a thorough differentiation among them.

Method	Description and Advantage	Technical Requirements	Drawbacks	Implementation
Value-based method				
DQN	Use a deep convolutional network to directly process raw graphical data and approximate the action-value function	• Experience replay • Target network • Q-learning	○ Excessive memory usage ○ Learning instability ○ Only for discrete action space	[85] [86] [87] [88]
Double DQN	Mitigate the DQN's maximization bias problem by using two separate networks: one for estimating the value, one for selecting action.	• Double Q-learning	○ Inherit DQN's drawbacks	[88]
Prioritized Experience Replay	Prioritize important transitions so that they are sampled more frequently. Improve sample efficiency	•Importance sampling	○ Inherit DQN's drawbacks ○ Slower than non-prioritized experience replay (speed)	[85] [88]
Dueling Network	Separate the DQN architecture into two streams: one estimates state-value function and one estimates the advantage of each action	• Dueling network architecture • Prioritized replay	○ Inherit DQN's drawbacks	[88]
Recurrent DQN	Integrate recurrency into DQN Extend the use of DQN in partially observable environments	• Long Short Term Memory	○ Inherit DQN's drawbacks	[88]
Attention Recurrent DQN	Highlight important regions of the environment during the training process	• Attention mechanism • Soft attention • Hard attention	○ Inherit DQN's drawbacks	[68]
Rainbow	Combine different techniques in DQN variants to provide the state-of-the-art performance on Atari domain	• Double Q-learning • Prioritized replay •Dueling network •Multi-step learning •Distributional RL •Noisy Net	○ Inherit DQN's drawbacks	[85]
Policy-based method				
A3C/A2C	Use actor-critic architecture to estimate directly the agent policy. A3C enables concurrent learning by allowing multiple learners to operate at the same time	• Multi-step learning • Actor-critic model • Advantage function • Multi-threading	○ Policy updates exhibit high variance	[86] [87] [88]
UNREAL	Use A3C and multiple unsupervised reward signals to improve learning efficiency in complicated environments	• Unsupervised reward signals	○ Policy updates exhibit high variance	[89]
DDPG	Concurrently learn a deterministic policy and a Q-function in DQN's fashion	• Deterministic policy gradient	○ Support only continuous action space	[87] [88]
TRPO	Limit policy update variance by using the conjugate gradient to estimate the natural gradient policy TRPO is better than DDPG in terms of sample efficiency	• Kullback-Leibler divergence • Conjugate gradient • Natural policy gradient	○ Computationally expensive ○ Large batch of rollouts ○ Hard to implement	[87] [88]
ACKTR	Inherit the A2C method Use Kronecker-Factored approximation to reduce computational complexity of TRPO ACKTR outperforms TRPO and A2C	• Kronecker-factored approximate curvature	○ Still complex	[87]
ACER	Integrate an experience replay into A3C Introduce a light-weight version of TRPO ACER outperforms TRPO and A3C	• Importance weight truncation & bias correction • Efficient TRPO	○ Excessive memory usage ○ Still complex	[87] [88]
PPO	Simplify the implementation of TRPO by using a surrogate objective function Achieve the best performance in continuous control tasks	• Clipped objective • Adaptive KL penalty coefficient	○ Require network tuning	[86] [87] [88]

Figure 3: RL algorithms. [5]

1.1.3 Deep Learning

Deep Learning, is a class of machine learning algorithms, based on artificial neural networks (ANNs) that with the use of multiple layers, tries to progressively extract higher-level features from an input. A common example to reflect how do they work is in image processing. In lower layers maybe edges are identified while in higher layers may be that more human concepts are identified such as faces, numbers or letters. This definition is according to [6].
There are numerous types of deep learning layers, each serving a specific purpose in neural network architectures. Presented below are some of the most important types, along with their respective functions. This information is derived from [7].

- **Convolutional Layer:** Used for feature extraction in convolutional neural networks (CNNs), particularly effective for image processing tasks.
- **Dense Layer:** Found in traditional feedforward neural networks, these layers connect every neuron from the previous layer to every neuron in the current layer, allowing for comprehensive feature learning.
- **Recurrent Layer:** Integral for processing sequential data, such as time series or natural language, by preserving information from previous time steps or words.
- **Pooling Layer:** Reduces the spatial dimensions of feature maps, aiding in translation invariance and computational efficiency in CNNs.
- **Dropout Layer:** Helps prevent overfitting by randomly dropping a proportion of neurons during training, forcing the network to learn more robust representations.
- **Batch Normalization Layer:** Normalizes the activations of each layer to improve training stability and accelerate convergence.

These layers represent fundamental building blocks in deep learning models, each contributing to the network's ability to learn and generalize from data

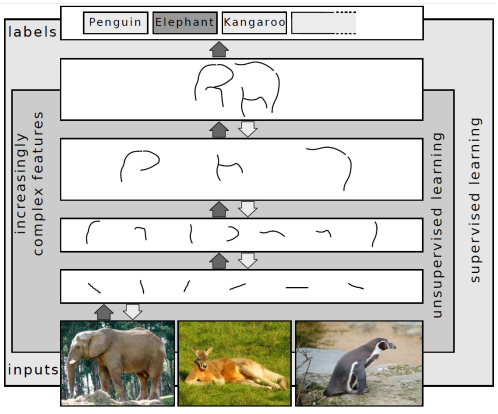

Figure 4: Image representation in multiple layers. [6]

1.2 Problem description

Addressing complexity in NP-hard problems presents a significant challenge. Many real-life problems fall into this category, rendering them intractable for large input sizes. The primary objective

of this thesis is to tackle this issue by proposing novel approaches grounded in reinforcement learning (RL) techniques.

1.3 Actors involved

It's very clear to understand that the thesis is aimed for researches into the artificial intelligence field and more specifically at deep reinforcement learning researchers.They are the ones who will use the result of this project to further expand the state of the art.

The potential beneficiaries are vast, since NP-hard problems encompass a wide array of real-life scenarios. For instance, enhancing heuristics for the Traveling Salesman Problem (TSP) could yield significant benefits across numerous industries reliant on efficient routing. By enabling companies to compute more optimal routes in less time and with fewer resources, this research stands to enhance operational efficiency and resource utilization across various sectors.

1.4 State of the art

The state of the art of this field is quite extensive and a primary objective of this thesis is to comprehensively analyze these existing approaches proposed for this broad problem in order to provide with a better solution. For instance, in [8] a notable solution employing attention layers, a type of deep learning architecture, alongside a Reinforce framework, a specific reinforcement learning technique, showcases promising results across sizable instances of combinatorial optimization problems, which are inherently NP-hard.

2 Justification of the chosen alternative

Traditionally, combinatorial optimization problems have been solved using two distinct manners: with exact methods and heuristic approaches.However, a recent trend has emerged, shifting towards a novel methodology of learning heuristics through models rather than relying solely on manually crafted logic. This alternative differantiation can be found on [3].

2.1 Exact Methods

Exact methods consist on hand-written logic that solve the instance of the problem with optimality. This approach boasts the advantage of guaranteeing the best possible solution for every provided instance, ensuring optimality. However, this advantage comes at the cost of high computational complexity.

The inherent complexity of exact methods presents a notable drawback, particularly evident as the input size of the problem instances increases. As the size of the input grows, the computational demands of exact methods escalate rapidly, often rendering them infeasible for practical application.

2.2 Heuristic approach

The traditional heuristic approach similarly relies on meticulously crafted algorithms tailored to solve problem instances. However, in contrast to exact methods, heuristics prioritize computational efficiency over guaranteeing optimal solutions.

This trade-off allows heuristic methods to deliver solutions that are often close to optimal, while exhibiting significantly lower computational complexity compared to exact methods. As a result, heuristic approaches demonstrate greater scalability with respect to the size of the input instances.

2.3 Model Learned heuristics

This lastly developed approach, as previously mentioned, diverges from manually crafted logic using instead machine learning models. This methodology, precedes the encountered heuristic with a training phase enabling the model to generate nearly instantaneous solutions with a low optimality gap when well-trained.
This innovative approach further enhances computational efficiency, significantly reducing resolution times. However, this efficiency comes at the cost of diminished explainability regarding the inner workings of the heuristic.

2.4 Justification

Reinforcement Learning framework framework represents a key component of the latest alternative proposed and nd serves as the focal point of study in this thesis. Being a relatively recent development, this approach remains relatively underexplored compared to other options. However, despite its nascent status, RL has demonstrated exceptional efficacy in solving complex problems.

As such, the compelling combination of RL's promising results and its relatively unexplored terrain justifies its consideration as a viable alternative for tackling the challenges at hand.

3 Project Scope

In every project, it's important to define the scope of it given the inherent limitations of time and resources. Additionally, it's always welcome to define clearly it's objectives, requirements and possible foreseen obstacles and risks is essential to have a more robust knowledge and strategy to fulfill all the proposed goals.

3.1 Objectives

The main objective of this graduation thesis is to expand the state of the art of the of reinforcement learning applied to NP-hard problem resolution with the intention to provide new knowledge to the scientific community of artificial intelligence. Given the expansive nature of the task and the constraints of a tight timeline, a sequential objective strategy is adopted. This strategy allows for a structured approach to the research, with defined objectives at each stage, while also providing flexibility for potential endpoints along the way:

1. **Investigate, Structure, and Synthesize the Current State of the Art (O1):** This initial phase involves conducting a comprehensive review of existing literature to gain insights into the latest developments and methodologies in reinforcement learning applied to NP-hard problems.
2. **Develop and Evaluate Different RL Frameworks for Solving Single Instances (O2):** In this phase, various reinforcement learning frameworks will be developed and evaluated for their efficacy in solving individual instances of NP-hard problems. These frameworks will

be assessed based on their performance metrics and compared against each other to identify strengths and weaknesses.

3. **Select Optimal Frameworks for Generalization Capacity (O3):** Building upon the findings from the previous stage, the most promising reinforcement learning frameworks will be selected for further adaptation to handle problems with generalization capacity. This involves enhancing the frameworks to enable them to generalize solutions across a broader range of problem instances and evaluate them based on comparison against existing approaches.

4. **Provide explainability on the developed heuristics (O4):** Finally, this phase focuses on gaining a deeper understanding of how the developed heuristics operate with the ambition to provide some kind of explainability to better understand RL frameworks.

3.2 Requirements

In this section, we outline the requirements necessary to develop a new functional framework. These requirements are divided into two categories: Functional Requirements, which detail the capabilities of the framework, and Non-Functional Requirements, which specify additional characteristics and criteria beyond the functional aspects. All the concepts presented in this section are defined and extracted from [7].

3.2.1 Functional Requirements

There are some key aspects that the developed framework must fulfill to be considered functional:

- **Environment Interface (R1):** The framework should provide an interface for interacting with the environment, allowing agents to observe states, take actions, and receive rewards. This interface should facilitate seamless communication between the agent and the environment.

- **Agent Architecture (R2):** The framework should support various types of agent architectures. It should also allow for custom agent architectures tailored to specific problem domains.

- **Training Infrastructure (R3):** The framework should include training infrastructure to facilitate the training of RL agents. This infrastructure may include tools for managing training data, orchestrating training processes, and monitoring agent performance.

- **Exploration and Exploitation Strategies (R4):** The framework should support mechanisms for balancing exploration (trying out different actions to discover optimal strategies) and exploitation (leveraging known strategies to maximize rewards). It should provide options for specifying exploration strategies, such as epsilon-greedy exploration or Boltzmann exploration.

- **Parallelization (R5):** Parallelization is closely related to efficiency and plays a crucial role in optimizing training processes. By leveraging parallel computing resources, the framework can potentially achieve faster and more comprehensive training, leading to improved results.

3.2.2 Non Functional Requirements

In addition to the functional requirements outlined earlier in this section, there are also key characteristics that the developed framework must fulfill:

- **Efficiency (R6):** Efficiency is paramount for improving upon current solutions, as computational complexity is a significant challenge in solving NP-hard problems and reinforcement learning training needs as many interactions as possible to get the results. Enhancing the efficiency of the framework will enable faster and more effective problem-solving.
- **Performance (R7):** The framework should demonstrate high performance in terms of speed, efficiency, and scalability. It should be capable of handling large-scale datasets and training RL agents efficiently.
- **Robustness (R8):** The framework should be robust against noisy or incomplete input data, environmental uncertainties, and adversarial attacks. It should be able to produce reliable results under various conditions and inputs.
- **Scalability (R9):** The framework should scale efficiently with increasing problem sizes, computational resources, and training data. It should support parallelization and distributed computing to leverage multiple processors or GPUs for accelerated training.

3.3 Obstacles and Risks

Reinforcement learning frameworks, while powerful, can encounter various obstacles and risks during development and deployment. Some potential obstacles and risks include:

- **Sample Efficiency:** Reinforcement learning algorithms often require a large number of samples or interactions with the environment to learn effective policies. Limited sample efficiency can prolong training times and hinder the scalability of the framework to real-world applications.

  To address the challenge of limited sample efficiency in RL, various strategies can be employed depending on the algorithm used. Examples of approaches to improve sample efficiency include:

  - Implement experience replay to reuse past experiences during training.
  - Prioritize experiences based on their relevance.
  - Use model-based methods to leverage environment dynamics.
  - Apply transfer learning to utilize knowledge from related tasks.

- **Exploration-Exploitation Tradeoff:** Reinforcement learning frameworks must balance exploration (trying out different actions to discover optimal strategies) and exploitation (leveraging known strategies to maximize rewards). Finding the right balance between exploration and exploitation can be non-trivial and may impact the performance of the framework.

  To manage the exploration-exploitation tradeoff in RL there exist techniques that help strike a balance between exploring new actions and exploiting known strategies to maximize performance:

- Use epsilon-greedy or softmax policies for a balanced approach.
- Apply UCB methods to prioritize actions while considering uncertainty.
- Employ Thompson sampling for probabilistic action selection.
- Adapt multi-armed bandit algorithms for sequential decision-making.

- **Generalization:** Reinforcement learning frameworks may struggle to generalize well to unseen or novel environments or tasks. Generalization limitations can hinder the transferability of learned policies to new scenarios or domains.

  To address the challenge of generalization limitations in RL, several strategies can be employed to mitigate generalization limitations and improve the transferability of learned policies to new environments or tasks:

  - Ensure diverse training data.
  - Apply regularization techniques.
  - Utilize transfer learning.
  - Employ ensemble methods.
  - Introduce domain randomization.

- **Training Stability:** Reinforcement learning training processes can be unstable, especially when using deep neural network architectures. Issues such as vanishing gradients, diverging gradients, or catastrophic forgetting can disrupt training and hinder convergence to optimal solutions.

  To address the challenge of training instability in reinforcement learning, several strategies can be employed to mitigate training instability and improve the convergence of training processes towards optimal solutions:

  - Use gradient clipping to prevent exploding gradients.
  - Apply batch normalization to stabilize training.
  - Implement learning rate scheduling for smoother convergence.
  - Utilize appropriate weight initialization techniques.
  - Employ experience replay to mitigate the impact of rare events.

- **Resource Constraints:** Reinforcement learning frameworks often require significant computational resources, including high-performance computing infrastructure and large-scale datasets. Resource constraints may limit the scalability and accessibility of the framework, particularly in resource-constrained environments. In the context of RL frameworks, overcoming this challenge typically entails substantial expenses.

All the concepts presented in this section are defined and extracted from [7].

4 Methodology

This project follows a structured approach consisting of an investigation phase, followed by brainstorming sessions to generate new ideas for implementation, a development phase, and finally, an evaluation phase. This cyclical process is designed to continuously iterate and improve upon the project's outcomes over time. This methodology can be referred to as a "Cyclic Waterfall" approach, where each cycle encompasses the stages of investigation, ideation, development, and evaluation, facilitating ongoing refinement and progress.

4.1 Validation

This process will be complemented by weekly meetings, fostering open communication of project developments and advancements. During these meetings, updates and progress are going to be shared, made since the previous meeting. Additionally, each meeting will include the proposal of a set of goals to be achieved before the subsequent meeting.

During the development phases, regular validation sessions will be conducted to ensure the integrity and accuracy of the development process within the framework. These sessions will serve as checkpoints to verify that the implementation aligns with project requirements and objectives. By incorporating these validation sessions into the development workflow, the team can identify and address any issues or discrepancies early on, ultimately enhancing the quality and effectiveness of the framework.

References

[1] Wikipedia. *NP-hardness*. `https://en.wikipedia.org/wiki/NP-hardness`. Accessed on February 21, 2024.

[2] Yunhao Yang and Andrew Whinston. "A survey on reinforcement learning for combinatorial optimization". In: *2023 IEEE World Conference on Applied Intelligence and Computing (AIC)*. IEEE. 2023, pp. 131–136.

[3] Leo Ardon. "Reinforcement Learning to Solve NP-hard Problems: an Application to the CVRP". In: *arXiv preprint arXiv:2201.05393* (2022).

[4] Wikipedia. *Reinforcement Learning*. `https://en.wikipedia.org/wiki/Reinforcement_learning`. Accessed on February 21, 2024.

[5] Ngoc Duy Nguyen et al. "Review, analysis and design of a comprehensive deep reinforcement learning framework". In: *arXiv preprint arXiv:2002.11883* (2020).

[6] Wikipedia. *Deep Learning*. `https://en.wikipedia.org/wiki/Deep_learning`. Accessed on February 21, 2024.

[7] Dr. J.W. Böhmer and Dr. M.T.J. Spaan. *Deep Reinforcement Learning (CS4400) Course Notes*. University of TU Delft. Unpublished course notes. 2024.

[8] Wouter Kool, Herke Van Hoof, and Max Welling. "Attention, learn to solve routing problems!" In: *arXiv preprint arXiv:1803.08475* (2018).