SOLVING OPTIMIZATION PROBLEMS WITH REINFORCEMENT LEARNING

BOULET Timothé, SELVESTREL Alexandre, PREVOT-HELLOCO Pierre
CentraleSupélec, course Reinforcement Learning

{timothe.boulet, alexandre.selvestrel, pierre.prevothelloco}@student-cs.fr

Introduction

Combinatorial optimization problems are widely encountered in various domains. However, traditional methods such as Linear Programming face limitations for non- linear constrained problems, and cannot tackle problems where the objective function is nonlinear.[1pt]

In this project, we studied how RL algorithms can be applied to solve such optimization problems formalized as tabular environments. Our RL algorithms are designed within a framework where they can be readily applied to any new environment.

Environments

We examined three renowned optimization problems: the knapsack problem, the bin packing problem, and the Facility Location Problem. These three dilemmas represent classic examples of combinatorial optimization problems.

Knapsack: pick objects to maximize the value with a capacity constraint

- **state:** which objects are in the bag : $(1_{\text{object i in the bag}})_{i=1}^{n_{\text{objects}}}$
- **action:** pick which remaining object to put in the bag (without exceeding the capacity)
- **reward:** the value of the object put in the bag

Facility Location Problem (FLP): assign $n_{\text{facilities}} \leq n_{\text{sites}}$ to minimize the sum of distances between clients and their nearest facility

- **state:** which sites have a facility : $(1_{\text{site i has a facility}})_{i=1}^{n_{\text{sites}}}$
- **action:** pick the site of the t-th facility
- **reward:** the reduction in cost at step t compared to previous step $t-1$

Bin Packing

- **state:** the remaning space in each bin : $(\text{remaining space in bin i})_{i=1}^{n_{\text{bins}}}$
- **action:** which bin to put the next object in, or create a new bin
- **reward:** -1 if a new bin is created, 0 otherwise

Results

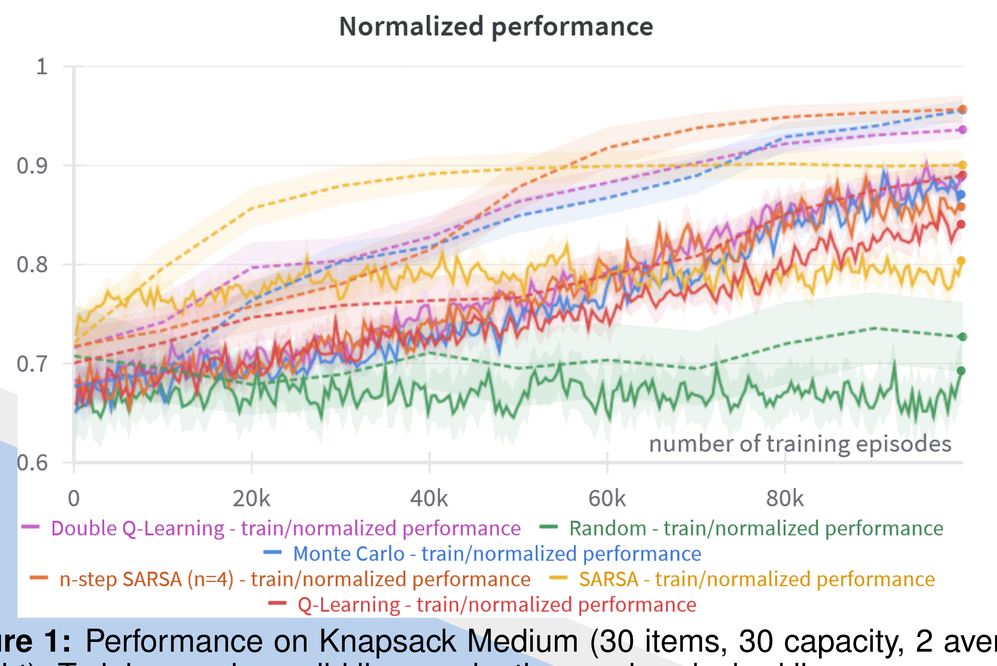

Figure 1: Performance on Knapsack Medium (30 items, 30 capacity, 2 average weight). Training mode : solid line, evaluation mode : dashed line

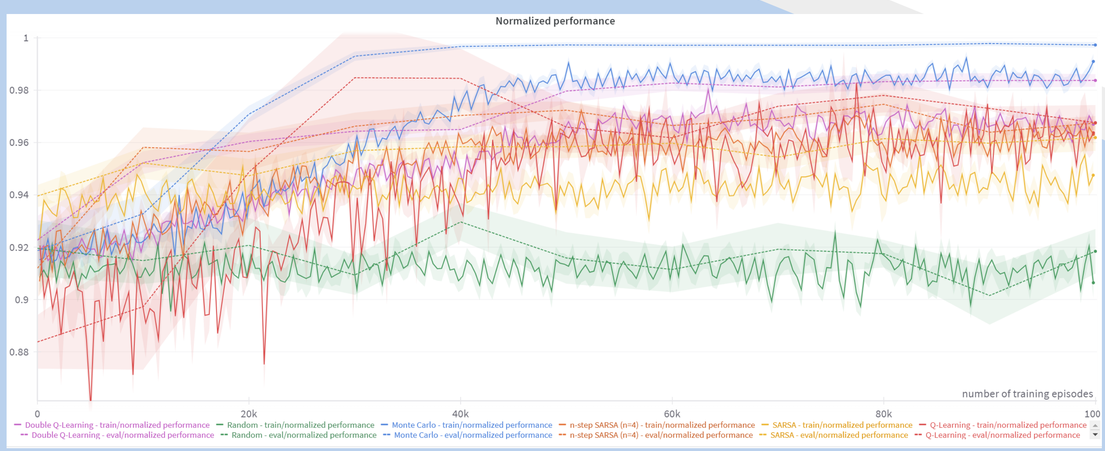

Figure 2: Performance on FLP Medium (10 facilities, 25 facility sites, 50 customer sites). Training mode : solid line, evaluation mode : dashed line

Normalized performance: $\dfrac{\text{return} - \text{worst return}}{\text{optimal return} - \text{worst return}}$

Worst return : a very bad solution or a lower bound on the optimal solution, e.g. $-n_{\text{objects}}$ for Bin Packing.

Optimal return : the solution found by Linear Programming, or a known optimal solution (by construction).

Reward densification

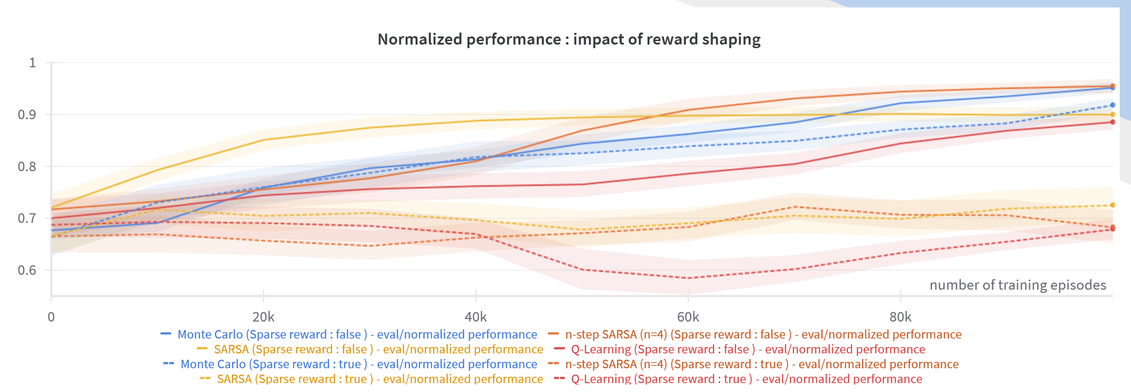

Figure 3: Performance with dense reward vs. sparse reward

Densification of the reward has a decisive role for the learning of the agent. Gain in performance is less significant in the case of Monte Carlo methods, as this doesn't change their target a lot.

Impact of the size of the instance

Figure 4: Performance on Knapsack environments of different instances : 10 items (solid line), 30 items (dashed) and 100 items (dotted)

Performance becomes poor when the size of the environment grows too large. The number of possible states increases exponentially with the size of the instance, making exploration slower as the number of states increases. The use of Deep Reinforcement learning would probably help to improve performance for the largest instances.

Explorative policies

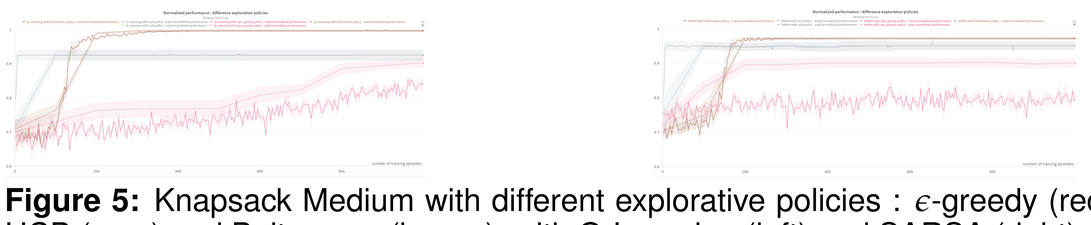

Figure 5: Knapsack Medium with different explorative policies : ϵ-greedy (red), UCB (grey) and Boltzmann (brown), with Q-Learning (left) and SARSA (right)

We observe that the explorative policy has a significant impact on the performance of the agent. UCB converges very fast to a local optimum, and Boltzmann is the most efficient policy for the knapsack problem.

Future work

- Implement Deep RL agents capable of generalizing across several instances of different sizes and characteristics.
- Use expert knowledge to design better policies to help exploration, e.g. greedy algorithm for Knapsack, or Best Fit algorithm for Bin Packing

References

[1] O. Rivlin. Reinforcement learning for combinatorial optimization, 2019.