A SOFTWARE FRAMEWORK FOR SOLVING COMBINATORIAL OPTIMIZATION TASKS

ISTVAN-GERGELY CZIBULA, GABRIELA CZIBULA AND MARIA-IULIANA BOCICOR

ABSTRACT. Due to the major practical importance of combinatorial optimization problems, many approaches for tackling them have been developed. As the problem of intelligent solution generation can be approached with reinforcement learning techniques, we aim at presenting in this paper a programming interface for solving combinatorial optimization problems using reinforcement learning techniques. The advantages of the proposed framework are emphasized, highlighting the potential of using reinforcement learning for solving optimization tasks. An experiment for solving the bidimensional protein folding problem developed using the designed interface is also presented.

1. INTRODUCTION

Combinatorial Optimization (CO) problems have a major practical importance. All these problems are searching for one or more optimal solutions in a well defined discrete problem space. The current real-life combinatorial optimization problems (COPs) are difficult in many ways: the solution space is huge, the parameters are linked, the decomposability is not obvious, the restrictions are hard to test, the local optimal solutions are many and hard to locate, and the uncertainty and the dynamicity of the environment must be taken into account.

Due to the importance of CO problems, many algorithms to tackle them have been developed [3]. Incremental search algorithms use past experience to form feasible solutions. The “goodness” (optimality) of the resultant solution is the only type of feedback available in many cases [8]. Since we expect the system to learn based on its past experience and generate better solutions over
time using only this limited information, the problem of intelligent solution
generation can be approached with reinforcement learning (RL) [10].

Reinforcement Learning (RL) is an approach to machine intelligence in
which an agent can learn to behave in a certain way by receiving punishments
or rewards on its chosen actions [10].

The aim of the approach proposed in this paper is to make an abstraction of
the issue of solving combinatorial optimization problems using reinforcement
learning techniques. In this direction we propose a programming interface
and we develop, using the proposed framework, an application for solving the
bidimensional protein folding problem using reinforcement learning.

The rest of the paper is structured as follows. Our RL based interface
proposal is presented in Section 2. An experiment for solving the bidimensional
protein folding problem developed using the proposed interface is reported in
Section 3. Conclusions and further work are outlined in Section 4.

2. The RL based framework

The idea of using RL in optimization problem solving has appeared before
[8], and several approaches were developed for solving particular optimization
problems [13, 8]. We have previously introduced in [4, 5] a reinforcement
learning based model for solving a well known optimization problem within
bioinformatics, the protein folding problem, which is an NP-complete problem
[2] that refers to predicting the structure of a protein from its amino acid
sequence. Protein structure prediction is one of the most important goals
pursued by bioinformatics and theoretical chemistry; it is highly important in
medicine (for example, in drug design) and biotechnology (for example, in the
design of novel enzymes).

In this section we propose an API that allows to simply develop appli-
cations for solving combinatorial optimization problems using reinforcement
learning techniques. In the framework that we propose we make an abstraction
of the way the optimization problem to be solved is modeled as a reinforcement
learning task [10]. This is the major advantage of our RL interface proposal:
the reinforcement learning algorithm is defined independent of the way the
environment, states and actions are defined.

The interface is realized in JDK 1.6 and has four basic modules: agent, environ-
ment, reinforcement learning, and simulation. As in a general agent
based system [12], the agent is the entity that interacts with the environment,
that receives perceptions and selects actions. The agent learns using rein-
forcement learning to achieve its goal, i.e to find an optimal solution of the
corresponding optimization problem. Generally, the inputs of the agent are
perceptions about the environment (in our case states from the environment),
the outputs are actions, and the environment offers rewards after interacting with it. The interaction between the agent and the environment is controlled by a simulation entity.

In the following, we will briefly describe the responsibility of the main elements from the programming interface that we introduce for solving combinatorial optimization problems using reinforcement learning.

**Agent.** The agent is the entity that interacts with the environment, receives perceptions (states) from it and selects actions. The agent learns by reinforcement and could have or not a model of the environment. It is the basic class for all the agents. The specific agents will implement the Agent interface. The main responsibility of the Agent class is to select the most appropriate action it has to perform in the environment.

**Environment.** The environment basically defines the optimization problem to solve. In our approach, the environment will have an explicit representation as a space of states. It is the basic class for all environments. The specific environments will implement the Environment interface. The Environment has a function that determines the environment to make a transition from a state to another, after executing a specific action. This function also gives the reward obtained after the transition. The environment stores an instance of its current state.

**Reinforcement Learning.** Is the class that is responsible with the reinforcement learning process. The framework provides implementations for $Q$−learning, SARSA and SARSA(λ). λ refers to the use of an eligibility trace [9] for obtaining a more general and efficient learning method. The EligibilityTrace class is responsible with managing eligibility traces.

**Simulation.** Is the object that manages the interaction between the agent and the environment. An instance of the simulation class is associated with an instance of an agent and an environment at the creation moment. The simulation object is responsible with collecting data, managing the learning process and providing the optimal policy that the agent has learned.

Figure 1 shows a simplified UML diagram [6] of the interface, illustrating the core of the RL interface. It is important to mention that all the classes provided by the interface remain unchanged in all applications for solving combinatorial optimization problems using reinforcement learning. What is outside the core are reference implementations.

As a conclusion, we summarize the main advantages of using the framework proposed in this paper:

- Provides an easy way to model optimization problem solving as a reinforcement learning problem.
• The effort for developing an application for solving optimization problems using reinforcement learning is reduced – we need to define only a few classes. The framework offers default implementations for most of the RL related algorithms, the user only needs to define the classes that describe the concrete optimization problem (classes that implement the State, Environment and Action interfaces).

• In a scenario of solving an optimization problem using reinforcement learning, the proposed interface simply allows experimentation with different conceptual models for the states, the actions, different RL algorithms, different reinforcement functions.

3. Experiment

In order to experimentally evaluate the proposed framework, we are addressing in the following the Bidimensional Protein Folding Problem (BPFP), more exactly the problem of predicting the bidimensional structure of proteins, a well known optimization problem within bioinformatics. In the proposed experiment, we use the reinforcement learning model for solving BPFP that have been previously introduced in [4].

Let us consider a HP protein sequence \( P = \text{HHPH} \), consisting of four amino acids. The aim is to find a configuration of \( P \) whose energy [1] is minimum. The state space of the RL model consists of 85 states, and the action space consists of four actions available to the problem solving agent.
and corresponding to the four possible directions $L$\textit{eft}, $U$p, $R$\textit{ight}, $D$\textit{OWN} used to encode a solution [4, 5].

In order to use the interface proposed in Section 2 for solving the above mentioned problem, we have defined specialized classes for which we executed the simulation ($\textit{BPFS}\textit{tate}$, $\textit{BPFE}\textit{nvironment}$ and $\textit{BPFA}\textit{ction}$). We have trained the $\textit{BPF}$ agent using the $Q$-learning algorithm. As proven in [11], the $Q$-learning algorithm converges to the real $Q$-values as long as all state-action pairs are visited an infinite number of times, the learning rate $\alpha$ is small (e.g. 0.01) and the policy converges in the limit to the $\textit{Greedy}$ policy. We remark the following regarding the parameters setting: the learning rate is $\alpha = 0.01$ in order to assure the convergence of the algorithm; the discount factor for the future rewards is $\gamma = 0.9$; the number of training episodes is 64; the $\epsilon$-Greedy action selection mechanism was used.

Using the above defined parameters and under the assumptions that the state action pairs are equally visited during training and that the agent explores its search space (the $\epsilon$ parameter is set to 1), the solution reported after the training of the $\textit{BFP}$ agent was completed is the path $\pi = (s_1s_2s_7s_8)$ having the associated configuration $a_\pi = (LUR)$, determined starting from state $s_1$, following the $\textit{Greedy}$ policy. The solution learned by the agent has an energy of $-1$. Consequently, the $\textit{BPF}$ agent learns the optimal solution of the bidimensional protein folding problem, i.e. the bidimensional structure of the protein $\mathcal{P}$ that has a minimum associated energy ($-1$).

Using the framework proposed in this paper we can simply select other conceptual models for the states space and the actions space within the RL scenario of solving the bidimensional protein folding problem. For example, we can think at the states space as the set of possible solutions of the $\textit{BPFP}$ and at the action space as the set of possible pull move transformations [7].

4. Conclusions and Further Work

We have introduced in this paper a framework that facilitates research in the direction of solving combinatorial optimization problems using reinforcement learning techniques. We have emphasized the advantages of the proposed framework, highlighting the potential of using reinforcement learning for solving optimization tasks.

Further work will be done in order to apply the framework for other optimization problems, to extend the evaluation of the proposed framework for larger bidimensional protein folding problem benchmarks, and to investigate other conceptual models for the states space and the actions space within the RL scenario of solving the $\textit{BPFP}$. 
ACKNOWLEDGEMENT

This work was supported by CNCSIS - UEFISCDI, project number PNII - IDEI 2286/2008.

REFERENCES