A State Aggregation Approach for Solving Knapsack Problem with Deep Reinforcement Learning

Reza Refaei Afshar, Yingqian Zhang, Murat Firat, Uzay Kaymak

Introduction

In this work, we aim to learn and improve the handcrafted heuristics to improve the quality of the solutions. We study *knapsack problem* (KP), which is one of the well-known benchmark problems in COPs. KP is defined as selecting some items from a given set such that the selected items fit in the knapsack and their total value is maximized. We propose a state aggregation method to discretize the feature values of items. A tabular reinforcement learning is used to learn the best aggregation strategy for each item. This discretized features not only provide a discrete representation of the problem instances, but also reduces the state space by reducing the number of unique values. Since even the reduced state space after applying the state aggregation is still large, Deep Reinforcement Learning (DRL) is employed as a powerful function approximation method. We use *Advantage Actor Critic* (A2C) algorithm to learn the policy of selecting items. A2C makes use of two DNNs for learning policy and value functions. The policy DNN has an output size that is equal to the number of items in the KP instance. The proposed method greedily solves the KP problem by successive item selections and placing them in the knapsack, each is done by following a greedy or Softmax algorithm on the output of the policy DNN.

DRL-based KP Solver

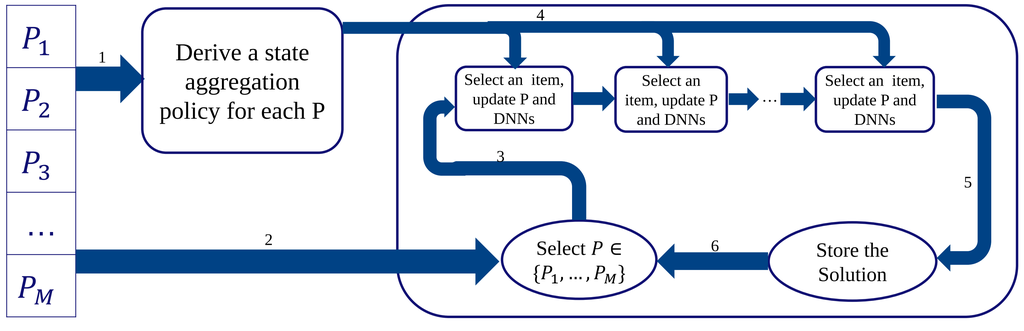

Figure 1. The overview of the KP solver. 1) A set of instances are used for deriving an aggregation policy. 2) The same set of instances are used with DRL. 3) A KP instance is selected for training. 4) Items are selected sequentially until finding a solution. At each step, the updated P is aggregated to find the state. A2C updates the parameters of DNNs. 5) The best solution is stored. 6) Another KP instance is selected and the process continues for a certain number of timesteps.

State Aggregation

Step 1: Define a table of problem instances where each row and column correspond to an instance and a value or a weight of an item, respectively.

	v_1	w_1	v_2	w_2	v_3	w_3	v_4	w_4
Instance 1								
…								
Instance M								

Step 2: Transform the values and the weights to normalized value and weight and sort the column descendingly based on normalized values.
Step 3: Define appropriate number of bins for each columns and map all the values of each column to the label of the bin.
Tabular reinforcement learning is used. States are columns and actions are the number of bins for each column.

Experiments and Results

3 types of problem instances: Random Instances (RI), Fixed Capacity Instances (FI) and Hard Instances (HI).

Type	n_P	Method	Average Solution value	Optimally solved instances	Average Optimal value	Bello et al 2017
HI	500	Greedy	80779.23	25	81103.99	-
HI	500	DRL w/ A	**81064.99**	**136**	**81103.99**	-
HI	500	DRL w/o A	81022.60	71	81103.99	-
FI	50	Greedy	20.10	172	20.15	20.15
FI	50	DRL w/ A	**20.15**	**773**	**20.15**	20.15
FI	50	DRL w/o A	20.14	740	20.15	20.15
FI	500	Greedy	111.68	204	111.73	-
FI	500	DRL w/ A	**111.70**	**261**	**111.73**	-
FI	500	DRL w/o A	111.63	64	111.73	-

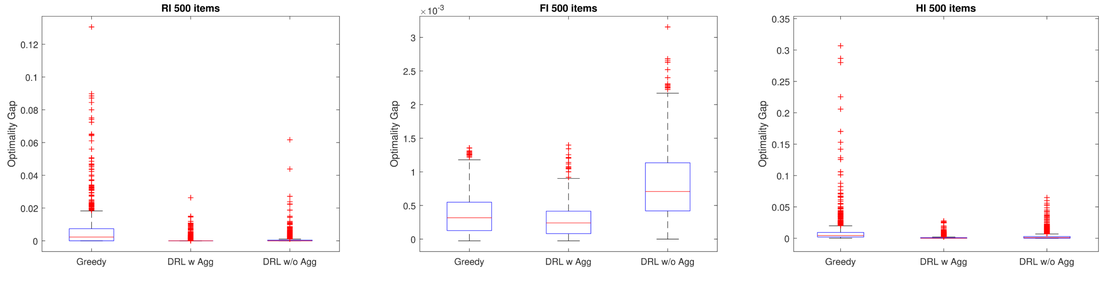

Figure 2. The box plot for the optimality gaps of solutions of the three instances types with 300 and 500 items

Department of Industrial Engineering and Innovation Sciences – Information Systems Group