Markovian Resource Control

We showed that stochastic resource allocation problems can be reformulated as *Markov decision processes*, more precisely, as *stochastic shortest path* problems, with favorable properties, e.g., all policies are proper.

An *adaptive sampling* based method was applied, by which the optimal value function is iteratively approximated. The updates are performed at the end of each episode (a simulation of the resource allocation process).

### Sampling and Regression Based Reinforcement Learning

1. Initialize $Q_t$, $L_t$, $\tau$ and let $i = 1$.
2. Repeat (for each episode)
   1. Set $\pi$, to a soft and semi-greedy policy w.r.t. $Q_{t-1}$, e.g.,
      
      $\pi(x,a) = \text{exp}(Q_{t-1}(x,a)/\tau)/\sum_{a'}\text{exp}(Q_{t-1}(x,a')/\tau)$
   2. Simulate a state-action trajectory from $x_i$ using policy $\pi$.
   3. For $j = 0$ to 1 (for each state-action pair in the episode) do
      1. Determine the features of the state-action pair, $\phi' = h(x'_i,a'_i)$.
      2. Compute the new action-value estimation for $x'_i$ and $a'_i$, e.g.,
         
         $\hat{Q}(x'_i,a'_i) = (1-\gamma)Q_{t-1}(x'_i,a'_i) + \gamma \left[ g(x'_i,a'_i) + \alpha \min_{a'' \in A} Q_{t-1}(x''_i,b'_i) \right]$
      3. End loop (end of state-action processing)
   4. Update sample set $L_{t-1}$, with \{ $(\phi'_j, \phi'_i) ; j = 1, \ldots, t_i$ \}. the result is $L_t$.
   5. Calculate $Q_t$ by fitting a smooth regression function to the sample of $L_t$.
   6. Increase the episode number $t$, and decrease the temperature $\tau$.
   7. Until some terminating conditions are met, e.g., $t$ reaches a limit or the estimated approximation error to $Q_t$ gets sufficiently small.

Output: the action-value function $Q_t$ for $\phi(x)$, e.g., the greedy policy w.r.t. $Q_t$.

The notations of the pseudo-code are as follows: variable $i$ is the episode number, $t_i$ is the length of episode $i$ and $j$ is a parameter for time-steps inside an episode. The Boltzmann temperature is denoted by $\tau$, $\pi_t$ is the control policy applied in episode $i$ and $\pi_0$ is the initial state. State $x'_i$ and action $a'_i$ correspond to step $j$ in episode $i$. Function $h$ computes features for state-action pairs while $\gamma_t$ denotes learning rates. Finally, $L_t$ denotes the regression sample and $Q_t$ is the fitted (state-action) value function in episode $i$.

### Additional Improvements

We applied several additional improvements to the core solution method:

- **Action space decomposition**: in order to decrease the available actions in the states, the action space was decomposed (figure on the right side).
- **Rollout methods**: to guide the initial exploration and to gather samples for the regression, limited-lookahead rollout algorithms were applied.
- **Clustering**: in order to divide the problem into several smaller subproblems, we clustered the tasks according to their expected slack times.
- **Distributed sampling**: we can exploit having more than one processors.

### Experiments & Conclusions

The method has proven to be very efficient on benchmark and industrial problems.

- It outperformed most previous algorithms on hard job-shop problems (top).
- It showed excellent performance on a simulated industrial problem (middle).
- Clustering not only resulted in speedup but also in performance gains (down).

Advantages over previous approaches:

- It is *general*, since it is applicable to a large class of resource control problems.
- It takes *uncertainties* into account.
- The method can also quickly *adapt* to disturbances and environmental changes.
- There are theoretical guarantees of approximating the global optimal solution.
- It *scales-well* with the size of the problem without dramatic performance losses.
- It can *use domain specific knowledge*, as well (e.g., in the rollouts or explorations).
- It is an iterative, *any-time* solution.