Distributed Value Functions

Jeff Schneider
Carnegie Mellon University
Pittsburgh, PA USA
schneide@cs.cmu.edu

Weng-Keen Wong
Carnegie Mellon University
Pittsburgh, PA USA
wkw@cs.cmu.edu

Andrew Moore
Carnegie Mellon University
Pittsburgh, PA USA
awm@cs.cmu.edu

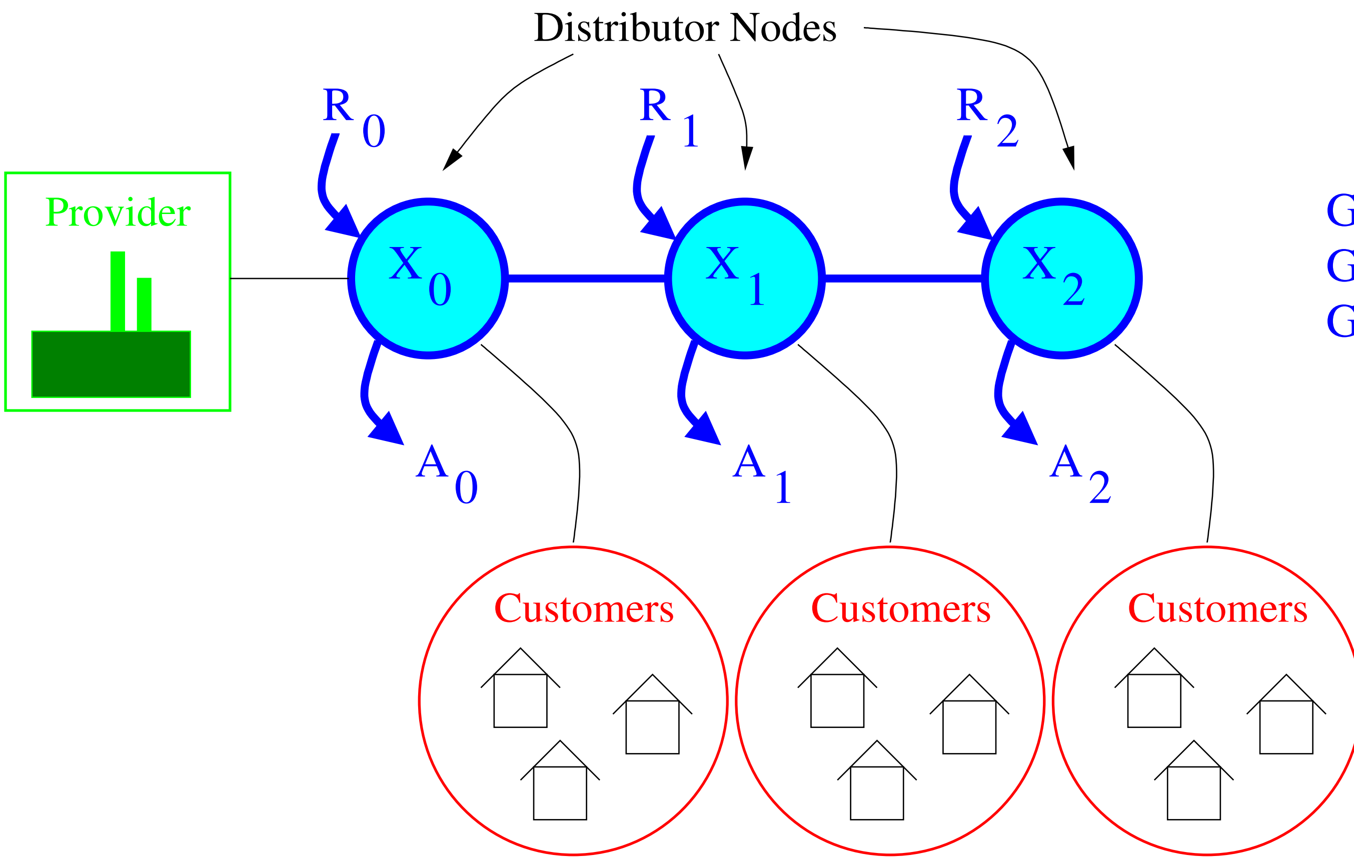
Martin Riedmiller
University of Karlsruhe
Karlsruhe, FRG
riedml@ira.uka.de

Abstract

Many interesting problems, such as power grids, network switches, and traffic flow, that are candidates for solving with reinforcement learning (RL), also have properties that make distributed solutions desirable. We propose an algorithm for distributed reinforcement learning based on distributing the representation of the value function across nodes. Each node in the system only has the ability to sense state locally, choose actions locally, and receive reward locally (the goal of the system is to maximize the sum of the rewards over all nodes and over all time). However each node is allowed to give its neighbors the current estimate of its value function for the states it passes through. We present a value function learning rule, using that information, that allows each node to learn a value function that is an estimate of a weighted sum of future rewards for all the nodes in the network. With this representation, each node can choose actions to improve the performance of the overall system.

We demonstrate our algorithm on the distributed control of a simulated power grid. We compare it against other methods including: use of a global reward signal, nodes that act locally with no communication, and nodes that share rewards (but not value function) information with each other. Our results show that the distributed value function algorithm outperforms the others, and we conclude with an analysis of what problems are best suited for distributed value functions and the new research directions opened up by this work.

Example Problem



Global State - X
Global Action - A
Global Reward - R

Possible Approaches/Related Work

- Lump everything together into a single, global problem
  - problem size
  - communication

- Assume the dynamics and rewards depend only on the local state and local action. Each node operates independently
  - fine if the assumptions hold

- As above, except the action choices of one node affect the available actions of another [Singh & Cohn 98]

- Shared global state and cost information, but each agent acts independently
  - elevator scheduling [Crites & Barto 96]
  - competing rather than cooperating agents [Littman 94]

- All agents try to learn the same value function and speed up learning by sharing information [Tan 93]

Possible Approaches/Related Work (continued)

- Competing “behaviors” and credit assignment
  - bucket brigade with an auction [Weiss 95]
  - combining behaviors to achieve multiple goals [Asada et. al. 94]
  - credit assignment based on economic models [Baum 98]

- Distributed packet routing problems [Boyan & Littman 94], [Choi & Yeung 96], [Subramanian et. al. 97]
  - local communication between nodes
  - actions impact the state and performance of other nodes
  - completely distributed approach

Global Reinforcement Learning

$$V(x) = \max_{a \in A} \left(R(x, a) + \gamma \sum_{x' \in X} p(x' | a, x) V(x') \right)$$

$$V^*(x) = E\left(\sum_{t=0}^{\infty} \gamma^t R\left(x_t, a_t \right) \right)$$

Distributed Reinforcement Learning (DRL)

ignore the issue of locally observable state for now .

Global Reward DRL

$$V_i(x) = \max_{a \in A_i}\left(R(x, a) + \gamma \sum_{x' \in X} p(x' | a, x) V(x') \right)$$

$$V_i^*(x) = E\left(\sum_{t=0}^{\infty} \gamma^t R\left(x_t, a_t\right)\right)$$

Local DRL

$$V_i(x) = \max_{a \in A_i}\left(R_i(x, a) + \gamma \sum_{x' \in X} p(x' | a, x) V_i(x') \right)$$

$$V_i(x) = E\left(\sum_{t=0}^{\infty} \gamma^t R_i\left(x_t, a_t \right) \right)$$

Distributed reward DRL

$$V_i(x) = \max_{a \in A_i}\left(\sum_j f(i,j)\,R_j(x,a) + \gamma \sum_{x' \in X} p(x'|a,x)\,V_i(x')\right)$$

$$V_i(x) = E\left(\sum_{t=0}^{\infty} \gamma^t \sum_j f(i,j)\,R_j\left(x_t, a_t\right)\right)$$

Distributed value function DRL

$$V_i(x) = \max_{a \in A_i}\left(R_i(x, a) + \gamma \sum_j f(i, j) \sum_{x' \in X} p(x' | a, x) V_i(x') \right)$$

$$V_i(x) = \sum_j f(i, j) R_j\left(x_t, a_{j, t} \right)$$

Handling Locally Observable State

- Locally observable state is like using function approximators to represent the value function, except that each node uses a different set of features for its function approximator .
- A practical problem: how are the updates computed?
- Since neighboring nodes do not share state information, there is no way for them to communicate about possible outcomes as in a standard value iteration-based solution.

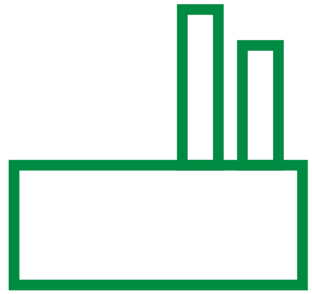
So we use a Q-learning style rule instead:

$$Q_i\left(x_i, a_i\right) \leftarrow (1-\alpha)\, Q_i\left(x_i, a_i\right) + \alpha\left(R_i\left(x_i, a_i\right) + \gamma \sum_j f(i,j)\, V_j\left(x_j'\right)\right)$$

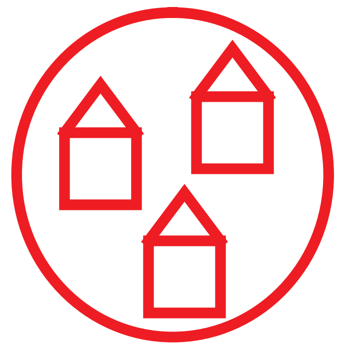
$$V_i\left(x_i\right) \leftarrow \max_{a \in A_i} Q_i\left(x_i, a_i\right)$$

Experimental Setup

A direct current simulation

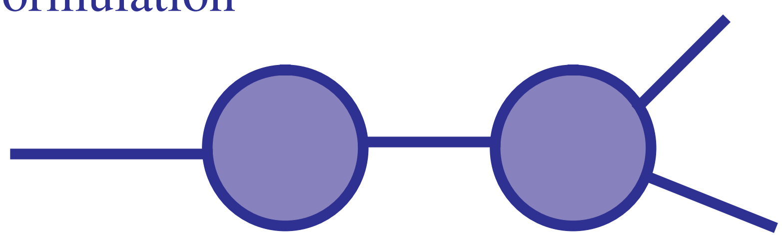


Providers - power generation facilities
fixed voltage source



Consumers - a desired voltage based on
a fixed resistance load

Distributors - the physical connectors between providers and consumers as well as the logical "node" in the DRL formulation



Distributors

Local actions - each node chooses (for each link) whether it would like to double, halve, or leave the resistance unchanged. An arbitration scheme makes the final change for each resistor .

Local state -

distributor/distributor - 18 possible states

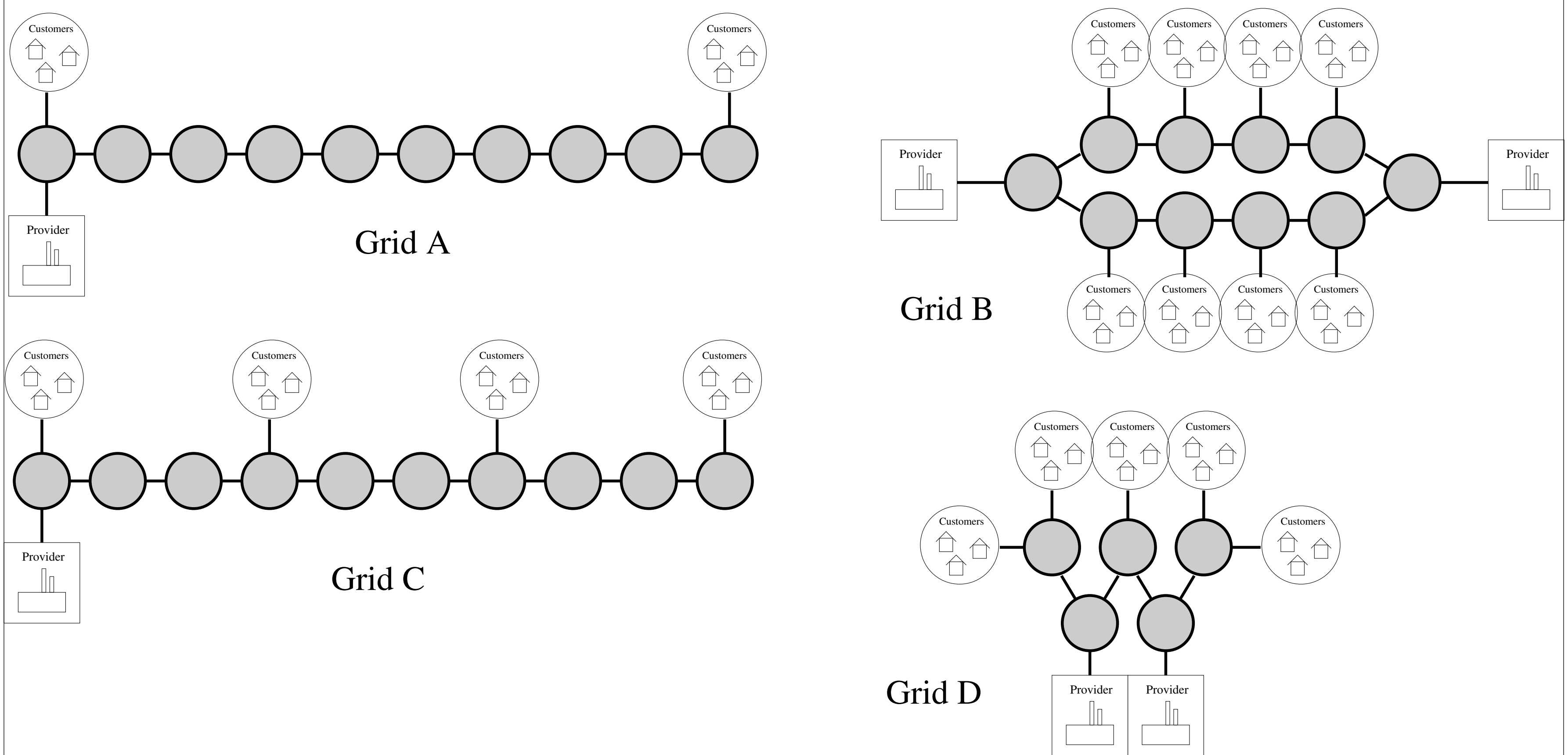
1. neighbor voltage is higher or not (2 states)
2. neighbor voltage decreased, increased, stayed same (3 states)
3. resistor is at min, max, or in between (3 states)

distributor/city - 12 possible states; 1 and 3 from above and whether city needs more power

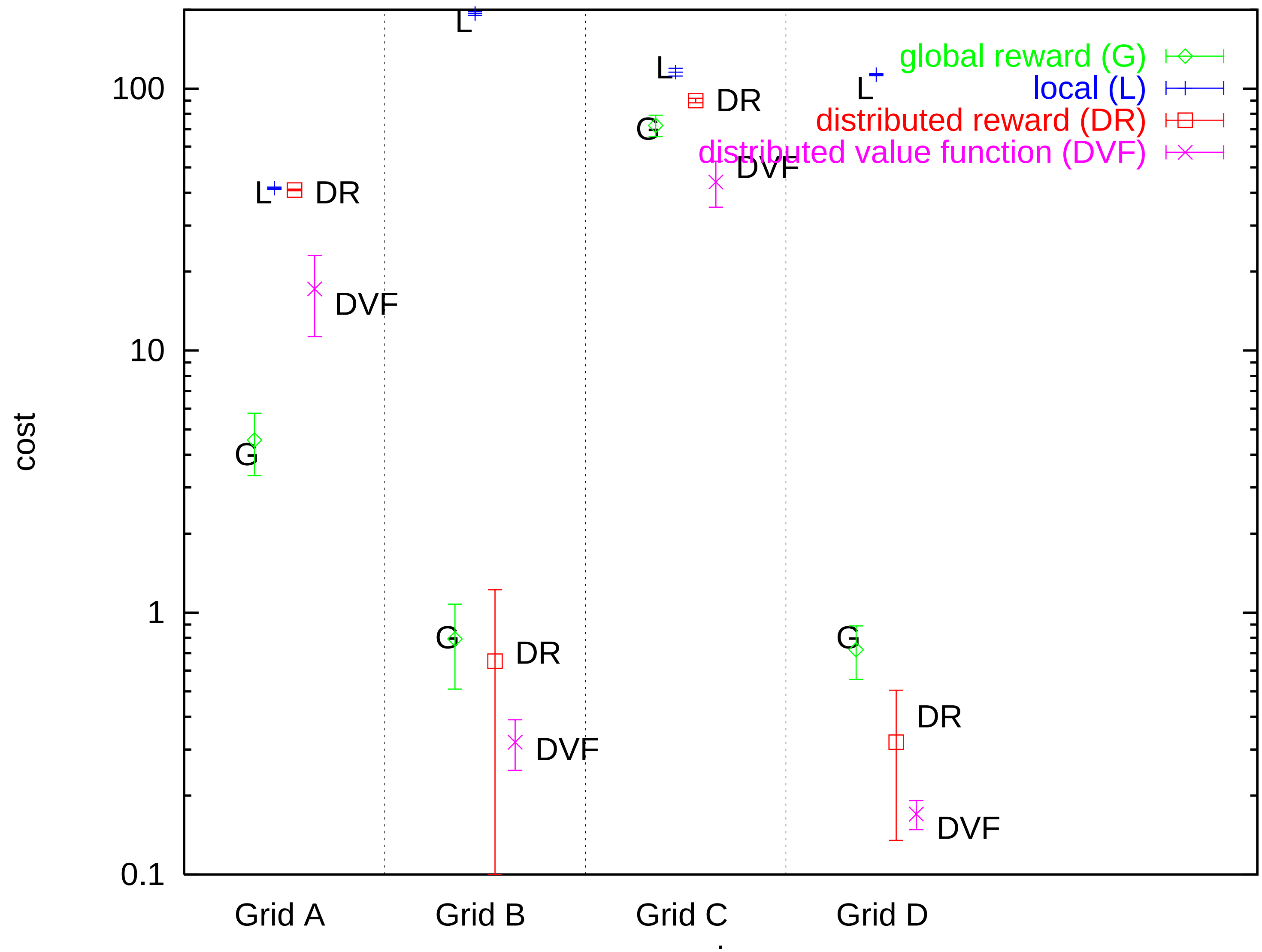
distributor/provider - 6 possible states; 1 and 3 from above

Local reward - a distributor receives a penalty equal to the difference between the desired and actual voltage of the customers it is attached to if the actual is too low , and zero otherwise.

Power Grids for Experimental Results



Experimental Results



Future W ork

- **Determining the neighbor function f(i,j)**
  - distributed value function can imitate global and local reward, but not distributed reward
  - scheduling the weighting function
- **Theoretical questions**
  - with global state, the distributed system can be converted into a centralized system with node as part fo the state. can any theoretical statements be made from this formulation?
- **More sophisticated power grids**
  - A.C. power
  - time-varying supply and demand

Dynamic Neighbor Weighting Functions

Agents have actions that dynamically connect them with other agents and cause value function information to be exchanged.

Can this lead to cooperation and team forming?