A fast point-based algorithm for POMDPs

Nikos Vlassis Matthijs T. J. Spaan
Informatics Institute, Faculty of Science, University of Amsterdam
Kruislaan 403, 1098 SJ Amsterdam, The Netherlands*
{*vlassis,mtjspaan*}*@science.uva.nl*

Abstract

We describe a point-based approximate value iteration algorithm for partially observable Markov decision processes. The algorithm performs value function updates ensuring that in each iteration the new value function is an upper bound to the previous value function, as estimated on a sampled set of belief points. A randomized belief-point selection scheme allows for fast update steps. Results indicate that the proposed algorithm achieves competitive performance, both in terms of solution quality as well as speed.

1 Introduction

Partially observable Markov decision processes (POMDPs) provide a rich mathematical framework for agent planning under uncertainty, with many applications in operations research and artificial intelligence (Sondik, 1971; Kaelbling et al., 1998). Unfortunately, the expressiveness of POMDPs is counterbalanced by the high cost of computing exact solutions (Papadimitriou and Tsitsiklis, 1987; Madani et al., 1999). Approximate (heuristic) solution techniques in POMDPs aim at reducing this complexity (Hauskrecht, 2000).

A classical method for solving POMDPs is value iteration (Sondik, 1971). This method incrementally builds a sequence of value function estimates that converge to the optimal—i.e., highest attainable—value function for the current task. Traditional exact value iteration algorithms (Sondik, 1971; Cheng, 1988; Kaelbling et al., 1998) search in each value iteration step the complete belief simplex for a minimal set of belief points that generate the necessary set of vectors for the new horizon value function. This typically requires solving a number of linear programs and is therefore costly in high dimensions.

In (Zhang and Zhang, 2001) it was shown that value iteration still converges to the optimal value function if the computed value function in each step is an upper bound to the value function of the previous step. Standard value iteration steps can then be interleaved with such 'partial' value iteration steps, resulting in a speedup of the total algorithm. However, linear programming is again needed in order to ensure that the new value function is an upper bound to the previous one.

On the other hand, in practical tasks one would like to compute solutions only for those parts of the belief simplex that are reachable, i.e., that can be actually encountered by interacting with the environment. This has recently motivated the use of approximate solution techniques for POMDPs (Poon, 2001; Roy and Gordon, 2003; Pineau et al., 2003), in which a set of belief points are sampled from the belief simplex—e.g., by random exploration—and then an approximate solution is sought that plans on these points only instead of the complete belief simplex.

In this paper we describe a simple approximate value iteration method for POMDPs that borrows ideas from (Zhang and Zhang, 2001) and (Pineau et al., 2003). The proposed algorithm performs a number of value update steps, making sure that in each step the new value function is an upper bound to the previous value function, as estimated on the sampled set of belief points. A randomized belief-point selection scheme allows for fast update steps. Experimental results indicate that it is capable of computing good solutions in short time.

* Also presented at NIPS'03 workshop "Planning for the Real World", Whistler, Canada, Dec 2003.

2 The POMDP model

In its simplest form, a POMDP model describes the interaction of an agent with its environment as the iteration of the following two steps:

1. At any time step the environment is in a state $s \in S$. An agent that is embedded in the environment takes an action $a \in A$ and receives a reward $r(s, a)$ from the environment as a result of this action. The environment switches to a new state s' according to a known stochastic transition model $p(s'|s, a)$.

2. The agent perceives an observation $o \in O$ that is conditional on its action. This observation provides the agent with information about the state s through a known stochastic observation model $p(o|s, a)$.

All sets S, O, and A are assumed discrete and finite. At any time step the agent can summarize all information about the past in the form of a probability distribution $b(s)$ over states s. This distribution, or *belief*, can be updated using Bayes' rule each time the agent takes an action a and receives an observation o:

$$b_a^o(s') \propto p(o|s', a) \sum_s p(s'|s, a)b(s) \qquad (1)$$

with $\sum_s b_a^o(s) = 1$.

The task of the agent is to find a policy—a mapping from beliefs to actions—that maximizes expected discounted future reward $E[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)]$, where γ is a discount rate. A policy is defined by a value function, which estimates the expected amount of future discounted reward for each belief. The value function of an optimal policy is called the optimal value function and is denoted by V^*. It is the fixed point of the equation $V = HV$, with H the Bellman backup operator:

$$V(b) = \max_a \Big[\sum_s r(s, a)b(s) + \gamma \sum_{o,s,s'} p(o|s', a)p(s'|s, a)b(s)V(b_a^o) \Big] \qquad (2)$$

where b_a^o is given by (1).

3 Value iteration in POMDPs

Value iteration in POMDPs iteratively builds better estimates of V^* by applying the operator H to an initially piecewise linear and convex value function V_0 (Sondik, 1971). The intermediate estimates $V_1, V_2, \ldots$ will then also be piecewise linear and convex. We will throughout assume that a value function V_n at step n is represented by a finite set of vectors $\{\alpha_n^1, \alpha_n^2, \ldots\}$, where each vector defines a hyperplane over the belief simplex. Additionally, with each vector an action is associated, which is the optimal one to take in the current step, assuming optimal actions are executed in following steps. The value of a belief point b is

$$V_n(b) = \max_{\alpha_n^i} b \cdot \alpha_n^i, \qquad (3)$$

where $(\cdot)$ denotes inner product.

Most value iteration algorithms for POMDPs build on the fact that for a given value function V_n and a particular belief point b we can easily compute the vector α of HV_n such that

$$\alpha = \arg\max_{\alpha_{n+1}^i} b \cdot \alpha_{n+1}^i \qquad (4)$$

where $\{\alpha_{n+1}^i\}$ is the (unknown) set of vectors for HV_n. We will denote this operation $\alpha = \texttt{backup}(b, V_n)$. The difficult task, however, is to ensure that *all* vectors of HV_n are generated. Standard exact algorithms (Sondik, 1971; Cheng, 1988; Kaelbling et al., 1998) search in the belief simplex for a minimal set of belief points $\{b_i\}$ that generate the necessary set of vectors for the new horizon value function:

$$\bigcup_{b_i} \texttt{backup}(b_i, V_n) = HV_n. \qquad (5)$$

Finding all these points may require solving a number of linear programs which can be expensive in high dimensions.

An alternative approach to value iteration has been proposed in (Zhang and Zhang, 2001). The authors show that the convergence of value iteration is not compromised if instead of computing in each iteration the optimal HV_n, one computes a value function V_{n+1} that is an upper bound to V_n, and therefore holds

$$V_n \leq V_{n+1} \leq HV_n. \qquad (6)$$

This additionally requires that the value function is appropriately initialized. As shown by the authors, this is trivially realized by choosing V_0 to be a single vector with all its components equal to $\frac{1}{1-\gamma}\min_{s,a} r(s,a)$. Given V_n, the authors propose creating a number of vectors of HV_n by applying `backup` to a fixed set of prototype belief points, and then solve a set of linear programs to ensure that $V_{n+1} \geq V_n$ over the whole belief simplex.

One can also abandon the idea of doing exact value backups and settle for useful approximations. In the recently introduced *point-based* techniques, a set of belief points are first sampled from the belief simplex by letting the agent interact with the environment, and then value updates are performed on these points only (Poon, 2001; Roy and Gordon, 2003; Pineau et al., 2003). In particular, the PBVI algorithm of (Pineau et al., 2003) samples a set B of belief points from the belief simplex (by stochastic simulation), and then it repeatedly applies the `backup` operator on each $b \in B$ for a number of steps, then expands the set B, and so forth.

Point-based solution techniques for POMDPs are justified by the fact that in most practical problems the belief simplex is sparse, in the sense that only a limited number of belief points can ever be reached by letting the agent directly interact with the environment. In these cases, one would like to plan only for those reachable beliefs (which is a tractable problem) instead of planning over the complete belief simplex (which is an intractable problem).

4 The proposed algorithm

In our algorithm we first let the agent randomly explore the environment and collect a set B of reachable belief points. We then initialize the value function V_0 by using a single vector as in (Zhang and Zhang, 2001). The algorithm performs value function update steps, making sure that in each step the new value function estimate V_{n+1} is an upper bound to V_n for all $b \in B$. The choice which belief points to back up is done by randomization over B.

Given V_n, a value update step is as shown below. (The set $\tilde{B}$ contains the belief points whose value has not been improved yet in the current value update step.)

1. Set $V_{n+1} = \emptyset$. Set $\tilde{B} = B$.
2. Sample a belief point b uniformly at random from $\tilde{B}$. Compute $\alpha = \texttt{backup}(b, V_n)$. If $b \cdot \alpha \geq V_n(b)$ then add α to V_{n+1}, otherwise add $\alpha_n = \arg\max_{\alpha_n^i} b \cdot \alpha_n^i$ to V_{n+1}.
3. Compute $\tilde{B} = \{b \in B : V_n(b) > V_{n+1}(b)\}$. If $\tilde{B} = \emptyset$ then stop, otherwise go to 2.

The hope is that by randomly sampling belief points from (increasingly smaller) subsets of B we quickly build a function V_{n+1} that is an upper bound to V_n over B (and often over the complete belief simplex).

As pointed out in (Hauskrecht, 2000, sec. 4), an approximate value iteration algorithm is characterized by the following elements: (1) the computational complexity of the value function backups, (2) the model complexity of the value function (i.e., how many vectors comprise V_n), (3) the ability of the method to bound the exact update operator H, (4) the convergence behavior of the algorithm, and (5) its control performance.

With regard to (1) and (2), the randomized `backup` steps in our algorithm allow for a quick build-up of an upper bound to V_n, especially in the first iterations where V_n and V_{n+1} differ substantially. As a result, the number of vectors generated in each backup step will be small compared to the size of B. Moreover, since the cost of `backup` depends (linearly) on the number of vectors of V_n, the first update steps are typically very fast. This allows the method to quickly reach a good approximate solution with few vectors (see the experiments below).

Concerning (3) and (4), a generated vector $\alpha = \texttt{backup}(b, V_n)$ of V_{n+1} will always be a member of HV_n if $b \cdot \alpha \geq V_n(b)$. Therefore, if the last condition is not violated, the generated V_{n+1} will lower bound HV_n (since not all vectors of HV_n are computed). On the other hand, if $b \cdot \alpha < V_n(b)$ the algorithm adds a vector from $V_n(b)$ so that $V_{n+1}(b) \geq V_n(b)$ for each n and all $b \in B$. In this case V_{n+1} may not lower bound HV_n. However, since each iteration improves the value of each belief point and V^* is an upper bound to V_n, the algorithm must converge (but not necessarily to V^*). With respect to (5) we cannot say much at the moment, except that we observed encouraging results in practice.

Name	$\|S\|$	$\|O\|$	$\|A\|$
Tiger-grid	33	17	5
Hallway	57	21	5
Hallway2	89	17	5
Tag	870	30	5

Table 1: Properties of the problem domains. The first three were introduced in (Littman et al., 1995), the last one in (Pineau et al., 2003).

The main difference with other point-based value update schemes is that we do not back up each $b \in B$, but instead back up (random) points until the value of every $b \in B$ has improved or remained the same. The intuition is that in this way we limit the growth of the number of vectors in the successive value function estimates. Moreover, we can afford to use a large set B of belief points sampled in advance, as opposed to other point-based algorithms that propose growing B incrementally. Larger B means higher chance of incorporating high-valued beliefs in the value backups. In particular, we can start the algorithm by choosing a belief $b^* = \arg\max_b \max_a r(b, a)$, and then perform value updates as described above. In this way, we can quickly propagate high rewards over the sampled belief space.

5 Experiments

We ran experiments applying the proposed algorithm on four problems from the POMDP literature. Table 1 summarizes these problems in terms of the size of S, O and A. The Hallway, Hallway2 and Tiger-grid problems are maze domains commonly used to test scalable POMDP solution techniques (Littman et al., 1995; Brafman, 1997; Zhou and Hansen, 2001; Pineau et al., 2003). The Tag domain (Pineau et al., 2003) is an order of magnitude larger than the first three problems, and models a search and tag game between two robots.

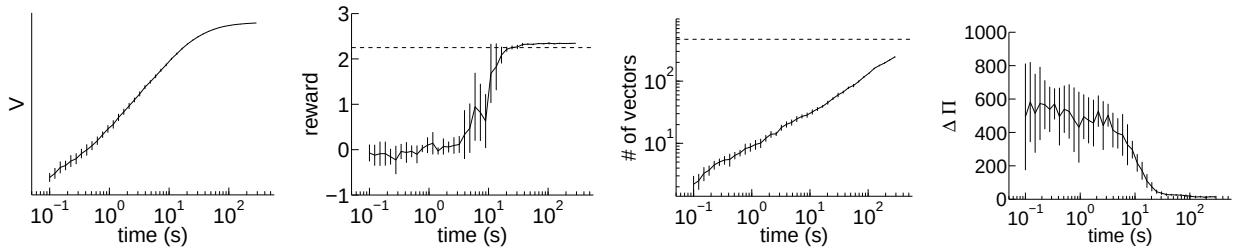
The maze problems

In (Littman et al., 1995) three larger maze domains were introduced: Tiger-grid, Hallway and Hallway2. All of them are navigation tasks: the objective is for an agent to reach a designated goal state as quickly as possible. The agent can observe each possible combination of the presence of a wall in four directions plus a unique observation indicating the goal state; in the Hallway problem three other landmarks are also available. At each step the agent can take one out of five actions: {*stay in place, move forward, turn right, turn left, turn around*}. Both the transition and the observation model are very noisy. For each of the maze problems we used a set B of a 1,000 points, collected at random by stochastic simulation.

Fig. 1 through 3 show the results of our algorithm on the maze problems: (a) the value as estimated on B, $\sum_{b \in B} V(b)$, (b) the expected discounted reward, (c) the number of vectors in the value function estimate, $|\{\alpha^i\}|$ and (d) the number of policy changes: the number of $b \in B$ which had a different optimal action in V_{n-1} compared to V_n. The dashed lines indicate the performance of PBVI. For all four problems we repeated our algorithm 10 times with different random seeds, the error bars indicate standard deviation within these 10 runs. To evaluate the computed policies we tested each of them on 10 runs times 100 starting positions ($\gamma = 0.95$; Tiger-grid: maximum 500 steps, reset at goal; Hallway and Hallway2: maximum 251 steps, terminate at goal). The results show our algorithm can reach good control quality using few vectors (and thus little computation time). Because we can afford to run the algorithm on large belief sets the (d) subfigures can be used as an indication of convergence: if the policy does not change much anymore as estimated on B one could assume the policy has converged. Table 2 summarizes our results, comparing our method to PBVI and Q_{MDP} (Littman et al., 1995). The latter is a simple approximation technique that treats the POMDP as if it were fully observable and solves the underlying MDP, e.g., using value iteration (Sutton and Barto, 1998). Then it uses the resulting $Q(s, a)$ values to define a control policy as $\pi(b) = \arg\max_a \sum_s b(s)Q(s, a)$. Q_{MDP} can be very effective in some domains, but the policies it computes will not take informative actions, as the Q_{MDP} solution assumes uncertainty regarding the state will disappear after taking one action.

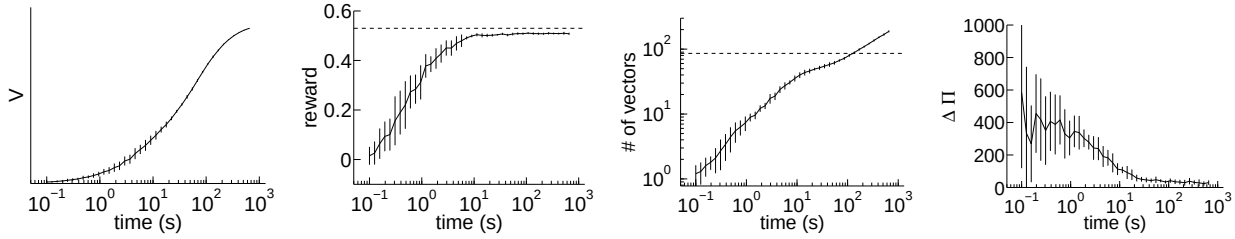
The Tag domain

The goal in the Tag domain, described in (Pineau et al., 2003), is for a robot to search for a moving opponent robot and tag it. The



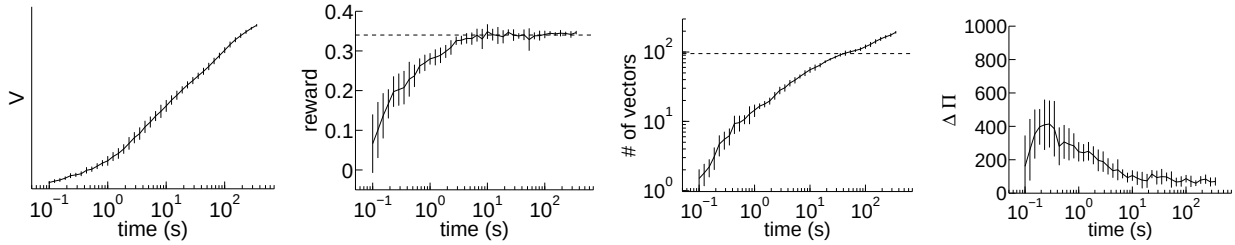
(a) Value. (b) Reward. (c) Number of vectors. (d) Changes in policy.

Figure 1: Results for the Tiger-grid problem.



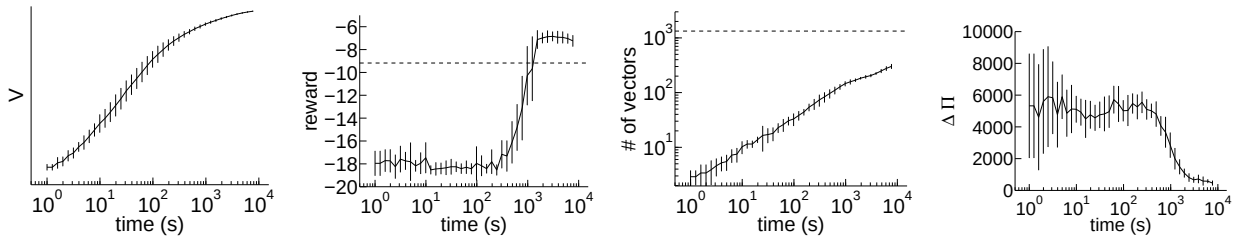
(a) Value. (b) Reward. (c) Number of vectors. (d) Changes in policy.

Figure 2: Results for the Hallway problem.



(a) Value. (b) Reward. (c) Number of vectors. (d) Changes in policy.

Figure 3: Results for the Hallway2 problem.



(a) Value. (b) Reward. (c) Number of vectors. (d) Changes in policy.

Figure 4: Results for the Tag problem.

Method	Reward	Vectors	Time (s)
Tiger-grid			
ours	2.34	134	104
PBVI	2.25	470	3448
Q_{MDP}	0.23	n.a.	2.76
Hallway			
ours	0.51	55	35
PBVI	0.53	86	288
Q_{MDP}	0.27	n.a.	1.34
Hallway2			
ours	0.35	56	10
PBVI	0.34	95	360
Q_{MDP}	0.09	n.a.	2.23
Tag			
ours	−6.85	205	3076
PBVI	−9.18	1334	180880
Q_{MDP}	−16.9	n.a.	16.1

Table 2: Summary of results. PBVI results are taken from (Pineau et al., 2003), so time comparisons are rough.

chasing robot cannot observe the opponent until they occupy the same position, at which time it should execute the *tag* action in order to win the game, and receive a reward of 10. If the opponent is not present at the same location, the reward will be −10, and the robot is penalized with a −1 reward for each motion action it takes. The opponent tries to escape from being tagged by moving away of the chasing robot, it however has a 0.2 chance of remaining at its location. The chasing robot has perfect information regarding its own position and its movement actions {*north*,*east*,*south*,*west*} are deterministic. The state space is represented as the cross-product of the states of the two robots. Both robots can be located in one of the 29 positions depicted in Fig. 5, and the opponent can also be in a special *tagged* state. In total the domain consists of 870 states.

The Tag domain has a high dimensionality compared to other POMDP problems studied in literature, but deals with solving the problems of partial observability on an abstract level. The transition and observation models are sparse, and the belief set resulting from experiencing the environment is also sparse (due to the special structure of the problem). Exploiting this sparsity is easy by taking advantage of sparse matrix computation facilities of available software.

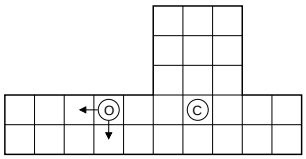


Figure 5: The Chasing and Opponent robot in the Tag domain.

Fig. 4 shows the performance of our algorithm when we used a belief set of 10,000 points, sampled by experiencing the environment uniformly at random. We seeded our algorithm with a belief that has maximal immediate reward. To compute the expected reward we sampled trajectories of at most 100 steps, other parameters are the same as in the maze problem experiments. The results show that in the Tag problem our algorithm displays better control quality than both Q_{MDP} and PBVI, while it uses fewer vectors than PBVI. A summary of the results comparing our algorithm to PBVI and Q_{MDP} can be found in Table 2. Our algorithm reaches very competitive control quality using a relatively small number of vectors.

6 Conclusions and discussion

We presented a new point-based value iteration algorithm for POMDPs that performs value update steps, trying to find in each step a minimal set of vectors V_{n+1} that upper bound the current value function V_n, as estimated on a sampled set B of belief points. The main difference with other point-based value iteration algorithms is that we do not back up the value on each $b \in B$, but back up on randomly selected points from B until the value of every $b \in B$ has improved (or at least remained the same). This allows us to use a large set B of belief points which, in turn, increases the chance that high-reward beliefs are utilized in the value update steps. Results indicate that the proposed algorithm achieves competitive performance, both in terms of solution quality as well as speed.

Our randomized algorithm and the 'full backup' PBVI can be regarded as two extremes of a spectrum of point-based backup schemes. One can think of other mechanisms where a

number of points are backed up in each value backup step in a prioritized manner in order to approximate HV_n. For instance, an idea is to derive a point-based version of the linear support algorithm of (Cheng, 1988), where points are selected from B that are as close as possible to the corner points of the current approximation.

As future work, we would like to study the behavior of our algorithm in more large-scale problems, and locate the class of problems for which the algorithm is most appropriate. We would also like to study multiagent extensions.

Acknowledgments

This research is supported by PROGRESS, the embedded systems research program of the Dutch organization for Scientific Research NWO, the Dutch Ministry of Economic Affairs and the Technology Foundation STW, project AES 5414.

References

R. I. Brafman. 1997. A heuristic variable grid solution method for POMDPs. In *Proceedings of the Fourteenth National Conference on Artificial Intelligence (AAAI '97)*.

H. T. Cheng. 1988. *Algorithms for partially observable Markov decision processes*. Ph.D. thesis, University of British Columbia.

M. Hauskrecht. 2000. Value function approximations for partially observable Markov decision processes. *Journal of Artificial Intelligence Research*, 13:33–95.

L. P. Kaelbling, M. L. Littman, and A. R. Cassandra. 1998. Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, 101:99–134.

M. L. Littman, A. R. Cassandra, and L. Pack Kaelbling. 1995. Learning policies for partially observable environments: Scaling up. In *Proc. 12th Int. Conf. on Machine Learning*, San Francisco, CA.

O. Madani, S. Hanks, and A. Condon. 1999. On the undecidability of probabilistic planning and infinite-horizon partially observable Markov decision problems. In *Proc. 16th National Conf. on Artificial Intelligence*, Orlando, Florida, July.

C. H. Papadimitriou and J. N. Tsitsiklis. 1987. The complexity of Markov decision processes. *Mathematics of operations research*, 12(3):441–450.

J. Pineau, G. Gordon, and S. Thrun. 2003. Point-based value iteration: An anytime algorithm for POMDPs. In *Proc. Int. Joint Conf. on Artificial Intelligence*, Acapulco, Mexico, August.

K.-M. Poon. 2001. A fast heuristic algorithm for decision-theoretic planning. Master's thesis, The Hong-Kong University of Science and Technology.

N. Roy and G. Gordon. 2003. Exponential family PCA for belief compression in POMDPs. In *Advances in Neural Information Processing Systems 15*, Cambridge, MA. MIT Press.

E. J. Sondik. 1971. *The optimal control of partially observable Markov decision processes*. Ph.D. thesis, Stanford University.

R. S. Sutton and A. G. Barto. 1998. *Reinforcement Learning: An Introduction*. MIT Press, Cambridge, MA.

N. L. Zhang and W. Zhang. 2001. Speeding up the convergence of value iteration in partially observable Markov decision processes. *Journal of Artificial Intelligence Research*, 14:29–51.

R. Zhou and E. A. Hansen. 2001. An improved grid-based approximation algorithm for POMDPs. In *Proc. Int. Joint Conf. on Artificial Intelligence*, Seattle, WA, August.