Distributed Asynchronous Computation of Fixed Points and Applications in Dynamic Programming

Dimitri P. Bertsekas

Laboratory for Information and Decision Systems
Massachusetts Institute of Technology

PCO '13
Cambridge, MA
May 2013

Joint Work with Huizhen (Janey) Yu

Summary

Background Review

- Asynchronous iterative fixed point methods
- Convergence issues
- Dynamic programming (DP) applications

New Research on Asynchronous DP Algorithms

- Asynchronous policy iteration (asynchronous value and policy updates by multiple processors)
- Failure of the “natural" algorithm (Williams-Baird counterexample, 1993)
- A radical modification of policy iteration/evaluation: Aim to solve an optimal stopping problem instead of solving a linear system
- Convergence properties are restored/enhanced
- Generalizations and abstractions

Summary

Background Review

- Asynchronous iterative fixed point methods
- Convergence issues
- Dynamic programming (DP) applications

New Research on Asynchronous DP Algorithms

- Asynchronous policy iteration (asynchronous value and policy updates by multiple processors)
- Failure of the “natural" algorithm (Williams-Baird counterexample, 1993)
- A radical modification of policy iteration/evaluation: Aim to solve an optimal stopping problem instead of solving a linear system
- Convergence properties are restored/enhanced
- Generalizations and abstractions

Summary

Background Review

- Asynchronous iterative fixed point methods
- Convergence issues
- Dynamic programming (DP) applications

New Research on Asynchronous DP Algorithms

- Asynchronous policy iteration (asynchronous value and policy updates by multiple processors)
- Failure of the “natural" algorithm (Williams-Baird counterexample, 1993)
- A radical modification of policy iteration/evaluation: Aim to solve an optimal stopping problem instead of solving a linear system
- Convergence properties are restored/enhanced
- Generalizations and abstractions

Summary

Background Review

- Asynchronous iterative fixed point methods
- Convergence issues
- Dynamic programming (DP) applications

New Research on Asynchronous DP Algorithms

- Asynchronous policy iteration (asynchronous value and policy updates by multiple processors)
- Failure of the “natural" algorithm (Williams-Baird counterexample, 1993)
- A radical modification of policy iteration/evaluation: Aim to solve an optimal stopping problem instead of solving a linear system
- Convergence properties are restored/enhanced
- Generalizations and abstractions

Summary

Background Review

- Asynchronous iterative fixed point methods
- Convergence issues
- Dynamic programming (DP) applications

New Research on Asynchronous DP Algorithms

- Asynchronous policy iteration (asynchronous value and policy updates by multiple processors)
- Failure of the “natural" algorithm (Williams-Baird counterexample, 1993)
- A radical modification of policy iteration/evaluation: Aim to solve an optimal stopping problem instead of solving a linear system
- Convergence properties are restored/enhanced
- Generalizations and abstractions

Summary

Background Review

- Asynchronous iterative fixed point methods
- Convergence issues
- Dynamic programming (DP) applications

New Research on Asynchronous DP Algorithms

- Asynchronous policy iteration (asynchronous value and policy updates by multiple processors)
- Failure of the “natural" algorithm (Williams-Baird counterexample, 1993)
- A radical modification of policy iteration/evaluation: Aim to solve an optimal stopping problem instead of solving a linear system
- Convergence properties are restored/enhanced
- Generalizations and abstractions

Summary

Background Review

- Asynchronous iterative fixed point methods
- Convergence issues
- Dynamic programming (DP) applications

New Research on Asynchronous DP Algorithms

- Asynchronous policy iteration (asynchronous value and policy updates by multiple processors)
- Failure of the “natural" algorithm (Williams-Baird counterexample, 1993)
- A radical modification of policy iteration/evaluation: Aim to solve an optimal stopping problem instead of solving a linear system
- Convergence properties are restored/enhanced
- Generalizations and abstractions

Summary

Background Review

- Asynchronous iterative fixed point methods
- Convergence issues
- Dynamic programming (DP) applications

New Research on Asynchronous DP Algorithms

- Asynchronous policy iteration (asynchronous value and policy updates by multiple processors)
- Failure of the “natural" algorithm (Williams-Baird counterexample, 1993)
- A radical modification of policy iteration/evaluation: Aim to solve an optimal stopping problem instead of solving a linear system
- Convergence properties are restored/enhanced
- Generalizations and abstractions

References

Current work

- D. P. Bertsekas and H. Yu, “Q-Learning and Enhanced Policy Iteration in Discounted Dynamic Programming," Math. of OR, 2012
- H. Yu and D. P. Bertsekas, "Q-Learning and Policy Iteration Algorithms for Stochastic Shortest Path Problems," to appear in Annals of OR
- D. P. Bertsekas and H. Yu, “Distributed Asynchronous Policy Iteration," Proc. Allerton Conference, Sept. 2010
- More works in progress

Connection to early work on theory of totally asynchronous algorithms

- D. P. Bertsekas, "Distributed Dynamic Programming," IEEE Transactions on Aut. Control, Vol. AC-27, 1982
- D. P. Bertsekas, "Distributed Asynchronous Computation of Fixed Points," Mathematical Programming, Vol. 27, 1983
- D. P. Bertsekas and J. N. Tsitsiklis, Parallel and Distributed Computation: Numerical Methods, Prentice-Hall, 1989

References

Current work

- D. P. Bertsekas and H. Yu, “Q-Learning and Enhanced Policy Iteration in Discounted Dynamic Programming," Math. of OR, 2012
- H. Yu and D. P. Bertsekas, "Q-Learning and Policy Iteration Algorithms for Stochastic Shortest Path Problems," to appear in Annals of OR
- D. P. Bertsekas and H. Yu, “Distributed Asynchronous Policy Iteration," Proc. Allerton Conference, Sept. 2010

- More works in progress

Connection to early work on theory of totally asynchronous algorithms

- D. P. Bertsekas, "Distributed Dynamic Programming," IEEE Transactions on Aut. Control, Vol. AC-27, 1982
- D. P. Bertsekas, "Distributed Asynchronous Computation of Fixed Points," Mathematical Programming, Vol. 27, 1983
- D. P. Bertsekas and J. N. Tsitsiklis, Parallel and Distributed Computation: Numerical Methods, Prentice-Hall, 1989

Outline

Outline

Some History: The First “Real" Implementation of a Distributed Asynchronous iterative Algorithm

Routing algorithm of the ARPANET (1969)

- Based on the Bellman-Ford shortest path algorithm

$$J(i) = \min_{j=0,1,\dots,n} \{a_{ij} + J(j)\}, \qquad i = 1, \dots, n,$$

where:

- $J(i)$: Estimated distance of node i to destination node 0 ($J(0) = 0$)
- a_{ij}: Length of the link connecting i to j

- Original ambitious implementation included congestion dependent a_{ij}
- Failed miserably because of violent oscillations
- The algorithm was “stabilized" by making a_{ij} essentially constant (1974)
- Showing convergence of this algorithm and its DP extensions was my entry point into the field (1979)

Some History: The First "Real" Implementation of a Distributed Asynchronous iterative Algorithm

Routing algorithm of the ARPANET (1969)

- Based on the Bellman-Ford shortest path algorithm

$$J(i) = \min_{j=0,1,\ldots,n} \{a_{ij} + J(j)\}, \qquad i = 1,\ldots,n,$$

where:

- $J(i)$: Estimated distance of node i to destination node 0 ($J(0) = 0$)
- a_{ij}: Length of the link connecting i to j

- Original ambitious implementation included congestion dependent a_{ij}
- Failed miserably because of violent oscillations
- The algorithm was "stabilized" by making a_{ij} essentially constant (1974)
- Showing convergence of this algorithm and its DP extensions was my entry point into the field (1979)

Some History: The First "Real" Implementation of a Distributed Asynchronous iterative Algorithm

Routing algorithm of the ARPANET (1969)

- Based on the Bellman-Ford shortest path algorithm

$$J(i) = \min_{j=0,1,\ldots,n} \{a_{ij} + J(j)\}, \qquad i = 1,\ldots,n,$$

where:

- $J(i)$: Estimated distance of node i to destination node 0 ($J(0) = 0$)
- a_{ij}: Length of the link connecting i to j

- Original ambitious implementation included congestion dependent a_{ij}
- Failed miserably because of violent oscillations
- The algorithm was "stabilized" by making a_{ij} essentially constant (1974)
- Showing convergence of this algorithm and its DP extensions was my entry point into the field (1979)

Some History: The First “Real" Implementation of a Distributed Asynchronous iterative Algorithm

Routing algorithm of the ARPANET (1969)

- Based on the Bellman-Ford shortest path algorithm

$$J(i) = \min_{j=0,1,\ldots,n} \{a_{ij} + J(j)\}, \qquad i = 1,\ldots,n,$$

where:

- $J(i)$: Estimated distance of node i to destination node 0 ($J(0) = 0$)
- a_{ij}: Length of the link connecting i to j

- Original ambitious implementation included congestion dependent a_{ij}
- Failed miserably because of violent oscillations
- The algorithm was “stabilized" by making a_{ij} essentially constant (1974)
- Showing convergence of this algorithm and its DP extensions was my entry point into the field (1979)

Some History: The First “Real" Implementation of a Distributed Asynchronous iterative Algorithm

Routing algorithm of the ARPANET (1969)

- Based on the Bellman-Ford shortest path algorithm

$$J(i) = \min_{j=0,1,\dots,n} \{a_{ij} + J(j)\}, \qquad i = 1,\dots,n,$$

where:

- $J(i)$: Estimated distance of node i to destination node 0 ($J(0) = 0$)
- a_{ij}: Length of the link connecting i to j

- Original ambitious implementation included congestion dependent a_{ij}
- Failed miserably because of violent oscillations
- The algorithm was “stabilized" by making a_{ij} essentially constant (1974)
- Showing convergence of this algorithm and its DP extensions was my entry point into the field (1979)

Early Work on Asynchronous Iterative Algorithms

- Initial proposal by Rosenfeld (1967) involved experimentation but no convergence analysis
- Sup-norm linear contractive iterations by Chazan and Miranker (1969)
- Sup-norm nonlinear contractive iterations by Miellou (1975), Robert (1976), Baudet (1978)
- DP algorithms (Bertsekas 1982, 1983) - both sup norm contractive and also noncontractive/monotone iterations (e.g., shortest path)
- Many subsequent works (nonexpansive, network, and other iterations)

Types of Asynchronism

- All asynchronous iterative algorithms proposed up to the early 80s involved "total asynchronism" and either sup-norm contractive or monotone mappings
- Subsequent work also involved "partial asynchronism," which could be applied to additional types of iterations (e.g., gradient methods for optimization)

Early Work on Asynchronous Iterative Algorithms

- Initial proposal by Rosenfeld (1967) involved experimentation but no convergence analysis
- Sup-norm linear contractive iterations by Chazan and Miranker (1969)
- Sup-norm nonlinear contractive iterations by Miellou (1975), Robert (1976), Baudet (1978)
- DP algorithms (Bertsekas 1982, 1983) - both sup norm contractive and also noncontractive/monotone iterations (e.g., shortest path)
- Many subsequent works (nonexpansive, network, and other iterations)

Types of Asynchronism

- All asynchronous iterative algorithms proposed up to the early 80s involved "total asynchronism" and either sup-norm contractive or monotone mappings
- Subsequent work also involved "partial asynchronism," which could be applied to additional types of iterations (e.g., gradient methods for optimization)

Early Work on Asynchronous Iterative Algorithms

- Initial proposal by Rosenfeld (1967) involved experimentation but no convergence analysis
- Sup-norm linear contractive iterations by Chazan and Miranker (1969)
- Sup-norm nonlinear contractive iterations by Miellou (1975), Robert (1976), Baudet (1978)
- DP algorithms (Bertsekas 1982, 1983) - both sup norm contractive and also noncontractive/monotone iterations (e.g., shortest path)
- Many subsequent works (nonexpansive, network, and other iterations)

Types of Asynchronism

- All asynchronous iterative algorithms proposed up to the early 80s involved "total asynchronism" and either sup-norm contractive or monotone mappings
- Subsequent work also involved "partial asynchronism," which could be applied to additional types of iterations (e.g., gradient methods for optimization)

Early Work on Asynchronous Iterative Algorithms

- Initial proposal by Rosenfeld (1967) involved experimentation but no convergence analysis
- Sup-norm linear contractive iterations by Chazan and Miranker (1969)
- Sup-norm nonlinear contractive iterations by Miellou (1975), Robert (1976), Baudet (1978)
- DP algorithms (Bertsekas 1982, 1983) - both sup norm contractive and also noncontractive/monotone iterations (e.g., shortest path)
- Many subsequent works (nonexpansive, network, and other iterations)

Types of Asynchronism

- All asynchronous iterative algorithms proposed up to the early 80s involved "total asynchronism" and either sup-norm contractive or monotone mappings
- Subsequent work also involved "partial asynchronism," which could be applied to additional types of iterations (e.g., gradient methods for optimization)

Early Work on Asynchronous Iterative Algorithms

- Initial proposal by Rosenfeld (1967) involved experimentation but no convergence analysis
- Sup-norm linear contractive iterations by Chazan and Miranker (1969)
- Sup-norm nonlinear contractive iterations by Miellou (1975), Robert (1976), Baudet (1978)
- DP algorithms (Bertsekas 1982, 1983) - both sup norm contractive and also noncontractive/monotone iterations (e.g., shortest path)
- Many subsequent works (nonexpansive, network, and other iterations)

Types of Asynchronism

- All asynchronous iterative algorithms proposed up to the early 80s involved "total asynchronism" and either sup-norm contractive or monotone mappings
- Subsequent work also involved "partial asynchronism," which could be applied to additional types of iterations (e.g., gradient methods for optimization)

Early Work on Asynchronous Iterative Algorithms

- Initial proposal by Rosenfeld (1967) involved experimentation but no convergence analysis
- Sup-norm linear contractive iterations by Chazan and Miranker (1969)
- Sup-norm nonlinear contractive iterations by Miellou (1975), Robert (1976), Baudet (1978)
- DP algorithms (Bertsekas 1982, 1983) - both sup norm contractive and also noncontractive/monotone iterations (e.g., shortest path)
- Many subsequent works (nonexpansive, network, and other iterations)

Types of Asynchronism

- All asynchronous iterative algorithms proposed up to the early 80s involved "total asynchronism" and either sup-norm contractive or monotone mappings
- Subsequent work also involved "partial asynchronism," which could be applied to additional types of iterations (e.g., gradient methods for optimization)

Early Work on Asynchronous Iterative Algorithms

- Initial proposal by Rosenfeld (1967) involved experimentation but no convergence analysis
- Sup-norm linear contractive iterations by Chazan and Miranker (1969)
- Sup-norm nonlinear contractive iterations by Miellou (1975), Robert (1976), Baudet (1978)
- DP algorithms (Bertsekas 1982, 1983) - both sup norm contractive and also noncontractive/monotone iterations (e.g., shortest path)
- Many subsequent works (nonexpansive, network, and other iterations)

Types of Asynchronism

- All asynchronous iterative algorithms proposed up to the early 80s involved "total asynchronism" and either sup-norm contractive or monotone mappings
- Subsequent work also involved "partial asynchronism," which could be applied to additional types of iterations (e.g., gradient methods for optimization)

Distributed "Totally" Asynchronous Framework for Fixed Point Computation

- Consider solution of general fixed point problem $J = TJ$, or

$$J(i) = T_i\big(J(1), \ldots, J(n)\big), \qquad i = 1, \ldots, n$$

- Network of processors, with a separate processor i for each component $J(i)$, $i = 1, \ldots, n$

Asynchronous Fixed Point Algorithm

- Processor i updates $J(i)$ at a subset of times $\mathscr{T}_i \subset \{0, 1, \ldots\}$
- Processor i receives (possibly outdated values) $J(j)$ from other processors $j \neq i$
- Update of processor i [with “delays" $t - \tau_{ij}(t)$]

$$J^{t+1}(i) = \begin{cases} T_i\big(J^{\tau_{i1}(t)}(1), \ldots, J^{\tau_{in}(t)}(n)\big) & \text{if } t \in \mathscr{T}_i, \\ J^t(i) & \text{if } t \notin \mathscr{T}_i. \end{cases}$$

Distributed "Totally" Asynchronous Framework for Fixed Point Computation

- Consider solution of general fixed point problem $J = TJ$, or

$$J(i) = T_i\big(J(1), \dots, J(n)\big), \qquad i = 1, \dots, n$$

- Network of processors, with a separate processor i for each component $J(i)$, $i = 1, \dots, n$

Asynchronous Fixed Point Algorithm

- Processor i updates $J(i)$ at a subset of times $\mathscr{T}_i \subset \{0, 1, \dots\}$
- Processor i receives (possibly outdated values) $J(j)$ from other processors $j \neq i$
- Update of processor i [with "delays" $t - \tau_{ij}(t)$]

$$J^{t+1}(i) = \begin{cases} T_i\big(J^{\tau_{i1}(t)}(1), \dots, J^{\tau_{in}(t)}(n)\big) & \text{if } t \in \mathscr{T}_i, \\ J^t(i) & \text{if } t \notin \mathscr{T}_i. \end{cases}$$

Distributed "Totally" Asynchronous Framework for Fixed Point Computation

- Consider solution of general fixed point problem $J = TJ$, or

$$J(i) = T_i\big(J(1), \dots, J(n)\big), \qquad i = 1, \dots, n$$

- Network of processors, with a separate processor i for each component $J(i)$, $i = 1, \dots, n$

Asynchronous Fixed Point Algorithm

- Processor i updates $J(i)$ at a subset of times $\mathscr{T}_i \subset \{0, 1, \dots\}$
- Processor i receives (possibly outdated values) $J(j)$ from other processors $j \neq i$
- Update of processor i [with "delays" $t - \tau_{ij}(t)$]

$$J^{t+1}(i) = \begin{cases} T_i\big(J^{\tau_{i1}(t)}(1), \dots, J^{\tau_{in}(t)}(n)\big) & \text{if } t \in \mathscr{T}_i, \\ J^t(i) & \text{if } t \notin \mathscr{T}_i. \end{cases}$$

Distributed "Totally" Asynchronous Framework for Fixed Point Computation

- Consider solution of general fixed point problem $J = TJ$, or

$$J(i) = T_i\big(J(1), \dots, J(n)\big), \qquad i = 1, \dots, n$$

- Network of processors, with a separate processor i for each component $J(i)$, $i = 1, \dots, n$

Asynchronous Fixed Point Algorithm

- Processor i updates $J(i)$ at a subset of times $\mathscr{T}_i \subset \{0, 1, \dots\}$
- Processor i receives (possibly outdated values) $J(j)$ from other processors $j \neq i$
- Update of processor i [with "delays" $t - \tau_{ij}(t)$]

$$J^{t+1}(i) = \begin{cases} T_i\big(J^{\tau_{i1}(t)}(1), \dots, J^{\tau_{in}(t)}(n)\big) & \text{if } t \in \mathscr{T}_i, \\ J^t(i) & \text{if } t \notin \mathscr{T}_i. \end{cases}$$

Distributed "Totally" Asynchronous Framework for Fixed Point Computation

- Consider solution of general fixed point problem $J = TJ$, or

$$J(i) = T_i\big(J(1), \dots, J(n)\big), \qquad i = 1, \dots, n$$

- Network of processors, with a separate processor i for each component $J(i)$, $i = 1, \dots, n$

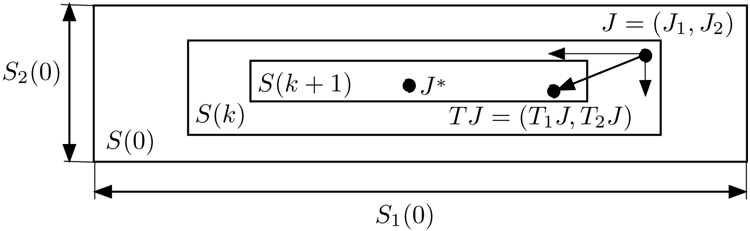
Asynchronous Fixed Point Algorithm

- Processor i updates $J(i)$ at a subset of times $\mathscr{T}_i \subset \{0, 1, \dots\}$
- Processor i receives (possibly outdated values) $J(j)$ from other processors $j \neq i$
- Update of processor i [with “delays" $t - \tau_{ij}(t)$]

$$J^{t+1}(i) = \begin{cases} T_i\big(J^{\tau_{i1}(t)}(1), \dots, J^{\tau_{in}(t)}(n)\big) & \text{if } t \in \mathscr{T}_i, \\ J^t(i) & \text{if } t \notin \mathscr{T}_i. \end{cases}$$

Distributed Convergence of Fixed Point Iterations

A general theorem for “totally asynchronous" iterations, i.e., $\mathscr{T}_i$ are infinite sets and $\tau_{ij}(t) \to \infty$ as $t \to \infty$ (Bertsekas, 1983)



- Assume there is a nested sequence of sets $S(k+1) \subset S(k)$ such that
  - (Synchronous Convergence Condition) We have

    $$TJ \in S(k+1), \qquad \forall\, J \in S(k),$$

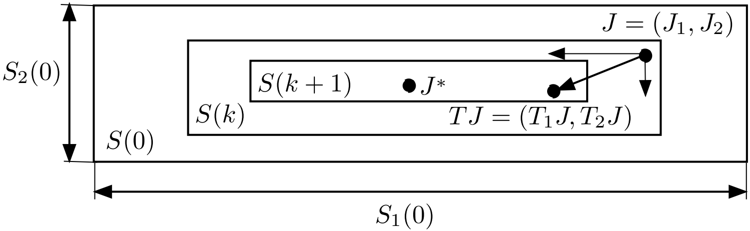
    and the limit points of all sequences $\{J^k\}$ with $J^k \in S(k)$, for all k, are fixed points of T.
  - (Box Condition) $S(k)$ is a Cartesian product:

    $$S(k) = S_1(k) \times \cdots \times S_n(k)$$

Then, if $J^0 \in S(0)$, every limit point of $\{J^t\}$ is a fixed point of T.

Distributed Convergence of Fixed Point Iterations

A general theorem for "totally asynchronous" iterations, i.e., $\mathscr{T}_i$ are infinite sets and $\tau_{ij}(t) \to \infty$ as $t \to \infty$ (Bertsekas, 1983)



- Assume there is a nested sequence of sets $S(k+1) \subset S(k)$ such that
  - (Synchronous Convergence Condition) We have

    $$TJ \in S(k+1), \qquad \forall\, J \in S(k),$$

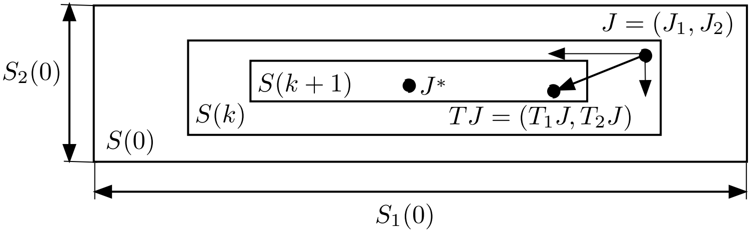
    and the limit points of all sequences $\{J^k\}$ with $J^k \in S(k)$, for all k, are fixed points of T.
  - (Box Condition) $S(k)$ is a Cartesian product:

    $$S(k) = S_1(k) \times \cdots \times S_n(k)$$

Then, if $J^0 \in S(0)$, every limit point of $\{J^t\}$ is a fixed point of T.

Distributed Convergence of Fixed Point Iterations

A general theorem for “totally asynchronous" iterations, i.e., $\mathscr{T}_i$ are infinite sets and $\tau_{ij}(t) \to \infty$ as $t \to \infty$ (Bertsekas, 1983)

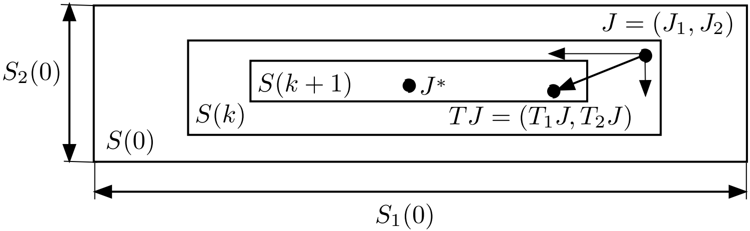


- Assume there is a nested sequence of sets $S(k+1) \subset S(k)$ such that
  - (Synchronous Convergence Condition) We have
  $$TJ \in S(k+1), \qquad \forall\, J \in S(k),$$
  and the limit points of all sequences $\{J^k\}$ with $J^k \in S(k)$, for all k, are fixed points of T.
  - (Box Condition) $S(k)$ is a Cartesian product:
  $$S(k) = S_1(k) \times \cdots \times S_n(k)$$

Then, if $J^0 \in S(0)$, every limit point of $\{J^t\}$ is a fixed point of T.

Distributed Convergence of Fixed Point Iterations

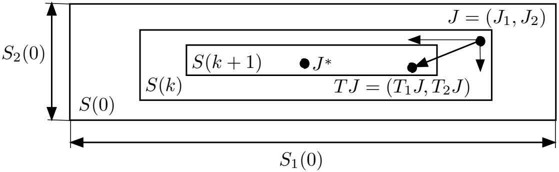
A general theorem for "totally asynchronous" iterations, i.e., $\mathscr{T}_i$ are infinite sets and $\tau_{ij}(t) \to \infty$ as $t \to \infty$ (Bertsekas, 1983)



- Assume there is a nested sequence of sets $S(k+1) \subset S(k)$ such that
  - (Synchronous Convergence Condition) We have
$$TJ \in S(k+1), \qquad \forall\, J \in S(k),$$
and the limit points of all sequences $\{J^k\}$ with $J^k \in S(k)$, for all k, are fixed points of T.
  - (Box Condition) $S(k)$ is a Cartesian product:
$$S(k) = S_1(k) \times \cdots \times S_n(k)$$

Then, if $J^0 \in S(0)$, every limit point of $\{J^t\}$ is a fixed point of T.

Applications of the Theorem



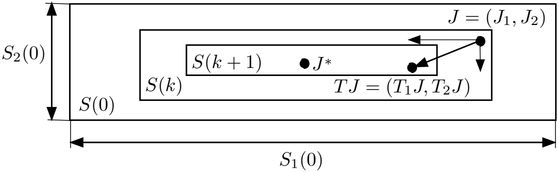
Major contexts where the theorem applies:

- T is a sup-norm contraction with fixed point J^* and modulus α:
$$S(k) = \{J \mid \|J - J^*\|_\infty \le \alpha^k B\}, \qquad \text{for some scalar } B$$
- T is monotone ($TJ \le TJ'$ for $J \le J'$) with fixed point J^*. Moreover,
$$S(k) = \{J \mid T^k \underline{J} \le J \le T^k \bar{J}\}$$
for some $\underline{J}$ and $\bar{J}$ with
$$\underline{J} \le T\underline{J} \le T\bar{J} \le \bar{J},$$
and $\lim_{k\to\infty} T^k \underline{J} = \lim_{k\to\infty} T^k \bar{J} = J^*$.

Both of these apply to various DP problems:

- 1st context applies to discounted problems
- 2nd context applies to undiscounted problems (e.g., shortest paths).

Applications of the Theorem



Major contexts where the theorem applies:

- T is a sup-norm contraction with fixed point J^* and modulus α:

$$S(k) = \{J \mid \|J - J^*\|_\infty \le \alpha^k B\}, \qquad \text{for some scalar } B$$

- T is monotone ($TJ \le TJ'$ for $J \le J'$) with fixed point J^*. Moreover,

$$S(k) = \{J \mid T^k \underline{J} \le J \le T^k \bar{J}\}$$

for some $\underline{J}$ and $\bar{J}$ with

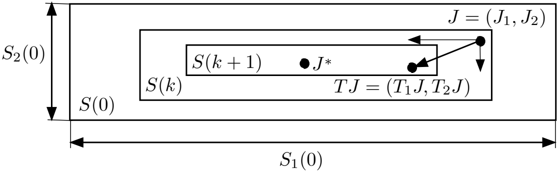
$$\underline{J} \le T\underline{J} \le T\bar{J} \le \bar{J},$$

and $\lim_{k\to\infty} T^k \underline{J} = \lim_{k\to\infty} T^k \bar{J} = J^*$.

Both of these apply to various DP problems:

- 1st context applies to discounted problems
- 2nd context applies to undiscounted problems (e.g., shortest paths)

Applications of the Theorem



Major contexts where the theorem applies:

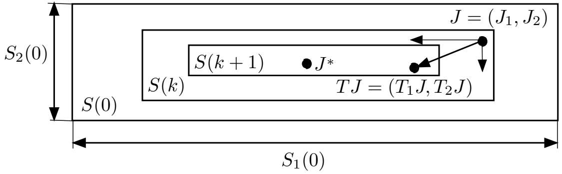
- T is a sup-norm contraction with fixed point J^* and modulus α:
$$S(k) = \{J \mid \|J - J^*\|_\infty \le \alpha^k B\}, \qquad \text{for some scalar } B$$

- T is monotone ($TJ \le TJ'$ for $J \le J'$) with fixed point J^*. Moreover,
$$S(k) = \{J \mid T^k \underline{J} \le J \le T^k \bar{J}\}$$
for some $\underline{J}$ and $\bar{J}$ with
$$\underline{J} \le T\underline{J} \le T\bar{J} \le \bar{J},$$
and $\lim_{k\to\infty} T^k \underline{J} = \lim_{k\to\infty} T^k \bar{J} = J^*$.

Both of these apply to various DP problems:

- 1st context applies to discounted problems
- 2nd context applies to undiscounted problems (e.g., shortest paths)

Applications of the Theorem



Major contexts where the theorem applies:

- T is a sup-norm contraction with fixed point J^* and modulus α:

$$S(k) = \{J \mid \|J - J^*\|_\infty \le \alpha^k B\}, \qquad \text{for some scalar } B$$

- T is monotone ($TJ \le TJ'$ for $J \le J'$) with fixed point J^*. Moreover,

$$S(k) = \{J \mid T^k \underline{J} \le J \le T^k \bar{J}\}$$

  for some $\underline{J}$ and $\bar{J}$ with

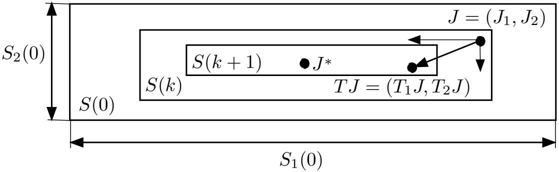
$$\underline{J} \le T\underline{J} \le T\bar{J} \le \bar{J},$$

  and $\lim_{k\to\infty} T^k \underline{J} = \lim_{k\to\infty} T^k \bar{J} = J^*$.

Both of these apply to various DP problems:

- 1st context applies to discounted problems
- 2nd context applies to undiscounted problems (e.g., shortest paths)

Applications of the Theorem



Major contexts where the theorem applies:

- T is a sup-norm contraction with fixed point J^* and modulus α:

$$S(k) = \{J \mid \|J - J^*\|_\infty \leq \alpha^k B\}, \qquad \text{for some scalar } B$$

- T is monotone ($TJ \leq TJ'$ for $J \leq J'$) with fixed point J^*. Moreover,

$$S(k) = \{J \mid T^k \underline{J} \leq J \leq T^k \bar{J}\}$$

  for some $\underline{J}$ and $\bar{J}$ with

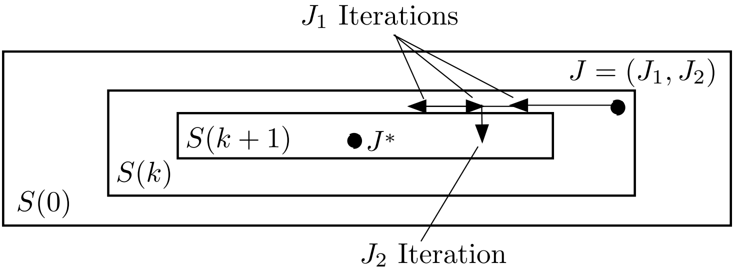
$$\underline{J} \leq T\underline{J} \leq T\bar{J} \leq \bar{J},$$

  and $\lim_{k\to\infty} T^k \underline{J} = \lim_{k\to\infty} T^k \bar{J} = J^*$.

Both of these apply to various DP problems:

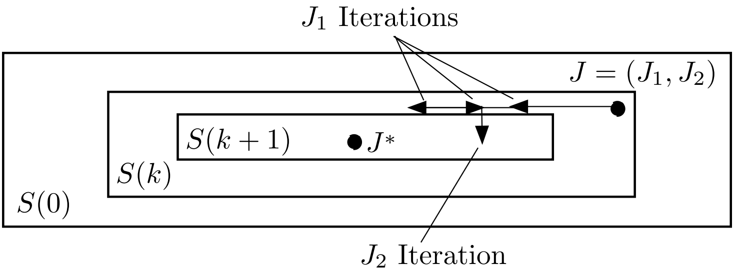
- 1st context applies to discounted problems
- 2nd context applies to undiscounted problems (e.g., shortest paths)

Asynchronous Convergence Mechanism



- Once we are in $S(k)$, iteration of any one component (say J_1) keeps us in the current box
- Once all components have been iterated once, we pass into the next box $S(k+1)$ (permanently)

Asynchronous Convergence Mechanism



- Once we are in $S(k)$, iteration of any one component (say J_1) keeps us in the current box
- Once all components have been iterated once, we pass into the next box $S(k+1)$ (permanently)

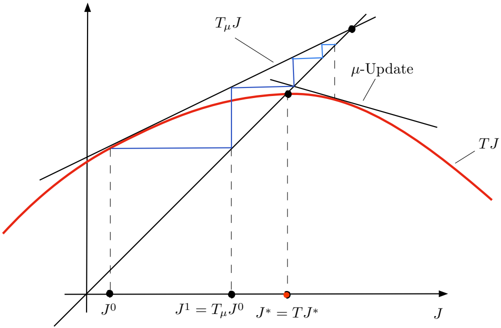
Finding Fixed Points of Parametric Mappings

- Problem: Find a fixed point J^* of a mapping $T : \Re^n \mapsto \Re^n$ of the form

$$(TJ)(i) = \min_{\mu \in \mathcal{M}_i} (T_\mu J)(i), \qquad i = 1, \ldots, n,$$

  where μ is a parameter from some set $\mathcal{M}_i$.

- Common idea (central in DP policy iteration): Instead of T, we iterate with some T_μ, then change μ once in a while



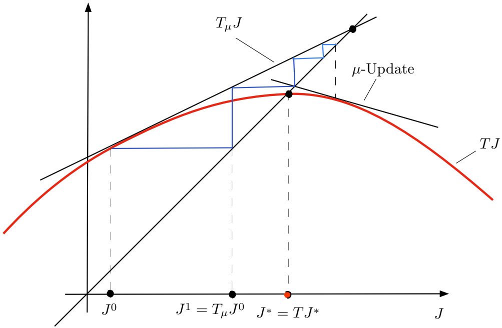
Finding Fixed Points of Parametric Mappings

- Problem: Find a fixed point J^* of a mapping $T : \Re^n \mapsto \Re^n$ of the form

$$(TJ)(i) = \min_{\mu \in \mathcal{M}_i} (T_\mu J)(i), \qquad i = 1, \ldots, n,$$

where μ is a parameter from some set $\mathcal{M}_i$.

- Common idea (central in DP policy iteration): Instead of T, we iterate with some T_μ, then change μ once in a while



Distributed Asynchronous Parametric Fixed Point Iterations

Partition J and μ into components, $J(1), \dots, J(n)$ and $\mu(1), \dots, \mu(n)$

Asynchronous Algorithm

- Processor i, at each time does one of three things:
  - Does nothing or
  - Updates $J(i)$ by setting it to

$$J(i) := (T_{\mu(i)} J)(i)$$

    or
  - Updates $J(i)$ and $\mu(i)$ by setting

$$(TJ)(i) := \min_{\mu \in \mathcal{M}_i} (T_\mu J)(i)$$

    and

$$\mu(i) := \arg \min_{\mu \in \mathcal{M}_i} (T_\mu J)(i)$$

A very chaotic environment!

Distributed Asynchronous Parametric Fixed Point Iterations

Partition J and μ into components, $J(1), \ldots, J(n)$ and $\mu(1), \ldots, \mu(n)$

Asynchronous Algorithm

- Processor i, at each time does one of three things:
  - Does nothing or
  - Updates $J(i)$ by setting it to

$$J(i) := (T_{\mu(i)} J)(i)$$

  or
  - Updates $J(i)$ and $\mu(i)$ by setting

$$(TJ)(i) := \min_{\mu \in \mathcal{M}_i} (T_\mu J)(i)$$

  and

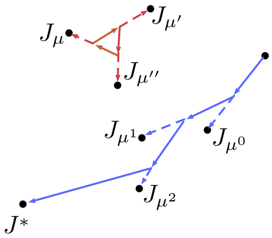
$$\mu(i) := \arg \min_{\mu \in \mathcal{M}_i} (T_\mu J)(i)$$

A very chaotic environment!

Difficulty with Parametric Fixed Point Algorithm

J_μ : Fixed Point of T_μ

$J*$: Fixed Point of T

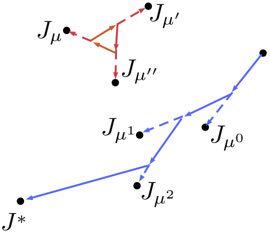


- T_μ has different fixed point than T ... so the target of the iterations keeps changing
- A (restrictive) convergence result: If each T_μ is both a sup-norm contraction and is monotone, convergence can be shown assuming that $T_{\mu^0}J^0 \leq J^0$
- Without the condition $T_{\mu^0}J^0 \leq J^0$, the synchronous algorithm converges but the asynchronous algorithm may oscillate

Difficulty with Parametric Fixed Point Algorithm

J_μ : Fixed Point of T_μ

$J*$: Fixed Point of T

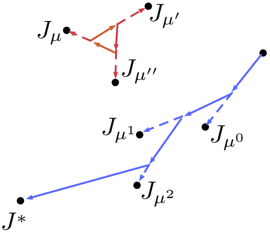


- T_μ has different fixed point than T ... so the target of the iterations keeps changing
- A (restrictive) convergence result: If each T_μ is both a sup-norm contraction and is monotone, convergence can be shown assuming that $T_{\mu^0}J^0 \leq J^0$
- Without the condition $T_{\mu^0}J^0 \leq J^0$, the synchronous algorithm converges but the asynchronous algorithm may oscillate

Difficulty with Parametric Fixed Point Algorithm

J_μ : Fixed Point of T_μ

J^*: Fixed Point of T



- T_μ has different fixed point than T ... so the target of the iterations keeps changing
- A (restrictive) convergence result: If each T_μ is both a sup-norm contraction and is monotone, convergence can be shown assuming that $T_{\mu^0} J^0 \le J^0$
- Without the condition $T_{\mu^0} J^0 \le J^0$, the synchronous algorithm converges but the asynchronous algorithm may oscillate

Addressing the Difficulty: A High-Level View

Assume that each T_μ is a sup-norm contraction

- Our approach: Embed both T and T_μ within another (uniform) contraction mapping G_μ that has the same fixed point for all μ
- The uniform contraction mapping G_μ operates on a larger space
- In the DP context, G_μ is associated with an optimal stopping problem
- Most of what follows applies beyond DP

Addressing the Difficulty: A High-Level View

Assume that each T_μ is a sup-norm contraction

- Our approach: Embed both T and T_μ within another (uniform) contraction mapping G_μ that has the same fixed point for all μ
- The uniform contraction mapping G_μ operates on a larger space
- In the DP context, G_μ is associated with an optimal stopping problem
- Most of what follows applies beyond DP

Addressing the Difficulty: A High-Level View

Assume that each T_μ is a sup-norm contraction

- Our approach: Embed both T and T_μ within another (uniform) contraction mapping G_μ that has the same fixed point for all μ
- The uniform contraction mapping G_μ operates on a larger space
- In the DP context, G_μ is associated with an optimal stopping problem
- Most of what follows applies beyond DP

Addressing the Difficulty: A High-Level View

Assume that each T_μ is a sup-norm contraction

- Our approach: Embed both T and T_μ within another (uniform) contraction mapping G_μ that has the same fixed point for all μ
- The uniform contraction mapping G_μ operates on a larger space
- In the DP context, G_μ is associated with an optimal stopping problem
- Most of what follows applies beyond DP

Addressing the Difficulty: A High-Level View

Assume that each T_μ is a sup-norm contraction

- Our approach: Embed both T and T_μ within another (uniform) contraction mapping G_μ that has the same fixed point for all μ
- The uniform contraction mapping G_μ operates on a larger space
- In the DP context, G_μ is associated with an optimal stopping problem
- Most of what follows applies beyond DP

Outline

Dynamic Programming - Markovian Decision Problems (MDP)

- **System:** Controlled Markov chain w/ transition probabilities $p_{ij}(u)$
- **States:** $i = 1, \ldots, n$
- **Controls:** $u \in U(i)$
- **Cost per stage:** $g(i, u, j)$
- **Stationary policy:** State to control mapping μ; apply $\mu(i)$ when at state i
- **Discounted MDP:** Find policy μ that minimizes the expected value of the infinite horizon cost:

$$\sum_{k=0}^{\infty} \alpha^k g\big(i_k, \mu(i_k), i_{k+1}\big)$$

where

$$i_k = \text{ state at time } k, \qquad i_{k+1} = \text{ state at time } k+1,$$

$$\alpha : \text{ discount factor } \quad 0 < \alpha < 1$$

Dynamic Programming - Markovian Decision Problems (MDP)

- System: Controlled Markov chain w/ transition probabilities $p_{ij}(u)$
- States: $i = 1, \ldots, n$
- Controls: $u \in U(i)$
- Cost per stage: $g(i, u, j)$
- Stationary policy: State to control mapping μ; apply $\mu(i)$ when at state i
- Discounted MDP: Find policy μ that minimizes the expected value of the infinite horizon cost:

$$\sum_{k=0}^{\infty} \alpha^k g\big(i_k, \mu(i_k), i_{k+1}\big)$$

where

$$i_k = \text{ state at time } k, \qquad i_{k+1} = \text{ state at time } k+1,$$

$$\alpha : \text{ discount factor } \quad 0 < \alpha < 1$$

Dynamic Programming - Markovian Decision Problems (MDP)

- System: Controlled Markov chain w/ transition probabilities $p_{ij}(u)$
- States: $i = 1, \ldots, n$
- Controls: $u \in U(i)$
- Cost per stage: $g(i, u, j)$
- Stationary policy: State to control mapping μ; apply $\mu(i)$ when at state i
- Discounted MDP: Find policy μ that minimizes the expected value of the infinite horizon cost:

$$\sum_{k=0}^{\infty} \alpha^k g(i_k, \mu(i_k), i_{k+1})$$

where

$$i_k = \text{ state at time } k, \qquad i_{k+1} = \text{ state at time } k+1,$$

$$\alpha : \text{ discount factor } \quad 0 < \alpha < 1$$

Dynamic Programming - Markovian Decision Problems (MDP)

- System: Controlled Markov chain w/ transition probabilities $p_{ij}(u)$
- States: $i = 1, \ldots, n$
- Controls: $u \in U(i)$
- Cost per stage: $g(i, u, j)$
- Stationary policy: State to control mapping μ; apply $\mu(i)$ when at state i
- Discounted MDP: Find policy μ that minimizes the expected value of the infinite horizon cost:

$$\sum_{k=0}^{\infty} \alpha^k g\big(i_k, \mu(i_k), i_{k+1}\big)$$

where

$$i_k = \text{ state at time } k, \qquad i_{k+1} = \text{ state at time } k+1,$$

$$\alpha: \text{ discount factor} \quad 0 < \alpha < 1$$

Dynamic Programming - Markovian Decision Problems (MDP)

- System: Controlled Markov chain w/ transition probabilities $p_{ij}(u)$
- States: $i = 1, \ldots, n$
- Controls: $u \in U(i)$
- Cost per stage: $g(i, u, j)$
- Stationary policy: State to control mapping μ; apply $\mu(i)$ when at state i
- Discounted MDP: Find policy μ that minimizes the expected value of the infinite horizon cost:

$$\sum_{k=0}^{\infty} \alpha^k g\big(i_k, \mu(i_k), i_{k+1}\big)$$

where

$$i_k = \text{ state at time } k, \qquad i_{k+1} = \text{ state at time } k+1,$$

$$\alpha : \text{ discount factor} \quad 0 < \alpha < 1$$

Dynamic Programming - Markovian Decision Problems (MDP)

- System: Controlled Markov chain w/ transition probabilities $p_{ij}(u)$
- States: $i = 1, \ldots, n$
- Controls: $u \in U(i)$
- Cost per stage: $g(i, u, j)$
- Stationary policy: State to control mapping μ; apply $\mu(i)$ when at state i
- Discounted MDP: Find policy μ that minimizes the expected value of the infinite horizon cost:

$$\sum_{k=0}^{\infty} \alpha^k g\big(i_k, \mu(i_k), i_{k+1}\big)$$

where

$$i_k = \text{ state at time } k, \qquad i_{k+1} = \text{ state at time } k+1,$$

$$\alpha : \text{ discount factor} \quad 0 < \alpha < 1$$

Major DP Results

- $J^*(i)$ = Optimal cost starting from state i
- $J_\mu(i)$ = Optimal cost starting from state i using policy μ
- Bellman's equation:

$$J^*(i) = \min_{u \in U(i)} \sum_{j=1}^{n} p_{ij}(u)\big(g(i,u,j) + \alpha J^*(j)\big), \qquad i = 1,\ldots,n$$

  A parametric fixed point equation.

- J^* is the unique fixed point.
- An optimal policy minimizes for each i in the RHS of Bellman's equation.
- Bellman's equation for a policy μ:

$$J_\mu(i) = \sum_{j=1}^{n} p_{ij}\big(\mu(i)\big)\big(g(i,\mu(i),j) + \alpha J_\mu(j)\big), \qquad i = 1,\ldots,n$$

- It is a linear system of equations with J_μ as its unique solution.

Major DP Results

- $J^*(i)$ = Optimal cost starting from state i
- $J_\mu(i)$ = Optimal cost starting from state i using policy μ
- Bellman's equation:

$$J^*(i) = \min_{u \in U(i)} \sum_{j=1}^{n} p_{ij}(u)\big(g(i,u,j) + \alpha J^*(j)\big), \qquad i = 1, \ldots, n$$

  A parametric fixed point equation.
- J^* is the unique fixed point.
- An optimal policy minimizes for each i in the RHS of Bellman's equation.
- Bellman's equation for a policy μ:

$$J_\mu(i) = \sum_{j=1}^{n} p_{ij}\big(\mu(i)\big)\big(g(i,\mu(i),j) + \alpha J_\mu(j)\big), \qquad i = 1, \ldots, n$$

- It is a linear system of equations with J_μ as its unique solution.

Major DP Results

- $J^*(i)$ = Optimal cost starting from state i
- $J_\mu(i)$ = Optimal cost starting from state i using policy μ
- Bellman's equation:

$$J^*(i) = \min_{u \in U(i)} \sum_{j=1}^{n} p_{ij}(u)\big(g(i,u,j) + \alpha J^*(j)\big), \qquad i = 1, \ldots, n$$

  A parametric fixed point equation.

- J^* is the unique fixed point.
- An optimal policy minimizes for each i in the RHS of Bellman's equation.
- Bellman's equation for a policy μ:

$$J_\mu(i) = \sum_{j=1}^{n} p_{ij}\big(\mu(i)\big)\big(g(i,\mu(i),j) + \alpha J_\mu(j)\big), \qquad i = 1, \ldots, n$$

- It is a linear system of equations with J_μ as its unique solution.

Major DP Results

- $J^*(i)$ = Optimal cost starting from state i
- $J_\mu(i)$ = Optimal cost starting from state i using policy μ
- Bellman's equation:

$$J^*(i) = \min_{u \in U(i)} \sum_{j=1}^{n} p_{ij}(u)\big(g(i,u,j) + \alpha J^*(j)\big), \qquad i = 1, \ldots, n$$

  A parametric fixed point equation.

- J^* is the unique fixed point.
- An optimal policy minimizes for each i in the RHS of Bellman's equation.
- Bellman's equation for a policy μ:

$$J_\mu(i) = \sum_{j=1}^{n} p_{ij}\big(\mu(i)\big)\big(g(i,\mu(i),j) + \alpha J_\mu(j)\big), \qquad i = 1, \ldots, n$$

- It is a linear system of equations with J_μ as its unique solution.

Major DP Results

- $J^*(i)$ = Optimal cost starting from state i
- $J_\mu(i)$ = Optimal cost starting from state i using policy μ
- Bellman's equation:

$$J^*(i) = \min_{u \in U(i)} \sum_{j=1}^{n} p_{ij}(u)\big(g(i,u,j) + \alpha J^*(j)\big), \qquad i = 1, \ldots, n$$

  A parametric fixed point equation.

- J^* is the unique fixed point.
- An optimal policy minimizes for each i in the RHS of Bellman's equation.
- Bellman's equation for a policy μ:

$$J_\mu(i) = \sum_{j=1}^{n} p_{ij}\big(\mu(i)\big)\big(g(i,\mu(i),j) + \alpha J_\mu(j)\big), \qquad i = 1, \ldots, n$$

- It is a linear system of equations with J_μ as its unique solution.

Major DP Results

- $J^*(i)$ = Optimal cost starting from state i
- $J_\mu(i)$ = Optimal cost starting from state i using policy μ
- Bellman's equation:

$$J^*(i) = \min_{u \in U(i)} \sum_{j=1}^{n} p_{ij}(u)\big(g(i,u,j) + \alpha J^*(j)\big), \qquad i = 1, \dots, n$$

  A parametric fixed point equation.

- J^* is the unique fixed point.
- An optimal policy minimizes for each i in the RHS of Bellman's equation.
- Bellman's equation for a policy μ:

$$J_\mu(i) = \sum_{j=1}^{n} p_{ij}\big(\mu(i)\big)\big(g(i,\mu(i),j) + \alpha J_\mu(j)\big), \qquad i = 1, \dots, n$$

- It is a linear system of equations with J_μ as its unique solution.

Major DP Results

- $J^*(i)$ = Optimal cost starting from state i
- $J_\mu(i)$ = Optimal cost starting from state i using policy μ
- Bellman's equation:

$$J^*(i) = \min_{u \in U(i)} \sum_{j=1}^{n} p_{ij}(u)\big(g(i,u,j) + \alpha J^*(j)\big), \qquad i = 1, \ldots, n$$

  A parametric fixed point equation.

- J^* is the unique fixed point.
- An optimal policy minimizes for each i in the RHS of Bellman's equation.
- Bellman's equation for a policy μ:

$$J_\mu(i) = \sum_{j=1}^{n} p_{ij}\big(\mu(i)\big)\big(g(i,\mu(i),j) + \alpha J_\mu(j)\big), \qquad i = 1, \ldots, n$$

- It is a linear system of equations with J_μ as its unique solution.

Major DP Results

- $J^*(i)$ = Optimal cost starting from state i
- $J_\mu(i)$ = Optimal cost starting from state i using policy μ
- Bellman's equation:

$$J^*(i) = \min_{u \in U(i)} \sum_{j=1}^{n} p_{ij}(u)\big(g(i,u,j) + \alpha J^*(j)\big), \qquad i = 1, \ldots, n$$

A parametric fixed point equation.

- J^* is the unique fixed point.
- An optimal policy minimizes for each i in the RHS of Bellman's equation.
- Bellman's equation for a policy μ:

$$J_\mu(i) = \sum_{j=1}^{n} p_{ij}\big(\mu(i)\big)\big(g(i,\mu(i),j) + \alpha J_\mu(j)\big), \qquad i = 1, \ldots, n$$

- It is a linear system of equations with J_μ as its unique solution.

Shorthand Notation - Fixed Point View

- Denote by T and T_μ the mappings that map $J \in \Re^n$ to the vectors TJ and $T_\mu J$ with components

$$(TJ)(i) \stackrel{\text{def}}{=} \min_{u \in U(i)} \sum_{j=1}^{n} p_{ij}(u)\big(g(i,u,j) + \alpha J(j)\big), \qquad i = 1, \ldots, n,$$

and

$$(T_\mu J)(i) \stackrel{\text{def}}{=} \sum_{j=1}^{n} p_{ij}\big(\mu(i)\big)\big(g(i,\mu(i),j) + \alpha J(j)\big), \qquad i = 1, \ldots, n$$

- Bellman's equations are written as

$$J^* = TJ^* = \min_\mu T_\mu J^*, \qquad J_\mu = T_\mu J_\mu$$

- Key structure: T and T_μ are sup-norm contractions with modulus α,

$$\|TJ - TJ'\|_\infty = \max_{i=1,\ldots,n} \big|(TJ)(i) - (TJ')(i)\big| \le \alpha \max_{i=1,\ldots,n} \big|J(i) - J'(i)\big| = \alpha \|J - J'\|_\infty$$

Shorthand Notation - Fixed Point View

- Denote by T and T_μ the mappings that map $J \in \Re^n$ to the vectors TJ and $T_\mu J$ with components

$$(TJ)(i) \stackrel{\text{def}}{=} \min_{u \in U(i)} \sum_{j=1}^{n} p_{ij}(u)\big(g(i,u,j) + \alpha J(j)\big), \qquad i = 1, \ldots, n,$$

and

$$(T_\mu J)(i) \stackrel{\text{def}}{=} \sum_{j=1}^{n} p_{ij}\big(\mu(i)\big)\big(g(i,\mu(i),j) + \alpha J(j)\big), \qquad i = 1, \ldots, n$$

- Bellman's equations are written as

$$J^* = TJ^* = \min_\mu T_\mu J^*, \qquad J_\mu = T_\mu J_\mu$$

- Key structure: T and T_μ are sup-norm contractions with modulus α,

$$\|TJ - TJ'\|_\infty = \max_{i=1,\ldots,n} \big|(TJ)(i) - (TJ')(i)\big| \le \alpha \max_{i=1,\ldots,n} \big|J(i) - J'(i)\big| = \alpha \|J - J'\|_\infty$$

Shorthand Notation - Fixed Point View

- Denote by T and T_μ the mappings that map $J \in \Re^n$ to the vectors TJ and $T_\mu J$ with components

$$(TJ)(i) \stackrel{\text{def}}{=} \min_{u \in U(i)} \sum_{j=1}^{n} p_{ij}(u)\big(g(i,u,j) + \alpha J(j)\big), \qquad i = 1, \ldots, n,$$

and

$$(T_\mu J)(i) \stackrel{\text{def}}{=} \sum_{j=1}^{n} p_{ij}\big(\mu(i)\big)\big(g(i,\mu(i),j) + \alpha J(j)\big), \qquad i = 1, \ldots, n$$

- Bellman's equations are written as

$$J^* = TJ^* = \min_\mu T_\mu J^*, \qquad J_\mu = T_\mu J_\mu$$

- Key structure: T and T_μ are sup-norm contractions with modulus α,

$$\|TJ - TJ'\|_\infty = \max_{i=1,\ldots,n} \big|(TJ)(i) - (TJ')(i)\big| \leq \alpha \max_{i=1,\ldots,n} \big|J(i) - J'(i)\big| = \alpha \|J - J'\|_\infty$$

Major (Synchronous) Methods for Finding Fixed Point of T

- Value iteration (generic fixed point method): Start with any J^0, iterate by

$$J^{t+1} = TJ^t$$

- Policy iteration (special method for T of the form $T = \min_\mu T_\mu$): Start with any J^0 and μ^0. Given J^t and μ^t, iterate by:
    - Policy evaluation: $J^{t+1} = (T_{\mu^t})^m J^t$ (m applications of T_{μ^t} on J^t; $m = \infty$ is possible)
    - Policy improvement: μ^{t+1} attains the min in TJ^{t+1} (or $T_{\mu^{t+1}} J^{t+1} = TJ^{t+1}$)

- Both methods converge to J^*:
    - Value iteration, thanks to contraction of T
    - Policy iteration, thanks to contraction and monotonicity of T and T_μ

- Typically, policy iteration (with a reasonable choice of m) is more efficient because application of T_μ is cheaper than application of T

Major (Synchronous) Methods for Finding Fixed Point of T

- Value iteration (generic fixed point method): Start with any J^0, iterate by

$$J^{t+1} = TJ^t$$

- Policy iteration (special method for T of the form $T = \min_\mu T_\mu$): Start with any J^0 and μ^0. Given J^t and μ^t, iterate by:
  - Policy evaluation: $J^{t+1} = (T_{\mu^t})^m J^t$ (m applications of T_{μ^t} on J^t; $m = \infty$ is possible)
  - Policy improvement: μ^{t+1} attains the min in TJ^{t+1} (or $T_{\mu^{t+1}} J^{t+1} = TJ^{t+1}$)

- Both methods converge to J^*:
  - Value iteration, thanks to contraction of T
  - Policy iteration, thanks to contraction and monotonicity of T and T_μ

- Typically, policy iteration (with a reasonable choice of m) is more efficient because application of T_μ is cheaper than application of T

Major (Synchronous) Methods for Finding Fixed Point of T

- Value iteration (generic fixed point method): Start with any J^0, iterate by

$$J^{t+1} = TJ^t$$

- Policy iteration (special method for T of the form $T = \min_\mu T_\mu$): Start with any J^0 and μ^0. Given J^t and μ^t, iterate by:
  - Policy evaluation: $J^{t+1} = (T_{\mu^t})^m J^t$ (m applications of T_{μ^t} on J^t; $m = \infty$ is possible)
  - Policy improvement: μ^{t+1} attains the min in TJ^{t+1} (or $T_{\mu^{t+1}} J^{t+1} = TJ^{t+1}$)

- Both methods converge to J^*:
  - Value iteration, thanks to contraction of T
  - Policy iteration, thanks to contraction and monotonicity of T and T_μ

- Typically, policy iteration (with a reasonable choice of m) is more efficient because application of T_μ is cheaper than application of T

Major (Synchronous) Methods for Finding Fixed Point of T

- Value iteration (generic fixed point method): Start with any J^0, iterate by
$$J^{t+1} = TJ^t$$
- Policy iteration (special method for T of the form $T = \min_\mu T_\mu$): Start with any J^0 and μ^0. Given J^t and μ^t, iterate by:
    - Policy evaluation: $J^{t+1} = (T_{\mu^t})^m J^t$ (m applications of T_{μ^t} on J^t; $m = \infty$ is possible)
    - Policy improvement: μ^{t+1} attains the min in TJ^{t+1} (or $T_{\mu^{t+1}} J^{t+1} = TJ^{t+1}$)
- Both methods converge to J^*:
    - Value iteration, thanks to contraction of T
    - Policy iteration, thanks to contraction and monotonicity of T and T_μ
- Typically, policy iteration (with a reasonable choice of m) is more efficient because application of T_μ is cheaper than application of T

Major (Synchronous) Methods for Finding Fixed Point of T

- Value iteration (generic fixed point method): Start with any J^0, iterate by

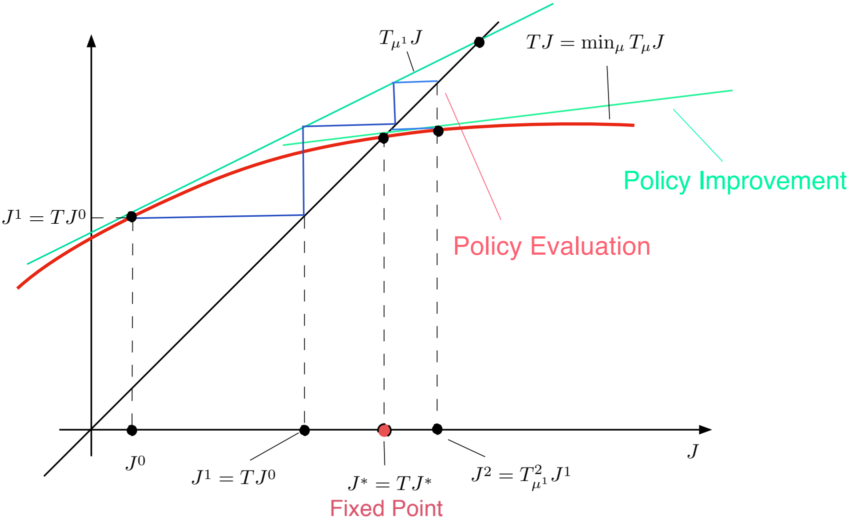
$$J^{t+1} = TJ^t$$

- Policy iteration (special method for T of the form $T = \min_\mu T_\mu$): Start with any J^0 and μ^0. Given J^t and μ^t, iterate by:
  - Policy evaluation: $J^{t+1} = (T_{\mu^t})^m J^t$ (m applications of T_{μ^t} on J^t; $m = \infty$ is possible)
  - Policy improvement: μ^{t+1} attains the min in TJ^{t+1} (or $T_{\mu^{t+1}} J^{t+1} = TJ^{t+1}$)

- Both methods converge to J^*:
  - Value iteration, thanks to contraction of T
  - Policy iteration, thanks to contraction and monotonicity of T and T_μ

- Typically, policy iteration (with a reasonable choice of m) is more efficient because application of T_μ is cheaper than application of T

Graphical Interpretation of Policy Iteration



Failure of Asynchronous Policy Iteration: W-B Example

Counterexample by Williams and Baird (1993)

- Deterministic discounted MDP with 6 states arranged in a circle
- 2 controls available in half the states, 1 control available in the other half
- Policy evaluations and improvements are one state at a time, no “delays"
- A cycle of 15 iterations is constructed that repeats the initial conditions

Failure of Asynchronous Policy Iteration: W-B Example

Counterexample by Williams and Baird (1993)

- Deterministic discounted MDP with 6 states arranged in a circle
- 2 controls available in half the states, 1 control available in the other half
- Policy evaluations and improvements are one state at a time, no “delays"
- A cycle of 15 iterations is constructed that repeats the initial conditions

Failure of Asynchronous Policy Iteration: W-B Example

Counterexample by Williams and Baird (1993)

- Deterministic discounted MDP with 6 states arranged in a circle
- 2 controls available in half the states, 1 control available in the other half
- Policy evaluations and improvements are one state at a time, no “delays"
- A cycle of 15 iterations is constructed that repeats the initial conditions

Failure of Asynchronous Policy Iteration: W-B Example

Counterexample by Williams and Baird (1993)

- Deterministic discounted MDP with 6 states arranged in a circle
- 2 controls available in half the states, 1 control available in the other half
- Policy evaluations and improvements are one state at a time, no “delays"
- A cycle of 15 iterations is constructed that repeats the initial conditions

Failure of Asynchronous Policy Iteration: W-B Example

Counterexample by Williams and Baird (1993)

- Deterministic discounted MDP with 6 states arranged in a circle
- 2 controls available in half the states, 1 control available in the other half
- Policy evaluations and improvements are one state at a time, no “delays"
- A cycle of 15 iterations is constructed that repeats the initial conditions

Outline

Rectifying the Difficulty - Q-Factors

- Q-factors, $Q(i,u)$ are functions of state-control pairs (i,u)
- The optimal Q-factors are given by

$$Q^*(i,u) = \sum_{j=1}^{n} p_{ij}(u)\big(g(i,u,j) + \alpha J^*(j)\big)$$

$Q^*(i,u)$: Cost of starting at i, using u first, then use optimal policy.

- Bellman's equation $J^* = TJ^*$ is written as

$$J^*(i) = \min_{u \in U(i)} Q^*(i,u)$$

- These two equations constitute a fixed point equation in (Q,J)
- For a fixed policy μ, the corresponding fixed point equation in (Q_μ, J_μ) is

$$Q_\mu(i,u) = \sum_{j=1}^{n} p_{ij}(u)\big(g(i,u,j) + \alpha Q_\mu(j,\mu(j))\big)$$

$$J_\mu(i) = Q_\mu(i,\mu(i))$$

Rectifying the Difficulty - Q-Factors

- Q-factors, $Q(i,u)$ are functions of state-control pairs (i,u)
- The optimal Q-factors are given by

$$Q^*(i,u) = \sum_{j=1}^{n} p_{ij}(u)\big(g(i,u,j) + \alpha J^*(j)\big)$$

$Q^*(i,u)$: Cost of starting at i, using u first, then use optimal policy.

- Bellman's equation $J^* = TJ^*$ is written as

$$J^*(i) = \min_{u \in U(i)} Q^*(i,u)$$

- These two equations constitute a fixed point equation in (Q,J)
- For a fixed policy μ, the corresponding fixed point equation in (Q_μ, J_μ) is

$$Q_\mu(i,u) = \sum_{j=1}^{n} p_{ij}(u)\big(g(i,u,j) + \alpha Q_\mu(j,\mu(j))\big)$$

$$J_\mu(i) = Q_\mu(i,\mu(i))$$

Rectifying the Difficulty - Q-Factors

- Q-factors, $Q(i,u)$ are functions of state-control pairs (i,u)
- The optimal Q-factors are given by

$$Q^*(i,u) = \sum_{j=1}^{n} p_{ij}(u)\big(g(i,u,j) + \alpha J^*(j)\big)$$

$Q^*(i,u)$: Cost of starting at i, using u first, then use optimal policy.

- Bellman's equation $J^* = TJ^*$ is written as

$$J^*(i) = \min_{u \in U(i)} Q^*(i,u)$$

- These two equations constitute a fixed point equation in (Q, J)
- For a fixed policy μ, the corresponding fixed point equation in (Q_μ, J_μ) is

$$Q_\mu(i,u) = \sum_{j=1}^{n} p_{ij}(u)\big(g(i,u,j) + \alpha Q_\mu(j,\mu(j))\big)$$

$$J_\mu(i) = Q_\mu(i,\mu(i))$$

Rectifying the Difficulty - Q-Factors

- Q-factors, $Q(i,u)$ are functions of state-control pairs (i,u)
- The optimal Q-factors are given by

$$Q^*(i,u) = \sum_{j=1}^{n} p_{ij}(u)\big(g(i,u,j) + \alpha J^*(j)\big)$$

$Q^*(i,u)$: Cost of starting at i, using u first, then use optimal policy.

- Bellman's equation $J^* = TJ^*$ is written as

$$J^*(i) = \min_{u \in U(i)} Q^*(i,u)$$

- These two equations constitute a fixed point equation in (Q, J)
- For a fixed policy μ, the corresponding fixed point equation in (Q_μ, J_μ) is

$$Q_\mu(i,u) = \sum_{j=1}^{n} p_{ij}(u)\big(g(i,u,j) + \alpha Q_\mu(j,\mu(j))\big)$$

$$J_\mu(i) = Q_\mu(i,\mu(i))$$

Sup-Norm Uniform Contraction Property

- Consider Q-factors $Q(i,u)$ and costs $J(i)$. For any μ, define mapping

$$(Q,J) \quad \mapsto \quad \big(F_\mu(Q,J), M_\mu(Q,J)\big)$$

where

$$F_\mu(Q,J)(i,u) \stackrel{\text{def}}{=} \sum_{j=1}^n p_{ij}(u)\big(g(i,u,j) + \alpha \min\big\{J(j), Q(j,\mu(j))\big\}\big),$$

$$M_\mu(Q,J)(i) \stackrel{\text{def}}{=} \min_{u\in U(i)} F_\mu(Q,J)(i,u)$$

- Key fact: A sup-norm contraction w/ common fixed point (Q^*, J^*) for all μ

$$\max\big\{\|F_\mu(Q,J)-Q^*\|_\infty, \|M_\mu(Q,J)-J^*\|_\infty\big\} \le \alpha \max\big\{\|Q-Q^*\|_\infty, \|J-J^*\|_\infty\big\}$$

- The asynchronous iteration for the fixed point problem

$$Q := F_\mu(Q,J), \qquad J := M_\mu(Q,J), \qquad \mu := \text{arbitrary}$$

is convergent.

- Even though we operate with different mappings F_μ and M_μ, they all have a common fixed point.

Sup-Norm Uniform Contraction Property

- Consider Q-factors $Q(i,u)$ and costs $J(i)$. For any μ, define mapping

$$(Q,J) \quad \mapsto \quad \big(F_\mu(Q,J), M_\mu(Q,J)\big)$$

where

$$F_\mu(Q,J)(i,u) \stackrel{\text{def}}{=} \sum_{j=1}^{n} p_{ij}(u)\Big(g(i,u,j) + \alpha \min\big\{J(j), Q(j,\mu(j))\big\}\Big),$$

$$M_\mu(Q,J)(i) \stackrel{\text{def}}{=} \min_{u\in U(i)} F_\mu(Q,J)(i,u)$$

- Key fact: A sup-norm contraction w/ common fixed point (Q^*, J^*) for all μ

$$\max\big\{\|F_\mu(Q,J)-Q^*\|_\infty, \|M_\mu(Q,J)-J^*\|_\infty\big\} \le \alpha \max\big\{\|Q-Q^*\|_\infty, \|J-J^*\|_\infty\big\}$$

- The asynchronous iteration for the fixed point problem

$$Q := F_\mu(Q,J), \qquad J := M_\mu(Q,J), \qquad \mu := \text{arbitrary}$$

is convergent.

- Even though we operate with different mappings F_μ and M_μ, they all have a common fixed point.

Sup-Norm Uniform Contraction Property

- Consider Q-factors $Q(i,u)$ and costs $J(i)$. For any μ, define mapping

$$(Q,J) \quad \mapsto \quad \big(F_\mu(Q,J), M_\mu(Q,J)\big)$$

where

$$F_\mu(Q,J)(i,u) \stackrel{\text{def}}{=} \sum_{j=1}^{n} p_{ij}(u)\big(g(i,u,j) + \alpha \min\{J(j), Q(j,\mu(j))\}\big),$$

$$M_\mu(Q,J)(i) \stackrel{\text{def}}{=} \min_{u \in U(i)} F_\mu(Q,J)(i,u)$$

- Key fact: A sup-norm contraction w/ common fixed point (Q^*, J^*) for all μ

$$\max\{\|F_\mu(Q,J)-Q^*\|_\infty, \|M_\mu(Q,J)-J^*\|_\infty\} \le \alpha \max\{\|Q-Q^*\|_\infty, \|J-J^*\|_\infty\}$$

- The asynchronous iteration for the fixed point problem

$$Q := F_\mu(Q,J), \qquad J := M_\mu(Q,J), \qquad \mu := \text{arbitrary}$$

is convergent.

- Even though we operate with different mappings F_μ and M_μ, they all have a common fixed point.

Sup-Norm Uniform Contraction Property

- Consider Q-factors $Q(i,u)$ and costs $J(i)$. For any μ, define mapping

$$(Q,J) \quad \mapsto \quad \big(F_\mu(Q,J), M_\mu(Q,J)\big)$$

where

$$F_\mu(Q,J)(i,u) \stackrel{\text{def}}{=} \sum_{j=1}^{n} p_{ij}(u)\Big(g(i,u,j) + \alpha \min\big\{J(j), Q(j,\mu(j))\big\}\Big),$$

$$M_\mu(Q,J)(i) \stackrel{\text{def}}{=} \min_{u\in U(i)} F_\mu(Q,J)(i,u)$$

- Key fact: A sup-norm contraction w/ common fixed point (Q^*, J^*) for all μ

$$\max\big\{\|F_\mu(Q,J)-Q^*\|_\infty,\ \|M_\mu(Q,J)-J^*\|_\infty\big\} \le \alpha \max\big\{\|Q-Q^*\|_\infty,\ \|J-J^*\|_\infty\big\}$$

- The asynchronous iteration for the fixed point problem

$$Q := F_\mu(Q,J), \qquad J := M_\mu(Q,J), \qquad \mu := \text{arbitrary}$$

is convergent.

- Even though we operate with different mappings F_μ and M_μ, they all have a common fixed point.

Asynchronous Policy Iteration: A More Detailed View

Asynchronous distributed policy iteration algorithm: Maintains J^t, μ^t, and V^t, where

$$V^t(i) = Q^t\big(i, \mu^t(i)\big) \qquad \approx \text{Q-factors of current policy}$$

- Processor i at time t does one of three things:
  - Does nothing
  - Or operates with F_{μ^t} at i:

    $$V^{t+1}(i) = \sum_{j=1}^{n} p_{ij}\big(\mu^t(i)\big)\Big(g\big(i, \mu^t(i), j\big) + \alpha \min\big\{J^t(j), V^t(j)\big\}\Big)$$

    and leaves $J^t(i)$, $\mu^t(i)$ unchanged.
  - Or operates with M_{μ^t} at i: Set

    $$J^{t+1}(i) = V^{t+1}(i) = \min_{u \in U(i)} \sum_{j=1}^{n} p_{ij}(u)\Big(g(i, u, j) + \alpha \min\big\{J^t(j), V^t(j)\big\}\Big)$$

    and sets $\mu^{t+1}(i)$ to a control that attains the minimum.
- Convergence follows by the asynchronous convergence theorem.

Asynchronous Policy Iteration: A More Detailed View

Asynchronous distributed policy iteration algorithm: Maintains J^t, μ^t, and V^t, where

$$V^t(i) = Q^t\big(i, \mu^t(i)\big) \qquad \approx \text{Q-factors of current policy}$$

- Processor i at time t does one of three things:
  - Does nothing
  - Or operates with F_{μ^t} at i:

$$V^{t+1}(i) = \sum_{j=1}^{n} p_{ij}(\mu^t(i))\big(g(i,\mu^t(i),j) + \alpha \min\{J^t(j), V^t(j)\}\big)$$

    and leaves $J^t(i)$, $\mu^t(i)$ unchanged.
  - Or operates with M_{μ^t} at i: Set

$$J^{t+1}(i) = V^{t+1}(i) = \min_{u \in U(i)} \sum_{j=1}^{n} p_{ij}(u)\big(g(i,u,j) + \alpha \min\{J^t(j), V^t(j)\}\big)$$

    and sets $\mu^{t+1}(i)$ to a control that attains the minimum.
- Convergence follows by the asynchronous convergence theorem.

Asynchronous Policy Iteration: A More Detailed View

Asynchronous distributed policy iteration algorithm: Maintains J^t, μ^t, and V^t, where

$$V^t(i) = Q^t\big(i, \mu^t(i)\big) \qquad \approx \text{Q-factors of current policy}$$

- Processor i at time t does one of three things:
  - Does nothing
  - Or operates with F_{μ^t} at i:

    $$V^{t+1}(i) = \sum_{j=1}^{n} p_{ij}(\mu^t(i))\big(g(i, \mu^t(i), j) + \alpha \min\{J^t(j), V^t(j)\}\big)$$

    and leaves $J^t(i)$, $\mu^t(i)$ unchanged.
  - Or operates with M_{μ^t} at i: Set

    $$J^{t+1}(i) = V^{t+1}(i) = \min_{u \in U(i)} \sum_{j=1}^{n} p_{ij}(u)\big(g(i, u, j) + \alpha \min\{J^t(j), V^t(j)\}\big)$$

    and sets $\mu^{t+1}(i)$ to a control that attains the minimum.
- Convergence follows by the asynchronous convergence theorem.

Asynchronous Policy Iteration: A More Detailed View

Asynchronous distributed policy iteration algorithm: Maintains J^t, μ^t, and V^t, where

$$V^t(i) = Q^t\big(i, \mu^t(i)\big) \qquad \approx \text{Q-factors of current policy}$$

- Processor i at time t does one of three things:
  - Does nothing
  - Or operates with F_{μ^t} at i:

$$V^{t+1}(i) = \sum_{j=1}^{n} p_{ij}\big(\mu^t(i)\big)\Big(g\big(i, \mu^t(i), j\big) + \alpha \min\big\{J^t(j), V^t(j)\big\}\Big)$$

    and leaves $J^t(i)$, $\mu^t(i)$ unchanged.
  - Or operates with M_{μ^t} at i: Set

$$J^{t+1}(i) = V^{t+1}(i) = \min_{u \in U(i)} \sum_{j=1}^{n} p_{ij}(u)\Big(g(i, u, j) + \alpha \min\big\{J^t(j), V^t(j)\big\}\Big)$$

    and sets $\mu^{t+1}(i)$ to a control that attains the minimum.
- Convergence follows by the asynchronous convergence theorem.

Asynchronous Policy Iteration: A More Detailed View

Asynchronous distributed policy iteration algorithm: Maintains J^t, μ^t, and V^t, where

$$V^t(i) = Q^t\big(i, \mu^t(i)\big) \qquad \approx \text{Q-factors of current policy}$$

- Processor i at time t does one of three things:
  - Does nothing
  - Or operates with F_{μ^t} at i:

    $$V^{t+1}(i) = \sum_{j=1}^{n} p_{ij}\big(\mu^t(i)\big)\Big(g\big(i, \mu^t(i), j\big) + \alpha \min\big\{J^t(j), V^t(j)\big\}\Big)$$

    and leaves $J^t(i)$, $\mu^t(i)$ unchanged.
  - Or operates with M_{μ^t} at i: Set

    $$J^{t+1}(i) = V^{t+1}(i) = \min_{u \in U(i)} \sum_{j=1}^{n} p_{ij}(u)\Big(g(i, u, j) + \alpha \min\big\{J^t(j), V^t(j)\big\}\Big)$$

    and sets $\mu^{t+1}(i)$ to a control that attains the minimum.
- Convergence follows by the asynchronous convergence theorem.

Asynchronous Policy Iteration: A More Detailed View

Asynchronous distributed policy iteration algorithm: Maintains J^t, μ^t, and V^t, where

$$V^t(i) = Q^t\big(i, \mu^t(i)\big) \qquad \approx \text{Q-factors of current policy}$$

- Processor i at time t does one of three things:
  - Does nothing
  - Or operates with F_{μ^t} at i:

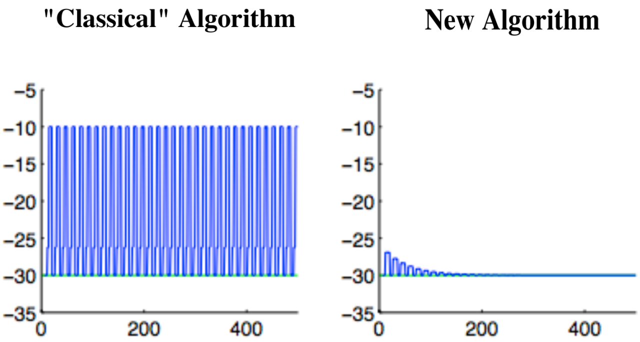
$$V^{t+1}(i) = \sum_{j=1}^{n} p_{ij}\big(\mu^t(i)\big)\Big(g\big(i, \mu^t(i), j\big) + \alpha \min\big\{J^t(j), V^t(j)\big\}\Big)$$

    and leaves $J^t(i)$, $\mu^t(i)$ unchanged.
  - Or operates with M_{μ^t} at i: Set

$$J^{t+1}(i) = V^{t+1}(i) = \min_{u \in U(i)} \sum_{j=1}^{n} p_{ij}(u)\Big(g(i, u, j) + \alpha \min\big\{J^t(j), V^t(j)\big\}\Big)$$

    and sets $\mu^{t+1}(i)$ to a control that attains the minimum.
- Convergence follows by the asynchronous convergence theorem.

Williams and Baird Counterexample



Outline

Generalized Mappings T and T_μ

- The preceding analysis uses only the contraction property of the discounted MDP (not monotonicity or the probabilistic structure)
- Abstract mappings T and T_μ:
  - Introduce a mapping $H(i,u,J)$ and denote
  $$(TJ)(i) = \min_{u \in U(i)} H(i,u,J), \qquad (T_\mu J)(i) = H(i,\mu(i),J)$$
  i.e., $TJ = \min_\mu T_\mu J$, where the min is taken separately for each component
  - Assume that for all i and $u \in U(i)$
  $$\big|H(i,u,J) - H(i,u,J')\big| \leq \alpha \|J - J'\|_\infty$$
- Asynchronous algorithm: At time t, processor i does one of three things:
  - Does nothing
  - Or does a "policy evaluation" at i: Sets
  $$V^{t+1}(i) = H\big(i, \mu^t(i), \min\{J^t, V^t\}\big)$$
  and leaves $J^t(i)$, $\mu^t(i)$ unchanged.
  - Or does a "policy improvement" at i: Sets
  $$J^{t+1}(i) = V^{t+1}(i) = \min_{u \in U(i)} H\big(i, u, \min\{J^t, V^t\}\big)$$
  sets $\mu^{t+1}(i)$ to a u that attains the minimum.

Generalized Mappings T and T_μ

- The preceding analysis uses only the contraction property of the discounted MDP (not monotonicity or the probabilistic structure)
- Abstract mappings T and T_μ:
  - Introduce a mapping $H(i, u, J)$ and denote

$$(TJ)(i) = \min_{u \in U(i)} H(i, u, J), \qquad (T_\mu J)(i) = H(i, \mu(i), J)$$

    i.e., $TJ = \min_\mu T_\mu J$, where the min is taken separately for each component
  - Assume that for all i and $u \in U(i)$

$$\left| H(i, u, J) - H(i, u, J') \right| \leq \alpha \|J - J'\|_\infty$$

- Asynchronous algorithm: At time t, processor i does one of three things:
  - Does nothing
  - Or does a "policy evaluation" at i: Sets

$$V^{t+1}(i) = H\big(i, \mu^t(i), \min\{J^t, V^t\}\big)$$

    and leaves $J^t(i)$, $\mu^t(i)$ unchanged.
  - Or does a "policy improvement" at i: Sets

$$J^{t+1}(i) = V^{t+1}(i) = \min_{u \in U(i)} H\big(i, u, \min\{J^t, V^t\}\big)$$

    sets $\mu^{t+1}(i)$ to a u that attains the minimum.

Generalized Mappings T and T_μ

- The preceding analysis uses only the contraction property of the discounted MDP (not monotonicity or the probabilistic structure)
- Abstract mappings T and T_μ:
  - Introduce a mapping $H(i,u,J)$ and denote

$$(TJ)(i) = \min_{u \in U(i)} H(i,u,J), \qquad (T\mu J)(i) = H\big(i,\mu(i),J\big)$$

    i.e., $TJ = \min_\mu T_\mu J$, where the min is taken separately for each component
  - Assume that for all i and $u \in U(i)$

$$\big|H(i,u,J) - H(i,u,J')\big| \le \alpha \|J - J'\|_\infty$$

- Asynchronous algorithm: At time t, processor i does one of three things:
  - Does nothing
  - Or does a "policy evaluation" at i: Sets

$$V^{t+1}(i) = H\big(i, \mu^t(i), \min\{J^t, V^t\}\big)$$

    and leaves $J^t(i)$, $\mu^t(i)$ unchanged.
  - Or does a "policy improvement" at i: Sets

$$J^{t+1}(i) = V^{t+1}(i) = \min_{u \in U(i)} H\big(i, u, \min\{J^t, V^t\}\big)$$

    sets $\mu^{t+1}(i)$ to a u that attains the minimum.

Generalized Mappings T and T_μ

- The preceding analysis uses only the contraction property of the discounted MDP (not monotonicity or the probabilistic structure)
- Abstract mappings T and T_μ:
  - Introduce a mapping $H(i, u, J)$ and denote

$$(TJ)(i) = \min_{u \in U(i)} H(i, u, J), \qquad (T_\mu J)(i) = H\big(i, \mu(i), J\big)$$

    i.e., $TJ = \min_\mu T_\mu J$, where the min is taken separately for each component
  - Assume that for all i and $u \in U(i)$

$$\big|H(i, u, J) - H(i, u, J')\big| \leq \alpha \|J - J'\|_\infty$$

- Asynchronous algorithm: At time t, processor i does one of three things:
  - Does nothing
  - Or does a "policy evaluation" at i: Sets

$$V^{t+1}(i) = H\big(i, \mu^t(i), \min\{J^t, V^t\}\big)$$

    and leaves $J^t(i)$, $\mu^t(i)$ unchanged.
  - Or does a "policy improvement" at i: Sets

$$J^{t+1}(i) = V^{t+1}(i) = \min_{u \in U(i)} H\big(i, u, \min\{J^t, V^t\}\big)$$

    sets $\mu^{t+1}(i)$ to a u that attains the minimum.

Generalized Mappings T and T_μ

- The preceding analysis uses only the contraction property of the discounted MDP (not monotonicity or the probabilistic structure)
- Abstract mappings T and T_μ:
  - Introduce a mapping $H(i, u, J)$ and denote
$$(TJ)(i) = \min_{u \in U(i)} H(i, u, J), \qquad (T\mu J)(i) = H\big(i, \mu(i), J\big)$$
    i.e., $TJ = \min_\mu T_\mu J$, where the min is taken separately for each component
  - Assume that for all i and $u \in U(i)$
$$\big|H(i, u, J) - H(i, u, J')\big| \le \alpha \|J - J'\|_\infty$$
- Asynchronous algorithm: At time t, processor i does one of three things:
  - Does nothing
  - Or does a "policy evaluation" at i: Sets
$$V^{t+1}(i) = H\big(i, \mu^t(i), \min\{J^t, V^t\}\big)$$
    and leaves $J^t(i)$, $\mu^t(i)$ unchanged.
  - Or does a "policy improvement" at i: Sets
$$J^{t+1}(i) = V^{t+1}(i) = \min_{u \in U(i)} H\big(i, u, \min\{J^t, V^t\}\big)$$
    sets $\mu^{t+1}(i)$ to a u that attains the minimum.

Generalized Mappings T and T_μ

- The preceding analysis uses only the contraction property of the discounted MDP (not monotonicity or the probabilistic structure)
- Abstract mappings T and T_μ:
  - Introduce a mapping $H(i,u,J)$ and denote
  $$(TJ)(i) = \min_{u \in U(i)} H(i,u,J), \qquad (T\mu J)(i) = H\big(i, \mu(i), J\big)$$
  i.e., $TJ = \min_\mu T_\mu J$, where the min is taken separately for each component
  - Assume that for all i and $u \in U(i)$
  $$\big|H(i,u,J) - H(i,u,J')\big| \le \alpha \|J - J'\|_\infty$$
- Asynchronous algorithm: At time t, processor i does one of three things:
  - Does nothing
  - Or does a "policy evaluation" at i: Sets
  $$V^{t+1}(i) = H\big(i, \mu^t(i), \min\{J^t, V^t\}\big)$$
  and leaves $J^t(i)$, $\mu^t(i)$ unchanged.
  - Or does a "policy improvement" at i: Sets
  $$J^{t+1}(i) = V^{t+1}(i) = \min_{u \in U(i)} H\big(i, u, \min\{J^t, V^t\}\big)$$
  sets $\mu^{t+1}(i)$ to a u that attains the minimum.

Generalized Mappings T and T_μ

- The preceding analysis uses only the contraction property of the discounted MDP (not monotonicity or the probabilistic structure)
- Abstract mappings T and T_μ:
  - Introduce a mapping $H(i, u, J)$ and denote

    $$(TJ)(i) = \min_{u \in U(i)} H(i, u, J), \qquad (T\mu J)(i) = H\big(i, \mu(i), J\big)$$

    i.e., $TJ = \min_\mu T_\mu J$, where the min is taken separately for each component
  - Assume that for all i and $u \in U(i)$

    $$\big|H(i, u, J) - H(i, u, J')\big| \leq \alpha \|J - J'\|_\infty$$

- Asynchronous algorithm: At time t, processor i does one of three things:
  - Does nothing
  - Or does a "policy evaluation" at i: Sets

    $$V^{t+1}(i) = H\big(i, \mu^t(i), \min\{J^t, V^t\}\big)$$

    and leaves $J^t(i)$, $\mu^t(i)$ unchanged.
  - Or does a "policy improvement" at i: Sets

    $$J^{t+1}(i) = V^{t+1}(i) = \min_{u \in U(i)} H\big(i, u, \min\{J^t, V^t\}\big)$$

    sets $\mu^{t+1}(i)$ to a u that attains the minimum.

Generalized Mappings T and T_μ

- The preceding analysis uses only the contraction property of the discounted MDP (not monotonicity or the probabilistic structure)
- Abstract mappings T and T_μ:
  - Introduce a mapping $H(i, u, J)$ and denote

    $$(TJ)(i) = \min_{u \in U(i)} H(i, u, J), \qquad (T\mu J)(i) = H\big(i, \mu(i), J\big)$$

    i.e., $TJ = \min_\mu T_\mu J$, where the min is taken separately for each component
  - Assume that for all i and $u \in U(i)$

    $$\big|H(i, u, J) - H(i, u, J')\big| \le \alpha \|J - J'\|_\infty$$

- Asynchronous algorithm: At time t, processor i does one of three things:
  - Does nothing
  - Or does a “policy evaluation" at i: Sets

    $$V^{t+1}(i) = H\big(i, \mu^t(i), \min\{J^t, V^t\}\big)$$

    and leaves $J^t(i)$, $\mu^t(i)$ unchanged.
  - Or does a “policy improvement" at i: Sets

    $$J^{t+1}(i) = V^{t+1}(i) = \min_{u \in U(i)} H\big(i, u, \min\{J^t, V^t\}\big)$$

    sets $\mu^{t+1}(i)$ to a u that attains the minimum.

DP Applications with Generalized T and T_μ

- DP models beyond discounted with standard policy evaluation
  - Optimistic/modified policy iteration for semi-Markov and minimax discounted problems
  - Stochastic shortest path problems
  - Semi-Markov decision problems
  - Stochastic zero sum games
  - Sequential minimax problems
- Multi-agent aggregation
  - Each agent updates costs at all states within a subset
  - Each agent uses detailed costs for the local states and aggregate costs for other states, as communicated by other agents

DP Applications with Generalized T and T_μ

- DP models beyond discounted with standard policy evaluation
  - Optimistic/modified policy iteration for semi-Markov and minimax discounted problems
  - Stochastic shortest path problems
  - Semi-Markov decision problems
  - Stochastic zero sum games
  - Sequential minimax problems
- Multi-agent aggregation
  - Each agent updates costs at all states within a subset
  - Each agent uses detailed costs for the local states and aggregate costs for other states, as communicated by other agents

DP Applications with Generalized T and T_μ

- DP models beyond discounted with standard policy evaluation
  - Optimistic/modified policy iteration for semi-Markov and minimax discounted problems
  - Stochastic shortest path problems
  - Semi-Markov decision problems
  - Stochastic zero sum games
  - Sequential minimax problems
- Multi-agent aggregation
  - Each agent updates costs at all states within a subset
  - Each agent uses detailed costs for the local states and aggregate costs for other states, as communicated by other agents

DP Applications with Generalized T and T_μ

- DP models beyond discounted with standard policy evaluation
  - Optimistic/modified policy iteration for semi-Markov and minimax discounted problems
  - Stochastic shortest path problems
  - Semi-Markov decision problems
  - Stochastic zero sum games
  - Sequential minimax problems
- Multi-agent aggregation
  - Each agent updates costs at all states within a subset
  - Each agent uses detailed costs for the local states and aggregate costs for other states, as communicated by other agents

DP Applications with Generalized T and T_μ

- DP models beyond discounted with standard policy evaluation
  - Optimistic/modified policy iteration for semi-Markov and minimax discounted problems
  - Stochastic shortest path problems
  - Semi-Markov decision problems
  - Stochastic zero sum games
  - Sequential minimax problems
- Multi-agent aggregation
  - Each agent updates costs at all states within a subset
  - Each agent uses detailed costs for the local states and aggregate costs for other states, as communicated by other agents

DP Applications with Generalized T and T_μ

- DP models beyond discounted with standard policy evaluation
  - Optimistic/modified policy iteration for semi-Markov and minimax discounted problems
  - Stochastic shortest path problems
  - Semi-Markov decision problems
  - Stochastic zero sum games
  - Sequential minimax problems
- Multi-agent aggregation
  - Each agent updates costs at all states within a subset
  - Each agent uses detailed costs for the local states and aggregate costs for other states, as communicated by other agents

DP Applications with Generalized T and T_μ

- DP models beyond discounted with standard policy evaluation
  - Optimistic/modified policy iteration for semi-Markov and minimax discounted problems
  - Stochastic shortest path problems
  - Semi-Markov decision problems
  - Stochastic zero sum games
  - Sequential minimax problems
- Multi-agent aggregation
  - Each agent updates costs at all states within a subset
  - Each agent uses detailed costs for the local states and aggregate costs for other states, as communicated by other agents

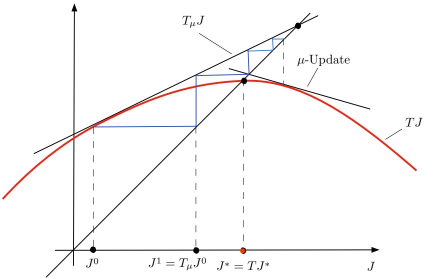
DP Applications with Generalized T and T_μ

- DP models beyond discounted with standard policy evaluation
  - Optimistic/modified policy iteration for semi-Markov and minimax discounted problems
  - Stochastic shortest path problems
  - Semi-Markov decision problems
  - Stochastic zero sum games
  - Sequential minimax problems
- Multi-agent aggregation
  - Each agent updates costs at all states within a subset
  - Each agent uses detailed costs for the local states and aggregate costs for other states, as communicated by other agents

DP Applications with Generalized T and T_μ

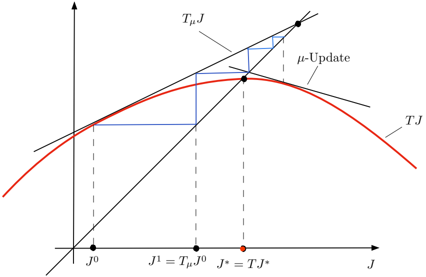
- DP models beyond discounted with standard policy evaluation
  - Optimistic/modified policy iteration for semi-Markov and minimax discounted problems
  - Stochastic shortest path problems
  - Semi-Markov decision problems
  - Stochastic zero sum games
  - Sequential minimax problems
- Multi-agent aggregation
  - Each agent updates costs at all states within a subset
  - Each agent uses detailed costs for the local states and aggregate costs for other states, as communicated by other agents

Fixed Points of Concave Sup-Norm Contractions



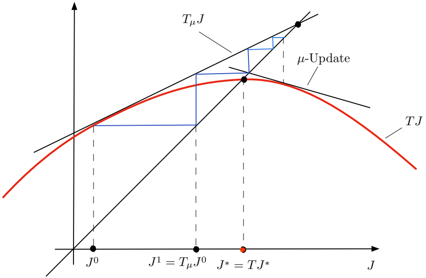
- Find a fixed point of a mapping $T : \Re^n \mapsto \Re^n$, where the components $T_i(\cdot) : \Re^n \mapsto \Re$ are concave sup-norm contractions
- T is the minimum of a collection of linear mappings T_μ involving the concave conjugate functions of the concave functions T_i
- An asynchronous parametric fixed point algorithm can be used:
  - Linearize at the current J
  - Iterate using the linearized mapping
  - Update the linearization
- All of the above are done asynchronously

Fixed Points of Concave Sup-Norm Contractions



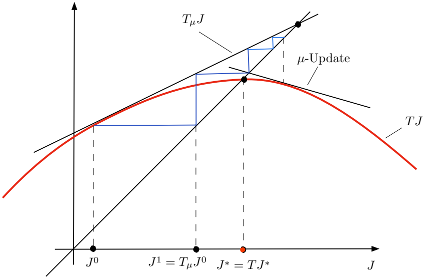
- Find a fixed point of a mapping $T : \Re^n \mapsto \Re^n$, where the components $T_i(\cdot) : \Re^n \mapsto \Re$ are concave sup-norm contractions
- T is the minimum of a collection of linear mappings T_μ involving the concave conjugate functions of the concave functions T_i
- An asynchronous parametric fixed point algorithm can be used:
  - Linearize at the current J
  - Iterate using the linearized mapping
  - Update the linearization
- All of the above are done asynchronously

Fixed Points of Concave Sup-Norm Contractions



- Find a fixed point of a mapping $T : \Re^n \mapsto \Re^n$, where the components $T_i(\cdot) : \Re^n \mapsto \Re$ are concave sup-norm contractions
- T is the minimum of a collection of linear mappings T_μ involving the concave conjugate functions of the concave functions T_i
- An asynchronous parametric fixed point algorithm can be used:
  - Linearize at the current J
  - Iterate using the linearized mapping
  - Update the linearization
- All of the above are done asynchronously

Fixed Points of Concave Sup-Norm Contractions



- Find a fixed point of a mapping $T : \Re^n \mapsto \Re^n$, where the components $T_i(\cdot) : \Re^n \mapsto \Re$ are concave sup-norm contractions
- T is the minimum of a collection of linear mappings T_μ involving the concave conjugate functions of the concave functions T_i
- An asynchronous parametric fixed point algorithm can be used:
  - Linearize at the current J
  - Iterate using the linearized mapping
  - Update the linearization
- All of the above are done asynchronously

Concluding Remarks

- Asynchronous policy iteration has fragile convergence properties
- We have provided a new approach to correct the difficulties
- Key idea: Embed the problem into one that involves a uniform sup-norm contraction
- Extensions to generalized DP models involving:
  - Other DP models (e.g., stochastic shortest path, games, etc)
  - Fixed point problems for nonDP mappings of the form $T = min_{\mu\in\mathcal{M}} T_\mu$, such as concave sup-norm contractions

Concluding Remarks

- Asynchronous policy iteration has fragile convergence properties
- We have provided a new approach to correct the difficulties
- Key idea: Embed the problem into one that involves a uniform sup-norm contraction
- Extensions to generalized DP models involving:
  - Other DP models (e.g., stochastic shortest path, games, etc)
  - Fixed point problems for nonDP mappings of the form $T = min_{\mu \in \mathcal{M}} T_\mu$, such as concave sup-norm contractions

Concluding Remarks

- Asynchronous policy iteration has fragile convergence properties
- We have provided a new approach to correct the difficulties
- Key idea: Embed the problem into one that involves a uniform sup-norm contraction
- Extensions to generalized DP models involving:
  - Other DP models (e.g., stochastic shortest path, games, etc)
  - Fixed point problems for nonDP mappings of the form $T = min_{\mu \in \mathcal{M}} T_\mu$, such as concave sup-norm contractions

Concluding Remarks

- Asynchronous policy iteration has fragile convergence properties
- We have provided a new approach to correct the difficulties
- Key idea: Embed the problem into one that involves a uniform sup-norm contraction
- Extensions to generalized DP models involving:
  - Other DP models (e.g., stochastic shortest path, games, etc)
  - Fixed point problems for nonDP mappings of the form $T = min_{\mu \in \mathcal{M}} T_\mu$, such as concave sup-norm contractions

Thank You!