北京科技大学
University of Science
and Technology Beijing

自动化学院
School of Automation
and Electrical Engineering

自适应动态规划最优控制方法

刘德荣

2016年11月6日

11月15日截稿

Call for Papers

The annual International Joint Conference on Neural Networks (IJCNN) is the flagship conference of the IEEE Computational Intelligence Society and the International Neural Network Society. It covers a wide range of topics in the field of neural networks, from biological neural network modeling to artificial neural computation.

Guidelines for Paper Submission

http://www.ijcnn.org/paper-submission

Important Dates

- Special session & competition proposals submission: September 15, 2016
- Tutorial and workshop proposal submission: ~~October 15, 2016~~ October 30, 2016
- Paper submission: November 15, 2016
- Paper decision notification: January 20, 2017
- Camera-ready submission: February 20, 2017

Topics Covered

NEURAL NETWORK MODELS

Follow Us!

IJCNN 2017

IJCNN 2017

Sponsoring Organizations

INNS - International Neural Network Society

1月1日截稿

Call for Papers

https://conference.cs.cityu.edu.hk/isnn/

Honorary Chair
Shun'ichi Amari, RIKEN Brain Science Institute, Tokyo, Japan

General Chairs
Hidenori Kawamura, Hokkaido University, Sapporo, Japan
Jun Wang, City University of Hong Kong, Hong Kong

Advisory Chairs
Kunihiko Fukushima and Takeshi Yamakawa, Fuzzy Logic Systems Institute, Fukuoka, Japan

Steering Chairs
Haibo He, University of Rhode Island, Kingston, USA
Derong Liu, University of Illinois – Chicago, Chicago, USA
Jun Wang, City University of Hong Kong, Hong Kong

Organizing Committee Chairs
Andrzej Cichocki, RIKEN Brain Science Institute, Tokyo, Japan
Min Han, Dalian University of Technology, Dalian, China
Bao-Liang Lu, Shanghai Jiao Tong University, Shanghai, China
Masahito Yamamoto, Hokkaido University, Sapporo, Japan

Program Chairs
Fengyu Cong, Dalian University of Technology, Dalian, China
Andrew C.-S. Leung, City University of Hong Kong, Hong Kong
Qinglai Wei, CAS Institute of Automation, Beijing, China

Special Sessions Chairs
Long Cheng, CAS Institute of Automation, Beijing, China
Satoshi Kurihara, Univ. of Electro-Communications, Tokyo, Japan

Sponsors/organizers: Hokkaido University and City University of Hong Kong
Technical co-sponsors: Asia Pacific Neural Network Society, IEEE Computational Intelligence Society, International Neural Network Society, and Japanese Neural Network Society

Following the successes of previous events, the 14th International Symposium on Neural Networks (ISNN 2017) will be held in Sapporo, Hokkaido, Japan. Located in northern island of Hokkaido, Sapporo is the fourth largest Japanese city and a popular summer/winter tourist venue. ISNN 2017 aims to provide a high-level international forum for scientists, engineers, and educators to present the state of the art of neural network research and applications in related fields. The symposium will feature plenary speeches given by world renowned scholars, regular sessions with broad coverage, and special sessions focusing on popular topics.

Call for Papers and Special Sessions

Prospective authors are invited to contribute high-quality papers to ISNN 2017. In addition, proposals for special sessions within the technical scopes of the symposium are solicited. Special sessions, to be organized by internationally recognized experts, aim to bring together researchers in special focused topics. Papers submitted for special sessions are to be peer-reviewed with the same criteria used for the contributed papers. Researchers interested in organizing special sessions are invited to submit formal proposals to ISNN 2017. A special session proposal should include the session title, a brief description of the scope and motivation, names, contact information and brief biographical information of the organizers.

Topic Areas

Topics areas include, but not limited to, computational neuroscience, connectionist theory and cognitive science, mathematical modeling of neural systems, neurodynamic analysis, neurodynamic optimization and

General Chair
Derong Liu, China

5月份截稿

大会总主席：
刘德荣

CALL FOR PAPERS

The 24th International Conference on Neural Information Processing will be held in Guangzhou, China, November 14–18, 2017. It is an annual event, organized since 1994 by the Asia Pacific Neural Network Society (APNNS, previously APNNA).

Guangzhou is the capital and largest city of Guangdong province, People's Republic of China. Located on the Pearl River, about 120 km (75 mi) north-northwest of Hong Kong and north-northeast of Macau, Guangzhou is a key national transportation hub and trading port. It is one of the five National Central Cities and holds sub-provincial administrative status. With a tropical climate, Guangzhou is warm all year long, and November is the best month over the year to visit Guangzhou with comfortable weather. There are so many snacks, delicious dim sum, great sea food and Cantonese cuisine. There is a great deal of tourist sites in Guangzhou, such as Beijing Road Walking Street, Haizhu Square, Sun Yat-sen's Memorial Hall, Yuexiu Park, Zhongxin Square, White Cloud Mountain, Canton Tower and so on. ICONIP 2017 aims to provide a high-level international forum for scientists, researchers, educators, industrial professionals, and students worldwide to present state-of-the-art research results, address new challenges, and discuss trends in neural information processing and applications. ICONIP 2017 invites scholars in all areas of neural network theory and applications, computational neuroscience, machine learning and others. In addition to regular technical sessions with oral and poster presentations, the conference program will include special sessions and tutorials on topics of current interest.

ICONIP 2017 features plenary/keynote and panel discussion sessions by world leading researchers as

Dynamic programming + Learning control

Dynamic Programming 1953

Curse-of-dimensionality - 1957

ADP - 1977

Adaptive dynamic programming
Approximate dynamic programming
Asymptotic dynamic programming
Neural dynamic programming
Adaptive critic designs

RL - 1984

Reinforcement learning
Richard Sutton (1984) *Temporal Credit Assignment in Reinforcement Learning* (PhD thesis). University of Massachusetts.

ADPRL

ADP for self-learning control

ADP for Self-Learning Control

- Dynamic Programming
- Adaptive Dynamic Programming
- Iterative ADP
- New Developments
- Concluding Remarks

Dynamic programming

- **Consider**

$$x_k = F(x_k, u_k)$$

  – $x_k \in X \subset R^n$ is the state

  – $u_k \in A \subset R^p$ is the control vector.

- **Performance index (or cost to go)**

$$J(x_k) = \sum_{i=k}^{\infty} \gamma^{i-k} U(x_i, u_i)$$

– U is the given utility function

– γ is the discount factor with $0 < \gamma \le 1$

– Infinite horizon problems.

Examples of utility functions

A Production output

B Energy usage

C Emissions

D Tracking control errors

There are many similar examples in practice

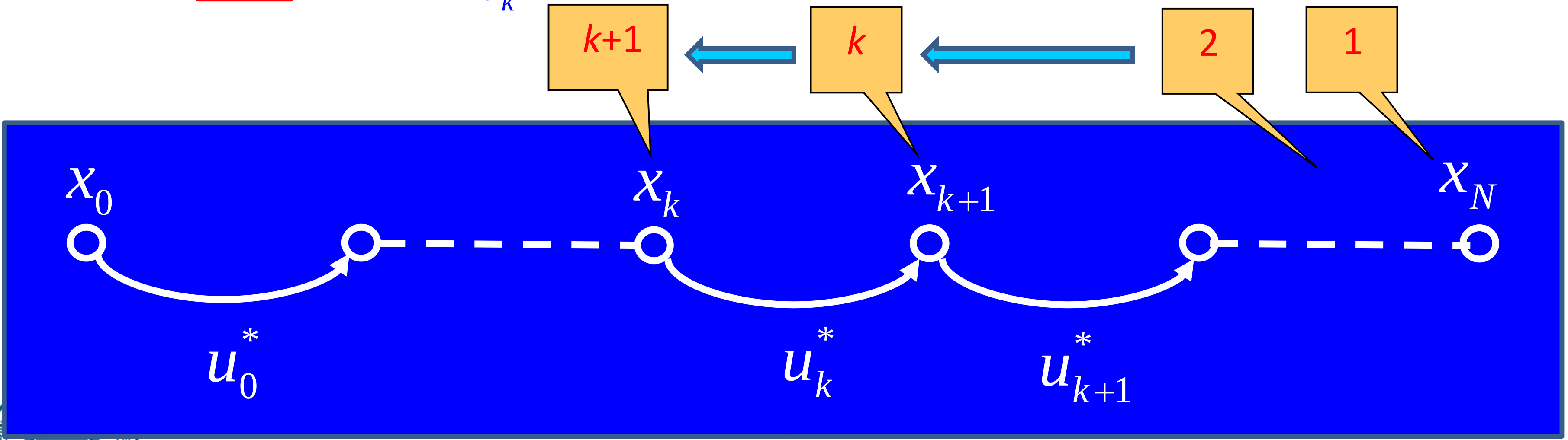
Bellman Equation

- **According to Bellman's principle of optimality,**

$$\boxed{J^*(x_k) = \min_{u_k}\{U(x_k, u_k) + \gamma J^*(x_{k+1})\}.}$$

–The optimal control u_k^* at time k is the u_k that achieves this minimum, i.e.,

$$\boxed{u_k^*} = \arg\min_{u_k}\{U(x_k, u_k) + \gamma J^*(x_{k+1})\}.$$



Issues with dynamic programming

- **Dynamic programming is applicable to problems that minimize/maximize a cost**
  - Applicable to many engineering problems
  - Model-based
- **Backward numerical process**
  - **Backward in time**
  - **Unknown function *J***

Curse of dimensionality

$$J(x_k) = \sum_{i=k}^{\infty} \gamma^{i-k} U(x_i, u_i)$$

- **Computational complexity increases exponentially as the number of variables**
  - Only suitable for small problems in practice

ADP for Self-Learning Control

- Dynamic Programming
- Adaptive Dynamic Programming
- Iterative ADP
- New Developments
- Concluding Remarks

Adaptive dynamic programming

Adaptive Dynamic Programming

Approximate Dynamic Programming

ADP

Asymptotic Dynamic Programming

Adaptive Critic Designs

Neurodynamic Programming

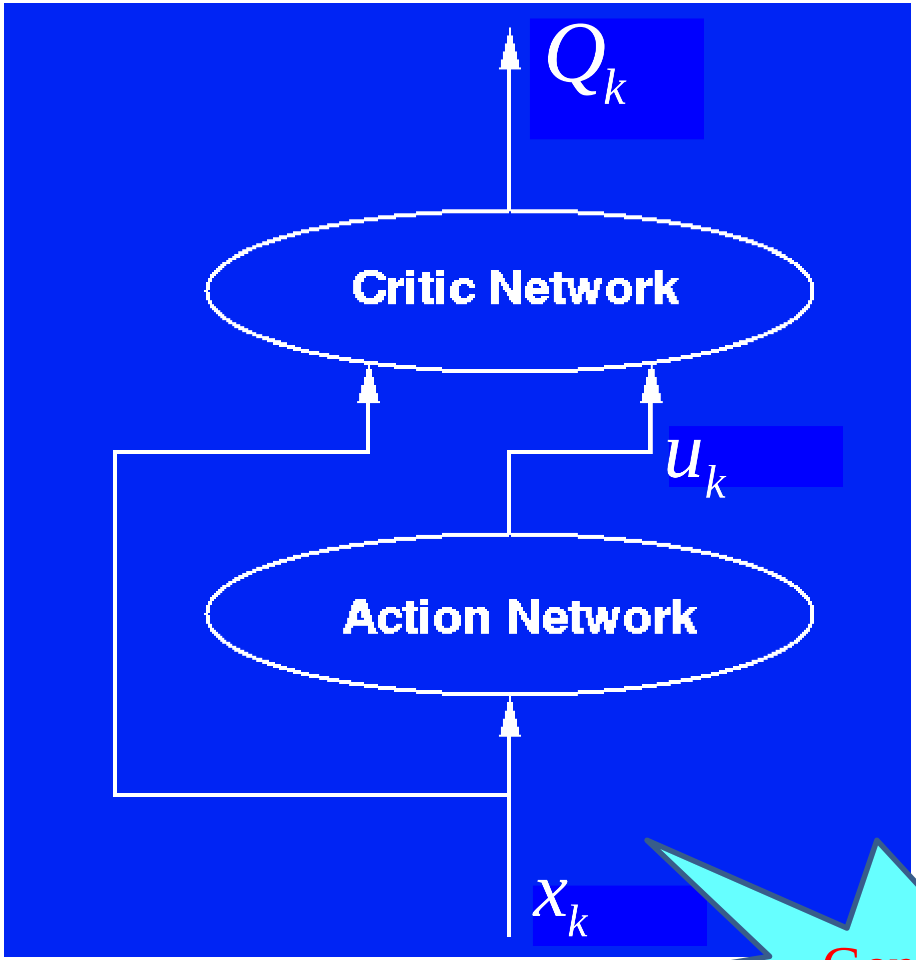
Relaxed Dynamic Programming

Reinforcement Learning

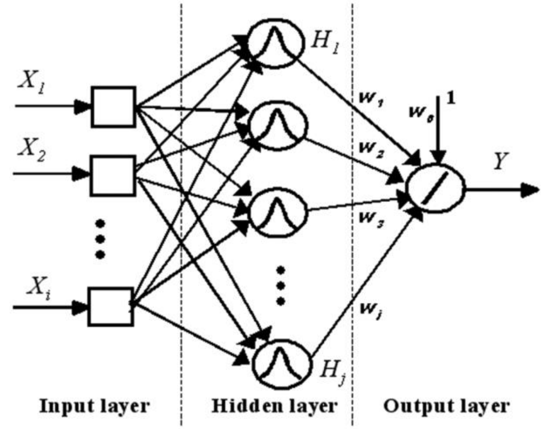
RL

To find solutions for dynamic programming

Adaptive dynamic programming

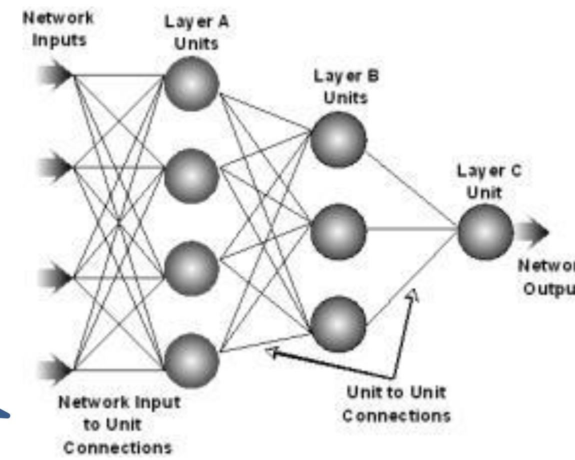


$$Q_k \approx J_{k+1}$$



Model-free HDP

Generate u_k to minimize Q_k



Work done before 1997

Widrow et al.

“Punish/Reward: Learning with a Critic in Adaptive Threshold Systems,” IEEE Transactions on Systems, Man, and Cybernetics (1973)

Paul Werbos

“Advanced Forecasting Methods for Global Crisis Warning and Models of Intelligence,” General Systems Yearbook (1977)

Prokhorov/Wunsch

“Adaptive Critic Designs,” IEEE Transactions on Neural Networks (1997)

Summarized all works before 1997.

Theoretical issues to be resolved

Stability

There was no proof for the control algorithms to be stable.

Convergence

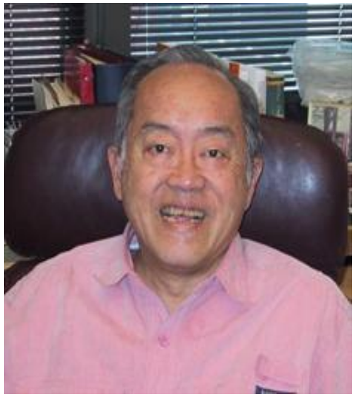
Whether the training/ learning process will converge is an open question.

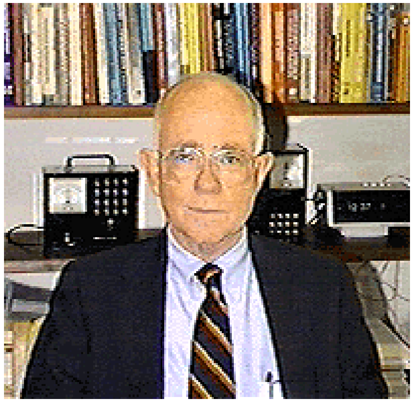
Optimality

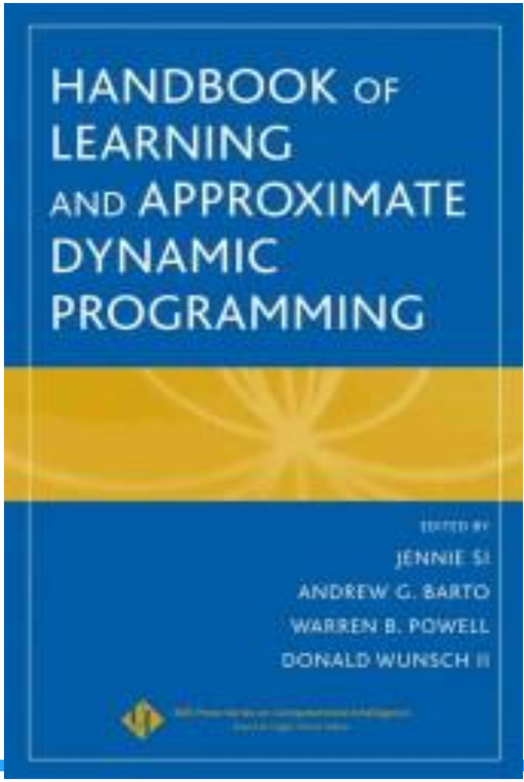
If it converges, as seen in many simulations, will it converge to the optimal function?

Workshops

- Workshop on Learning a
  - Sponsored by the NS
  - **April 8-10, 2002**, in
  - 29 researchers were
  - Including Larry Ho,
  - Published a book: Handbook of Learning and Approximate Dynamic Programming
  - http://www.fulton.asu.edu/~nsfadp/



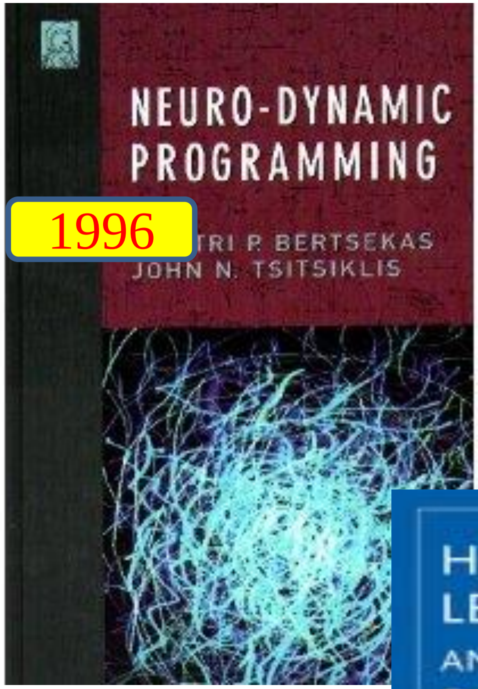


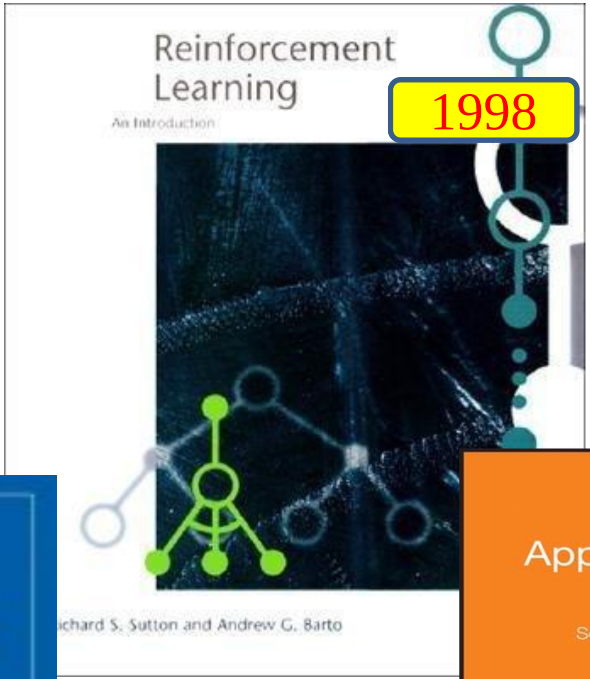


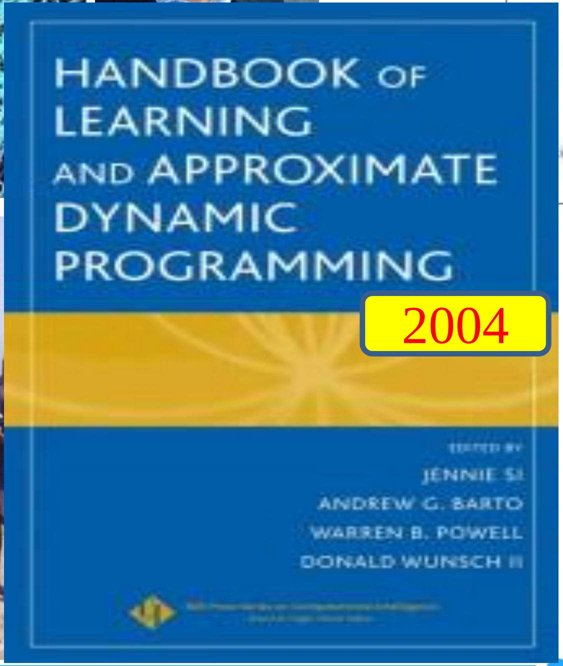
Workshops

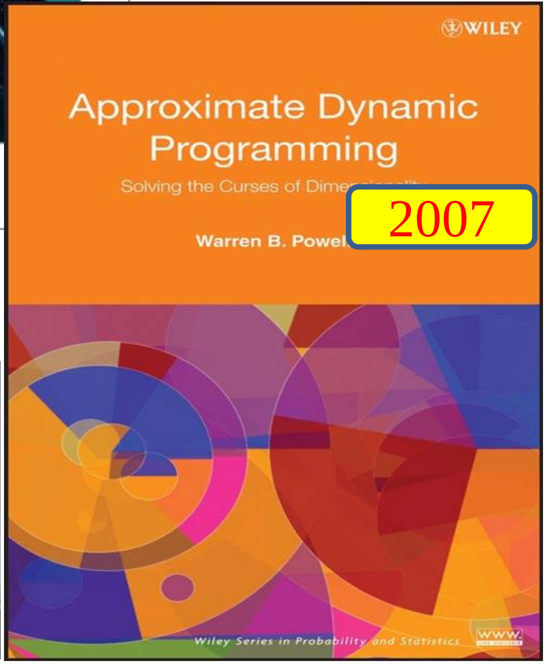
- Workshop on Approximate Dynamic Programming
  - Sponsored by the NSF
  - **April 3-6, 2006**, in Mexico
  - 42 researchers were invited
  - Including Dimitri Bertsekas, Frank Lewis, and Paul Werbos
  - Outreach to Mexican students and researchers
  - http://www.fulton.asu.edu/~nsfadp/

Books by 2007









ADP for Self-Learning Control

- **Dynamic Programming**
- **Adaptive Dynamic Programming**
- **Iterative ADP**
- **New Developments**
- **Concluding Remarks**

Iterative adaptive dynamic programming

- Bellman's principle of optimality

$$J^*(x_k) = \min_{u_k}\{U(x_k, u_k) + \gamma J^*(x_{k+1})\}. \qquad \boxed{\gamma = 1}$$

We will use a function $V(x_k)$ to approximate $J^*(x_k)$.

- We have

$$V(x_k) = \min_{u_k}\{U(x_k, u_k) + V(x_{k+1})\}.$$

Iterative adaptive dynamic programming

- We need to solve for $V(x_k)$ from

$$V(x_k) = \min_{u_k}\{U(x_k, u_k) + V(x_{k+1})\}.$$

- Considering an algebraic equation

$$x = f(x),$$

we can solve this equation by iterative method

$$x_{i+1} = f(x_i) \longrightarrow x_\infty = f(x_\infty)$$

starting from any initial value x_0, if the above iterative process is convergent.

Iterative adaptive dynamic programming

- We can do the same for $V(x_k)$ from

$$V(x_k) = \min_{u_k}\{U(x_k, u_k) + V(x_{k+1})\}$$

to use

$$V_{i+1}(x_k) = \min_{u_k}\{U(x_k, u_k) + V_i(x_{k+1})\}$$

starting from any initial function $V_0(\cdot)$, if the above iterative process is convergent. We can get

$$V_\infty(x_k) = \min_{u_k}\{U(x_k, u_k) + V_\infty(x_{k+1})\}$$

Summary

- The solution of

$$J^*(x_k) = \min_{u_k}\{U(x_k, u_k) + J^*(x_{k+1})\},$$

can be obtained from

$$V_{i+1}(x_k) = \min_{u_k}\{U(x_k, u_k) + V_i(x_{k+1})\},$$

starting from any initial function $V_0(\cdot)$.

- If this iterative process is convergent. We can get

$$V_\infty(x_k) = \min_{u_k}\{U(x_k, u_k) + V_\infty(x_{k+1})\}.$$

Relaxed dynamic programming

- Theorem due to Rantzer and his co-workers:

$$\left[1+\frac{\alpha-1}{(1+\rho^{-1})^i}\right]J^* \le V_i \le \left[1+\frac{\beta-1}{(1+\rho^{-1})^i}\right]J^*$$

where

$$0 \le \alpha J^* \le V_0 \le \beta J^*$$

$$J^*(F(x,u)) \le \rho U(x,u)$$

Proof of convergence and optimality

Rantzer et al.
"Relaxing Dynamic Programming," IEEE Transactions on Automatic Control (2006)
and
IEE Proceedings - Control Theory and Applications (2006)

Relaxed dynamic programming

- Theorem due to Rantzer and his co-workers:

$$\left[1+\frac{\alpha-1}{(1+\rho^{-1})^i}\right]J^* \le V_i \le \left[1+\frac{\beta-1}{(1+\rho^{-1})^i}\right]J^*$$

where

$$0 \le \alpha J^* \le V_0 \le \beta J^*$$

How to choose V_0

Usually $0 < \alpha < 1$ and $\beta > 1$

$$J^*(F(x,u)) \le \rho U(x,u)$$

Restrictions on J^* and U

$U(x,u) \neq 0$, and $U(x,u) = 0$ only when $J^*(F(x,u)) = 0$

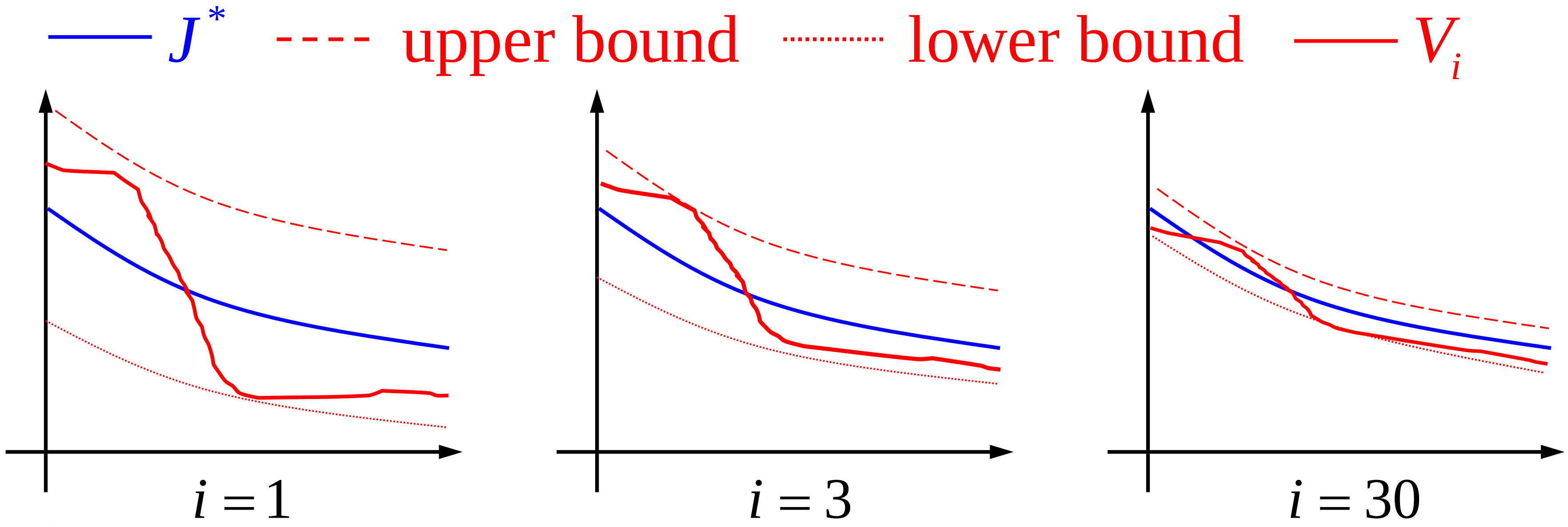
Large ρ $\rightarrow$ slow convergence

Relaxed dynamic programming

- Theorem due to Rantzer and his co-workers:

$$\left[1+\frac{\alpha-1}{(1+\rho^{-1})^{i}}\right]J^{*} \le V_i \le \left[1+\frac{\beta-1}{(1+\rho^{-1})^{i}}\right]J^{*}$$

$\alpha < 1$
$\beta > 1$



Relaxed dynamic programming

- In the theorem due to Rantzer and his co-workers:

$$\left[1+\frac{\alpha-1}{(1+\rho^{-1})^{i}}\right]J^{*}\le V_i\le\left[1+\frac{\beta-1}{(1+\rho^{-1})^{i}}\right]J^{*}$$

where

$$0\le \alpha J^{*}\le V_0\le \beta J^{*}$$

Rantzer suggested to choose $V_0\equiv 0$

$$\Rightarrow\ \alpha=0$$

Iterative adaptive dynamic programming

- Theorem due to Frank Lewis and his co-workers:
  - The limit $V_\infty(x_k)$ of the value function sequence $\{V_i\}$ exists;
  - $\lim_{i\to\infty} V_i(x_k) = V_\infty(x_k) = J^*(x_k)$;
  - $\lim_{i\to\infty} v_i(x_k) = u^*(x_k)$.

Proof of convergence and optimality

Al-Tamimi, Frank Lewis, Abu-Khalaf, "Discrete-Time Nonlinear HJB Solution Using Approximate Dynamic Programming: Convergence Proof," IEEE Transactions on Systems, Man, and Cybernetics – Part B (2008)

Iterative adaptive dynamic programming

- Choose the initial value function as $V_0(\cdot) \equiv 0.$

- For $i = 0, 1, 2, \ldots,$ solve the control law as

$$v_i(x_k) = \arg\min_{u_k} \{U(x_k, u_k) + V_i(x_{k+1})\},$$

and update the value funtion as

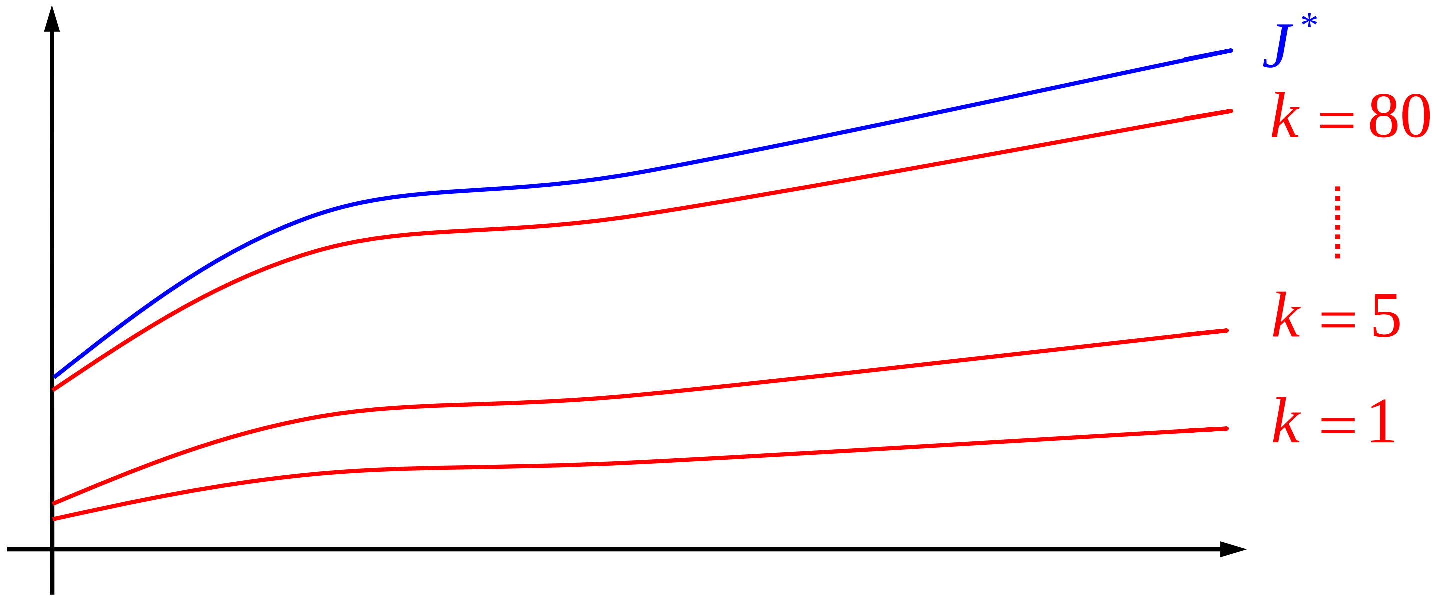
$$V_{i+1}(x_k) = U(x_k, v_i(x_k)) + V_i(F(x_k, v_i(x_k))).$$

$$\underbrace{V_0 \to v_0 \to V_1}_{i=0} \to \underbrace{v_1 \to V_2}_{i=1} \to \underbrace{v_2 \to V_3}_{i=2} \to \cdots\cdots$$

Value iteration

Iterative adaptive dynamic programming

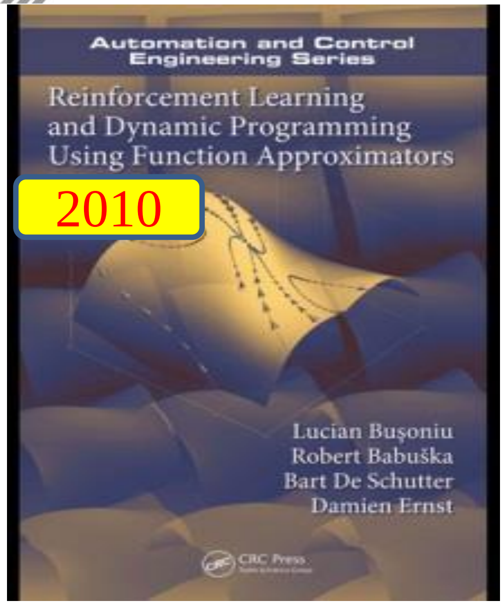
- Theorem due to Frank Lewis and his co-workers:
  – The sequence $\{V_i\}$ is monotonically non-decreasing.

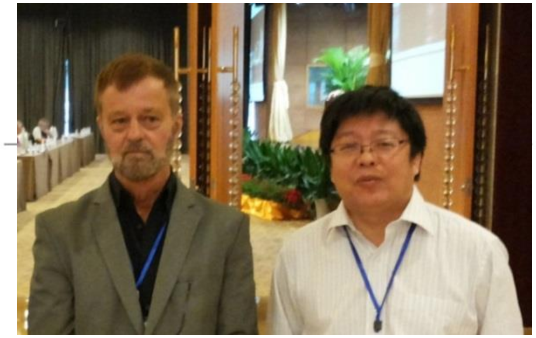


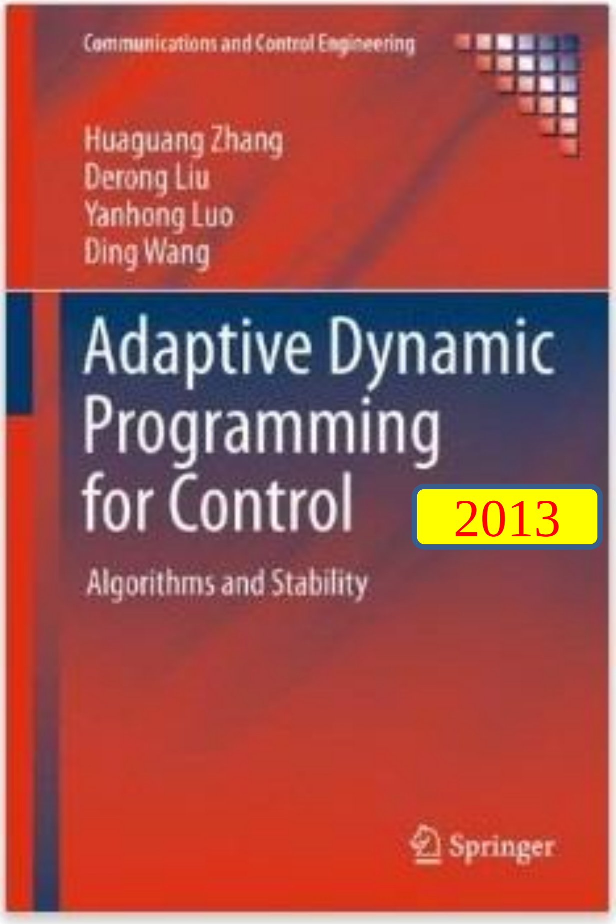
Iterative adaptive dynamic programming

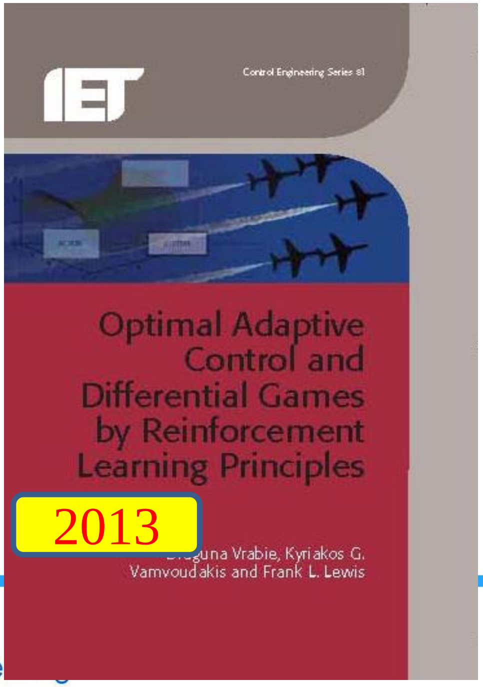
- In the theorem due to Frank Lewis and his co-workers:
  - The sequence $\{V_i\}$ is monotonically non-decreasing because of the choice $V_0 \equiv 0$;
  - From the ith iteration, one obtains v_i and V_{i+1}.

More books by 2013









A New Book

Derong Liu, Qinglai Wei, Ding Wang,
Xiong Yang, Hongliang Li

2016

Adaptive Dynamic Programming with Applications in Optimal Control

October 10, 2016

Springer

xv

Iterative adaptive dynamic programming

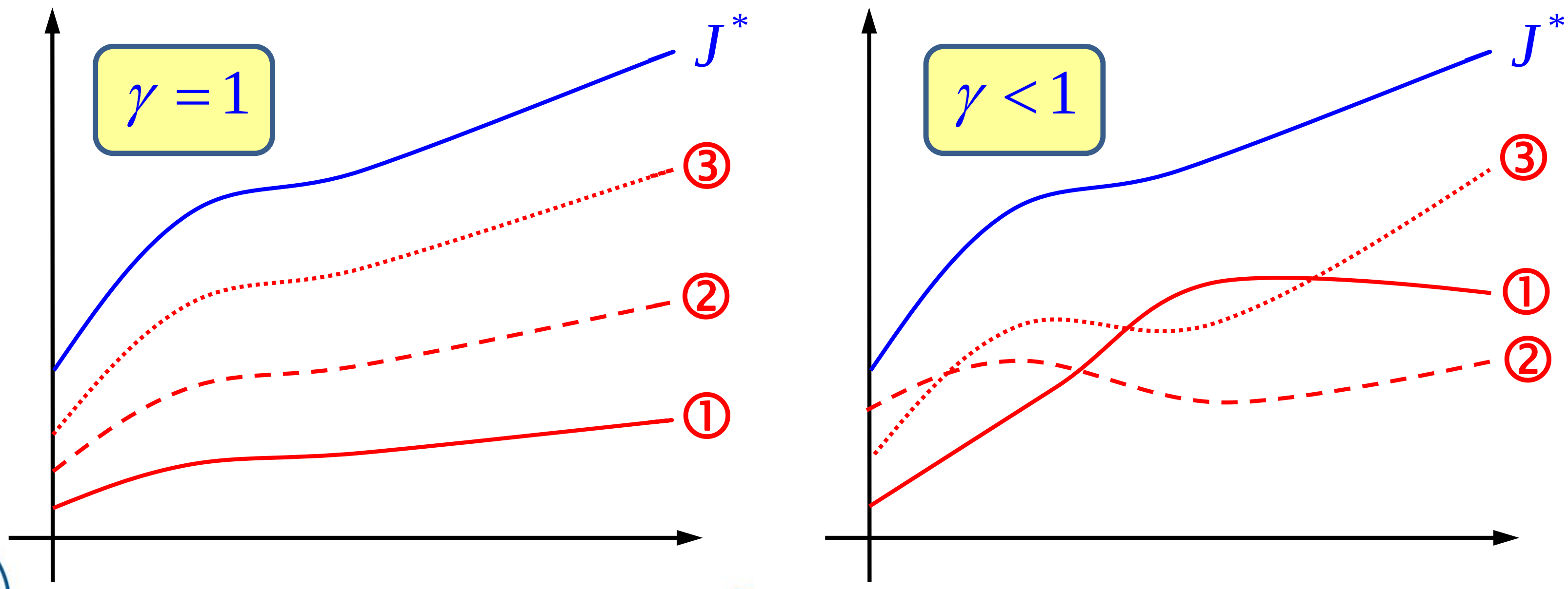
- Bellman Equation:

$$J^*(x_k) = \min_{u_k}\{U(x_k,u_k) + \gamma J^*(x_{k+1})\}$$

- Performance index (or cost to go)

$$J(x_k) = \sum_{i=k}^{\infty} \gamma^{i-k} U(x_i,u_i)$$

Most results are for $\gamma = 1$.

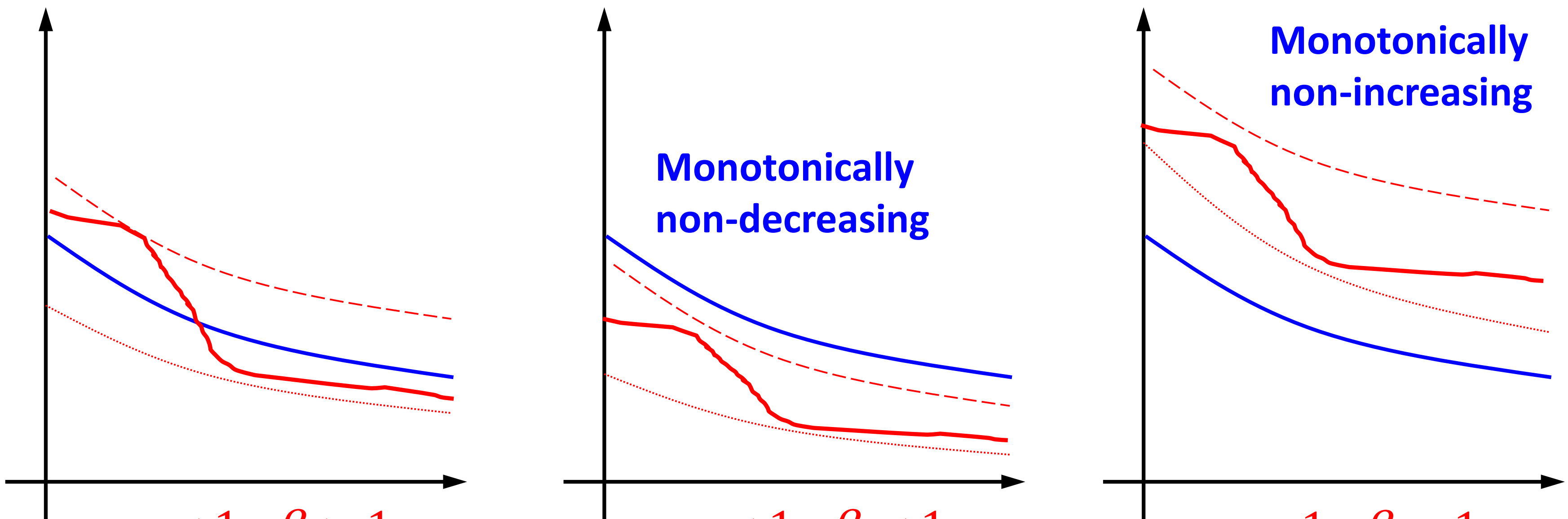


Relaxed dynamic programming

- Theorem due to Rantzer and his co-workers:

$$\left[1+\frac{\alpha-1}{(1+\rho^{-1})^{i}}\right]J^{*}\le V_{i}\le\left[1+\frac{\beta-1}{(1+\rho^{-1})^{i}}\right]J^{*}$$

$$\boxed{0\le\alpha J^{*}\le V_{0}\le\beta J^{*}}$$

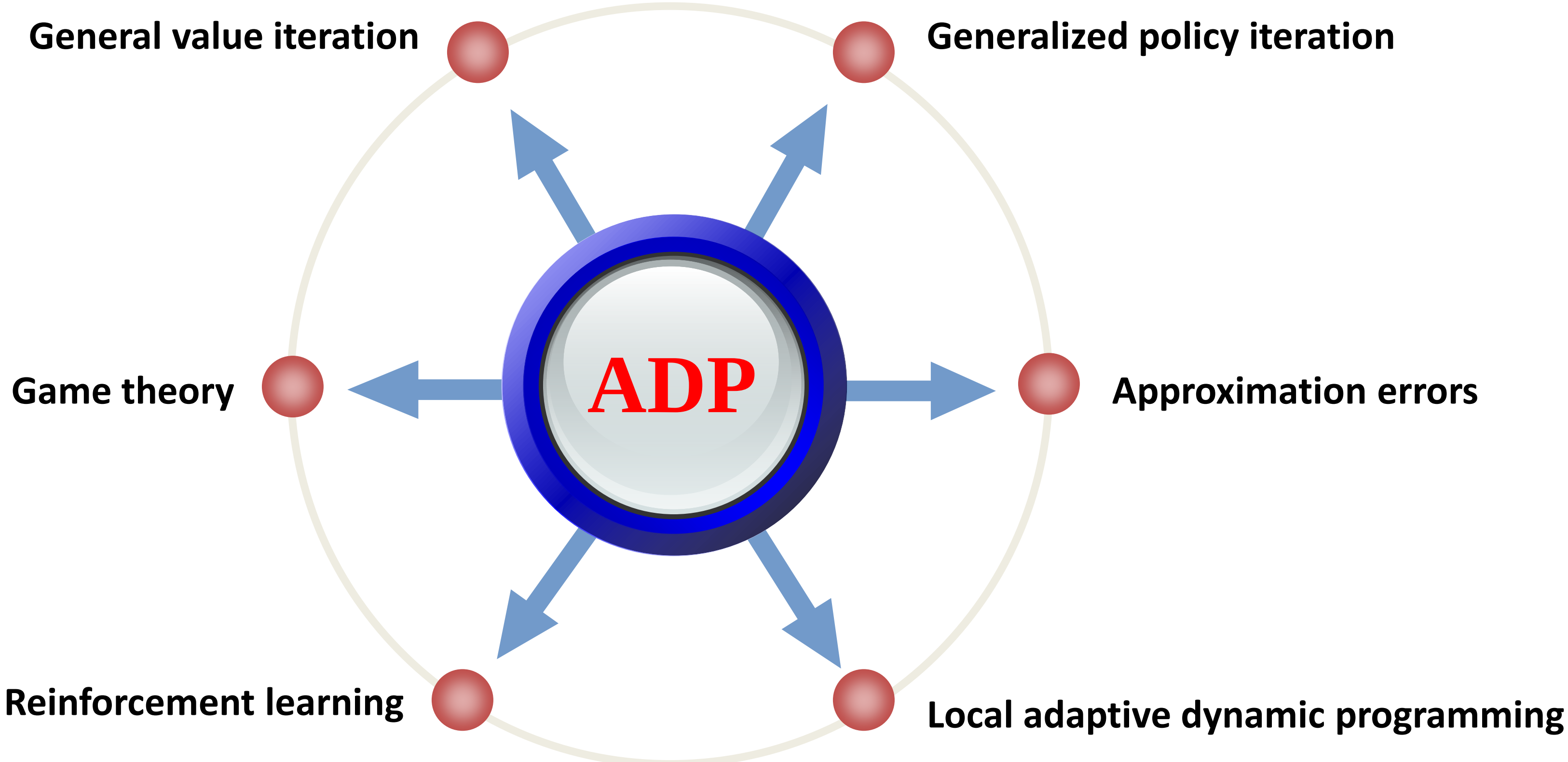


$\alpha<1,\ \beta>1$ $\alpha<1,\ \beta<1$ $\alpha>1,\ \beta>1$

ADP for Self-Learning Control

- Dynamic Programming
- Adaptive Dynamic Programming
- Iterative ADP
- New Developments
- Concluding Remarks

New developments



Value iteration

- Value iteration can be used to solve

$$J^*(x_k) = \min_{u_k}\{U(x_k,u_k) + J^*(x_{k+1})\}.$$

Choose the initial value function as $V_0(\cdot) \equiv 0.$

For $i = 0,1,2,\ldots,$ solve the control law as

$$v_i(x_k) = \arg\min_{u_k}\{U(x_k,u_k) + V_i(x_{k+1})\},$$

and update the value funtion as

$$V_{i+1}(x_k) = U(x_k, v_i(x_k)) + V_i(F(x_k, v_i(x_k)).$$

First, calculate

$$v_0(x_k) = \arg\min_{u_k}\{U(x_k,u_k) + V_0(x_{k+1})\}.$$

For $i = 0,1,2,\ldots,$ do value function update

$$V_i(x_k) = \min_{u_k}\{U(x_k,u_k) + V_{i-1}(x_{k+1})\}$$

$$= U(x_k, v_{i-1}(x_k)) + V_{i-1}(F(x_k, v_{i-1}(x_k)))$$

and policy improvement

$$v_i(x_k) = \arg\min_{u_k}\{U(x_k,u_k) + V_i(x_{k+1})\}$$

$$= \arg\min_{u_k}\{U(x_k,u_k) + V_i(F(x_k,u_k)\}.$$

① General value iteration

- The initial value function V_0 can be chosen as:

$V_0(x_k) = x_k^{\mathrm{T}} P_0 x_k$
– P_0 is positive definite

$V_0(x_k) = \Psi(x_k)$
– positive semidefinite

First, calculate

$$v_0(x_k) = \arg\min_{u_k} \{U(x_k, u_k) + V_0(x_{k+1})\}.$$

For $i = 0, 1, 2, \ldots,$ do value function update

$$V_i(x_k) = \min_{u_k} \{U(x_k, u_k) + V_{i-1}(x_{k+1})\} = U(x_k, v_{i-1}(x_k)) + V_{i-1}(F(x_k, v_{i-1}(x_k)))$$

and policy improvement

$$v_i(x_k) = \arg\min_{u_k} \{U(x_k, u_k) + V_i(x_{k+1})\} = \arg\min_{u_k} \{U(x_k, u_k) + V_i(F(x_k, u_k))\}.$$

① General value iteration

- The initial value function V_0 can be chosen as:

$V_0(x_k) = x_k^{\mathrm{T}} P_0 x_k$
– P_0 is positive definite

$V_0(x_k) = \Psi(x_k)$
– positive semidefinite

◎ If $V_0 \le V_1$, then $V_i \le V_{i+1}$ for all i.

◎ If $V_0 \ge V_1$, then $V_i \ge V_{i+1}$ for all i.

The monotonicity of $\{V_i\}$ depends on the relationship between V_0 and V_1.

$V_0 = 0 \;\Rightarrow$ monotonically non-decreasing;

θ-ADP: $V_0 = \theta\Psi$, $\theta > 0$ large and $\Psi > 0 \;\Rightarrow$ monotonically non-decreasing.

① General value iteration

θ-ADP: $V_0 = \theta\,\Psi,\ \theta > 0$ large and $\Psi > 0 \ \Rightarrow$ monotonically non-decreasing.

$$\Psi \in \bar{\Psi}_x = \{\Psi(x_k):\ \Psi > 0 \text{ and } \exists v(x_k) \text{ s.t. } \boxed{\Psi(F(x_k, v(x_k))) < \Psi(x_k)\}}$$

◎ Value function update

$$V_i(x_k) = \min_{u_k}\{U(x_k, u_k) + V_{i-1}(x_{k+1})\}$$

$$= U(x_k, v_{i-1}(x_k)) + V_{i-1}(F(x_k, v_{i-1}(x_k)))$$

is more likely to obtain a non-increasing sequence.

Policy iteration

- Policy iteration can be used to solve

$$J^*(x_k)=\min_{u_k}\{U(x_k,u_k)+J^*(x_{k+1})\}.$$

Choose an initial value function. Calculate

$$v_0(x_k)=\arg\min_{u_k}\{U(x_k,u_k)+V_0(x_{k+1})\}.$$

For $i=0,1,2,\ldots,$ do value function update

$$V_i(x_k)=\min_{u_k}\{U(x_k,u_k)+V_{i-1}(x_{k+1})\}$$
$$=U(x_k,v_{i-1}(x_k))+V_{i-1}(F(x_k,v_{i-1}(x_k)))$$

and policy improvement

$$v_i(x_k)=\arg\min_{u_k}\{U(x_k,u_k)+V_i(x_{k+1})\}$$
$$=\arg\min_{u_k}\{U(x_k,u_k)+V_i(F(x_k,u_k))\}.$$

Value iteration

Choose an initial admissible control law $v_0(x_k)$.

Stable + cost function is finite.

For $i=1,2,\ldots,$ do policy evaluation

$$V_i(x_k)=U(x_k,v_{i-1}(x_k))+V_i(F(x_k,v_{i-1}(x_k)))$$

and policy improvement

$$v_i(x_k)=\arg\min_{u_k}\{U(x_k,u_k)+V_i(x_{k+1})\}$$
$$=\arg\min_{u_k}\{U(x_k,u_k)+V_i(F(x_k,u_k))\}.$$

Policy iteration

Policy iteration

- Policy iteration can be used to solve

$$J^*(x_k)=\min_{u_k}\{U(x_k,u_k)+J^*(x_{k+1})\}.$$

Choose an initial admissible control law

$v_0(x_k)$.

For $i=1,2,...,$ do policy evaluation

$$V_i(x_k)=U(x_k,v_{i-1}(x_k))+V_i(F(x_k,v_{i-1}(x_k)))$$

and policy improvement

$$\begin{aligned}v_i(x_k)&=\arg\min_{u_k}\{U(x_k,u_k)+V_i(x_{k+1})\}\\&=\arg\min_{u_k}\{U(x_k,u_k)+V_i(F(x_k,u_k))\}.\end{aligned}$$

Require infinite number of iterations to evaluate!

② Generalized policy iteration

Choose an initial admissible control law $v_{(-1)}(x_k)$.

Construct V_0 from $v_{(-1)}$ using

$$V_0(x_k) = U(x_k, v_{(-1)}(x_k)) + V_0(F(x_k, v_{(-1)}(x_k)))$$

and calculate v_0 by

$$v_0(x_k) = \arg\min_{u_k}\{U(x_k,u_k) + V_0(x_{k+1})\} = \arg\min_{u_k}\{U(x_k,u_k) + V_0(F(x_k,u_k)\}.$$

For each i, $i = 1, 2, \dots,$ do N_i-step policy evaluation

$$V_{i,j_i}(x_k) = U(x_k, v_{i-1}(x_k)) + V_{i,j_i-1}(F(x_k, v_{i-1}(x_k))),\ j_i = 1, 2, \dots, N_i,$$

to obtain $V_i(x_k) = V_{i,N_i}(x_k)$ with $V_{i,0}(x_{k+1}) = V_{i-1}(x_{k+1})$.

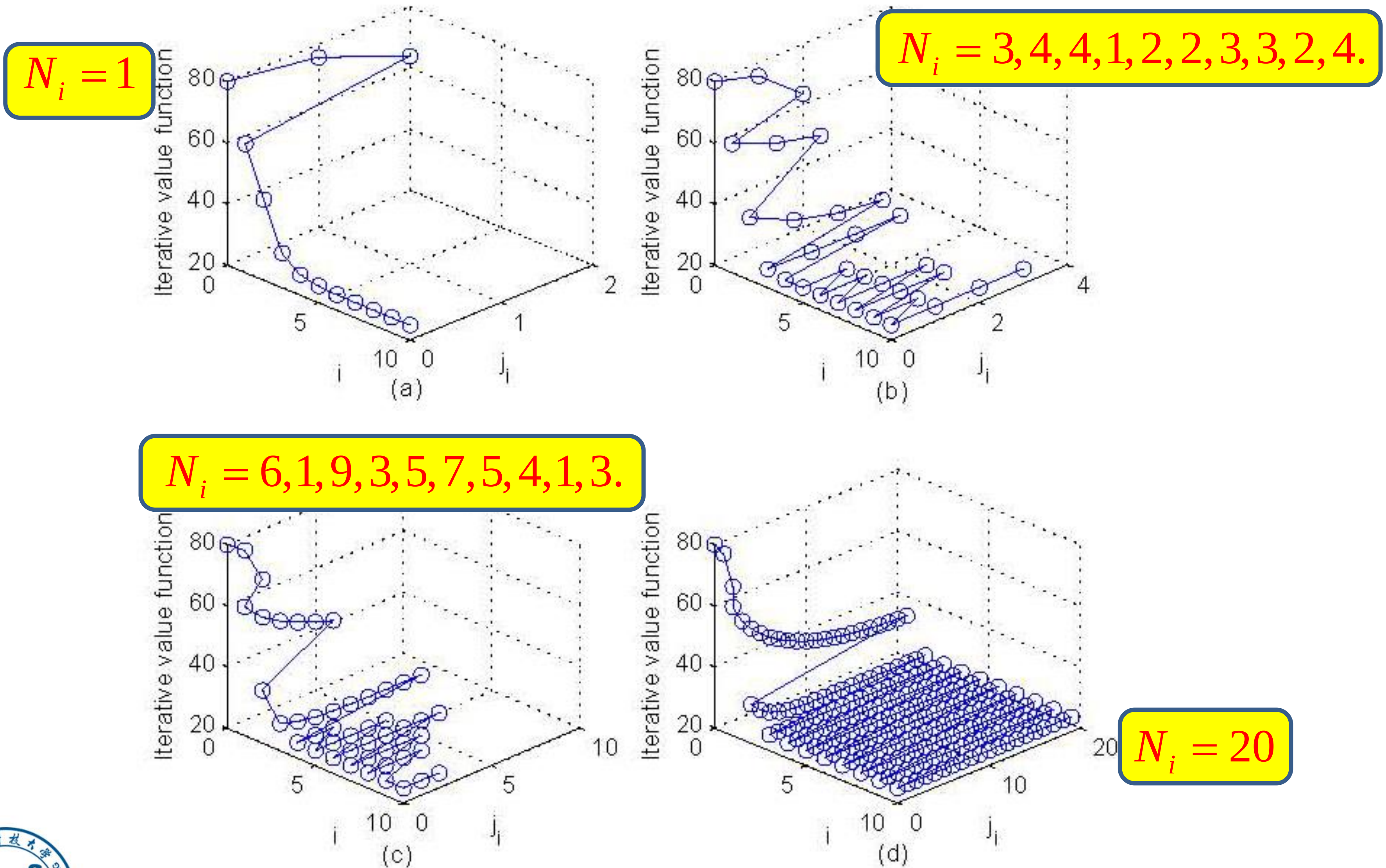
$N_i = 1 \Rightarrow$ VI

$N_i = \infty \Rightarrow$ PI

For each i, do policy improvement

$$v_i(x_k) = \arg\min_{u_k}\{U(x_k,u_k) + V_i(x_{k+1})\} = \arg\min_{u_k}\{U(x_k,u_k) + V_i(F(x_k,u_k)\}.$$

② Generalized policy iteration



③ ADP with approximation errors

- **Considering NN approximation errors, realistic updates are**

$$\hat{v}_i(x_k) = \arg\min_{u_k}\left\{U(x_k,u_k) + \hat{V}_i(F(x_k,u_k))\right\} + \eta_i(x_k)$$

$$\widehat{V}_{i+1}(x_k) = U(x_k,\widehat{v}_i(x_k)) + \widehat{V}_i(F(x_k,\widehat{v}_i(x_k))) + \pi_i(x_k)$$

- **Convergence result now becomes**

$$\hat{V}_i(x_k) \le \sigma\left[1 + \sum_{j=1}^{i}\frac{\rho^j\sigma^{j-1}(\sigma-1)}{(\rho+1)^j} + \frac{\rho^i\sigma^i(\delta-1)}{(\rho+1)^i}\right]J^*(x_k)$$

$$V_0 \le \delta J^*,\ J^*(f(x,u)) \le \rho U(x,u)$$

③ ADP with approximation errors

- **The result becomes**

$$\lim_{i\to\infty}\widehat{V_i}(x_k)=\hat{V}_\infty(x_k)\le\sigma\left[1+\frac{\rho(\sigma-1)}{1-\rho(\sigma-1)}\right]J^*(x_k)$$

Finite neighborhood of the optimal solution

if $$1\le\sigma\le\frac{\rho+1}{\rho}$$

where

$$\hat{V}_i(x_k)\le\sigma\,\Gamma_i(x_k)$$

$$\Gamma_i(x_k)=\min_{u_k}\left\{U(x_k,u_k)+\hat{V}_{i-1}(x_{k+1})\right\}$$

④Local adaptive dynamic programming

- **Every update is required to be done for the whole state space: VI, GVI, PI, GPI.**

◎ Partition state space

$$\Omega_x = L_x^0 \cup L_x^1 \cup \cdots \cup L_x^{\zeta} = \bigcup_{j=0}^{\zeta} L_x^j$$

◎ At each iteration step, only update in one of the subspaces L_x^j.

Asynchronous ADP

⑤ Connections to reinforcement learning

- TD, TD(λ), Q-learning, SARSA, Q(λ), SARSA(λ).

- ADP:
  - HDP $\Leftrightarrow$ TD
  - ADHDP $\Leftrightarrow$ Q-learning
  - DHP, ADDHP
  - GDHP, ADGDHP

- SARSA, SARSA(λ): On-policy

$$Q_{i+1}(s_t,a_t)=Q_t(s_t,a_t)+\alpha\left[r_{t+1}+\gamma Q_i(s_{t+1},a_{t+1})-Q_i(s_t,a_t)\right].$$

Q-learning, Q(λ): Off-policy

$$Q_{i+1}(s_t,a_t)=Q_t(s_t,a_t)+\alpha\left[r_{t+1}+\gamma\max_a\left\{Q_i(s_{t+1},a)\right\}-Q_i(s_t,a_t)\right].$$

- Off-policy learning: Evaluating one policy while following another.

⑤ Connections to reinforcement learning

- HDP $\Leftrightarrow$ TD

◎ In TD (TD(λ) as well):

$$V_{i+1}(s_t) = V_i(s_t) + \alpha\underbrace{[r_{t+1} + \gamma V_i(s_{t+1}) - V_i(s_t)]}_{\boxed{[\text{Target} - \text{OldValue}]}}$$

◎ In model-free HDP, critic network training by minimizing:

$$E = [\underbrace{U_{k+1} + \gamma V_i(x_{k+1})}_{\text{Training target}} - \underbrace{V_i(x_k)}_{\text{To be trained}}]$$

⑥ Game theory

saddle point solution or the Nash equilibrium

- **Consider**

$$x_{k+1} = f(x_k) + g(x_k)u_k + h(x_k)w_k$$

- **Performance index (or cost to go)**

$$V(x_k, u_k, w_k) = \sum_{i=k}^{\infty}\left\{x_i^{\mathrm{T}}Qx_i + u_i^{\mathrm{T}}Ru_i - \gamma^2 w_i^{\mathrm{T}}w_i\right\}$$

- **Our goal is to find the control policy (player 1) and disturbance policy (player 2) so that**

$$J^*(x_0) = \min_{u_k}\max_{w_k}\{V(x_0, u_k, w_k)\}$$

⑥ Game theory

- Iterative algorithm based on ADP

$$V_i(x_k) = \min_{u_k} \max_{w_k} \left\{ x_k^{\mathrm{T}} Q x_k + u_k^{\mathrm{T}} R u_k - \gamma^2 w_k^{\mathrm{T}} w_k \right.$$
$$\left. + V_{i-1}\left(f(x_k) + g(x_k) u_k + h(x_k) w_k \right) \right\}$$

$$u_i(x_k) = -\frac{1}{2} R^{-1} g^{\mathrm{T}}(x_k) \frac{\partial V_i(x_{k+1})}{x_{k+1}}$$

$$w_i(x_k) = \frac{1}{2} \gamma^{-2} h^{\mathrm{T}}(x_k) \frac{\partial V_i(x_{k+1})}{x_{k+1}}$$

- Convergence analysis + extensions to multiplayer games
- Linear systems and ***nonlinear*** systems

ADP for Self-Learning Control

- Dynamic Programming
- Adaptive Dynamic Programming
- Iterative ADP
- New Developments
- Concluding Remarks

Concluding Remarks

① ADP has a long history. Theoretical development started in the 1970s.

② Adaptive dynamic programming is a robust learning control approach.

③ It has a close relationship to reinforcement learning.

④ Ultimate goal: To solve the curse of dimensionality – Need new ideas, to solve dynamic programming with less computation.

For more information

- **derongliu@foxmail.com**
- **http://www.derongliu.org**
- **13810670526**

For more information

- **derongliu@foxmail.com**
- **http://www.derongliu.org**
- **13810670526**