文章编号: 1001-0920(2007)02-0232-05

结构化状态空间中的递阶再励学习方法

孟江华, 朱纪洪, 孙增圻
(清华大学 计算机科学与技术系, 北京 100084)

摘 要: 在状态空间满足结构化条件的前提下, 通过状态空间的维度划分直接将复杂的原始 MDP 问题递阶分解为一组简单的 MDP 或 SMDP 子问题, 并在线对递阶结构进行完善. 递阶结构中嵌入不同的再励学习方法可以形成不同的递阶学习. 所提出的方法在具备递阶再励学习速度快、易于共享等优点的同时, 降低了对先验知识的依赖程度, 缓解了学习初期回报值稀少的问题.

关键词: 再励学习(RL); 递阶再励学习; 结构化状态空间
中图分类号: TP181 文献标识码: A

Hierarchical reinforcement learning algorithm based on structural state-space

MENG Jiang-hua, ZHU Ji-hong, SUN Zeng-qi
(Department of Computer Science and Technology, Tsinghua University, Beijing 100084, China. Correspondent: MENG Jiang-hua, E-mail: meng_jianghua@263.net)

Abstract: In terms of structural state-space, the complex MDP problem is divided into a set of simple MDP or SMDP problems hierarchically according to dimensionality of the state-space. The hierarchically structure is improved in learning process. It forms different hierarchical reinforcement learning (RL) method by adopting different RL algorithm. While having merits of higher speed and knowledge transferring, the proposed algorithm depends less on aforehand knowledge and can weaken the curse of reward lack in the beginning of learning process.

Key words: Reinforcement learning(RL); Hierarchical RL; Structural state-space

1 引 言

递阶再励学习是对传统再励学习方法的重要发展和提高. 在传统再励学习的基础上, 利用"抽象"的概念建立起学习和决策的分层递阶结构, 并将传统的马可夫决策(MDP)问题转化为半马可夫决策(SMDP)问题. 通过构造递阶的宏或宏操作, 使每一层可以忽略其下层的细节, 从而简化学习和决策的过程. 递阶再励学习在解决大维度学习问题、先验知识使用、任务间的知识共享方面有独特的优势, 近年来已经成为再励学习领域研究的热点[1]. 基于 Option 的方法、递阶抽象机(HAM)法以及 MAXQ 分解法是目前主要的递阶再励学习方法.

Sutton 等提出的 Option 方法[2,3]是对再励学习的动作集合进行递阶扩展. 用 Option 集合代替动作集合, Option 由策略、中止条件、输入集 3 部分组成. 在 Option 执行时, 直接根据其内部策略选择动作, 直到满足中止条件. Option 内部也可以选择其他 Option 作为动作.

Parr 提出的 HAM 法[4,5], 在策略上引入了递阶结构. 该方法将策略定义为一个程序, 主程序和可以互相调用的子程序对应着各自的抽象机, 对动作的选择不仅依赖于 MDP 问题的当前状态, 还取决于当前抽象机的状态. 上层策略仅在少量的抽象机选择状态下才需进行决策, 从而实现策略空间的裁减.

Dietterich[6]提出的 MAXQ 分解法是在值函数空间上引入递阶概念, 将原始任务分解为多层子任务, 形成任务图, 并确定各个子任务的完成顺序. 原始任务的值函数相应地递阶分解为子任务的值函数, 可以加快值函数的学习过程.

本文提出了一种新的递阶再励学习方法. 在状态空间满足结构化条件的前提下, 通过状态空间的维度划分, 直接对复杂的原始 MDP 问题进行递阶分解, 分解为一组简单的 MDP 或 SMDP 子问题. 分

收稿日期: 2005-10-21; 修回日期: 2006-03-06.
作者简介: 孟江华(1976-), 男, 江苏镇江人, 博士生, 从事机器学习、移动机器人导航的研究; 孙增圻(1943-), 男, 江苏靖江人, 教授, 博士生导师, 从事机器人智能控制等研究.

解出的子问题可以同步进行学习, 从而加快了学习过程. 与现有的递阶再励学习方法相比, 本文方法不必事先构造出完整的递阶结构, 而是在学习过程中逐步进行构造, 降低了对先验知识的依赖. 本文方法同时提供了一种递阶框架结构, 可以嵌入各种已有的再励学习算法加以实现, 不需设计专门的算法.

2 结构化状态空间的递阶描述

在传统的再励学习中, 状态大多采用表格法描述, 用一维或多维的表格索引号表示状态, 学习过程与状态的索引号分配无关, 核心 MDP 问题(即原始 MDP 问题) 涉及的所有状态组成状态空间, 用 S_{core} 表示.

本文方法首先对结构化的状态空间进行递阶定义, 状态用向量 S 表示, 一个 m 维的状态空间可定义为 $S = [S^1, S^2, \cdots, S^m]$. 状态向量的每一个元素对应于状态的一个维度, 维度之间存在层次关系, 元素 S^i 表示第 i 层维度. 每一个元素的取值都是一个集合 $\{S^i\} = \{s^i_1, s^i_2, \cdots, s^i_{ni}\}$. 递阶的集合 $S \subset S_{core}$, 最多可能状态数为 $n_1 \times n_2 \times \cdots \times n_m$.

对于状态空间 S_{core}, 可以有不同的递阶定义, 但为支持后续的算法, 要求 S 满足下列条件:

条件 1 对于任意的 i, 在状态向量前 i 个元素不变的情况下, 存在策略 P, 可以实现第 i+ 1 个元素所有取值的遍历.

$$P\ i{:}1 \le i \le m-1,\ P\{S^1, S^2, \cdots, S^i\},$$
$$\forall P = s^{i+1}_S \to s^{i+1}_T \mid s^{i+1}_S, s^{i+1}_T \in \{S^{i+1}\}. \quad (1)$$

条件 2 每个状态有且只有一个合法表示.

在许多实际问题中, 状态空间的递阶定义都能满足上述条件. 例如, 可以用楼号、层号、房间号的三维向量定义小区中住户组成的住户空间等.

3 MDP 问题的递阶分解

在状态空间递阶定义的基础上, 将核心 MDP 问题递阶分解为多个简单的 MDP 或 SMDP 问题序列, 从而降低整个问题的复杂度, 快速求出问题的近似最优策略.

3.1 核心 MDP 的分解

核心 MDP 问题的初始和目标状态分别为

$$s_S = [s^1_S, s^2_S, \cdots, s^m_S],\ s_T = [s^1_T, s^2_T, \cdots, s^m_T],$$

则核心 MDP 问题的最优策略 P^* 为

CoreMDP(m):

$$P^* = [s^1_S, s^2_S, \cdots, s^m_S] \to [s^1_T, s^2_T, \cdots, s^m_T].$$

这一策略可分为 2 个步骤实现:

Step1: 实现第 1 层维度元素的转换, 即 $s^1_S \to s^1_T$ 的转换. 在这一步中, 个体不考虑向量中其他元素的变化. 这个过程也是一个 MDP 问题, 称为对核心 MDP 问题的约减 MDP, 记为

reduMDP(S^1):

$$P = [s^1_S, s^2_S, \cdots, s^m_S] \to [s^1_T, s^2_{x_2}, \cdots, s^m_{x_m}],$$

其中 x_2 和 x_m 可以取任意合法值.

Step2: 实现其他层维度元素的转换, 即 $[s^1_T, s^2_{x_2}, \cdots, s^m_{x_m}] \to [s^1_T, s^2_T, \cdots, s^m_T]$ 的转换. 根据式(1), 这一过程中可以保持第 1 层元素取值不变, 问题等效于状态空间 $[S^2, S^3, \ldots, S^m]$ 上的 m- 1 维 MDP 问题, 即

CoreMDP(m- 1):

$$P = [s^2_S, \cdots, s^m_S] \to [s^2_T, \cdots, s^m_T].$$

这样, 核心 MDP 问题便被分解为 2 个子问题, 即

$$\text{coreMDP}(m) = \text{coreMDP}(m-1) + \text{reduMDP}(S^1).$$

对上式进行迭代, 完成递阶分解, 即

$$\text{coreMDP}(m) = \text{coreMDP}(1) + \text{reduMDP}(S^1) + \cdots + \text{reduMDP}(S^{m-1}),$$

其中 $\text{coreMDP}(1) = \text{reduMDP}(m)$. 所以

$$\text{coreMDP}(m) = \sum_{i=1}^{m} \text{reduMDP}(i). \quad (2)$$

将式(2) 右边的子 MDP 解策略顺序相连, 就可以求得 P^* 的近似解.

3.2 reduMDP(S^i) 的定义

reduMDP(S^i) 是在状态向量前 i- 1 个元素保持不变的情况下, 使第 i 个元素完成从任意的 s^i_x 到目标状态 s^i_T 的转变, 这是一个较为简单的 MDP 问题. 用 reduA(S^i) 表示该问题的动作集合, 集合中的动作称为第 i 层的转换 MDP 问题, 实现在前 i- 1 个元素保持不变的情况下, 第 i 层上相邻状态之间的转换过程, 记为 transMDP(s^i_{k1}, s^i_{k2}).

$$\text{reduA}(S^i) = \{\text{transMDP}(s^i_x, s^i_T) \mid s^i_x \in \{S^i\}\}.$$

3.3 transMDP(s^i_{k1}, s^i_{k2}) 的递阶定义

第 i 层的转换 MDP 问题的可选动作是 1 组第 i + 1 层的转换 MDP 问题, 递阶定义如下:

问题

$$\text{transMDP}(s^i_{k1}, s^i_{k2});$$

状态集合

$$S = S^i + \{U\} = \{s^i_1, s^i_2, \cdots, s^i_{ni}, U\},$$

U 表示动作执行过程中状态向量前 i- 1 个元素发生变化的情况, 此时认为转换 MDP 子问题执行失败;

动作集合

$$A^i_s = \{a(s) \mid s \in \{S^i\}\}. \quad (3)$$

集合中每个可选动作都会使状态第 i 层元素取值发生变化, 假设变化后状态为 sc. 同时考虑第 i+ 1 层元素的取值, 设执行前状态为 $s = [\cdots, s^{i+1}_{k1}, \cdots]$,

,], 结果状态为 sc = [, , s^i_{k2}, ,], 则动作可表示为

$$a(s) = a(s^i_{k1}, s^{i+1}_{k1}): y\ s^i_{k2}, U$$

其中 U表示第 i + 1 层元素可以取任意合法的值. 这样, 动作 a(s) 就等效于一个第 i + 1层的转换 MDP, 即 a(s) = a(s^i_{k1}, s^{i+1}_{k1}) : y s^i_{k2} = transMDP(s^{i+1}_{k1}, U).

式(3) 可改写为

$$A^i_s(s^i_{k1}, s^{i+1}_{k1}) = \{transMDP(s^{i+1}_{k1}, U)\}.$$

第 i 层转换 MDP 问题动作集合由第 i + 1 层的转换 MDP 问题组成, 最后一层的转换 MDP 就是核心 MDP 问题中的基本动作.

转换概率: 在递阶的定义下, 动作执行所需的时间是不确定的, 取决于下层任务的内部策略. 因此, 除第 m 层外, 其他子问题都是 SMDP 问题. 在随机问题中, 动作执行后的结果状态是以一定概率出现的. 参考文献[3] 的工作, SMDP 问题中动作 a(s) = a(s^i_{k1}, s^{i+1}_{k1}) : y s^i_{k2}, U执行时, 转换概率定义为

$$P(s^i_{k2} \mid s^i_{k1}, a^i(s^i_{k1}, s^{i+1}_{k1}): y\ s^i_{k2}, U) = \sum_{S=1}^{\infty} C^S \# P(s^i_{k2}, S \mid s^i_{k1}, a^i(s^i_{k1}, s^{i+1}_{k1})).$$

其中: C为折扣因子; P (s^i_{k2}, S| s^i_{k1}, a^i(s^i_{k1}, s^{i+1}_{k1})) 表示在当前 s^i_{k1} 状态下执行动作 a(s), 在 S时间后个体进入状态 s^i_{k2} 的概率.

立即回报: 核心 MDP 问题中, 通常只有当个体进入目标状态时才能获得正的立即回报 r. 本文方法中, 对核心MDP问题进行了分解, 每个子MDP 问题都有明确的子目标. 只要实现了子目标, 对应的子问题就可以获得立即回报 r.

3.4　分解后的子问题集合

核心MDP 问题分解为m 个约减MDP子问题与递阶定义的转换MDP 子问题集合. 每个子问题可以共享探索的场景信息, 利用现有的多种学习方法进行学习.

递阶定义的转换 MDP 子问题的数目与具体任务有关, 只要在第 i 层上 s^i_{k1} 和 s^i_{k2} 是相邻的状态, 就有对应的 transMDP(s^i_{k1}, s^i_{k2}) . 这里的相邻关系定义为: 状态向量前 i - 1 个元素取值不变的情况下, 存在策略 P, 能够以一个非零的概率, 不经过任意其他的 s^i_{k3} 就可以实现从 s^i_{k1} 到 s^i_{k2} 的状态转变.

4　多 MDP 问题的同步学习算法

4.1　多个 MDP 问题学习的同步性

分解出的多个子问题可以同步进行学习. 每个子问题都有各自的子目标, 不论场景是否成功, 只要实现了子目标, 对应的子问题就可以获得立即回报, 从而对值函数进行更新. 执行一个场景可以使多个子问题获得正的立即回报, 实现同步学习.

通常实现子目标比实现全局目标容易. 学习的开始阶段, 在随机性很高的场景中分解的子问题更容易获得成功的场景, 从而在一定程度上缓解了再励学习方法中初期回报值稀少的问题.

4.2　递阶结构的在线生成

对核心MDP 问题进行递阶分解, 最大的困难在于难以事先充分枚举每层状态间的相邻关系, 因此无法充分定义转换MDP 子问题. 本文给出了一种在学习过程中在线构造转换 MDP 递阶结构的方法.

转换MDP 只存在于相邻的状态之间, 这种相邻关系可以在学习场景中检测出来. 开始时将转换 MDP 子问题集合设为空, 检测到新的相邻状态时动态地进行添加, 这便降低了对先验知识的要求. 如果对动态添加过程设置一定的条件(例如相邻关系出现的次数等) , 还可以避免引入过多的无用子 MDP 问题.

当子问题集合为空, 即没有可选的动作时算法会随机产生一个基本动作.

4.3　算法结构描述

分解后的多个子问题形成递阶框架结构, 可以采用各种现有的再励学习方法加以实现.

(1) 初始化: 结构化状态空间的 m 阶递阶定义; 初始化 m 个约减 MDP 子问题; 初始化空的转换 MDP 子问题集合.

(2) 学习场景的循环: 根据设定条件, 循环执行多个场景; 在每个场景中依次执行式(2) 中右侧的子问题, 直至场景成功; 如果某个子问题超过了规定步数还未完成或状态向量前 i - 1 个元素取值发生变化, 则认为该问题未能成功完成, 场景自动结束.

(3) 每个学习场景内的学习过程:

Step1: 每层的子问题都将下层子问题集合作为自己的动作集合, 最终由最底层的子问题产生基本动作. 下层子问题集合为空时随机产生基本动作.

Step2: 执行该基本动作, 记录动作及其结果.

Step3: 检查状态向量的变化, 如果:

1) 仅有第 m 个元素发生变化, 则不作处理.

2) 状态向量的前 i - 1 个元素取值不变, 而第 i 个元素发生变化, 则认为检测到 1 个第 i 层的相邻关系; 如果是新的相邻关系, 则初始化对应的转换 MDP 问题, 并加入到子问题集合中; 如果对应的子问题已经存在, 则产生一个成功场景, 利用选定的再励学习方法对该子问题进行学习.

3) 如果某个约减 MDP 问题成功完成, 则利用选定的再励学习方法对该子问题进行学习.

5　仿真实验

在 6 @6 的网格环境中(见图 1) , 汽车从 S 点出发, 将分别位于 A 点和 B 点的两位客人送到目的地

T 点, 汽车可以选择上下左右 4 个动作, 求汽车运行的最短路径. 在核心 MDP 问题中, 汽车有 36 个可能的位置, A 点和 B 点两位客人是否在车上共有 4 种可能, 所以一共有 4 @36 = 144 种状态.

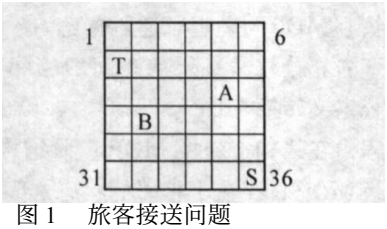


图 1 旅客接送问题

状态空间递阶定义为 $S = [S^1, S^2]$, S^1 表示客人是否在车上(0 表示 AB 都不在, 1 表示 A 在 B 不在, 2 表示 B 在 A 不在, 3 表示 AB 都在); S^2 表示汽车的位置, 取值 1 ~ 36(左上网格为 1, 右下网格为 36). 问题的开始状态为[0, 35], 目标状态为[3, 7] .

核心 MDP 分解为

coreMDP(2) =

reduMDP(S^1) + reduMDP(S^2) .

reduMDP(S^1) 完成从[0, 35] 到[3, *] 的状态转变, reduMDP(S^2) 完成从[3, *] 到[3, 7] 的转变.

仿真中在线生成转换 MDP 子问题. 在状态向量第 1 层元素上存在 4 个相邻关系, 算法检测的结果如图 2 所示.

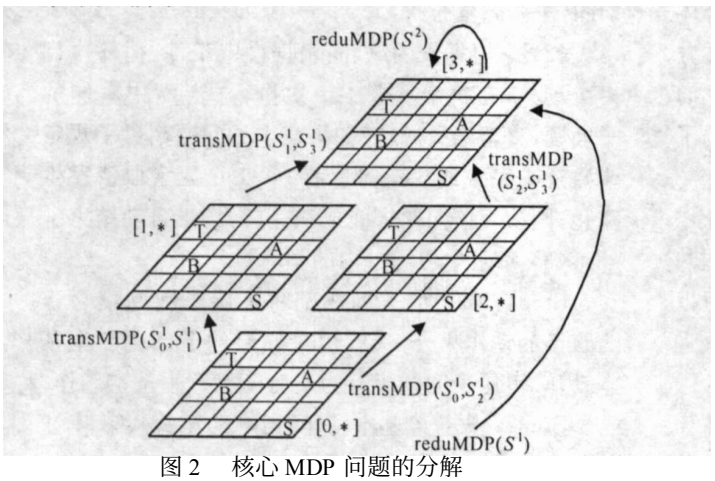
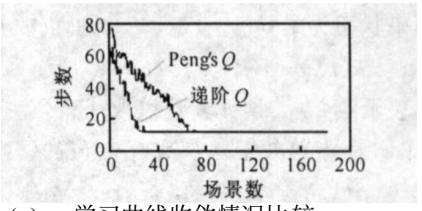


图 2 核心 MDP 问题的分解

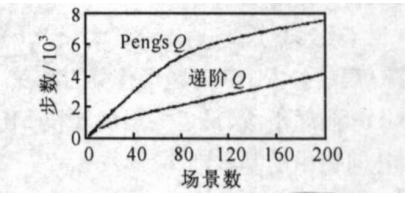
每个子问题的学习都采用 Peng. s Q 学习方法, 与直接使用 Peng. s Q 方法进行比较. 经过 100 次实验, 平均后的仿真结果如图 3 和图 4 所示.

图 3 是同等参数条件下, 本文的递阶 Q 学习与 Peng. s Q 学习的学习曲线比较. 可见, 本文方法的学习速度明显快于 Peng. s Q 方法, 尤其在学习的开始阶段(见图3(a)) . 同时在学习过程中, 平均每个场景的步数明显少于 Peng. s Q 方法(见图3(b)) . 图 4 是子问题的学习曲线(第 1 个约减 MDP 问题过于简单, 没有列在图中) .

在同等硬件条件下完成200个场景的学习,

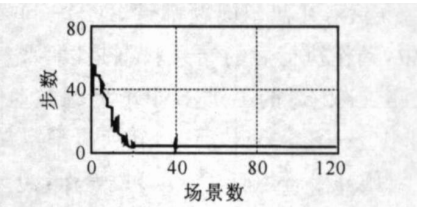


(a) 学习曲线收敛情况比较

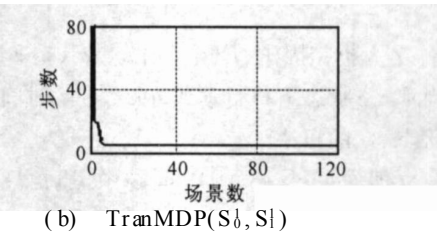


(b) 算法所消耗总步数比较

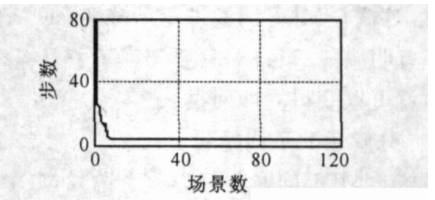
图 3 递阶 Q 与 Peng. s Q 的学习曲线比较



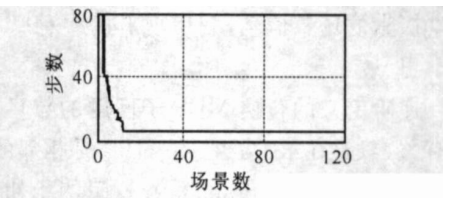
(a) reduMDP(S^2)



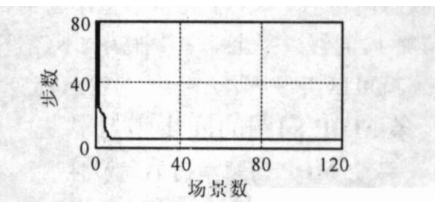
(b) TranMDP(S^1_0, S^1_1)



(c) transMDP(S^1_1, S^1_3)



(d) transMDP(S^1_0, S^1_2)



(e) transMDP(S^1_2, S^1_3)

图 4 子问题的学习曲线

Peng. s Q 方法用时 3. 052 s, 本文方法用时 1. 833 s(均为 CPU 时间) .

6 结 论

与传统的再励学习方法相比, 本文方法中每个

子任务都是一个较小的 MDP 或 SMDP 问题, 多个子任务同步学习, 学习速度明显加快. 在学习初期, 即使在失败的场景中, 只要子任务成功完成, 就可以获得正的立即回报值, 从而在一定程度上缓解了传统再励学习中初期回报值稀少的问题.

在仿真问题中, 如果改变目标点 T 的位置, Peng. s Q 方法中所有的 Q 值都需要进行重新学习. 而本文方法仅有 2 个 reduMDP( ) 问题需要更新 Q 值, 其他 4 个转换 MDP 问题则不需要重新学习. 这些子问题可以在 T 不同的问题之间直接共享.

由于本文方法只需事先确定状态空间的维度划分, 转换 MDP 问题可以在学习的过程中检测出来. 因此, 在对先验知识的依赖程度上, 要优于其他递阶再励学习方法.

本文方法只能得到近似最优的策略. 从全局看, 因为在执行子问题过程中, 内部策略只是局部最优的, 这样得到的整体策略将是递归最优[6]的. 但是, 在满足条件(/ 在第 i 层转换 MDP 问题成功执行时, 所有可能目标状态的第 i+ 1 层元素取值唯一0对所有转换MDP 问题和所有 i 成立.) 时, 本文方法可以得到全局最优解策略(递阶最优策略[6]).

参考文献(References)

[1] Barto A G, Mahadevan S. Recent advances in hierarchical reinforcement learning[J]. Discrete Event Dynamic Systems: Theory and Applications, 2003, 13(4): 4122277.

[2] Sutton R, Precup D, Singh S. Between MDPs and Semi2MDPs: A framework for temporal abstraction in reinforcement learning[J]. Artificial Intelligence, 1999, 112(1): 1812211.

[3] Precup D. Temporal abstraction in reinforcement learning[D]. Amherst: University of Massachusetts, 2000.

[4] Parr R. Hierarchical control and learning for Markov decision proc[D]. Berkeley: University of California, 1998.

[5] Parr R, Russell S. Reinforcement iearning with hierarchies of machines [C]. Advances in Neural Information Processing Systems: Proc of the 1997 Conf. Cambridge: MIT Press, 1998: 1043211049.

[6] Dietterich T G. Hierarchical reinforcement learning with the MAXQ value function decomposition[J]. J of Articial Intelligence Research, 2000, 13(2): 2272232.

(上接第 232 页)

[5] 唐巍, 李殿璞. 电力系统经济负荷分配的混沌优化方法[J]. 中国电机工程学报, 2000, 20(10): 3620240.
(Tang W, Li D P. Chaotic optimization for economic dispatch of power systems[J]. Proc of the CSEE, 2000, 20(10): 3620240.)

[6] 侯云鹤, 鲁丽娟, 熊信良, 等. 改进粒子群算法及其在电力系统经济负荷分配中的应用[J]. 中国电机工程学报, 2004, 24(7): 952100.
(Hou Y H, Lu L J, Xiong X L, et al. Enhanced particle swarm optimization algorithm and its application on economic dispatch of power systems[J]. Proc of the CSEE, 2004, 24(7): 952100.)

[7] Aruldoss Albert Victoire T, Ebenezer Jeyakumar A. Hybrid PSO2SQP for economic dispatch with valve2point effect[J]. Electric Power System Research, 2004, 71(1): 51259.

[8] Hansen J V. Genetic search methods in air traffic control[J]. Computers and Operations Research, 2004, 31(3): 4452459.

[9] Saleh H A, Chelouah, R. The design of the global navigation satellite system surveying networks using genetic algorithms[J]. Engineering Applications of Artificial Intelligence, 2004, 17(1): 1112122.

[10] Baskar S, Subbaraj P, Rao M V C. Hybrid real coded genetic algorithm solution to economic dispatch problem[J]. Computers and Electrical Engineering, 2003, 29(3): 4072419.

[11] 唐焕文, 张立卫, 王雪华. 一类约束不可微优化问题的极大熵方法[J]. 计算数学, 1993, 15(3): 2682275.
(Tang H W, Zhang L W, Wang X H. A maximum entropy method for a sort of constrained nondiffrentiable optimization problems[J]. Calculate Mathematics, 1993, 15(3): 2682275.)

[12] 李兴斯. 一类不可微优化问题的有效解法[J]. 中国科学(A辑), 1994, 24(4): 3712377.
(Li X S. A effective method for a sort of nondiffrentiable optimization problems[J]. Science China(Series A), 1994, 24(4): 3712377.)