目录

[1 绪论 2](#_Toc420867498)

[**1.1** **选题背景及研究意义** 2](#_Toc420867499)

[**1.2** **国内外研究现状** 2](#_Toc420867500)

[**1.2.1** **调度算法** 2](#_Toc420867501)

[**1.2.2** **POMDP 问题** 4](#_Toc420867502)

[**1.2.3** **粒子滤波算法** 5](#_Toc420867503)

[**1.3** **研究目标和内容** 6](#_Toc420867504)

[**1.4** **技术路线** 6](#_Toc420867505)

[**1.5** **论文组织** 7](#_Toc420867506)

[2 调度算法研究 7](#_Toc420867507)

[**2.1** **贪婪算法** 7](#_Toc420867508)

[**2.2** **动态规划算法** 8](#_Toc420867509)

[**2.3** **rollout算法研究** 9](#_Toc420867510)

[**2.3.1** **consecutive rollout** 10](#_Toc420867511)

[**2.3.2** **exhaustive rollout** 10](#_Toc420867512)

[3 粒子滤波算法研究 11](#_Toc420867513)

[**3.1** **基本算法** 11](#_Toc420867514)

[**3.2** **重要性采样** 14](#_Toc420867515)

[**3.3** **公式化** 16](#_Toc420867516)

[4 POMDP建模 17](#_Toc420867517)

[**4.1** **有限环境** 18](#_Toc420867518)

[**4.1.1** **一个说明性例子** 19](#_Toc420867519)

[**4.2** **belief空间中的价值迭代** 26](#_Toc420867520)

[**4.3** **计算价值函数** 27](#_Toc420867521)

[5 仿真实验结果 32](#_Toc420867522)

[**5.1** **模型建立** 32](#_Toc420867523)

[**5.2** **问题公式化** 33](#_Toc420867524)

[**5.2.1** **系统状态，动作和状态转换律** 34](#_Toc420867525)

[**5.2.2** **单步成本** 35](#_Toc420867526)

[**5.2.3** **观测和观测律** 36](#_Toc420867527)

[**5.3** **算法流程** 37](#_Toc420867528)

[**5.3.1** **粒子滤波流程** 37](#_Toc420867529)

[**5.3.2** **Rollout 算法流程** 40](#_Toc420867530)

[**5.4** **实验结果** 41](#_Toc420867531)

# 绪论

* 1. **选题背景及研究意义**

雷达自从其被发明那天开始就在人类的生产生活中占据了重要的位置。在科学技术日益进步的今天，无论是无人驾驶还是飞机巡航等方面都少不了雷达的身影。为了提高对目标的探测的准确性,以及在有多个目标的情况下实现低误差性,现代的雷达系统都具有多种工作模式对目标进行探测[1]。对于多个不同的模式或波束进行调度是一个很重要的问题，也即本选题的波束调度问题。对于该问题，一方面需要提高结果的准确性,另一方面又要减小传感器状态转变或者探测的花费。传感器调度问题一般为离散优化问题。对于该问题有很多种解决方法,如动态规划、贪心算法、枚举法，或者是本选题采用的Rollout算法。Rollout算法可以说是一个权衡计算复杂度与算法精确性之后的结果，具有重要的应用价值。因此，本课题将Rollout算法应用于雷达多波束调度问题，并与短视算法、动态规划算法进行比较分析，为工程应用提供指导和借鉴。

* 1. **国内外研究现状**
     1. **调度算法**
        1. **精确性算法**

精确性算法用于得到问题的最优解。在求解整数规划问题中，精确性算法主要有 0-1 规划算法、动态规划算法、分支定界算法三个方向。0-1 规划问题是决策变量仅取值为 0 或 1 的问题，求解这类问题主要为隐枚举法(如分支定界法)，Patterson 和 Huber 首先使用 0-1 规划对此类问题进行建模，然后使用定界法对问题进行求解，最后与枚举法进行对比。求解结果优于枚举法[6]。动态规划是将多阶段决策过程转化为一系列单阶段决策问题，Garruthers 和Battersby 最早将动态规划算法应用到资源受限项目调度的研究[7]，虽然求解结果有较大的改进，但是仍然不能解决这类问题。分支定界算法在求解资源受限问题上，由于其计算效率和结果的优势，被较多的学者研究。基本思想是：对满足当前约束的所有解空间进行搜索，将全部可行解空间分割成多个子集，再使用一定规则淘汰掉不可能产生最优解的子集，选择可能产生最优解的子集，来缩小搜索空间，从而找到最优解。Patterson 等提出基于紧前树的分支定界算法[8]，Demeulemeester 等通过引入延迟替代集来解决资源冲突[9]。使用精确性算法求解此类问题比较适用于小规模问题，能在一定的时间内得到问题的精确解。但由于该问题是 NP-hard 问题，问题的求解时间和内存会随任务数目的增加而呈指数增长，当问题规模较大时，求解时间会太长，同时会出现内存不足的现象，无法得出调度结果。

* + - 1. **智能优化算法**

在此类问题中，大规模问题无法使用精确性算法求解，通常使用智能优化算法来求解，可以在一个较短的时间内得到问题的次优解，下面具体讨论智能优化算法在项目调度中的应用。  
（1）遗传算法遗传算法是由生物学的进化规律演变而来的随机性搜索算法，Davis 首次将遗传算法应用到项目调度中[10]，解决了车间项目调度问题。Hartmann 使用任务列表编码[11]，提出了基于优化规则和随机性的遗传算法。刘士新以遗传算法为基础，设计一种基于定界策略的近似算法[12]，杨宏利结合任务存储的邻接矩阵，有效的解决编码中无效的任务调度现象[13]。  
（2）蚁群算法蚁群算法是以蚁群觅寻食物源最短路径为背景提出的智能优化算法，Dorigo 和Colorni 首先提出蚁群系统[14]，并将该算法首先成功的应用在旅行商问题。Merkle等将蚁群算法应用在项目调度中[15]，将蚂蚁走过的路径映射为调度方案，通过结合优先规则来选择任务。  
（3）粒子群算法粒子群算法是通过粒子间信息共享和协作来寻找最优解，Kennedy 等[16]在 1995年首先提出这种算法。Zhang[17]将该算法应用项目调度中，对基于任务列表和优先粒子两种情况，分别采用不同的进化策略求解。  
（4）模拟退火算法模拟退火算法是对局部搜索算法的改进[18]，Boctor 在 1996 年首次将该算法应用到 RCPSP 问题中，并将结果与禁忌算法相比较，结果显示该算法的实用性和有效性[19]。Cho 和 Kim 使用一个队列来表示一个调度方案，队列中的元素代表任务序号，通过模拟退火方法需找优先队列[20]。  
（5）禁忌算法禁忌算法通过对优化过程进行记录和选择，建立禁忌表，来避免陷入局部最优。Nonobe 和 lbarakiP 根据实际应用对 RCPSP 问题进行扩展，并使用禁忌算法Rollout算法求解问题[21]。Atli 分别使用禁忌算法和 CPLEX 软件求解 RCPSP 问题，结果表明禁忌算法在求解时间上有着明显的优势，CPLEX软件仅适合求解小规模问题[22]。

* + - 1. **Rollout算法**

Rollout算法内容比较丰富，主要分为：基于优先规则、局部搜索技术和元Rollout算法。其中基于优先规则的Rollout算法应用较广泛[23]，它通过一定的启发规则快速得到问题的解，但不保证解的质量。基于优先规则的Rollout算法主要由两个部分组成：进度生成机制和任务优先规则。进度生成机制是大多数Rollout算法的核心部分，根据扩展方式的不同，分为以任务为阶段变量和以时间为阶段变量两个类型，其中以任务、时间为阶段变量分别称为串行进度生成机制(SSGS)和并行进度生成机制(PSGS)[2]，也即consecutive rollout算法和exhaustive rollout算法。Hartmann 等[24]对两种不同机制下调度问题进行研究，结果表明，当项目中任务比较多，PSGS 较优，而任务比较少时，则SSGS较优。由于PSGS的是非延迟进度机制，搜索空间少于SSGS，当任务比较少，使用 SSGS 搜索解与精确性偏差小，当任务较多时，使用 PSGS 能很快搜索问题的解空间。任务优先规则用于在项目调度中赋予任务优先权，在进度生成机制时基于任务优先权分配任务开始时间和结束时间。文献[25]分别提出最短工期、最长工期、最多紧后任务、最多后续任务、最大秩序权重、最早开始时间和最早结束时间六种优先规则。刘士新等[26]对众多优先规则进行研究，结果表明使用优先规则求解的效果与项目的内部结构有关，需要根据项目的参数特点来选择一个或多个优先规则组合对问题求解。由于进度生成机制和优先规则的多样性，Rollout方法可分为单次算法和多次算法。如果只使用一种进度生成机制和一种优先规则对项目进行调度，则称为单次算法；如果使用两种进度生成机制或多种优先规则对项目进行调度，则可以得到多个调度方案，并从中挑选较好的，则称为多次算法，多次算法调度结果优于单次调度算法[27]。

* + 1. **POMDP 问题**

俄罗斯数学家马尔科夫于 1907 提出的马尔科夫决策过程，这类决策过程有一个共同的特性即为马尔科夫性，也称无后效性。它是指某阶段的状态一旦确定，则此后过程的演变不再受此前各状态的影响。具体地说，如果一个问题被划分各个阶段之后，后一个阶段中的状态只受前一阶段的影响，与其他状态没有关系，且是由前一阶段的状态通过状态转移方程得来的。MDP 是基于决策理论的规划方法中应用最多的模型。它描述了决策节点与环境相互作用的过程。

POMDP 引入了belief状态这一概念，在belief状态具有马尔科夫性，同时由于信念状态空间的连续n维性，使得 POMDP 的求解更加困难。但是，POMDP 是在MDP 基础上扩展的一个更一般化的模型，它能够很好地对环境、动作和观察的不确定性进行建模，相对于 MDP 更贴近于现实情况，具有更广泛的应用。所以，从POMDP 模型提出至今，一直都得到了各个领域学者们的广泛关注，先后产生了很多有用的技术，以解决不确定环境下的决策与规划问题。

为了快速高效准确的求解 POMDP 问题，从最初的精确求解算法到后来的近似求解方法，先后出现了很多的求解算法。POMDP 的求解算法可以按照前向或后向分类，也可按照精确求解和近似求解分类。本文按照后者来进行分类介绍。1971 年 Sondik 在其博士论文中提出的一次合格算法拉开了精确求解 POMDP的序幕。随后出现了一系列的精确求解算法，例如 Monahan 的算法，线性支撑算法，目击算法，增量剪枝算法等。由于 POMDP 精确求解的困难性，以精确算法为基础，研究高效的近似算法将是解决 POMDP 问题的重要方向。文献[5]总结了一些近似算法，例如网格方法，有限历史方法，Rollout方法，因子化信念状态分解方法，神经网络方法等。这些方法大部分都是 2000 年以前的文献中出现的。基于点的 POMDP 求解方法 PBVI[6]于 2003 年提出，开启了基于点的 POMDP 方法的历程，至今基于点的方法都是求解 POMDP 相对最有效的方法；04 年提出的新的Rollout搜索方法求解 POMDP[7]；05 年基于点的方法有了新的重要发展，提出了 Perseus[8]方法；06 年的混合 POMDP 算法[9]；07 年的AEMS[10]，Forward search value iteration[11]，基于点策略迭代[12]等；08 年基于点的另一个重要方法被提出，即 SARSOP[13]，同时文献[14]提出的在线求解方法也是一个重要的方法；2009 年文献[15]通过对信念空间分解或压缩来求解 POMDP；2010年文献[16]使用局部逼近的方法求解高维连续 POMDP 问题，文献[17]用 KaczmarzIterative Method 进行 POMDP 值函数的剪枝，以及通过对信念空间的聚类或压缩来实现 POMDP 的快速求解[18]，另外还有 Density Projection[19]方法，在线方法 AnOnline Algorithm for Constrained POMDPs[20]等等。随着 POMDP 求解方法的不断发展，其应用研究也得到了广泛关注。

* + 1. **粒子滤波算法**

国外对粒子滤波的研究起步较早。早在二十世纪五十年代Hammersley等人就提出一种被称为Sequential Important Sampling(SIS)的方法。到了六十年代后期Handschi等将SIS法应用于控制领域。七十年代各个领域的学者继续沿着SIS的思路研究，在这以后一系列改进的SIS算法[1 ,2 ]相继出现。但是，SIS算法容易导致粒子退化现象(Particle Degeneracy)，影响了它在实际中的推广应用。直到1993 年， Gordon 等人提出了采样重要性重采样算法 (Sampling importanceResampling, SIR)[ 3]，才基本解决了粒子退化问题。从此，粒子滤波又被广泛关注，并取得了重大进展，提出了一些重要的SIR算法。如，多项式重采样算法(Multinomial Resampling)[4 ]，分层重采样算法(Stratified Resampling)[5 ]，残差重采样算法(Residual Resampling)[6 ]，系统重采样算法(Systematic Resampling)[7 ]等，文献[8]详细介绍了重采样及相关算法，文献[9]对重采样模式进行了比较。由于SIR算法是对权值较大的粒子进行多重复制，造成粒子多样性损失，最终仍然可能出现粒子退化问题。文献[10~14]对重采样算法进行了分析研究，做了一些改进。国内对粒子滤波的研究开始较晚。但是许多大学和科研院所都对其十分关注，并进行了相关应用与理论研究。在粒子滤波理论方面，文献[15~17]介绍了粒子滤波基本理论和国内外最新进展；在粒子滤波的改进方面，文献[18]提出了一种使用非等权值粒子的确定性粒子滤波算法，文献[19] 提出一种扩展卡尔曼粒子滤波算法的修正方法；在粒子滤波的应用方面，文献[20]将粒子滤波应用于轮廓线跟踪，文献[21]应用于行人跟踪，文献[22]用粒子滤波在闪烁噪声环境下进行目标跟踪，文献[23]讨论粒子滤波在混合状态与参数估计中的应用，文献[24，25]将粒子滤波用于被动定位跟踪。目前国内对粒子滤波的研究水平与国外相比尚有较大差距，但各相关领域的学者都在孜孜不倦的研究探讨，相信会逐渐接近并超过国外的研究水平。

* 1. **研究目标和内容**

1. 基于POMDP的多目标跟踪雷达波束调度建模
2. 基于短视策略的多目标跟踪雷达波束调度方法
3. 基于动态规划的多目标跟踪雷达波束调度方法
4. 基于Rollout的目标跟踪雷达波束调度算法
   1. **技术路线**
5. 对国内外相关经典文献进行调研、分析、整理；
6. 分析多目标跟踪雷达波束调度的状态空间、行动空间、量测空间、状态转移函数与观测函数，将目标状态后验概率分布作为置信状态，在雷达波束使用代价约束下，以跟踪精度作为待优化目标，建立基于POMDP的波束调度模型；
7. 假设目标为匀速直线运动，量测在雷达坐标系下，无杂波，检测概率不为1，采用粒子滤波估计目标状态，采用概率数据关联方法解决量测来源不确定问题，对b中所建立的待优化问题，采用贪婪算法计算针对不同目标应该选择的波束；
8. 假设如c，针对非短视下的调度问题，基于动态规划方法，编写Matlab仿真程序，给出小规模问题下的最优解；
9. 假设如c，进一步分析雷达波束调度问题，选取基策略，给出对应的Rollout算法，从理论上分析Rollout算法所能达到的性能及计算时间/空间复杂度，编写Matlab仿真程序，在算法性能、计算量等方面与短视策略、动态规划算法进行比较分析并给出算法结论。
   1. **论文组织**

# 调度算法研究

* 1. **贪婪算法**
     1. 算法思路

贪婪算法是一种短视算法，它会每次选择一个局部最优解，而不会考虑到该解对于今后的影响。一般来说，他的时间复杂度比较低，虽然贪婪算法并不从整体上最优加以考虑，导致它所作出的选择只是某种意义下的局部最优选择，这在工程实践中反而比较常用。

通常贪婪算法具有五部分构成：

1. 一个待选元素集合，从该集合中选择元素，得到问题的解决方案。
2. 一个选择函数，每次依据该函数进行选择，选择当前最优的情况，加入已选序列。
3. 一个可行性函数，用来判断，该候选是否可以被用于作为解决方案的一部分。
4. 一个目标函数，用来衡量最终解决方案。
5. 一个解决方案函数，用来展示发现完全解决方案的时间。

对于一些数学问题，贪婪算法可以产生很好的解，但是对一些可能并没有好的解。对于可以产生优秀解的问题，一般有贪婪选择和最优子结构这两个特性。

我们在每一刻都选择当前时刻最优的解，然后解决以后出现的子问题。由贪婪算法做出的选择可能取决于迄今取得而不是对未来的选择或者所有的解决方案的子问题的选择。它反复让人贪婪的选择，将每个给定的问题变成一个较小的一个。换句话说，贪婪算法从未重新考虑其选择。这不同于动态规划，每一个阶段后，动态规划基于在前一阶段作出的所有决定进行决策，并可能重新考虑前一阶段的算法路径得到解决方案。如果这个问题的最佳解决方案包含最优解的子问题那么这个问题呈现最优子结构。

* + 1. 算法流程  
       把求解的问题分成若干个子问题。对每一子问题求解，得到子问题的局部最优解。把子问题的解局部最优解合成原来解问题的一个解。

输入：重量序列，其中，容积b

输出：解的集合序列，价值

初始化解的序列，剩余容积，以及价值

for i = 1 to n(每一项) do

if (物体重量没有超过目前的容积) then

把物体i添加到解序列,

更新剩余容积 ，以及价值

else

Stop and return S, U

end if

end for

Return S, U

* 1. **动态规划算法**
     1. 算法思路  
        动态规划过程是：每次决策依赖于当前状态，又随即引起状态的转移。一个决策序列就是在变化的状态中产生出来的，所以，这种多阶段最优化决策解决问题的过程就称为动态规划。  
        动态规划的基本思想与分治法类似，也是将待求解的问题分解为若干个子问题（阶段），按顺序求解子阶段，前一子问题的解，为后一子问题的求解提供了有用的信息。在求解任一子问题时，列出各种可能的局部解，通过决策保留那些有可能达到最优的局部解，丢弃其他局部解。依次解决各子问题，最后一个子问题就是初始问题的解。  
        由于动态规划解决的问题多数有重叠子问题这个特点，为减少重复计算，对每一个子问题只解一次，将其不同阶段的不同状态保存在一个二维数组中。  
        动态规划与分治法最大的差别是：适合于用动态规划法求解的问题，经分解后得到的子问题往往不是互相独立的（即下一个子阶段的求解是建立在上一个子阶段的解的基础上，进行进一步的求解）。  
        能采用动态规划求解的问题的一般要具有3个性质：  
        (1) 最优化原理：如果问题的最优解所包含的子问题的解也是最优的，就称该问题具有最优子结构，即满足最优化原理。  
        (2) 无后效性：即某阶段状态一旦确定，就不受这个状态以后决策的影响。也就是说，某状态以后的过程不会影响以前的状态，只与当前状态有关。  
        （3）有重叠子问题：即子问题之间是不独立的，一个子问题在下一阶段决策中可能被多次使用到。（该性质并不是动态规划适用的必要条件，但是如果没有这条性质，动态规划算法同其他算法相比就不具备优势）
     2. 算法流程  
        动态规划所处理的问题是一个多阶段决策问题，一般由初始状态开始，通过对中间阶段决策的选择，达到结束状态。这些决策形成了一个决策序列，同时确定了完成整个过程的一条活动路线(通常是求最优的活动路线)。如图所示。动态规划的设计都有着一定的模式，一般要经历以下几个步骤。  
        初始状态→│决策１│→│决策２│→…→│决策ｎ│→结束状态  
        (1)划分阶段：按照问题的时间或空间特征，把问题分为若干个阶段。在划分阶段时，注意划分后的阶段一定要是有序的或者是可排序的，否则问题就无法求解。  
        (2)确定状态和状态变量：将问题发展到各个阶段时所处于的各种客观情况用不同的状态表示出来。当然，状态的选择要满足无后效性。  
        (3)确定决策并写出状态转移方程：因为决策和状态转移有着天然的联系，状态转移就是根据上一阶段的状态和决策来导出本阶段的状态。所以如果确定了决策，状态转移方程也就可写出。但事实上常常是反过来做，根据相邻两个阶段的状态之间的关系来确定决策方法和状态转移方程。  
        (4)寻找边界条件：给出的状态转移方程是一个递推式，需要一个递推的终止条件或边界条件。  
        一般，只要解决问题的阶段、状态和状态转移决策确定了，就可以写出状态转移方程（包括边界条件）。  
        实际应用中可以按以下几个简化的步骤进行设计：  
        （1）分析最优解的性质，并刻画其结构特征。  
        （2）递归的定义最优解。  
        （3）以自底向上或自顶向下的记忆化方式（备忘录法）计算出最优值  
        （4）根据计算最优值时得到的信息，构造问题的最优解
  2. **rollout算法研究**

rollout算法对各种动态和离散优化问题具有十分优异的表现。作为近似的动态规划算法，一个rollout算法，在每个决策阶段会通过以称为基策略的贪心策略模拟未来事件的方法来使得选择近似理想的方案。在大部分情况下rollout算法，可以保证执行情况和它们的基策略一样，如今已有一些理论成果显示采用rollout算法，性能会有额外的改善。根据现有文献的随机模型，本节分析被称之为consecutive rollout和exhaustive rollout的两种rollout方法，这两者采用简单的贪婪算法作为基策略。我们分析这两种算法在普遍问题下的应用场景及算法流程，然后结合我们的实际问题，对这两种算法进行分析。

* + 1. **consecutive rollout**

consecutive rollout算法对于每一次迭代，会对所有可能情况进行一次遍历，分别计算出不同情况的价值，然后比较这些价值，选取较大的价值。下面以背包问题为例，表示其算法流程。

输入：重量序列，其中，容积b

输出：解的集合序列，价值

初始化解的序列，剩余容积，以及价值

for i = 1 to n(每一项) do

估计加入物体i的价值，

估计未加入物体i的价值，

if (加入物体的估计值更大) then

把物体i添加到解序列,

更新剩余容积 ，以及价值

end if

end for

Return S, U

* + 1. **exhaustive rollout**

exhaustive rollout 算法在每次迭代中，用t进行索引，该算法考虑在现有的顺序以及移动每个项目到序列的前部这些不同的顺序，然后对他们进行基策略得到所有可能对应的价值函数值。然后，该算法将具有最高估计值的项添加到容积中。在下一次迭代中，继续用项目的其余序列前进。下面以背包问题为例，表示其算法流程。

输入：重量序列，其中，容积b

输出：解的集合序列，价值

初始化解的序列，剩余容积，以及价值

for t = 1 to n do

for(剩余项序列中的每一项) do

用表示将i移动第一位的序列

估计价值序列，

end for

if then

决定具有最大估计值的项,

把项添加到解序列，

更新剩余容积 ，以及价值

end if

end for

Return S, U

我们可以发现，对于 consecutive rollout 算法，他会依次对当前的是否选择情况进行一个基策略，看是选择还是不选择的结果会好一些，对于 exhaustive rollout 算法，它需要对每次剩余的结果进行穷举，然后把其中的某一项移动最前方，对该序列再次进行遍历，然后从其中选择一个最优的。  
结合我们的实际情况，本文需要调度的是M个传感器的选择问题，每一步的选择情况都是固定的，是一个和M有关的表达式。可以很方便的使用 consecutive rollout 算法。而在传感器的选择方面，具有很强的时序相关性，时间序列无法随意改变，所以 exhaustive rollout 算法显然不适用于本文的情况。  
另外本文采用的rollout算法，均为单次迭代算法，在今后的工作中我们可以考虑rollout算法的第二次迭代。一个相关的议题是仍考虑rollout算法只是第一次迭代，但具有较大的超前长度（如试图对所有项目进行了 exhaustive rollout，而不仅仅是单独的每个项）。

# 粒子滤波算法研究

* 1. **基本算法**

粒子滤波器是贝叶斯滤波器的一种可替代的非参数实现，粒子滤波器通过有限数量的参数逼近后验概率。粒子滤波器的核心思想是由一组根据这个后验概率得到的状态的随机样本来表示后验概率。粒子滤波用一组从这个分布得到的采样来表示这个分布，而不是由参数形式表示分布(例如表示一个正态分布概率密度的指数函数)。这样的表示是近似的，但它是非参数，因此它可以表示更广阔的分布比空间。

在粒子滤波器中，一个后验分布的采样称为粒子，由下式表示

‑

每个粒子是在时间t状态的的具体实例，也就是，对于在真实时间t状态可以是在什么的一个假设。 在这里M表示粒子数。在实践中，粒子的数量M通常是一个大数，例如，。在一些实现中，M是一个t或者其他和belief 有关量的函数。

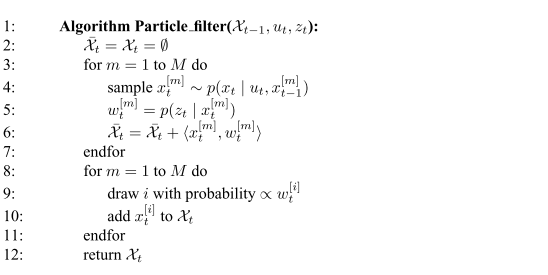


表 3‑1

表 3‑1粒子滤波算法，基于重要性采样的贝叶斯滤波器的变体。

粒子滤波器的重点是根据粒子集合近似belief state 。理想的情况下，状态假设包括在粒子集合中的可能性应该正比于它的贝叶斯滤波后验概率:

‑

由上式可知，一个状态空间的一个子区域由采样填充的越密集，真实的状态就越可能落入这个区域。正如我们将在下面讨论，对于标准粒子滤波算法该特性仅在M近似于时成立。对于有限的M，粒子是从具有略微不同的分布中得到。在实践中，只要粒子的数量不能太小(例如，)，这种差异是可以忽略的。

粒子滤波算法递归构建belief 根据上一个时间步长的belief 。因为belief是由粒子集合表示，这意味着粒子滤波器构造的粒子从到 递归。粒子滤波器算法的最基本的变体可以参考表 3‑1。

该算法的输入是粒子集合，最近的控制和最近的量测。该算法首先构造一个临时粒子集合，它类似于(但不等同)的。它如下系统地处理输入粒子集合中的每个粒子

1. 第4行基于粒子和控制在时间t产生一个假想状态。所得采样由m作为索引，这表明它是由的第m个粒子产生的。该步骤包括从下一状态分布的采样。对于任意分布，没有从状态转移概率采样的统一公式。然而，生产实践中的许多主要分布具有有效的采样算法。步骤4中这组由迭代m次产生的粒子是的滤波器的表示。
2. 第5行计算每个粒子的重要性因子，记作。重要性因子用于合并观测进入粒子集合。因此，重要性因子是粒子得到观测的的概率。即，。如果我们的把考虑为一个粒子的权重，那么加权粒子集合表示(近似地)贝叶斯滤波器后验概率。
3. 第8行至11行。这些行实现了所谓的重采样或重要性重采样。该算法对于临时集合中的M个粒子判断是否需要。保留每个粒子的概率是由它的重要性权重给出。重采样将一个具有M个粒子的集合转换为另一个具有相同的大小的粒子集。通过将重要性权重纳入重采样过程中，粒子的分布因此而变化：在重采样步骤之前，他们根据分布，经过重采样后，它们近似根据后验概率。事实上，所得的样本集合通常具有许多重复的。更重要的是没有被包括在中的粒子：那些于具有较低重要性权重的粒子。

重采样步骤具有强制粒子返回后验概率的重要作用。实际上，另一种(通常比较差)版本的粒子滤波器将永远不会重采样，而是将维持每个粒子的重要性权重，权重被初始化为1然后根据下式更新：

3‑3

这样的粒子滤波算法仍接近后验概率，但它的许多粒子将在后验概率很低的区域结束。其结果是，它需要更多的粒子，需要的数量的多少取决于后验概率的大小。重采样步骤是优胜劣汰的达尔文想法的概率实现：重新调整它的粒子集合到高后验概率状态空间的区域。通过这样做，它把滤波器算法的计算资源分配到最重要的状态空间所在区域。

* 1. **重要性采样**

用于粒子滤波器的推导，应当证明是有用的，下面以更详细地步骤讨论采样。图 3‑1显示了采样步骤下的intuition。图 3‑1 a展示了一个被称为目标分布的概率分布的密度函数f。我们想实现的是从计算采样。然而，从f直接采样几乎是不可能的。相反，我们可以从相关密度生成粒子，在图 3‑1 b标记为g。对应于概率密度g的分布称为建议分布(proposal distribution)。概率密度g为必须满足函数意味着，使得对于从任何f可能生成的状态，根据g进行采样时，都可以有一个非零的概率来生成一个粒子。在图 3‑1b的底部示出的所得到的粒子集合，是根据g而不是f分布的。特别地，对于任何一个区间范围(或更一般地，任何博雷尔集A)粒子的经验计数落入一个收敛于A下对g的积分：  
(4.25)  
为了消除f和g之间的不同，粒子是由商加权  
(4.26)

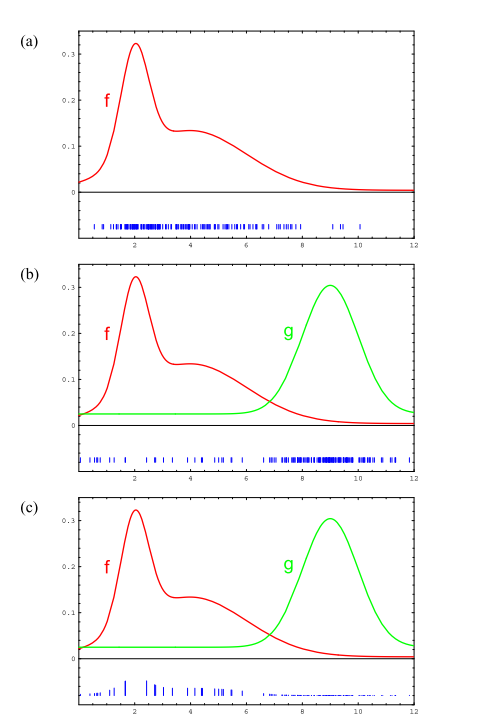


图 3‑1

这由图 3‑1c表示：该图中的竖线表示重要性权重的大小。重要性权重是每个粒子的非归一化概率质量。特别地，我们有

‑  
其中第一项作为用于所有重要性权重的正规化因子。换言之，尽管我们由密度g生成粒子，适当加权的粒子收敛于密度f。显然，一个粒子集合表示一个离散分布，而在我们的例子中是连续的。由于这个原因，就没有可能与粒子集合相关联的密度。因此，收敛的是的累积分布函数，而不是密度本身。重要性采样的一个很好的特性是如果在时都有，那么它趋近于真密度。在大多数情况下，收敛律为，其中是采样的数目。恒定系数取决于和的相似度。  
在粒子滤波器中，密度f对应于目标belief 。在此假设下，中的粒子根据分布，密度g对应于乘积分布：

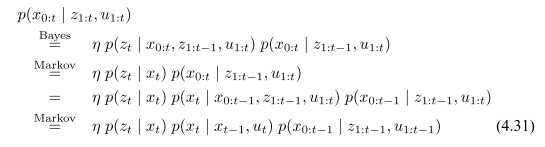
‑  
这个分布称为建议分布。

* 1. **公式化**

为了获得粒子滤波器的数学表示，我们可以把粒子当作状态序列的采样

‑  
相应地修改该算法也很容易：只需追加粒子到状态采样序列这种粒子滤波器计算在所有的状态序列上的后验概率：

‑  
而不是belief 。诚然，在所有状态序列上的空间是巨大的，并且用粒子覆盖它通常是显然不可行。然而，这不应阻止我们在这里，因为这个定义只是推导粒子滤波算法的手段。  
根据文献(Thrun 2000)我们有，如下公式



‑

验证初始条件是不重要的的，假设根据先验概率采样获得我们的第一个粒子集。让我们假定粒子集在时刻，根据分布。对于这个集合中的第m个粒子在我们的算法的步骤4中产生的样本是从建议分布产生的：

3‑9

3‑

常量η没有任何作用，因为重采样发生概率与正比于重要性权重。通过重采样概率正比于重要性权重的粒子，所得的粒子根据建议分布与重要性权重的乘积分布

‑  
(请注意，常数系数这里不同于3‑10中的因子。)在表 3‑1的算法转换为从一个简单地观测得到，如为根据分布，则状态采样根据分布。  
正如我们下面讨论的，由于我们考虑到规一化常数，这个推导仅对于正确。然而，即使是有限的M，它也是有效的。

# POMDP建模

现在让我们将焦点切换到部分可观马尔科夫决策过程的问题。在马尔科夫决策过程中只解决动作中的不确定性，但他们假设状态可以有把握地确定。对于完全可观马尔可夫决策过程，已经存在很多很成熟的算法来解决问题。现在，我们考虑更一般的情况，其中的状态不是完全可观的，也即缺乏预测性。缺乏观测性是指传感器只能在可能的状态估计一个P后验分布，我们称之为belief。此集合也称为部分可观马尔可夫决策过程，或POMDPs。  
在本章的其余部分要解决的核心问题是：对于POMDPs，制定规划和控制算法。最优策略算法仅存在于有限的世界，其中状态空间，动作空间，观测空间，以及计划时域T均为有限的。不幸的是，这些精确的方法的计算是十分复杂的。除此之外，对于连续的情况下，最有名的算法都是近似的。  
在MDPs中的中心更新方程如下

4‑1

在POMDPs中，我们应用非常接近的想法，然而，状态并不是可观的。所以，我们只可以得到状态的后验分布belief state b。因此对于POMDPs的计算方程如下

4‑2

并且我们使用这个方程来生成控制

4‑3

不幸的是，在belief空间计算价值要比在状态空间复杂的多。belief是一个概率分布，因此，在POMDPs中的价值是有关概率分布的函数。如果状态空间是有限的，由于belief space是所有状态空间的分布，belief state就是连续的。对于连续状态空间的问题，这个情况更加复杂，在这种情况下，belief space是一个无线维的连续体。除此之外，公式4‑2和公式4‑3对于所有的beliefs 进行积分。由于belief space的复杂度，积分并不容易精确计算或者有效地近似，幸运的是，对于有限的情况，存在确切的解法。

* 1. **有限环境**

下面考虑一种十分重要的特殊情况，有限环境。在这里，我们假设我们有一个有限状态空间，在每个状态都具有有限的动作，有限数量的观测，和有限的计划时域。在这些情况下，最优值函数可以被精确计算，也即具有最优策略。然而这并不是对所有情况都很明显，即使状态空间是有限的，事实上可以有无限多的可能的belief。然而，在这一点，下面我们建立一个众所周知的结果，最优值函数是凸的并且由有限多个线性部分组成。因此，即使该值函数被定义在一个连续体，它可以在计算机中以浮点数的精度来表示。

* + 1. **一个说明性例子**

一个有限的世界的一个例子参考图 4‑1。这个具体的例子很简单但他可以帮助熟悉POMDPs的关键问题。我们注意到，图 4‑1具有四种状态至，两个动作 和，和两个观测和。初始状态是在两个顶状态和中随机抽取，现在机器人给出一个选择：执行动作，这将具有高概率(但不总是)瞬间移动到相应的其他状态。另一种可选择方案，它可以执动作作，这导致这两个底状态或中的一个，转变的概率的如图所示。在中，机器人将收到一个大的正收益，而在的收益为负。这两个状态是终止状态，也就是说，一旦进入任务就结束了。因此，机器人的目标是在状态时要执动作作。如果机器人知道它是在什么状态，执行任务是非常简单：简单地套用动作，直到状态是，然后应用动作。然而，机器人不知道其状态。相反，它可以感知两个观测和。不幸的是，这两个观测可能在这两种状态，虽然概率不同，这取决于观测的机器人是什么状态。由于机器人并不知道它的初始状态是还是，它必须仔细跟踪过去的观测来计算它的belief。因此，关键的策略问题是：什么时候应该机器人尝试动作？具有多大的置信度去执动作作，需要多长时间来尝试这个动作？这些和其他问题将在下面进一步描述。

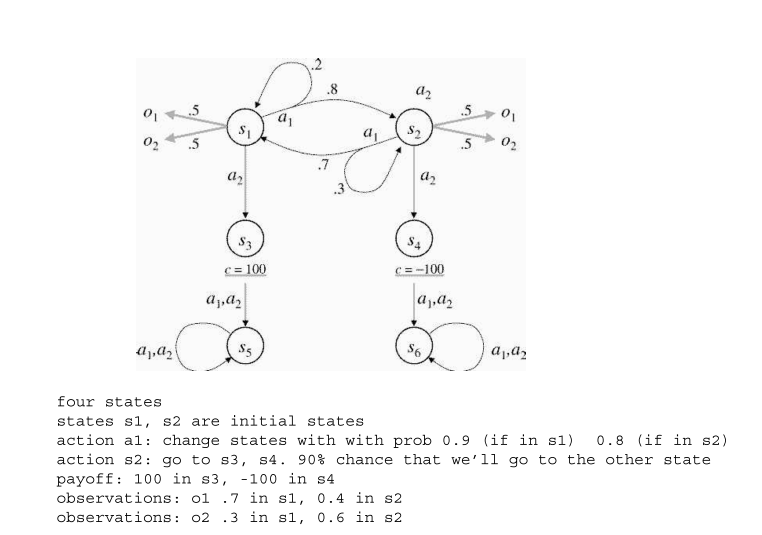


图 4‑1

本节的目标是实现一种在有限状态和有限时域T的情况下来计算最优值函数的算法。让我们用T来表示计划时域，用表示状态。利用状态空间的有限，我们注意到，belief 由n个概率给出

4‑4

其中。  
belief必须满足的条件

‑  
因为最后一个条件，b(s)可通过n-1个参数而不是n个参数指定。剩余的概率的可以根据如下公式计算：

‑  
因此，如果状态空间是有限的，并且大小为n的话，则belief是一个(n-1)维时域。  
在我们的图 4‑1示出的例子中，因为有四个状态，所以belief是由三个数值的概率值指定的。然而，操作将状态和状态分开。因此，该机器人可能遇到的唯一的不确定性是状态和或者状态和之间的的混淆。两者均可以由一个单一的数值的概率值来表示，由此，belief状态的唯一连续成分是一维的。这使得这个例子方便绘得到函数。  
现在，让我们专注于一个问题，我们是否可以计算出最优值函数，在有限的领域准确地最优策略。如果没有，我们需要尽可能的趋近于它。起初，人们可能由于belief空间是连续的，所以会认为计算最优值函数是不可能的。但是，我们观察到的有限的情况下，值函数有一个特殊的特性：它是由有限多个线性函数构成。这使得能够计算有限时间内的最优值函数。我们还注意到，价值函数是凸的和连续的，这后两种特性也适用于最优值函数在连续状态空间和无限的规划时域。  
我们考虑在开始的belief状态b的直接收益。b的收益可以由下式计算所得：

‑

根据b唯一地由概率确定，我们可以得到：

‑

它对于确实是线性。

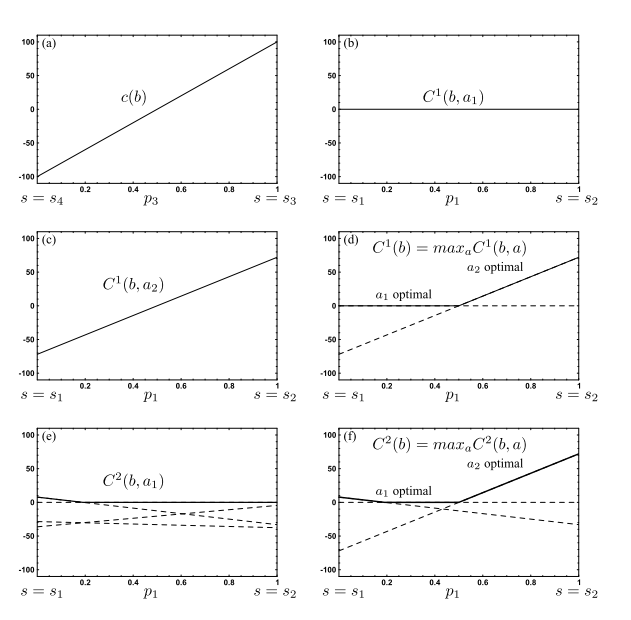


图 4‑2期望收益c(b)为belief参数 的函数，假设机器人或者在状态或。

易知，在状态回报是100，而在状态的回报是-100。图 4‑2a展示了函数c(b)通过定义的belief子空间，这个子空间展示了出现状态和所有的可能性。这里的期望c(b)是概率的函数。显然，如果时，环境状态是，并回报将是。另一方面，如果，环境的状态是和回报将是。在此期间，期望是线性的，从而导致图 4‑2a所示的曲线图。

这种考虑使我们能够对于规划时域计算价值函数。从现在开始，我们将只考虑belief空间的所有的可能性是和这两种状态子集。这种belief空间可以由一个单一的参数参数化，因为和。值函数对于动作恒为零：

‑

因为无论机器人的真实状态，动作不会导致它的状态，使得它得到非零的回报。该值函数如图 4‑2b所示。对于动作，如果环境的状态是，这个动作将有的几率得到状态，机器人将得到的100的收益，有的概率会在状态结束，它的回报是-100。因此，在状态的预期收益为。同理，状态的预期收益为-80。在这两者之间，期望是线性的，产生的价值函数

‑

这里我们取贴现因子。这个函数表示在图 4‑2c，对于该类型的belief。

那么，什么是正确的动作选择策略？使得预期收益最大的动作取决于我们目前的belief，假设它准确反映了真实的情况。如果概率，最佳动作将是，因为我们期望正回报。对于值小于0.5的情况，最佳动作将是，因为它避免了与动作相关的负面预期收益。对应的值函数是对于特定动作使得值函数最大：

4‑

(16.11)  
该值函数和对应的动作选择策略如图 4‑2d所示，虚线部分说明了最大化两个线性成分。我们注意到，这个数值函数不是整个区间上的线性函数。相反，它是分段线性和凸的。非线性来源于对不同部分的belief空间的最优策略的不同。  
根据图 4‑2d的值函数，我们无法得出对于的belief没有办法获得正收益的结论。因为，该值函数是仅适合于时域的情况。对于较大的T，可以先执动作作，接着执行动作。执行动作具有两个有益效果：首先，它有助于我们估计当前状态是否更好。第二，以高概率将状态从转移至或相反。因此，一个良好的策略可能是执行动​​作，直到我们有理由相信，机器人的状态是。然后，机器人应执行的动作。  
为了使这个更加一般化，让我们考虑时域时的最优值函数。假设我们执行的动作。然后，两件事情可能发生：我们要么得到观测或。让我们先假设我们观察到。那么新的belief状态可以使用贝叶斯滤波器来计算：

‑

其中，是贝叶斯法则的规则化系数。同理，我们计算在状态的后验概率：

‑

由于已知 ，我们计算: 

4‑14  
我们也注意到，变量是一个有用的概率：它是执行动作后观测到的概率。  
下面让我们计算belief状态的值，代入新的belief到值函数 (4‑11)和：

‑

我们现在将因子移出最大化函数并且将0.9移入，并得到：

‑

其是一个以belief b为参数的分段线性的，凸的函数。  
推导在执行动作后得到观察的方法类似。我们得到后验概率

‑

以及规一化系数 

‑

对应值为 

4‑19

另外，我们也注意到，。这是因为在贝叶斯滤波器中正规化系数是观察概率

4‑20

并且，在我们的例子中，恰好有两种可能的观测，和。  
现在让我们计算值，这是执行动作的期望值。显然，该值是值和，通过实际得到观察和分别的概率加权的和。我们有

‑  
术语已经如上所定义。概率执动作作之后观察是。因此，我们有所有成分计算所需值中：

‑

该表达式可以被重新表示为四个线性函数的最大值：

‑

图 4‑2e表示对于belief空间的四个线性函数。该值函数是这四个线性函数的最大值。易知，其中二者足以定义最大值;量歪两个在整个范围内均较小。这使我们能够改写为仅有两个，而不是四个变量的最大值函数：

4‑  
最后，我们要确定值，它是下面两个量中的最大值：

‑

容易验证，最大值函数中的第二项中是与上述规划时域T = 1的情况完全相同：

4‑

因此，我们根据(4‑24)和(4‑26)得到：

‑

这个函数是对于规划时域T的最优值函数，图 4‑2f展示了在belief子空间为情况下， 和其组成部分。容易发现，函数是分段线性和凸出的。特别是，它包括三个线性部分。对于在最左边的两部分belief(​​即)，是最佳的动作。对于，这两个动是一样好。对应于最右边的线性区域的belief，即的belief，最佳的动作是。如果为最优函数，下一个动作取决于在belief空间的初始点和观测。  
价值函数是仅适合于时域2。但是，我们的分析得到了几个重要的观点。  
首先，最优值函数在任何有限的时域是连续的，分段线性的和凸的。每个线性区域对应于在将来某个时候的一个不同的动作选择或者可以进行不同的观测。价值函数的凸性可以很直观的发现，即已知对于未知的先验。给定两个belief状态b和b’, belief状态值函数的混和大于或等于该belief状态混和的值函数，对于特定的混合参数：

‑  
二，线性区域数量可以极大地增长，特别是如果不注重线性函数的舍弃。在我们的例子中，定义)的四个直线约束中的两个是没有必要。有效实施POMDPs的“诀窍”在于尽早排除重复的线性函数，从而使得计算价值函数时，没有计算被浪费。不幸的是，即使我们仔细排除一切不必要的线性约束，线性函数的数量仍然可以增长得非常迅速。这是有限POMDPs的精确值迭代解决方案固有的扩展限制。

* 1. **belief空间中的价值迭代**

上一节展示了如何计算有限的价值函数。现在让我们回到在belief空间上价值迭代的一般问题。特别是，在这一章的介绍中，我们指出了价值函数，我们在这里简要地重申基本更新方程：

‑

我们将开发一个可在计算机上实现的通用的在belief空间计算价值迭代的算法。首先，我们注意到，公式(4‑2)提出对所有的belief进行积分，其中每一个belief是一个概率分布。如果状态空间是有限的，并且状态数为n，所有的概率分布的空间是具有n-1维的连续体。我们注意到，为了在n个离散事件确定概率分布，至少需要n-1个数值 (第n参数可以省略，因为概率加起来为1)。如果状态空间是连续的，belief空间具有无限多的维度。因此，对所有的belief的积分似乎是计算艰巨的任务。然而，我们可以通过重新制定问题，并对所有观测进行积分来代替。  
让我们看看条件概率P(b’|a，b)，其中规定了在分配后的belief b’给定一个belief b和动作a。如果只有b和a是已知的，该后验belief b’不是唯一的，而且P(b’|a，b)是belief的一个真正的概率分布。但是，如果如果我们也知道测量o’，执行动作时，后验b’是独特的，P(b’|a，b)是一个point-mass分布的退化。从动作执行之前的belief b，动作a，和随后的观测o’，贝叶斯滤波器计算一个单一的准确的后验概率belief b’。因此，我们得出结论，只要我们知道o’，对所有的belief的积分会退化。  
这种洞察力可以通过重新表达p利用

‑

其中，p(b’|a，b，o’)是由贝叶斯滤波器计算的point-mass 分布将积分代入价值迭代的定义(4‑2)，我们得到

4‑  
内积分

‑  
仅包含一个非零项。这其中b’是由b，a和o的分布通过贝叶斯滤波器计算得到。让我们用b(b，a，o’)来表示这个分布，那么有

4‑  
我们注意到，正规化分母，，是值更新方程(4‑31)的一个因子。因此，将代入(4‑31)消除了这一量，从而导致了值迭代算法的递归描述：

‑  
这种形式是比在(4‑2)更加方便，因为它只需要对所有可能的测量o’，而不是所有可能的belief分布b’进行积分，这种转变是派生此一章所有值迭代算法的一个重要观点。事实上，它是含蓄地用在上面的例子中，在一个新的价值函数是由有限多个分段线性函数混合获得的。  
下面，很容易根据积分获得最大化价值的动作。因此，我们会注意到(16.34)可以改写为下列两个等式：

4‑

4‑  
在这里，是时域t对于假设下一个动作a的belief b的函数。

* 1. **计算价值函数**

在我们的例子中，最优值函数是分段线性和凸的。这种情况，对于一般的有限情况均满足。对于所有具有有限时域的有限POMDPs的最优值函数都是分段线性和凸的。分段线性度意味着价值函数由线性函数的集合表示。如果是线性的，它可以通过一组系数来表示

‑  
其中，像往常一样，是一种belief分布的参数。在我们的示例中，分段线性和凸值函数可以通过K个线性函数集合的最大值表示

‑  
其中，分别表示第k个线性函数的参数，以及k表示线性片段的数量。一组有限的线性函数的最大值事实上是凸的分段线性的函数。

在值迭代，初始值函数由下式给出

‑  
我们注意到，这个数值函数是线性的，因此，它也是分段线性和凸。这种分配建立了递归值迭代算法的基础。  
我们现在衍生除一个计算值函数的递归方程。该公式假定上一步的值函数，由一个分段线性函数如上规定表示。作为派生的一部分，我们将假设是分段线性和凸的，是也分段线性和凸出的。下面将证明具有有限时域的值函数是分段线性和凸的。  
方程(4‑35)和(4‑36)定义如下：

‑

‑  
在有限的空间中，所有的积分可以通过有限和取代，我们得到：

4‑

‑  
belief 使用下面的表达式获得，该式为用和来替换(4‑33)中的积分所得到：

‑

如果belief b由参数表示，那么belief b’中的第i个参数由下式计算得到

4‑

为了计算价值函数，我们将现在的条件和中衍生出更多实用的表达式中)，从公式(4‑42)开始，具有参数，我们从得到了

‑  
代入4‑45，我们得到

4‑  
由于不依赖于i，后者变换是合法的。这种形式仍然包含一个难以计算的表达式：。然而，在有限的情况下这个表达式可以被消掉。因此，它不必被任何进一步考虑。  
在(4‑42)中对量的推导类似于上面。特别是，我们不得不用唯一的分段线性和凸的值函数C^(T-1)取代直接回报函数c。然后可以得到

‑  
。如上所述，我们代入后验概率belief b(见(4‑45))，并获得：

‑  
其中 不依赖于i或k，故可移出求和和最大化部分，得到：

4‑  
该表达式也是分段线性和凸的可能并不明显。然而，对该式进行整理得到下式：

‑

在这种形式下，很容易验证，对于任何的固定值k，使其最大化的参数是确实线性的参数。因而，这些线性段的最大值为分段线性和凸的。此外，线性段的数量是有限的。  
现在让我们回到本节中的主要问题，即计算和时域T的值函数。我们简要地重申方程(4‑42)：

4‑

让我们首先用公式(4‑47)和(4‑50)计算总和

‑  
经过一些整理得到，

‑

将这个表达式代入公式4‑52

‑  
我们注意到，项同时出现在在此表达式的分子和分母中从而抵消了，得到：

4‑

这个消除是仅对有限POMDP满足的。所需的价值函数通过对于所有动作a最大化  
得到

4‑  
很容易验证，在belief参数为的情况是确实分段线性和凸的。具体地，对于每个固定k(4‑56)的大括​​号内的量是线性的。这些k个线性函数的最大值是一个分段线性函数，包括最多有k段。分段线性的凸函数的总和为再次分段线性和凸的。因此，因此对于所有观测的乘积的总和再次产生一个分段线性凸函数。最后，对于在方程(4‑57)中的所有动作的最大值也是分段线性的凸函数。因此，我们的结论是在任何有限时域T，是分段线性和凸出的。

# 仿真实验结果

* 1. **模型建立**

我们的实验选取两个运动目标和四个传感器（雷达）。传感器0在该区域的中心，其他的3个传感器到传感器0 的距离为30公里，彼此以120度分离。所有四个传感器具有相同的观测范围，，并且具有相同的启动/停止成本0.1。传感器0比其它传感器使用成本更高，传感器3比其它传感器更准确。传感器的概要参数如下：

传感器0：在（0,0）km，花费0.5 /步骤，，，.

传感器1：在（-30，0）km，花费0.1 /步骤，，，。

传感器2：在（15，26）km，花费0.1 /步骤，，，。

传感器3：在（15，-26）km，花费0.1 /步骤，，，。

这两个运动目标由具有不确定加速度的NCV模型决定，在此例中使用的目标运动模型是具有高斯不确定性加速度的近匀速（NCV）模型[15]：

‑

这里的噪音和分别表示目标在和方向上的加速度不确定性，我们假设它们是独立的并且服从均值为0的高斯。

采样间隔为，我们模拟运行300秒。在我们的实验中，目标检测概率为，并且误报率，对应于PF的误报概率。我们在粒子滤波中使用2000个粒子。

* 1. **问题公式化**

一个POMDP [7,8]是由它的状态空间,动作空间，观测空间，状态转换律，观测律，初始状态分布，和一步成本函数指定。它基本上是一个马尔可夫决策过程（MDP），其中状态空间仅仅是通过L部分可观的。

在初始时刻,初始状态服从已知的分配，一个POMDP演变如下。在步骤k中，该系统的状态值和观测是已知的。然后选择一个动作会产生一个花费。在此之后，系统根据转化律，和由观测率随机生成观测。

由于状态是不能直接观测到的，一个POMDP保持跟踪belief state ，被定义为，以观测和操作历史记录为条件的状态的后验概率分布。这里的目标是要基于 belief state ，从一组可用动作选择一个动作使得预期总成本最小化。策略定义为从belief 状态到动作的映射的序列。

让预计总成本从初始belief 状态开始并且使用策略,经过H步后得到，其中，以及期望由所有可能的状态和观测序列接管。这里的目的是为了找到最佳策略最小化。

我们定义在状态采取动作的花费为.则在belief state 采取动作的值是

‑

其中是开始于下一个belief state 经过 步最佳值。贝尔曼对于POMDP问题的最优律表明最低期望总成本是由给出，并且，在第k步选择动作是最优的。因为一个动作的Q值总结了采取这一动作的未来成本，贝尔曼的原则产生了称为”lookahead”的控制方法。

当H很大的时候，最佳的策略可以被假定为静止的。在这种情况下，最佳的策略可以被近似为在每个步骤中仍然剩余H步。因此，最佳的操作是由下式给出

‑

这种方法被称为receding horizon control。由此产生的最优策略在当前的belief state下选择在h步之后最小化Q值的动作。

在这里所研究的传感器调度的情况下，有M个传感器分布在传感器区域内跟踪T个目标。中央控制器收集并且处理来自这些传感器的数据，并管理传感器的激活状态。我们的目标是选择可以折衷跟踪精度和传感器使用的传感器的数量和组合。

* + 1. **系统状态，动作和状态转换律**

系统状态向量包含T个目标的状态，M个传感器的状态，和滤波器的状态。在时间步骤k中，系统状态被写成

‑

‑

‑

‑

在这里，是目标的状态，其中包括该目标的位置和在笛卡尔坐标系下的速度。向量元素是传感器的激活状态。如果传感器在时间步骤k中激活，那么= 1;否则。滤波器状态可以描述任何用于belief state估计的滤波器。由于传感器调度的目标是要权衡传感器成本与跟踪误差，我们所需要滤波器的状态来估计跟踪误差，并且选择用于观测的传感器。在本文所考虑的特定设置下，滤波器是一个粒子滤波器，并且包含N个粒子，每个粒子是目标状态的一个采样

‑

动作是一个M维的向量，其中， = 1或 = 0来指定第m个传感器在时间步骤k+1是否激活以生成在时间k+1基于系统状态的观测。

状态转换律是由状态动态值定义，其中代表在状态转化过程中的随机性。如果我们假设目标独立地移动，则状态动态可以分解成每个目标的状态动态、传感器的状态转变和滤波器的状态转变

‑

传感器状态转换是由给出。滤波器状态的进展是由第3.1节的粒子滤波算法唯一定义

* + 1. **单步成本**

一步成本结合了目标跟踪误差和传感器的使用成本。在我们的实验中，我们使用

5‑

在这里，为目标i的由滤波器状态决定的估计位置，和分别表示上述传感器m的操作成本（每步）和开始/停止成本，是用来调整跟踪误差和传感器成本的相对重要性的权重因子。我们在第4节探讨改变的效果。

* + 1. **观测和观测律**

观测律取决于传感器模型。我们使用控制跟踪和数据合并的相关联的传感器模型[16]，在该模型中，每个传感器测量由若干观测组成。这些观测既可能是从既定目标有效的测量，或是来自杂波或者目标逃脱的误报。整体观测是来自所有激活的传感器的观测的集合，并且每个传感器扫描可以输出多个观测。假设在时刻k，传感器m的输出的包括个观测，整体观测可以写成

‑

连接每个传感器的是一个覆盖区域。在每个覆盖区域中，测量的质量与目标和传感器之间的距离相关。设表示目标i和传感器m之间的距离。如果或，传感器m不生成来自目标的任何观测。否则，传感器m生成基于目标的状态和检测概率的至多一个测量。

每个单独的观测包括目标或者错误警报的距离，角度和距离变化率。如果由目标生成，然后它取决于目标的如下的状态：

‑

其中，

‑

‑

‑

在这里，和是传感器m的（X，Y）位置，，和代表测量噪声，假定为零均值的高斯。由于测量方差与1 / SNR成比例，并且SNR与成比例[17]，方差，，和正比于。

我们假设误报的数量是具有参数（利率）的泊松分布。在空间上，误报在整个监测体积上均匀分布。

* 1. **算法流程**
     1. **粒子滤波流程**

如果我们忽略控制变量，在每一步belief state可以使用贝叶斯滤波更新，更新分为两个步骤：预测和更新[9,10]。POMDP的特点是在状态转移律和观测映射中的控制变量。预测步骤由下式给出[18,19]

5‑

更​​新步骤由下式给出

5‑

粒子滤波是上述通过蒙特卡罗模拟的最优递归​​贝叶斯滤波的实现。此方法使用一组N个粒子（采样）来近似belief state。在我们的问题中，只有系统状态（）中的目标状态部分（）的需要被估计，所以每个粒子代表目标的状态：。这里是狄拉克函数，是粒子的重要性权重。

在粒子滤波中，当新的观测变得可知时，滤波器状态递归更新。有许多粒子滤波算法的变中。我们使用重要性重采样（SIR）算法。在SIR中，滤波器状态更新分为三个步骤完成。首先，根据之前的粒子通过运动学先验分布，对采样产生新的粒子。然后，根据其与观测的吻合度更新粒子权重：

‑

最后，重采样步骤防止粒子退化。

对于多目标跟踪，有一个数据关联的问题：确定哪个观测与哪个目​​标相关联。这个问题来自于相关的测量模型，为了相关练得目标和观测配对，在目标状态上的观测依赖性必须明确给出。在粒子滤波上，这个问题是在多目标的情况下不能直接从得到。为了解决这个问题，我们用联合概率数据关联（JPDA）算法。在JPDA中的目标观测的关联假说的概率计算为已知的目标观测配对提供了和观测律之间的良好联系[20]。

我们首先考虑单一传感器的情况（假设它是传感器m）然后在后面包含多传感器数据融合。设，是所有可能的关联假设。假设的数目，，是由目标数量T和观测数量决定。设是根据关联假设 对于目标在中的观测索引，而0意味着有检测到目标。未分配到任何目标的观测被认为是误报。使用全概率定理，可以计算如下：

5‑

在JPDA中，概率，使用下式计算得到

5‑

这里和分别是在关联中误报和侦测到目标的数量，的是从目标获得观测的可能性，它是由测量噪声的分布来确定的。（对于上述的传感器模型，从目标在传感器上得到观测的可能性由下式给出

，

5‑

如果多个传感器被同时激活，需要应用多传感器融合来提高跟踪精度。从JPDA扩展到并行执行的多传感器JPDA（MS-JPDA）的扩展由[22]给出。后来，对持续MS-JPDA [23]进行了研究，并已证明其在跟踪性能方面是比并行计算更高效性能更优越。在持续MS-JPDA中，每次只有一个传感器的观测被解析。在每个传感器读数后，JPDA和卡尔曼滤波被用来计算中间belief state估计，然后下一个传感器的观测被用来进一步提高该中间状态估计。在这里，我们融合了持续MS-JPDA和粒子滤波，即在每个传感器读数后我们用粒子滤波而不是卡尔蛮滤波更新滤波器状态

集成持续MS-JPDA和粒子滤波的算法如下：

（ⅰ）初始化。

每个粒子，来自初始分布的采样，并设置，。

（ⅱ）预测。

对于每个粒子，

（a）对于每个目标根据预测

（b）使用下式构造一个新的粒子

（ⅲ）更新权重。

对于每个粒子。

（a）对于每个传感器，使用（5‑19），（5‑20）和（5‑21）计算权重

（b）计算权重。

规一化：对于， 。

（ⅳ）重采样。基于，从选择的N个粒子。

（ⅴ）设，并去预测步骤。

随着粒子的数量变得非常大，则粒子滤波器接近在（5‑16）和（5‑17）中的确切贝叶斯更新。然而，受限于计算机内存和计算复杂度，在模拟中使用的粒子的数目必然是有限的。这会在测量的方差远小于由粒子分布的方差的时候导致问题。在这种情况下，大部分粒子具有非常小的权重（有时比计算机可处理的最小数更小）除了最接近测量的少数粒子。这就导致在重采样阶段损失粒子多样性。随着时间的推移，跟踪可能会丢失。为了确保跟踪的稳定性，粒子的数目必须足够大以匹配测量精度。对于我们的传感器模型，测量的方差正比于R4并且在一个很宽的范围内变化。当目标非常接近于传感器，需要的粒子数超出了我们的计算能力。我们对于这个问题的解决方案是利用固定数量的粒子（在我们的模拟中是2000）并且轻微的改变似然度计算。如果，其中为在和之间的一些预定义的阈值时，我们在（5‑21）中使用固定，，和（而不是很小，和），人为地增加了粒子的权重和重采样后的多样性。通过这个变化，跟踪器更稳定，并具有更小的跟踪误差。

* + 1. **Rollout 算法流程**

让我们定义目标和传感器的有效距离，如果为否则为。对于单目标（目标0）和单传感器的情况，贪婪策略选择使得最小的传感器来观测目标0。此策略可以推广到包括多个目标，即，要选择传感器使得最小的作为多个目标的最近传感器。对于多传感器多目标跟踪问题，我们构造了一个基于贪婪策略的最接近传感器策略，该策略选择动作使得所有目标和它们最近的激活的传感器距离之和最小：

‑

为了激活最多两个传感器来跟踪两个目标，贪婪策略对每个目标激活最接近的传感器。如果对两个目标来说最接近传感器是相同的，只有该传感器被激活。为了基本策略和rollout之间公平比较，通过rollout策略选择激活的传感器的最大数量也是两个。

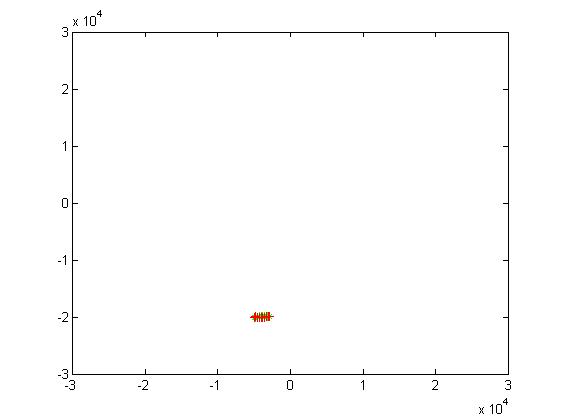
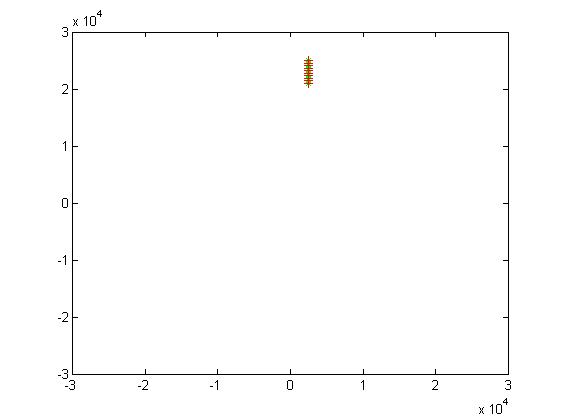
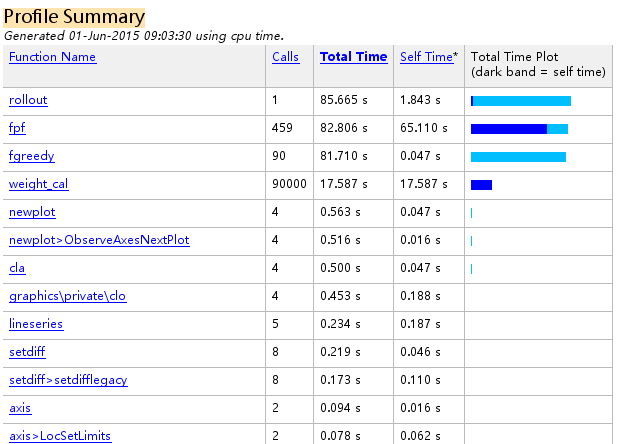
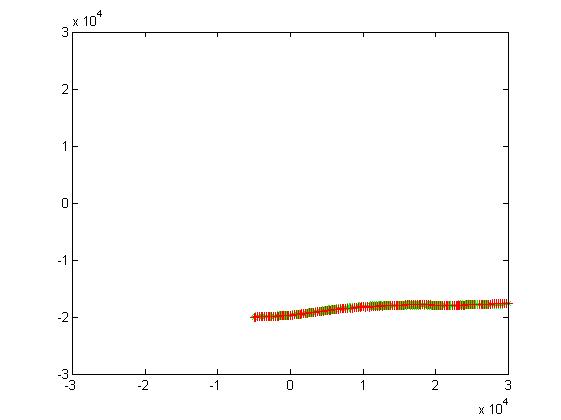
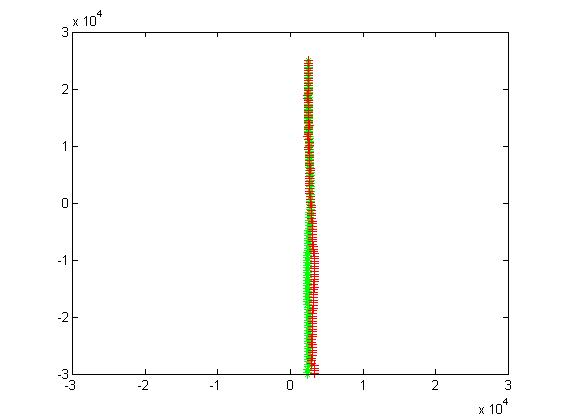
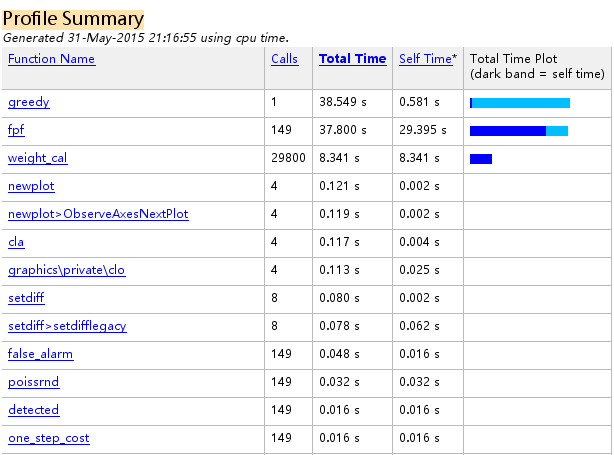
* 1. **实验结果**

我们比较贪婪策略（最多激活两个传感器）及使用贪婪策略作为基础策略的Rollout策略的性​​能。

图1和图2分别给出了采用贪婪策略和Rollout算法的两个目标的真正的轨迹（由绿线示出），目标位置估计（用红线示出），和运行时间。在这个例子中，目标0在(2.5,25)，处开始并且以222m/s的速度向南，目标1开始于(-5,-20)，以120m/s的速度向东移动。

我们注意到，在这个例子中我们的多传感器多目标跟踪器效果很好：估计的目标位置非常接近目标轨迹。这两者的主要不同体现在花费时间上，由于在rollout算法中，在第步会运行次贪婪算法，其中，所以rollout算法的时间复杂度是贪心算法的倍。

由于运行时间的问题，贪婪算法一共运行100拍，而rollout算法只运行了10拍，我们可以从运行结果发现，rollout算法总共调用了次贪婪算法，符合之前的理论分析。



通过调整（5‑10）中的值，优先级可以被放置在跟踪误差或传感器的成本上。随着减小，跟踪误差增加，而传感器使用成本减少。