**摘要**

**Abstract**

**绪论**

**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。

**国内外研究现状**

雷达波束调度也即传感器调度可以概括为在不同系统中对资源的动态分配理论的研究，受复杂的环境以及噪声影响，随机模型分析是其重要的数学基础。对于这种持续决策过程的问题，且在每次决策中上一次的观测具有的很大的不确定性，通常可以将其抽象为部分可观的马尔可夫决策过程（POMDP）。  
而对于POMDP问题，目前主流的算法可以分为三类[2]。  
第一类是myopic（短视）调度算法，例如文献[3-5]中的算法。对于myopic策略，其最主要的优点是计算简单、实践方便也可以获得比较好的结果。而缺点也是很明显的由于只考虑到当前的状态，所以在某些情况下会获得很差的结果，例如在传感器长时间无法捕捉到目标时。  
第二类是随机动态规划算法，这是一种理想的算法，其计算结果是最优的，但是由于维数灾难会导致计算量过大，只适合解决小规模问题，不适合应用于工程实践当中。在此基础上，人们又提出了近似动态规划算法，例如人工神经网络、遗传算法[6]、模拟退火算法[7]等。此种算法通过近似解决了维数灾难问题，也可以得到较优解。  
第三类是Rollout算法，Rollout算法是一种启发性算法,由Tesauro和Galperin为解决蒙特卡洛问题所首次提出[8]，其思想源于动态规划的策略迭代思想[9]。而传感器的目标跟踪问题即可采用蒙特卡洛方法解决,通过对问题建立概率模型,采用反复随机抽样的手段,求解出结果[10-12]。Rollout算法以一种短视算法，例如Greedy算法，作为基础算法，然后实现Rollout算法。Rollout算法分为两类，分别是 consecutive rollout 和 exhaustive rollout[13] 。Rollout算法是一种十分适合处理概率抽象问题的算法，并且其是一种边界可控[14]、性能优  
于Greedy等算法，且计算迅速可用于多目标跟踪问题[15]的算法。  
- rollout算法  
- 雷达波束调度  
- POMDP 问题  
- 粒子滤波算法

**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

**技术路线**

1. 对国内外相关经典文献进行调研、分析、整理；
2. 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；
3. 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2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；
4. 假设如3，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；
5. 假设如3，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。  
   预期目标:  
   采用Rollout方法实现对单雷达多目标跟踪的多波束调度。仿真比较分析Rollout方法与贪婪算法、动态规划方法的性能。

**论文组织**

**控制算法研究**

**动态规划算法**

动态规划算法是一个精确的算法。动态规划，简单说来就是知道一个状态，然后根据状态转移方程得到另外一个状态。根据这两个状态间的转换，递推获得最终解。  
考虑这样一个问题，假如问题有n个阶段，每个阶段都有多个状态，不同阶段的状态数不必相同，一个阶段的一个状态可以得到下个阶段的所有状态中的几个。那我们要计算出最终阶段的状态数自然要经历之前每个阶段的某些状态。  
如果下一步最优是从当前最优得到的，所以为了计算最终的最优值，只需要存储每一步的最优值即可，解决符合这种性质的问题的算法就变成了前文提到的贪婪算法。  
如果一个阶段的最优无法用前一个阶段的最优得到呢？这时你需要保存的是之前每个阶段所经历的那个状态，根据这些信息才能计算出下一个状态！每个阶段的状态或许不多，但是每个状态都可以转移到下一阶段的多个状态，所以解的复杂度就是指数的，因此时间复杂度也是指数的。这种之前的选择会影响到下一步的选择的情况就叫做有后效性。  
每个阶段的最优状态可以从之前某个阶段的某个或某些状态直接得到这个性质叫做最优子结构；

而不管之前这个状态是如何得到的这个性质叫做无后效性。  
动态规划算法的设计可以分为如下四个步骤

1. 描述最优解的结构
2. 递归定义最优解的值
3. 按自底向上的方式计算最优解的值
4. 由计算出的结果构造最优解

**贪婪算法**

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。

**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。虽然在很多情况下rollout算法，可以保证执行情况和它们的基策略一样，已经出现了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，本文分析被称之为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。

**consecutive rollout**

* 算法思路  
  consecutive rollout算法的程序流程见下面，该算法输入为。。。，输出为。。。。，并调用贪婪算法作为其子程序。  
  算法流程。。
* 边界分析

**exhaustive rollout**

* 算法思路  
  在每次迭代中，用t索引，该算法考虑在现有的顺序一它计算由移动每个项目序列的前部，施加致盲贪婪算法得到的值的所有项目。然后，该算法具有最高估计值添加项（如果它存在）到溶液中。我们暗含的假设一致的打破平局方法，如该项目优先考虑以最低的指数。下一次迭代，然后用物品的其余序列前进。  
  考虑  
  算法流程。
* 边界分析

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**

**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。就像histogram滤波器，粒子滤波器通过有限数量的参数逼近后验概率。但是，它们产生这些参数的方式以及状态空间的转移是不同的。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bel(xt)。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

X\_t := x\_t^([1]),x\_t^([2]),…,x\_t^([M])  
每个粒子x\_t^([m])(1≤m≤M)是在时间t状态的的具体实例，也就是，对于在真实世界时间t状态可以是在什么的一个假设。 在这里M表示

书78PR 90pdf页数  
表4.3 粒子滤波算法，基于重要性采样的贝叶斯滤波器的变体。

粒子数。在实践中，粒子的数量M通常是一个大数，例如，M = 1000。在一些实现中，M是一个t或者其他和belief bel(x\_t)有关量的函数。  
粒子滤波器的重点是根据粒子集合X\_t近似belief state bel(x\_t)。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bel(x\_t)：

x\_t^([m])~p(x\_t|z\_(1:t),u\_(1:t))

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，真实的状态就跟可能落入这个区域。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M，粒子是从具有略微不同的分布中得到。在实践中，这种差异是可以忽略的，只要粒子的数量不能太小（例如，M≥100）。

粒子滤波算法递归构建belief bel(x\_t)根据早一个时间步长的belief  
bel(x\_(t-1))。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XT-1到XT递归。粒子滤波器算法的最基本的变体可以参考<(￣3￣)>  
表！ 。该算法的输入是  
粒子集合XT-1，最近的控制UT和最近的量测ZT。该算法首先构造一个临时粒子集合ˉx，它类似于（但不等同）的bel(x\_t)。它如下系统地处理输入粒子集合X\_(t-1)中的每个粒子X [M]  
1. 第4行基于粒子×〔米]叔1和控制UT在时间t产生一个假想状态。。。。 t。所得采样由m作为索引，这表明它是由XT-1的第m个粒子产生的。该步骤包括从下一状态分布p（XT | UT，XT-1）的采样。对于任意分布P（XT | UT，XT-1），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bel(x\_t)的滤波器的表示。  
2，第5行计算每个粒子X [M]的重要性因子，记作W [M]。重要性因子用于合并观测ZT进入粒子集合。因此，重要性因子是粒子×〔米得到观测ZT的的概率。即，瓦特[米]  
T = P（ZT | X [M] T）。如果我们的把W [M] T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bel(x\_t)。  
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ˉXT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bel(x\_t)分布，经过重采样后，它们近似根据后验概率bel(x\_t) η= P（ZT | X [M]T）bel(x\_t)。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XT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bel(x\_t)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：  
W [米]T = P（ZT | X [M] t）的W [M] T-1（4.24）

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。

**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重采样。图4.2显示了采样步骤下的intuition。图4.2A示出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f计算采样。然而，从f直接采样可能是不可能的。相反，我们可以从相关密度生成粒子，在图4.2B标记g。对应于概率密度g的分布称为(proposal distribution)建议分布。概率密度g为必须满足函数f（x）> 0意味着g（x）的> 0，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。然而，所得到的粒子集合，在图4.2B的底部示出，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Borel集合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x^([m])是由商加权

（4.26）  
p81 tu%%%%%%%%%

这由图4.2C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  
4.27）  
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。  
具体的趋近设计对于集合A的一个积分。显然，一个粒子集合表示一个离散分布，而在我们的例子中f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，收敛是f的累积分布函数，而不是密度本身。重要抽样的一个很好的特性是如果在f（x）> 0时都有g（x）> 0它趋近于真密度。在大多数情况下，收敛律为O(1/sqrt(M))，其中M是采样的数目。恒定系数取决于f(s)和g(s)的相似度。  
在粒子滤波器中，密度f对应于目标belief bel(x\_t)。在此假设下，X\_(t-1)中的粒子根据bel(x\_t-1)分布，密度g对应于乘积分布：  
P（XT | UT，XT-1）bel(x\_t-1) （4.28）  
这个分布称为建议分布。

**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  
(4.29）  
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：  
bel（X0：T）= P（X0：T | U1：T，Z1：T）（4.30）  
而不是belief bel(x\_t)= P（XT | U1：T，Z1：T）。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xxx我们有，如下公式  
（4.31）

验证初始条件是不重要的的，假设根据先验概率P（X0）采样获得我们的第一个粒子集。让我们假定粒子集在时刻t-1，根据bel（X0：T-1）分布。对于这个集合中的第m个粒子X [米]  
0：T-1，在我们的算法的步骤4中产生的样本x [M] T是从建议分布产生的：  
ρ（XT | XT-1，UT）bel（X0：T-1）=ρ（XT | XT-1，UT）P（X0：叔1 | Z0：T-1，U0：T-1）4.32）  
（4.33）  
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proposal与重要性权重的乘积分布  
（4.34）  
（请注意，常数系数η这里不同于（4.33）中的因子。）在表4.2的算法转换为从一个简单地观测得到，如果x [米]0：t为根据bel（X0：吨）分布，则状态采样x [米]t根据bel(x\_t)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M→∞正确。然而，即使是有限的M，它也是有效的。

**误差来源**

**POMDP建模**

现在让我们将焦点切换到部分可观的问题。在马尔科夫决策过程中只解决行动中的不确定性，但他们假设世界的状态可以有把握地确定。对于完全可观马尔可夫决策过程，我们设计了一个数值迭代算法用于控制随机域中的机器人。现在，我们将有兴趣在更一般的情况，其中的状态不是完全可观的。缺乏观测性是指机器人只能估计一个P后验分布在可能的世界状态，我们称之为belief。此设置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我们能否制定规划和控制算法？答案是肯定的，但具有前提。寻求最优策略的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，以及计划向量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。对于更有趣的连续的情况下，最有名的算法都是近似的。  
在MDPs中的中心更新方程如下  
15。14  
在POMDPs中，我们应用非常接近的想法，然而，状态并不是可观的。我们只可以得到states 的后验分布belief state b。因此对于POMDPs的计算方程如下  
16.2  
并且我们使用这个方程来生成控制  
16.3  
不幸的是，在belief空间计算价值要比在state空间复杂的多。belief是一个概率分布，因此，在POMDPs中的价值C\_T是有关概率分布的函数，这就是问题所在。如果状态空间是有限的，由于belief space是所有状态空间的分布，所以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。。和公式。。对于所有的beliefs b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。  
本章安排如下，  
第二节，讨论了一种更加普遍的同时考虑两种不确定性：动作不确定性和观测不确定性的调度算法。这种算法可以被应用于belief space的表示，在这种算法下的框架称作Partially Obeservable Markov Decision Processes(POMDPs)。这种算法可以估计不确定性，高效的获得信息，并且可以提高精确度在增加计算复杂度的情况下。  
16.2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，在每个状态具有有限的动作，有限数量的观测，和有限计划时域.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。  
16.2.1说明性的例子  
一个有限的世界的一个例子示于图16.1。这个具体的例子极端简单，其唯一的作用是在讨论一般的解决方案前帮助熟悉POMDPs的关键问题。我们注意到，图16.1具有四种状态，标记S1至S4中，两个动作，a1和a2，和两个观察，标记O1和O2。初始状态是随机抽取来自两个顶端状态，s1和s2。现在机器人给出一个选择：执动作作A1，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A2，这导致在这两个底状态，S3或S4的概率中的一个的转变所示。在S3中，机器人将收到一个大的正收益，而在S4的收益为负。这两个state是终端的状态，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S1时要执动作作A2。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A1，直到状态是S1，然后应用动作A2。然而，机器人不知道其状态。相反，它可以感知两个观测O1和O2。不幸的是，这两个观测可能在这两种状态，虽然观察一比其他的概率不同，这取决于什么状态的机器人是由于机器人并不知道它的初始状态是S1或S2，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A2？具有多大的置信度去执动作作a2，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。  
本节的目标是开发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。让我们表示用T来表示计划时域和用S1,…,SN表示状态。利用状态空间的有限，

图16.1有限状态环境，用来说明价值迭代的belief空间。

我们注意到，belief B（S）由n个概率给出  
圆周率= B（S= SI）（16.4）  
其中i=1，。 。 。 ，N。  
belief必须满足的条件  
PI≥0  
（16.5）  
因为最后一个条件的，B（S）可通过n-1个参数p1,…,p\_n而不是n个参数指定。剩余的概率的pn可以根据如下公式计算：  
（16.6）。  
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16.1示出的例子中，可能出现的是由三个数值的概率值指定的belief，因为有四个状态。然而，操作A2将状态{S1，S2}和状态{S3，S4}分开。因此，该机器人可能遇到的唯一的不确定性是状态s1和s2或者状态S3和S4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制值的函数。  
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，准确地在有限的领域最优策略。如果没有，我们可能会被迫以接近它。起初，人们可能会得出这样的结论计算最优值函数是不可能的，由于这一事实，即belief空间是连续的。但是，我们观察到的有限的世界，值函数有一个特殊的形状：它是由有限多个线性件。这使得能够计算在有限时间内的最优值函数。我们还注意到，价值函数是凸和连续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视野。  
我们开始的belief状态B的直接回报我们的考虑。回想一下，b的收益是由下概率分布B的收益C中的给定期望：  
C（二）=？ C（S）B（S）DS（16.7）  
使用使得b唯一地由概率p1指定的事实。 。 。 ，PN，我们可以写C（B）= C（P1，。，PN）=  
ñ  
？  
C（SI）PI  
I = 1  
（16.8），它确实是线性p1中，。 。 。 ，PN。  
图16.2期望收益C（B）为belief参数P3的功能，假设机器人无论是在状态S3或S4。  
有趣的是绘制函数c（b）在置信度分布在两种状态S3和S4在我们的例子中，只有国家与非零回报。在状态S3回报  
是100，而在状态S4的回报是-100。图16.2a示出了函数c（b）为相信通过？0,0，第3页，1-P3限定的子空间？这是一种belief的空间，放置在美国S3和S4所有的可能性。这里的期望C（b）的绘制为  
函数的概率P3的。显然，如果P3 =0时，环境状态是S4，并回报将是C（S4）= -100。另一方面，如果P3 =1，环境的  
状态是S3和回报将是C（S3）= 100。在此期间，期望是线性的，从而导致在图16.2a所示的曲线图。  
这种考虑使我们能够计算价值函数规划时域T = 1。从现在开始，我们将只考虑belief空间的地方的子集  
在两种状态S1和S2所有的可能性。这种belief空间是由一个单一的参数参数化，p1的，因为P2 = 1 - P1和P3 = P4 = 0的值函数C1（二，a1）中是恒定的零动作A1：  
C1（二，A1）= 0（16.9）  
因为无论机器人的真实状态，动作A1不会导致它的状态，使得它收到非零的回报。该值函数C1（B，A1）如果绘制如图16.2b。图片变得更有趣动作A2。如果环境的状态是S1，这个动作将导致90％的几率状态S3，其中机器人将收到的100的收益有10％的概率会结束在状态S4，在那里它会回报  
-100。因此，在状态S1的预期收益为0.9·100 + 0.1·（-100）= 80。类似的说法，状态S2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  
C1（B，A2）=γ（80p1 -80p2）= 72p1 -72p2（16.10）  
这里我们使用的折现因子γ= 0.9。这个函数表示在图16.2c，该类型的belief？P1，1-P1，0，0℃。  
那么，什么是正确的动作选择策略？继最大化预计这将对收益的理由，最好的动作取决于我们目前的belief，假设它准确  
反映了我们对现实世界的知识。如果概率P1≥0.5，最佳动作将A2，因为我们期望正回报。对于值小于0.5，最佳动作将A1，因为它避免了与动作A2相关的负面预期收益。对应的价值功能的动作特定值功能最大：  
C1（B）=最大为C1（B，A）  
=最大值{0; 72p1-72p2}（16.11）  
该值函数和动作选择了相应的策略由图16.2d，从而最大程度地说明了两个线性成分的固体图所示  
由虚线。我们注意到，这个数值函数不是线性的任何更长的时间。相反，它是分段线性和凸出的。非线性产生的事实，不同的操作是最佳的belief空间的不同部分。  
从图16.2d的价值功能，我们可以得出结论，beliefP1 <0.5是没有办法获得的收益大于0？当然，答案是否定的。该  
值函数C1（b）是仅最适合于时域T = 1。对于较大的视野，也能够先执动作作A1，接着动作A2。执动作作A1具有两个有益效果：首先，它有助于我们估计当前状态更好，由于这样的事实，在我们可以感知，并且第二，以高概率从s1至s2的，或反之亦然改变状态。因此，一个良好的策略可能会执行的动​​作A1，直到我们有理由相信，机器人的状态是S1。然后，机器人应执行的动作A2。  
为了使这个比较正规的，让我们得出的最优值函数的时域T = 2，假设我们执行的动作A1。然后，两件事情可能发生：我们要么遵守O1 O2或。让我们先假设我们观察到O1。那么新的belief状态可以使用贝叶斯滤波器来计算：  
2个P？ 1 =η1P（O 1？|？第1条）  
？  
P（S 1 |？A1，SI）PI  
I = 1  
=η10.7（0.1p1 + 0.8p2）=η1（0.07p1 + 0.56p2）  
（16.12）  
其中，η1是贝叶斯法则的正规化。同样，我们得到的是S2后概率：  
2 p? 2 = η1 P(o? 1|s? 2)  
?  
P(s? 2|a1, si) pi  
i=1  
= η1 0.4(0.9p1 +0.2p2) = η1 (0.36p1 +0.08p2)  
(16.13) Since we know that p? 1 + p? 2 = 1—after all, when executing action a1 the state will  
either be s1 or s2—we obtain for η1: η1 =  
1 0.07p1 +0.56p2 +0.36p1 +0.08p2 = 1 0.43p1 +0.64p2 (16.14)  
我们也注意到，变量η1是一个有用的概率：这是执行的动作A1（无论后状态的）后观察O1的概率。  
我们现在有表情刻画因为我们执行AC-后belief  
A1化观察O1。那么，什么是这个国家的belief价值？答案是转播tained通过插入新的belief到价值函数C1（b）的公式定义（16.11），和γ贴现的结果：  
C2（B，A1，01）=γ最大值{0; 72P？ 1 -72p？ 2}  
= 0.9最大值{0; 72η1（0.36p1 + 0.08p2）-72η1（0.43p1 + 0.64p2）} = 0.9最大值{0; 72η1（0.36p1 + 0.08p2 -0.43p1 -0.64p2）} = 0.9最大值{0; 72η1（-0.07p1 -0.56p2）} = 0.9最大值{0; η1（-5.04p1 -40.32p2）}  
（16.15）  
我们现在移动的因素η1出来的最大化和移动0.9里面，并取得：C2（B，A1，01）=η1最大值{0; -4.563p1 -36.288p2}  
（16.16），其是一个分段线性的，凸的函数的beliefb的参数。  
推导的执行动作A1后观察O2是完全类似。特别是，我们得到的后验  
p?  
1 = η2 0.3(0.1p1 +0.8p2) = η2 (0.03p1 +0.24p2)  
p?  
2 = η2 0.6(0.9p1 +0.2p2) = η2 (0.54p1 +0.12p2)  
with the normalizer η2 =  
1 0.57p1 +0.36p2  
The corresponding value is C2(b, a1, o1) = γ max{0 ; 72p?  
1 −72p? 2} = η2 max{0 ; −33.048p1 +7.776p2} (16.17) (16.18) (16.19)  
顺便说一句，我们也注意到，η-1 1 +η-1 2 = 1。这直接遵循这样的事实  
在贝叶斯正规化过滤是观察概率ηI=  
1 P（O I |？A1，B）（16.20）和事实恰好有两种可能的观测我们的例子中，O1和O2。  
现在让我们计算值C2（二，a1）中，这是在执动作作A1的预期值。显然，该值是值C2（二，A1，01）和C2（二，A1，O2），通过实际观察O1和O2分别的概率加权的混合物。放入数学符号，我们有  
C2（二，A1）= 2  
？  
C2（B，A1，OI）P（OI | A1，B）  
I = 1  
（16.21）  
术语C 2（二，A1，OI）已经如上所定义。概率P（OI | A1，B）执动作作A1之后观察OI是η-1  
我。因此，我们有所有成分  
计算所需值C2（二，a1）中：C2（二，A1）=η-1  
1η1最大值{0; -4.563p1 -36.288p2} +η-1  
2η2最大值{0; -33.048p1 + 7.776p2}（16.22）= MAX {0; -4.563p1 -36.288p2} +最大{0; -33.048p1 + 7.776p2}  
该表达式可以被重新表示为四个线性函数的最大值：C2（二，A1）=最大{0; -4.563p1-36.288p2;  
（16.23）-33.048p1+7.776p2; -37.611p1-28.512p2}  
图16.2e表示这四个线性函数的belief间隔p1，1 - P 2，0，0℃。该值函数C2（B，A1）是这四个线性函数的最大值。由于是  
容易被看到的那样，二者的这些功能是足以定义最大;其他  
2顷在整个频谱较小。这使我们能够改写的C2（二，a1）中为最大值的仅有的两个方面，而不是四个：C2（二，A1）=最大{0; -33.048p1+7.776p2}  
（16.24）  
最后，我们要确定的值C2（b）中，这是下面两个条件中的最大值：  
C2（B）=最大{C2（二，a1）中，C 2（二，A2）}（16.25）  
正如很容易地验证，则第二项的最大化，C2（二，a2）中，是完全与上述相同的用于规划时域T = 1：  
C2（二，A2）= C1（二，A2）= 72p1 -72p2  
因此，我们获得（16.24）和（16.26）：  
（16.26）  
C2（B）=最大{0; -33.048p1 + 7.776p2; 72p1 -72p2}  
（16.27）这个函数是最优值函数规划时域T.图？图  
C2（b）和其组成部分的belief子空间P1，1 - P 2，0，0℃。正如很容易看到的，函数是分段线性和凸出的。特别是，它包括三个  
线性件。对于在最左边的两个片belief（​​即P1 <0.5），A1是最佳的动作。对于P1 = 0.5，这两个动作都是一样的好。对应于最右边的线性片的belief，即用P1> 0.5的belief，有最佳的动作A2。如果a1为最优函数，下一个动作取决于在belief空间的初始点和在观察。  
价值函数C 2（b）是仅最适合于时域2。但是，我们的分析示出了几个重要的观点。  
首先，最优值函数的任何有限的视距是连续的，分段线性，凸。每个线性一块对应于一个不同的操作选择在将来某个时候，或者，可以进行不同的观察。价值函数的凸性表示相当直观的观察，即明知是AL-方面都优于不知道。给定两个belief状态B和B?,的belief状态的混合值大于或等于该混合belief状态的值，对于某些混合参数β0≤β≤1：  
βC（二）+（1-β）C（二？）≥-C（βB+（1-β）B’）（16.28）  
二，直线条的数量可以极大地增长，特别是如果一个人不注重线性函数变得过时。在我们的玩具例子中，两个出的四个直线约束限定的C2（二，a1）的是没有必要。 EF-ficiently实施POMDPs的“诀窍”在于早识别过时线性函数  
如可能，从而使计算的价值函数时，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num- BER仍然可以增长得非常迅速。这对对有限POMDPs的精确值迭代解决方案固有的扩展限制。

16.2.2belief空间中的价值迭代  
在上一节展示来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，在介绍这一章中，我们指出了价值函数，我们简要地重申这里的基本更新方程：  
（16.29）  
在这个和下面的章节中，我们将开发一个可在数字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（16.2）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维数n-1的连续体。我们注意到，为了指定在n个离散事件的概率分布n-1个数值是必需的（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的问题，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 |？的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Ò，执行动作时，后验B’是独特的，P（B|？的a，b）是一个point-mass分布的退化。从动作执行之前的beliefb，动作a，和随后的观测O 3，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 3，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

（16.30）

其中，P（B |？的a，b，邻）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（16.2），我们得到  
（16.31）  
内积分  
（16.32）  
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，邻？）来表示这个分布，那么有  
（16.33）  
我们注意到，正规化分母，P（O|？A，B），是值更新方程（16.31）的一个因素。因此，将B（B，A，O 2）代入（16.31）消除了这一量，从而导致了值迭代算法的递归描述：  
（16.34）  
这种形式是比在（16.2）更加方便，因为它只需要对所有可能的测量O 2，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：  
（16.35）  
（16.36）  
在这里，CT（B，A）是时域T对于假设下一个动作a的belief b的函数。

16.2.3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CT由线性函数的集合表示。如果CT是线性的，它可以通过一组系数。。。CT来表示

（16.37）  
其中，像往常一样，P1。 。 。 ，PN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CT（B）可以通过K个线性函数集合的最大值表示

（16.38）  
其中，CT K，1，。 。 。 ，CT K，n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  
C0 = 0（16.39）  
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CT（b）的递归方程。该公式假定上一步的值函数CT-1（b），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CT-1（b）是分段线性和凸的，CT（b）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（16.35）和（16.36）定义如下：  
（16.40）  
（16.41）  
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：

（16.43）  
belief B（？B，A，O）使用下面的表达式获得，该式为用和来替换（16.33）中的积分所得到：

B（？B，A，O）（S？）= P（O|？A，B）P（O|？S）？P（S|？A，S）B（S）1秒（16.44）

如果belief B由参数{p1，。。。 ，PN}表示，那么belief b’中的第i个参数由下式计算得到

（16.45）

为了计算价值函数CT（B，A），我们将现在的条件c（B（B，A，O 2））和CT-1（B（B，A，O 2） 中衍生出更多实用的表达式中），从公式（16.42）开始，具有参数B（B，A，O 2）={对？  
1，。 。 。页？ N}，我们从。。得到了

（16.46）  
代入（16.45），我们得到  
（16.47）  
由于P（邻？|的a，b）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：P（O| A，B？）。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（16.42）中对量CT-1（B（B，A，O 2））的推导类似于上面C（B（B，A，O 2））。特别是，我们不得不用唯一的分段线性和凸的值函数CT-1取代直接回报函数c。然后可以得到

（16.48）  
。如上所述，我们代入后验概率belief B（见（16.45）），并获得：

（16.49）  
其中P（ω0| A，B）-1不依赖于i或K，故可移出求和和最大化部分，得到：  
（16.50）  
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：

（16.51）

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PJ。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CT（B，A）和时域T的值函数。我们简要地重申方程（16.42）：  
16.52）

让我们首先用公式（16.47）和（16.50）计算总和C（B（B，A，O）？）+ CT-1（B（B，A，O）？），：

(16.53)   
经过一些整理得到，  
(16.54)

将这个表达式代入公式16.52  
(16.55)  
我们注意到，项P（邻|的a，b？）同时出现在在此表达式的分子和分母中从而抵消了，得到：  
（16.56）

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最大化CT（B，A）：  
CT（B）得到

（16.57）  
很容易验证，在belief参数为P1，。 。 。 ，PN的情况CT（b）是确实分段线性和凸的。具体地，对于每个固定k（16.56）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Ø？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（16.57）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，CT（b）是分段线性和凸出的。

**仿真实验结果**

# 2问题公式化

一个POMDP [7,8]是由它的状态空间X,动作空间U，观测空间Y，状态转换律（X，X’∈X，u∈U），观测律（Z∈Y），初始状态分布P⁰，和一步成本函数r（X，u）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X⁰服从已知的分配P⁰，一个POMDP演变如下。在时间步骤k中，该系统的状态是值和观测是已知的。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条件的。这里的目标是要基于 belief state ，从一组可用动作U（）选择一个动作使得预期总成本最小化。策略定义为从belief state 到动作的映射的序列。

让预计总成本从初始belief state 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u的花费为.则在belief state 采取动作u的Q值是

其中是开始于下一个belief state 经过 H -k-1的时间步长的最佳值。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

（2）

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。我们规则化POMDP如下。

## 2.1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 = 0。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

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i在x和y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。

## 2.2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

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。

## 2.3 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）β的泊松分布。在空间上，误报在整个监测体积上均匀分布。

## 3.1粒子滤波用于belief state 估计

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u。预测步骤由下式给出[18,19]

（6）

更​​新步骤由下式给出

（7）

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（也称为标准粒子滤波）。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。

在我们以前的工作[11-13]中，我们在单传感器单目标跟踪中使用了SIR粒子滤波算法。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：

（8）

在JPDA中，概率，使用下式计算得到

（9）

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。（见第6章中[21]的细节和例子。）对于2.3节中的传感器模型，从目标在传感器上得到观测的可能性由下式给出

，

（10）

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（8），（9）和（10）计算权重

（b）计算权重。

规一化：对于， 。

（ⅳ）重采样。基于页从选择的N个粒子。

（ⅴ）设，并去预测步骤。

随着粒子的数量变得非常大，则粒子滤波器接近的在（6）和（7）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（10）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，跟踪器是更稳定，并具有多更小的跟踪误差。

# 4仿真实验

我们的实验在一个40×40公里的监视区内有两个运动目标和四个传感器（雷达）。这两个运动目标由具有不确定加速度的NCV模型决定。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，，，。

采样间隔为，并且我们模拟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，并使用rollout Q值近似方法中的8个步骤。

在单传感器单目标跟踪问题中，CPA策略是常用的[11,12]。由于测量方差正比于，选择最接近目标的传感器是很自然的。但是CPA是一种“贪婪”的方法，因为它没有考虑到不同的传感器的成本或它们的差错统计。

让我们定义的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，CPA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CPA策略的最接近传感器策略（CSP），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，CSP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CO-rollout之间公平比较，通过CO-rollout策略选择激活的传感器的最大数量也是两个。需要注意的是，只要底层目标状态来计算，根据CO-rollout近似算法的要求，在CSP基策略具有映射目标状态到行动的属性。

我们比较CSP策略（12）（最多激活两个传感器）及使用CSP作为底层基础策略的CO-rollout策略（11）的性​​能。图1分别给出了采用CSP和CO-rollout算法的一个例子目标的真正的轨迹（由实线示出），目标位置估计（用标记示出），和所选择的传感器（用不同的形状标记）。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，并且目标交叉处理的很正确。接下来，我们注意到这两项策略在传感器选择的不同之处。首先，因为传感器0的位置是最接近至少其中一个目标的，CSP在所有的时间均激活传感器0，然而考虑到它教高的使用成本，CO-rollout仅在一个短的时间激活传感器0。第二，CSP在目标交叉后当传感器3成为最接近目标0的传感器激活传感器3，而CO-rollout由于其观测的高精确度在目标交叉之前激活传感器3。有时CSP仅选择一个传感器，当它是最接近两个目标的传感器时，但CO-rollout可能会发现激活两个传感器更有益。然而在其他时间，CSP激活2个传感器，而CO-rollout决定一个就够了。

图2展示了四个策略的定量比较：有1或2个传感器激活的CSP，和有1或2个传感器激活的CO-rollout。跟踪误差，传感器的使用，以及总成本是超过100次随机模拟的平均值和90％置信区间。这里的总成本在（4）中定义且。注意到通过激活最多两个传感器，该总成本在CSP中减少了15％以及在CO-rollout中降低30％。我们还可以得出结论，我们的CO-rollout策略单传感器激活提高20％、双传感器激活提高40％。

显然这是一个在跟踪误差和传感器的使用之间的折衷。我们在我们的算法中研究这种权衡并在图3中展示出来。通过调整（4）中的值，优先级可以被放置在跟踪误差或传感器的成本上。这里，结合的优先级得到，跟踪优先级得到，和使用优先级得到。正如我们在图3中看到的，随着减小，跟踪误差增加，而传感器使用减少