Solutions for Chapter 16
Making Simple Decisions

16.1 It is interesting to create a histogram of accuracy on this task for the students in the class. It is also interesting to record how many times each student comes within, say, 10% of the right answer. Then you get a profile of each student: this one is an accurate guesser but overly cautious about bounds, etc.

16.2

Pat is more likely to have a better car than Chris because she has more information with which to choose. She is more likely to be disappointed, however, if she takes the expected utility of the best car at face value. Using the results of exercise 16.11, we can compute the expected disappointment to be about 1.54 times the standard deviation by numerical integration.

16.3

a. The probability that the first heads appears on the nth toss is 2^{-n}, so

$$EMV(L) = \sum_{n=1}^{\infty} 2^{-n} \cdot 2^n = \sum_{n=1}^{\infty} 1 = \infty$$

b. Typical answers range between \$4 and \$100.

c. Assume initial wealth (after paying c to play the game) of $\$(k-c)$; then

$$U(L) = \sum_{n=1}^{\infty} 2^{-n} \cdot (a \log_2(k - c + 2^n) + b)$$

Assume $k - c = \$0$ for simplicity. Then

$$\begin{aligned} U(L) &= \sum_{n=1}^{\infty} 2^{-n} \cdot (a \log_2(2^n) + b) \\ &= \sum_{n=1}^{\infty} 2^{-n} \cdot an + b \\ &= 2a + b \end{aligned}$$

d. The maximum amount c is given by the solution of

$$a \log_2 k + b = \sum_{n=1}^{\infty} 2^{-n} \cdot (a \log_2(k - c + 2^n) + b)$$

第16章的解决方案
做出简单的决定

16.1为班上的学生创建一个关于这项任务的准确性直方图是很有趣的。 记录每个学生进入正确答案的10%的次数也很有趣。 然后你得到每个学生的个人资料：这个是一个准确的猜测，但对边界过于谨慎，等等。

16.2

帕特更有可能拥有比克里斯更好的汽车，因为她有更多的信息可供选择。 然而，如果她以面值获得最好的汽车的预期效用，她更有可能感到失望。 使用练习16.11的结果，我们可以通过数值积分计算预期的失望值约为标准偏差的1.54倍。

16.3

a. 第一个头出现在第n次折腾上的概率是2-n，所以

$$EMV(L) = \sum_{n=1}^{\infty} 2^{-n} \cdot 2^n = \sum_{n=1}^{\infty} 1 = \infty$$

b. 典型的答案介于$4和$100之间。 c. 假设初始财富（在支付c玩游戏之后）为$（k-c）;然后

$$U(L) = \sum_{n=1} 2^{-n} \cdot (a \log_2(k - c + 2^n) + b)$$

为了简单起见，假设k-c=$0。 然后

$$\begin{aligned} U(L) &= \sum_{n=1} 2^{-n} \cdot (a \log_2(2^n) + b) \\ &= \sum_{n=1}^{\infty} 2^{-n} \cdot an + b \\ &= 2a + b \end{aligned}$$

d. 最大量c由解给出

$$a \log_2 k + b = \sum_{n=1}^{\infty} 2^{-n} \cdot (a \log_2(k - c + 2^n) + b)$$

For our simple case, we have

$$a \log_2 c + b = 2a + b$$

or $c = \$4$.

16.4 The program itself is pretty trivial. But note that there are some studies showing you get better answers if you ask subjects to move a slider to indicate a proportion, rather than asking for a probability number. So having a graphical user interface is an advantage. The main point of the exercise is to examine the data, expose inconsistent behavior on the part of the subjects, and see how people vary in their choices.

16.5

a. Networks (ii) and (iii) can represent this network but not (i).

(ii) is fully connected, so it can represent any joint distribution.

(iii) follows the generative story given in the problem: the flavor is determined (presumably) by which machine the candy is made by, then the shape is randomly cut, and the wrapper randomly chosen, the latter choice independently of the former.

(i) cannot represent this, as this network implies that the wrapper color and shape are marginally independent, which is not so: a round candy is likely to be strawberry, which is in turn likely to be wrapped in red, whilst conversely a square candy is likely to be anchovy which is likely to be wrapped in brown.

b. Unlike (ii), (iii) has no cycles which we have seen simplifies inference. Its edges also follow the, so probabilities will be easier to elicit. Indeed, the problem statement has already given them.

c. Yes, because Wrapper and Shape are d-separated.

d. Once we know the Flavor we know the probability its wrapper will be red or brown. So we marginalize Flavor out:

$$\begin{aligned}
\mathbf{P}(Wrapper = red) &= \sum_f \mathbf{P}(Wrapper = red, Flavor = f) \\
&= \sum_f \mathbf{P}(Flavor = f)\mathbf{P}(Wrapper = red | Flavor = f) \\
&= 0.7 \times 0.8 + 0.3 \times 0.1 \\
&= 0.59
\end{aligned}$$

e. We apply Bayes theorem, by first computing the joint probabilities

$$\begin{aligned}
&\mathbf{P}(Flavor = strawberry, Shape = round, Wrapper = red) \\
&= \mathbf{P}(Flavor = strawberry) \times \mathbf{P}(Shape = round | Flavor = strawberry) \\
&\quad \times \mathbf{P}(Wrapper = red | Flavor = strawberry) \\
&= 0.7 \times 0.8 \times 0.8 \\
&= 0.448 \\
&\mathbf{P}(Flavor = anchovy, Shape = round, Wrapper = red) \\
&= \mathbf{P}(Flavor = anchovy) \times \mathbf{P}(Shape = round | Flavor = anchovy)
\end{aligned}$$

对于我们的简单案例，我们有

$$a \log_2 c + b = 2a + b$$

or $c = \$4$.

16.4程序本身是相当微不足道的。 但请注意，有一些研究表明，如果您要求受试者移动滑块以指示比例，而不是要求概率数字，则会获得更好的答案。 所以具有图形用户界面是一个优点。 练习的要点是检查数据，暴露受试者的不一致行为，并查看人们在选择方面的差异。

16.5

A.网络(ii)和(iii)可以代表此网络，但不能代表(i)。

（ii）是完全连接的，因此它可以表示任何联合分布。 （iii）遵循问题中给出的生成故事：风味是由哪台机器（预先总结）制造糖果的，然后形状是随机切割的，包装是随机选择的，后者是独立于前者的选择。

(i)不能代表这一点，因为这个网络意味着包装的颜色和形状是略微独立的，事实并非如此：圆形糖果很可能是草莓，而草莓又可能被红色包裹，而相反，方形糖果很可能是凤尾鱼，而凤尾鱼很可能被棕色包裹。 b.与（ii）不同，（iii）没有我们已经看到简化推理的循环。 它的边缘也跟着，所以概率会更容易引出。 事实上，问题陈述已经给了他们。 C.是的，因为包装和形状是d分隔的。 一旦我们知道味道，我们就知道它的包装是红色或棕色的可能性。 所以我们把味道边缘化了：

$$\begin{aligned}
\text{P（包装=红色）} &= \sum_f \text{P（包装=红色，风味=f）} \\
&= \sum_f \text{P（风味=f）P（包装=红色/风味=f）} \\
&= 0.7 \times 0.8 + 0.3 \times 0.1 \\
&= 0.59
\end{aligned}$$

我们应用贝叶斯定理，首先计算联合概率

P（味道=草莓，形状=圆形，包装=红色）

=P（风味=草莓）×P（形状=圆形/风味=草莓）

×P(包装=红色/风味=草莓)

=

$= 0.448$

P（风味=凤尾鱼，形状=圆形，包装=红色）

=P（风味=凤尾鱼）×P（形状=圆形/风味=凤尾鱼）

$$\times \mathbf{P}(Wrapper = red|Flavor = anchovy)$$
$$= 0.3 \times 0.1 \times 0.1$$
$$= 0.003$$

Normalizing these probabilities yields that it is strawberry with probability $0.448/(0.448+0.003) \approx 0.9933$.

f. Its value is the probability that you have a strawberry upon unwrapping times the value of a strawberry, plus the probability that you have a anchovy upon unwrapping times the value of an anchovy or

$$0.7s + 0.3a \ .$$

g. The value is the same, by the axiom of decomposability.

16.6

First observe that $C \sim [0.25, A; 0.75, \$0]$ and $D \sim [0.25, B; 0.75\$0]$. This follows from the axiom of decomposability. But by substitutability this means that the preference ordering between the lotteries A and B must be the same as that between C and D.

16.7

As mentioned in the text, agents whose preferences violate expected utility theory demonstrate irrational behavior, that is they can be made either to accept a bet that is a guaranteed loss for them (the case of violating transitivity is given in the text), or reject a bet that is a guaranteed win for them. This indicates a problem for the agent.

16.8 The expected monetary value of the lottery L is

$$EMV(L) = \frac{1}{50} \times \$10 + \frac{1}{2000000} \times \$1000000 = \$0.70$$

Although $\$0.70 < \$1$, it is not *necessarily* irrational to buy the ticket. First we will consider just the utilities of the monetary outcomes, ignoring the utility of actually playing the lottery game. Using $U(S_{k+n})$ to represent the utility to the agent of having n dollars more than the current state, and assuming that utility is linear for small values of money (i.e., $U(S_{k+n}) \approx n(U(S_{k+1}) - U(S_k))$ for $-10 \leq n \leq 10$), the utility of the lottery is:

$$\begin{aligned} U(L) &= \frac{1}{50}U(S_{k+10}) + \frac{1}{2,000,000}U(S_{k+1,000,000}) \\ &\approx \frac{1}{5}U(S_{k+1}) + \frac{1}{2,000,000}U(S_{k+1,000,000}) \end{aligned}$$

This is more than $U(S_{k+1})$ when $U(S_{k+1,000,000}) > 1,600,000U(\$1)$. Thus, for a purchase to be rational (when only money is considered), the agent must be quite risk-seeking. This would be unusual for low-income individuals, for whom the price of a ticket is non-trivial. It is possible that some buyers do not internalize the magnitude of the very low probability of winning—to imagine an event is to assign it a "non-trivial" probability, in effect. Apparently, these buyers are better at internalizing the large magnitude of the prize. Such buyers are clearly acting irrationally.

×P(包装=红色/风味=凤尾鱼)

=

$$= 0.003$$

对这些概率进行归一化会得到概率为0的结果。 448 / (0 . 448+ 0 . 003) ≈ 0 . 9933 . F.它的值是你有一个草莓在解开倍草莓的价值的概率，加上你有一个凤尾鱼在解开倍凤尾鱼的价值或

$$0.7s + 0.3a \ .$$

g.值是相同的，由可分解性公理。

16.6

首先观察C [0。 25，A；0． 75， 0]和D [0. 25，B；0． 75$0] . 这源于可分解性的公理。 但通过可替换性，这意味着彩票A和B之间的优先顺序必须与c和D之间的优先顺序相同。

16.7

如文中所述，其偏好违反预期效用理论的代理人表现出非理性行为，即他们既可以接受对他们来说是一个有保证的损失的赌注（违反传递性的情况在文中给出），也可以拒绝对他们来说是一个有保证的胜利的赌注。 这表明代理存在问题。

16.8 彩票L的预期货币价值为

$$EMV(L) = \frac{1}{50} \times \$10 + \frac{1}{2000000} \times \$1000000 = \$0.70$$

虽然$0。 70<$1，买票不一定不合理。 首先，我们将只考虑货币结果的效用，而忽略实际玩彩票游戏的效用。 使用U(S k+n)来表示效用给比当前状态多n美元的代理，并且假设效用对于金钱的小值是线性的（即，U(S k+n)÷n(U(S k+1)-U(S k))对于-10≤n ≤10)，彩票的效用是:

$$\begin{aligned} U(L) &= \frac{1}{50}U(S_{k+10}) + \frac{1}{2,000,000}U(S_{k+1,000,000}) \\ &\approx \frac{1}{5}U(S_{k+1}) + \frac{1}{2,000,000}U(S_{k+1,000,000}) \end{aligned}$$

这比U(S k+1)当U(S k+1 , 000 , 000) > 1 , 600 , 000 U($1)。 因此，购买是理性的（当只考虑金钱时），代理人必须是相当冒险的。 这对于低收入个人来说是不寻常的，对他们来说，机票的价格是非常重要的。 这是可能的，一些买家不内化的非常低的获胜概率的大小-想象一个事件是赋予它一个"不平凡"的概率，实际上。 显然，这些买家更善于将奖金的巨大幅度内化。 这样的买家显然是非理性的行为。

Some people may feel their current situation is intolerable, that is, $U(S_k) \approx U(S_{k\pm1}) \approx u_\perp$. Therefore the situation of having one dollar more or less would be equally intolerable, and it would be rational to gamble on a high payoff, even if one that has low probability.

Gamblers also derive pleasure from the excitement of the lottery and the temporary possession of at least a non-zero chance of wealth. So we should add to the utility of playing the lottery the term t to represent the thrill of participation. Seen this way, the lottery is just another form of entertainment, and buying a lottery ticket is no more irrational than buying a movie ticket. Either way, you pay your money, you get a small thrill t, and (most likely) you walk away empty-handed. (Note that it could be argued that doing this kind of decision-theoretic computation decreases the value of t. It is not clear if this is a good thing or a bad thing.)

16.9 This is an interesting exercise to do in class. Choose $M_1 = \$100$, M_2 = \$100, \$1000, \$10000, \$1000000. Ask for a show of hands of those preferring the lottery at different values of p. Students will almost always display risk aversion, but there may be a wide spread in its onset. A curve can be plotted for the class by finding the smallest p yielding a majority vote for the lottery.

16.10 The protocol would be to ask a series of questions of the form "which would you prefer" involving a monetary gain (or loss) versus an increase (or decrease) in a risk of death. For example, "would you pay \$100 for a helmet that would eliminate completely the one-in-a-million chance of death from a bicycle accident."

16.11

First observe that the cumulative distribution function for $\max\{X_1, \ldots, X_k\}$ is $(F(x))^k$ since

$$\begin{aligned} P(\max\{X_1, \ldots, X_k\} \leq x) &= P(X_1 \leq x, \ldots, X_k \leq x) \\ &= P(X_1 \leq x) \ldots P(X_k \leq x) \\ &= F(x)^k \end{aligned}$$

the second to last step by independence. The result follows as the probability density function is the derivative of the cumulative distribution function.

16.12

a. This question originally had a misprint: $U(x) = -e^{x/R}$ instead of $U(x) = -e^{-x/R}$. With the former utility function, the agent would be rather unhappy receiving \$1000000 dollars.

Getting \$400 for sure has expected utility

$$-e^{-400/400} = -1/e \approx -0.3679$$

while the getting \$5000 with probability 0.6 and \$0 otherwise has expected utility

$$0.6 - e^{-5000/400} + 0.5 - e^{-0/400} = -(0.6e^{-12.5} + 0.5) \approx -0.5000$$

so one would prefer the sure bet.

b. We want to find R such that

$$e^{-100/R} = 0.5e^{-500/R} + 0.5$$

有些人可能会觉得自己目前的状况是无法容忍的，即U(S k） U(S k±1） u 。因此，多或少一美元的情况同样是不可容忍的，即使是概率较低的回报，也要赌高回报是合理的。

赌徒也从彩票的兴奋和暂时拥有至少非零的财富机会中获得乐趣。因此，我们应该添加到玩彩票的效用术语t来表示参与的快感。这样看，彩票只是另一种娱乐形式，买彩票并不比买电影票更不合理。无论哪种方式，你付出你的钱，你得到一个小小的刺激t，并且（最有可能）你空手而归。（请注意，可以说，做这种决策理论计算会降低t的值。目前尚不清楚这是好事还是坏事。）

16.9这是一个有趣的练习在课堂上做。选择M1=\$100,M2 = \$100, \$1000, \$10000, \$1000000. 要求那些喜欢以不同的p值抽签的人举手. 学生几乎总是会表现出风险厌恶情绪，但其发病可能会广泛传播。一条曲线可以通过找到最小的p产生对彩票的多数票来为类绘制。

16.10协议将提出一系列形式为"您希望哪种"的问题，涉及货币收益（或损失）与死亡风险的增加（或减少）。例如，"你会支付helmet100的头盔，将完全消除一个百万分之一的机会，从自行车事故死亡。"

16.11

首先观察累积分布函数为max{X1,。..,X k}为(F(x))k since

$$\begin{aligned} P(\max\{X_1, \ldots, X_k\} \leq x) &= P(X_1 \leq x, \ldots, X_k \leq x) \\ &= P(X_1 \leq x) \ldots P(X_k \leq x) \\ &= F(x)^k \end{aligned}$$

独立的第二个到最后一个步骤。结果如下，概率密度函数是累积分布函数的导数。

16.12

a.这个问题本来有一个错印：U(x)=-e x/R而不是U(x)=-e-x/R。与前效用函数，代理将是相当不高兴收到\$1000000美元。

获得sure400肯定有预期的效用

$$-e^{-400/400} = -1/e \approx -0.3679$$

而以0.6和0的概率获得5000美元的预期效用

$$0.6 - e^{-5000/400} + 0.5 - e^{-0/400} = -(0.6e^{-12.5} + 0.5) \approx -0.5000$$

所以人们更喜欢肯定的赌注。 b .我们想找到R这样

$$e^{-100/R} = 0.5e^{-500/R} + 0.5$$

Solving this numerically, we find $R = 152$ up to 3sf.

16.13 The information associated with the utility node in Figure 16.6 is an action-value table, and can be constructed simply by averaging out the $Deaths$, $Noise$, and $Cost$ nodes in Figure 16.5. As explained in the text , modifications to aircraft noise levels or to the importance of noise do not result in simple changes to the action-value table. Probably the easiest way to do it is to go back to the original table in Figure 16.5. The exercise therefore illustrates the tradeoffs involved in using compiled representations.

16.14 The answer to this exercise depends on the probability values chosen by the student.

16.15

a. See Figure S16.1.

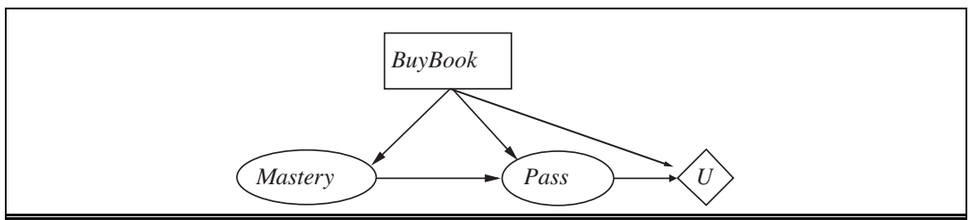


Figure S16.1 A decision network for the book-buying problem.

b. For each of $B = b$ and $B = \neg b$, we compute $P(p|B)$ and thus $P(\neg p|B)$ by marginalizing out M, then use this to compute the expected utility.

$$\begin{aligned} P(p|b) &= \sum_m P(p|b,m)P(m|b) \\ &= 0.9 \times 0.9 + 0.5 \times 0.1 \\ &= 0.86 \\ P(p|\neg b) &= \sum_m P(p|\neg b,m)P(m|\neg b) \\ &= 0.8 \times 0.7 + 0.3 \times 0.3 \\ &= 0.65 \end{aligned}$$

The expected utilities are thus:

$$\begin{aligned} EU[b] &= \sum_p P(p|b)U(p,b) \\ &= 0.86(2000 - 100) + 0.14(-100) \\ &= 1620 \\ EU[\neg b] &= \sum_p P(p|\neg b)U(p,\neg b) \\ &= 0.65 \times 2000 + 0.14 \times 0 \\ &= 1300 \end{aligned}$$

通过数值求解，我们发现R=152到3sf。

16.13与图16.6中的效用节点相关联的信息是动作-值表，并且可以简单地通过平均出图16.5中的死亡、噪声和成本节点来构造。 正如文中所解释的，对飞机噪音水平或噪音重要性的修改并不会导致对行动价值表的简单更改。 可能最简单的方法是回到图16.5中的原始表。 因此，该练习说明了使用编译表示所涉及的权衡。

16.14 这个练习的答案取决于stu选择的概率值
dent.

16.15

A.见图S16.1。

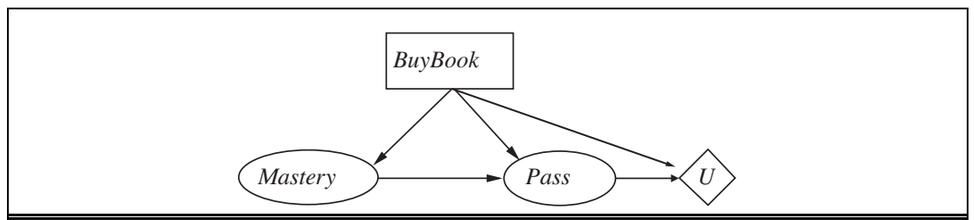


Figure S16.1 买书问题的决策网络。

b.对于B=b和B=b中的每一个，我们通过边际化M来计算P（p|B），从而计算P（p|B），然后用它来计算预期的效用。

$$\begin{aligned} P(p|b) &= \sum_m P(p|b,m)P(m|b) \\ &= 0.9 \times 0.9 + 0.5 \times 0.1 \\ &= 0.86 \\ P(p|\neg b) &= \sum_m P(p|\neg b,m)P(m|\neg b) \\ &= 0.8 \times 0.7 + 0.3 \times 0.3 \\ &= 0.65 \end{aligned}$$

因此，预期的公用事业是:

$$\begin{aligned} EU[b] &= \sum_p P(p|b)U(p,b) \\ &= 0.86(2000 - 100) + 0.14(-100) \\ &= 1620 \\ EU[\neg b] &= \sum_p P(p|\neg b)U(p,\neg b) \\ &= 0.65 \times 2000 + 0.14 \times 0 \\ &= 1300 \end{aligned}$$

c. Buy the book, Sam.

16.16 This exercise can be solved using an influence diagram package such as IDEAL. The specific values are not especially important. Notice how the tedium of encoding all the entries in the utility table cries out for a system that allows the additive, multiplicative, and other forms sanctioned by MAUT.

One of the key aspects of the fully explicit representation in Figure 16.5 is its amenability to change. By doing this exercise as well as Exercise 16.9, students will augment their appreciation of the flexibility afforded by declarative representations, which can otherwise seem tedious.

a. For this part, one could use symbolic values (high, medium, low) for all the variables and not worry too much about the exact probability values, or one could use actual numerical ranges and try to assess the probabilities based on some knowledge of the domain. Even with three-valued variables, the cost CPT has 54 entries.

b. This part almost certainly should be done using a software package.

c. If each aircraft generates half as much noise, we need to adjust the entries in the $Noise$ CPT.

d. If the noise attribute becomes three times more important, the utility table entries must all be altered. If an appropriate (e.g., additive) representation is available, then one would only need to adjust the appropriate constants to reflect the change.

e. This part should be done using a software package. Some packages may offer VPI calculation already. Alternatively, one can invoke the decision-making package repeatedly to do all the what-if calculations of best actions and their utilities, as required in the VPI formula. Finally, one can write general-purpose VPI code as an add-on to a decision-making package.

16.17 This question is a simple exercise in sequential decision making, and helps in making the transition to Chapter 17. It also emphasizes the point that the value of information is computed by examining the *conditional* plan formed by determining the best action for each possible outcome of the test. It may be useful to introduce "decision trees" (as the term is used in the decision analysis literature) to organize the information in this question. (See Pearl (1988), Chapter 6.) Each part of the question analyzes some aspect of the tree. Incidentally, the question assumes that utility and monetary value coincide, and ignores the transaction costs involved in buying and selling.

a. The decision network is shown in Figure S16.2.

b. The expected net gain in dollars is

$$P(q^+)(2000-1500)+P(q^-)(2000-2200)=0.7\times 500+0.3\times -200=290$$

c. The question could probably have been stated better: Bayes' theorem is used to compute $P(q^+|Pass)$, etc., whereas conditionalization is sufficient to compute $P(Pass)$.

$$\begin{aligned}P(Pass) &= P(Pass|q^+)P(q^+)+P(Pass|q^-)P(q^-)\\ &= 0.8\times 0.7+0.35\times 0.3=0.665\end{aligned}$$

c.买这本书，山姆。

16.16本练习可以使用i DEAL等影响图包解决。 的具体数值并不是特别重要。 请注意，对实用程序表中的所有条目进行编码的枯燥乏味对于允许maut认可的加法，乘法和其他形式的系统来说是如此。

图16.5中完全明确表示的关键方面之一是其易于改变。 通过做这个练习以及练习16.9，学生将增强他们对陈述性陈述所提供的灵活性的欣赏，否则这可能看起来很乏味。

a. 对于这一部分，可以对所有变量使用符号值（高，中，低），而不必太担心确切的概率值，或者可以使用实际的数值范围并尝试根据域的一些知识来评估概率。 即使使用三值变量，成本CPT也有54个条目。 b. 这部分几乎肯定应该使用软件包来完成。 c. 如果每架飞机产生一半的噪音，我们需要调整噪音CPT中的条目。 d. 如果noise属性变得重要三倍，则必须全部更改实用程序表条目。 如果一个适当的（例如，加性）表示是可用的，那么人们只需要调整适当的常数来反映变化。 e. 这部分应该使用软件包来完成。 有些软件包可能已经提供了VPI计算。 或者，可以根据VPI公式的要求，重复调用决策包来执行最佳操作及其实用程序的所有假设计算。 最后，可以将通用VPI代码作为决策包的附加组件编写。

16.17这个问题是顺序决策的一个简单练习，有助于过渡到第17章。 它还强调了信息的价值是通过检查通过确定测试的每个可能结果的最佳操作而形成的条件计划来计算的。 引入"决策树"（正如决策分析文献中使用的术语）来组织这个问题中的信息可能是有用的。 （见Pearl（1988），第6章。 问题的每个部分都分析了树的某些方面。 顺便说一句，这个问题假设效用和货币价值是一致的，而忽略了买卖所涉及的交易成本。

a. 决策网络如图S16.2所示。 b. 以美元计算的
预期净收益是

$$P(q^+)(2000-1500)+P(q^-)(2000-2200)=0.7\times 500+0.3\times -200=290$$

c. 这个问题可能已经陈述得更好：贝叶斯定理用于计算P（q+/Pass）等。，而条件化足以计算P（Pass）。

$$\begin{aligned}P(Pass) &= P(Pass|q\ \ )P(q\ \ )+P(Pass|q\ \ )P(q\ \ )\\ &= 0.8\times 0.7+0.35\times 0.3=0.665\end{aligned}$$

Using Bayes' theorem:

$$\begin{aligned}
P(q^+|Pass) &= \frac{P(Pass|q^+)P(q^+)}{P(Pass)} = \frac{0.8 \times 0.7}{0.665} \approx 0.8421 \\
P(q^-|Pass) &\approx 1 - 0.8421 = 0.1579 \\
P(q^+|\neg Pass) &= \frac{P(\neg Pass|q^+)P(q^+)}{P(\neg Pass)} = \frac{0.2 \times 0.7}{0.335} \approx 0.4179 \\
P(q^-|\neg Pass) &\approx 1 - 0.4179 = 0.5821
\end{aligned}$$

d. If the car passes the test, the expected value of buying is

$$\begin{aligned}
&P(q^+|Pass)(2000 - 1500) + P(q^-|Pass)(2000 - 2200) \\
&= 0.8421 \times 500 + 0.1579 \times -200 = 378.92
\end{aligned}$$

Thus buying is the best decision given a pass. If the car fails the test, the expected value of buying is

$$\begin{aligned}
&P(q^+|\neg Pass)(2000 - 1500) + P(q^-|\neg Pass)(2000 - 2200) \\
&= 0.4179 \times 500 + 0.5821 \times -200 = 92.53
\end{aligned}$$

Buying is again the best decision.

e. Since the action is the same for both outcomes of the test, the test itself is worthless (if it is the only possible test) and the optimal plan is simply to buy the car without the test. (This is a trivial conditional plan.) For the test to be worthwhile, it would need to be more discriminating in order to reduce the probability $P(q^+|\neg Pass)$. The test would also be worthwhile if the market value of the car were less, or if the cost of repairs were more.

An interesting additional exercise is to prove the general proposition that if α is the best action for all the outcomes of a test then it must be the best action in the absence of the test outcome.

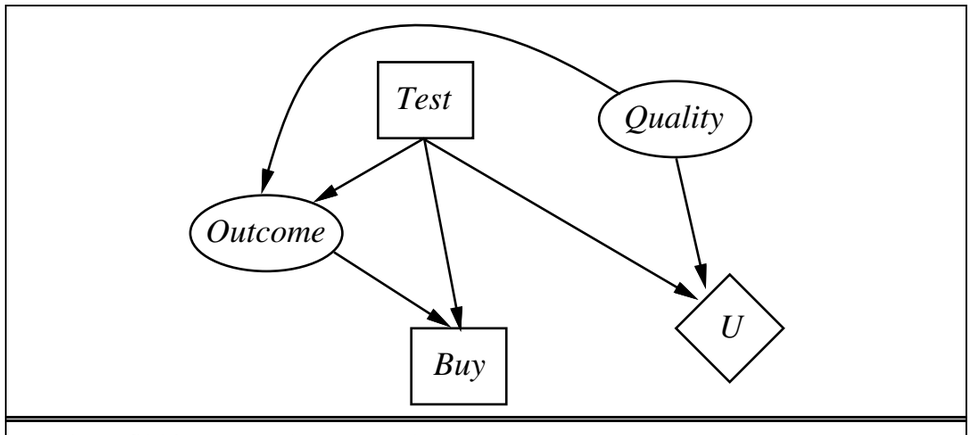


Figure S16.2 A decision network for the car-buying problem.

Using Bayes' theorem:

$$\begin{aligned}
P(q^+|Pass) &= \frac{P(Pass|q^+)P(q^+)}{P(Pass)} = \frac{0.8 \times 0.7}{0.665} \approx 0.8421 \\
P(q^-|Pass) &\approx 1 - 0.8421 = 0.1579 \\
P(q^+|\neg Pass) &= \frac{P(\neg Pass|q^+)P(q^+)}{P(\neg Pass)} = \frac{0.2 \times 0.7}{0.335} \approx 0.4179 \\
P(q^-|\neg Pass) &\approx 1 - 0.4179 = 0.5821
\end{aligned}$$

d. 如果汽车通过测试，购买的预期价值是

$$\begin{aligned}
&P(q^+|Pass)(2000 - 1500) + P(q^-|Pass)(2000 - 2200) \\
&= 0.8421 \times 500 + 0.1579 \times -200 = 378.92
\end{aligned}$$

因此，购买是最好的决定给予通行证。 如果汽车测试失败，购买的预期价值是

$$\begin{aligned}
&P(q^+|\neg Pass)(2000 - 1500) + P(q^-|\neg Pass)(2000 - 2200) \\
&= 0.4179 \times 500 + 0.5821 \times -200 = 92.53
\end{aligned}$$

购买再次是最好的决定。 e. 由于测试的两个结果的动作是相同的，因此测试本身是毫无价值的（如果它是唯一可能的测试），最佳计划只是在没有测试的情况下购买汽车。 （这是一个微不足道的有条件的计划。）为了使测试有价值，它需要更具鉴别性，以减少概率P（q+|Pass）。 如果汽车的市场价值较低，或者修理成本较高，测试也是值得的。

一个有趣的额外练习是证明一般命题，如果α是测试的所有结果的最佳动作，那么它必须是在没有测试结果的情况下的最佳动作。

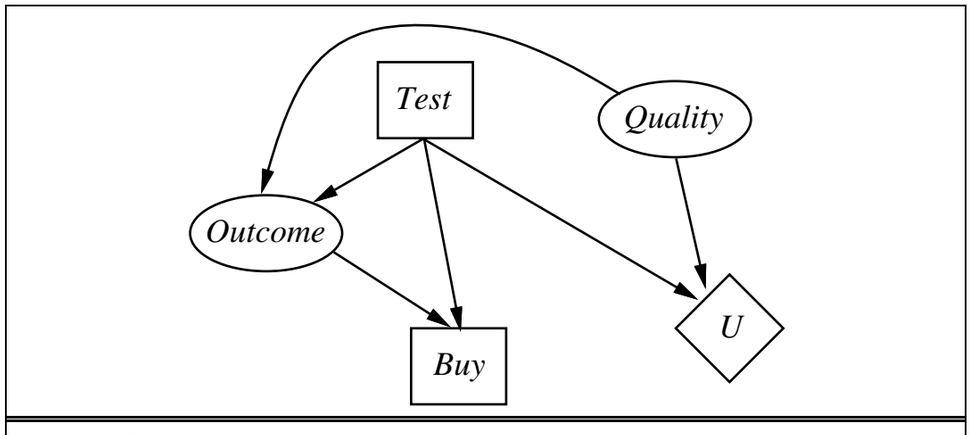


Figure S16.2 购车问题的决策网络。

16.18

a. Intuitively, the value of information is nonnegative because in the worst case one could simply ignore the information and act as if it was not available. A formal proof therefore begins by showing that this policy results in the same expected utility. The formula for the value of information is

$$VPI_E(E_j) = \left(\sum_k P(E_j = e_{jk}|E)EU(\alpha_{e_{jk}}|E, E_j = e_{jk})\right) - EU(\alpha|E)$$

If the agent does α given the information E_j, its expected utility is

$$\sum_k P(E_j = e_{jk}|E)EU(\alpha|E, E_j = e_{jk}) = EU(\alpha|E)$$

where the equality holds because the LHS is just the conditionalization of the RHS with respect to E_j. By definition,

$$EU(\alpha_{e_{jk}}|E, E_j = e_{jk}) \geq EU(\alpha|E, E_j = e_{jk})$$

hence $VPI_E(E_j) \geq 0$.

b. One explanation is that people are aware of their own irrationality and may want to avoid making a decision on the basis of the extra information. Another might be that the value of information is small compared to the value of surprise—for example, many people prefer not to know in advance what their birthday present is going to be.

c. Value of information is not submodular in general. Suppose that A is the empty set and B is the set $Y = 1$; and suppose that the optimal decision remains unchanged unless both $X = 1$ and $Y = 1$ are observed.

16.18

a. 直观地说，信息的价值是非负的，因为在最坏的情况下，人们可以简单地忽略这些信息，就好像它不可用一样。 因此，正式证明首先表明此策略导致相同的预期效用。 息的值的公式为

$$VPI_E(E_j) = \left(\sum_k P(E_j = e_{jk}|E)EU(\alpha_{e_{jk}}|E, E_j = e_{jk})\right) - EU(\alpha|E)$$

如果代理在给定信息E j的情况下执行α，则其预期效用为

$$\sum_k P(E_j = e_{jk}|E)EU(\alpha|E, E_j = e_{jk}) = EU(\alpha|E)$$

其中平等成立，因为LHS只是RHS相对于E j的条件化。 根据定义，

$$EU(\alpha_{e_{jk}}|E, E_j = e_{jk}) \geq EU(\alpha|E, E_j = e_{jk})$$

因此V PI E(E j)≥0。 b. 一种解释是，人们意识到自己的不合理性，可能希望避免根据额外的信息做出决定。 另一个可能是信息的价值与惊喜的价值相比较小–例如，许多人宁愿事先不知道他们的生日礼物是什么。 c. 信息的价值不是一般的子模块. 假设A是空集，B是集合Y=1;并且假设最优决策保持不变，除非同时观察到X=1和Y=1。

Solutions for Chapter 17
Making Complex Decisions

17.1 The best way to calculate this is NOT by thinking of all ways to get to any given square and how likely all those ways are, but to compute the occupancy probabilities at each time step. These are as follows:

		Up	*Up*	*Right*	*Right*	*Right*
(1,1)	1	.1	.02	.026	.0284	.02462
(1,2)		.8	.24	.258	.2178	.18054
(1,3)			.64	.088	.0346	.02524
(2,1)		.1	.09	.034	.0276	.02824
(2,3)				.512	.1728	.06224
(3,1)			.01	.073	.0346	.02627
(3,2)				.001	.0073	.04443
(3,3)					.4097	.17994
(4,1)				.008	.0656	.08672
(4,2)					.0016	.01400
(4,3)						.32776

Projection in an HMM involves multiplying the vector of occupancy probabilities by the transition matrix. Here, the only difference is that there is a different transition matrix for each action.

17.2 If we pick the policy that goes *Right* in all the optional states, and construct the corresponding transition matrix $\mathbf{T}$, we find that the equilibrium distribution—the solution to $\mathbf{Tx} = \mathbf{x}$—has occupancy probabilities of 1/12 for (2,3), (3,1), (3,2), (3,3) and 2/3 for (4,1). These can be found most simply by computing $\mathbf{T}^n\mathbf{x}$ for any initial occupancy vector $\mathbf{x}$, for n large enough to achieve convergence.

17.3 Stationarity requires the agent to have identical preferences between the sequence pair $[s_0, s_1, s_2, \ldots]$, $[s_0, s'_1, s'_2, \ldots]$ and between the sequence pair $[s_1, s_2, \ldots]$, $[s'_1, s'_2, \ldots]$. If the utility of a sequence is its maximum reward, we can easily violate stationarity. For example,

$$[4, 3, 0, 0, 0, \ldots] \sim [4, 0, 0, 0, 0, \ldots]$$

but

$$[3, 0, 0, 0, \ldots] \succ [0, 0, 0, 0, \ldots] .$$

第17章的解决方案
做出复杂的决策

17.1计算这一点的最佳方法不是考虑到达任何给定正方形的所有方法以及所有这些方法的可能性，而是计算每个时间步的占用概率。 这些如下:

		向上右				*Right*
(1,1)	1	.1	.02	.026	.0284	.02462
(1,2)		.8	.24	.258	.2178	.18054
(1,3)			.64	.088	.0346	.02524
(2,1)		.1	.09	.034	.0276	.02824
(2,3)				.512	.1728	.06224
(3,1)			.01	.073	.0346	.02627
(3,2)				.001	.0073	.04443
(3,3)					.4097	.17994
(4,1)				.008	.0656	.08672
(4,2)					.0016	.01400
(4,3)						.32776

HMM中的投影涉及将占用概率的向量乘以过渡矩阵。 在这里，唯一的区别是每个动作都有不同的过渡矩阵。

17.2如果我们选择在所有可选状态中正确的策略，并构造cor响应过渡矩阵T，我们发现平衡分布-Tx=x的解-具有1/12的占用概率。(2,3), (3,1), (3,2), (3,3) 和2/3为（4,1）。 这些可以最简单地通过对任何初始占用向量x计算T n x来找到，对于足够大以实现收敛的n。

17.3平稳性要求代理在序列对之间具有相同的偏好[s0，s1，s2，。..],[s0,s'1，s'2，。..]. 如果一个序列的效用是它的最大回报，我们很容易违反平稳性。 例如,

$$[4, 3, 0, 0, 0, \ldots] \sim [4, 0, 0, 0, 0, \ldots]$$

but

$$[3, 0, 0, 0, \ldots] \succ [0, 0, 0, 0, \ldots] .$$

We can still define $U^{\pi}(s)$ as the expected maximum reward obtained by executing π starting in s. The agent's preferences seem peculiar, nonetheless. For example, if the current state s has reward $R_{\max}$, the agent will be indifferent among all actions, but once the action is executed and the agent is no longer in s, it will suddenly start to care about what happens next.

17.4 This is a deceptively simple exercise that tests the student's understanding of the formal definition of MDPs. Some students may need a hint or an example to get started.

a. The key here is to get the max and summation in the right place. For $R(s,a)$ we have

$$U(s) = \max_a [R(s,a) + \gamma \sum_{s'} T(s,a,s')U(s')$$

and for $R(s,a,s')$ we have

$$U(s) = \max_a \sum_{s'} T(s,a,s')[R(s,a,s') + \gamma U(s')] \,.$$

b. There are a variety of solutions here. One is to create a "pre-state" $pre(s,a,s')$ for every s, a, s', such that executing a in s leads not to s' but to $pre(s,a,s')$. In this state is encoded the fact that the agent came from s and did a to get here. From the pre-state, there is just one action b that always leads to s'. Let the new MDP have transition T', reward R', and discount γ'. Then

$$\begin{aligned}
&T'(s,a,pre(s,a,s')) = T(s,a,s') \\
&T'(pre(s,a,s'),b,s') = 1 \\
&R'(s,a) = 0 \\
&R'(pre(s,a,s'),b) = \gamma^{-\frac{1}{2}} R(s,a,s') \\
&\gamma' = \gamma^{\frac{1}{2}}
\end{aligned}$$

c. In keeping with the idea of part (b), we can create states $post(s,a)$ for every s, a, such that

$$\begin{aligned}
&T'(s,a,post(s,a,s')) = 1 \\
&T'(post(s,a,s'),b,s') = T(s,a,s') \\
&R'(s) = 0 \\
&R'(post(s,a,s')) = \gamma^{-\frac{1}{2}} R(s,a) \\
&\gamma' = \gamma^{\frac{1}{2}}
\end{aligned}$$

17.5 This can be done fairly simply by:

- Call `policy-iteration` (from `"uncertainty/algorithms/dp.lisp"`) on the Markov Decision Processes representing the 4x3 grid, with values for the step cost ranging from, say, 0.0 to 1.0 in increments of 0.02.
- For any two adjacent policies that differ, run binary search on the step cost to pinpoint the threshold value.
- Convince yourself that you haven't missed any policies, either by using too coarse an increment in step size (0.02), or by stopping too soon (1.0).

我们仍然可以将U π（s）定义为从s开始执行π获得的预期最大奖励。 尽管如此，代理人的偏好似乎很奇怪。 例如，如果当前状态s具有奖励R max，则代理将在所有动作中无动于衷，但是一旦动作被执行并且代理不再在s中，它将突然开始关心接下来发生的事情。

17.4这是一个看似简单的练习，测试学生对MDPs for mal定义的理解。 有些学生可能需要一个提示或一个例子来开始。

a. 这里的关键是在正确的地方获得最大值和求和。 对于R（s，a）我们有

$$U(s) = \max_a [R(s,a) + \gamma \sum_{s'} T(s,a,s')U(s')$$

对于R（s，a，s'），我们有

$$U(s) = \max_a \sum_{s'} T(s,a,s')[R(s,a,s') + \gamma U(s')] \,.$$

b. 这里有各种各样的解决方案。 一种是为每个s，a，s'创建一个"预状态"pre(s,a,s')，使得在s中执行a不会导致s'而是pre(s,a,s')。 在这种状态下，编码的事实是，代理来自s，并做了一个到达这里。 从前状态来看，只有一个动作b总是导致s'。 让新的MDP具有过渡T'，奖励R'和折扣γ'。 然后

$$\begin{aligned}
&T'(s,a,pre(s,a,s')) = T(s,a,s') \\
&T'(pre(s,a,s'),b,s') = 1 \\
&R'(s,a) = 0 \\
&R'(pre(s,a,s'),b) = \gamma^{-\frac{1}{2}} R(s,a,s') \\
&\gamma' = \gamma^{\frac{1}{2}}
\end{aligned}$$

c. 根据部分（b）的想法，我们可以为每个s，a创建状态post（s，a），这样

$$\begin{aligned}
&T'(s,a,post(s,a,s')) = 1 \\
&T'(post(s,a,s'),b,s') = T(s,a,s') \\
&R'(s) = 0 \\
&R'(post(s,a,s')) = \gamma^{-\frac{1}{2}} R(s,a) \\
&\gamma' = \gamma^{\frac{1}{2}}
\end{aligned}$$

17.5 这可以相当简单地通过:

*调用策略迭代（来自"不确定性/算法/dp。在表示4x3网格的马尔可夫决策过程上，步进成本的值从0.0到1.0，增量为0.02。 •对于任何两个不同的相邻策略，请在步长成本上运行二进制搜索以确定阈值。 *说服自己，你没有错过任何策略，要么通过使用步长（0.02）的过粗增量，要么通过过早停止（1.0）。

One useful observation in this context is that the expected total reward of any fixed policy is linear in r, the per-step reward for the empty states. Imagine drawing the total reward of a policy as a function of r—a straight line. Now draw all the straight lines corresponding to all possible policies. The reward of the *optimal* policy as a function of r is just the max of all these straight lines. Therefore it is a piecewise linear, convex function of r. Hence there is a very efficient way to find *all* the optimal policy regions:

- For any two consecutive values of r that have different optimal policies, find the optimal policy for the midpoint. Once two consecutive values of r give the same policy, then the interval between the two points must be covered by that policy.
- Repeat this until two points are known for each distinct optimal policy.
- Suppose (r_{a1}, v_{a1}) and (r_{a2}, v_{a2}) are points for policy a, and (r_{b1}, v_{b1}) and (r_{b2}, v_{b2}) are the next two points, for policy b. Clearly, we can draw straight lines through these pairs of points and find their intersection. This does not mean, however, that there is no other optimal policy for the intervening region. We can determine this by calculating the optimal policy for the intersection point. If we get a different policy, we continue the process.

The policies and boundaries derived from this procedure are shown in Figure S17.1. The figure shows that there are *nine* distinct optimal policies! Notice that as r becomes more negative, the agent becomes more willing to dive straight into the –1 terminal state rather than face the cost of the detour to the +1 state.

The somewhat ugly code is as follows. Notice that because the lines for neighboring policies are very nearly parallel, numerical instability is a serious problem.

```
(defun policy-surface (mdp r1 r2 &aux prev (unchanged nil))
  "returns points on the piecewise-linear surface
   defined by the value of the optimal policy of mdp as a
   function of r"
  (setq rvplist
    (list (cons r1 (r-policy mdp r1)) (cons r2 (r-policy mdp r2))))
  (do ()
      (unchanged rvplist)
    (setq unchanged t)
    (setq prev nil)
    (dolist (rvp rvplist)
      (let* ((rest (cdr (member rvp rvplist :test #'eq)))
             (next (first rest))
             (next-but-one (second rest)))
        (dprint (list (first prev) (first rvp)
                      '* (first next) (first next-but-one)))
        (when next
          (unless (or (= (first rvp) (first next))
                      (policy-equal (third rvp) (third next) mdp))
            (dprint "Adding new point(s)")
            (setq unchanged nil)
            (if (and prev next-but-one
                     (policy-equal (third prev) (third rvp) mdp)
                     (policy-equal (third next) (third next-but-one) mdp))
```


在这种情况下，一个有用的观察是，任何固定策略的预期总奖励在r中是线性的，即空状态的每步奖励。 想象一下，将策略的总奖励绘制为r-a直线的函数。 现在绘制所有可能策略对应的所有直线。 最优策略作为r的函数的奖励只是所有这些直线的最大值。 因此它是r的分段线性，凸函数。 因此，有一个非常有效的方法来找到所有最优的政策区域:

•对于具有不同最优策略的r的任意两个连续值，为中点找到最优策略。 一旦r的两个连续值给出相同的策略，那么两点之间的间隔必须由该策略复盖。 *重复此操作，直到每个不同的最优策略都知道两个点。 *假设(r a1,v a1)和(r a2,v a2)是策略a的点，而(r b1,v b1)和(r b2,v b2)是策略b的下两个点。 显然，我们可以通过这些点对绘制直线并找到它们的交点。 然而，这并不意味着对干预区域没有其他最佳政策。 我们可以通过计算交叉点的最优策略来确定这一点。 如果我们得到不同的政策，我们继续这个过程。

从该过程得出的策略和边界如图S17.1所示。 图中显示有九种截然不同的最优策略！请注意，随着r变得更负，代理变得更愿意直接潜入-1终端状态，而不是面对绕道到+1状态的成本。

有些难看的代码如下。 请注意，由于相邻策略的线非常接近平行，因此数值不稳定是一个严重的问题。

(defun policy-surface(mdp r1r2&aux prev(unchanged nil))"返回分段线性曲面上的点
由作为r的函数的mdp的最优策略的值定义"(setq rvplist

```
    (list (cons r1 (r-policy mdp r1)) (cons r2 (r-policy mdp r2))))
  (do ()
      (unchanged rvplist)
    (setq unchanged t)
    (setq prev nil)
    (dolist (rvp rvplist)
      (let* ((rest (cdr (member rvp rvplist :test #'eq)))
             (next (first rest))
             (next-but-one (second rest)))
        (dprint (list (first prev) (first rvp)
                      '* (first next) (first next-but-one)))
        (when next
          (unless (or (= (first rvp) (first next))
```

(policy-equal(third rvp)(third next)mdp))(dprint"Adding new point(s)")(setq不变nil)(if(and prev next-but-one

```
                     (policy-equal (third prev) (third rvp) mdp)
                     (policy-equal (third next) (third next-but-one) mdp))
```

```
                (let* ((intxy (policy-vertex prev rvp next next-but-one))
                       (int (cons (xy-x intxy) (r-policy mdp (xy-x intxy)))))
                  (dprint (list "Found intersection" intxy))
                  (cond ((or (< (first int) (first rvp))
                             (> (first int) (first next)))
                         (dprint "Intersection out of range!")
                         (let ((int-r (/ (+ (first rvp) (first next)) 2)))
                           (setq int (cons int-r (r-policy mdp int-r))))
                         (push int (cdr (member rvp rvplist :test #'eq))))
                        ((or (policy-equal (third rvp) (third int) mdp)
                             (policy-equal (third next) (third int) mdp))
                         (dprint "Found policy boundary")
                         (push (list (first int) (second int) (third next))
                               (cdr (member rvp rvplist :test #'eq)))
                         (push (list (first int) (second int) (third rvp))
                               (cdr (member rvp rvplist :test #'eq))))
                        (t (dprint "Found new policy!")
                         (push int (cdr (member rvp rvplist :test #'eq))))))
              (let* ((int-r (/ (+ (first rvp) (first next)) 2))
                     (int (cons int-r (r-policy mdp int-r))))
                (dprint (list "Adding split point" (list int-r (second int))))
                (push int (cdr (member rvp rvplist :test #'eq))))))))
      (setq prev rvp))))

(defun r-policy (mdp r &aux U)
  (set-rewards mdp r)
  (setq U (value-iteration mdp
                           (copy-hash-table (mdp-rewards mdp)) #'identity)
                           :epsilon 0.0000000001))
  (list (gethash '(1 1) U) (optimal-policy U (mdp-model mdp) (mdp-rewards mdp))))

(defun set-rewards (mdp r &aux (rewards (mdp-rewards mdp))
                               (terminals (mdp-terminal-states mdp)))
  (maphash #'(lambda (state reward)
               (unless (member state terminals :test #'equal)
                   (setf (gethash state rewards) r)))
           rewards))

(defun policy-equal (p1 p2 mdp &aux (match t)
                                    (terminals (mdp-terminal-states mdp)))
  (maphash #'(lambda (state action)
               (unless (member state terminals :test #'equal)
                 (unless (eq (caar (gethash state p1)) (caar (gethash state p2)))
                   (setq match nil))))
           p1)
  match)

(defun policy-vertex (rvp1 rvp2 rvp3 rvp4)
  (let  ((a (make-xy :x (first rvp1) :y (second rvp1)))
         (b (make-xy :x (first rvp2) :y (second rvp2)))
         (c (make-xy :x (first rvp3) :y (second rvp3)))
         (d (make-xy :x (first rvp4) :y (second rvp4)))))
```



```
(let* ((intxy (policy-vertex prev rvp next next-but-one))
       (int (cons (xy-x intxy) (r-policy mdp (xy-x intxy)))))
  (dprint (list "Found intersection" intxy))
  (cond ((or (< (first int) (first rvp))
```

(>(first int)(first next)))(dprint"交叉点超出范围！")(let((int-r(/(+(first rvp)(first next))2)))(setq int(cons int-r(r-policy mdp int-r))))(push int(cdr(member rvp rvplist:test#'eq))))(or(policy-equal(third rvp)(third int)mdp)

(policy-equal(third next)(third int)mdp))(dprint"Found policy boundary")(push(list(first int)(second int)(third next))(cdr(member rvp rvplist:test#'eq)))(push(list(first int)(second int)(third rvp))(cdr(member rvp rvplist:test#'eq))))(t(dprint"Found new policy!")

(push int(cdr(member rvp rvplist:test#'eq))))))(let*((int-r(/(+(first rvp)(first next))2))(int(cons int-r(r-policy mdp int-r))))(dprint(list"Adding split point"(list int-r(second int))))(push int(cdr(member rvp rvplist:test#'eq)))))))(setq prev rvp))))

```
(defun r-policy (mdp r &aux U)
  (set-rewards mdp r)
  (setq U (value-iteration mdp
                           (copy-hash-table (mdp-rewards mdp)) #'identity)
                           :epsilon 0.0000000001))
  (list (gethash '(1 1) U) (optimal-policy U (mdp-model mdp) (mdp-rewards mdp))))

(defun set-rewards (mdp r &aux (rewards (mdp-rewards mdp))
                               (terminals (mdp-terminal-states mdp)))
  (maphash #'(lambda (state reward)
```

（除非（成员国终端：test#'equal）（setf（gethash state rewards）r））））奖励））

```
(defun policy-equal (p1 p2 mdp &aux (match t)
                                    (terminals (mdp-terminal-states mdp)))
  (maphash #'(lambda (state action)
```

（除非（成员国终端：测试#'相等）

（除非（eq（caar（gethash状态p1））（caar（gethash状态p2））））（setq匹配nil））））

```
           p1)
  match)

(defun policy-vertex (rvp1 rvp2 rvp3 rvp4)
  (let  ((a (make-xy :x (first rvp1) :y (second rvp1)))
         (b (make-xy :x (first rvp2) :y (second rvp2)))
         (c (make-xy :x (first rvp3) :y (second rvp3)))
         (d (make-xy :x (first rvp4) :y (second rvp4)))))
```

```
    (intersection-point (make-line :xy1 a :xy2 b)
                        (make-line :xy1 c :xy2 d))))

(defun intersection-point (l1 l2)
;;; l1 is line ab; l2 is line cd
;;; assume the lines cross at alpha a + (1-alpha) b,
;;;     also known as beta c + (1-beta) d
;;; returns the intersection point unless they're parallel
  (let* ((a (line-xy1 l1))
         (b (line-xy2 l1))
         (c (line-xy1 l2))
         (d (line-xy2 l2))
         (xa (xy-x a)) (ya (xy-y a))
         (xb (xy-x b)) (yb (xy-y b))
         (xc (xy-x c)) (yc (xy-y c))
         (xd (xy-x d)) (yd (xy-y d))
         (q (- (* (- xa xb) (- yc yd))
               (* (- ya yb) (- xc xd)))))
    (unless (zerop q)
      (let ((alpha (/ (- (* (- xd xb) (- yc yd))
                         (* (- yd yb) (- xc xd)))
                      q)))
        (make-xy :x (float (+ (* alpha xa) (* (- 1 alpha) xb)))
                 :y (float (+ (* alpha ya) (* (- 1 alpha) yb))))))))
```

17.6

a. To find the proof, it may help first to draw a picture of two arbitrary functions f and g and mark the maxima; then it is easy to find a point where the difference between the functions is bigger than the difference between the maxima. Assume, w.l.o.g., that $\max_a f(a) \geq \max_a g(a)$, and let f have its maximum value at a^*. Then we have

$$\begin{aligned}
|\max_a f(a) - \max_a g(a)| &= \max_a f(a) - \max_a g(a) \qquad \text{(by assumption)} \\
&= f(a^*) - \max_a g(a) \\
&\leq f(a^*) - g(a^*) \\
&\leq \max_a |f(a) - g(a)| \qquad \text{(by definition of max)}
\end{aligned}$$

b. From the definition of the B operator (Equation (17.6)) we have

$$\begin{aligned}
|(B\,U_i - B\,U_i')(s)| &= |R(s) + \gamma \max_{a \in A(s)} \sum_{s'} P(s' \mid s,a) U_i(s') \\
&\quad - R(s) - \gamma \max_{a \in A(s)} \sum_{s'} P(s' \mid s,a) U_i'(s')| \\
&= \gamma |\max_{a \in A(s)} \sum_{s'} P(s' \mid s,a) U_i(s') - \max_{a \in A(s)} \sum_{s'} P(s' \mid s,a) U_i'(s')| \\
&\leq \gamma \max_{a \in A(s)} |\sum_{s'} P(s' \mid s,a) U_i(s') - \sum_{s'} P(s' \mid s,a) U_i'(s')|
\end{aligned}$$

161

```
    (intersection-point (make-line :xy1 a :xy2 b)
                        (make-line :xy1 c :xy2 d))))
```

(defun intersection-point(l1 l2);;;l1是线ab;l2是线cd;;;假设线在alpha a+(1-alpha)b处交叉，;;;

也称为beta c+(1-beta)d;;;返回交点，除非它们是平行的(let*((a(line-xy1 l1))(b(line-xy2 l1))(c(line-xy1 l2))(d(line-xy2 l2))(xa(xy-x a))(ya(xy-y a))(xb(xy-x b))(yb(xy-y b))(xc(xy-x c))(yc(xy-y c))(xd(xy-x d))(yd(xy-y d))(q((*(xa xb)(yc yd))

```
           (* (- ya yb) (- xc xd)))))
    (unless (zerop q)
      (let ((alpha (/ (- (* (- xd xb) (- yc yd))
                         (* (- yd yb) (- xc xd)))
                      q)))
        (make-xy :x (float (+ (* alpha xa) (* (- 1 alpha) xb)))
                 :y (float (+ (* alpha ya) (* (- 1 alpha) yb))))))))
```

17.6

a. 为了找到证明，首先绘制两个任意函数f和g的图片并标记极大值;然后很容易找到一个函数之间的差异大于极大值之间的差异的点。 假设，w.l.o.g.，max a f(a)≥max a g(a)，设f在a∞处具有其最大值。 那么我们有

$$\begin{aligned}
|\max_a f(a) - \max_a g(a)| &= \max_a f(a) - \max_a g(a) \qquad \text{(by assumption)} \\
&= f(a^*) - \max_a g(a) \\
&\leq f(a^*) - g(a^*) \\
&\leq \max_a |f(a) - g(a)| \qquad \text{（根据max的定义）}
\end{aligned}$$

b. 从B算子的定义（方程（17.6））我们有

$$\begin{aligned}
|(B\,U_i - B\,U_i')(s)| &= |R(s) + \gamma \max_{a \in A(s)} \sum_{s'} P(s' \mid s,a) U_i(s') \\
&\quad - R(s) - \gamma \max_{a \in A(s)} \sum_{s'} P(s' \mid s,a) U_i'(s')| \\
&= \gamma |\max_{a \in A(s)} \sum_{s'} P(s' \mid s,a) U_i(s') - \max_{a \in A(s)} \sum_{s'} P(s' \mid s,a) U_i'(s')| \\
&\leq \gamma \max_{a \in A(s)} |\sum_{s'} P(s' \mid s,a) U_i(s') - \sum_{s'} P(s' \mid s,a) U_i'(s')|
\end{aligned}$$

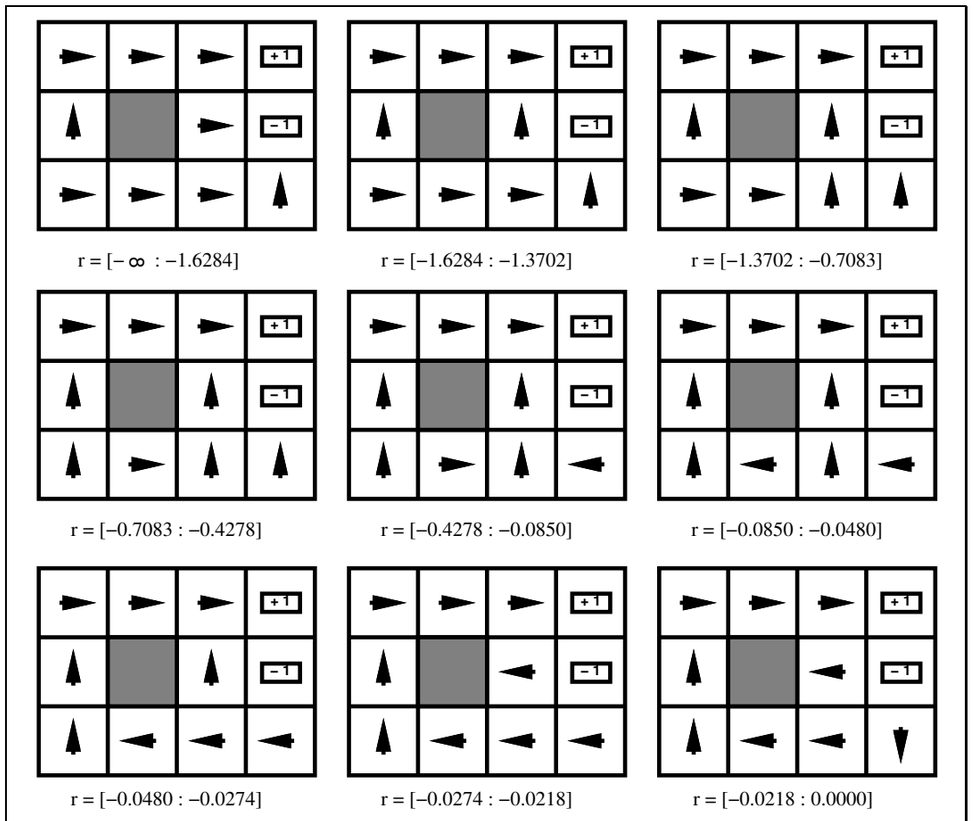
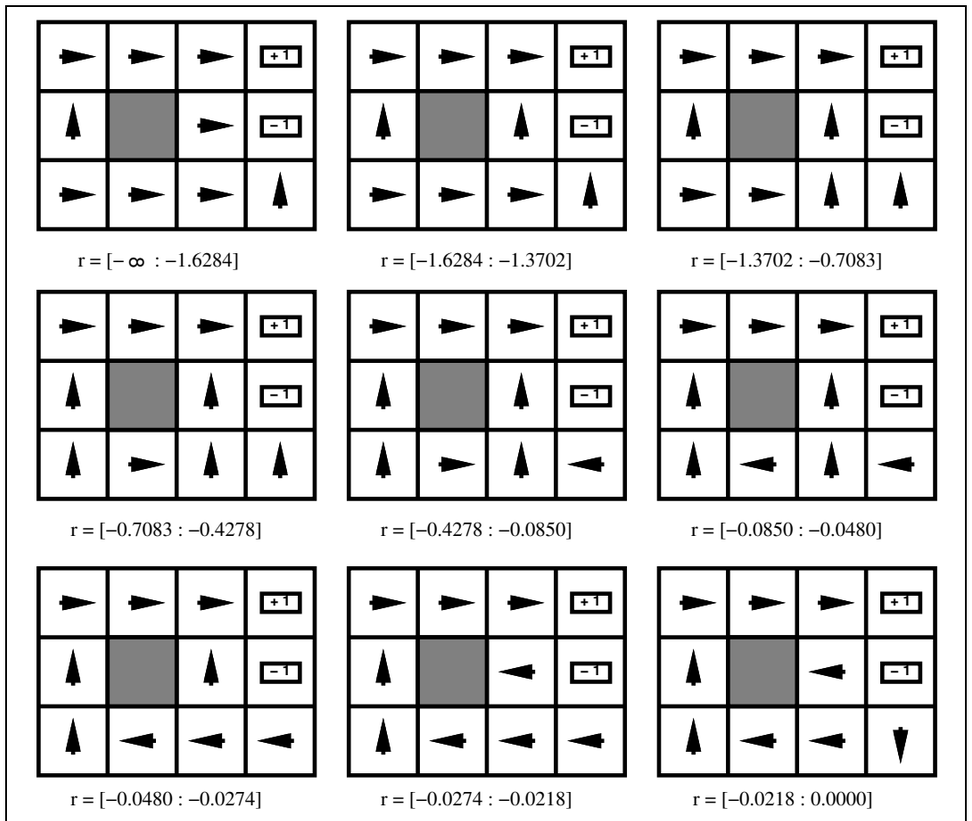


Figure S17.1 Optimal policies for different values of the cost of a step in the 4×3 environment, and the boundaries of the regions with constant optimal policy.

$$= \gamma |\sum_{s'} P(s' \mid s, a^*(s)) U_i(s') - \sum_{s'} P(s' \mid s, a^*(s)) U'_i(s')|$$
$$= \gamma |\sum_{s'} P(s' \mid s, a^*(s)) (U_i(s') - U'_i(s')|$$

Inserting this into the expression for the max norm, we have

$$\begin{aligned} ||B\,U_i - B\,U'_i)|| &= \max_s |(B\,U_i - B\,U'_i)(s)| \\ &\leq \gamma \max_s |\sum_{s'} P(s' \mid s, a^*(s))(U_i(s') - U'_i(s'))| \\ &\leq \gamma \max_s |U_i(s) - U'_i(s)| = \gamma ||U_i - U'_i|| \end{aligned}$$



图S17.1在4×3环境中一步成本的不同值的最优策略，以及具有恒定最优策略的区域边界。

$$= \gamma |\sum_{s'} P(s' \mid s, a^*(s)) U_i(s') - \sum_{s'} P(s' \mid s, a^*(s)) U'_i(s')|$$
$$= \gamma |\sum_{s'} P(s' \mid s, a^*(s)) (U_i(s') - U'_i(s')|$$

将此插入到最大范数的表达式中，我们有

$$\begin{aligned} ||B\,U_i - B\,U'_i)|| &= \max_s |(B\,U_i - B\,U'_i)(s)| \\ &\leq \gamma \max_s |\sum_{s'} P(s' \mid s, a^*(s))(U_i(s') - U'_i(s'))| \\ &\leq \gamma \max_s |U_i(s) - U'_i(s)| = \gamma ||U_i - U'_i|| \end{aligned}$$

a. For U_A we have

$$U_A(s) = R(s) + \max_a \sum_{s'} P(s'|a,s)U_B(s')$$

and for U_B we have

$$U_B(s) = R(s) + \min_a \sum_{s'} P(s'|a,s)U_A(s') \ .$$

b. To do value iteration, we simply turn each equation from part (a) into a Bellman update and apply them in alternation, applying each to all states simultaneously. The process terminates when the utility vector for one player is the same as the previous utility vector *for the same player* (i.e., two steps earlier). (Note that typically U_A and U_B are not the same in equilibrium.)

c. The state space is shown in Figure S17.2.

d. We mark the terminal state values in bold and initialize other values to 0. Value iteration proceeds as follows:

	(1,4)	(2,4)	(3,4)	(1,3)	(2,3)	(4,3)	(1,2)	(3,2)	(4,2)	(2,1)	(3,1)
U_A	0	0	0	0	0	**+1**	0	0	**+1**	**–1**	**–1**
U_B	0	0	0	0	–1	**+1**	0	–1	**+1**	**–1**	**–1**
U_A	0	0	0	–1	+1	**+1**	–1	+1	**+1**	**–1**	**–1**
U_B	–1	+1	+1	–1	–1	**+1**	–1	–1	**+1**	**–1**	**–1**
U_A	+1	+1	+1	–1	+1	**+1**	–1	+1	**+1**	**–1**	**–1**
U_B	–1	+1	+1	–1	–1	**+1**	–1	–1	**+1**	**–1**	**–1**

and the optimal policy for each player is as follows:

	(1,4)	(2,4)	(3,4)	(1,3)	(2,3)	(4,3)	(1,2)	(3,2)	(4,2)	(2,1)	(3,1)
π_A^*	(2,4)	(3,4)	(2,4)	(2,3)	(4,3)		(3,2)	(4,2)			
π_B^*	(1,3)	(2,3)	(3,2)	(1,2)	(2,1)		(1,3)	(3,1)			

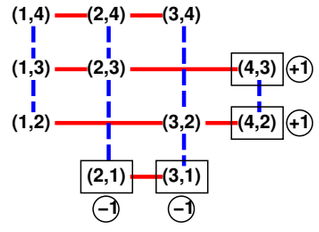


Figure S17.2 State-space graph for the game in Figure 5.17.

17.8

a. $r = 100$.

u	l	.
u	l	d
u	l	l

a. 对于U A我们有

$$U_A(s) = R(s) + \max_a \sum_{s'} P(s'|a,s)U_B(s')$$

而对于U B我们有

$$U_B(s) = R(s) + \min_a \sum_{s'} P(s'|a,s)U_A(s') \ .$$

b. 为了进行值迭代，我们只需将部分（a）中的每个方程转换为Bellman更新，并交替应用它们，同时将每个方程应用于所有状态。 当一个播放器的实用程序向量与同一播放器的前一个实用程序向量相同（即前两个步骤）时，该过程终止。 （请注意，通常U A和U B在平衡状态下并不相同。)c． 状态空间如图S17.2所示。 d. 我们用粗体标记终端状态值，并将其他值初始化为0。 值迭代进行如下:

	(1,4)	(2,4)	(3,4)	(1,3)	(2,3)	(4,3)	(1,2)	(3,2)	(4,2)	(2,1)	(3,1)
U_A	0	0	0	0	0	**+1**	0	0	**+1**	**–1**	**–1**
U_B	0	0	0	0	–1	**+1**	0	–1	**+1**	**–1**	**–1**
U_A	0	0	0	–1	+1	**+1**	–1	+1	**+1**	**–1**	**–1**
U_B	–1	+1	+1	–1	–1	**+1**	–1	–1	**+1**	**–1**	**–1**
U_A	+1	+1	+1	–1	+1	**+1**	–1	+1	**+1**	**–1**	**–1**
U_B	–1	+1	+1	–1	–1	**+1**	–1	–1	**+1**	**–1**	**–1**

每个玩家的最佳策略如下:

	(1,4)	(2,4)	(3,4)	(1,3)	(2,3)	(4,3)	(1,2)	(3,2)	(4,2)	(2,1)	(3,1)
π_A^*	(2,4)	(3,4)	(2,4)	(2,3)	(4,3)		(3,2)	(4,2)			
π_B^*	(1,3)	(2,3)	(3,2)	(1,2)	(2,1)		(1,3)	(3,1)			

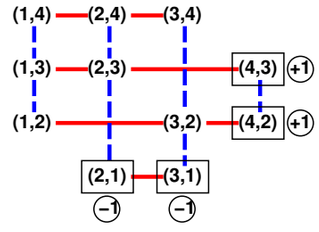


Figure S17.2 图5.17中游戏的状态空间图。

17.8

a. $r = 100$.

u	l	.
u	l	d
u	l	l

See the comments for part d. This should have been $r = -100$ to illustrate an alternative behavior:

r	r	.
d	r	u
r	r	u

Here, the agent tries to reach the goal quickly, subject to attempting to avoid the square $(1, 3)$ as much as possible. Note that the agent will choose to move Down in square $(1, 2)$ in order to actively avoid the possibility of "accidentally" moving into the square $(1, 3)$ if it tried to move Right instead, since the penalty for moving into square (1, 3) is so great.

b. $r = -3$.

r	r	.
r	r	u
r	r	u

Here, the agent again tries to reach the goal as fast as possible while attempting to avoid the square $(1, 3)$, but the penalty for square $(1, 3)$ is not so great that the agent will try to actively avoid it at all costs. Thus, the agent will choose to move Right in square $(1, 2)$ in order to try to get closer to the goal even if it occasionally will result in a transition to square $(1, 3)$.

c. $r = 0$.

r	r	.
u	u	u
u	u	u

Here, the agent again tries to reach the goal as fast as possible, but will try to do so via a path that includes square $(1, 3)$ if possible. This results from the fact that square $(1, 3)$ does not incur the reward of -1 in all other non-goal states, so it reaching the goal via a path through that square can potentially have slightly greater reward than another path of equal length that does not pass through $(1, 3)$.

d. $r = 3$.

u	l	.
u	l	d
u	l	l

17.9 The utility of Up is

$$50\gamma - \sum_{t=2}^{101} \gamma^t = 50\gamma - \gamma^2 \frac{1 - \gamma^{100}}{1 - \gamma},$$

while the utility of Down is

$$-50\gamma + \sum_{t=2}^{101} \gamma^t = -50\gamma + \gamma^2 \frac{1 - \gamma^{100}}{1 - \gamma}.$$

Solving numerically we find the indifference point to be $\gamma \approx 0.9844$: larger than this and we want to go Down to avoid the expensive long-term consequences, smaller than this and we want to go Up to get the immediate benefit.

这应该是r=-100来说明另一种行为:

rr. drurru在这里，代理试图快速达到目标，但要尽量避开正方形（1,3）。请注意，代理将选择在正方形（1,2）中向下移动，以便主动避免"意外"移动到正方形（1,3）的可能性，如果它试图向右移动，因为移动到正方形（1,3）的惩罚是如此之大。 b.r=-3.

rr. rruRru在这里，代理再次尝试尽可能快地达到目标，同时试图避开正方形（1,3），但对正方形（1,3）的惩罚并不是那么大，以至于代理将尝试不惜一切因此，代理将选择在正方形（1，2）中向右移动，以便尝试更接近目标，即使它偶尔会导致向正方形（1，3）的过渡。 c.r=0.

rr. uUUUUU在这里，代理再次尝试尽可能快地达到目标，但如果可能的话，将尝试通过包括square(1,3)的路径这样做。 这是由于square(1,3)在所有其他非目标状态中都不会产生-1的奖励，因此它通过通过该square的路径到达目标可能比没有通过（1,3）的等长路径具有稍大的奖励。 d.r=3.

u l .

u	l	d
u	l	l

17.9 Up的效用是

$$50\gamma - \sum_{t=2}^{101} \gamma^t = 50\gamma - \gamma^2 \frac{1 - \gamma^{100}}{1 - \gamma},$$

而Down的效用是

$$-50\gamma + \sum_{t=2}^{101} \gamma^t = -50\gamma + \gamma^2 \frac{1 - \gamma^{100}}{1 - \gamma}.$$

通过数值求解，我们发现冷漠点为γ∈0。 9844：比这更大，我们想下去以避免昂贵的长期后果，比这更小，我们想上去以获得直接的利益。

17.10

a. Intuitively, the agent wants to get to state 3 as soon as possible, because it will pay a cost for each time step it spends in states 1 and 2. However, the only action that reaches state 3 (action b) succeeds with low probability, so the agent should minimize the cost it incurs while trying to reach the terminal state. This suggests that the agent should definitely try action b in state 1; in state 2, it might be better to try action a to get to state 1 (which is the better place to wait for admission to state 3), rather than aiming directly for state 3. The decision in state 2 involves a numerical tradeoff.

b. The application of policy iteration proceeds in alternating steps of value determination and policy update.

- *Initialization*: $U \leftarrow \langle -1,\ -2,\ 0\rangle$, $P \leftarrow \langle b,\ b\rangle$.
- *Value determination*:

$$\begin{aligned} u_1 &= -1 + 0.1u_3 + 0.9u_1 \\ u_2 &= -2 + 0.1u_3 + 0.9u_2 \\ u_3 &= 0 \end{aligned}$$

That is, $u_1 = -10$ and $u_2 = -20$.

Policy update: In state 1,

$$\sum_j T(1,a,j)u_j = 0.8 \times -20 + 0.2 \times -10 = -18$$

while

$$\sum_j T(1,b,j)u_j = 0.1 \times 0 \times 0.9 \times -10 = -9$$

so action b is still preferred for state 1.

In state 2,

$$\sum_j T(1,a,j)u_j = 0.8 \times -10 + 0.2 \times -20 = -12$$

while

$$\sum_j T(1,b,j)u_j = 0.1 \times 0 \times 0.9 \times -20 = -18$$

so action a is preferred for state 1. We set $unchanged? \leftarrow \text{false}$ and proceed.

- *Value determination*:

$$\begin{aligned} u_1 &= -1 + 0.1u_3 + 0.9u_1 \\ u_2 &= -2 + 0.8u_1 + 0.2u_2 \\ u_3 &= 0 \end{aligned}$$

Once more $u_1 = -10$; now, $u_2 = -15$. *Policy update*: In state 1,

$$\sum_j T(1,a,j)u_j = 0.8 \times -15 + 0.2 \times -10 = -14$$

165

17.10

a. 直观地说，代理希望尽快到达状态3，因为它将为它在状态1和2中花费的每个时间步骤支付成本。 但是，到达状态3的唯一操作（操作b）以低概率成功，因此代理应尽量减少在尝试到达终端状态时所产生的成本。 这表明代理肯定应该在状态1中尝试动作b;在状态2中，尝试动作a到达状态1（这是等待进入状态3的更好位置）可能会更好，而不是直接瞄准状态3。 状态2中的决定涉及数值权衡。

b. 策略迭代的应用以值确定和策略更新的交替步骤进行。

*初始化：U← -1,-2,0 ,P b b *价值确定：

$$\begin{aligned} u_1 &= -1 + 0.1u_3 + 0.9u_1 \\ u_2 &= -2 + 0.1u_3 + 0.9u_2 \\ u_3 &= 0 \end{aligned}$$

也就是说，u1=-10和u2=-20。

政策更新：状态1,

$$\sum_j T(1,a,j)u_j = 0.8 \times -20 + 0.2 \times -10 = -18$$

while

$$\sum_j T(1,b,j)u_j = 0.1 \times 0 \times 0.9 \times -10 = -9$$

所以动作b对于状态1仍然是优选的。 在状态2,

$$\sum_j T(1,a,j)u_j = 0.8 \times -10 + 0.2 \times -20 = -12$$

while

$$\sum_j T(1,b,j)u_j = 0.1 \times 0 \times 0.9 \times -20 = -18$$

所以动作a对于状态1是优选的。 我们设置不变？ 错误，继续。 *价值确定：

$$\begin{aligned} u_1 &= -1 + 0.1u_3 + 0.9u_1 \\ u_2 &= -2 + 0.8u_1 + 0.2u_2 \\ u_3 &= 0 \end{aligned}$$

再次u1=-10;现在，u2=-15。 政策更新：状态1,

$$\sum_j T(1,a,j)u_j = 0.8 \times -15 + 0.2 \times -10 = -14$$

while

$$\sum_j T(1,b,j)u_j = 0.1 \times 0 \times 0.9 \times -10 = -9$$

so action b is still preferred for state 1.

In state 2,

$$\sum_j T(1,a,j)u_j = 0.8 \times -10 + 0.2 \times -15 = -11$$

while

$$\sum_j T(1,b,j)u_j = 0.1 \times 0 \times 0.9 \times -15 = -13.5$$

so action a is still preferred for state 1. $unchanged?$ remains true, and we terminate.

Note that the resulting policy matches our intuition: when in state 2, try to move to state 1, and when in state 1, try to move to state 3.

c. An initial policy with action a in both states leads to an unsolvable problem. The initial value determination problem has the form

$$\begin{aligned} u_1 &= -1 + 0.2u_1 + 0.8u_2 \\ u_2 &= -2 + 0.8u_1 + 0.2u_2 \\ u_3 &= 0 \end{aligned}$$

and the first two equations are inconsistent. If we were to try to solve them iteratively, we would find the values tending to $-\infty$.

Discounting leads to well-defined solutions by bounding the penalty (expected discounted cost) an agent can incur at either state. However, the choice of discount factor will affect the policy that results. For γ small, the cost incurred in the distant future plays a negligible role in the value computation, because γ^n is near 0. As a result, an agent could choose action b in state 2 because the discounted short-term cost of remaining in the non-terminal states (states 1 and 2) outweighs the discounted long-term cost of action b failing repeatedly and leaving the agent in state 2. An additional exercise could ask the student to determine the value of γ at which the agent is indifferent between the two choices.

17.11 The framework for this problem is in "uncertainty/domains/4x3-mdp.lisp". There is still some synthesis for the student to do for answer b. For c. some experimental design is necessary.

17.12 (Note: Early printings used "value determination," a term accidentally left over from the second edition, instead of "policy evaluation.") The policy evaluation algorithm calculates $U^\pi(s)$ for a given policy π. The policy for an agent that thinks $U$is the true utility and P is the true model would be based on Equation (17.4):

$$\pi(s) = \operatorname*{argmax}_{a \in A(s)} \sum_{s'} P(s' \mid s,a)U(s') \ .$$

while

$$\sum_j T(1,b,j)u_j = 0.1 \times 0 \times 0.9 \times -10 = -9$$

所以动作b对于状态1仍然是优选
的。 在状态2,

$$\sum_j T(1,a,j)u_j = 0.8 \times -10 + 0.2 \times -15 = -11$$

while

$$\sum_j T(1,b,j)u_j = 0.1 \times 0 \times 0.9 \times -15 = -13.5$$

所以动作a对于状态1仍然是优选的。 不变？ 保持真实，我们终止。

请注意，生成的策略与我们的直觉相匹配：当处于状态2时，尝试移动到状态1，当处于状态1时，尝试移动到状态3。 c. 在两个州采取行动a的初步政策导致了一个无法解决的问题。 初始值确定问题具有形式

$$\begin{aligned} u_1 &= -1 + 0.2u_1 + 0.8u_2 \\ u_2 &= -2 + 0.8u_1 + 0.2u_2 \\ u_3 &= 0 \end{aligned}$$

和前两个方程不一致。 如果我们尝试迭代地解决它们，我们会发现倾向于-∞的值。
折扣通过限制代理在任一状态下可能产生的惩罚（预期的dis计算成本），导致定义良好的解决方案。 然而,折扣系数的选择将影响所产生的政策. 对于γ小，在遥远的未来发生的成本在值计算中起着可忽略的作用，因为γ n接近于0。 因此，代理可以在状态2中选择操作b，因为在非终端状态（状态1和2）中重新主控的折扣短期成本超过重复失败并使代理处于状态2的折扣长期成本。 一个额外的exer cise可以要求学生确定代理人在两个选择之间无动于衷的γ值。

17.11 这个问题的框架在"不确定性/域/4x3-mdp。lisp"。
对于答案，学生还需要做一些综合。对于c来说，一些实验性的符号是必要的。

17.12（注：早期印刷使用"价值确定"，这是第二版意外遗留下来的一个术语，而不是"政策评估"。")策略评估算法针对给定策略π计算U π(s)。 对于认为U是真实效用且P是真实模型的代理的策略将基于等式（17.4）：

$$\pi(s) = \operatorname*{argmax}_{a \in A(s)} \sum_{s'} P(s' \mid s,a)U(s') \ .$$

Given this policy, the policy loss compared to the true optimal policy, starting in state s, is just $U^{\pi^*}(s) - U^{\pi}(s)$.

17.13 The belief state update is given by Equation (17.11), i.e.,

$$b'(s') = \alpha\, P(e \mid s') \sum_s P(s' \mid s, a) b(s) \; .$$

It may be helpful to compute this in two stages: update for the action, then update for the observation. The observation probabilities $P(e \mid s')$ are all either 0.9 (for squares that actually have one wall) or 0.1 (for squares with two walls). The following table shows the results. Note in particular how the probability mass concentrates on (3,2).

		Left	1 wall	*Left*	1 wall
(1,1)	.11111	.20000	.06569	.09197	.02090
(1,2)	.11111	.11111	.03650	.04234	.00962
(1,3)	.11111	.20000	.06569	.09197	.02090
(2,1)	.11111	.11111	.03650	.27007	.06136
(2,3)	.11111	.11111	.03650	.05985	.01360
(3,1)	.11111	.11111	.32847	.06861	.14030
(3,2)	.11111	.11111	.32847	.30219	.61791
(3,3)	.11111	.02222	.06569	.03942	.08060
(4,1)	.11111	.01111	.00365	.00036	.00008
(4,2)	0	.01111	.03285	.03321	.06791
(4,3)	0	0	0	0	0

17.14 In a sensorless environment, POMDP value iteration is essentially the same as ordinary state-space search—the branching ocurs only on acton choices, not observations. Hence the time complexity is $O(|A|^d)$.

17.15 Policies for the 2-state MDP all have a threshold belief p, such that if $b(0) > p$ then the optimal action is Go, otherwise it is $Stay$. The question is, what does this change do to the threshold? By making sensing more informative in state 0 and less informative in state 1, the change has made state 0 more desirable, hence the threshold value p *increases*.

17.16 This question is simple a matter of examining the definitions. In a dominant strategy equilibrium $[s_1, \ldots, s_n]$, it is the case that for every player i, s_i is optimal for every combination t_{-i} by the other players:

$$\forall i \;\; \forall t_{-i} \;\; \forall s_i' \;\; [s_i, t_{-i}] \succsim [s_i', t_{-i}] \; .$$

In a Nash equilibrium, we simply require that s_i is optimal for the particular current combination s_{-i} by the other players:

$$\forall i \;\; \forall s_i' \;\; [s_i, s_{-i}] \succsim [s_i', s_{-i}] \; .$$

Therefore, dominant strategy equilibrium is a special case of Nash equilibrium. The converse does not hold, as we can show simply by pointing to the CD/DVD game, where neither of the Nash equilibria is a dominant strategy equilibrium.

给定此策略，与从状态s开始的真正最优策略相比，策略损失仅为u π -s)-U π(s)。

17.13 信念状态更新由等式(17.11)给出，即，

$$b'(s') = \alpha\, P(e \mid s') \sum_s P(s' \mid s, a) b(s) \; .$$

分两个阶段计算可能会有所帮助：更新动作，然后更新观察。 观察概率P（e|s'）都是0.9（对于实际有一面墙的正方形）或0.1（对于有两面墙的正方形）。 下表显示了结果。 特别注意概率质量如何集中在（3,2）上。

		Left	1 wall	*Left*	1 wall
(1,1)	.11111	.20000	.06569	.09197	.02090
(1,2)	.11111	.11111	.03650	.04234	.00962
(1,3)	.11111	.20000	.06569	.09197	.02090
(2,1)	.11111	.11111	.03650	.27007	.06136
(2,3)	.11111	.11111	.03650	.05985	.01360
(3,1)	.11111	.11111	.32847	.06861	.14030
(3,2)	.11111	.11111	.32847	.30219	.61791
(3,3)	.11111	.02222	.06569	.03942	.08060
(4,1)	.11111	.01111	.00365	.00036	.00008
(4,2)	0	.01111	.03285	.03321	.06791
(4,3)	0	0	0	0	0

17.14在无传感器环境中，POMDP值迭代本质上与ordi nary状态空间搜索相同-分支仅在acton选择上，而不是观测值上。 因此时间复杂度为O(|A/d)。

17.15双态MDP的策略都有一个阈值信念p，这样如果b(0)>p，那么最佳动作是Go，否则是停留。 问题是，这种变化对阈值做了什么？ 通过使感测在状态0中更具信息性而在状态1中更少具有信息性，所述改变已经使状态0更可取，因此阈值p增加。

17.16这个问题很简单，就是检查定义。 在占优势的斯特拉特均衡[s1,。 ..，s n]，情况是对于每个玩家i，s i对于其他玩家的每个组合t-i都是最优的:

$$\forall i \;\; \forall t_{-i} \;\; \forall s_i' \;\; [s_i, t_{-i}] \succ [s_i', t_{-i}] \; .$$

在纳什均衡中，我们只需要s i对于其他玩家的特定当前组合s-i是最优的:

$$\forall i \;\; \forall s_i' \;\; [s_i, s_{-i}] \succ [s_i', s_{-i}] \; .$$

因此，主导策略均衡是纳什均衡的一个特例。 相反是不成立的，因为我们可以简单地通过指向CD/DVD游戏来证明，其中纳什均衡都不是主导策略均衡。

17.17 In the following table, the rows are labelled by A's move and the columns by B's move, and the table entries list the payoffs to A and B respectively.

	R	P	S	F	W
R	0,0	–1,1	1,-1	–1,1	1,-1
P	1,-1	0,0	–1,1	–1,1	1,-1
S	–1,1	1,-1	0,0	–1,1	1,-1
F	1,-1	1,-1	1,-1	0,0	–1,1
W	–1,1	–1,1	–1,1	1,-1	0,0

Suppose A chooses a mixed strategy $[r : R;\ p : P;\ s : S;\ f : F;\ w : W]$, where $r + p + s + f + w = 1$. The payoff *to A* of B's possible pure responses are as follows:

$$\begin{aligned} R &: +p - s + f - w \\ P &: -r + s + f - w \\ S &: +r - p + f - w \\ F &: -r - p - s + w \\ W &: +r + p + s - f \end{aligned}$$

It is easy to see that no option is dominated over the whole region. Solving for the intersection of the hyperplanes, we find $r = p = s = 1/9$ and $f = w = 1/3$. By symmetry, we will find the same solution when B chooses a mixed strategy first.

17.18 We apply iterated strict dominance to find the pure strategy. First, *Pol: do nothing* dominates *Pol: contract*, so we drop the *Pol: contract* row. Next, *Fed: contract* dominates *Fed: do nothing* and *Fed: expand* on the remaining rows, so we drop those columns. Finally, *Pol: expand* dominates *Pol: do nothing* on the one remaining column. Hence the only Nash equilibrium is a dominant strategy equilibrium with *Pol: expand* and *Fed: contract*. This is not Pareto optimal: it is worse for both players than the four strategy profiles in the top right quadrant.

17.19 This question really has two answers, depending on what assumption is made about the probability distribution over bidder's private valuations v_i for the item.

In a Dutch auction, just as in a first-price sealed-bid auction, bidders must estimate the likely private values of the other bidders. When the price is higher than v_i, agent i will not bid, but as soon as the price reaches v_i, he faces a dilemma: bid now and win the item at a higher price than necessary, or wait and risk losing the item to another bidder. In the standard models of auction theory, each bidder, in addition to a private value v_i, has a probability density $p_i(v_1, \ldots, v_n)$ over the private values of all n bidders for the item. In particular, we consider **independent private values**, so that the distribution over the other bidders' values is independent of v_i. Each bidder will choose a bid—i.e., the first price at which they will bid if that price is reached—through a bidding function $b_i(v_i)$.

INDEPENDENT PRIVATE VALUES

We are interested in finding a Nash equilibrium (technically a **Bayes–Nash equilibrium** in which each bidder's bidding function is optimal given the bidding functions of the other agents. Under risk-neutrality, optimality of a bid b is determined by the expected payoff, i.e., the probability of winning the auction with bid b times the profit when paying that amount

17.17在下表中，行由A的移动标记，列由B的移动标记，表项分别列出对A和B的回报。

	R	P	S	F	W
R	0,0	–1,1	1,-1	–1,1	1,-1
P	1,-1	0,0	–1,1	–1,1	1,-1
S	–1,1	1,-1	0,0	–1,1	1,-1
F	1,-1	1,-1	1,-1	0,0	–1,1
W	–1,1	–1,1	–1,1	1,-1	0,0

假设A选择混合策略[r:R;p:P;s:S;f:F;w:W]，其中r+p+s+f+w=1。 B的可能纯响应对A的回报如下:

$$\begin{aligned} R &: +p - s + f - w \\ P &: -r + s + f - w \\ S &: +r - p + f - w \\ F &: -r - p - s + w \\ W &: +r + p + s - f \end{aligned}$$

很容易看出，没有任何选择在整个地区占主导地位。 求解超平面的交点，我们发现r=p=s=1/9和f=w=1/3。 通过对称性，我们将在B首先选择混合策略时找到相同的解决方案。

17.18我们应用迭代的严格支配来寻找纯粹的策略。 首先，pol:do nothing支配Pol:contract，所以我们删除Pol:contract行。 接下来，fed：contract支配Fed：什么都不做Fed：扩展剩余的行，所以我们删除这些列。 最后，pol:expand支配pol:在剩下的一列上什么都不做。 因此，唯一的纳什均衡是具有Pol：expand和Fed：contract的主导策略均衡。 这不是帕累托最佳：这对两个玩家来说都比右上象限的四个策略配置文件更糟糕。

17.19这个问题实际上有两个答案,这取决于对投标人的私人估价v i的概率分布作了什么假设。
在荷兰式拍卖中，就像在第一价密封投标拍卖中一样，投标人必须估计其他投标人可能的私人价值。 当价格高于v i时，代理i不会出价，但一旦价格达到v i，他就面临两难境地：现在出价并以比必要更高的价格赢得物品，或者等待并冒险将物品丢失给另 在拍卖理论的标准模型中，每个投标人除了一个私有值v i外，还有一个概率密度p i(v1,。 ..，v n）超过该项目的所有n个投标人的私人价值。 特别是，我们考虑独立的私人价值，以便在其他投标人的价值上分配

INDEPENDENT 私人价值观

独立于v i。 每个投标人将通过投标函数b i（v i）选择一个投标—即，如果达到该价格，他们将投标的第一个价格。
我们有兴趣找到一个纳什均衡（技术上是贝叶斯-纳什均衡，其中每个投标人的投标函数是最优的，考虑到其他代理的投标函数。 在风险中立下，出价b的最优性由预期回报决定，即以出价b赢得拍卖的概率是支付该金额时利润的两倍

for the item. Now, agent i wins the auction with bid b if all the other bids are less than b; let the probability of this happening be $W_i(b)$ for whatever fixed bidding functions the other bidders use. ($W_i(b)$ is thus a cumulative probability distribution and nondecreasing in b; under independent private values, it does not depend on v_i.) Then we can write the expected payoff for agent i as

$$Q_i(v_i, b) = W_i(b)(v_i - b)$$

and the optimality condition in equilibrium is therefore

$$\forall\, i, b \;\; W_i(b_i(v_i))(v_i - b_i(v_i)) \geq W_i(b)(v_i - b)\;. \tag{17.1}$$

We now prove that the bidding functions $b_i(v_i)$ must be **monotonic**, i.e., *nondecreasing* in the private valuation v_i. Let v and v' be two different valuations, with $b = b_i(v)$ and $b' = b_i(v')$. Applying Equation (17.1) twice, first to say that (v, b) is better than (v, b') and then to say that (v', b') is better than (v', b), we obtain

$$\begin{aligned} W_i(b)(v - b) &\geq W_i(b')(v - b') \\ W_i(b')(v' - b') &\geq W_i(b)(v' - b) \end{aligned}$$

Rearranging, these become

$$\begin{aligned} v(W_i(b) - W_i(b')) &\geq W_i(b)b - W_i(b')b' \\ v'(W_i(b') - W_i(b)) &\geq W_i(b')b' - W_i(b)b \end{aligned}$$

Adding these equations, we have

$$(v' - v)(W_i(b') - W_i(b)) \geq 0$$

from which it follows that if $v' > v$, then $W_i(b') \geq W_i(b)$. Monotonicity does not follow immediately, however; we have to handle two cases:

- If $W_i(b') > W_i(b)$, or if W_i is strictly increasing, then $b' \geq b$ and $b_i(\cdot)$ is monotonic.
- Otherwise, $W_i(b') = W_i(b)$ and W_i is flat between b and b'. Now if W_i is flat in any interval $[x, y]$, then an optimal bidding function will prefer x over any other bid in the interval since that maximizes the profit on winning without affecting the probability of winning; hence, we must have $b' = b$ and again $b_i(\cdot)$ is monotonic.

Intuitively, the proof amounts to the following: if a higher valuation could result in a lower bid, then by swapping the two bids the agent could increase the *sum* of the payoffs for the two bids, which means that *at least one* of the two original bids is suboptimal.

Returning to the question of efficiency—the property that the item goes to the bidder with the highest valuation—we see that it follows immediately from monotonicity in the case where the bidders' prior distributions over valuations are **symmetric** or identically distributed.[1]

[1] According to Milgrom (1989), Vickrey (1961) proved that under this assumption, the Dutch auction is efficient. Vickrey's argument in Appendix III for the monotonicity of the bidding function is similar to the argument above but, as written, seems to apply only to the uniform-distribution case he was considering. Indeed, much of his analysis beginning with Appendix II is based on an inverse bidding function, which implicitly *assumes* monotonicity of the bidding function. Many other authors also begin by assuming monotonicity, then derive the form of the optimal bidding function, and then show it is monotonic. This proves the existence of an equilibrium with monotonic bidding functions, but not that all equilibria have this property.

为该项目。 现在，如果所有其他出价都小于b，代理i以b出价赢得拍卖;让这种情况发生的概率为W i（b）对于其他投标人使用的任何固定出价功能。 （W i（b）因此是b中的累积概率分布和非累积;在独立的私有值下，它不依赖于v i。）然后我们可以将代理i的预期回报写为

$$Q_i(v_i, b) = W_i(b)(v_i - b)$$

因此，平衡中的最优条件是

$$\forall\, i, b \;\; W_i(b_i(v_i))(v_i - b_i(v_i)) \geq W_i(b)(v_i - b)\;. \tag{17.1}$$

我们现在证明投标函数b i（v i）必须是单调的，即在私人估值v i中不重复。 设v和v'是两个不同的估值，b=b i（v）和b'=b i（v'）。 应用方程（17.1）两次，首先说（v，b）优于（v，b'），然后说（v'，b'）优于（v'，b），我们得到

$$\begin{aligned} W_i(b)(v - b) &\geq W_i(b')(v - b') \\ W_i(b')(v' - b') &\geq W_i(b)(v' - b) \end{aligned}$$

重新排列，这些成为

$$\begin{aligned} v(W_i(b) - W_i(b')) &\geq W_i(b)b - W_i(b')b' \\ v'(W_i(b') - W_i(b)) &\geq W_i(b')b' - W_i(b)b \end{aligned}$$

添加这些方程，我们有

$$(v' \quad v)(W_i(b') \quad W_i(b)) \quad 0$$

由此可知，如果v'>v，则W i(b')≥W i(b)。 然而，单调性不会立即跟随;我们必须处理两种情况:

•如果W i（b'）>W i（b），或者如果W i是严格增加的，那么b'≥b并且b i（·）是单调的。 •否则，W i（b'）=W i（b）并且W i在b和b'之间是平坦的。 现在，如果W i在任何区间[x，y]中是平坦的，那么一个最佳出价函数将更喜欢x而不是区间中的任何其他出价，因为这最大化了获胜的利润而不影响获胜的概率;因此，我们必须有b'=b并且再次b i（·）是单调的。

直觉上，证明相当于以下内容：如果较高的估值可能导致较低的出价，那么通过交换两个出价，代理人可以增加两个出价的回报总和，这意味着两个原始出价中至少有一个是次优的。

回到效率问题–该项目以最高估值交给投标人的财产–我们看到，在投标人对估值的先前分布对称或相同的情况下，它立即从单调性跟随。 1

1根据Milgrom（1989），Vickrey（1961）证明，在这个假设下，荷兰式拍卖是有效的。 Vickrey在附录III中关于投标函数单调性的论点类似于上面的论点，但正如所写的那样，似乎只适用于他正在考虑的均匀分布情况。 事实上，他从附录二开始的大部分分析都是基于逆投标函数，它隐含地假定投标函数的单调性。 许多其他作者也从假设单调性开始，然后推导出最优出价函数的形式，然后表明它是单调的。 这证明了具有单调竞价函数的均衡的存在，但不是所有均衡都具有这种性质。

Vickrey (1961) proves that the auction is *not* efficient in the asymmetric case where one player's distribution is uniform over $[0, 1]$ and the other's is uniform over $[a, b]$ for $a > 0$. Milgrom (1989) provides another, more transparent example of inefficiency: Suppose Alice has a known, fixed value of \$101 for an item, while Bob's value is \$50 with probability 0.8 and \$75 with probability 0.2. Given that Bob will never bid higher than his valuation, Alice can see that a bid of \$51 will win *at least* 80% of the time, giving an expected profit of *at least* $0.8 \times (\$101 - \$51) = \$40$. On the other hand, any bid of \$62 or more cannot yield an expected profit at most \$39, regardless of Bob's bid, and so is dominated by the bid of \$51. Hence, in any equilibrium, Alice's bid at most \$61. Knowing this, Bob can bid \$62 whenever his valuation is \$75 and be sure of winning. Thus, with 20% probability, the item goes to Bob, whose valuation for it is lower than Alice's. This violates efficiency.

Besides efficiency in the symmetric case, monotonicity has another important consequence for the analysis of the Dutch (and first-price) auction : it makes it possible to derive the exact form of the bidding function. As it stands, Equation (17.1) is difficult or impossible to solve because the cumulative distribution of the other bidders' bids, $W_i(b)$, depends on their bidding functions, so all the bidding functions are coupled together. (Note the similarity to the Bellman equations for an MDP.) With monotonicity, however, we can define W_i in terms of the known valuation distributions. Assuming independence and symmetry, and writing $b_i^{-1}(b)$ for the inverse of the (monotonic) bidding function, we have

$$Q_i(v_i, b) = (P(b_i^{-1}(b)))^{n-1}(v_i - b)$$

where $P(v)$ is the probability that an individual valuation is less than v. At equilibrium, b maximizes Q_i, the first derivative must be zero:

$$\frac{\partial Q}{\partial b} = 0 = \frac{(n-1)(P(b_i^{-1}(b)))^{n-2}p(b_i^{-1}(b))(v_i - b)}{b_i'(b_i^{-1}(b))} - (P(b_i^{-1}(b)))^{n-1}$$

where we have used the fact that $df^{-1}(x)/dx = 1/f'(f^{-1}(x))$.

For an equilibrium bidding function, of course, $b_i^{-1}(b) = v_i$; substituting this and simplifying, we find the following differential equation for b_i:

$$b_i'(v_i) = (v_i - b_i(v_i)) \cdot (n-1)p(v_i)/P(v_i) \ .$$

To find concrete solutions we also need to establish a boundary condition. Suppose v_0 is the lowest possible valuation for the item; then we must have $b_i(v_0) = v_0$ (Milgrom and Weber, 1982). Then the solution, as shown by McAfee and McMillan (1987), is

$$b_i(v_i) = v_i - \frac{\int_{v_0}^{v_i}(P(v))^{n-1}dv}{(P(v_i))^{n-1}} \ .$$

For example, suppose p is uniform in $[0, 1]$; then $P(v) = v$ and $b_i(v_i) = v_i \cdot (n-1)/n$, which is the classical result obtained by Vickrey (1961).

17.20 In such an auction it is rational to continue bidding as long as winning the item would yield a profit, i.e., one is willing to bid up to $2v_i$. The auction will end at $2v_o + d$, so the winner will pay $v_o + d/2$, slightly less than in the regular version.

Vickrey（1961）证明，在不对称的情况下，一个玩家的分布在[0,1]上是均匀的，而另一个玩家的分布在[a,b]上是均匀的，因为a>0。 Milgrom（1989）提供了另一个更透明的低效率示例：假设Alice对一个项目有一个已知的固定值$101，而Bob的值为probability50，概率为0.8，$75，概率为0.2。 鉴于Bob永远不会出价高于他的估值，Alice可以看到出价$51将赢得至少80%的时间，从而获得至少0的预期利润。 8 × ($101 - $51) = $40 .另一方面，无论Bob的出价如何，任何$62或更多的出价都不能产生最多$39的预期利润，因此以bid51的出价为主。 因此，在任何均衡情况下，爱丽丝的出价最多为61美元。 知道这一点，鲍勃可以出价62美元，只要他的估值是75美元，并确保获胜。 因此，以20%的概率，该项目转到Bob，其估值低于Alice的。这违反了效率。

除了对称情况下的效率之外，单调性对于分析荷兰（和第一价格）拍卖还有另一个重要的conse quence：它可以推导出价函数的确切形式。 就目前而言，方程（17.1）很难或不可能解决，因为其他投标人出价的累积分布W i（b）取决于他们的投标函数，因此所有投标函数都耦合在一起。 （请注意与Mdp的Bellman方程类似的ity。 然而，对于单调性，我们可以根据已知的估值分布来定义W i。 假设独立性和对称性，并写出b-1

i（b）对于（单调）出价函数的逆，我们有

$$Q_i(v_i, b) = (P(b_i^{-1}(b)))^{n-1}(v_i - b)$$

其中P(v)是个体估值小于v的概率。 在平衡时，其中b最大化Q i，一阶导数必须为零:

$$\frac{\partial Q}{\partial b} = 0 = \frac{(n-1)(P(b_i^{-1}(b)))^{n-2}p(b_i^{-1}(b))(v_i - b)}{b_i'(b_i^{-1}(b))} - (P(b_i^{-1}(b)))^{n-1}$$

其中我们使用了df-1（x）/dx=1/f'（f-1（x））的事实。
i（b）=v i;代入这个并模拟plifying，我们找到b i的以下微分方程：

$$b_i'(v_i) = (v_i - b_i(v_i)) \cdot (n-1)p(v_i)/P(v_i) \ .$$

为了找到具体的解决方案，我们还需要建立边界条件。 假设v0是该项目的最低可能估值;那么我们必须有b i（v0）=v0（Milgrom和Weber，1982）。 然后解决方案，如McAfee和McMillan（1987）所示，是

$$b_i(v_i) = v_i - \frac{\int_{v_0}^{v_i}(P(v))^{n-1}dv}{(P(v_i))^{n-1}} \ .$$

例如，假设p在[0,1]中是均匀的;那么P（v）=v和b i（v i）=v i·（n-1）/n，这是Vickrey（1961）获得的经典结果。

17.20在这种拍卖中，只要赢得该物品会产生利润，即愿意出价2v i，继续竞标是合理的。 拍卖将在2v o+d结束，因此获胜者将支付v o+d/2，略低于常规版本。

17.21 Every game is either a win for one side (and a loss for the other) or a tie. With 2 for a win, 1 for a tie, and 0 for a loss, 2 points are awarded for every game, so this is a constant-sum game.

If 1 point is awarded for a loss in overtime, then for some games 3 points are awarded in all. Therefore, the game is no longer constant-sum.

Suppose we assume that team A has probability r of winning in regular time and team B has probability s of winning in regular time (assuming normal play). Furthermore, assume team B has a probability q of winning in overtime (which occurs if there is a tie after regular time). Once overtime is reached (by any means), the expected utilities are as follows:

$$U_A^O = 1 + p$$
$$U_B^O = 1 + q$$

In normal play, the expected utilities are derived from the probability of winning plus the probability of tying times the expected utility of overtime play:

$$U_A = 2r + (1 - r - s)(1 + p)$$
$$U_B = 2s + (1 - r - s)(1 + q)$$

Hence A has an incentive to agree if $U_A^O > U_A$, or

$$1 + p > 2r + (1 - r - s)(1 + p) \quad \text{or} \quad rp - r + sp + s > 0 \quad \text{or} \quad p > \frac{r - s}{r + s}$$

and B has an incentive to agree if $U_B^O > U_B$, or

$$1 + q > 2s + (1 - r - s)(1 + q) \quad \text{or} \quad sq - s + rq + r > 0 \quad \text{or} \quad q > \frac{s - r}{r + s}$$

When both of these inequalities hold, there is an incentive to tie in regulation play. For any values of r and s, there will be values of p and q such that both inequalities hold.

For an in-depth statistical analysis of the actual effects of the rule change and a more sophisticated treatment of the utility functions, see "Overtime! Rules and Incentives in the National Hockey League" by Stephen T. Easton and Duane W. Rockerbie, available at `http://people.uleth.ca/~rockerbie/OVERTIME.PDF`.

171

17.21每场比赛要么是一方获胜（另一方则是输球），要么是平局。 赢了2分，平了1分，输了0分，每场比赛都能得到2分，所以这是一个恒和游戏。

如果在加时赛中输掉1分，那么在一些比赛中，总共获得3分。 因此，游戏不再是constant-sum。

假设我们假设团队A在常规时间和团队中获胜的概率r
B在常规时间内获胜的概率s（假设正常游戏）。 此外，假设b队在加时赛中获胜的概率为q（如果在常规时间之后出现平局，则会发生这种情况）。 一旦达到加班（通过任何方式），预期的实用程序如下:

$$U_A^O = 1 + p$$
$$U_B^O = 1 + q$$

在正常比赛中，预期效用是从获胜的概率加上加时赛预期效用的倍的概率得出的:

$$U_A = 2r + (1 - r - s)(1 + p)$$
$$U_B = 2s + (1 \quad r \quad s)(1 + q)$$

因此A有动机同意如果U O $> U_A$, or

$$1 + p > 2r + (1 - r - s)(1 + p) \quad \text{or} \quad rp - r + sp + s > 0 \quad \text{or} \quad p > \frac{r - s}{r + s}$$

和B有动机同意，如果U O $> U_B$, or

$$\frac{s \quad r}{}$$

r+s当这两种不等式都成立时，就会有动力在监管方面发挥作用。 对于r和s的任何值，都会有p和q的值，使得两个不等式都成立。

有关规则更改的实际影响的深入统计分析以及效用函数的更复杂处理，请参阅"加班！ 国家曲棍球联盟的规则和奖励"由Stephen T.Easton和Duane W.Rockerbie提供，可在http://people.uleth.ca/rockerbie/OVERTIME.PDF ...