Discussion Paper Series

RIEB

Kobe University

DP2016-23

41 Counterexamples to Property (B) of the Discrete Time Bomber Problem

Takashi KAMIHIGASHI

June 16, 2016

Research Institute for Economics and Business Administration

Kobe University

2-1 Rokkodai, Nada, Kobe 657-8501 JAPAN

41 Counterexamples to Property (B) of the Discrete Time Bomber Problem*

Takashi Kamihigashi†

June 16, 2016

Abstract

The discrete time "bomber problem" has been one of the longest standing open problems in operations research. In particular, the validity of one of the natural monotonicity conjectures—known as property (B)—has been an unresolved issue since 1968. In this paper we report 41 counterexamples to property (B) of this problem. We have found them by computing the exact solutions for nearly one million pairs of parameter values utilizing the GNU Multiple Precision (GMP) Arithmetic Library. All our counterexamples can readily be verified using a simple Mathematica program included in this paper.

Keywords: The discrete time bomber problem; error-free methods; GNU Multiple Precision (GMP) Arithmetic Library; stochastic dynamic programming

2010 Mathematics Subject Classification: 62L05, 93E20

*I would like to thank Çağrı Sağlam for bringing the bomber problem to my attention, and Professor Richard Weber (whom I have never met) for the inspiring expositions of this problem in Weber (2013) and his home page.

†Research Institute for Economics and Business Administration, Kobe University, Rokkodai, Nada, Kobe 657-8501 Japan. Email: tkamihig@rieb.kobe-u.ac.jp.

1 Introduction

At Professor Richard Weber's home page,[1] the discrete time bomber problem appears at the top of his list of unsolved problems in operations research. In this problem, a bomber with $n \in \mathbb{N}$ anti-aircraft missiles must survive $t \in \mathbb{N}$ hours before reaching its destination. In each hour, it encounters an enemy plane with probability r. The bomber survives for sure if it encounters no enemy plane. In the event of encountering an enemy plane, it survives with probability $1 - q^k$ if it fires k missiles at the enemy plane. The objective is to maximize the probability of reaching the destination.

This problem can easily be solved numerically by dynamic programming, or backward induction. For this purpose, let $N \in \mathbb{N}$ and $T \in \mathbb{N}$ be the largest numbers of missiles n and hours t to be considered. Define

$$p(n, 0) = 1, \quad \forall n \in \{0, \ldots, N\}. \tag{1}$$

Let $p(n, t)$ be the optimal survival probability when the bomber has n missiles with t hours to go. Then for all $n = 0, \ldots, N$ and $t = 1, \ldots T$, $p(n, t)$ satisfies

$$p(n, t) = (1 - r)p(n, t - 1) + rv(n, k), \tag{2}$$

where

$$v(n, t) = \max_{k \in \{0, \ldots, n\}} (1 - q^k)p(n - k, t - 1). \tag{3}$$

Let $k(n, t)$ be the smallest solution k of the above maximization problem:

$$k(n, t) = \min \operatorname*{argmax}_{k \in \{0, \ldots, n\}} (1 - q^k)p(n - k, t - 1). \tag{4}$$

The following three monotonicity properties have been extensively studied in the literature:

(A) $k(n, t)$ is nonincreasing in t.

(B) $k(n, t)$ is nondecreasing in n.

(C) $n - k(n, t)$ is nondecreasing in n.

[1] http://www.statslab.cam.ac.uk/~rrw1/

The above problem was originally formulated in continuous time by Klinger and Brown (1968), who proved property (C) for the original continuous time model. They proved (A) assuming (B), and left (B) as an unsolved problem:

> It seems intuitively obvious that $k(n,t) \geq k(n-1,t)$; that is, with a larger supply one is always willing to make at least as generous an allocation. The extensive tables we computed have confirmed this conjecture. However, determined efforts by a number of people at RAND have failed to yield a rigorous proof that this is indeed the case. (Klinger and Brown, 1968, p. 182, Ψ instead of k in the original)

Subsequently, Samuel (1970) proved (A) without assuming (B), but "found no proof of (B)." Simons and Yao (1990) formulated the problem in discrete time, proving (A) and (C) for the discrete time case (Simons and Yao, 1990, Lemma 1, Corollary 1). They noted that a proof of (B) was "elusive," but their numerical work supported the validity of (B):

> Already, we have numerically 'verified' Conjecture B for tens of thousands of randomly generated pairs (q,r). Mostly, these were checked for $t \leq 12$ and $n \leq 20$, but some larger values of t and n were checked when q is not too small. The truth of Conjecture B was always supported, except for a very few instances when unavoidable difficulties with round-off errors were clearly indicated, because of an extreme value of q or r.

Weber (2013, p. 199) also noted that no counterexample to (B) had been found "despite a truly enormous amount of computational experimentation." He summarized the status of (B) as follows:

> *Open problem for the bomber* Despite 40 years of research, it is still not known if (B) is true for the bomber problem. So far as I know, the best we can say about (B) is that $k(n+1,t) \geq k(n,t)$ if either $n \leq 3$ or $t \leq 3$, and also that $k(n,t) = 1 \Rightarrow k(n-1,t) = 1$ for all t. (Weber, 2013, p. 192, italics in the original)

In this paper we close this open problem by reporting 41 counterexamples to (B). In the next section, we briefly discuss "unavoidable difficulties with round-off errors" associated with floating point numbers. In Section

3 we introduce an error-free algorithm consisting only of integer addition, subtraction, multiplication, and comparison. Implementing this algorithm in C with the GNU Multiple Precision (GMP) Arithmetic Library to solve the problem for *all* $q, r \in \{0.001, 0.002, \ldots, 0.999\}$, we have found 41 counterexamples to property (B). We have also obtained the identical results by solving the problem with rational numbers for the same set of (q, r) values using the GMP library. All our counterexamples can readily be verified using a simple Mathematica program provided in this paper. In Section 4 we discuss the robustness of our examples.

In closing the introduction, we should mention that various problems related to the bomber problem are still actively studied (e.g., Bartroff et al., 2010; Bartroff and Samuel-Cahn, 2011; Elguedria et al., 2013; Krieger and Samuel-Cahn, 2013). We refer the reader to Weber (2013) for an excellent survey of the literature surrounding the bomber problem.

2 Difficulties

Algorithm 1 shows pseudocode for the dynamic programming procedure specified by (1)–(4). Throughout the paper we fix N and T as follows:

$$N = T = 100. \tag{5}$$

We have implemented Algorithm 1 in C with 64 bit "long double" precision for all

$$q, r \in \{0.01, 0.02, \ldots, 0.99\}. \tag{6}$$

In the solutions obtained, there are many numerical violations of properties (A), (B), and (C) even though (A) and (C) are known to be true. More specifically, there are 25,802 quadruples (q, r, n, t) violating (A), 29,584 quadruples violating (B), and 2,381 quadruples violating (C). These numbers are unstable, depending on the system and software used to implement the algorithm.

Figure 1 shows an example of a numerical solution that violates both (A) and (B). One can see in panel (b) that there are many violations of both properties. Precisely, both are violated at $(n, t) = (85, 83), (90, 88), (92, 89), (94, 92), (96, 93), (99, 94), (98, 95), (99, 97)$.

In fact, even a very elementary property of $k(n, t)$ is violated in Figure 1. To see this, note from (1) and (3) that

$$k(n, 1) = n, \quad \forall n \in \{0, \ldots, N\}. \tag{7}$$

Algorithm 1: Pseudocode for the bomber problem

```
1  for n = 0, ..., N
2      p(n, 0) = 1
3  for t = 1, ..., T
4      for n = 0, ..., N
5          v* = 0, k* = 0
6          for k = 1, ..., n
7              v = (1 − q^k) p(n − k, t − 1)
8              if v > v* then
9                  v* = v, k* = k
10         p(n, t) = (1 − r) p(n, t − 1) + r v*
11         k(n, t) = k*
   output: k
```

This simply means that if the bomber encounters an enemy plane in the last hour, it should fire all the avaiable missiles. This obvious property is clearly violated in panel (a). In this example, we have $\max_{n,t\in\{1,\ldots,100\}} k(n,t) = 9$ even though (7) requires that $k(100,1) = 100$. This is because $1 - 0.01^k$ is rounded to 1 for all $k \geq 9$ in C with long double precision, which implies that the strict inequality in line 8 of Algorithm 1 is never satisfied for any $k > 9$.

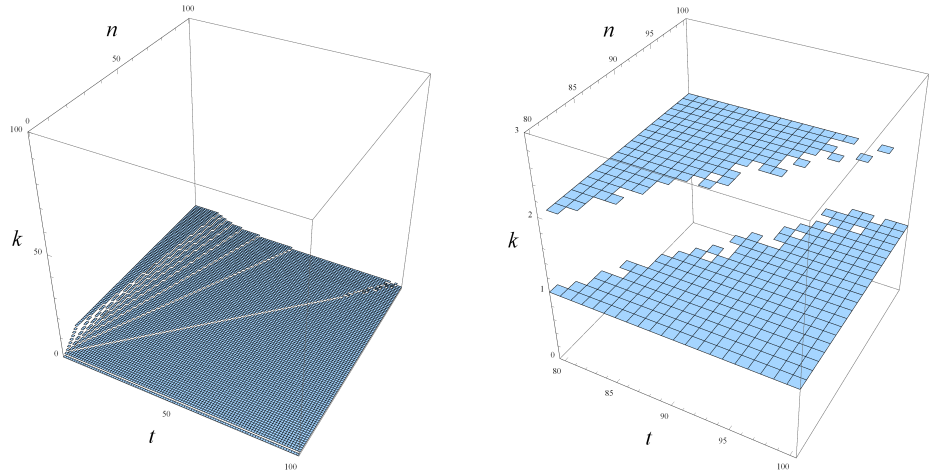
3 Error-Free Methods

Numerical errors are unavoidable as long as floating point numbers are used. However, there are several ways to implement Algorithm 1 without introducing numerical errors. For example, it is possible to compute $k(n,t)$ by using only integers, provided that both q and r are rational numbers. To be more specific, suppose that there are integers $Q, R, B \in \mathbb{N}$ such that

$$q = \frac{Q}{B}, \quad r = \frac{R}{B}. \tag{8}$$

As in the original problem, define

$$P(n,0) = 1, \quad \forall k \in \{0,\ldots,N\}. \tag{9}$$



(a) $n, t \in \{0, 1, \ldots, 100\}$ (b) $n, t \in \{80, \ldots, 100\}$

Figure 1: $k(n,t)$ for $(q,r) = (0.01, 0.88)$ computed with long double precision

For $n \in \mathbb{Z}_+$ and $t \in \mathbb{N}$, define $P(n,t)$ recursively as follows:

$$P(n,t) = B^N(B-R)P(n,t-1) + B^{N-n}RV(n,k), \tag{10}$$

where

$$V(n,t) = \max_{k\in\{0,\ldots,n\}}(B^n - B^{n-k}Q^k)P(n-k,t-1). \tag{11}$$

Equation (10) can be obtained by multiplying both sides of (2) by $B^{(N+1)t}$. Thus $P(n,t)$ and $p(n,t)$ satisfy $P(n,t) = B^{(N+1)t}p(n,t)$. Note that

$$(B^n - B^{n-k}Q^k)P(n-k,t-1) \tag{12}$$

$$= B^nB^{(N+1)(t-1)}(1-q^k)p(n-k,t-1). \tag{13}$$

Hence the solution of (11) is identical to that of (3).

A useful feature of this equivalent formulation is that as long as $P(n,t-1)$ is an integer for each $n = 0, \ldots, N$, so is $P(n,t)$.

Algorithm 2 shows pseudocode for the procedure given by (9)–(11). All variables remain integers throughout the algorithm; the problem is that they can be extremely large. Fortunately, arbitrarily large integers can be handled using the GMP library. Figure 2 shows the exact optimal policy computed

Algorithm 2: An error-free algorithm for the bomber problem

```
for n = 0, ..., N
    P(n, 0) = 1
for t = 1, ..., T
    for n = 0, ..., N
        V* = 0, k* = 0
        for k = 1, ..., n
            V = (B^n − B^{n−k} Q^k) P(n − k, t − 1)
            if V > V* then
                V* = V, k* = k
        P(n, t) = B^N (B − R) P(n, t − 1) + B^{N−n} R V*
        k(n, t) = k*
output: k
```

by implementing Algorithm 2 in C with this library. This policy corresponds to that in Figure 1. In sharp contrast to Figure 1, panel (b) in Figure 2 shows that both (A) and (B) are clearly satisfied; panel (a) shows that (7) is also satisfied.

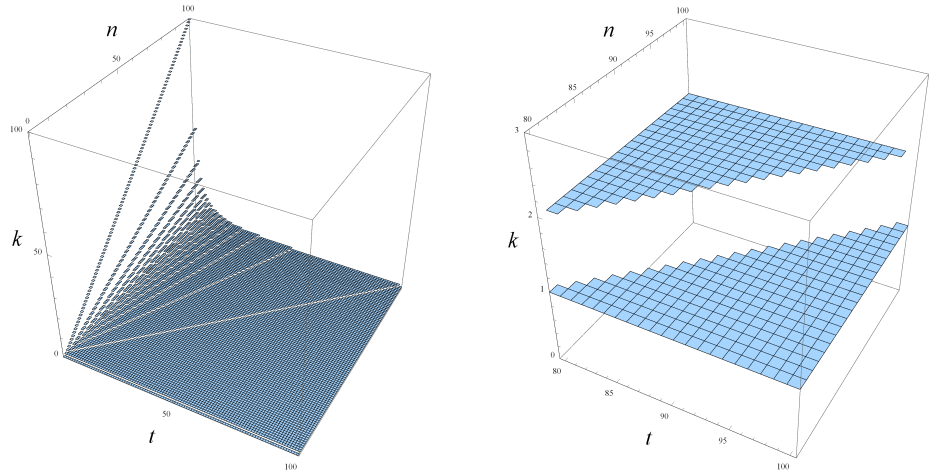
To investigate the validity of (B), we have implemented Algorithm 2 in the same way for all (q, r) given by (6) ($B = 100$ and $Q, R \in \{1, \ldots, 99\}$). In stark contrast to the results obtained with long double precision mentioned in the previous section, we found no violation of any of properties (A), (B), and (C) for any (q, r) given by (6) and $n, t \in \{1, \ldots, 100\}$.[2]

To further investigate the validity of (B), we have tested (A), (B), and (C) for all

$$q, r \in \{0.001, 0.002, \ldots, 0.999\} \tag{14}$$

($B = 1000$ and $Q, R \in \{1, \ldots, 999\}$). In the solutions obtained, there is no violation of (A) or (C), which is consistent with the theoretical results mentioned in the introduction. However, there are 41 violations of (B). All of them are reported in Table 1, which shows all the quadruples (Q, R, n, t) for which $k(n, t) < k(n - 1, t)$. These k values are also reported in the table.

[2]Our actual C code does not return the entire policy; it temporarily stores sufficient data in memory to check (A), (B), and (C) as soon as $k(n, t)$ is determined for each (n, t).



(a) $t, k \in \{0, 1, \ldots, 100\}$ (b) $t, k \in \{80, \ldots, 100\}$

Figure 2: $k(n, t)$ for $(q, r) = (0.01, 0.88)$ computed error-free with Algorithm 2

Figure 3 shows combinations of parameter values for which (B) is violated. Note from this figure and Table 3 that all the (q, r) pairs lie in the region $[0.4, 1] \times [0.8, 1]$, and that the values of t are restricted to 4, 5, and 6. Since the smallest values of n, t, and $k(n, t)$ in Table 1 are 31, 6, and 12, respectively, our counterexamples are consistent with Weber's (2013) results quoted in the introduction.

Observe that for each (Q, R) in Table 1, there is exactly one violation of (B). Hence a violation of (B) is an exception even for the (Q, R) pairs in the table, for each of which there are $100^2 - 1$ pairs of (n, t) values satisfying (B). It is worth noting that there are *only* 41 violations of (B) out of $999^2 \times 100^2$ quadruples of (Q, R, n, t) values. It took approximately 33 hours to test all the quadruples against (A), (B), and (C) using Algorithm 2 on a dedicated Linux workstation with dual Intel Xeon E5-2699v3 2.30 Hz CPUs (72 threads in total).

Since the GMP library allows one to handle arbitrarily large rational numbers without numerical errors in addition to integers, we have also implemented Algorithm 1 in C with this library for all (q, r) given by (14). The results were identical to those obtained from Algorithm 2. It took approximately 15 hours to test all the quadruples (Q, R, n, t) against (A), (B), and (C). Hence, at least in our case, it is considerably more efficient to let the

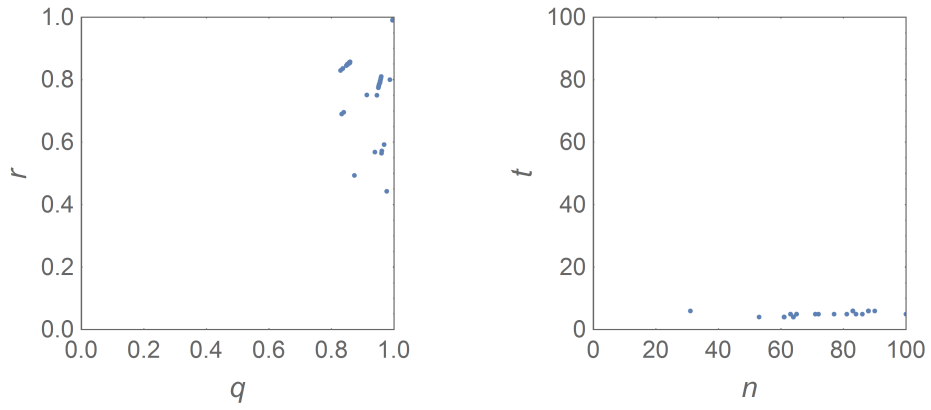


Figure 3: Configurations of parameter values for which property (B) is violated

GMP library directly handle rational numbers than to transform the problem so that all variables remain integers.

We have computed the optimal policies for all (Q, R) in Table (1) using the above two methods, which generated identical results. As an example, Table 2 shows the optimal policy $k(n, t)$ for Example #19, which has the smallest value of n in Table 1. One can see that (B) is indeed violated at $(n, t) = (31, 6)$.

So far we have discussed only our error-free C implementations of Algorithms 1 and 2. However, there are other ways to implement Algorithm 1 without numerical errors. An example is given by Algorithm 3, which shows a simple Mathematica program that generates the optimal policy in Table 2; the program is essentially identical to Algorithm 1. This Mathematica program is sufficiently efficient for verifying a relatively small number of examples. Using a modified version of this program, we have verified the optimal policies corresponding to all (Q, R) reported in Table 1. We have further confirmed all the optimal policies using Python as well. Thus for each (Q, R) in Table 1, we have cross-checked the optimal policy using the four different methods.

#	Q	R	n	t	$k(n,t)$	$k(n-1,t)$
1	829	830	88	6	16	17
2	833	690	77	5	18	19
3	835	835	88	6	16	17
4	836	836	88	6	16	17
5	839	696	81	5	19	20
6	847	845	88	6	16	17
7	850	849	83	6	15	16
8	851	848	88	6	16	17
9	851	850	83	6	15	16
10	852	849	88	6	16	17
11	853	850	88	6	16	17
12	856	854	83	6	15	16
13	857	855	83	6	15	16
14	858	854	88	6	16	17
15	858	856	83	6	15	16
16	859	857	83	6	15	16
17	873	494	72	5	21	22
18	913	751	86	5	21	22
19	939	568	31	6	12	13
20	945	750	53	4	18	19
21	950	774	64	4	21	22
22	951	778	64	4	21	22
23	951	779	61	4	20	21
24	952	782	64	4	21	22
25	953	786	61	4	20	21
26	954	789	64	4	21	22
27	954	790	61	4	20	21
28	955	793	64	4	21	22
29	955	794	61	4	20	21
30	956	798	61	4	20	21
31	957	802	61	4	20	21
32	958	806	61	4	20	21
33	959	810	61	4	20	21
34	960	565	63	5	25	26
35	961	572	63	5	25	26
36	968	592	100	5	36	37
37	977	443	90	6	41	42
38	987	800	71	5	28	29
39	995	990	65	5	15	16
40	996	992	65	5	15	16
41	999	998	84	5	19	20

Table 1: 41 counterexamples to property (B) with $B = 1000$

$n \setminus t$	1	2	3	4	5	6	7	8	9	10
1	1	1	1	1	1	1	1	1	1	1
2	2	2	2	2	2	2	2	2	2	2
3	3	3	3	3	3	3	2	2	2	2
4	4	4	4	4	3	3	3	3	3	3
5	5	5	5	4	4	4	3	3	3	3
6	6	6	6	5	4	4	4	4	4	4
7	7	7	6	5	5	5	4	4	4	4
8	8	8	7	6	5	5	5	5	4	4
9	9	9	7	6	6	6	5	5	5	5
10	10	9	8	7	6	6	6	5	5	5
11	11	10	8	7	7	6	6	6	5	5
12	12	10	9	8	7	7	6	6	6	5
13	13	11	9	8	8	7	7	6	6	6
14	14	11	10	9	8	7	7	7	6	6
15	15	12	10	9	8	8	7	7	7	6
16	16	12	11	10	9	8	8	7	7	7
17	17	13	11	10	9	8	8	8	7	7
18	18	13	12	11	10	9	8	8	7	7
19	19	14	12	11	10	9	9	8	8	7
20	20	14	13	11	10	9	9	8	8	8
21	21	15	13	12	10	10	9	9	8	8
22	22	15	14	12	11	10	9	9	8	8
23	23	16	14	12	11	10	10	9	9	8
24	24	16	15	13	11	11	10	9	9	8
25	25	17	15	13	12	11	10	10	9	9
26	26	17	15	13	12	11	11	10	9	9
27	27	18	16	14	12	12	11	10	10	9
28	28	18	16	14	13	12	11	10	10	9
29	29	19	16	14	13	12	11	11	10	10
30	30	19	17	15	13	13	11	11	10	10
31	31	20	17	15	14	12	12	11	10	10
32	32	20	17	15	14	13	12	11	11	10
33	33	21	18	16	14	13	12	12	11	10
34	34	21	18	16	15	13	13	12	11	11
35	35	22	19	16	15	14	13	12	11	11
36	36	22	19	17	15	14	13	12	11	11
37	37	23	20	17	16	14	13	12	12	11
38	38	23	20	17	16	15	13	13	12	11
39	39	24	20	18	16	15	14	13	12	11
40	40	24	21	18	16	15	14	13	12	12
41	41	25	21	18	17	15	14	13	13	12
42	42	25	21	19	17	15	14	13	13	12
43	43	26	22	19	17	16	15	14	13	12
44	44	26	22	19	17	16	15	14	13	12
45	45	27	22	20	18	16	15	14	13	13
46	46	27	23	20	18	16	15	14	13	13
47	47	28	23	20	18	17	15	14	14	13
48	48	28	23	21	18	17	16	15	14	13
49	49	29	24	21	19	17	16	15	14	13
50	50	29	24	21	19	17	16	15	14	13

$n \setminus t$	1	2	3	4	5	6	7	8	9	10
51	51	30	25	22	19	18	16	15	14	14
52	52	30	25	22	19	18	17	16	15	14
53	53	31	26	22	20	18	17	16	15	14
54	54	31	26	23	20	18	17	16	15	14
55	55	32	26	23	20	19	17	16	15	14
56	56	32	27	23	21	19	17	16	15	15
57	57	33	27	23	21	19	18	16	16	15
58	58	33	27	24	21	19	18	17	16	15
59	59	34	28	24	21	20	18	17	16	15
60	60	34	28	24	21	20	18	17	16	15
61	61	35	28	25	22	20	18	17	16	15
62	62	35	29	25	22	20	19	17	16	15
63	63	36	29	25	22	20	19	18	17	16
64	64	36	29	25	23	21	19	18	17	16
65	65	37	30	26	23	21	19	18	17	16
66	66	37	30	26	23	21	19	18	17	16
67	67	38	30	26	23	21	20	18	17	16
68	68	38	31	27	24	21	20	19	17	16
69	69	39	31	27	24	22	20	19	18	17
70	70	39	31	27	24	22	20	19	18	17
71	71	40	32	27	24	22	20	19	18	17
72	72	40	32	28	25	22	21	19	18	17
73	73	41	32	28	25	23	21	19	18	17
74	74	41	33	28	25	23	21	20	18	17
75	75	42	33	29	25	23	21	20	19	18
76	76	42	33	29	26	23	21	20	19	18
77	77	43	34	29	26	24	22	20	19	18
78	78	43	34	29	26	24	22	20	19	18
79	79	44	34	30	26	24	22	21	19	18
80	80	44	35	30	27	24	22	21	19	18
81	81	45	35	30	27	24	22	21	20	19
82	82	45	36	30	27	25	23	21	20	19
83	83	46	36	31	27	25	23	21	20	19
84	84	46	37	31	28	25	23	21	20	19
85	85	47	37	31	28	25	23	22	20	19
86	86	47	37	31	28	25	23	22	20	19
87	87	48	38	32	28	26	24	22	21	19
88	88	48	38	32	28	26	24	22	21	20
89	89	49	38	32	29	26	24	22	21	20
90	90	49	39	33	29	26	24	22	21	20
91	91	50	39	33	29	26	24	23	21	20
92	92	50	39	33	29	27	25	23	21	20
93	93	51	40	34	30	27	25	23	21	20
94	94	51	40	34	30	27	25	23	22	20
95	95	52	40	34	30	27	25	23	22	21
96	96	52	41	34	30	27	25	23	22	21
97	97	53	41	35	31	28	25	24	22	21
98	98	53	41	35	31	28	26	24	22	21
99	99	54	42	35	31	28	26	24	22	21
100	100	54	42	35	31	28	26	24	23	21

Table 2: $k(n,t)$ for Example #19 for $n = 1, \ldots, 100$ and $t = 1, \ldots, 10$

Algorithm 3: Mathematica code to generate $k(n,t)$ in Table 2

```
q = 939/1000;
r = 568/1000;
pnt = Table[1, {n, 0, 100}, {t, 0, 10}];
knt = Table[0, {n, 0, 100}, {t, 1, 10}];
For[t = 1, t <= 10, t++,
   For[n = 0, n <= 100, n++,
      v0 = 0; k0 = 0;
      For[k = 1, k <= n, k++,
         v = (1 - q^k)*pnt[[1+n-k]][[1+t-1]];
         If[v > v0,
            v0 = v; k0 = k;
         ];
      ];
      pnt[[1+n]][[1+t]] = (1 - r)*pnt[[1+n]][[1+t-1]] + r*v0;
      knt[[1+n]][[t]] = k0;
   ];
];
Print[Grid[knt]];
```

4 Robustness

To consider the robustness of our counterexamples, let $\overline{k}(n,t)$ be the largest solution k of the maximization problem in (3):

$$\overline{k}(n,t) = \max \operatorname*{argmax}_{k \in \{0,\ldots,n\}} (1-q^k)p(n-k,t-1). \tag{15}$$

This can be computed by replacing the strict inequality in line 8 of Algorithm 1 with the weak inequality $\geq$. With this modification, we have computed the optimal policies $\overline{k}(n,t)$ for all (Q,R) reported in Table 1. We have also cross-checked these solutions using the four methods discussed above. In the solutions, there are only two pairs (Q,R) such that $k(n,t) \neq \overline{k}(n,t)$ for some $n,t \in \{1,\ldots,100\}$. These (Q,R) pairs are given by Examples #3 and #4 in Table 1. For both cases we have $k(n,t) < \overline{k}(n,t)$ exactly for the (n,t) pairs reported in Table 3 (t is always 2). Since (B) is never violated for $t \leq 3$ in Table 1, it follows that the violations of (B) reported in Table 1 are shared by $\overline{k}(n,t)$.

An important implication of the above comparison is that for each (n,t) reported in Table 1, $k(n,t)$ and $k(n-1,t)$ are the unique solutions of the

n	2	4	6	8	10	12	14	16	18	20	22	24	26	28	30	32	34	36	38	40	42	44	46	48	50
t	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2

n	52	54	56	58	60	62	64	66	68	70	72	74	76	78	80	82	84	86	88	90	92	94	96	98	100
t	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2

Table 3: All pairs of (n,t) values for which $k(n,t) < \overline{k}(n,t)$ with $(Q,R) = (835,835),(836,836)$

corresponding maximization problems in (3) (replace n by $n-1$ in (3) for $k(n-1,t)$). Since $(1-q^k)p(n-k,t-1)$ is continuous in (q,r), it follows that $k(n,t)$ and $k(n-1,t)$ remain the unique solutions under small perturbations of (q,r). Thus the strict inequality $k(n,t) < k(n-1,t)$ is preserved under small perturbations of (q,r). This implies that there are in fact uncountably many counterexamples to property (B) of the bomber problem.

References

Bartroff, J., Goldstein, L, Rinott, Y., Samuel-Cahn, E., 2010, On optimal allocation of a continuous resource using an iterative approach and total positivity, Advances in Applied Probability 42, 795–815.

Bartroff, J., Samuel-Cahn, E., 2011, The fighter problem: optimal allocation of a discrete commodity, Advances in Applied Probability 43, 121–130.

Elguedria, Z., Boutheina, J., Ghédira, K., 2013, MAS-BPM: multi-agent system bomber problem model, in Advances on Practical Applications of Agents and Multi-Agent Systems, Demazeau, Y., Ishida, T., Corchado, J.M., Bajo, J., ed., Springer-Verlag, Berlin, pp. 61–72.

Klinger, A., Brown, T.A., 1968, Allocating unreliable units to random demands, in Stochastic Optimization and Ctonrol: Procedings of an Advances Seminar Conducted by the Mathematics Research Center and the United States Army at the University of Wisconsin, Madison, October 2–4, 1967, Karreman, H.F., ed., John Wiley & Sons, New York, pp. 173–209.

Krieger, A.M., Samuel-Cahn, E., 2013, Generalized bomber and fighter problems: offline optimal allocation of a discrete asset, Journal of Applied Probability 50, 403–418.

Samuel, E., 1970, On some problems in operations research, Journal of Applied Probability 7, 157–164.

Simons, G., Yao, Y-C., 1990, Some results on the bomber problem, Advances in Applied Probability 22, 412–432.

Weber, R., 2013, ABCs of the bomber problem and its relatives, Annals of Operations Research 208, 187–208.