Karl-Franzens-Universität Graz & Technische Universität Graz

SPEZIALFORSCHUNGSBEREICH F 003

OPTIMIERUNG

und

KONTROLLE

Projektbereich

DISKRETE OPTIMIERUNG

Günter Rote Guochuan Zhang

Optimal Logistics for Expeditions: the Jeep Problem with Complete Refilling

Bericht Nr. 71 — Juni 1996

Optimal Logistics for Expeditions — The Jeep Problem with Complete Refilling

Günter Rote Guochuan Zhang

June 24, 1996

Abstract

We consider a variant of the classical jeep problem. We have n cans of fuel on the edge of a desert and a jeep with an empty tank whose capacity is just one can. The jeep can carry one can in addition the fuel in its tank. Moreover, when a can is opened, the fuel must immediately be filled into the jeep's tank. The goal is to find the farthest point in the desert which the jeep can reach by consuming the n cans of fuel. Derick Wood [1984] treated this problem similarly to the classical problem and gave the first solution. Ute and Wilfried Brauer [1989] presented a new strategy and got a better solution than Wood's. They also conjectured that their solution was optimal for infinitely many values of n. We give an algorithm which produces a better solution than Brauers' for all $n > 6$, and we use a linear programming formulation to derive an upper bound which shows that our solution is optimal.

1 The jeep problem

A jeep starts at a depot in the desert with n cans of fuel and wants to reach a distant oasis. The jeep has a certain capacity for carrying fuel, and it can deposit fuel at intermediate depots along the way. This logistics problem has obvious applications to arctic expeditions, space travel, or military logistics. A version of this problem was posed more than 1000 years ago in the *propositiones ad acuendos iuvenes*, the oldest known mathematical puzzle collection in Latin, attributed to Alcuin of York (around 732–804). In the 52-nd problem, *propositio de homine patrefamilias* (a lord of the manor), a certain amount of grain is to be carried across a given distance by a camel which eats some of the grain on the way. The original Latin text with a German translation can be found in Gericke and Folkerts [1993, pp. 356–357], based on a critical edition of the text by Folkerts [1978, pp. 74–75], see also Folkerts [1993].

This research was supported by the Spezialforschungsbereich F 003, *Optimierung und Kontrolle*, Projektbereich Diskrete Optimierung

Authors' address: Institut für Mathematik, Technische Universität Graz, Steyrergasse 30, A-8010 Graz, Austria, e-mail: rote@opt.math.tu-graz.ac.at, zhang@opt.math.tu-graz.ac.at

There is an annotated English translation of Hadley and Singmaster [1992, p. 124–125], who note that the given solution is not the correct optimal solution. They also point out references to the literature where the problem appears in other guises.

The jeep problem was introduced in the modern mathematical literature by Fine [1947]. Since then, many variations of the problem have been proposed. Phipps [1947] considered the case of a group of jeeps. Suppose m jeeps fully loaded with fuel set out from a depot in the desert. The goal is to advance one of them to the greatest possible distance away from the depot. Phipps also mentioned several related variations.

The problem has continued to attract the attention of mathematicians and puzzle solvers alike, see Gale [1970, 1994] or Dewdney [1987]. Some recent references are Hausrath, Jackson, Mitchem, and Schmeichel [1995] and Jackson, Mitchem, and Schmeichel [1995].

The jeep problem with refilling of whole cans only. In the *classical* jeep problem, depots can be set up anywhere, and fuel can be freely exchanged between depots and the tank. We consider a variation with the following constraints.

1. Fuel can only be stored in cans (besides the jeep's tank).
2. In addition to the fuel in its tank, the jeep can carry one can of fuel.
3. No fuel is ever moved from the tank into a can.
4. The jeep can be refilled only when its tank is empty.
5. The jeep's tank is always refilled to its full capacity.

In other words, when a can is opened the whole content is immediately filled into the jeep's tank. Thus, in contrast to the classical problem, intermediate dumps can only contain an integral multiple of one canful of fuel. In the context of the jeep in the desert, these constraints appear natural. We might think of the fuel as stored in drums or barrels which cannot be sealed once they are opened. Fuel in open drums is wasted because it evaporates too fast in the desert.

We make the simplest additional assumptions:

6. All cans have the same size. The jeep's tank contains precisely *one* can of fuel.
7. The fuel consumption is independent of the load. Without loss of generality we define one unit of length as the distance which the jeep can travel with one tankful.
8. The jeep's tank is initially empty.

Under these constraints, we treat the following problem.

> With n cans of fuel at the starting point, what is the most distant point which the jeep can reach?

This problem was first stated in an exercise in a computer science textbook by Wood [1984, section 8.3, pp. 173–180]. Wood proposed an algorithm which is analogous to the optimal algorithm for the classical jeep problem. He declared that his algorithm yields the optimal distance. However, while the algorithm serves its purpose in illustrating the principles of recursion in algorithms and reduction of a given problem to a simpler problem, it is not optimal. Optimality of his algorithm must have appeared so obvious to Wood that he did not even mention that such an optimality claim ought to be proved, thus fostering among computer scientists a negligent attitude towards rigor and mathematical proof.

Brauer and Brauer [1989] were the first to note that Wood's algorithm is not optimal and proposed a better algorithm, which follows a greedy strategy. They conjectured that their algorithm is optimal for a certain infinite family of values of n.

We will give the optimal solution to the problem, which is better than Brauer and Brauer's algorithm for all $n > 6$. It is a modified greedy algorithm, and it will be described in Section 2. Section 3 is devoted to the optimality proof. In Section 3.1 we derive some conditions on a feasible tour. We introduce appropriate variables and derive conditions which any solution must necessarily fulfill. In Section 3.2, we model the problem as a linear programming problem. In Section 3.3 we derive an upper bound for the maximum distance from the inequalities of the linear program. Since this bound coincides with the value achieved by our algorithm, optimality is proved.

In Appendix B we show that any set of values satisfying the inequalities gives rise to a feasible solution of the jeep problem, thus showing that our model of Section 3.2 is complete.

Related work. A similar linear programming formulation, based on essentially the same ideas, was used by Jackson, Mitchem, and Schmeichel [1995] to solve another variant of the jeep problem proposed by Dewdney [1987]. We will discuss this problem and the relation to our solution in the concluding section 4.

Problem variations. Actually, Wood explicitly stated only constraint 4 but not 5. If one does not interpret the phrase "re*filling* the tank" in the strictest sense, one need not assume property 5.

Both Wood's and Brauer and Brauer's algorithms satisfy property 5. The omission of property 5 allows even better solutions, as we will show in the final section.

2 The optimal algorithm

For describing the algorithm we will need some terminology. The jeep moves on a line between the starting point and the destination. We place the origin 0 at the starting point, and we let the positive coordinate direction point towards the destination. As mentioned above, one unit is the distance which the jeep can travel with one tankful (one can) of gas.

The part of the trip from one filling of the tank to the next will be called a *move*. A move can be at most one unit long, and it is clear that every move must start at a depot where a nonempty can is available, and every move except the last must end at such a position. The structure of the algorithm is best described by combining two successive moves into one *double-move*. At the start and end of each *double-move*, the situation is always in a well-structured state:

1. There is a sequence of depots containing cans, and each depot contains an *even* number of full cans.

For this reason we may also group the full cans into pairs which we call *double-cans*. Let $e_i \le e_{i+1} \le \cdots \le e_j$ denote the current positions of full double-cans from left to right.

2. The jeep is positioned at the left-most full double-can e_i and the jeep's tank is empty.

3. The jeep carries no can.

4. The following invariant is maintained:

$$e_{k+1} \le e_k + 1/2 \text{ for } k = i, \ldots, j-1. \tag{1}$$

   This invariant ensures that the algorithm can continue.

If n is even these conditions are initially fulfilled, and we can set $i := 1$, $j := n/2$, and

$$e_1 = e_2 = \cdots = e_{n/2} = 0.$$

If n is odd we make one exceptional single move to satisfy the conditions: We transport two cans from 0 to $1/4$ and return to 0, using up fuel of one can. We set $i := 2$, $j := (n+1)/2$, and

$$e_1 = e_2 = \cdots = e_{(n-1)/2} = 0, e_{(n+1)/2} = \tfrac{1}{4}.$$

Now we describe the double-move. It is illustrated in Figure 1. We assume that $j > i+1$, that is, at least three double-cans are available.

1. The jeep fills the tank with one can of the double-can e_i and loads the other can.

2. The jeep brings this can forward to $f_{i+1/2} := (e_i + e_{i+1})/2 + 1/2$ and unloads it there.

3. The jeep moves back to position e_{i+1}. Now the jeep has traveled precisely one unit, and the first move is finished.

4. The second move is more complicated: The jeep fills the tank with one can of the double-can e_{i+1} and loads the other can.

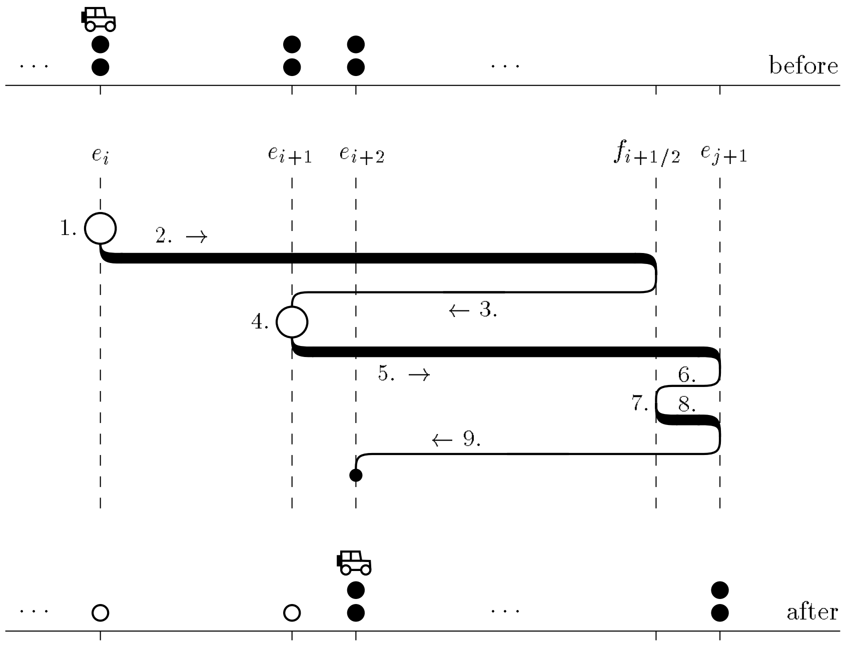

Figure 1: A double-move. The numbers indicate the operations as described in the text. The successive operations are shown from top to bottom. Whenever the jeep carries a full can the jeep's path is drawn as a thick line. (This occurs precisely when the jeep moves forward.) When the jeep fills the tank this is indicated by a circle, which symbolizes the empty can that is left behind.

5. The jeep brings this can forward to $e_{j+1} := (e_i + 2e_{i+1} + e_{i+2})/4 + 1/2$ and unloads it there.

6. The jeep moves back to position $f_{i+1/2}$.

7. The jeep loads the can that was deposited there in Step 2.

8. The jeep brings also this can forward to e_{j+1} and unloads it there.

9. The jeep moves back to position e_{i+2}. Now the jeep has traveled one more unit, and the second move is completed.

This double move has removed the double cans at e_i and e_{i+1}, and it has established a new double-can at e_{j+1}. The jeep is now positioned at the leftmost non-empty double-can e_{i+2}. Thus we can set $j := j+1$ and $i := i+2$, and the situation is ready for the next double-move. Note that the double-move leaves precisely two empty cans at positions e_i and e_{i+1}. To show that all required moves can be carried out in the described direction we must check that

$$\begin{aligned} e_i, e_{i+1} \leq f_{i+1/2}, \text{ and} \\ e_{i+1}, f_{i+1/2}, e_{i+2} \leq e_{j+1} \end{aligned} \tag{2}$$

holds for the new values of $f_{i+1/2}$ and e_{j+1} defined above. This follows easily from the invariant (1). It follows also that $e_{j+1} \leq e_j + 1/2$, and thus (1) is again fulfilled.

When $e_i = e_{i+1} = e_{i+2}$, we have $f_{i+1/2} = e_{j+1}$ and Steps 6–8 are void. All other inequalities in (2) are satisfied as strict inequalities.

For showing that all required moves are possible with the fuel of the two cans, it can be easily checked that

$$(f_{i+1/2} - e_i) + (f_{i+1/2} - e_{i+1}) = 1, \text{ and}$$
$$(e_{j+1} - e_{i+1}) + (e_{j+1} - f_{i+1/2}) + (e_{j+1} - f_{i+1/2}) + (e_{j+1} - e_{i+2}) = 1.$$

At the end we have to make a modified double-move and two exceptional last moves. Since the number of double-cans decreases at each step. we will eventually come to the situation when $j = i+1$, that is, only four cans are left. It will turn out that $i = n-3$ then. In this case we carry out a modified double move as follows. (A modified double-move appears in Figure 2 between e_{19} and e_{21}.) Steps 1–4 are as above, but the second move is modified. There is no point in returning to e_{i+2} at the end of the second move because e_{i+2} is not even defined yet. Therefore we make the following modified steps.

5′. The jeep brings the can which it has just loaded from e_{i+1} forward to $e_{j+1} \equiv e_{i+2} := (e_i + 2e_{i+1})/3 + 2/3$ and unloads it there.

6′. The jeep moves back to position $f_{i+1/2}$.

7′. The jeep loads the can that was deposited there in Step 2.

8′. The jeep brings also this can forward to e_{j+1}. This concludes the modified double-move. Now there are just two cans remaining, and they are located at $e_{j+1} \equiv e_{n-1}$.

9′. Now the algorithm makes two final moves. It fills the tank with the can which was deposited in Step 5′.

10′. It moves the can which is still loaded to $e_{n-1} + 1$;

11′. It fills the tank from the loaded can.

12′. It drives to $e_n := e_{n-1} + 2$ and stops at e_n.

With the correct values of i and j the new location in Step 5′ is written as

$$e_{n-1} := \frac{e_{n-3} + 2e_{n-2}}{3} + \frac{2}{3}. \tag{3}$$

Summary of the algorithm. We assume that $n \geq 4$.

Set $e_1 := e_2 := \cdots := e_{\lfloor n/2 \rfloor} := 0$;
if n is odd **then** Move two cans from 0 to 1/4 and return to 0;
Set $e_{(n+1)/2} := 1/4$; $i := 2$; $j := (n+1)/2$;
else Set $i := 1$; $j := n/2$;

while $j < n-2$ **do**
Perform a double-move, transforming two double-cans
at positions e_i and e_{i+1} into one double-can
at position $e_{j+1} := (e_i + 2e_{i+1} + e_{i+2})/4 + 1/2$;
Set $i := i+2$; $j := j+1$;
end while;
Perform a modified double-move, transforming two double-cans
at positions e_{n-3} and e_{n-2} into one double-can
at position $e_{n-1} := (e_{n-3} + 2e_{n-2})/3 + 2/3$;
Make a move to $e_{n-1}+1$, and from there make a move to $e_n := e_{n-1}+2$.

When $n = 1$ and $n = 2$, the problem is trivial. When $n = 3$, the jeep first moves two cans of fuel to $e_2 = 1/3$ and the following is the same as the case of $n = 2$.

The following example gives the sequence $(e_1, \ldots, e_n)$ for $n = 22$.

$$\left(0,0,0,0,0,0,0,0,0,0,0,\tfrac{1}{2},\tfrac{1}{2},\tfrac{1}{2},\tfrac{1}{2},\tfrac{1}{2},\tfrac{7}{8},1,\tfrac{35}{32},\tfrac{191}{128},\tfrac{389}{192},\tfrac{773}{192}\right)$$

The corresponding tour of the jeep, which reaches a distance of $\frac{773}{192} = 4\frac{5}{192}$, is shown in Figure 2.

Several observations can be made.

- Whenever the jeep moves backward it has no can loaded.
- Whenever the jeep moves forward it has a full can loaded, except in the last move (Step 12′). In this sense, there are no "wasted" moves.
- There are "big" dumps with many double-cans at half-integer positions 0, 1/2, 1, 3/2, ...
- In addition, there are "singular" dumps with single double-cans at intermediate positions. Between 0 and 1/2, there is a singular dump at position 1/4 whenever n is odd. Between successive big dumps beyond the point 1/2, there are in general two singular dumps, but there may also be only one.
- All can positions are fractions whose denominator is a power of 2, except e_{n-1} and e_n, which may have an additional factor of 3 in the denominator.
- At any time during the algorithm there are at most two big dumps and possibly two more single dumps plus possibly one "temporary" dump of the type $f_{i+1/2}$ with only one can deposited after Step 2 of the algorithm. However, at a time, there can be at most four dumps with non-empty cans.

The values e_i can be given by an explicit formula, which is however a little complicated to describe. Since i and j change simultaneously in increments of 2 and 1, respectively, we always have $n - i = 2(n - j) - 1$. Writing j as $n - k - 1$, the recursive equation defining e_{j+1} takes the following form.

$$e_{n-k} := \frac{e_{n-2k-1} + 2e_{n-2k} + e_{n-2k+1}}{4} + 1/2, \qquad \text{for } k = 1, \ldots, \lfloor n/2 \rfloor - 1 \tag{4}$$

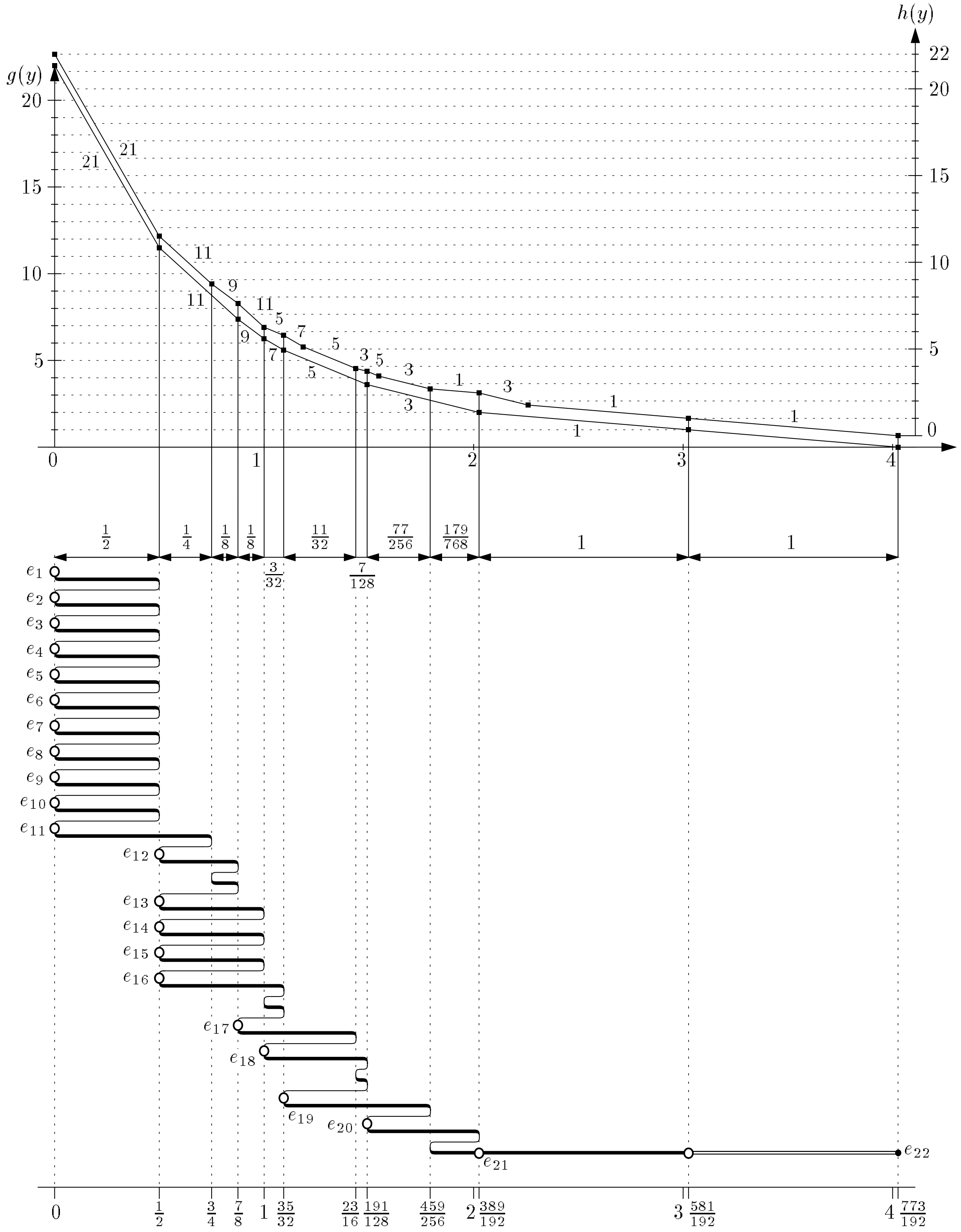

Figure 2: The optimal tour of the jeep with $n = 22$ cans. The top part shows the functions $g(y)$ and $h(y)$ defined in Section 3.1 (equations (8) and (9), respectively). The graph of $h(y)$ has been slightly offset from $g(y)$. The small numbers adjacent to the graphs are the negative slopes of the linear pieces.

Note that this equation is true even for $k = 1$, where it is equivalent to (3). Throughout this paper we will denote

$$p := \lfloor \log_2(n-1) \rfloor - 1,$$

i. e., we have $2^{p+1} < n \le 2^{p+2}$.

For $0 \le q \le p$ we set

$$a_q := \left\lfloor \frac{n-1}{2^q} \right\rfloor, \qquad b_q := n - 1 - a_q 2^q, \qquad \bar{b}_q := 2^q - b_q. \tag{5}$$

In other words, $b_q = (n-1) \bmod 2^q$, and $\bar{b}_q$ is the "complementary" remainder. Then we have $n = a_q 2^q + b_q + 1$. Furthermore, we set

$$A_q := n - a_q, \text{ for } q \ge 0.$$

The indices $i = A_q$ and $i = A_q - 1$ are those indices for which e_i is possibly not a multiple of $1/2$ and therefore the position of a singular dump. The following formula defines e_i for $1 \le i \le n - 3$.

$$\begin{aligned}
e_{A_q-1} &= \frac{q-1}{2} + \frac{\lfloor (b_q/2)^2 \rfloor}{4^q}, && \text{for } q \ge 1,\ A_q - 1 \le n-3 \\
e_{A_q} &= \frac{q}{2} - \frac{\lfloor (\bar{b}_q/2)^2 \rfloor}{4^q}, && \text{for } q \ge 0,\ A_q \le n-3 \\
e_i &= \frac{q}{2}, && \text{for } A_q < i < A_{q+1} - 1,\ \ q \ge 0,\ \ i \le n-3
\end{aligned} \tag{6}$$

The last three values are given as follows.

$$e_{n-2} = \begin{cases} p/2 - 2 + 5(n-1)/2^{p+2} - 3\lfloor (n-1)^2/4 \rfloor / 4^{p+1} & (2^{p+1} < n \le 3 \cdot 2^p) \\ p/2 + 1/4 - (n-1)/2^{p+2} + \lfloor (n-1)^2/4 \rfloor / 4^{p+1} & (3 \cdot 2^p < n \le 2^{p+2}) \end{cases}$$

$$e_{n-1} = \frac{p-1}{2} + \frac{n-1}{2^{p+1}} - \frac{\lfloor (n-1)^2/4 \rfloor}{6 \cdot 4^p}$$

$$e_n = e_{n-1} + 2 = \frac{p+3}{2} + \frac{n-1}{2^{p+1}} - \frac{\lfloor (n-1)^2/4 \rfloor}{6 \cdot 4^p} \tag{7}$$

The proof that these values fulfill the inductive definition (4) is somewhat tedious and is given in Appendix A.

Theorem 1 *The maximum distance L_n which the jeep can reach with n cans and restricted refilling is given by*

$$L_n = \frac{p+3}{2} + \frac{n-1}{2^{p+1}} - \frac{\lfloor (n-1)^2/4 \rfloor}{6 \cdot 4^p},$$

where $2^{p+1} < n \le 2^{p+2}$, for $n \ge 2$, and $L_1 = 1$.

n	L_n	n	L_n
1	$1 = 1 = 1.0000000$	21	$383/96 = 1532/384 = 3.9895833$
2	$2 = 2 = 2.0000000$	22	$773/192 = 1546/384 = 4.0260417$
3	$7/3 = 14/6 = 2.3333333$	23	$1559/384 = 1559/384 = 4.0598958$
4	$8/3 = 16/6 = 2.6666667$	24	$131/32 = 1572/384 = 4.0937500$
5	$17/6 = 68/24 = 2.8333333$	25	$33/8 = 1584/384 = 4.1250000$
6	$3 = 72/24 = 3.0000000$	26	$133/32 = 1596/384 = 4.1562500$
7	$25/8 = 75/24 = 3.1250000$	27	$1607/384 = 1607/384 = 4.1848958$
8	$13/4 = 78/24 = 3.2500000$	28	$809/192 = 1618/384 = 4.2135417$
9	$10/3 = 320/96 = 3.3333333$	29	$407/96 = 1628/384 = 4.2395833$
10	$41/12 = 328/96 = 3.4166667$	30	$273/64 = 1638/384 = 4.2656250$
11	$335/96 = 335/96 = 3.4895833$	31	$549/128 = 1647/384 = 4.2890625$
12	$57/16 = 342/96 = 3.5625000$	32	$69/16 = 1656/384 = 4.3125000$
13	$29/8 = 348/96 = 3.6250000$	33	$13/3 = 6656/1536 = 4.3333333$
14	$59/16 = 354/96 = 3.6875000$	34	$209/48 = 6688/1536 = 4.3541667$
15	$359/96 = 359/96 = 3.7395833$	35	$6719/1536 = 6719/1536 = 4.3743490$
16	$91/24 = 364/96 = 3.7916667$	36	$1125/256 = 6750/1536 = 4.3945313$
17	$23/6 = 1472/384 = 3.8333333$	37	$565/128 = 6780/1536 = 4.4140625$
18	$31/8 = 1488/384 = 3.8750000$	38	$1135/256 = 6810/1536 = 4.4335938$
19	$501/128 = 1503/384 = 3.9140625$	$n = 2^{p+1} + 1$ $(p \geq 0)$:	$L_n = p/2 + 7/3$
20	$253/64 = 1518/384 = 3.9531250$	$n = 3 \cdot 2^p + 1$ $(p \geq 1)$:	$L_n = p/2 + 21/8$

Table 1: Values of L_n

We have just seen that the distance L_n, which coincides with the formula for e_n given above, can be attained. The proof that L_n is also an upper bound will be given in Section 3.

Table 1 gives a few values of L_n, and Figure 3 shows L_n as a function of n. The figure exhibits a logarithmic behavior, and indeed, when one interpolates a logarithmic function through the values $L_{2^k+1} = (3k + 11)/6$, one gets a very good approximation:

$$\frac{11}{6} + \log_4(n-1) - 0.005 < L_n < \frac{11}{6} + \log_4(n-1) + 0.005, \quad \text{for all } n \geq 9.$$

When restricted to even values of n or to odd values of n, the function is piecewise quadratic, with breakpoints at the powers of 2. The recurrence relation $L_{4n+1} = L_{2n+1} + 1/2$, which can be proved easily, is also in accordance with the logarithmic growth of L_n.

Comparison with Brauer and Brauer's solution Brauer and Brauer [1989] conjectured that for the values of the form $n = (4^k + 2)/3$ $(k \geq 0)$, their solution achieves the optimal distance $f_U(n) = k+1$. For these values of n, we have $p = 2k-3$,

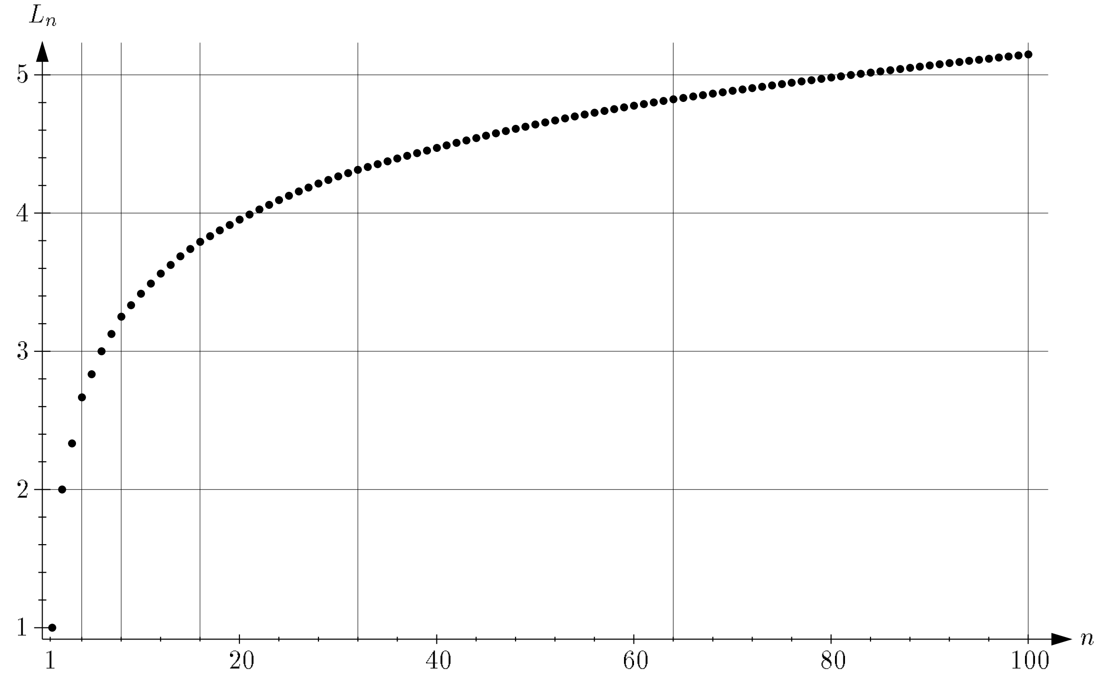

Figure 3: The maximum attainable distance L_n for the jeep problem with n cans and with refilling of whole cans only.

and we get

$$L_n = k + 1 + \frac{1}{27} \cdot \left(1 - \frac{5}{4^{k-1}} + \frac{1}{4^{2k-3}}\right).$$

Therefore, when $n = 2$ ($k = 1$) or $n = 6$ ($k = 2$), our solution is the same as Brauer and Brauer's solution, but for $n > 6$ ($k \geq 3$), our solution is better, and the difference approaches $1/27$ as $k \to \infty$. In contrast to Wood's solution, whose length is bounded by a constant fraction of the optimum, Brauer and Brauer's solution achieves the correct asymptotic growth factor.

One characteristic of our algorithm is that cans are always kept in pairs. To see why this is a good idea, consider the situation after a double-move, as shown in Figure 1, but assume that position e_{i+2} contains only a single can. Then the jeep can only fill the tank there but carry no useful load. (The other possibility would be to skip this can, but then the can would be wasted.) This unsatisfactory situation can never be avoided when the jeep tries to follow the straightforward *greedy strategy* that always goes from e_i to $f_{i+1/2}$ and back to e_{i+1}. The optimal algorithm also follows the greedy approach, but with the additional proviso that the single can $f_{i+1/2}$ is never left alone: in the next move, instead of bringing a new can to $f_{i+1+1/2}$, it is better to fetch the single can from $f_{i+1/2}$ and have the two cans in one place. This approach leads naturally to the structure of double-moves.

3 The upper bound

3.1 Valid inequalities

Let's take the attitude of the Desert Intelligence Service who would like to prove that the driver of the jeep could not have completed the journey without illicit help. Besides the traces on the road-track, which are not very reliable, the clues which are at our disposal for investigating the matter are the empty cans on the way. We assume that, whenever the tank is filled, the empty can is left behind, because an empty can would block the transport of a full can. As an exception to this rule, we assume that, after filling the tank for the last time, in the absence of other cargo, the jeep carries the last empty can to the final destination. Let's denote the positions of the empty cans by $e_1 \leq e_2 \leq \cdots \leq e_n$. It is no coincidence that we use the same notation e_i as for the positions of the double-cans in Section 2, because in that solution, one empty can is left at each double-can position. The jeep starts at position $0 = e_1$ and terminates at e_n. We can assume without loss of generality that $e_n \geq e_{n-1} + 1$, because at least one can was opened to the right of e_{n-1}, and thus the jeep could certainly have reached $e_{n-1} + 1$. (If this condition is not fulfilled we redefine $e_n := e_{n-1} + 1$.)

Let's look at a segment $[e_i, e_{i+1}]$ with $e_i < e_{i+1}$. Between e_i and e_{i+1}, we know that the jeep must have passed at least $2(n-i)-1$ times: Since $n-i$ cans were moved into the interval $[e_{i+1}, \infty]$, the jeep must have passed at least $n-i$ times in the forward direction, in order to bring these cans to e_{i+1} or beyond. Hence it must have passed at least $n-i-1$ times in the backward direction. We denote this *necessary driving density*, which depends on the location y, by $c(y)$. So we have

$$c(y) = 2 \cdot |\{\, j : 1 \leq j \leq n,\ e_j \geq y \,\}| - 1, \qquad \text{for } 0 \leq y \leq e_n$$

This gives us, for any position y between 0 and e_n, a lower bound $g(y)$ on the total distance which the jeep has driven in the interval $[y, \infty]$, or in other words, on *the total amount of fuel spent* in this interval. If $e_i \leq y \leq e_{i+1}$ for some $1 \leq i \leq n$ then

$$\begin{aligned} g(y) &:= \int_y^{e_n} c(x)\,dx \\ &= (e_i - y) \cdot (2(n-i)-1) \ + \sum_{j=i+1}^{n-1} (e_{j+1} - e_j) \cdot (2(n-j)-1) \\ &= -y(2n-2i-1) \ + \sum_{j=i+1}^{n-1} 2e_j \ + \ e_n \ = \ \sum_{j=i+1}^{n-1} 2(e_j - y) \ + \ (e_n - y). \end{aligned} \tag{8}$$

Now we derive an upper bound $h(y)$ on the amount of fuel used in the interval $[y, \infty]$. For this bound, we will just consider the positions e_i at which the tank was filled, without caring how the cans got there.

Lemma 1 *For any y with $0 \le y \le e_n$, the amount of fuel consumed by driving in the interval $[y, \infty]$ is bounded by*

$$h(y) := \sum_{j=1}^{i-1} \max\{1 + e_j + e_{j+1} - 2y, 0\} + (1 + e_i - y) + (n - i), \tag{9}$$

where i $(1 \le i < n)$ is an index for which $e_i \le y \le e_{i+1}$ holds.

An example of the function h is shown in the upper part of Figure 2.

Before proving this bound we note that the bound $h(y)$ in the lemma is actually achieved by a "right-extreme consumption tour", which visits the cans in the left-to-right order $e_1, \dots, e_n$. Between e_j and e_{j+1}, the jeep moves forward to the turning point $f_{j+1/2} = (e_j + e_{j+1} + 1)/2$ and retreats to e_{j+1}. If $e_{j+1} \le e_j + 1$ and such a tour is at all possible, $h(y)$ equals the amount of fuel consumed by driving in the interval $[y, \infty]$, for all y with $0 \le y \le e_n$.

Proof. Let us call a move *relevant* if the jeep reaches some point in the interval $[y, \infty]$ during the move. Let $l_1, l_2, \dots, l_s$ denote the starting cans and let $m_1, m_2, \dots, m_{s-1}$ denote the ending cans of the relevant moves. (The last move has no ending can.) Each can $1, \dots, n$ occurs at most once in the list $l_1, l_2, \dots, l_s$ and at most once in the list $m_1, m_2, \dots, m_{s-1}$, and each can $i+1, \dots, n$ occurs either in both lists or in none (in case the can is not used at all). We now delete cans $\ge i+1$ from both lists. Let $l_1, l_2, \dots, l_t$ $(1 \le t \le i)$ denote the starting cans of the relevant moves among the first i cans, and let $m_1, m_2, \dots, m_{t-1}$ denote the ending cans of the relevant moves among the first i cans. There are at most $t + (n - i)$ relevant moves, which contribute to the fuel consumed in the interval $[y, \infty]$. From the total amount of fuel consumed during these moves, $t + n - i$, we must subtract

$$(y - e_{l_1}) + (y - e_{l_2}) + \cdots + (y - e_{l_t})$$

for the fuel that is necessary to reach y from l_j, and

$$(y - e_{m_1}) + (y - e_{m_2}) + \cdots + (y - e_{m_{t-1}})$$

for the fuel to return from y to m_j. Thus we have

$$\begin{aligned} h(y) \le (n - i) + t - [(y - e_{l_1}) + (y - e_{l_2}) + \cdots + (y - e_{l_t})] \\ - [(y - e_{m_1}) + (y - e_{m_2}) + \cdots + (y - e_{m_{t-1}})]. \end{aligned}$$

Since the l_j and the m_j are distinct indices from the set $\{1, \dots, i\}$ and $e_1 \le e_2 \le \cdots \le e_i$ we get

$$\begin{aligned} h(y) &\le (n - i) + t - [(y - e_i) + (y - e_{i-1}) + \cdots + (y - e_{i-t+1})] \\ &\quad - \; [(y - e_i) + (y - e_{i-1}) + \cdots + (y - e_{i-t+2})] \\ &= (n - i) + (1 + e_i - y) + \sum_{j=i-t}^{i-1} (1 + e_j + e_{j-1} - 2y). \\ &\le (n - i) + (1 + e_i - y) + \sum_{j=i-t}^{i-1} \max\{1 + e_j + e_{j-1} - 2y, 0\} \end{aligned}$$

This is clearly bounded by expression (9). ■

With $g(y)$ and $h(y)$ as defined above, we can write a necessary condition that the sequence $e_1, \ldots, e_n$ comes from a feasible solution.

$$g(y) \leq h(y), \text{ for all } y \in [0, e_n] \tag{10}$$

These are infinitely many conditions, but we will show that it suffices to check them at a finite number of points.

3.2 A linear programming formulation

First, note that the function $g(y)$ is continuous, piecewise linear, and *convex*, see Figure 2. Its slope $g'(y)$, when it is defined, equals $-c(y)$. As y decreases from e_{n+1} to e_1, this quantity decreases (becomes more negative, i. e., the function becomes steeper). Therefore $g(y)$ is convex.

The function $h(y)$, on the other hand, is also piecewise linear and continuous, but neither concave nor convex. As y increases from 0 to e_n, the function will make a downward bend at each point e_j (and its slope $h'(y)$ decreases by 2) because the summation bound i in (9) increases by 1. On the other hand, $h(y)$ will make an upward bend at each point y for which $1 + e_j + e_{j+1} - 2y = 0$ (and $h'(y)$ increases by 2). These are just the points $y = f_{j+1/2} = (e_j + e_{j+1} + 1)/2$. Of course, if several of these events occur at the same point y, their effects may cumulate or cancel each other, and h may either bend upward or downward.

What is important, however, is that h may bend upward only at the points $f_{j+1/2}$. Between two such points, it is a *concave* function.

Now if some function g is convex and another function h is concave over some interval, then $g(y) \leq h(y)$ holds for the whole interval if and only if it holds for the endpoints of the interval. Thus by decomposing the whole range into subintervals where h is concave, we see that it is sufficient to check (10) at the possible points of nonconcavity of h, i. e, at the points $f_{j+1/2}$, and at the endpoints 0 and e_n of the whole interval. All values $y = f_{j+1/2}$ lie in the interval $[0, e_n]$ and can be tested, since $f_{n-1/2} \leq e_n$ follows from $e_n \geq e_{n-1} + 1$. At $y = e_n$, we have $g(y) = h(y) = 0$, and thus $g(y) \leq h(y)$ is automatically fulfilled. Condition (10) takes now the following form.

$$g(y) \leq h(y), \text{ for } y = 0, f_{3/2}, f_{5/2}, \ldots, f_{n-1/2} \tag{11}$$

We will now substitute these values y into (10), using (8) and (9). For $y = 0$ we get

$$g(0) = \sum_{j=1}^{n-1} 2e_j + e_n \tag{12}$$

and $h(0) = n$, which yields the first inequality:

$$\sum_{j=1}^{n-1} 2e_j + e_n \leq n \tag{13}$$

For the values $y = f_{k+1/2} = (e_k + e_{k+1} + 1)/2$ $(k = 1, 2, \dots, n-1)$, we can simplify the expression (9) for $h(y)$, because we have $1 + e_j + e_{j+1} - 2y \geq 0$ for $j \geq k$, and $1 + e_j + e_{j+1} - 2y \leq 0$ for $j < k$.

$$\begin{aligned} h(f_{k+1/2}) = h(y) &= \sum_{j=k}^{i-1}(1 + e_j + e_{j+1} - 2y) + (1 + e_i - y) + (n - i) \\ &= (e_k - y) + \sum_{j=k+1}^{i} 2(e_j - y) + (n - k + 1) \end{aligned}$$

Combining this with (8) gives

$$\begin{aligned} 0 &\geq -h(y) + g(y) \\ &= (y - e_k) + \sum_{j=k+1}^{i} 2(y - e_j) - (n - k + 1) + \sum_{j=i+1}^{n-1} 2(e_j - y) + (e_n - y) \\ &= e_n - e_k + \sum_{j=k+1}^{n-1} 2|e_j - y| - (n - k + 1). \end{aligned}$$

The last equation is true because $e_j \leq y$ for $j \leq i$ and $e_j \geq y$ for $j > i$. So we finally get the following inequalities.

$$e_n - e_k + \sum_{j=k+1}^{n-1} 2 \cdot \left|e_j - f_{k+1/2}\right| \leq n - k + 1, \qquad \text{for } k = 1, \dots, n-1$$

It will be convenient to change the index variable from k to $k' = n - k$. Writing again k instead of k' gives us the final form in which we want to use the inequalities.

$$e_n - e_{n-k} + \sum_{j=n-k+1}^{n-1} 2|e_j - f_{n-k+1/2}| \leq k + 1, \qquad \text{for } k = 1, \dots, n-1 \tag{14}$$

To summarize, we get the following optimization problem whose objective value is an upper bound on the solution of the jeep problem.

$$\begin{aligned} &\text{maximize } e_n \\ &\text{subject to (13), (14), and } 0 = e_1 \leq e_2 \leq \dots \leq e_{n-1} \leq e_n - 1. \end{aligned} \tag{15}$$

This problem can be formulated as a linear programming problem. The standard way to get rid of the absolute value signs in the constraints is as follows: replace each absolute value $|E|$ of an expression E that occurs on the left side by a new variable x, and add the two inequalities $E \leq x$ and $-E \leq x$ to the set of constraints. This results in a standard linear programming problem, which can be solved by the simplex algorithm or any of the more recent linear optimization algorithms. This is how we initially obtained the values e_i which are given in Section 2, for different values of n.

It turns out that the optimal solution is highly degenerate: all inequalities (13) and (14) are fulfilled with equality, and many of the expressions in (14) whose absolute value is taken are zero. Thus, when solving the linear programs on the computer by the simplex method, it seems preferable to solve the dual linear program. Indeed, we observed a speedup factor of almost 10 for large problems when we switched from the primal simplex method to the dual simplex method.

3.3 An upper bound

The inequalities (14) are difficult to handle because they contain absolute values. Using the fact that $x \le |x|$ and $-x \le |x|$ holds for all x, we can replace each expression $|e_j - f_{n-k+1/2}|$ either by $(e_j - f_{n-k+1/2})$ or by $-(e_j - f_{n-k+1/2})$, as we please, and we derive a valid inequality.

We now take inequality (14) for each even k and replace $|e_j - f_{n-k+1/2}|$ by $(e_j - f_{n-k+1/2})$ for $j > n - k/2$ and by $-(e_j - f_{n-k+1/2})$ otherwise. This gives the following inequalities

$$e_n - e_{n-k+1} - \sum_{j=n-k+2}^{n-k/2} 2(e_j - f_{n-k+1/2}) + \sum_{j=n-k/2+1}^{n-1} 2(e_j - f_{n-k+1/2}) \le k.$$

The terms $f_{n-k+1/2}$ cancel, and we get

$$-e_{n-k+1} - \sum_{j=n-k+2}^{n-k/2} 2e_j + \sum_{j=n-k/2+1}^{n-1} 2e_j + e_n \le k, \qquad \text{for } 2 \le k \le n-1,\ k \text{ even.} \tag{16}$$

Examples. For $n = 10$, this gives, together with (13), the following 5 inequalities:

$$-e_9 + e_{10} \le 2 \qquad (I_1)$$
$$-e_7 - 2e_8 + 2e_9 + e_{10} \le 4 \qquad (I_2)$$
$$-e_5 - 2e_6 - 2e_7 + 2e_8 + 2e_9 + e_{10} \le 6 \qquad (I_3)$$
$$-e_3 - 2e_4 - 2e_5 - 2e_6 + 2e_7 + 2e_8 + 2e_9 + e_{10} \le 8 \qquad (I_4)$$
$$2e_1 + 2e_2 + 2e_3 + 2e_4 + 2e_5 + 2e_6 + 2e_7 + 2e_8 + 2e_9 + e_{10} \le 10 \qquad (I_0)$$

Multiplying the inequalities by $\frac{2}{3}$, $\frac{1}{6}$, $\frac{1}{24}$, $\frac{1}{24}$ and $\frac{1}{12}$, and summing them gives

$$\frac{e_1}{6} + \frac{e_2}{6} + \frac{e_3}{8} + \frac{e_4}{12} + \frac{e_5}{24} + e_{10} \le \frac{41}{12}.$$

Since $e_1, e_2, e_3, e_4, e_5 \ge 0$, this implies that the farthest distance that a jeep can travel with 10 cans of fuel is at most $e_{10} = 41/12$. This is the value achieved by the algorithm in Section 2.

For $n = 9$, we get the following 5 inequalities:

$$-e_8 + e_9 \le 2 \qquad (I_1)$$
$$-e_6 - 2e_7 + 2e_8 + e_9 \le 4 \qquad (I_2)$$
$$-e_4 - 2e_5 - 2e_6 + 2e_7 + 2e_8 + e_9 \le 6 \qquad (I_3)$$
$$-e_2 - 2e_3 - 2e_4 - 2e_5 + 2e_6 + 2e_7 + 2e_8 + e_9 \le 8 \qquad (I_4)$$
$$2e_1 + 2e_2 + 2e_3 + 2e_4 + 2e_5 + 2e_6 + 2e_7 + 2e_8 + e_9 \le 9 \qquad (I_0)$$

The same multipliers as above lead to

$$\frac{e_1}{6}+\frac{e_2}{8}+\frac{e_3}{12}+\frac{e_4}{24}+e_9\le\frac{10}{3},$$

giving a bound of 10/3 for 9 cans.

Generally, we proceed as follows. We denote the inequalities (16) by $I_1, I_2, \dots, I_{\lfloor(n-1)/2\rfloor}$, where I_l denotes the inequality with $k=2l$. By I_0 we denote the inequality (13). Consider the infinite sequence $(m_1, m_2, \dots) = (\frac{2}{3}, \frac{1}{6}, \frac{1}{24}, \frac{1}{24}, \frac{1}{96}, \frac{1}{96}, \frac{1}{96}, \frac{1}{96}, \frac{1}{384}, \frac{1}{384}, \dots)$ defined by

$$m_l = g_q := \frac{1}{6\cdot 4^q}, \qquad \text{for } 2^q < l \le 2^{q+1},\ q=-1,0,1,2,\dots,$$

which has $\sum_{l=1}^{\infty} m_l = 1$. Now we multiply I_l by m_l for $l=1,\dots,\lfloor(n-1)/2\rfloor$ and I_0 by r, where

$$r = 1-\sum_{l=1}^{\lfloor(n-1)/2\rfloor} m_l$$

denotes the remaining part of the series $\sum_{l=1}^{\infty} m_l$. By summing these inequalities we obtain one inequality of the following form.

$$c_1e_1+c_2e_2+\cdots+c_{n-1}e_{n-1}+c_ne_n\le S_n \tag{17}$$

Lemma 2 *$c_1 \ge c_2 > c_3 > \cdots > c_{\lfloor n/2\rfloor} > 0 = c_{\lfloor n/2\rfloor+1} = \cdots = c_{n-1} = 0$, and $c_n = 1$.*

Proof. e_1 and e_2 have the same coefficients in all inequalities except in $I_{(n-1)/2}$, when n is odd. There the coefficient of e_1 is 0 and the coefficient of e_2 is -1. Thus $c_1 \ge c_2$.

If $2 \le j \le n/2$, e_j and e_{j+1} have the same coefficients in all inequalities except $I_{\lceil(n-j)/2\rceil}$, where the coefficients of e_j and e_{j+1} are -1 and -2 if $n-j$ is odd 0 and -1 if $n-j$ is even. In any case, the coefficient of e_{j+1} is smaller than the coefficient of e_j and hence $c_j > c_{j+1}$.

For $n/2 \le j \le n-2$, we will first show that $c_j = c_{j+1}$; in the end we will show that $c_{n-1}=0$. If $n/2\le j\le n-2$, then e_j and e_{j+1} have different coefficients only in the two inequalities I_{n-j} and $I_{\lceil(n-j)/2\rceil}$:

$$-e_{2j-n+1}-\cdots-2e_j+2e_{j+1}+\cdots\le 2n-2j \tag{I_{n-j}}$$

The other inequality $I_{\lceil(n-j)/2\rceil}$ reads

$$-e_j-2e_{j+1}+\cdots+e_n\le n-j+1, \tag{$I_{(n-j+1)/2}$}$$

if $n-j$ is odd, and

$$-e_{j+1}-2e_{j+2}+\cdots+e_n\le n-j, \tag{$I_{(n-j)/2}$}$$

if $n-j$ is even. If $2^q < \lceil(n-j)/2\rceil \le 2^{q+1}$, then $2^{q+1} < n-j \le 2^{q+2}$. Hence, $m_{\lceil(n-j)/2\rceil} = g_q$ and $m_{n-j} = g_{q+1}$ for some q, and

$$c_j - c_{j+1} = g_{q+1}(-2-(+2)) + g_q(-1-(-2)) = -4g_{q+1}+g_q = 0,$$

if $n-j$ is odd, and

$$c_j - c_{j+1} = g_{q+1}(-2-(+2)) + g_q(0-(-1)) = 0,$$

if $n-j$ is even. Finally, we consider c_{n-1} and c_n. We have

$$c_{n-1} = -2/3 + 2(1-2/3) = 0$$

and

$$c_n = r + \sum_{l=1}^{\lfloor (n-1)/2 \rfloor} m_l = 1.$$

The proof is complete. ■

Lemma 3 *The constant term S_n in inequality (17) is given by the expression*

$$S_n = \frac{p+3}{2} + \frac{n-1}{2^{p+1}} - \frac{\lfloor (n-1)^2/4 \rfloor}{6 \cdot 4^p},$$

where $2^{p+1} < n \le 2^{p+2}$.

Proof. We have

$$S_n = nr + \sum_{l=1}^{\lfloor (n-1)/2 \rfloor} 2l m_l$$

Since $2^{p+1} < n \le 2^{p+2}$, we have $2^p \le \lfloor (n-1)/2 \rfloor \le 2^{p+1} - 1$, and the two parts in this expression can be calculated as follows.

$$\begin{aligned}
r = 1 - \sum_{l=1}^{\lfloor (n-1)/2 \rfloor} m_l &= 1 - \left(\frac{2}{3} + \sum_{q=0}^{p-1} 2^q g_q + (\lfloor \tfrac{n-1}{2} \rfloor - 2^p) g_p \right) \\
&= 1 - \left(\frac{2}{3} + \sum_{q=0}^{p-1} \frac{2^q}{6 \cdot 4^q} + \left(\lfloor \tfrac{n-1}{2} \rfloor - 2^p \right) \frac{1}{6 \cdot 4^p} \right) \\
&= \frac{1}{3} - \frac{1}{6} \cdot 2 \left(1 - \frac{1}{2^p} \right) - \left(\lfloor \tfrac{n-1}{2} \rfloor - 2^p \right) \frac{1}{6 \cdot 4^p} \\
&= \frac{1}{3 \cdot 2^p} + \frac{1}{6 \cdot 2^p} - \frac{\lfloor (n-1)/2 \rfloor}{6 \cdot 4^p} = \frac{1}{2^{p+1}} - \frac{\lfloor (n-1)/2 \rfloor}{6 \cdot 4^p}
\end{aligned}$$

$$\begin{aligned}
\sum_{l=1}^{\lfloor (n-1)/2 \rfloor} 2l m_l &= \frac{4}{3} + \sum_{q=0}^{p-1} \frac{1}{6 \cdot 4^q} \sum_{l=2^q+1}^{2^{q+1}} 2l + \frac{1}{6 \cdot 4^p} \sum_{l=2^p+1}^{\lfloor (n-1)/2 \rfloor} 2l \\
&= \frac{4}{3} + \sum_{q=0}^{p-1} \frac{4^{q+1} + 2^{q+1} - 4^q - 2^q}{6 \cdot 4^q} + \frac{\lfloor \frac{n-1}{2} \rfloor^2 + \lfloor \frac{n-1}{2} \rfloor - 4^p - 2^p}{6 \cdot 4^p} \\
&= \frac{4}{3} + \frac{p}{2} + \frac{2}{6} \left(1 - \frac{1}{2^p} \right) + \frac{\lfloor \frac{n-1}{2} \rfloor^2 + \lfloor \frac{n-1}{2} \rfloor}{6 \cdot 4^p} - \frac{1}{6} - \frac{1}{6 \cdot 2^p} \\
&= \frac{3}{2} + \frac{p}{2} + \frac{\lfloor \frac{n-1}{2} \rfloor^2 + \lfloor \frac{n-1}{2} \rfloor}{6 \cdot 4^p} - \frac{1}{2^{p+1}}
\end{aligned}$$

Therefore,

$$S_n = \frac{p+3}{2} + \frac{n}{2^{p+1}} - \frac{1}{2^{p+1}} + \frac{\lfloor \frac{n-1}{2} \rfloor^2 - n\lfloor \frac{n-1}{2} \rfloor + \lfloor \frac{n-1}{2} \rfloor}{6 \cdot 4^p}$$

When n is odd,

$$\lfloor \tfrac{n-1}{2} \rfloor^2 - n\lfloor \tfrac{n-1}{2} \rfloor + \lfloor \tfrac{n-1}{2} \rfloor = -(n-1)^2/4 = -\lfloor \tfrac{(n-1)^2}{4} \rfloor,$$

and when n is even,

$$\lfloor \tfrac{n-1}{2} \rfloor^2 - n\lfloor \tfrac{n-1}{2} \rfloor + \lfloor \tfrac{n-1}{2} \rfloor = -n(n-2)/4 = -\lfloor \tfrac{(n-1)^2}{4} \rfloor.$$

So we finally get

$$S_n = \frac{p+3}{2} + \frac{n-1}{2^{p+1}} - \frac{\lfloor (n-1)^2/4 \rfloor}{6 \cdot 4^p},$$

and the proof is complete. ■

With the two previous lemmas, the proof of Theorem 1 can be completed. By Lemma 2, (17) gives rise to

$$e_n \le c_1 e_1 + c_2 e_2 + \cdots + c_{n-1} e_{n-1} + e_n \le S_n, \tag{18}$$

and by Lemma 3, the upper bound S_n is equal to the value L_n claimed in Theorem 1.

Remark 1. The optimal values of $e_1, \ldots, e_n$ are unique. This can be seen as follows. Since the upper bound in (18) is tight, all inequalities that entered into the derivation of $e_n \le S_n$ must be fulfilled as equations. These are the inequalities I_0, I_1, ..., $I_{\lfloor (n-1)/2 \rfloor}$, i. e. the relations (13) and (16); all of them were used with positive multipliers m_l and r. In addition, the inequalities $e_1 \ge 0$, $e_2 \ge 0$, ... $e_{\lfloor n/2 \rfloor} \ge 0$ were used in (18) with positive multipliers, by Lemma 2. (Note that the relations $e_i \le e_{i+1}$ were not used in bounding e_n, but of course they are important for deriving the constraints (10) and hence (13) and (14).)

So any solution with $e_n = S_n$ must have $e_1 = e_2 = \cdots = e_{\lfloor n/2 \rfloor} = 0$. This corresponds to the initialization step of the algorithm in Section 2. Now we use the relations (16) as equations. From (16) with $k = 2$ obtain directly $e_n = e_{n-1} + 2$. If we take the equations (16) for two successive even values $k = 2l$ and $k = 2l + 2$ and form their difference, we obtain

$$-e_{n-2l-1} - 2e_{n-2l} - e_{n-2l-1} + 4e_{n-l} = 2, \quad \text{for } l = 1, \ldots, \lfloor (n-1)/2 \rfloor - 1.$$

(See the examples after (16).) This is just the recursive relation (4), which includes (3) as a special case. One more relation can be obtained by taking the difference of (13) and (16) for $k = 2\lfloor \frac{n-1}{2} \rfloor$. Simplifying by the fact that the first values $e_1, e_2, \ldots, e_{\lfloor n/2 \rfloor}$ are 0, we obtain $4e_{n/2+1} = 2$ if n is even, and $4e_{(n+1)/2} = 1$ if n is odd. In the even case, this corresponds to the case $e_{n/2+1} := \frac{e_1+2e_2+e_3}{4} + 1/2 = 1/2$ of the recursion (4), which was still missing. If n is odd, this corresponds to the initialization statement $e_{(n+1)/2} := 1/4$ of the algorithm, which was also missing. So we have derived all initial conditions and recursive equations that are necessary to determine the values e_i uniquely.

The above argument can also be reversed to show that the values e_i given by the initial conditions and the recursion fulfill (16), (13), and hence (18) as equations. This gives an independent verification that the value e_n calculated in Appendix A must equal the value S_n computed in Lemma 3.

Remark 2. All inequalities (13) and (14) are fulfilled with equality. We have just observed that (14) is fulfilled as inequality for all even k because (16) is derived from these inequalities. If we take (14) for an odd value k' and use the same rules for replacing the absolute values as at the beginning of Section 3.3 in the derivation of (16), it turns out that we get precisely the same inequality as (16) with $k = k'+1$, and hence equality must hold for *all* k.

Remark 3. Theorem 2 and Corollary 1 in Appendix B show that the necessary inequalities for e_i of problem (15), which we have derived in this section, are sufficient to guarantee a feasible trip where the jeep leaves the empty cans at the specified positions. (Actually, it turns out that a very restricted set of inequalities is already sufficient for this purpose.)

In the proof, we construct the tour for the jeep in a way which is somewhat similar to the algorithm of Section 2, except that the values e_i are given in advance, and there is the possibility of failure, in which case one violated condition must be identified. Although this construction in its generality is not needed for proving the optimality results in our paper, we think that it might be useful to treat other variations of the problem.

We used a construction procedure similar to the one given in Appendix B to find the feasible solutions for some sets of values $(e_1, \ldots, e_n)$ which we had obtained from our model with linear programming software. By studying many optimal solutions which were obtained in this way we finally discovered the easy strategy lying behind them which is described in Section 2.

Finally, knowing the optimal solution, we went back to the constraints of the linear program to investigate which of them are really needed to bound the optimum. This is how we obtained the above optimality proof.

4 Other Variants of the Problem

Dewdney [1987] proposed a variant of the problem in which only constraint 1 but none of the constraints 4 and 5 is enforced, i. e., fuel can only be stored in cans, but any amount of fuel can be filled in the tank at any time. This problem was solved by Jackson, Mitchem, and Schmeichel [1995]. Actually, Dewdney proposed a variation where each can holds $C = 5$ tankfuls of fuel, and this variation is the problem which was explicitly solved by Jackson et al. As they pointed out in their paper, their solution can be applied for general C. Their approach also uses a linear programming problem which is derived in a very similar way as the inequality in our paper. However, their solution has a completely different character from the solution of our problem. Dewdney's constraint leads to a different expression for $h(y)$, and the finite set of "critical values" y at which the inequality $g(y) \leq h(y)$ must be checked consists of the values $e_i + 1/2$, as opposed to $f_{i+1/2}$ in our case. Moreover, they could show that their linear program can be solved in a greedy manner from right to left, by setting e_n at some arbitrary value and successively minimizing $e_{n-1}, e_{n-2}, \dots, e_1$. This is in contrast to our solution, where e_i is successively determined from left to right. All depot locations e_i are "singular" in the sense that only one can is stored in each location, apart from the cans which remain at the starting location and are never moved. It even turns out that locations of the cans which are moved form a pattern which is independent of n. In other words, for any n, the sequence of numbers $e_n - e_{n-1}, e_n - e_{n-2}, e_n - e_{n-3}, \dots$ forms an initial segment of an infinite sequence which is independent of n.

Despite this large amount of structure, it seems difficult to give an explicit formula for the maximum distance that can be traveled with n cans, or even to compute the precise order of magnitude.

We initially developed our solution for our version of the jeep problem independently of the paper of Jackson et al. [1995]. However, their paper made us aware that assumption 5 must be stated explicitly, and in general one must distinguish very carefully the different constraints that are formulated in different versions of the jeep problem. "It is good exercise to cultivate the habit of being very wary about the exact wording of a puzzle. It teaches exactitude and caution," as Dudeney [1917, p. vi] put it in the introduction to his classic collection of mathematical puzzles.

When only property 5 but not 4 is omitted, we have another possible interpretation of Wood's jeep problem, as mentioned in the introduction: Any amount of fuel may be filled into the tank, but only when the tank is empty. The range of permissible solutions is now intermediate between our problem and Dewdney's problem, and the optimal solution is different from both problems.

The example in Figure 4 shows a solution for the case $n = 22$. The distance traveled exceeds the optimal solution of our problem (shown in Figure 2) by $1/224$. Whereas the final part (from e_{14} to the end) follows the same scheme of double-moves as our algorithm of section 2, the initial part is different. The parameters of the initial part have been "locally" optimized, but we do not claim that the optimal algorithm would follow this type of strategy.

The difference between the problem with constraint 4 and Dewdney's jeep problem, where constraint 4 is omitted, can be ascertained already for $n = 5$. The optimal solution of Dewdney's problem has the unique values $e_1 = e_2 = 0$, $e_3 = 1/5$, $e_4 = 13/15$, and $e_5 = 43/15$. It can be checked quite easily that it is impossible to attain these values when constraint 4 is in effect.

Acknowledgement. We thank Professor W. Oberschelp for pointing out the references to Alcuin's problem collection.

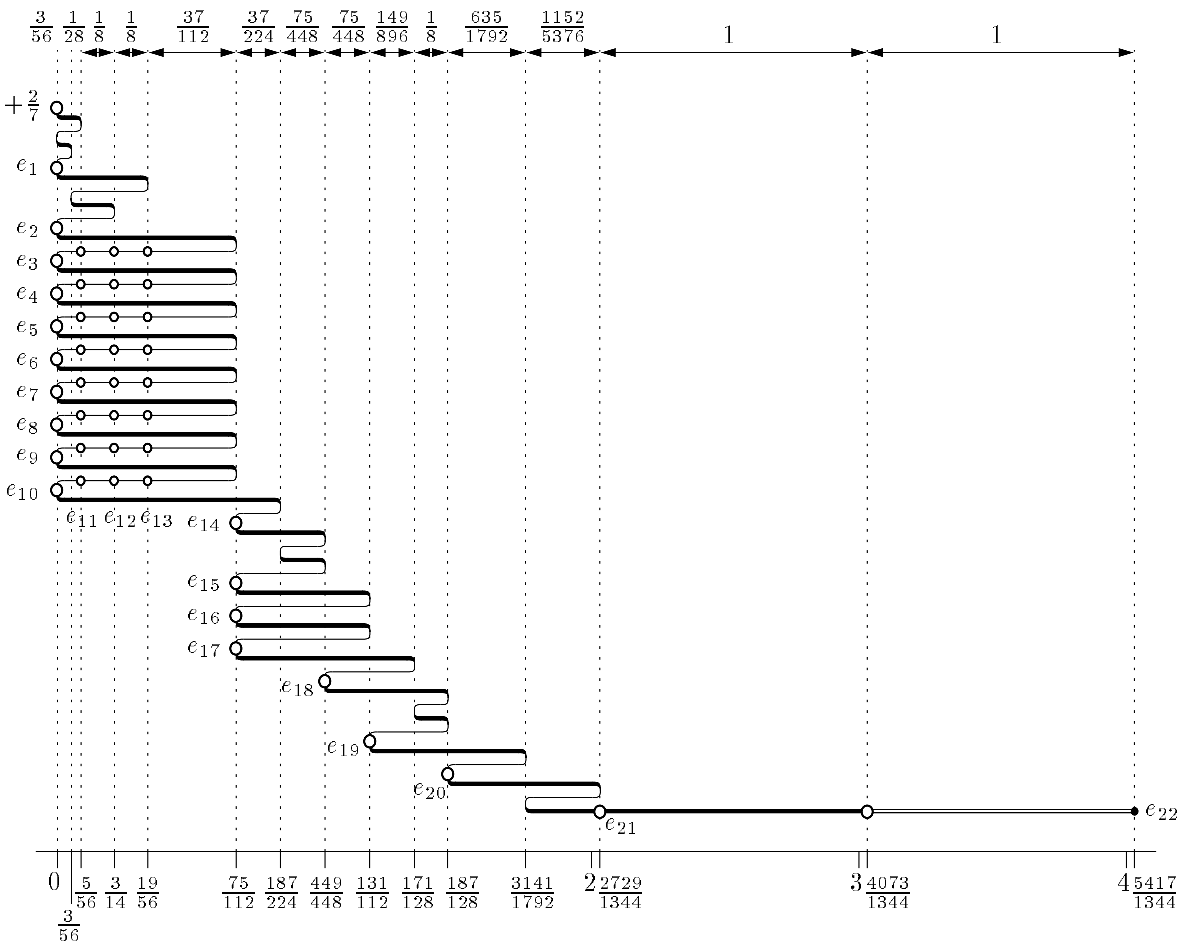

Figure 4: A solution for $n = 22$ cans where partial refilling is allowed. The jeep starts by filling the tank only to a level 2/7 and carrying the partially emptied can to $e_{11} := 5/56$. With the remaining fuel and an additional can, two full cans are deposited at $e_{12} := 12/56$ and $e_{13} := 19/56$. These two cans and the can at e_{11}, which contains 5/7 tankfuls of fuel, are used to support the transportation of 8 full cans from 0 to 75/128. On the back movements of these 8 trips, the jeep takes in just enough fuel at e_{13}, e_{12}, and e_{11} to reach the next depot with an empty tank. In total, the 8 trips use up all the fuel from the three support cans. The remaining tour of the jeep, from e_{10} onward, follows the strategy of Section 2.

References

U. Brauer and W. Brauer [1989]
A new approach to the jeep problem, *Bull. EATCS* No. **38** (June 1989), 145–154.

A. K. Dewdney [1987]
Computer Recreations, *Sci. Amer.* **256**, 6 (June 1987), 106–109. Solutions in **257**, 4 (October 1987), p. 169, and **257**, 5 (November 1987), p. 122. German translation: Computer-Kurzweil, *Spektrum der Wissenschaft*, September 1987, 6–10; January 1988, p. 12; February 1988, p. 13.

Henry E. Dudeney [1917]
Amusements in Mathematics, Thomas Nelson & Sons, 1917. Reprinted by Dover Publications, 1958.

N. J. Fine [1947]
The jeep problem, *Amer. Math. Monthly* **54** (January 1947), 24–31.

M. Folkerts [1978]
Die älteste mathematische Aufgabensammlung in lateinischer Sprache: Die Alkuin zugeschriebenen *Propositiones ad acuendos iuvenes*. Überlieferung, Inhalt, kritische Edition. *Österreich. Akad. Wiss. Math.-Natur. Kl. Denkschr.* **116,2**, (6. Abh.), 13–80.

M. Folkerts [1993]
Die Alkuin zugeschriebenen "*Propositiones ad acuendos iuvenes* in: *Science in Western and Eastern Civilization in Carolingian Times*, ed. P. L. Butzer and D. Lohrmann, Birkhäuser Verlag, Basel 1993, pp. 273–281.

D. Gale [1970]
The jeep once more or jeeper by the dozen, *Amer. Math. Monthly* **77** (1970), 493–501.

D. Gale [1994]
The return of the jeep, *Math. Intelligencer* **16**, No. 1, (Winter 1994), 42–44.

H. Gericke and M. Folkerts [1993]
Die Alkuin zugeschriebenen *Propositiones ad acuendos iuvenes* (Aufgaben zur Schärfung des Geistes der Jugend), in: *Science in Western and Eastern Civilization in Carolingian Times*, ed. P. L. Butzer and D. Lohrmann, Birkhäuser Verlag, Basel 1993, pp. 283–362.

J. Hadley and D. Singmaster [1992]
Problems to sharpen the young, *Math. Gazette* **76** (No. 475, March 1992), 102–126.

A. Hausrath, B. Jackson, J. Mitchem, and E. Schmeichel [1995]
Gale's round-trip jeep problem, *Amer. Math. Monthly* **102** (April 1995), 299–309.

B. Jackson, J. Mitchem, and E. Schmeichel [1995]
A solution to Dewdney's jeep problem, in: *Graph Theory, Combinatorics, and Applications: Proc. 7th Quadrenn. Int. Conf. Theory Appl. Graphs*, Kalamazoo (Michigan), Vol. 1, Y. Alavi and A. Schwenk (eds.), Wiley, 1995.

C. G. Phipps [1947]
The jeep problem: A more general solution, *Amer. Math. Monthly* **54** (October 1947), 458–462.

Derick Wood [1984]
Paradigms and programming with PASCAL, Computer Science Press, Rockville 1984.

A Calculation of the values e_i

We will now prove that the values e_i given by (6) fulfill the inductive definition (4) and (3). We assume $n \geq 6$. (For smaller n, correctness of (6) can be checked by direct computation.) From the definitions in (5) we obtain

$$\begin{aligned} a_0 &= n-1, & b_0 &= 0, & \bar{b}_0 &= 1; & & \\ a_{q+1} &= \frac{a_q}{2}, & b_{q+1} &= b_q, & \bar{b}_{q+1} &= \bar{b}_q + 2^q, & &\text{if } a_q \text{ is even;} \\ a_{q+1} &= \frac{a_q - 1}{2}, & b_{q+1} &= b_q + 2^q, & \bar{b}_{q+1} &= \bar{b}_q, & &\text{if } a_q \text{ is odd.} \end{aligned} \tag{19}$$

First we check the initial conditions. From (19) we have $a_1 = \lfloor (n-1)/2 \rfloor$, and therefore

$$A_0 = 1 \quad \text{and} \quad A_1 = \lfloor n/2 \rfloor + 1.$$

So,

$$\begin{aligned} e_1 &= e_{A_0} = 0 - \lfloor (1/2)^2 \rfloor = 0, \\ e_i &= 0, \qquad \text{for } 1 < i < \lfloor n/2 \rfloor. \end{aligned}$$

Since $b_1 = 0$ or $b_1 = 1$, we have

$$e_{n/2} = e_{A_1 - 1} = 0 + \frac{\lfloor (b_1/2)^2 \rfloor}{4} = 0.$$

If n is odd, $\bar{b}_1 = 2$, and

$$e_{\frac{n+1}{2}} = e_{A_1} = \frac{1}{2} - \frac{\lfloor (2/2)^2 \rfloor}{4} = \frac{1}{4}$$

Thus the initial conditions are fulfilled.

To check that the values in (6) satisfy the inductive definition (4) of e_{n-k} for $k \geq 3$, let's find out the formula which we have to apply in order to compute e_{n-2k-1}, e_{n-2k}, and e_{n-2k+1}, respectively: If $n-k = A_{q+1}$ or $n-k = A_{q+1}-1$ then these formulas depend on the parity of a_q. The following table holds for $n-k \leq n-3$, i. e., $k \geq 3$.

case	$n-k$	parity of a_q	e_{n-2k-1}	e_{n-2k}	e_{n-2k+1}	range of q
1	$\in [A_q+1, A_{q+1}-2]$	arbitrary	$\frac{q-1}{2}$	$\frac{q-1}{2}$	$\frac{q-1}{2}$	$(q \geq 1)$
2	$A_{q+1}-1$	a_q even	$\frac{q-1}{2}$	$\frac{q-1}{2}$	e_{A_q-1}	$(q \geq 1)$
3		a_q odd	$\frac{q-1}{2}$	e_{A_q-1}	e_{A_q}	$(q \geq 1)$
4	A_{q+1}	a_q even	e_{A_q-1}	e_{A_q}	$\frac{q}{2}$	$(q \geq 1)$
5		a_q odd	e_{A_q}	$\frac{q}{2}$	$\frac{q}{2}$	$(q \geq 0)$

The last column gives the range of q for which (4) applies, i. e., $k \leq \lfloor n/2 \rfloor - 1$.

Let's check case 1: We have $n-k \in [A_q+1, A_{q+1}-2]$, or $a_{q+1}+2 \leq k \leq a_q - 1$. Using the inequalities $a_{q+1} \geq a_q - 1/2$ and $a_q \leq a_{q-1}/2$, we obtain

$$n-2k+1 \leq n-2a_{q+1}-3 \leq n-a_q-2 < A_q - 1$$

and

$$n-2k-1 \geq n-2a_q+1 \geq n-a_{q-1}+1 > A_{q-1}.$$

Hence,

$$e_{n-2k-1} = e_{n-2k} = e_{n-2k+1} = \tfrac{q-1}{2}.$$

In case 2, we have $n-k = A_{q+1}-1$ and a_q is even. We must have $q \geq 1$; otherwise, for $q=0$, $A_{q+1}-1 = A_1 - 1 = n - \frac{n+1}{2}$, and the value $k = \frac{n+1}{2}$ is beyond the bound $\lfloor n/2 \rfloor - 1$ for which (4) holds.

Since a_q is even, we have $k = a_{q+1}+1 = a_q/2+1$, and thus $n-2k+1 = n-a_q-1 = A_q-1$, and $n-2k-1, n-2k < A_q - 1$. On the other hand, since $k \geq 3$ we have $a_{q+1} \geq 2$, $a_q \geq 2a_{q+1} \geq 4$, and $a_{q-1} \geq 2a_q \geq a_q+4$. Therefore,

$$n-2k-1 = n-a_q-3 \geq n-a_{q-1}+1 > A_{q-1},$$

and

$$e_{n-2k-1} = e_{n-2k} = \tfrac{k-1}{2}.$$

Cases 3–5 are similar.

Now it is a matter of straightforward computation to check (4). For example, let us consider case 3, which is, together with its symmetric counterpart, case 4, the most difficult case. In order to inductively prove the formula for $e_{A_{q+1}-1}$ when a_q is odd we have to substitute the expressions given by (6) into the right side of the following equation.

$$\begin{aligned} e_{A_{q+1}-1} &= 1/2 + \frac{e_{n-2k-1} + 2e_{n-2k} + e_{n-2k+1}}{4} \\ &= \tfrac{1}{2} + \tfrac{1}{4} \cdot \left[\tfrac{q-1}{2} + 2 \cdot e_{A_q-1} + e_{A_q}\right] \end{aligned}$$

Using the substitution $\bar{b}_q := 2^q - b_q$, the expression in (6) for e_{A_q} can be written in the following more convenient equivalent form:

$$e_{A_q} = \frac{q-1}{2} + \frac{1}{4} + \frac{b_q/2}{2^q} - \frac{\lfloor (b_q/2)^2 \rfloor}{4^q}, \quad \text{for } q \geq 1.$$

This gives

$$\begin{aligned}
e_{A_{q+1}-1} &= \frac{1}{2} + \frac{1}{4} \cdot \left[\frac{q-1}{2} + 2 \cdot e_{A_q-1} + e_{A_q}\right] \\
&= \frac{1}{2} + \frac{1}{4} \cdot \left[\frac{q-1}{2} + 2 \cdot \left(\frac{q-1}{2} + \frac{\lfloor (b_q/2)^2 \rfloor}{4^q}\right)\right. \\
&\qquad\qquad \left. + \left(\frac{q-1}{2} + \frac{1}{4} + \frac{b_q/2}{2^q} - \frac{\lfloor (b_q/2)^2 \rfloor}{4^q}\right)\right] \\
&= \frac{q}{2} + \frac{1}{16} + \frac{b_q}{2^{q+1}} + \frac{\lfloor (b_q/2)^2 \rfloor}{4^{q+1}} \\
&= \frac{q}{2} + \frac{4^q/4 + (2 \cdot 2^q \cdot b_q)/4 + \lfloor b_q^2/4 \rfloor}{4^{q+1}} = \frac{q}{2} + \frac{\lfloor (2^q + b_q)^2/4 \rfloor}{4^{q+1}} = \frac{q}{2} + \frac{\lfloor b_{q+1}^2/4 \rfloor}{4^{q+1}}.
\end{aligned}$$

The last equation uses (19). The final expression equals the claimed value of $e_{A_{q+1}-1}$. The other four cases are handled in the same way.

The above formulas extend only up to e_{n-3}. We still have to compute e_{n-2}, e_{n-1}, and e_n. Substituting $e_{n-2} = \frac{1}{2} + \frac{e_{n-5}}{4} + \frac{e_{n-4}}{2} + \frac{e_{n-3}}{4}$ into (3) gives

$$e_n = 2 + e_{n-1} = 3 + \frac{e_{n-5}}{6} + \frac{e_{n-4}}{3} + \frac{e_{n-3}}{2}. \tag{20}$$

Since e_{n-2} and e_{n-1} are only intermediate values and not directly of interest, we shall prove only the formula for e_n. To find out which formulas in (5) apply to these indices, we again make a case distinction. Since $2^{p+1} < n \leq 2^{p+2}$, the value of $a_p = \lfloor (n-1)/2^p \rfloor$ can be 2 or 3.

Case 1. $a_p = 3$. In this case, $b_p = n - 1 - 3 \cdot 2^p$ and moreover, $A_p = n - 3$, $A_p - 1 = n - 4$, $A_{p-1} \leq n - 6 < n - 5 < A_p - 1$.

$$\begin{aligned}
e_{n-3} = e_{A_p} &= \frac{p}{2} - \frac{\left\lfloor ((2^{p+2} - n + 1)/2)^2 \right\rfloor}{4^p} = \frac{p}{2} - 4 + \frac{n-1}{2^{p-1}} - \frac{\lfloor (n-1)^2/4 \rfloor}{4^p} \\
e_{n-4} = e_{A_p-1} &= \frac{p-1}{2} + \frac{\left\lfloor ((n-1-3 \cdot 2^p)/2)^2 \right\rfloor}{4^p} \\
&= \frac{p-1}{2} + \frac{9}{4} - \frac{3(n-1)}{2^{p+1}} + \frac{\lfloor (n-1)^2/4 \rfloor}{4^p} \\
e_{n-5} &= \frac{p-1}{2}
\end{aligned}$$

Substituting these values into (20) gives (7).

Case 2. $a_p = \lfloor (n-1)/2^p \rfloor = 2$. Hence $A_p - 1 = n-3$ and $b_p = n-1-2^{p+1}$. This gives

$$\begin{aligned} e_{n-3} = e_{A_p-1} &= \frac{p-1}{2} + \frac{\left\lfloor ((n-1-2^{p+1})/2)^2 \right\rfloor}{4^p} \\ &= \frac{p-1}{2} + 1 - \frac{n-1}{2^p} + \frac{\lfloor (n-1)^2/4 \rfloor}{4^p} \end{aligned}$$

We further distinguish two subcases according to the value of $a_{p-1} = \lfloor (n-1)/2^{p-1} \rfloor$, which can be 4 or 5.

Subcase 2.1. $a_{p-1} = 4$. In this case, $A_{p-1} = n-4$, $A_{p-1} - 1 = n-5$, and $b_{p-1} = b_p = n-1-2^{p+1}$, $\bar{b}_{p-1} = 5 \cdot 2^{p+1} - n + 1$.

$$\begin{aligned} e_{n-4} = e_{A_{p-1}} &= \frac{p-1}{2} - \frac{\left\lfloor ((5 \cdot 2^{p-1} - n + 1)/2)^2 \right\rfloor}{4^{p-1}} \\ &= \frac{p-1}{2} - \frac{25}{4} + \frac{5(n-1)}{2^p} - \frac{\lfloor (n-1)^2/4 \rfloor}{4^{p-1}} \end{aligned}$$

$$\begin{aligned} e_{n-5} = e_{A_{p-1}-1} &= \frac{p-2}{2} + \frac{\left\lfloor ((n-1-2^{p+1})/2)^2 \right\rfloor}{4^{p-1}} \\ &= \frac{p-2}{2} + 4 - \frac{n-1}{2^{p-2}} + \frac{\lfloor (n-1)^2/4 \rfloor}{4^{p-1}} \end{aligned}$$

Again, substituting these values of e_{n-5}, e_{n-4}, and e_{n-3} into (20) gives (7).

Subcase 2.2. $a_{p-1} = 5$. In this case, $A_{p-1} = n-5$, $A_{p-1} < n-4 < A_p - 1$, and $\bar{b}_{p-1} = 3 \cdot 2^{p+1} - n + 1$. This gives $e_{n-4} = \frac{p-1}{2}$ and

$$e_{n-5} = e_{A_{p-1}} = \frac{p-1}{2} - \frac{\left\lfloor ((3 \cdot 2^p - n + 1)/2)^2 \right\rfloor}{4^{p-1}} = \frac{p-1}{2} - 9 + \frac{3(n-1)}{2^{p-1}} - \frac{\lfloor (n-1)^2/4 \rfloor}{4^{p-1}}.$$

Again, we substitute into (20) and obtain formula (7) for e_n.

B Sufficiency of the conditions

In this section we show that the constraints (13) and (14) which we used to derive an upper bound on L_n are sufficient for characterizing the possible empty can positions e_i of feasible trips. In fact we will show that this characterization is even achieved by a very restricted set of $O(n)$ linear inequalities including (13), (16), and the relations $e_i \le e_{i+1}$.

It is clear that in a feasible solution there cannot be a gap larger than 1 between e_i and e_{i+1}, since the jeep cannot cross an interval larger than 1 without opening a can. We show that this condition arises as a consequence of (13) and (16).

Lemma 4 *If* $0 = e_1 \le e_2 \le \cdots \le e_{n-1} \le e_n - 1$ *satisfies* (13) *and* (16), *then*

$$\begin{gathered} e_{i+1} \le e_i + 1, \text{ for } i = 1, \dots, n-2, \text{ and} \\ e_n \le e_{n-1} + 2. \end{gathered}$$

Proof. By setting $k = 2$ in (16) we directly obtain $e_n \le e_{n-1} + 2$. If $\frac{n+1}{2} \le i \le n-2$, we take (16) with $k := 2n - 2i$ and we obtain

$$-e_{2i-n+1} - \sum_{j=2i-n+2}^{i} 2e_j + \sum_{j=i+1}^{n-1} 2e_j + e_n \le 2n - 2i.$$

Using the fact that $e_j \le e_i$ for $j \le i$, $e_j \ge e_{i+1}$ for $j \ge i+1$, and $e_n \ge e_{i+1} + 1$, we get

$$-(2n - 2i - 1)e_i + (2n - 2i - 1)e_{i+1} + 1 \le 2n - 2i,$$

which clearly implies $e_{i+1} \le e_i + 1$. For $1 \le i \le n/2$, we argue similarly, starting from (13):

$$n \ge \sum_{j=2}^{n-1} 2e_j + e_n \ge \sum_{j=i+1}^{n-1} 2e_j + e_n \ge (2n - 2i - 1)e_{i+1} + 1 \ge (n-1)e_{i+1} + 1$$

This gives $1 \ge e_{i+1} \ge e_{i+1} - e_i$. ■

Theorem 2 *Let $e_1, \dots, e_n$ be any feasible solution of the inequalities* (13), (16), *and $0 = e_1 \le e_2 \le \cdots \le e_{n-1} \le e_n - 1$. Then there is a feasible trip of the jeep which reaches e_n, leaving empty cans at positions $e_1, \dots, e_n$.*

Proof. We will prove this by an algorithm which constructs a trip for the jeep, and we will show its correctness with the help of some intermediate lemmas.

We plan the jeep's trip move by move. (A typical move will look as shown in Figure 5 below.) However, the tasks belonging to a single move *are not planned in the sequence in which they are finally carried out.* We rather start with a required initial move of the form $e_i \to y \to e_{i+1}$. Afterwards we extend the move by giving the jeep additional tasks on its way.

The quantity s will denote the fuel which is still available for new tasks of this move. These new tasks are planned by sweeping from left to right. During this sweep, s decreases with the amount of driving that is planned.

The algorithm tries to establish depots of double-cans at the positions $e_2, e_3, \dots$. All remaining cans which have not reached their current destination in some depot e_i are at some common position y, which advances from one move to the next. Between two moves the jeep and the full cans are in one of three well-structured situations, which are described below.

It may happen that there is still fuel available ($s > 0$) but there is no further work for the jeep to be done on this move, and so the remaining fuel is wasted. A variable *waste* accumulates the total amount of wasted fuel.

Description of a move. We now specify how to construct the i-th move, which starts at e_i and ends at e_{i+1}, for $1 \le i \le n-2$. We assume that the jeep is at position e_i with an empty tank. The description will be somewhat informal, since we will let a variable y vary continuously, and other variables will vary as a function of y until something happens. The reader who is unsure about the precise meaning of the following description may consult the realization of the algorithm in Maple in Appendix C.

Case 1. There are double-cans at positions $e_i, e_{i+1}, \ldots, e_l$ with $l \geq i+1$. In addition there are $n_y > 0$ cans at some position y with $e_l \leq y \leq e_{l+1}$.

1.1. Plan the initial move $e_i \to y \to e_{i+1}$: fill the tank and load a can at e_i; move the can to y; and return to e_{i+1}.

Let $s := 1 - (y - e_i) - (y - e_{i+1})$. [We will show below that $s \geq 0$, and hence the initial move is possible.]

Set $n_y := n_y + 1$. This accounts for the additional can brought by the initial move. [Now we have $n_y = n - 2l + i$.]

1.2. While $s > 0$ and $n_y > 0$, move all n_y cans forward, increase y, and decrease s at speed $2n_y$. This means that an increase from y to $y + \Delta y$ will cause a decrease of s from s to $s - 2n_y \cdot \Delta y$. The increase of y and decrease of s continues until one of the following two events occurs.

- If s becomes 0, then stop. Set $i := i + 1$ and plan the next move. If we still have $i < l$, we continue with Case 1; otherwise the next move will be in Case 2.
- If y reaches e_{l+1} and $n_y > 2$, then drop a double-can there: set $l := l + 1$, $n_y := n_y - 2$, and continue. [During the sweep, we alway maintain the condition $n_y = n - 2l + i$.] However, if $n_y \leq 2$, then n_y would become zero or negative at this point. In this case we cannot continue. We set $\mathit{waste} := \mathit{waste} + s$; $i := i + 1$, and go to the next move.

Case 2. There is a double-can at position e_i, and $n_y > 0$ cans at some position y with $e_i \leq y \leq e_{i+1}$. [We have $n_y = n - i - 1$.]

2.1. Plan the initial move $e_i \to e_{i+1}$: carry a can from e_i to e_{i+1}. Let $s := 1 - (e_{i+1} - e_i)$. By Lemma 4, $s \geq 0$.

2.2. While $s > 0$, increase y, decreasing s at speed $2n_y$.

- If s becomes 0 before y reaches e_{i+1}, (i. e., if $s/(2n_y) \leq e_{i+1} - y$) then stop. Set $i := i + 1$ and plan the next move. (The next move will be in Case 3.)
- If y reaches e_{i+1}, and if $n_y > 1$, then drop a can there. (Another can was placed there during the initial move.) Set $n_y := n_y - 1$, $l := i + 1$, and continue as in Step 1.2.
  However, if $n_y = 1$, we stop at this point $y = e_{i+1}$. We set $\mathit{waste} := \mathit{waste} + s$; $i := i + 1$, and continue with the next move. (This will happen if $i = n - 2$.)

Case 3. There is a single can at position e_i, and $n_y > 0$ cans at some position $y \leq e_i$. [We have $n_y = n - i$.]

3.1. Plan the initial move $e_i \to y \to e_{i+1}$: fill the tank at e_i; move back to y and load a can there; bring the can forward to e_{i+1}. Set $n_y := n_y - 1$. [We have $n_y = n - i - 1$, as in Step 2.1.]

Let $s := 1 - (e_i - y) - (e_{i+1} - y)$. If s is negative then stop. There is no solution.

3.2. This is identical to Step 2.2.

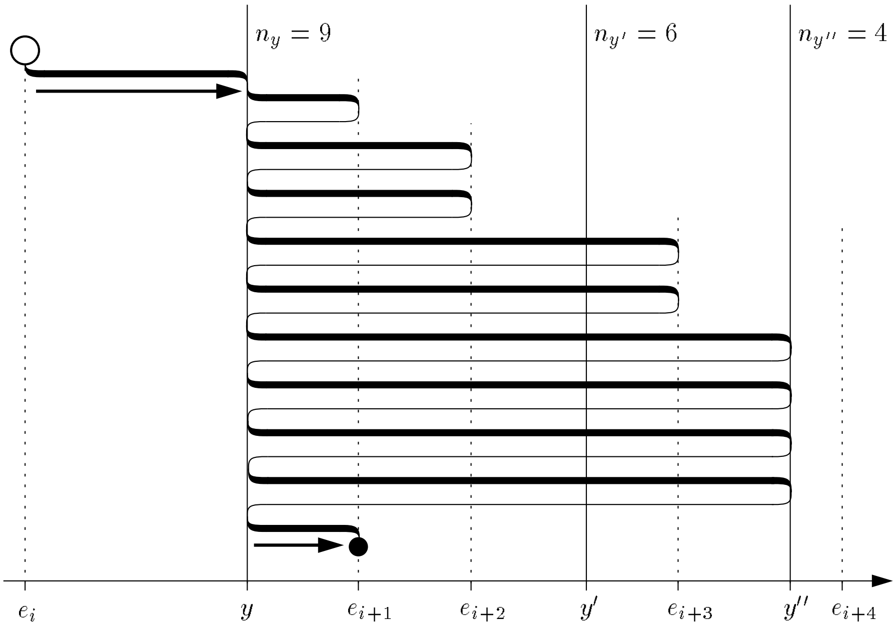

Figure 5: A typical move constructed be the algorithm. This i-th move starts at e_i in Case 2 with $n_y = 9$. During the planning of the move, y advances over y' to its final value y'', and n_y decreases to $n_{y''} = 4$ on the way. In the end, we have $l = i+3$, and the $(i+1)$-st move will start in Case 1. The parts which are accompanied by a thick arrow constitute the "initial move".

We initialize the algorithm by setting $i := 1$, $y := 0$, $n_y := n - 2$, *waste* $:= 0$. The starting situation is Case 2. It is straightforward to verify that at the end of each move, the situation of the jeep is again as described in one of the three cases, and that the expressions for n_y in terms of i and l given in brackets remain valid during the algorithm, and n_y remains positive.

The last two moves must be modified a little. For $i = n-1$, one can only arrive in Case 3, with one can at position y, because the other cases would lead to $n_y \le 0$. The $(n-1)$-st move is of the form $e_{n-1} \to y \to e_n - 1$, like in Step 3.1. This move is only possible if $s := 1 - (e_{n-1} - y) - (e_n - 1 - y) \ge 0$. The last move goes straight from $e_n - 1$ to e_n. It is always possible.

Each move of the jeep is initially of the form $e_i \to y \to e_{i+1}$. In addition, the jeep may have to transport a number of cans from y forward, "dropping" some cans on the way. So the jeep interrupts the normal course of the initial move upon reaching y, carries out the additional operations, and resumes the initial move at the end. The number of additional cans is decreasing as y increases, and so there is no problem in carrying out the additional transport. A typical move generated by the algorithm is shown in Figure 5. The program in Appendix C gives details about the construction of the move.

We will first show that, unless the algorithm stops in Step 3.1, it produces a feasible solution. In the end we will demonstrate that the algorithm can only stop in Step 3.1 if one of the conditions (16) is violated.

Lemma 5 *In Step* 1.1*, s does not become negative.*

Proof. The previous move has started from e_{i-1}, reached y, and terminated in e_i. Therefore,

$$1 \geq (y - e_{i-1}) + (y - e_i) \geq (y - e_i) + (y - e_{i+1}). \qquad \blacksquare$$

The total amount of driving is at least $g(0)$, where $g(0)$ is given by (12). Since there are n cans available, it is clear that at most $n - g(0)$ cans of fuel can be wasted. As soon as we have wasted more than this amount, we are sure to fail. The following lemma shows that the moves constructed by our algorithm will never waste more than this bound.

Lemma 6 *During the algorithm, the total waste is always bounded by $n - g(0)$.*

Proof. Assume that a move terminates in Step 1.2 with $s > 0$. The total amount of fuel which has been used equals i cans minus the total waste so far. This must be equal to the amount B of driving done. We always have $n_y = n - 2l + i$ and this must be 1 or 2 upon reaching e_{l+1}. If $n + i$ is even, we have $l + 1 = (n + i)/2$, and the last value of n_y is 2. If $n + i$ is odd, we have $l + 1 = (n + i + 1)/2$, and the last value of n_y is 1.

The quantity B can now be computed by observing how often the jeep has gone back and forth between each e_i and e_{i+1}, as in the derivation of (8).

$$\begin{aligned} B &= (2n-1)e_1 + (2n-3)(e_2 - e_1) + \cdots + \big(2(n-i)-1\big)(e_{i+1} - e_i) \\ &\quad + \big(2(n-i)-4\big)(e_{i+2} - e_{i+1}) + \big(2(n-i)-8\big)(e_{i+3} - e_{i+2}) + \cdots \\ &= 2e_1 + 2e_2 + \cdots + 2e_i + 3e_{i+1} + 4e_{i+2} + \cdots \begin{cases} +\,4e_{l+1}, & \text{if } n+i \text{ is even} \\ +\,4e_l + 2e_{l+1}, & \text{if } n+i \text{ is odd} \end{cases} \end{aligned}$$

We are interested in the quantity $g(0) - B$. If $n + i$ is even, we have

$$g(0) - B = -e_{i+1} - 2e_{i+2} - \cdots - 2e_{l+1} + 2e_{l+2} + \cdots + 2e_{n-1} + e_n \leq n - i.$$

This follows by taking $k = n - i$ in (16). If $n + i$ is odd, we have similarly

$$g(0) - B = -e_{i+1} - 2e_{i+2} - \cdots - 2e_l + 2e_{l+2} + \cdots + 2e_{n-1} + e_n \leq n - i.$$

The last inequality may be derived by taking (16) for $k = n-i-1$ and for $k = n-i+1$. The average of these two inequalities together with $e_i \leq e_{i+2}$ gives the above relation. Summarizing, we have $g(0) - B \leq n - i$ and $B \leq i - \textit{waste}$, which gives the desired result. $\blacksquare$

Now the only thing that can happen is that the algorithm terminates with failure in Step 3.1 of the i-th move, for $2 \leq i \leq n - 1$. In this case we compare the amount of driving which was done with the amount of fuel spent, as above, and we will obtain a contradiction. Let us first assume that $i \le n-2$.

The situation is as follows. The total amount of fuel spent is $i - \mathit{waste}$. With this fuel, the jeep has moved cans to $e_1, e_2, \ldots, e_i$ (they are now empty), and it has moved $n_y = n - i$ additional cans to y; but the jeep did not succeed to bring the last can from y to e_{i+1}. Thus, if we increase y to $(e_i + e_{i+1} - 1)/2$ and assume that a can was brought to e_{i+1}, we get an amount of driving C which is more work than the jeep has done:

$$C > i - \mathit{waste} \ge i - n - g(0), \tag{21}$$

by Lemma 6. With $y := (e_i + e_{i+1} - 1)/2$ we can now write

$$\begin{aligned} C &= 2e_1 + 2e_2 + \cdots + 2e_i + e_{i+1} \ + \ 2(n-i-1)y \\ &= 2e_1 + 2e_2 + \cdots + 2e_{i-1} + (n-i+1)e_i + (n-i)e_{i+1} - (n-i-1). \end{aligned}$$

Combining this with (21) gives

$$\begin{aligned} 0 &< C - g(0) + n - i \\ &= (n-i-1)e_i + (n-i-2)e_{i+1} - 2e_{i+2} - \cdots - 2e_{n-1} - (e_n - 1) < 0, \end{aligned}$$

using the fact that $e_i \le e_{i+1} \le \cdots \le e_{n-1} \le e_n - 1$. This is a contradiction.

For the $(n-1)$-st move, the role of e_{i+1} in the expression for C is taken by $e_n - 1$ instead of e_n, and a suitably modified calculation leads again to a contradiction. This ends the proof of Theorem 2. ■

It is remarkable that the very small set of constraints (13) and (16), together with the ordering of e_i, is sufficient to characterize feasible solutions. The full generality of (14) is not necessary.

Corollary 1 *Any feasible solution of Problem* (15) *gives rise to a journey of the jeep which reaches* e_n.

Proof. Since the conditions (16) were derived from (14), the assumptions of Theorem 2 follow from the constraints of Problem (15). ■

C Program for the algorithm of Appendix B

The following program implements the algorithm described above. Given the number n of cans, and the sorted array $e_1, \ldots, e_n$ of desired final position of empty cans, it constructs a feasible tour, or otherwise it prints an error message exhibiting one of the conditions which is violated.

The program is written in the programming language Maple, but it should be understandable by anybody who is familiar with some procedural programming language.

The three cases are encoded by the variable l which is defined by the condition that $e_{l+1}, e_{l+2}, \ldots$ are the positions beyond e_i at which no double-cans have yet been established. The result is a sequence of "commands" (procedure calls) which tell the jeep what to do, in the correct order. The program assumes that the conditions $e_1 = 0$, $0 \le e_i - e_{i+1} \le 1$, and $1 \le e_{n-1} - e_n \le 2$ have been checked beforehand.[1]

```
max_waste := n − 2 ∗ sum(e[i], 'i' = 1 .. n − 1) − e[n];
if max_waste < 0 then ERROR('(13) is violated.') fi;
y := 0; n_y := n − 2; waste := 0; l := 1;  # The jeep starts at e[1] = 0.
for i from 1 to n − 2 do
   fill_tank();
   s := 1 − abs(e[i] − y) − abs(e[i + 1] − y);
   # carry out the first part of the initial move, up to y:
   if l > i then      # CASE 1.  e[i + 1] ≤ e[l] ≤ y
       load_can(); move_to(y); unload_can();
       n_y := n_y + 1;
   elif l = i then    # CASE 2.  e[i] ≤ y ≤ e[i + 1]
       load_can(); move_to(y); unload_can();
   else               # CASE 3.  l = i − 1 and y ≤ e[i]
       move_to(y);
       l := l + 1; n_y := n_y − 1;
   fi;

   # plan additional work:
   y_old := y;
   while s > 0 do
      Delta_y := min(e[l + 1] − y, s/(2 ∗ n_y));
      y := y + Delta_y;
      s := s − 2 ∗ Delta_y ∗ n_y;
      if s = 0 or
        n_y ≤ 1 or (l > i and n_y ≤ 2) then break fi; # exit the while loop.
      load_can(); move_to(y); unload_can(); move_to(y_old);
      n_y := n_y − 1;
      if l > i then
          load_can(); move_to(y); unload_can(); move_to(y_old);
          n_y := n_y − 1;
      fi;
      l := l + 1;
   od;
   for j from 1 to n_y do  # We always have n_y > 0.
       load_can(); move_to(y); unload_can(); move_to(y_old);
   od;

   waste := waste + s;
   if waste > max_waste then
```

[1]The PostScript file from which the copy of the paper which you are now looking at was generated contains the text of a Maple procedure which carries out these statements, together with some auxiliary procedures, near its beginning.

```
    if (n − i) mod 2 = 0 then ERROR('(16) is violated for k = n − i.');
    else ERROR('(16) is violated for k = n − i − 1 or for k = n − i + 1.');
  fi fi;

  # carry out the final part of the initial move, from y_old:
  if y_old < e[i + 1] then load_can(); move_to(e[i + 1]); unload_can();
  else move_to(e[i + 1]);
  fi;
od;
# the last two moves:
s := 1 − abs(e[n − 1] − y) − abs(e[n] − 1 − y);
waste := waste + s;
fill_tank(); move_to(y); load_can(); move_to(e[n] − 1);
fill_tank(); move_to(e[n]); unload_can();
```