The Capacity-C Torch Problem

Roland Backhouse

rcb@cs.nott.ac.uk
School of Computer Science University of Nottingham, Nottingham NG8 1BB, England

Abstract. The torch problem (also known as the bridge problem or the flashlight problem) is about getting a number of people across a bridge as quickly as possible under certain constraints. Although a very simply stated problem, the solution is surprisingly non-trivial. The case in which there are just four people and the capacity of the bridge is two is a well-known puzzle, widely publicised on the internet. We consider the general problem where the number of people, their individual crossing times and the capacity of the bridge are all input parameters. We present an algorithm that determines the shortest total crossing time; the number of primitive computations executed by the algorithm (i.e. the worst-case time complexity of the algorithm) is proportional to the square of the number of people.
Keywords: algorithm derivation, shortest path, dynamic programming, algorithmic problem solving

The (capacity-C) torch problem is as follows.

> N people wish to cross a bridge. It is dark, and it is necessary to use a torch when crossing the bridge, but they only have one torch between them. The bridge is narrow and at most C people can be on it at any one time. The people are numbered from 1 thru N. Person i takes time $t.i$ to cross the bridge; when a group of people cross together they must all proceed at the speed of the slowest.
> Construct an algorithm that will get all N people across in the shortest time. Provide a clear justification that the algorithm does indeed find the shortest time.

The torch problem is an abstraction from a problem involving four people wishing to cross a bridge of capacity two and with specific concrete times. In this form, the problem is believed to have first appeared in 1981. Rote [3] gives a comprehensive bibliography.

The main interest in the torch problem is that what is "obvious" or "intuitive" is often wrong. For example, the "obvious" solution of letting the fastest person repeatedly accompany $C-1$ people across the bridge is wrong. (If $N=4$, $C=2$ and the travel times are 1, 1, 2 and 2, this solution takes time 7 whereas

the shortest crossing time is 6[1].) Also, the "obvious" property that the shortest time is achieved when the number of crossings is minimised is incorrect. (If $N=5$, $C=3$ and the travel times are 1, 1, 4, 4 and 4, the shortest time is 8, which is achieved using 5 crossings. The shortest time using 3 crossings is 9.) It is not difficult to determine an *upper bound* on the crossing time, even in the general case. Nor is it difficult to provide counterexamples to incorrect solutions. The difficulty is to establish an irrefutable *lower bound* on the crossing time. A proper solution to the problem poses a severe test of our standards of proof.

In our solution, we assume that the people are ordered so that $t.i < t.j$ if $i < j$. If the given times are such that $t.i = t.j$ for some i and j, where $i < j$, we can always consider pairs $(t.i\,,\, i)$, where i ranges over people, ordered lexicographically. Renaming the crossing "times" to be such pairs, we obtain a total ordering on times with the desired property[2]. We also assume that N is at least $C{+}1$. (When N is at most C, it is obvious that exactly one crossing gives the optimal solution. When N is at least $C{+}1$, more than one crossing is required.)

For brevity, some of the more straightforward proofs at the beginning of the paper. A full version of the paper, which includes the details of all proofs, is available from the author's website.

1 Outline Strategy

An outline of our solution is as follows.

We call a sequence of crossings that gets everyone across in accordance with the rules a *putative sequence*. We will say that one putative sequence *subsumes* another putative sequence if the time taken by the first is at most the time taken for the second. Note that the subsumes relation is reflexive (every putative sequence subsumes itself) and transitive (if putative sequence a subsumes putative sequence b and putative sequence b subsumes putative sequence c then putative sequence a subsumes putative sequence c). An *optimal* sequence is a putative sequence that subsumes all putative sequences. A putative sequence is *suboptimal* if it is not optimal. The problem is to find an optimal sequence.

Recall that, when crossing the bridge, the torch must always be carried. This means that crossings, both of groups of people and of each individual person, alternate between "forward" and "return" trips, where a *forward trip* is a crossing in the desired direction, and a *return trip* is a crossing in the opposite direction.

A *regular forward trip* means a crossing in the desired direction *made by at least two people*, and a *regular return trip* means a trip in the opposite direction *made by exactly one person*. A *regular sequence* is a putative sequence that consists entirely of regular forward and return trips.

[1] Our examples are chosen so that it is easy for the reader to discover the fastest crossing time. Of course, the examples in puzzle books are deliberately chosen to make it difficult.

[2] Strictly, we also need to extend addition to pairs. Defining $(t, i){+}(u, j)$ to be $(t{+}u\,,\, i{\downarrow}j)$ guarantees the appropriate algebraic structure, in particular distributivity of addition over minimum [2].

The first step (lemma 1) is to show that every optimal putative sequence is regular. The significance of this is threefold. First, it means that the search space for an optimal solution is finite. (This is because a forward trip followed by a return trip reduces the number of people at the start; hence there are at most $N-1$ forward trips in any regular sequence.) Second, the time taken by a regular putative sequence can be evaluated knowing only which forward trips are made. (Knowing the bag of forward trips, it is easy to determine how many times each person makes a return trip. This is because each person makes one fewer return trips than forward trips. In this way, the time taken for the return trips can be calculated.) Finally, and most importantly, knowing just the bag of forward trips in a regular putative sequence is sufficient to reconstruct a regular putative sequence. This is proved in theorem 1. Since all such sequences take the same total time, we can replace the problem of finding an optimal sequence of forward and return trips by the problem of finding an optimal bag of forward trips.

Finding an optimal bag of forward trips begins by establishing a number of lemmas with the goal of reducing the size of the search space. Subsequently, we can formulate the problem as, essentially, a shortest-path problem on an acyclic graph. More precisely, we present a collection of equations each of which corresponds to a component of an algorithm for non-deterministically constructing a bag of forward trips. By calculating the (unique) solution to these equations, we can resolve the non-determinacy in the construction and so obtain an optimal bag of forward trips. Then theorem 1 is applied to obtain a regular sequence that optimises the total travel time. The number of terms in the collection of equations is quadratic in the number of people and cubic in the capacity of the bridge, from which we can deduce the worst-case solution time.

2 Terminology

Let us suppose a putative sequence is given. By extracting just the forward trips in the sequence and ignoring the order in which they are made, we obtain a bag (multiset) of non-empty sets. We use F to denote such a bag. Note that a bag is a set with multiplicities. By a slight abuse of notation, we write $T{\in}F$ and call T an *element* of F if T is an element of the set underlying bag F; we also write $\#_F T$ for the multiplicity of T in the bag F. The bag is completely defined by listing its elements together with their multiplicities.

Since everyone must cross at some time, the bag F satisfies the property that

(1) $\quad \langle \forall i \,:\, 1 \leq i \leq N \,:\, \langle \exists T : T{\in}F : i{\in}T \rangle \rangle \quad .$

Also, since each forward trip is non-empty and the capacity of the bridge is C,

(2) $\quad \langle \forall T \,:\, T{\in}F \,:\, 1 \leq |T| \leq C \rangle \quad .$

From the bag F, we can determine the number of times each individual makes a forward trip. This is given by the function f which is defined by

(3) $\quad f_F.i \;=\; \langle \Sigma T : i{\in}T : \#_F T \rangle \quad .$

The number of times that each person returns is given by the function r; since each person makes one more forward trip than return trip, we have

(4) $\quad r_F.i \;=\; f_F.i - 1 \quad .$

We distinguish two types of person:

(a) Someone who never makes a return trip is called a *settler*.

$$settler_F.i \quad \equiv \quad r_F.i = 0 \quad .$$

(b) Someone who does make a return trip is called a *nomad*.

$$nomad_F.i \quad \equiv \quad r_F.i > 0 \quad .$$

(Note that $settler_F.i \not\equiv nomad_F.i$.) We further subdivide the settlers into "pure" and "mixed" settlers.

(a) A *pure* settler is a settler who crosses with (only) other settlers.

$$pure_F.i \quad \equiv \quad \langle \forall T : T{\in}F \wedge i{\in}T : \langle \forall j : j{\in}T : settler_F.j \rangle \rangle \quad .$$

(b) A *mixed* settler is a settler who crosses with at least one nomad.

$$mixed_F.i \quad \equiv \quad settler_F.i \;\wedge\; \langle \exists\, T,j : T{\in}F \wedge i{\in}T \wedge j{\in}T : nomad_F.j \rangle \quad .$$

Correspondingly, we divide the forward trips into "pure", "mixed" and "nomadic".

(a) A *pure trip* is a forward trip in which everyone involved is a settler.

$$pure_F.T \quad \equiv \quad \langle \forall j : j{\in}T : settler_F.j \rangle \quad .$$

(b) A *mixed trip* is a forward trip involving both settlers and nomads.

$$mixed_F.T \quad \equiv \quad \langle \exists i,j : i{\in}T \wedge j{\in}T : settler_F.i \wedge nomad_F.j \rangle \quad .$$

(c) A *nomadic trip* is a forward trip in which everyone involved is a nomad.

$$nomadic_F.T \quad \equiv \quad \langle \forall i : i{\in}T : nomad_F.i \rangle \quad .$$

A *full* trip is a forward trip in which C people cross. That is, the trip has no spare capacity.

$$full.T \quad \equiv \quad |T| = C \quad .$$

The *leader* of a trip is the slowest person in the trip[3]:

$$lead.T \quad = \quad \langle \Uparrow i : i{\in}T : i \rangle \quad .$$

Mixed and pure trips have multiplicity 1 in the bag F, and each settler is an element of exactly one element of F. It is therefore possible to define a function from settlers to people which identifies the slowest person in the trip made by the settler. Let us call this function $boss_F$. Then the defining property of $boss_F$ is

$$\langle \forall\, i,T : settler_F.i \wedge T{\in}F \wedge i{\in}T : lead.T = boss_F.i \rangle \quad .$$

For nomads, the function $boss_F$ is undefined.

[3] The symbols $\Uparrow$ and $\Downarrow$ denote the maximum and minimum quantifiers, respectively; the symbols $\uparrow$ and $\downarrow$ denote the binary maximum and minimum operators.

3 Regular Sequences

Recall that a "regular" sequence is a sequence in which each forward trip involves at least two people and each return trip involves exactly one person. The following lemma restricts attention to just the regular sequences. The proof is omitted.

Lemma 1. Every putative sequence containing irregular trips is suboptimal. □

3.1 Scheduling Forward Trips

In view of lemma 1, we now consider bags of forward trips that correspond to regular putative sequences. Suppose F is such a bag. Then, with the function r defined by (4), the total time taken by the sequence is

$$(5) \quad \langle \Sigma T \,:\, T{\in}F \,:\, \langle \Uparrow i : i{\in}T : t.i\rangle \times \#_F T\rangle \;+\; \langle \Sigma i :: t.i \times r_F.i\rangle \quad .$$

(Forward trip T takes time $\langle \Uparrow i : i{\in}T : t.i\rangle$ and has multiplicity $\#_F T$, and person i makes $r_F.i$ return trips each of which takes time $t.i$ because the sequence is regular.) Note that the total time is independent of the order in which the trips are scheduled.

Also, since the number of forward trips is $|F|$ and each return trip is undertaken by exactly one person,

$$(6) \quad |F| \;=\; \langle \Sigma i :: r_F.i\rangle + 1 \quad .$$

In a regular sequence, each forward trip involves at least 2 and at most C people, thus sharpening property (2):

$$(7) \quad \langle \forall T \,:\, T{\in}F \,:\, 2 \le |T| \le C\rangle \quad .$$

Finally, as before, each person must cross at least once:

$$(8) \quad \langle \forall i \;:\; 1 \le i \le N \;:\; 1 \le f_F.i\rangle \quad .$$

Crucially, given a bag of sets, F, such that properties (6), (7) and (8) hold of F, it is always possible to construct a regular putative sequence S such that the bag of forward trips in S is F. To establish this theorem, we first prove several properties relating the number of pure trips, the number of nomads and the number of non-pure trips in F.

To this end, we define the functions n ("number of nomads"), nc ("nomad count"), rc ("return count"), sc ("settler count"), np ("the number of non-pure trips") and pc ("pure-trip count") as follows. In the definitions, G is an arbitrary bag of sets, and T ranges over elements of G. The multiplicity of T in G is denoted by $\#_G T$.

$$(9) \qquad n_G = \langle \Sigma i : nomad_G.i : 1\rangle$$

$$(10) \quad nc_G.T = \langle \Sigma i : nomad_G.i \wedge i{\in}T : 1\rangle$$

(11) $rc_G = \langle \Sigma i :: r_G.i \rangle$

(12) $sc_G.T = \langle \Sigma i : settler_G.i \wedge i \in T : 1 \rangle$

(13) $np_G = \langle \Sigma T : \neg(pure_G.T) : \#_G T \rangle$

(14) $pc_G = \langle \Sigma T : pure_G.T : 1 \rangle$

(Note that pure trips always have a multiplicity of 1.)

The following lemma and its corollary identify some straightforward relations between the various counts. Note that lemma 2 is true of all bags, whereas corollary 1 exploits a relation between the size of the bag and its return count.

Lemma 2. Suppose G is a bag of sets. Then

(15) $n_G = 0 \;\equiv\; rc_G = 0$,

(16) $n_G \leq rc_G$,

(17) $rc_G \;=\; \langle \Sigma T :: nc_G.T \times \#_G T \rangle \;-\; n_G$,

(18) $np_G \neq 1$.

□

Corollary 1. If G is a bag of sets such that $|G| = rc_G + 1$ then

(19) $n_G = 0 \;\equiv\; |G| = 1$.

(20) $n_G = 1 \;\Rightarrow\; \langle \forall T :: \neg(pure_G.T) \rangle$.

□

In general, the implication in (20) cannot be strengthened to equivales. For example, the bag G equal to $\{\{1,3\},\{1,2,4\},\{2,5\}\}$ satisfies the property that $|G| = rc_G + 1$ and every trip in G is non-pure. However, the set of nomads in G is $\{1,2\}$. That is, $n_G \neq 1$. The converse implication does hold for the bag of forward trips corresponding to an optimal putative sequence.

Theorem 1. Suppose F is a bag of sets satisfying (6), (7) and (8). Then there is a regular putative sequence of which the bag of forward trips equals F.[4]
Proof Consider the algorithm below. It constructs a sequence S of forward and return trips. On termination, the bag of forward trips defined by S (denoted by $ForwardBag.S$ in the algorithm) equals F.

The symbol ε denotes the empty sequence and $S +\!\!+ S'$ denotes a sequence obtained by appending a sequence S' to the end of S. The sequence S' begins with the trip T and has total length $2 \times nc_G.T$. The trip T is followed in S' by a sequence of alternating return and forward trips, beginning and ending with a return trip. The return trips are made by the $nc_G.T$ nomads in T, the order being arbitrary; the forward trips are all pure, their choice is also arbitrary. The

[4] Thanks to Arjan Mooij for providing the key insight in the proof of this theorem.

choice of T at each iteration (indicated by the "$[\!]T$" quantification[5]) is a non-pure trip with the property that $nc_G.T \leq pc_G + 1$. Note that whether or not a trip is pure is evaluated with respect to the bag G and not the bag F. The guard on the choice of T guarantees that S' can be constructed from the elements of G. The removal of one occurrence of T and the pure trips in S' from the bag G results in the bag denoted by $G \ominus S'$. The symbol "$\dot{\cup}$" in the invariant denotes bag union.

The invariant is truthified by the initialisation because F satisfies (6). It is also maintained by the loop body because $|G|$ is decreased by $nc_G.T$ and, simultaneously, $r_G.i$ is decreased by 1 for $nc_G.T$ instances of i; also, the forward trips added to the sequence S are precisely the trips removed from the bag G.

On termination of the loop, we claim that G has size 1; the sequence S is concluded by the one trip remaining in G.

$$
\begin{array}{l}
S,G \;:=\; \varepsilon,F \;; \\
\{\;\; \textbf{Invariant}\text{:} \quad |G| \;=\; rc_G + 1 \quad \wedge \quad F = G \,\dot{\cup}\, ForwardBag.S \;\;\} \\
\mathsf{do}\;\; \langle [\!] T \\
\qquad : \quad T \in G \wedge \neg(pure_G.T) \wedge nc_G.T \leq pc_G + 1 \\
\qquad : \quad \{\;\; \text{See text above for the definition of } S' \;\;\} \\
\qquad\quad S,G \;:=\; S +\!\!+ S' \,, G \ominus S' \\
\quad \rangle \\
\mathsf{od} \\
\{\;\; |G| = 1 \;\;\} \;; \\
\langle [\!] T : G = \{T\} : S \;:=\; S +\!\!+ [T] \rangle \\
\{\;\; F = ForwardBag.S \;\;\}
\end{array}
$$

The key to the correctness of this algorithm is the claim that the assertion "$|G| = 1$" is implied by the condition for terminating the loop:

$$\langle \forall T \;:\; T \in G \wedge \neg(pure_G.T) \;:\; \neg(nc_G.T \leq pc_G + 1) \rangle \quad .$$

The contrapositive of this claim is that, when $|G| \neq 1$, there is a non-pure trip available to extend the sequence S. We prove this as follows. Assume that $|G| \neq 1$. Then, by (19), $\langle \exists T : T \in G : \neg(pure_G.T) \rangle$. So,

$$
\begin{array}{ll}
 & \langle \exists T \;:\; T \in G \wedge \neg(pure_G.T) \;:\; nc_G.T \leq pc_G + 1 \rangle \\
= & \qquad \{ \qquad \text{property of minimum} \quad \}
\end{array}
$$

[5] Choice quantifiers are used frequently in this paper. Formally, $\langle [\!] k : R : S \rangle$ introduces a local variable k with scope delimited by the angle brackets; k is non-deterministically initialised to a value satisfying R, following which statement S is executed. The type of k is implicit. Here, T is a trip.

$$
\begin{array}{ll}
 & \langle \Downarrow T : T{\in}G \wedge \neg(pure_G.T) : nc_G.T \rangle \ \leq pc_G + 1 \\
\Leftarrow & \quad \{ \quad \text{pigeon-hole principle (the minimum of a non-empty} \\
 & \qquad \text{bag of integers is at most the average),} \\
 & \qquad \text{(13) and integer inequalities} \quad \} \\
 & \langle \Sigma T : \neg(pure_G.T) : nc_G.T \times \#_G T \rangle \ < \ np_G \times (pc_G + 2) \\
= & \quad \{ \quad (17) \quad \} \\
 & rc_G + n_G \ < \ np_G \times (pc_G + 2) \\
\Leftarrow & \quad \{ \quad (16) \quad \} \\
 & 2 \times rc_G \ < \ np_G \times (pc_G + 2) \\
= & \quad \{ \quad \text{by range splitting, } |G| = pc_G + np_G \ ; \\
 & \qquad \text{also, by invariant, } |G| = rc_G + 1 \quad \} \\
 & 2 \times (pc_G + np_G - 1) \ < \ np_G \times (pc_G + 2) \\
\Leftarrow & \quad \{ \quad \text{arithmetic} \quad \} \\
 & 2 \ \leq \ np_G \\
= & \quad \{ \quad (18) \quad \} \\
 & 0 \neq np_G \\
= & \quad \{ \quad \text{(19) and assumption: } |G| \neq 1 \quad \} \\
 & \mathsf{true} \ .
\end{array}
$$

□

4 The Optimisation Problem

The optimisation problem we now focus on is to determine a bag of sets F such that properties (6), (7) and (8) hold of F which minimises the total travel time as given by (5). A bag, F, with the properties (6), (7) and (8) will be called a *regular* bag.

In our analysis, we refer to the subterm $\langle \Sigma T : T{\in}F : \langle \Uparrow i : i{\in}T : t.i \rangle \rangle$ in (5) as F's *forward time*, and $\langle \Sigma i :: t.i \times r_F.i \rangle$ as F's *return time*. We also refer to the subterm $\langle \Uparrow i : i{\in}T : t.i \rangle$ as T's *trip time* and $t.i \times r_F.i$ as person i's *return time*. It is important to note that we also use this terminology for bags of trips, F, that are not necessarily regular.

We continue to use the notion of "subsumption" but now applied to (regular) bags rather than sequences. So regular bag F subsumes regular bag G if F's total travel time is at most that of G. A bag is optimal if it is regular and subsumes all other bags, and is suboptimal if it is regular but not optimal.

Our solution is based on the following theorem.

Theorem 2. A bag of trips, F, that does *not* satisfy the following properties is *suboptimal*.

(a) For each T in F, the nomads in T are persons 1 thru $nc_F.T$. That is,

$$\langle \forall i,T \,:\, T{\in}F \wedge i{\in}T \,:\, nomad_F.i \equiv 1 \leq i \leq nc_F.T \rangle \quad .$$

(b) The function $boss_F$ is monotonically increasing. That is, for all settlers i and j,

$$boss_F.i \leq boss_F.j \Leftarrow i \leq j \quad .$$

(c) All pure trips in F are full. There is at most one non-full mixed trip in F and, if there is one, it is the fastest mixed trip and it has n_F nomads.

(d) For all non-nomadic trips, the function nc is a decreasing function of the leader of the trip. That is, for all non-nomadic trips T and U in F,

$$nc_F.T \geq nc_F.U \Leftarrow lead.T \leq lead.U \quad .$$

□

In words, 2(a) expresses the property that, in an optimal bag, the nomads are the fastest, and always make forward trips in a contiguous group which includes person 1. 2(b) expresses the property that the trips divide the settlers into contiguous groups. 2(d) has the corollary that the pure settlers are the slowest. So, in summary, theorem 2 establishes the "intuitively obvious" property that the search for an optimal solution can be restricted to bags of trips in which, in order of increasing travel times, the groups of people are: the nomads, the settlers in a non-full mixed trip, the mixed settlers in full trips and the pure settlers.

To prove theorem 2 we use *proof-by-contradiction*. We prove a property P by contradiction by showing that every regular bag, F, that does not satisfy P can be transformed to a regular bag, F', that does satisfy P and has a strictly smaller total travel time. To establish a succession of properties, P and Q say, we first prove P and then assume P when proving Q.

Note that non-nomadic trips have multiplicity 1 in F. Thus, for non-nomadic trips T, there is no confusion between the trip T and the individual occurrences of T in F. On the other hand, nomadic trips may have multiplicity greater than 1 in F. For such trips, we are careful to make clear whether the transformation is applied to all occurrences of the trip or just one.

4.1 Choosing Nomads

We begin by proving part (a) of theorem 2. We first establish that the nomads are persons 1 thru n, for some n.

Lemma 3. Every regular bag of forward trips is subsumed by a regular bag in which all settlers are slower than all nomads.

Proof Suppose that, within regular bag F, p is the fastest settler and q is the slowest nomad. Suppose p is faster than q.

Interchange p and q everywhere in F. We get a regular bag, F'. The return time is clearly reduced by at least $t.q - t.p$.

The times for the forward trips in F involving q are not increased in F' (because $t.p < t.q$). The time for the *one* forward trip in F involving p is increased in F' by an amount that is at most $t.q - t.p$. This is verified by considering two cases. The first case is when q is an element of p's forward trip. In this case, swapping p and q has no effect on the trip, and the increase in time taken is 0. In the second case, q is not an element of p's forward trip. In this case, it suffices to observe that, for any x (representing the maximum time taken by the other participants in p's forward trip),

$$
\begin{array}{ll}
 & t.p \uparrow x \ + \ (t.q - t.p) \\
= & \qquad \{ \quad \text{distributivity of sum over max, arithmetic} \quad \} \\
 & t.q \uparrow (x + (t.q - t.p)) \\
\geq & \qquad \{ \quad t.p \leq t.q, \text{ monotonicity of max} \quad \} \\
 & t.q \uparrow x \quad .
\end{array}
$$

Finally, the times for all other forward trips are unchanged.

The net effect is that the total time taken does not increase. That is, F' subsumes F. Also, the total forward-trip time of the settlers is strictly increased. Thus, repeating the process of swapping the fastest settler with the slowest nomad whilst the former is faster than the latter is guaranteed to terminate with a bag that subsumes the given bag and in which all settlers are slower than all nomads.

□

Lemma 4. Every regular bag of forward trips is subsumed by a bag, F, that satisfies 2(a).

Proof Suppose a regular bag F of forward trips is given. By lemma 3, F is subsumed by a bag G in which the nomads are persons 1 thru n_G. (Bags F and G may be the same, but that is not significant.)

For each trip T in G, consider the set of nomads in T. Specifically, define $nom.T$ to be

$$T \cap \{i \,|\, nomad_G.i\} \quad .$$

Recall that $nc_G.T$ is the number of nomads in set T. That is, $nc_G.T = |nom.T|$.

Replace T in the bag G by

$$(T \cap \{i \,|\, settler_G.i\}) \ \cup \ \{i \mid 1 \leq i \leq nc_G.T\} \quad .$$

This replaces G by a bag F'. To see that F' is regular, we observe that the replacement of T increases the number of forward trips by person i only when $i \leq nc_G.T$. But $nc_G.T \leq n_G$; so, settlers in G are also settlers in F'. Hence, the

size of the bag T is unchanged by the replacement. That is, property (7) is an invariant of the replacement. The number of forward trips made by nomads i in G such that $nc_G.T < i \leq n_G$ decreases by at most 1. So, such nomads may not be nomads in F'. However, the number of forward trips each makes remains strictly positive (since a nomad makes at least 2 forward trips, by definition), and each decrease in the number of forward trips made by such a nomad is compensated by an increase in the number of forward trips made by some nomad i, where $1 \leq i \leq nc_G.T$. That is, properties (8) and (6) are invariant under the replacement. Finally, the replacement decreases the total trip time because the times of the return trips are decreased, and the times of the forward trips are not increased. That is, F' subsumes G; by the transitivity of the subsumes relation, F' also subsumes F.
□

Corollary 2. Every regular bag is subsumed by a bag, F, in which the number of nomads, n_F, is at most C.

Proof Every regular bag is subsumed by a bag, F, satisfying 2(a), and

$$
\begin{array}{ll}
 & n_F \leq C \\
= & \quad \{ \quad 2(a) \quad \} \\
 & \langle \Uparrow T : T \in F : nc_F.T \rangle \leq C \\
= & \quad \{ \quad \text{definition of maximum } (\Uparrow) \quad \} \\
 & \langle \forall T : T \in F : nc_F.T \leq C \rangle \\
= & \quad \{ \quad \text{definition of } nc_F, \ (7) \quad \} \\
 & \mathsf{true} \ .
\end{array}
$$

□

4.2 Permuting Settlers

In this section, we prove part (b) of theorem 2. We begin, however, with a similar lemma which is used later in the proof of part (d).

Lemma 5. Suppose bag F satisfies 2(a). Then either the settler count in each trip is a monotonic function of the leader of the trip (i.e.

$$sc_F.T \leq sc_F.U \Leftarrow lead.T \leq lead.U \quad)$$

or F is suboptimal.

Proof Take trips T and U in F such that $sc_F.T > sc_F.U$ and $lead.T \leq lead.U$. It follows that $T \neq U$. Because $sc_F.T > 0$ and F satisfies 2(a), $lead.T$ is a settler. Now, because $lead.T \leq lead.U$ and F satisfies 2(a), it follows that $lead.U$ is also a settler. But settlers are elements of exactly one trip. We conclude that $sc_F.T > sc_F.U > 0$, both T and U have multiplicity 1 in F, and $lead.T < lead.U$.

Rearrange the settlers in T and U so that $sc_F.T$ and $sc_F.U$ are unchanged (thus guaranteeing a regular bag) and the slowest settlers are in T and the fastest settlers are in U. Using primes to denote the new values of T and U, we have

$$
\begin{array}{ll}
 & t.(lead.T') + t.(lead.U') \\
= & \quad \{ \quad lead.U \text{ is the slowest settler, so } lead.T' = lead.U \quad \} \\
 & t.(lead.U) + t.(lead.U') \\
< & \quad \{ \quad sc_F.U' = sc_F.U < sc_F.T \text{ and } U' \text{ contains the fastest settlers;} \\
 & \qquad \text{so } lead.U' < lead.T \quad \} \\
 & t.(lead.U) + t.(lead.T) \quad .
\end{array}
$$

Other trips are unchanged, so the effect is to strictly decrease the total travel time.
□

Lemma 6. A bag F that does not satisfy 2(b) is suboptimal.

Proof Take any two settlers i and j such that $boss_F.i > boss_F.j$ and $i \leq j$. It follows that $i \neq j$ and they must be in different trips, T and U say. Swap $boss_F.j$ (the slowest person in trip U) with the fastest settler in trip T. Then, using primes to denote the new trips,

$$
\begin{array}{ll}
 & t.(lead.T') + t.(lead.U') \\
= & \quad \{ \quad i < j \leq boss_F.j < boss_F.i \quad ; \quad \text{so } i \neq boss_F.i \\
 & \qquad \text{hence } lead.T' = lead.T = boss_F.i \quad \} \\
 & t.(lead.T) + t.(lead.U') \\
< & \quad \{ \quad lead.U = boss_F.j, \\
 & \qquad boss_F.j \text{ has been replaced by } k \text{ where } k \leq i < j \leq boss_F.j \quad \} \\
 & t.(lead.T) + t.(lead.U) \quad .
\end{array}
$$

Other trips are unchanged, so the effect is to strictly decrease the total travel time.
□

4.3 Filling Non-Nomadic Trips

Part (c) of theorem 2 is about filling non-nomadic trips as far as possible with nomads. The proof is split into several lemmas. The proofs themselves are omitted because they add no new techniques.

Lemma 7. A bag F that satisfies 2(a) but has a non-full pure trip is suboptimal.
□

Lemma 8. A bag F that satisfies 2(a) but has a non-full mixed trip that is not the fastest mixed trip is suboptimal.
□

Lemma 9. Suppose bag F satisfies 2(a). Suppose F has a non-full mixed trip that is the fastest mixed trip but all mixed trips in F have fewer than n_F nomads. Then F is suboptimal.
□

Lemma 10. Suppose bag F satisfies 2(a). Suppose there is a mixed trip, T say, with n_F nomads and a non-full mixed trip, U say, with n nomads where $n < n_F$. Suppose U is the fastest mixed trip. Then F is suboptimal.
□

Corollary 3. A bag F that satisfies 2(a) but does not satisfy theorem 2(c) is suboptimal. A bag F that satisfies 2(a) but does not satisfy theorem 2(d) is suboptimal.
□

5 Constructing an Optimal Bag of Forward Trips

In this section, we use theorem 2 to give a lower bound on the time taken to cross. In the process of calculating the lower bound, an optimal bag of forward trips can be constructed. Then, using the construction given in section 4, an optimal putative sequence can be constructed from the bag.

Our algorithm for constructing an optimal bag constructs in stages an "ordered" bag of sets where "ordered" is defined below.

Definition 1 (Ordered). We say that a bag of sets, F, is *ordered* if

(21) $\langle \forall T : T \in F : 2 \leq |T| \leq C \rangle$

and it satisfies the four properties stated in theorem 2.
□

The algorithm constructs a bag of trips, F, starting with the slowest trips. The measure of progress is a pair (m, p) ordered lexicographically, where m is a measure of the number of people not yet included in a trip and p measures the "excess" of pure trips over return trips. Formally, we exploit the following theorem.

Theorem 3. If a bag of sets, F, is optimal, it is ordered and

(22) $\langle \forall i \ : \ 1 \leq i \leq N \ : \ 1 \leq f_F.i \rangle$,

and

(23) $pc_F \ = \ \langle \Sigma i : 2 \leq i \leq n_F : r_F.i \rangle$.

Proof Comparing the definition of regular bags (properties (6), (7) and (8)) with the definition of ordered bags, we see that (21) and (7) are identical, as are (22) and (8). Thus, any ordered bag is a solution if it also satisfies (6). We now show that (6) and (23) are equivalent when the bag F satisfies the properties stated in theorem 2. That is, we prove that

$$|F| \ = \ \langle \Sigma i :: r_F.i \rangle + 1 \quad \equiv \quad pc_F \ = \ \langle \Sigma i : 2 \leq i \leq n_F : r_F.i \rangle \ .$$

We have:

$$
\begin{array}{ll}
 & |F| \\
= & \quad \{ \quad \text{definition of } |F|, \text{ range splitting} \quad \} \\
 & pc_F \ + \ np_F \\
= & \quad \{ \quad \text{by 2(a), } 1 \in T \equiv \neg(pure_F.T), \text{ definition of } f_F.1 \quad \} \\
 & pc_F \ + \ f_F.1 \ \ .
\end{array}
$$

Hence,

$$
\begin{array}{ll}
 & pc_F \ = \ \langle \Sigma i : 2 \leq i \leq n_F : r_F.i \rangle \\
= & \quad \{ \quad \text{above, arithmetic} \quad \} \\
 & |F| - f_F.1 \ = \ \langle \Sigma i : 2 \leq i \leq n_F : r_F.i \rangle \\
= & \quad \{ \quad f_F.1 - 1 = r_F.1, \text{ arithmetic} \quad \} \\
 & |F| \ = \ \langle \Sigma i : 1 \leq i \leq n_F : r_F.i \rangle + 1 \\
= & \quad \{ \quad \text{by 2(a)}, \quad r_F.i \neq 0 \ \equiv \ 1 \leq i \leq n_F \quad \} \\
 & |F| \ = \ \langle \Sigma i :: r_F.i \rangle + 1 \ \ .
\end{array}
$$

□

Henceforth, we call $pc_F - \langle \Sigma i : 2 \leq i \leq n_F : r_F.i \rangle$ the *excess* of the bag F. Theorem 3 states that an optimal bag is obtained by constructing an ordered bag in which everyone makes a trip (property (22)) and the number of pure trips equals the number of return trips made by nomads other than person 1 (property (23)).

Theorem 3 offers no way of determining the number of pure trips in an optimal bag except by considering all the different possibilities. The basic structure of our solution is thus to evaluate the minimum over all p of the total travel time of a regular, ordered bag of forward trips in which the number of pure trips is p. So, the problem becomes one of determining a lower bound on the travel time incurred by a bag of mixed and nomadic forward trips with an "excess" p. This problem is solved by identifying a collection of (acyclic) equations on the travel times; by determining the (unique) solution of the equations we obtain the desired lower bound on the total travel time; simultaneously, the bag of forward trips can be constructed in the standard way.

For convenience, we define rt by

$$
(24) \quad rt.n \ = \ \langle \Sigma i : 1 \leq i \leq n : t.i \rangle \ \ .
$$

In the equations below, occurrences of the function rt record the return time for a given number of nomads; occurrences of the function t record the forward time of some trip.

5.1 Outline Algorithm

The basic structure of our solution is to design a non-deterministic algorithm that constructs regular, ordered bags. The algorithm is designed so that every such bag is constructed by some resolution of the non-determinism.

An outline of the algorithm is shown below.

$$
\begin{array}{l}
\{\ \ 2 \le C < N\ \ \} \\
\langle [] p \\
\quad : \quad 0 \le p \le \left\lfloor \dfrac{N-2}{C} \right\rfloor \\
\quad : \quad AddPureTrips \\
\qquad \{\ \ |F| = p = pc_F\ \ \}\ ; \\
\qquad \langle [] \, n, n' \\
\qquad\qquad : \quad n = n' = 0 \\
\qquad\qquad : \quad \langle [] m \\
\qquad\qquad\qquad : \quad m = N - p \times C \\
\qquad\qquad\qquad : \quad AddFullMixedTrips \\
\qquad\qquad\qquad\qquad \{\ \ 2 \le m \le C\ \ \}\ ; \\
\qquad\qquad\qquad\qquad IncludeRest \\
\qquad\qquad\qquad\qquad \{\ \ \langle \forall i\ :\ 1 \le i \le N\ :\ 1 \le f_F.i \rangle\ \ \}\ ; \\
\qquad\qquad\qquad\qquad \mathsf{if}\ \ 0 = p\ \ \rightarrow\ \ \mathsf{skip} \\
\qquad\qquad\qquad\qquad \Box\ \ 0 < p\ \ \rightarrow\ \ FinaliseNumberOfNomads \\
\qquad\qquad\qquad\qquad \mathsf{fi} \\
\qquad\qquad \rangle\ ; \\
\qquad\qquad AddNomadicTrips \\
\qquad \rangle \\
\rangle \\
\{ \qquad (ordered.F \wedge \langle \forall i : 1 \le i \le N : 1 \le f_F.i \rangle) \\
\quad \wedge\ \ pc_F = \langle \Sigma i : 2 \le i \le n_F : r_F.i \rangle\ \ \}
\end{array}
$$

The algorithm constructs a bag, F, of trips. At all stages, F is ordered. The algorithm begins by introducing a variable p which is non-deterministically initialised to a natural number at most $\left\lfloor \dfrac{N-2}{C} \right\rfloor$. The step $AddPureTrips$ initialises the variable F to a bag of p pure trips. Informally, p is the excess of the bag F. (This isn't quite true as explained below.) Subsequent stages reduce p to zero.

Next, variables n and n' are introduced, both with initial value zero; an invariant of n is that persons 1 thru n are nomads in F. The value of n' is always at least n. Persons $n{+}1$ thru n' have a forward count of one (and so are settlers); these persons will, however, eventually become nomads in F.

The variable m is introduced next. An invariant of m is that persons $n'{+}1$ thru m are the ones with a forward count of zero in F; since all pure trips are full, the initial value of m is thus $N - p \times C$. The step $AddFullMixedTrips$ adds full

mixed trips to F while C is less than m. Then *IncludeRest* adds one additional trip to F in order to guarantee that every person is included in at least one trip. This additional trip may be full or non-full.

The final step is to add nomadic trips to F in order to reduce the excess p to 0, if this is not already the case. The number of nomads in these trips is at least $n' \uparrow 2$ and at most $m \downarrow (p+1)$.

The total travel time is simply the minimum over all possible choices of the travel times of the constructed bags. The calculation of the optimal travel time is equivalent to a shortest-path problem. Formally, each non-deterministic choice is interpreted as a minimum and the addition of a set to F adds the return-trip time of the nomads in the added trip (where the predicate *nomad* is evaluated once the trip is added) and the forward-trip time to the total trip time. We exploit the fact that addition distributes over minimum —the formal equivalent of the "principle of optimality" of dynamic programming— in order to obtain a polynomial-time algorithm.

Let us now give the details of the individual steps.

5.2 Adding Pure Trips

The first step (after the non-deterministic initialisation of p) is *AddPureTrips* which initialises F to a set of p pure trips. In order to guarantee that the nomad count nc_F is a decreasing function of the leader, $boss_F$ is increasing, and all pure trips are full, the assignment to F is simply

$$F \quad := \quad \langle \cup k \;:\; 1 \leq k \leq p \;:\; \{\{i \mid N - k \times C < i \leq N - (k-1) \times C\}\} \rangle \quad .$$

All trips added to F are full, and the leader of the kth trip is person $N-(k-1)\times C$.

On completion of this assignment, F is ordered and $|F| = p = pc_F$.

The forward-trip time for the pure trips is constant. It is given by the function PT:

$$(25) \quad PT.p \quad = \quad \langle \Sigma i : 0 \leq i < p : t.(N - C \times i) \rangle \quad .$$

Thus

$$(26) \quad TOT \quad = \quad \left\langle \Downarrow p : 0 \leq p \leq \left\lfloor \frac{N-2}{C} \right\rfloor \;:\; PT.p \;+\; NP.p \right\rangle \quad .$$

The value of the function NP is determined by the mixed and nomadic trips added to F in the later stages.

(Formally, (26) is a consequence of the fact that addition distributes over minimum. In other words, the time for the pure trips can be "factored out" of the calculation of the total travel time.)

5.3 Adding Full Mixed Trips

In the second stage, variables n, n' and m are initialised to 0, 0 and $N - p \times C$, respectively, and a set of full mixed trips is added to F as follows.

$$
\begin{array}{lll}
\{\ \textbf{Invariant}: & & ordered.F \\
& \wedge & 0 \leq n \leq n' \leq C{-}1\ \wedge\ n'{\uparrow}1 < m \\
& \wedge & \langle \forall i : 1 \leq i \leq n : 1 \leq r_F.i \rangle \\
& \wedge & \langle \forall i : n < i \leq n' \vee m < i \leq N : 1 = f_F.i \rangle \\
& \wedge & pc_F \ =\ \langle \Sigma i : 2 \leq i \leq n : r_F.i \rangle + p + [0 = n < n'] \\
& \wedge & d.m.n.n' \leq p\ \ \}
\end{array}
$$

$$
\begin{array}{l}
\textsf{do}\ C < m\ \rightarrow \\
\qquad \langle [] i \\
\qquad\quad : \quad n'{\uparrow}1 \leq i \leq C{-}1\ \wedge\ d.(m{-}C{+}i).n'.i \leq p - n'{\uparrow}1 + 1 \\
\qquad\quad : \quad F \ :=\ F \cup \{\{j \mid 1 \leq j \leq i\} \cup \{j \mid m{-}C{+}i < j \leq m\}\}\ ; \\
\qquad\qquad\quad m, n, n', p \ :=\ m{-}C{+}i\ ,\ n'\ ,\ i\ ,\ p - n'{\uparrow}1 + 1 \\
\qquad \rangle \\
\textsf{od}
\end{array}
$$

The second, third and fourth clauses of the invariant of this loop express precisely the functions of m, n and n'. Refer back to section 5.1 for an informal account of their function.

The clause

$$pc_F \ =\ \langle \Sigma i : 2 \leq i \leq n : r_F.i \rangle + p + [0 = n < n']$$

in the invariant expresses precisely the relation between p and the pure count of F. Note that when $0 = n$ all trips in F are pure; when, in addition, $n < n'$ there is one trip in F which includes persons 1 thru n'. At a later stage, this trip will become a mixed trip and, so, is not counted in p. The term $[0 = n < n']$ evaluates to 1 if $0 = n < n'$ and to 0 otherwise. The inclusion of this term compensates for not counting the trip in the excess p.

The clause

$$d.m.n.n' \leq p$$

requires some explanation. Its function is to guarantee that the non-deterministic choices do not abort; equivalently, the value of p is always at least zero. Specifically,

$$(27)\quad d.m.n.n' \ =\ \left(\left\lceil \frac{m{-}C}{C - n'{\uparrow}1} \right\rceil + 1\right) \times (n'{\uparrow}1 - 1)\ \ .$$

In order to guarantee 2(d) —the number of nomads in non-nomadic trips is a decreasing function of the leader of the trip— , the value of n' is increasing; each mixed trip that is added to F has at least $n'{\uparrow}1$ nomads and at most $C - n'{\uparrow}1$ settlers. Thus, in this stage, at least $\left\lceil \frac{m{-}C}{C - n'{\uparrow}1} \right\rceil$ additional trips are added to

F, each of which causes p to be reduced by at least $n'\uparrow 1-1$. The third stage is executed when $m \leq C$; this stage guarantees that all people make at least one trip by adding to F a single trip consisting of persons 1 thru m. This also causes p to be reduced by at least $n'\uparrow 1-1$. Thus $d.m.n.n'$ is a lower bound on the amount that p will be reduced by the later addition of trips to F.

We leave the verification of the invariant to the reader.

From the algorithm, we can determine the minimum total travel time for the mixed trips. We have, for all p such that $0 \leq p \leq \left\lfloor \frac{N-2}{C} \right\rfloor$,

$$NP.p \;=\; MX.(N - p{\times}C).0.0.p \quad .$$

For the moment, we define $MX.m.n.n'.p$ only for the case that $C < m$. Specifically, for all n and n' such that $0 \leq n \leq n' \leq C-1$, all m such that $C < m$, and all p such that $d.m.n.n' \leq p$, we have

$$\begin{aligned} MX.m.n.n'.p = \; & \langle \Downarrow i \\ & \quad : \; n'\uparrow 1 \leq i \leq C-1 \;\wedge\; d.(m-C+i).n'.i \leq p - n'\uparrow 1 + 1 \\ & \quad : \; MX.(m-C+i).n'.i.(p - n'\uparrow 1 + 1) \\ & \rangle \;+\; t.m \;+\; rt.n' \quad . \end{aligned}$$

The justification of this equation is that, for a given choice of i, a trip is added to F with leader m. The forward time for the trip is thus $t.m$. The trip adds 1 to the forward count of each person from 1 thru i; after the addition of the trip, persons 1 thru n' are the nomads in F and the addition of the trip to F adds $rt.n'$ to F's return-trip time. As before, the distributivity of addition over minimum is used to "factor out" the contribution of each trip to the total travel time.

5.4 Completing the Mixed Trips

The third stage, *IncludeRest*, ensures that everyone makes at least one forward trip. The assignment is simply

$$F, n, p \quad := \quad F \cup \{\{j \mid 1 \leq j \leq m\}\}, n', p - n'\uparrow 1 + 1$$

From the invariant of the second stage, we determine that the assignment is executed when $n'\uparrow 1 < m \leq C$; this guarantees that the added trip is regular. Also, from the invariant, $d.m.n.n' \leq p$. That is, $n'\uparrow 1 - 1 \leq p$. After the assignment, it is thus that case that $0 \leq p$. In more detail, the postcondition established by the assignment is

$$\begin{aligned} & ordered.F \\ \wedge \quad & \langle \forall i : 1 \leq i \leq N : 1 \leq f_F.i \rangle \\ \wedge \quad & pc_F \;=\; \langle \Sigma i : 2 \leq i \leq n : r_F.i \rangle + p + [0 = n < n'] \\ \wedge \quad & (0 < p \;\vee\; 0 < n) \quad . \end{aligned}$$

The final conjunct is a consequence of the assumption $C < N$. At the conclusion of the second stage, F is non-empty; so, $|F|$ is at least 2 after the above assignment. By inspection of the assignments in the two stages, we conclude that either $0 < p$ or $0 < n'$. The conjunct follows because n is assigned the value of n'.

The trip that is added to F may be full (when $m = C$) or non-full (when $m < C$); it may also be (or become) a mixed trip or it may be a nomadic trip. Immediately following the assignment, the statement

$$
\begin{array}{lll}
\textsf{if} & 0 = p \;\rightarrow\; \textsf{skip} & \\
\Box & 0 < p \;\rightarrow\; \langle [] i & \\
 & \qquad : \quad n'{\uparrow}2 \leq i \leq m{\downarrow}(p{+}1) & \\
 & \qquad : \quad n' \;:=\; i & \\
 & \rangle & \\
\textsf{fi} &&
\end{array}
$$

is executed. This chooses the final value of the number of nomads. (The choice of i cannot abort because, as remarked above, $n'{\uparrow}1 < m \leq C$; as a consequence, if $0 < p$ then $n'{\uparrow}2 \leq m{\downarrow}(p{+}1)$.)

The addition of this trip extends our definition of the function MX. Anticipating the fact that when $0 = p$ no further trip is added to F, we have: for all m, n and n' such that $m \leq C$,

$$MX.m.n.n'.(n'{\uparrow}1 - 1) \;=\; t.m + rt.n' \quad .$$

Also, for all m, n and n' such that $2 \leq m \leq C$, and all p such that $0 < p$,

$$
\begin{array}{l}
MX.m.n.n'.(p + n'{\uparrow}1 - 1) \\
= \langle \Downarrow i : n'{\uparrow}2 \leq i \leq m{\downarrow}(p{+}1) : NT.n'.i.p \rangle + t.m + rt.n' \quad .
\end{array}
$$

The function NT gives the time taken by the nomadic trips. See below.

5.5 Adding Nomadic Trips

The final stage in the construction of F is the possible addition of a number of nomadic trips. When this stage is executed, we have the following properties of F, n, n' and p:

$$
\begin{array}{ll}
 & 0 \leq n \leq n' \leq C \;\wedge\; (0 = p \;\vee\; 2 \leq n' \leq p{+}1) \\
\wedge & \mathit{ordered.F} \\
\wedge & pc_F \;=\; \langle \Sigma i : 2 \leq i \leq n : r_F.i \rangle + p + [0 = n < n'] \\
\wedge & \langle \forall i : 1 \leq i \leq n : 1 \leq r_F.i \rangle \;\wedge\; \langle \forall i : n < i \leq N : 1 = f_F.i \rangle
\end{array}
$$

The goal is to add nomadic trips to F in such a way that F remains ordered, the number of nomads in F becomes n' (if it is not n' already) and p is reduced to zero.

We present a non-deterministic algorithm to achieve this task. The algorithm is designed so that every bag that satisfies the specification corresponds to one way of resolving the non-determinism.

The algorithm begins by ensuring that F has n' nomads, and then repeatedly adds nomadic trips to F whilst $0<p$. By choosing the trips in order of decreasing size, 2(a) is guaranteed without loss of choice. The symbl "$\dot\cup$" denotes bag union. (It is only at this point in the construction that trips in F may have multiplicity greater than 1.)

$\{$ Precondition given above $\}$

if $0=p \rightarrow$ **skip**

$\Box$ $0<p \wedge n<n' \;\rightarrow\; F,p,n \;:=\; F \cup \{\{j \mid 1\le j\le n'\}\},p-n'+1,n'$

fi

$\{\; \langle \forall i : 1\le i\le n' : 1\le r_F.i\rangle \;\wedge\; \langle \forall i : n'<i\le N : 1=f_F.i\rangle \;\}$;

do $0<p\rightarrow$ $\langle [\!] i$

$\quad : \quad 2\le i\le (p+1)\downarrow n$

$\quad : \quad F,p,n \;:=\; F \mathbin{\dot\cup} \{\{j \mid 1\le j\le i\}\},p-i+1,i$

$\rangle$

od

$\{\; ordered.F \;\wedge\; n_F=n' \;\wedge\; pc_F \;=\; \langle \Sigma i : 2\le i\le n_F : r_F.i\rangle \;\}$

Now we determine the additional travel time incurred by adding these trips to F. Letting $NT.n.n'.p$ denote the additional time, we have:

$$NT.n.n'.0 \;=\; 0 \;\;.$$

Also, for all p, n and n' such that $0<p$, $0\le n<n'$ and $2\le n'\le (p+1)\downarrow C$,

$$NT.n.n'.p \;=\; t.n' + rt.n \;+\; TNT.n'.(p-n\uparrow 1+1) \;\;.$$

Finally, for all p and n such that $0<p$, and $0\le n$,

$$NT.n.n.p \;=\; TNT.n.p \;\;.$$

The function TNT is given by:

$$TNT.n.0 \;=\; 0 \;\;,$$

and, for all p such that $0<p$ and all n such that $2\le n\le (p+2)\downarrow C$,

$$TNT.n.p \;=\; \langle \Downarrow i : 2\le i\le (p+1)\downarrow n : t.i \;+\; rt.i \;+\; TNT.i.(p-i+1)\rangle \;\;.$$

5.6 Solving the Equations

Finding the total travel time incurred by an optimal bag of forward trips is achieved by determining the greatest solution to the above equations. This can be done in worst-case time $O(N^2 \times C^3)$ by first tabulating PT and TNT, then NT followed by MX, and finally TOT. Each set of equations is evaluated by imposing an appropriate lexicographic ordering on the parameters; for example, $MX.m.n.n'.p$ is evaluated in lexicographic order on p, m, $C-n'$ and $C-n$. (As remarked earlier, the process is equivalent to solving a shortest-path problem, so other shortest-path algorithms could be used. These are likely to be more effective in practice; but they may have poorer worst-case complexities.) An optimal bag (rather than just the travel time) can be determined in the standard way by recording the terms that realise the minimum when evaluating any such quantification.

6 Conclusion

That the torch problem can be solved at the level of generality in this paper in time quadratic in the number of people appears to be new. The case of 4 people and a bridge capacity of 2 is very widely discussed on the internet (although its origin appears to be unknown). The case of N people and a bridge capacity of 2 has been solved by Rote [3]. (The reader is referred to Rote's paper for other publications and web links.) I believe that the capacity-C problem has been attempted, but presumably never solved (in polynomial time), because I was told before embarking on it that it was believed to be "hard" [Tom Verhoeff, private communication]. The problem is indeed difficult, but it is not "hard" in any technical sense of the word (as, for example, in "NP-hard"). As we have shown, the problem can be solved in time quadratic in the number of people.

I was motivated to tackle the problem because I use the capacity-2 problem in a course on algorithmic problem solving [1] to entry-level Computer Science students at the University of Nottingham. The capacity-2 problem is interesting because it demonstrates that "obvious" solutions may be incorrect; also, obtaining a correct solution demands particular attention to the avoidance of unnecessary detail. (Like the capacity-C problem, the solution is obtained by focusing on just the forward trips.) Initially, I thought that the capacity-C problem would not be much more difficult than the capacity-2 problem. It turned out to be far harder to solve than I anticipated.

Of course, the solution presented here specialises in the case that the capacity is 2. In this case, the solution is essentially the same as that presented by Rote. (In the case that the capacity is 2, all the complications concerning the possibility of non-full bags disappear; for this reason, the construction simplifies considerably.) Rote describes the solution in terms of "multigraphs" rather than bags. For capacity 2, the difference is superficial. Each edge of a "multigraph" connects two people and, hence, is just a set of two people. However, Rote's "multigraphs" do not appear to generalise to capacity N, whereas the use of bags does.

For capacity 2, the solution can be determined in logarithmic time. For full details see [1, chapter 8]. Rote describes the more obvious "greedy", linear-time algorithm.

The current solution leaves much to be desired. The underlying calculations have not been done to the level of rigour and detail that I would nowadays demand. The fact that the capacity-2 problem can be solved by a greedy algorithm suggests that there is much scope for improving the worst-case complexity of the solution presented here. The reason I suspect that improvements can be made is that the "edges" in the underlying path problem connect vertices labelled with a four-tuple one of whose components is p but the "length" of an edge does not depend on p. This suggests that, at the very least, a linear-time algorithm should be possible.

Acknowledgements Thanks go to a number of people who have helped me at various stages.

I thank Diethard Michaelis for stressing the importance of regularity and for his suggestions on naming (which I have adopted). Thanks also for his reading and commenting on drafts and for his efforts to improve on notation (which I have yet to do anything about!).

Thanks to Arjan Mooij for helping me prove that any regular bag of forward trips can be transformed to a putative sequence. Section 3.1 is essentially due to him — but the responsibility for errors is mine. Thanks also to João Ferreira and Arjan Mooij for suggesting improvements to some calculations.

Finally, thanks to Tom Verhoeff and Günter Rote for providing me with bibliographic information on the problem.

Note added 14/02/2013. The statement of theorem 3 in the published conference paper is incorrect. (Essentially the "only if" statement became an "if" statement by virtue of an unintended interchange of the words "optimal" and "ordered".) I am grateful to Ole Rummel for pointing out the error.

References

1. Roland Backhouse. Algorithmic problem solving. Lecture notes, School of Computer Science, University of Nottingham. Updated at least annually and widely available on the internet, but see author's website for latest version.
2. Roland Backhouse. Regular algebra applied to language problems. *Journal of Logic and Algebraic Programming*, (66):71–111, 2006.
3. Günter Rote. Crossing the bridge at night. *Bulletin of the European Association for Theoretical Computer Science*, 78:241–246, October 2002.