IMPROVED SORTING NETWORKS WITH O(LOG N) DEPTH

M.S.Paterson

Department of Computer Science
University of Warwick
Coventry, CV4 7AL, England

Abstract

The sorting network described by Ajtai, Komlós and Szemerédi was the first to achieve a depth of O(log n). The networks introduced here are simplifications and improvements based strongly on their work. While the constants obtained for the depth bound still prevent the construction being of practical value, the structure of the presentation offers a convenient basis for further development.

1. Introduction

We consider **networks** which are constructed using components of a single type, the **comparator**. A comparator has two inputs and yields as its two outputs the input elements in sorted order. The N inputs are presented on N wires and at each successive level of the network at most $N/2$ disjoint pairs of wires are put into comparators. After each level the N wires carry the original elements in some permuted order and the network is a **sorting network** if the elements are always in sorted order after the final level of comparators. The **depth** of a network is just the number of levels.

One very simple regular sorting network, "odd-even sort", has depth N for $N>2$. An elegant recursive network due to Batcher [3] requires only about $(\log N)^2/2$ depth. A useful source for background and references in this area is [6]. The familiar $\Omega(N \log N)$-comparisons lower bound for sorting immediately gives an $\Omega(\log N)$ lower bound for the depth of sorting networks. Following the appearance of [3] in 1968, a longstanding open problem has been to close this depth complexity gap. This was finally achieved in 1983 by Ajtai, Komlós and Szemerédi ([1], [2]) with a sorting network of $O(\log N)$ depth. Their construction and proof are of some intricacy and since their main concern was just to provide an existence proof for such networks the numerical constant in the depth bound is enormous. In this paper I will present a simplification of their construction which allows a more accessible proof. The constant obtained in our proof is still so large that Batcher's network has less depth for all practical sizes of networks, but we have some hope that further refinements may yield a substantially improved constant.

The sorting network described here was originally presented at the Complexity Theory meeting in Oberwolfach in October 1983 and again at Theory Day at Columbia University in March 1984. I would like to thank Pippenger, Rackoff and Skyum for valuable discussions during visits to Aarhus University and IBM San Jose in 1983, and Cole for useful comments on a preliminary draught of this paper. I also acknowledge the patient urging of many colleagues without whom this work would have remained in oral tradition, and the support of the Science and Engineering Research Council with a Senior Fellowship from 1985.

2. Overview of network

Consider a binary tree with 'bags' at each node. Initially the set of N elements to be sorted is contained in the single bag at the root. Suppose we were to partition the elements from the root bag into a left and a right half, and we transferred these to the left and right daughter bags respectively. (We shall use the terminology of left and right rather than small and large in the sorted order of elements to accord with a geometrical picture of a binary tree with the root at the top and branches going down to left and right.) If we were to continue in the same way then after $\log N$ stages the elements would have been sorted. Unfortunately the task of partitioning a set of n elements into left and right halves requires $\Omega(\log n)$ levels of comparators. The idea used by Ajtai, Komlós and Szemerédi ([1], [2]) is to take an approximate partition of elements, which can be achieved in only a constant number of comparator levels, but to introduce some error-recovery structure into the sorting scheme. In our most basic scheme this is done by partitioning the bag of elements at each node into four parts: the main left and right 'halves', which are sent down to the daughter bags, and in addition two small fragments from the extreme ends of the partitioning process, which are intended to include most of the elements that were wrongly routed from higher in the tree and which are now returned to the parent bag above. Our network operates almost uniformly in this manner from start to finish. At any time the sizes of the bags at any one level in the tree are equal and this size increases in a geometrical progression with the depth of this level below the root. The upper smaller bags of the tree are concerned with recycling that small fraction of the elements which may have been misclassified in some partitioning process. As time progresses the size of each bag is reduced, again in a geometric progression, thus 'squeezing' the elements down the tree towards their final locations at the leaves. The correctness of the network is demonstrated by proving an invariant which bounds the proportion, within each bag, of elements with any particular degree of displacement from their 'proper' positions. The notion of 'strangeness' used in this invariant is a simplification of that introduced in [1]. During the account below of the network and the correctness proof, various parameters are required. At each point we set out the inequalities which the parameters have to satisfy and produce an example of suitable parameters in order to animate the description.

We initially describe the sizes of various sets of elements as if they were real numbers. Ultimately we will show how appropriate integer values can be chosen so that the required inequalities still hold.

3. Definitions and building blocks

In [1] the notion of an ε-halver is introduced. For any $\varepsilon > 0$, an **ε-halver** for m elements is a comparator network with m inputs, and with outputs partitioned into a left and a right block each of size $m/2$. The ε-halver has the property that, for any set of inputs and any $k \le m/2$, the number of elements from the k leftmost in the ordering which are output in the right block, and from the k rightmost which are output in the left block, are each less than εk. An ε-halver can be constructed in constant depth (depending on ε), for example by using expander graphs. This is described in [1], but where those authors go on to build "ε-nearsorts", we shall use the more limited component which is described immediately. A **$(\lambda, \varepsilon, \varepsilon_0)$-separator** (on m elements) returns a partition of its m input values into four parts FL, CL, CR, FR, of sizes $\lambda m/2$, $(1-\lambda)m/2$, $(1-\lambda)m/2$ and $\lambda m/2$ respectively. The set FL (for "far-left") has the property that for any k, $k \le \lambda m/2$, the number of elements from the set of k leftmost input values which are not in FL is less than εk. The same holds for FR ("far right") with respect to the rightmost elements. Also, for any k, $k \le$

$m/2$, the number of elements from the set of k leftmost input values which are not in FL or CL ("centre-left") is less than $\varepsilon_0 k$, and similarly for elements from the right half of the ordering which end up not in FR or CR. It is easy to build some $(\lambda, \varepsilon, \varepsilon_0)$-separator from a constant number of ε_0-halvers. In particular if we use an ε_0-halver on the m inputs, then apply an ε_0-halver to each of the resulting output sets of size $m/2$, then two ε_0-halvers to each extreme set of size $m/4$ and so on through p levels, the resulting network yields a $(2^{-p+1}, p\varepsilon_0, \varepsilon_0)$-separator. The p levels of halvers produce a sequence of $2p$ blocks. The extreme blocks are taken for FL and FR, while the left and right halves of the remaining sequence are combined to form CL and CR respectively. To verify the value of ε we note that a proportion ε_0 of some set of extreme elements may escape to the 'wrong' output block at each of the p layers of ε_0-halvers.

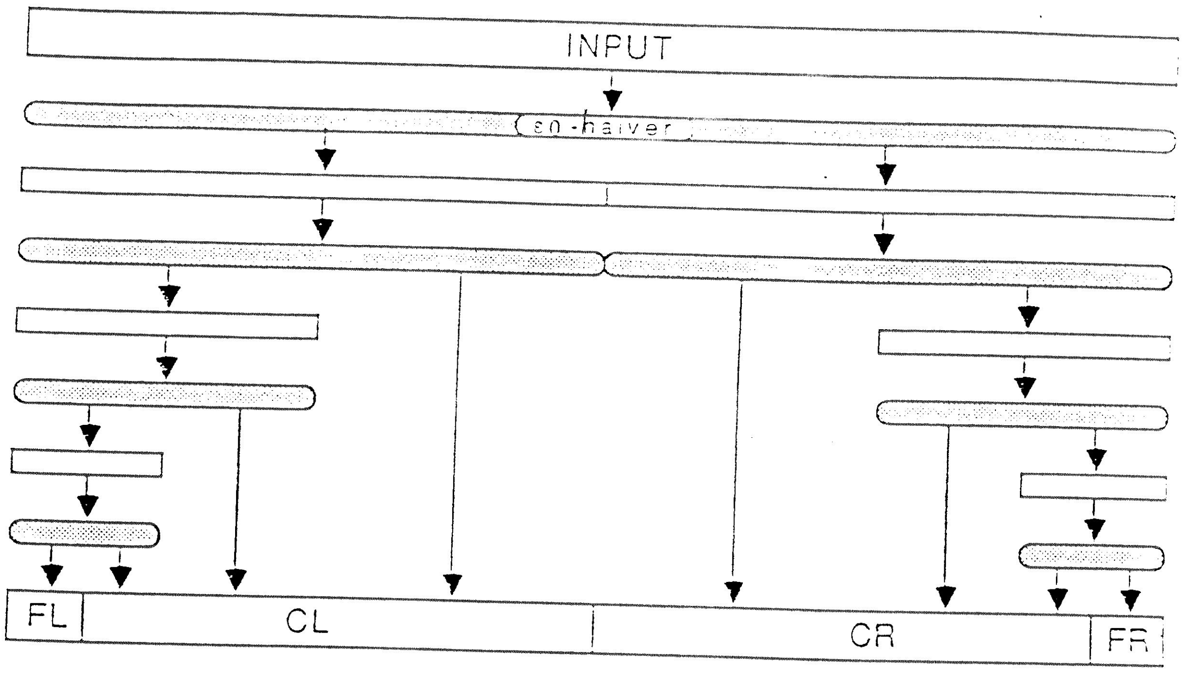

Fig. 1. Construction of a $(1/8, \varepsilon, \varepsilon_0)$-separator from a sequence of ε_0-halvers

(i) For sample parameters we will take: $p = 4, \lambda = 1/8, \varepsilon_0 = 1/72$, and $\varepsilon = 1/18$.

At various places in our construction we want to sort sets of a small constant size. It is convenient to use there the sorting network due to Batcher [3], the depth of which for m elements is approximately $(\log_2 m)^2/2$.

4. The network

The sorting network is structured about a complete binary tree which we shall imagine with the root at the top and leaves below. Associated with each node of the treee is a **bag** which contains a number of the elements being sorted. The **capacity** of a bag is the maximum number of elements which can be stored in that bag. For most of the algorithm each bag is either empty or filled to its capacity. With the root considered to be at level 0 in the tree, the capacity of each bag at level d is $r.A^d$ for some constant A, and some value of r decreasing with time.

(ii) For example: $A = 3$.

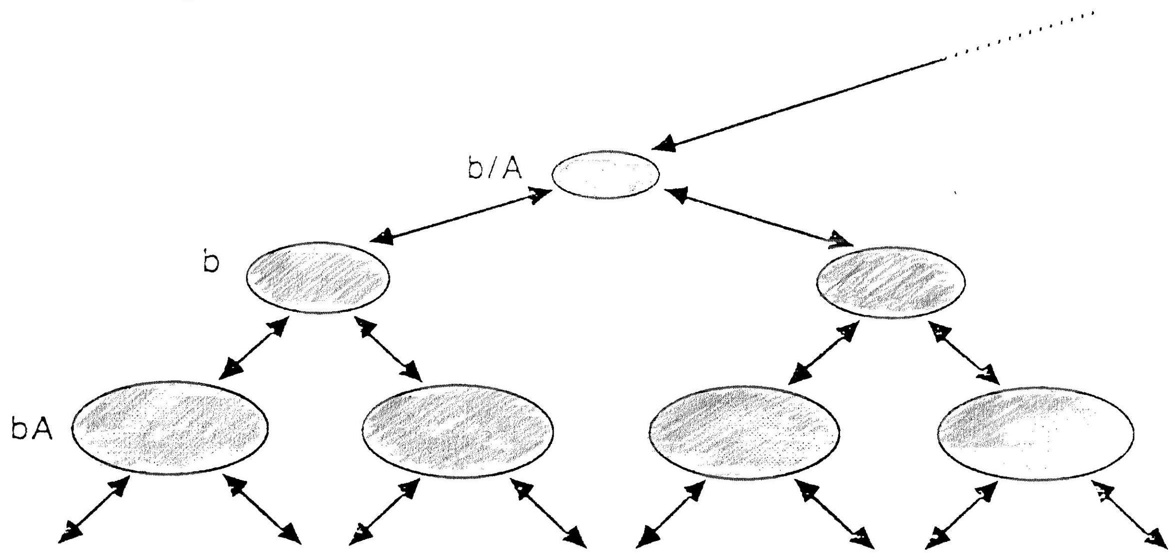

Fig. 2. Tree structure of bags.

Special situations occur at the topmost and lowest nonempty levels of the tree so we start with a description of the sorting process at intermediate levels. The algorithm works in **stages** beginning at stage 1. At odd stages all the bags at odd levels are empty and the bags at (some) even levels are full, while the opposite holds at even stages. At each stage the elements in any full bag are partitioned by a separator, the far-left and far-right parts are sent up to the parent bag and the centre-left and centre-right parts are transferred down to the left and right daughter bags respectively.

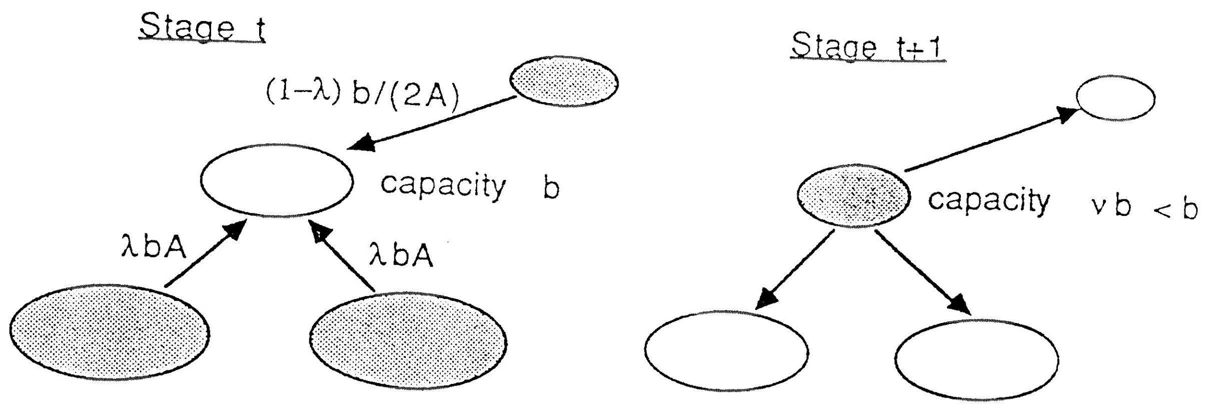

Fig. 3. Reduction of bag capacities after each stage.

Consider a bag, empty with capacity b at the beginning of some stage which is filled to its new capacity vb at the end of the stage. Thus:

$$vb = 2\lambda bA + (1-\lambda)b/(2A).$$

We require $v < 1$ so that the capacities diminish at each stage and elements are squeezed down the tree in the course of the algorithm, i.e.

(1) $$v = 2\lambda A + (1-\lambda)/(2A) < 1$$

(iii) With our sample parameters: $v = 43/48$.

The capacity of each bag at level d after stage t will be $c \cdot v^t \cdot A^d$ for some constant c.

To each bag there corresponds naturally an interval within the sorted order of the elements. The root bag corresponds to the whole ordered set, its left and right daughter bags correspond to the left and right halves of the ordering, and, for example, the bag reached by taking the path LRL down from the root corresponds to the third eighth from the left. To work out the **strangeness** of some element in a particular bag we count the number of steps up the tree from that bag towards the root which are needed to arrive at a bag within whose natural interval the element lies. Thus:

(i) all elements in the root bag have strangeness zero;

(ii) the strangeness of any element if nonzero is decreased by one if the element is sent up to the parent bag; and

(iii) when any element is sent down to a daughter bag its strangeness increases by one, except only when its strangeness is zero and it is sent down to the 'correct' daughter in which case its strangeness remains zero.

For any bag B and integer $j > 0$, we define $S_j(B)$ at some time to be the number of elements currently in B with strangeness j or more, expressed as a proportion of the capacity of B. The invariant that we maintain in order to assure the correctness of our construction is that:

(2) $$S_j(B) < \mu \cdot \delta^{j-1} \quad \text{for all } B \text{ and for all } j \geq 1.$$

(iv) For example, $\mu = 1/36$, $\delta = 1/40$.

In [1] the authors' concern was solely with an existence proof and they use corresponding parameters $\mu = 10^{-74}$, $\delta = 10^{-6}$.

At each stage the elements in each bag are partitioned by a $(\lambda, \varepsilon, \varepsilon_0)$-separator, the parts FL and FR are sent up to its parent; CL and CR are sent down to its left and right daughter respectively. Assuming that (2) held at the previous stage we can give an upper bound on the number of elements of strangeness j or more in some bag B at the following stage. Suppose the new capacity of B to be vb and the old capacities of B's parent bag and daughter bags to be b/A and bA respectively. For $j > 1$ the elements of strangeness j or more that find their way into B are either elements of strangeness $j+1$ or more from the daughter bags, or elements of strangeness $j-1$ or more which are sent down the wrong way by the parent bag. If we ensure that the FL and FR parts of the separator are each sufficiently large to accommodate any elements of positive strangeness then only at most a proportion ε of these are sent down. We therefore require that:

(3) $$\mu \leq \lambda/2.$$

Then the new strangeness of B, $S'_j(B)$, satisfies:

$$S'_j(B) \cdot vb < 2bA\mu\delta^j + \varepsilon(b/A)\mu\delta^{j-2} \quad \text{for } j > 1.$$

To ensure that (2) holds for the new stage we will choose parameters satisfying:

(4) $$2A^2\delta^2 + \varepsilon \leq vA\delta.$$

(v) With $A=3$, $\delta=1/40$, $\varepsilon=1/18$, $v=43/48$, we have $9/800 + 1/18 < 43/640$, and (4) holds.

Proving the required bound for $S'_1(B)$ is more complicated. The first term, $2bA\mu\delta$, appears just as before, representing the import of elements of strangeness two or more from B's daughters. Now however, some elements are misdirected downwards to B by B's parent not only because of mistakes by the separator but also because the parent bag may contain *too many* of the elements appropriate to B's sister, C say. Let this set of elements of strangeness zero with respect to C be denoted by V. The 'natural' location for V would be the whole contents of the subtree rooted at C together with one half of the contents of B's parent, one eighth of her greatgrandparent, and so on, assuming for the present an infinite chain of ancestors. The sizes of the bags will be such that V would fit exactly in this space. In reality some elements of V may have been displaced from this area and so might be occupying more than half of B's parent's bag.

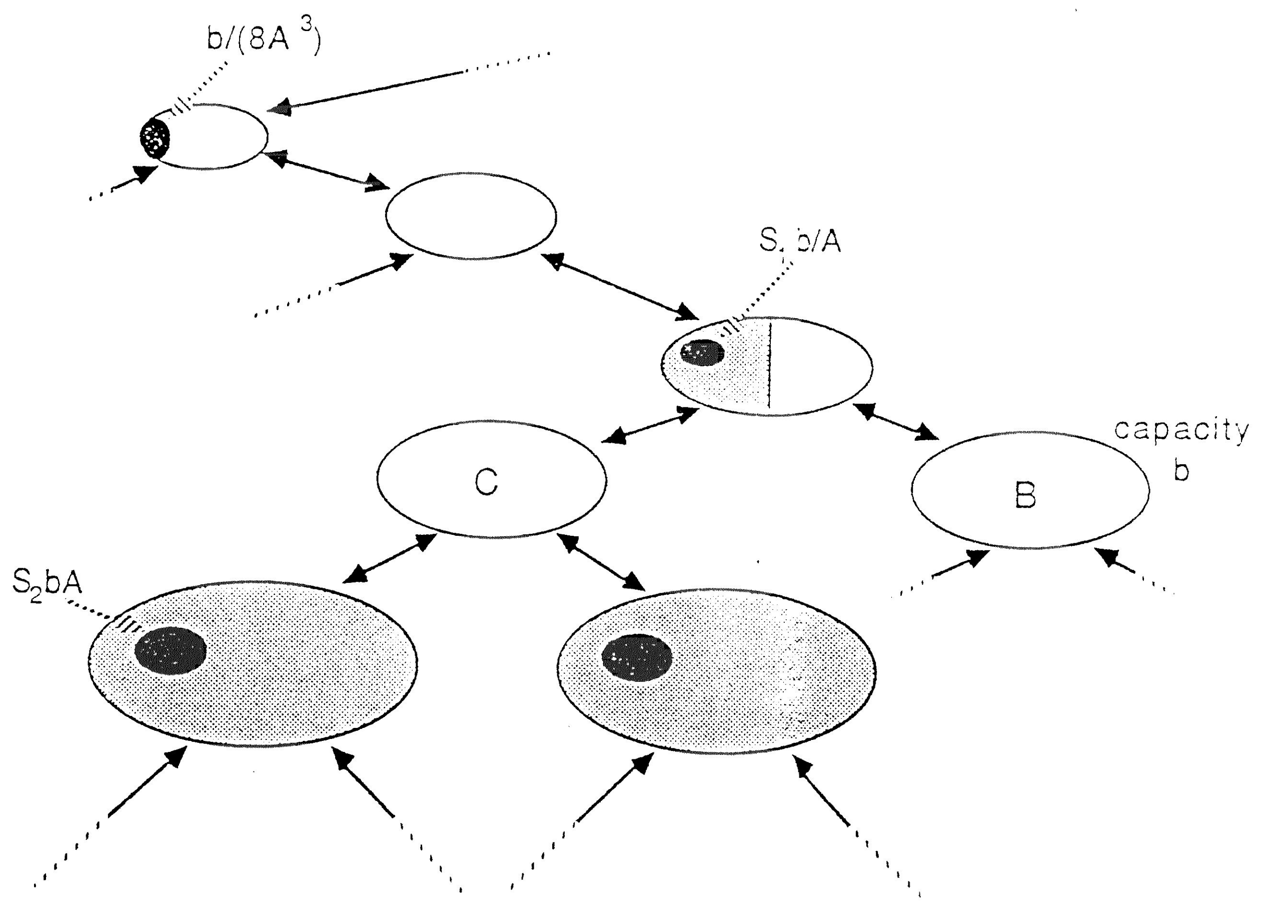

Fig.4. Sources of errors for bounding $S_1(B)$

The bag of a daughter of C may contain up to S_2bA elements from outside V. A bag two levels further down the tree may contain S_4bA^3 such elements, and so on. The total number of elements thus intruding into V's area in levels lower than B is at most

$$2S_2bA + 8S_4bA^3 + 32S_6bA^5 + \ldots < 2\mu\delta bA/(1 - 4\delta^2A^2).$$

The area above B's parent appropriate to V may in fact contain no elements of V. The total space here amounts to

$$b/(8A^3) + b/(32A^5) + \ldots = b/(8A^3 - 2A).$$

In addition B's parent may contain up to S_1b/A ($< \mu b/A$) strangers.

Thus, even if the initial halving of B's parent were done perfectly, a surplus of elements of V and other strangers with respect to B may spill across into B. The surplus is bounded by the sum of the three terms identified immediately above. Splitting error adds a further term of $b\varepsilon_0/(2A)$ since up to that number of elements may be misplaced in the initial split. Thus we have:

$$S'_1(B)\cdot v < 2\mu\delta A + 2\mu\delta A/(1 - 4\delta^2A^2) + 1/(8A^3 - 2A) + \mu/A + \varepsilon_0/(2A).$$

Since we require $S'_1(B) < \mu$, our parameters must satisfy:

(5) $$2\mu\delta A^2 + 2\mu\delta A^2/(1 - 4\delta^2A^2) + 1/(8A^2 - 2) + \mu + \varepsilon_0/2 \leq v\mu A$$

(vi) The choice $\varepsilon_0 = 1/72$ suffices since $1/80 + 5/391 + 1/70 + 1/36 + 1/144 < 43/576$.

5. Boundary situations

During the course of the algorithm the elements migrate down through the tree. We will arrange that there is at most one partially filled level. Above this, the levels are alternately empty and full as already described; below, all levels are empty. To this end we require that at this **partial level** each separator should send up to its parent bag the normal number of elements if it has sufficiently many. After this requirement is met, any remaining elements can be sent down to the daughter bags in equal numbers. Since the strangeness bounds are expressed as a proportion of bag capacity rather than as a proportion of the elements present, it is easy to satisfy these bounds at the partial level. To ensure that no more than the permitted number of elements with a certain strangeness are sent down, it suffices to adapt the usual separators in the following manner. After the initial split into halves we introduce two sets of 'virtual elements' to make up the full number of elements in each half. The virtual elements added to the left half are always 'more right' in comparisons with existing elements of the left half, whereas the virtual elements of the right half are 'more left' than the real elements of that half. The modified separator is obtained by deleting from the usual separator all of those comparisons involving virtual elements and routing the real elements appropriately. If this results in any virtual elements reaching FL or FR then they should now be replaced by arbitrary real elements from CL or CR respectively.

We must check that the strangeness invariants are satisfied around the partial level. For S'_1, the initial split will leave no greater *number* of the elements than usual in the 'wrong' half, though the *proportion* of these elements may be greater. For S'_j, $j > 1$, the introduction of the virtual elements into each half serves only to assist the strange elements into parts FL and FR, and the number of strange elements sent down will not be excessive.

The other abnormal boundary level is at the root of the tree. Here we would like the root node to behave as if it were an ordinary interior node of the tree. We therefore keep above it a subset of the elements, the **cold storage**, with which the root exchanges elements as with a parent. No comparisons are required within the cold storage, and the strangeness invariants are satisfied at the root since by definition all elements have strangeness zero there. The arguments in Section 4 involving an infinite chain of ancestors carry through if we regard the cold storage as simulating half the root's parent, one quarter of her grandparent, and so on. The capacity of the cold storage is therefore to be:

$$r/(2A) + r/(8A^3) + \dots = 2Ar/(4A^2 - 1) \qquad \text{at even stages,}$$

and:

$$r/(4A^2) + r/(16A^4) + \dots = r/(4A^2 - 1) \qquad \text{at odd stages,}$$

where r is the capacity at the root. To begin the algorithm $N(1-1/(4A^2))$ of the input elements are placed in the root bag and the remaining $N/(4A^2)$ elements are considered to be the cold storage.

6. Final stages

Stages proceed as described, with each bag getting smaller and the elements migrating down the tree, until the size of the root bag gets sufficiently small for us to split the tree in two. At an odd stage, when all the odd levels are empty, the number of elements in a bag at level $2j$ which are in the wrong half of the tree is bounded above by:

$$rA^{2j}S_{2j} < rA^{2j}\mu\delta^{2j-1} \leq rA^2\mu\delta$$

provided we have:

(6) $A\delta \leq 1.$

Therefore this number of elements is less than one, i.e. zero, by the time r has been reduced to $1/(A^2\mu\delta)$.

(vii) We have $A\delta = 3/40 \leq 1$ and the critical value of $r = 1/(A^2\mu\delta) = 160$.

At such a time then, there are no elements in the wrong half of the tree, so we *exactly* separate the set of all elements in the root bag and the cold storage into a left and a right half. From these halves the root and cold storage for each subtree can be immediately formed. To achieve the exact separation for a set of bounded size a Batcher sorting network [3] may be used. From this stage on a new root-splitting step will be required at regular bounded intervals, resulting in a rapidly growing forest of independent subtrees. Finally, when these subtrees reach a conveniently small bounded size, each can be exactly sorted to give the final result.

To estimate the total number of stages required by the algorithm, we note that the bags at level $\log N$ are initially of capacity $\Theta(N \cdot A^{\log N})$, where our logarithms are to base 2. At the last stage of the algorithm these bags are of capacity $\Theta(1)$, since they are within a bounded number of levels of bags of bounded capacity. Thus the number, k, of stages of the algorithm satisfies:

$$N \cdot A^{\log N} \cdot \nu^k = \Theta(1),$$

i.e. $$k = \log_2 N \cdot \log(2A)/(-\log \nu) + O(1).$$

(viii) We have: $\log(2A)/(-\log \nu) = (\log 6) / \log (48/43) < 17.$

7. Integer rounding

In previous sections we have specified the 'ideal' sizes of various sets of elements as real numbers and it remains to show how to pick integer sizes satisfying all the constraints. Provided that the sizes chosen are all very close to the ideal sizes and that the capacities of all the bags are sufficiently large, the inequalities governing our constants will still be satisfied by our proposed values. Since the strangeness bounds are expressed as a proportion of the *capacity* of a bag the possibly small actual sizes at the partial level cause no special difficulty. The smallest capacities in our algorithm arise at the root of a tree when we are about to split the tree. (For example, in Section 6 the critical root capacity was given as 160 for our parameters.) In case the corresponding value for some choice of paramaters is too small to absorb the effects of rounding, the root-splitting may be done at an earlier stage. Referring to Section 6, we see that no elements at or below level $2j$ are in the wrong half of the tree provided that:

$$r \leq 1/(A^{2j}\mu\delta^{2j-1}).$$

At an odd stage the number of elements above level $2j$ is about $r(2A)^{2j}/(4A^2 - 1)$. If we use Batcher's method to sort exactly this set of elements (instead of just the root bag and cold storage) then the tree can be properly split. For all of the networks we describe later, $j = 2$ suffices.

(ix) With $A=3$, $\mu=1/36$, $\delta=1/40$, $j=2$, we could maintain $r > 28{,}000$ by sorting sets of size about one million.

It is convenient and efficient for the design of separators that the actual number of elements in each bag be even. Note that this can be achieved even when N is odd because of our use of cold storage. We provide a simple recipe for the integer sizes of the bags which will ensure that these cannot stray far from their ideal sizes. If the ideal total content of some *subtree* is α, then the actual content is to be $2\lceil\alpha/2\rceil$. Suppose then that for some node the ideal content of its subtree is α, while the ideal contents of its daughter subtrees are β. If the ideal and integer sizes of the bag at that node are b and $Z(b)$ respectively then:

$$b = \alpha - 2\beta, \quad Z(b) = 2\lceil\alpha/2\rceil - 4\lceil\beta/2\rceil, \text{ so } b - 4 \leq Z(b) < b+2.$$

Since only the partial level of bags can ever become very small there is no difficulty in maintaining the relationship. Thus $Z(b)$ is even and the close agreement between b and $Z(b)$ satisfies our requirements.

We shall see that our ideal $(\lambda', \varepsilon, \varepsilon_0)$-separator, with say $\lambda' = 2^{-p+1}$, can be refined to take account of integer sizes and to allow the size of FL and FR to be arbitrary within the range $[\lambda' b/2, b/2]$, while maintaining the same error bounds. Thus we need only ensure that λ' is sufficiently small that our algorithm with parameter λ, say, never requires the size of FL or RL to be less than $\lambda' Z(b)/2$. Suppose that some full bag has ideal capacity b and the ideal content for the corresponding subtree is α. Our recipe specifies:

$$|FL| = |FR| = \lceil \alpha/2 \rceil - \lceil (\alpha - \lambda b)/2 \rceil > \lceil \alpha/2 \rceil - \alpha/2 + \lambda b/2 - 1,$$

whereas:

$$\lambda' Z(b)/2 < \lambda'(\lceil \alpha/2 \rceil - \alpha/2 + b/2),$$

and so it suffices to have:

$$\lambda \geq \lambda' + 2/b.$$

In the interests of clarity we ignored this consequence of rounding in our earlier description and identified λ and λ', but now values for λ and ν must be increased appropriately. The effect on our final constant depend on the minimum size of b.

(x) With $b \geq 160$ and $\lambda' = 1/8$, we need $\lambda \geq 0.1375$, and when ν increases correspondingly the number of stages more than triples. However with $b \geq 28{,}000$ the increase in the number of stages is less than 0.5%.

The principal motivation for making the sizes of bags even numbers is to simplify the analysis of halvers and to permit the relatively clean bound of the theorem in the next section. This is of significance only for the initial (symmetrical) halver step in our separators. The later halving steps are not applied to sets of even size but these can be dealt with by introducing a 'virtual' element as described below. Were this done in the first step also, it would degrade the performance unacceptably for small values of k.

Consider a halver which is on the left side of the separator structure and whose function therefore is to ensure, for some ε and for some range of values of k, that fewer than εk elements from the extreme left k of its input elements are output in the right (i.e. wrong) half. If the number of input elements is $2n - 1$ we introduce a new 'virtual' element which is considered to be to the right of all the real elements. After the application of a 'standard' halver for $2n$ elements, the virtual element will have been output to the right and is discarded. The specifications for the standard halver will have the same value of ε and only have to hold for the same range of values for k despite the number of inputs being greater.

Finally we note that whenever the size of FL and FR has to be increased to achieve the specified bag sizes we can transfer elements arbitrarily from CL to FL and from CR to FR without worsening the error bounds.

8. Separators and constants

We prepare for a more economical construction of separators by generalising the notion of halver which we defined in Section 3. For any $\varepsilon > 0$ and α, $0 < \alpha \leq 1$, an **(ε, α)-halver** for $m=2n$ elements is defined in the same way as an ε-halver except that the key property is only required to hold for sets of size k with $k \leq \alpha n$ instead of for all $k \leq n$. We shall see that this less stringent requirement allows a considerable economy in depth when $\alpha << 1$.

Theorem For $0 < \varepsilon \leq 1/2$, $0 < \alpha \leq 1$ and $n > 0$, there exists an (ε, α)-halver for $2n$ elements with depth $C(\alpha,\varepsilon)$ where:

$$C(\alpha, \varepsilon) = \lceil 1 + \left(h(\varepsilon\alpha) + h((1-\varepsilon)\alpha)\right) / \left(-\varepsilon\alpha \ln((1-\varepsilon)\alpha)\right)\rceil,$$

and:

$$h(x) = -x \ln x - (1-x) \ln (1-x).$$

The proof of this theorem is given in the Appendix. An essential feature of the above result is that an explicit depth is given which holds for all values of n. Previously published results such as [4] give just asymptotic values as n tends to infinity. Since our halvers are to work synchronously at all levels of the tree and since the root level is small for a significant proportion of the stages, this stronger result is necessary. We have, however, taken a slightly larger value for C than that given by the asymptotic bounds in order to simplify our analysis. Our proof does not yield an explicit halver network; all currently known explicit constructions require a much greater depth. See references [8], [5], and [7] for further information and recent work.

An easy analysis reveals that, when $\varepsilon \to 0$:

$$C(1, \varepsilon) \to -2\ln \varepsilon / \varepsilon,$$

while for fixed α, $0 < \alpha < 1$:

$$C(\alpha, \varepsilon) \to -h(\alpha)/(\varepsilon\alpha \ln \alpha).$$

The total depth of the network we have described is approximately

$$\log_2 N \cdot p \cdot C(1, \varepsilon_0) \cdot \log(2A)/(-\log v)$$

when we construct a $(2^{-p+1}, p\varepsilon_0, \varepsilon_0)$-separator using p levels of ε_0-halvers.

(xi) Our sample parameters $p=4$, $\lambda'=1/8$, $\varepsilon_0=1/72$, $A=3$, $\mu=1/36$ and $\delta=1/40$ yield a value that is just under $50{,}000 \log_2 N$.

It is possible to do a little better just by varying the parameters. For example we have found that the values $p=4$, $\lambda'=1/8$, $\varepsilon_0=3/223$, $A=2.7$, $\mu=1/16$ and $\delta=1/34$ can reduce the constant below 30,000. However we shall show next how this constant may be more substantially improved after a closer analysis of separators.

The error estimate ε is the sum of error contributions from each of the p levels. Suppose we allow an error of ε_i at level i for $i = 1,\dots, p$, where $\varepsilon = \varepsilon_1 + \varepsilon_2 + \dots + \varepsilon_p$. In the coarse analysis we used until now we adopted the pessimistic setting of $\alpha = 1$, and have used a halver of depth $C(1, \varepsilon_i)$ at level i. Referring to figure 1 again, we see that at level i the ratio of the number of strangers to the size of a half is at worst $(\mu : 1/2^i)$ so that we could actually construct a separator with total depth

$$C(2\mu, \varepsilon_1) + C(4\mu, \varepsilon_2) + \dots + C(2^p\mu, \varepsilon_p).$$

Note that our constraints ensure that $2^p\mu \le 1$.

For the ε_0-bound that also has to be assured by the first-level halver, we can again improve on the naive estimate of $\alpha = 1$. Referring back to the derivation of the bound on S'_1 in Section 4, we see that the number of elements of V in the bag of B's parent is at most

$$b(1 + \eta)/(2A) \quad \text{where } \eta = 4\mu\delta A^2/(1 - 4\delta^2A^2) + 1/(4A^2 - 1).$$

In addition there may be up to $S_1 b/A$ strangers. Since all but a proportion ε of 'right' strangers would get sent up the tree with FR, the worst case in the estimation of S'_1 is when there are $b\mu/A$ strangers and they are all to the 'left'. (This case can arise only when B is the right (left) daughter of a right (left respectively) daughter. In the situation shown in figure 4 any elements with strangeness *exactly* one for B's parent must be 'left' strangers. Taking account of this asymmetry could improve the bound slightly but would complicate our analysis.) At worst the number of *good* elements (i.e. those appropriate to B) in this bag is only $b(1 - \eta - 2\mu)/(2A)$, and so to limit to $\varepsilon_0/(2A)$ the further influx of strangers to B due to good elements being sent the wrong way the first level halver only requires depth $C(1 - \eta - 2\mu, \varepsilon_0)$ instead of our former estimate of $C(1, \varepsilon_0)$. The total depth needed for a separator is now reduced to

$$\max\{\ C(2\mu, \varepsilon_1), C(1 - \eta - 2\mu, \varepsilon_0)\ \} + C(4\mu, \varepsilon_2) + \ldots + C(2^p\mu, \varepsilon_p).$$

With this more refined construction the parameters

$$p=4,\ \lambda'=1/8,\ A=2.7,\ \mu=1/20,\ \delta=1/40,\ \varepsilon_0=1/59,\ \varepsilon_1=1/134,\ \varepsilon_2=1/85,\ \varepsilon_3=1/83,\ \varepsilon_4=1/62$$

yield a constant below 6500. This comes from a network with about $9.5 \log_2 N$ stages and separators of depth 678. The best constant we have found is under 6100 and achieved using parameters given approximately by:

$$p=5,\ A=4.75,\ \mu=1/50,\ \delta=1/57,\ \text{and for } i=0,\ldots,5,\ \ 1/\varepsilon_i = 62,\ 199,\ 110,\ 109,\ 106,\ 90,$$

to yield a network with about $6.15 \log_2 N$ stages each of depth less than a thousand.

9. Ideas for improvements

When we classify the contents of a bag we must expect a small proportion of elements from the middle range of values to be misclassified between CL and CR so that they are sent down to the wrong daughter bag and acquire strangeness 1. A more cautious classification would produce a borderline class which was to be retained in the same bag at the next stage. The emptiness at alternate levels in the construction described above was a convenient simplification but is not an essential feature. To incorporate the modifications suggested in this section all the bags above a certain level in the tree are maintained full to capacity. Consequent changes are required at the upper and lower extreme levels.

Consider the extreme left elements which may be found in a bag at a *left* daughter node in the tree. It will be clear that if these elements need to be moved left and have nonzero strangeness then they must have strangeness at least two and could usefully be sent up *two* levels at once. Similar reasoning holds for extreme right elements in a right daughter bag.

For the refinements suggested above there is a trade-off between, on the one hand, limiting the amount of misdirection and allowing the more rapid return of strangers, and, on the other, speeding the progress of correctly sorted elements down the tree towards the leaves. The constraints on the parameters are easily handled but the technology of producing efficient classifiers of the more complicated forms which are required becomes much more intricate.

10. Conclusion

We have tried to describe these depth $\Theta(\log n)$ sorting networks as cleanly as possible. The constraints on the parameters have been extracted as a simple set of inequalities. While the numerical bounds we have proved suggest that the present construction is still far from efficient, we hope that the framework presented will encourage further progress towards a practical algorithm.

Open Problem Develop an error analysis which measures more closely the performance of this kind of sorting network.

Conjecture The depth of network required for the correctness of our algorithm is overestimated hugely by our analysis.

References

[1] M. Ajtai, J. Komlós and E. Szemerédi, "An O(n log n) sorting network," *Proc. 15th Ann. ACM Symp. on Theory of Computing* (1983) 1-9.

[2] M. Ajtai, J. Komlós and E. Szemerédi, "Sorting in C log N parallel steps," *Combinatorica* 3(1983) 1-19.

[3] K. Batcher, "Sorting networks and their applications," *AFIPS Spring Joint Computer Conference* 32(1968) 307-314.

[4] F. Chung, "On concentrators, superconcentrators, generalizers, and nonblocking networks," *The Bell System Technical Journal* 58.8(1978) 1765-1777.

[5] O. Gabber and Z. Galil, "Explicit constructions of linear-sized superconcentrators," *J. Comp. Sys. Sci.* 22(1981) 407-420.

[6] D.E. Knuth, **The Art of Computer Programming**, Vol.3, "Sorting and Searching" (Addison-Wesley 1973).

[7] A. Lubotzky, R. Phillips and P. Sarnak, "Explicit expanders and the Ramanujan conjectures," *Proc. 18th Ann. ACM Symp. on Theory of Computing* (1986) 240-246.

[8] G.A. Margulis, "Explicit constructions of concentrators," *Problemy Inf. Trans.* 9(1973) 325-332.

[9] H. Robbins, "A remark on Stirling's formula," *Amer. Math. Monthly* 62(1955) 26-29.

Appendix (Proof of Halver Theorem)

Theorem For $0 < \varepsilon \leq 1/2$, $0 < \alpha \leq 1$ and $n > 0$, there exists an (ε, α)-halver for $2n$ elements with depth $C(\alpha, \varepsilon) = \lceil G(\alpha, \varepsilon) \rceil$ where:

$$G(\alpha, \varepsilon) = 1 + \big(h(\varepsilon\alpha) + h((1-\varepsilon)\alpha)\big) / \big(-\varepsilon\alpha \ln((1-\varepsilon)\alpha)\big),$$

and:

$$h(x) = -x \ln x - (1-x) \ln (1-x).$$

Proof We shall describe the halving networks in terms of comparison algorithms using storage registers and compare-and-exchange operations on pairs of these registers. This is an equivalent model to that with wires and comparators used above. The registers are divided into equal sets A and B with n registers in each. At every step of the algorithm, comparisons are made between the contents of n disjoint pairs of registers, one from A and one from B. For each pair an exchange is made if necessary so that the more 'left' value is put in the A-register and the more 'right' in the B-register. Suppose that the algorithm proceeds in this way for c steps and at step i, $1 \leq i \leq c$, the set of pairs to be compared is given by the bijection $g_i : A \rightarrow B$, i.e. for all $a \in A$, registers $a \in A$ and $g_i(a) \in B$ are compared. We will find that for c sufficiently large *most* of the algorithms of this form satisfy our halving condition. The total number of distinct algorithms with c steps is exactly $(n!)^c$. We calculate an upper bound for the number of such algorithms which *fail* to be (ε, α)-halvers.

Consider a failing algorithm Q and some set of inputs for which Q fails. Thus there is a $k \leq \alpha n$ such that the set S of the k rightmost (or lefmost) input elements is badly distributed at the end of the algorithm, i.e. some set X of A-registers (or B-registers respectively) of size $r = \lceil \varepsilon k \rceil$ still contains r elements of S. Since for any r we may as well choose k maximal subject to $r = \lceil \varepsilon k \rceil$ and $k \leq \alpha n$, we can assume $k = \max\{ \lfloor r/\varepsilon \rfloor, \lfloor \alpha n \rfloor \}$. A sufficient set of pairs $< r,k >$ to cover all possible failing algorithms is given by:

$$R(n, \varepsilon, \alpha) = \{< r,k > \mid 1 < r < \varepsilon p, k = \lfloor r/\varepsilon \rfloor \} \cup \{< \lceil \varepsilon p \rceil, p >\} \text{ where } p = \lfloor \alpha n \rfloor.$$

This follows from the inequalities:

$$\lfloor \lceil \varepsilon p \rceil / \varepsilon \rfloor \geq p \quad \text{and} \quad \lfloor (\lceil \varepsilon p \rceil - 1) / \varepsilon \rfloor < p.$$

Without loss of generality we consider the case $X \subseteq A$. It is clear that the contents of any A-register can only become more 'left' as the result of an exchange, whilst B-registers become more 'right'. Therefore at the beginning of the algorithm X contained elements from S, and furthermore every B-register compared with any register in X during the algorithm must have also contained an element of S at the time and so will still do so at the end of the algorithm. Denoting the whole set of such B-registers by $Q(X)$, we have shown that:

$$|Q(X)| \leq k - r.$$

For any $X \subseteq A$ with $|X| = r$, and $Y \subseteq B$ with $|Y| = k - r$, the number of bijections $g : A \to B$ such that $g(X) \subseteq Y$ is

$$(k-r)!(n-r)!/(k-2r)!$$

Taking into account the number of possible choices for r, X and Y, and the alternatives of $X \subseteq A$ or $X \subseteq B$, we obtain an upper bound for the proportion, $P(c)$, of failing algorithms with c steps of:

$$P(c) \leq 2\sum_{\langle r,k\rangle \in R(n,\varepsilon,\alpha)} \binom{n}{r}\binom{n}{k-r}\left(\binom{k-r}{r}\Big/\binom{n}{r}\right)^{c}$$

$$\leq 2\sum_{R(n,\varepsilon,\alpha)} \binom{n}{r}\binom{n}{k-r}\left((k-r)/n\right)^{rc}$$

We define h on the interval $[0,1]$ by:

$$h(x) = -x \ln x - (1-x)\ln(1-x) \qquad \text{for } 0 < x < 1, \text{ and}$$

$$h(0) = h(1) = 0.$$

h is a familiar entropy function except that we find it more convenient here to use natural logarithms. The approximation we shall use for binomial coefficients is given in the following lemma.

Lemma For $n > 0$ and $0 < r < n$,

$$\ln\binom{n}{r} < n\cdot h(r/n) - \tfrac{1}{2}\ln(\min\{r, n-r\}) - \tfrac{1}{2}\ln \pi.$$

Proof A result of Robbins [9] yields the following inequalities:

$$1/(12n+1) < \ln n! + n - n\ln n - \tfrac{1}{2}(\ln 2\pi + \ln n) < 1/(12n) \quad \text{for } n > 0.$$

Hence for $0 < r < n$,

$$\ln\binom{n}{r} - n\cdot h(r/n) + {}^1/_2\ln(\min\{r, n-r\})$$
$$< -{}^1/_2\ln(\max\{r, n-r\}) + {}^1/_2\ln n - {}^1/_2\ln 2\pi + 1/(12n) - 1/(12r+1) - 1/(12(n-r)+1).$$
$$< -{}^1/_2\ln \pi. \quad \blacklozenge$$

Now,

$$\ln\left\{\binom{n}{r}\binom{n}{k-r}((k-r)/n)^{rc}\right\}$$

$$< n\{h(r/n) + h((k-r)/n) + cr/n\cdot \ln((k-r)/n)\} - {}^1/_2\ln r - {}^1/_2\ln(\min\{k-r, n-k+r\}) - \ln \pi$$

$$\leq r\ln((k-r)/n) - \ln r - \ln \pi \quad \text{since} \quad \min\{k-r, n-k+r\} \geq r,$$

provided we choose a value for c satisfying:

$$c \geq 1 + \big(h(r/n) + h((k-r)/n)\big)/\big(-r/n\cdot \ln((k-r)/n)\big) = G(k/n, r/k).$$

If we can choose c to satisfy this inequality for all values of r and k in the summation then:

$$P(c) < \frac{2}{\pi}\sum_{R(n,\varepsilon,\alpha)}\frac{1}{r}\left(\frac{k-r}{n}\right)^r \leq \frac{2}{\pi n}\sum_{R(n,\varepsilon,\alpha)}\frac{k-r}{r}$$

$$< \frac{2\lceil\varepsilon p\rceil}{\pi n}\cdot\frac{1}{\varepsilon} \leq \frac{2}{\pi}\left(1+\frac{1}{\varepsilon n}\right) < 1 \quad \text{for } \varepsilon n \geq 2.$$

If $\varepsilon n < 2$ then $P(c) < 1$ follows immediately since $r \leq \lceil\varepsilon p\rceil \leq 2$.

All that remains to be proved is that $c = \lceil G(\alpha, \varepsilon)\rceil$ is an adequate choice for c. Since all $<r, k>$ in $R(n, \varepsilon, \alpha)$ satisfy $r/k \geq \varepsilon$ and $k/n \leq \alpha$, the inequality, $c \geq G(k/n, r/k)$, is satisfied provided that $G(\alpha, \varepsilon)$ is an increasing function of α and a decreasing function of ε.

Lemma For all $0 < \alpha \leq 1$, $0 < \varepsilon < 1$,

(i) $\partial G/\partial\varepsilon < 0$,

and (ii) $\partial G/\partial\alpha > 0$ if $\varepsilon \leq {}^1/_2$.

Proof

Let $u = h(\varepsilon\alpha) + h((1-\varepsilon)\alpha)$, $v = -\varepsilon\alpha \ln((1-\varepsilon)\alpha$, and $F = u/v$. Then $u,v > 0$ and $G(\alpha,\varepsilon) = 1 + F$.

Now,

$$\varepsilon v \partial F/\partial\varepsilon = \varepsilon(\partial u/\partial\varepsilon - F\partial v/\partial\varepsilon)$$

$$= -\varepsilon\alpha \ln(\varepsilon\alpha) + \varepsilon\alpha \ln(1-\varepsilon\alpha) - \varepsilon\alpha \ln((1-\varepsilon)\alpha) + \varepsilon\alpha \ln(1-(1-\varepsilon)\alpha) - u - F\varepsilon^2\alpha^2/(1-\varepsilon)$$

$$= \varepsilon\alpha \ln(1-\varepsilon\alpha) + \alpha \ln((1-\varepsilon)\alpha) + (1-\alpha) \ln(1-(1-\varepsilon)\alpha) - F\varepsilon^2\alpha^2/(1-\varepsilon)$$

< 0 for all $0 < \alpha \leq 1$, $0 < \varepsilon < 1$, and (i) is proved.

Similarly for (ii), if $\varepsilon \leq 1/2$,

$$\alpha v \partial F/\partial\varepsilon = \alpha(\partial u/\partial\alpha - F\partial v/\partial\alpha)$$

$$= -\varepsilon\alpha \ln(\varepsilon\alpha) + \varepsilon\alpha \ln(1-\varepsilon\alpha) - (1-\varepsilon)\alpha \ln((1-\varepsilon)\alpha) + (1-\varepsilon)\alpha \ln(1-(1-\varepsilon)\alpha) - u + F\varepsilon\alpha$$

$$= \ln(1-\varepsilon\alpha) + \ln(1-(1-\varepsilon)\alpha) + F\varepsilon\alpha$$

$\geq \ln(1-\varepsilon\alpha) - h(\varepsilon\alpha)/\ln(\varepsilon\alpha) + \ln(1-(1-\varepsilon)\alpha) - h((1-\varepsilon)\alpha)/\ln((1-\varepsilon)\alpha)$ since $1-\varepsilon \geq \varepsilon$

> 0 since $h(y) > \ln y \cdot \ln(1-y)$ for $0 < y < 1$.

This inequality for h follows from the substitutions $z = y$ and $z = (1-y)$ in the simple inequality:

$$z > -(1-z) \ln(1-z) \text{ for } 0 < z < 1. \quad ♦$$

Establishing this lemma completes the proof of the Halver Theorem. ♦