# MAX-PLANCK-INSTITUT FÜR INFORMATIK

## On a Compaction Theorem of Ragde

Torben Hagerup

MPI-I-91-121 November 1991


INFORMATIK

Im Stadtwald
W 6600 Saarbrücken
Germany

# On a Compaction Theorem of Ragde

Torben Hagerup

MPI–I–91–121 November 1991

# On a Compaction Theorem of Ragde

Torben Hagerup *

*Max-Planck-Institut für Informatik, W–6600 Saarbrücken, Germany*

Ragde demonstrated that in constant time a PRAM with $n$ processors can move at most $k$ items, stored in distinct cells of an array of size $n$, to distinct cells in an array of size at most $k^4$. We show that the exponent of 4 in the preceding sentence can be replaced by any constant greater than 2.

*Keywords*: parallel algorithms, approximate compaction

## 1. Introduction

The problem of *approximate compaction* has been the focus of a great deal of attention lately [10, 8, 6, 7]. For $n, s \in \mathbb{R}$ and $k \in \mathbb{N}$, define the $(n, k, s)$-*compaction* problem as follows:

Given the integer $k$ and an array $A$ of at most $n$ cells containing at most $k$ items stored in distinct cells (the remaining cells of $A$ contain nothing), move the items in $A$ to distinct cells in an array of at most $s$ cells.

For reasons of convenience, we allow $n$ and $s$ to have noninteger values. The problem is of interest for $k \leq s < n$; intuitively, we are trying to "compact" the items, i.e., to move them more closely together, thereby reducing the number of empty cells between them. Compaction has always been fundamental in efficient parallel algorithms. Ideally, we would like to compact exactly, i.e., to move however many items are present in an array of size $n$, this quantity not being specified as part of the input, to an array whose size exactly equals the number of items. Exact compaction reduces to prefix summation and can therefore be performed in $O(\log n/\log\log n)$ time on a CRCW PRAM [3]. On the other hand, computing the parity of $n$ bits reduces to exact compaction, which therefore requires $\Omega(\log n/\log\log n)$ time with any polynomial number of processors [2]. As it turns out, however, approximate compaction can be done in constant time. This was first realized by Ragde [11], who showed that $(n, k, k^4)$-compaction problems can be solved in constant time with $n$ processors. Ragde's result triggered the development of a number of very fast randomized algorithms for fundamental problems [12, 10, 8, 9, 1, 6, 7]. Matias and Vishkin [10] claimed without proof the ability to solve $(n, k, k^{2.829})$-compaction problems in constant time with $n$ processors. We improve this result by showing that $n$ processors can solve $(n, k, k^{2+\epsilon})$-compaction problems in constant time, for arbitrary fixed $\epsilon > 0$. Although we shall not give any details, this can be used to reduce the probability of failure in almost all of the very fast algorithms mentioned above.

* Supported in part by the Deutsche Forschungsgemeinschaft, SFB 124, TP B2, VLSI Entwurfsmethoden und Parallelität, and in part by the ESPRIT II Basic Research Actions Program of the EC under contract No. 3075 (project ALCOM).

## 2. The algorithm

Our model of computation is the ARBITRARY CRCW PRAM [4], a synchronous parallel machine with a global memory accessible to all processors. Concurrent reading from the same cell is allowed, and in the event of concurrent writing to the same cell, some (arbitrary) processor succeeds and writes its value. We assume constant-time operations for integer addition, subtraction, multiplication and division with remainder. All algorithmic steps described in this section are supposed to be executed in constant time, i.e., in time independent of $n$ and $k$; this will not be stated explicitly on every occasion. As an aid to seeing that constant time suffices, note the simple fact that for every fixed rational number $q$, the function $x \mapsto \min\{n, \lfloor x^q \rfloor\}$ can be evaluated in constant time for arguments in $\mathbb{N}$ on a CRCW PRAM with $n$ processors.

When placing items in an array of size at most $s$, we speak of mapping or compacting them into space $s$. A cell or array will be called *nonempty* exactly if it contains at least one item.

Ragde's algorithm, as well as ours, is based on the following result, proved by Fredman, Komlós and Szemerédi.

**Lemma** 1 [5]: Let $m, k \in \mathbb{N}$ and let $p > m$ be a prime. Then for every subset $X$ of $\{1, \ldots, m\}$ with $|X| = k$, there exists an integer $a$ with $1 \leq a < p$ such that the restriction to $X$ of the mapping $x \mapsto (ax \bmod p) \bmod k^2$ is injective.

For every $m \in \mathbb{N}$, the set $\{m, \ldots, 2m\}$ contains at least one prime, and a prime in this set can be found in constant time with $m^2$ processors: Simply test each number in the set against every possible divisor. Therefore Lemma 1 immediately furnishes an algorithm for $(n, k, k^2)$-compaction that uses $O(n^2)$ processors: Compute a prime $p$ with $p = O(n)$, then test all possible values of $a$ in parallel and choose one whose associated mapping is injective when restricted to the set of (indices of) nonempty cells. Our remaining effort aims at reducing the number of processors used to $O(n)$.

Ragde proceeded from Lemma 1 to show the following:

**Lemma** 2 [11]: For arbitrary $r \in \mathbb{R}$ with $r = O(\sqrt{n})$, $O(n)$ processors can solve $(r\sqrt{n}, k, k^2 r)$-compaction problems in constant time.

**Proof**: Divide the input array into $O(\sqrt{n})$ subarrays of size at most $r$ each. Compute the set of (indices of) nonempty subarrays, which is of cardinality at most $k$, and use Lemma 1 with $m = O(\sqrt{n})$ to map it into an array of $k^2$ *blocks*, where a block is simply a set of consecutive cells temporarily considered as one entity. Making each block just large enough to hold one subarray, we have mapped the items to an array of size at most $k^2 r$. ∎

Ragde's original result on $(n, k, k^4)$-compaction follows by two successive applications of Lemma 2: first the $(\sqrt{n} \cdot \sqrt{n}, k, k^4)$-compaction problem is reduced to a $(k^2\sqrt{n}, k, k^4)$-compaction problem ($r = \sqrt{n}$), then the $(k^2\sqrt{n}, k, k^4)$-compaction problem is solved ($r = k^2$). The second application of Lemma 2 is legal only if $k = O(n^{1/4})$, but for $k \geq n^{1/4}$ the original problem is trivial.

Our proof uses similar ideas. In particular, we divide the input array into roughly $\sqrt{n}$ subarrays of $\sqrt{n}$ cells each. Before we compact the set of (indices of) nonempty subarrays, however, we recursively attempt to compact each subarray into less space, which will allow us to use smaller blocks. Since we do not know how many items to expect in a subarray, we try to compact it into arrays of many different sizes and use the smallest one into which the items in the subarray will fit. Of course, this requires us to provide blocks of different sizes. Another important point is that since we aim for a constant running time, we cannot let the recursion run until subproblems have been reduced to constant size, which would take $\Theta(\log\log n)$ steps. Instead we finish off the recursion in constant depth by an application of Ragde's algorithm. We now give the details.

For all $\epsilon \in \mathbb{R}_+$, denote by $P(\epsilon)$ the assertion

$P(\epsilon)$: $(n, k, k^{2+\epsilon})$-compaction problems can be solved in constant time with $O(n)$ processors.

Our goal ist to show that $P(\epsilon)$ holds for all $\epsilon \in \mathbb{R}_+$. To this end, let $f : \mathbb{R}_+ \to \mathbb{R}_+$ be the continuous function given by

$$f(\epsilon) = \frac{\epsilon - 2 + \sqrt{5\epsilon^2 + 12\epsilon + 4}}{4 + \epsilon},$$

for all $\epsilon \in \mathbb{R}_+$. We will show that $f$ has the following properties:

(A) For all $\epsilon, \epsilon' \in \mathbb{R}_+$, $\epsilon < \epsilon' \Rightarrow f(\epsilon) < f(\epsilon')$;

(B) For all $\epsilon \in \mathbb{R}_+$, $f(\epsilon) < \epsilon$;

(C) For all $\epsilon, \beta \in \mathbb{R}_+$, $P(\epsilon) \Rightarrow P(f(\epsilon) + \beta)$.

We first assume that (A)–(C) hold and demonstrate that they imply the desired result. We begin by showing the following generalization of property (C), where $f^{(i)}$ denotes the function defined as $i$-fold repeated application of $f$.

(D) For all $\epsilon, \beta \in \mathbb{R}_+$ and for all $i \in \mathbb{N}$, $P(\epsilon) \Rightarrow P(f^{(i)}(\epsilon) + \beta)$.

We prove (D) by induction on $i$. For $i \in \mathbb{N}$, let $Q(i)$ denote the assertion

$Q(i)$: For all $\epsilon, \beta \in \mathbb{R}_+$, $P(\epsilon) \Rightarrow P(f^{(i)}(\epsilon) + \beta)$.

$Q(1)$ is just (C). Hence let $i \in \mathbb{N}$, assume that $Q(i)$ holds and let $\epsilon, \beta \in \mathbb{R}_+$. We must show that $P(\epsilon) \Rightarrow P(f^{(i+1)}(\epsilon) + \beta)$. By the continuity of $f^{(i+1)}$, we can choose $\epsilon' > \epsilon$ such that $f^{(i+1)}(\epsilon') \leq f^{(i+1)}(\epsilon) + \beta/2$. But then

$$P(\epsilon) \Rightarrow P(f(\epsilon')) \Rightarrow P(f^{(i+1)}(\epsilon') + \beta/2) \Rightarrow P(f^{(i+1)}(\epsilon) + \beta),$$

where the first implication follows from (A) and (C), and the second implication is an instance of the induction hypothesis. This ends the proof of (D).

To see that (B) and (D) imply $P(\epsilon)$ for all $\epsilon \in \mathbb{R}_+$, simply observe that $P(2)$ is Ragde's result, and that $\lim_{i\to\infty} f^{(i)}(2) = 0$. All that remains is to verify (A)–(C).

**Lemma 3:** For all $\epsilon, \epsilon' \in \mathbb{R}_+$, $\epsilon < \epsilon' \Rightarrow f(\epsilon) < f(\epsilon')$.

**Proof:** It is straightforward to show that $f'(\epsilon) > 0$ for all $\epsilon \in \mathbb{R}_+$. ∎

**Lemma** 4: For all $\epsilon \in \mathbb{R}_+$, $f(\epsilon) < \epsilon$.

**Proof**: Equating the two expressions for $f(\epsilon)$, it is easy to see that

$$f(\epsilon) = \frac{4\epsilon}{2 - \epsilon + \sqrt{5\epsilon^2 + 12\epsilon + 4}}.$$

Moving $2 - \epsilon$ to the opposite side of the inequality and squaring both sides, it is also easy to verify that

$$2 - \epsilon + \sqrt{5\epsilon^2 + 12\epsilon + 4} > 4.$$

The claim follows. ∎

**Lemma** 5: For all $\epsilon, \beta \in \mathbb{R}_+$, $P(\epsilon) \Rightarrow P(f(\epsilon) + \beta)$.

**Proof**: Fix $\epsilon, \beta \in \mathbb{R}_+$ and assume that $P(\epsilon)$ holds. Take $\gamma = f(\epsilon) + \beta$ and $\alpha = 1/(2 + \epsilon)$. By continuity, we can assume that $\epsilon$ and $\gamma$ are rational. Suppose that we are given an input array of size at most $n$ containing at most $k$ items and observe that because Lemma 5 will be proved for all $\beta \in \mathbb{R}_+$, it suffices to show how to compact the items into an array of size at most $\phi(\epsilon, \beta)k^{2+\gamma}$, where $\phi : \mathbb{R}_+^2 \to \mathbb{R}_+$ is an arbitrary continuous function. We now describe an algorithm to accomplish this task.

Divide the input array into $O(\sqrt{n})$ subarrays of size at most $\sqrt{n}$ each. For some constant $\delta \in \mathbb{R}_+$ (i.e., $\delta$ may depend on $\epsilon$ and $\beta$, but not on $n$ and $k$), chosen below to make $1/\delta$ a multiple of 4, use the algorithm implied by the assertion $P(\epsilon)$ to attempt to compact each subarray into an array of size $\lfloor n^{i\delta} \rfloor$, for $i = 1, \ldots, t = 1/(2\delta) - 1$. Define the *level* of a subarray as 0 if even the compaction of the subarray for $i = 1$ succeeds, as $t$ if no compaction of the subarray succeeds, and otherwise as an arbitrary integer $i$ with $1 \leq i < t$ such that the compaction into $\lfloor n^{(i+1)\delta} \rfloor$ space succeeds, but the compaction into $\lfloor n^{i\delta} \rfloor$ space does not. For $i = 0, \ldots, t$, let $S_i$ be the set of those nonempty subarrays whose level is $i$. Our goal is to compact the items stored in subarrays in $S_i$ into $k^{2+\gamma}$ space, for $i = 0, \ldots, t$. Since $t$ depends only on $\delta$, this solves the given problem.

Fix $i \in \{0, \ldots, t\}$. By the correctness of the algorithm implied by $P(\epsilon)$, each subarray in $S_i$ contains more than $n^{i\delta\alpha}$ items. Hence $|S_i| \leq kn^{-i\delta\alpha}$. Using $O(n)$ processors, we can therefore compact $S_i$ into an array of at most $(kn^{-i\delta\alpha})^2$ blocks of $\lfloor n^{(i+1)\delta} \rfloor$ cells each, a total of at most $k^2 n^{\delta(i+1-2i\alpha)}$ cells. Since $\alpha < 1/2$ and $i\delta < 1/2$,

$$\delta(i + 1 - 2i\alpha) = i\delta(1 - 2\alpha) + \delta \leq 1/2 - \alpha + \delta.$$

The items in $S_i$ are therefore located in an array of size at most $M = k^2 n^{1/2 - \alpha + \delta}$. If $M \leq k^{2+\gamma}$, we are done. Hence assume that this is not the case, i.e., $n^{1/2-\alpha+\delta} > k^\gamma$. Then $M < n^\zeta$, where

$$\zeta = \left(\frac{1}{2} - \alpha + \delta\right)\left(1 + \frac{2}{\gamma}\right) = \frac{1}{2} + \frac{1 - 2\alpha}{\gamma} - \alpha + \delta\left(1 + \frac{2}{\gamma}\right).$$

Although it is not obvious from the above bound, we can clearly ensure that the array containing the items in $S_i$ is of size $O(n)$. Hence use Lemma 2 to move the elements in $S_i$ to an array of size at most

$$N = k^2 n^{(1-2\alpha)/\gamma - \alpha + \delta(1+2/\gamma)}.$$

As above, if $N \leq k^{2+\gamma}$, we are done. If not, $N < n^\theta$, where

$$\theta = \left(\frac{1-2\alpha}{\gamma} - \alpha + \delta\left(1+\frac{2}{\gamma}\right)\right)\left(1+\frac{2}{\gamma}\right).$$

We will show that

$$\left(\frac{1-2\alpha}{f(\epsilon)} - \alpha\right)\left(1+\frac{2}{f(\epsilon)}\right) = \frac{1}{2}. \qquad (*)$$

Since $\alpha < 1/2$ and $\gamma > f(\epsilon)$, it then follows that for sufficiently small values of $\delta$, the exponent $\theta$ will also be bounded by $1/2$. But then the compaction can be completed by one application of Lemma 2 with $r = 1$.

We still have to show $(*)$. Since $1 - 2\alpha = \epsilon/(2+\epsilon)$, we can rewrite the relation as

$$\left(\frac{\epsilon}{(2+\epsilon)f(\epsilon)} - \frac{1}{2+\epsilon}\right)\left(1+\frac{2}{f(\epsilon)}\right) = \frac{1}{2}$$

or as

$$2(\epsilon - f(\epsilon))(2+f(\epsilon)) = (2+\epsilon)f(\epsilon)^2,$$

which a last rewriting turns into

$$(4+\epsilon)f(\epsilon)^2 + (4-2\epsilon)f(\epsilon) - 4\epsilon = 0.$$

Finally observe that $f(\epsilon)$ is indeed the positive solution to the quadratic equation

$$(4+\epsilon)x^2 + (4-2\epsilon)x - 4\epsilon = 0. \quad \blacksquare$$

**Theorem:** For every fixed $\epsilon > 0$, $(n, k, k^{2+\epsilon})$-compaction problems can be solved in constant time on an ARBITRARY CRCW PRAM with $O(n)$ processors. $\blacksquare$

## References


[1] H. BAST AND T. HAGERUP, Fast and Reliable Parallel Hashing, *in* Proc. 3rd Annual ACM Symposium on Parallel Algorithms and Architectures (1991), pp. 50–61.

[2] P. BEAME AND J. HASTAD, Optimal Bounds for Decision Problems on the CRCW PRAM, *J. ACM* **36** (1989), pp. 643–670.

[3] R. COLE AND U. VISHKIN, Faster Optimal Parallel Prefix Sums and List Ranking, *Inform. and Comput.* **81** (1989), pp. 334–352.

[4] D. EPPSTEIN AND Z. GALIL, Parallel Algorithmic Techniques for Combinatorial Computation, *Ann. Rev. Comput. Sci.* **3** (1988), pp. 233–283.

[5] M. L. FREDMAN, J. KOMLÓS, AND E. SZEMERÉDI, Storing a Sparse Table with $O(1)$ Worst Case Access Time, *J. ACM* **31** (1984), pp. 538–544.

[6] J. GIL, Y. MATIAS, AND U. VISHKIN, Towards a Theory of Nearly Constant Time Parallel Algorithms, *in* Proc. 32nd Annual Symposium on Foundations of Computer Science (1991), pp. 698–710.

[7] M. T. GOODRICH, Using Approximation Algorithms to Design Parallel Algorithms that May Ignore Processor Allocation, *in* Proc. 32nd Annual Symposium on Foundations of Computer Science (1991), pp. 711–722.

[8] T. HAGERUP, Fast Parallel Space Allocation, Estimation and Integer Sorting, Tech. Rep. no. MPI–I–91–106 (1991), Max-Planck-Institut für Informatik, Saarbrücken.

[9] T. HAGERUP, Fast Parallel Generation of Random Permutations, *in* Proc. 18th International Colloquium on Automata, Languages and Programming (1991), Springer Lecture Notes in Computer Science, Vol. 510, pp. 405–416.

[10] Y. MATIAS AND U. VISHKIN, Converting High Probability into Nearly-Constant Time — with Applications to Parallel Hashing, *in* Proc. 23rd Annual ACM Symposium on Theory of Computing (1991), pp. 307–316.

[11] P. RAGDE, The Parallel Simplicity of Compaction and Chaining, *in* Proc. 17th International Colloquium on Automata, Languages and Programming (1990), Springer Lecture Notes in Computer Science, Vol. 443, pp. 744–751.

[12] R. RAMAN, The Power of Collision: Randomized Parallel Algorithms for Chaining and Integer Sorting, *in* Proc. 10th Conference on Foundations of Software Technology and Theoretical Computer Science (1990), Springer Lecture Notes in Computer Science, Vol. 472, pp. 161–175.

[6] J. GIL, Y. MATIAS, AND U. VISHKIN, Towards a Theory of Nearly Constant Time Parallel Algorithms, *in* Proc. 32nd Annual Symposium on Foundations of Computer Science (1991), pp. 698–710.

[7] M. T. GOODRICH, Using Approximation Algorithms to Design Parallel Algorithms that May Ignore Processor Allocation, *in* Proc. 32nd Annual Symposium on Foundations of Computer Science (1991), pp. 711–722.

[8] T. HAGERUP, Fast Parallel Space Allocation, Estimation and Integer Sorting, Tech. Rep. no. MPI–I–91–106 (1991), Max-Planck-Institut für Informatik, Saarbrücken.

[9] T. HAGERUP, Fast Parallel Generation of Random Permutations, *in* Proc. 18th International Colloquium on Automata, Languages and Programming (1991), Springer Lecture Notes in Computer Science, Vol. 510, pp. 405–416.

[10] Y. MATIAS AND U. VISHKIN, Converting High Probability into Nearly-Constant Time — with Applications to Parallel Hashing, *in* Proc. 23rd Annual ACM Symposium on Theory of Computing (1991), pp. 307–316.

[11] P. RAGDE, The Parallel Simplicity of Compaction and Chaining, *in* Proc. 17th International Colloquium on Automata, Languages and Programming (1990), Springer Lecture Notes in Computer Science, Vol. 443, pp. 744–751.

[12] R. RAMAN, The Power of Collision: Randomized Parallel Algorithms for Chaining and Integer Sorting, *in* Proc. 10th Conference on Foundations of Software Technology and Theoretical Computer Science (1990), Springer Lecture Notes in Computer Science, Vol. 472, pp. 161–175.