Self-organization and Annealing

by

Dana Richards

Computer Science Report #TR85-23

November 1985

Submitted for publication to Algorithmica.

Self-organization and Annealing

Dana Richards

University of Virginia

Department of Computer Science
Thornton Hall
University of Virginia
Charlottesville, Virginia 22903

ABSTRACT

We investigate the use of self-organization techniques in conjunction with simulated annealing to reorganize the records of a list to find a static ordering that minimizes expected search time. Each search begins where the previous search ended. The work began with the simpler and related problem of annealed self-organized linear lists which is of interest in itself. We review the known algorithms and propose new time varying rules that give better performance and are probabilistically optimal in the limit. The most surprising result is that utility of annealing is greatly reduced in the short run when using the new self-organization rules. The results were superior to more complicated techniques.

Keywords: Self-organizing data structures, searching, simulated annealing, disk organization, linear lists, fingers.

Self-organization and Annealing

Dana Richards

University of Virginia

1. Introduction

A self-organizing data structure dynamically rearranges its records so that the expected access time is minimized. In the literature the linear list and the binary search tree are the most studied (e.g. [BITN79, RIVE76]). With a linear list each search proceeds sequentially from the first item and with trees all accesses begin at the root. In all cases an accessed record is moved "up", i.e. towards the beginning of the list or closer to the root. We reserve the term *self-organizing* for schemes which only use $O(1)$ extra storage; with enough extra storage truly optimal reorganization would be possible.

A linear list with one *finger*, an *f-list*, is a list of records with a global pointer, the finger, to one of the records. To access an item the search is sequential from the current position of the finger. This is sometimes known as catenated search [KNUT73]. The search is not a search for a key. (Otherwise, with an unsorted list, we would not know which direction, i.e. to the left or the right, to begin our search.) Instead we assume we know our destination and use a sequential search due to the nature of the data structure. This formulation was motivated by the work of Carson and Reynolds [CARS85] on disk reorganization. There are two obvious advantages of f-lists over linear lists: the bidirectional reduces access time and, in many cases, it is better to conduct subsequent accesses in the vicinity of previous ones.

We are concerned with the problem of reducing access times for searchs on an f-list. Clearly by reorganizing the records we could reduce the expected seek time.

There are two important differences from the list and tree problems. First, we assume that actually moving records is quite expensive and disruptive. Hence we would do our reorganization on a "virtual" f-list and after a period of time reorganize the entire f-list. In the previous work on lists and trees *expected access time* is a dynamic quantity, so the formulas do not necessarily hold for any particular snapshot of the data structure. We need a scheme which after a period of time will produce a good static arrangement. We have investigated using "annealing" techniques.

The second difference concerns the effect of dependent access probabilities. For lists and trees the importance of the fact that one record is likely to follow another record is greatly minimized since all accesses begin at the same location. (That is not to say dependent probabilities can be ignored [KONN81, LAM84].) Since a search on an f-list begins at the previous record's position the dependent probabilities would cause clustering in an optimal record arrangement. If the access probabilities are known to be independent then a simple symmetric "organ-pipe" arrangement is known to be optimal [WONG83]. However the general case is an NP-complete optimization problem [VAQU80]. (The reduction is from Optimal Linear Arrangement [GARE79].)

Simulated annealing is a technique for solving hard optimization problems that has recently found many applications [KIRK83]. Annealing is actually a framework for which detailed guidelines are still emerging. Each new application still involves much guesswork on how to set parameters, choose cost functions, update solutions, etc. The basic idea is that a trial solution is generated from a previous trial solution and the two are compared by a cost function. If the new solution has lower cost it is accepted, otherwise it is accepted probabilistically according to the current "temperature" (usually with respect to the Boltzmann distribution.) As this process iterates the temperature decreases leading to a "frozen" locally optimal solution.

Our application of simulated annealing is somewhat different from those in the literature. Since we allow only $O(1)$ extra storage it is impossible to have a global cost function that depends on the arrangement of more than $O(1)$ records. Given that we must make a blind choice we considered two alternatives. First we could have each solution cost no more than the previous solution, that is we always accept it. Second we could assume the new solution costs more by an amount which is a function of the most recent access. In particular, the reorganization implied by accessing a record is accepted probabilistically as a function of the access time and temperature. The first alternative is a degenerate case of the second as we will formulate it.

We began our investigation by looking at the simpler (and presumably related) problem of linear lists without a finger. We used several self-organization techniques with several particular annealing procedures to produce final static orderings of a set of records. Given the (independent) access probabilities it is well-known that the optimal static ordering is in decreasing order of access probability. Since these were known to us we could easily evaluate each static ordering produced by our algorithm by computing the expected access time and calculating the percentage over the optimal ordering.

We have performed hundreds of trials to search for the best self-organization technique, the best annealing schedule, and the best cost function. (We feel that there are many more schedules and functions to be tried.) We did half the trials with a linear distribution and the rest with Zipf's distribution. The latter better models the (stationary) access probabilities for disks and file accesses. The most surprising aspect of our work is the robustness of the naive method which has an identically 0 cost function, that is unaffected self-organization. Further, it rarely made even a negligible improvement to vary the cost function though often it would produce much worse solutions, presumably due to the effect of slow convergence rates. The Zipf distributed data produced similar results to the linear data, only

with even slower convergence.

After the experiments with linear lists we did many trials with f-lists. Instead of using random data we used actual access sequences from normally operating disks. Here it is much harder to evaluate our results since we cannot determine the optimal arrangement. We measured, with an actual sequence of accesses, the average search time of the initial ordering and the final static ordering. The self-organization techniques were the same as before only we moved toward the previous record instead of the front. Again we found the naive algorithm to be incredibly robust.

2. Self-Organizing Linear Lists

In this section we review what is known about self-organizing linear lists. We restrict our attention in this paper to rules that move an accessed record towards the beginning of the list and leaves the remaining records in the same relative order. Let p_i be the independent probability of accessing R_i, the ith record. Let C_A be the cost or average access time in the limit if the rule A is used for self-organization. OPT is the trivial rule that maintains the list in static order by decreasing access probabilities. OPT is used for comparison since it is the optimal static ordering [MCCA65]. Let there be n records, $R_0, \ldots, R_{n-1}$, and m accesses, where $m \to \infty$ unless otherwise stated.

The best known techniques are *transpose* (TR), moving one step towards the front, and *move-to-front* (MTF), moving all the way to the front. It is known that

$$C_{MTF} = 1 + 2 \sum_{i<j} \frac{p_i p_j}{p_i + p_j}$$

and that $C_{MTF} \leqslant \frac{\pi}{2} C_{OPT}$ [CHUN85]. A complicated formula for the probability that R_i is in position j is known [BURV73]. The most important result about TR is that $C_{TR} \leqslant C_{MTF}$ [HEND76, RIVE76]. Let $b(i,j)$ be the probability in the limit that that R_i is before R_j. The reason analytic results are easy to obtain for MTF but not for TR (or for f-lists as well) is that in the latter case $b(i,j)$ is hard to calculate directly. MTF has been analyzed for several distributions [GONN81] including Zipf's distribution for which $C_{MTF} \leqslant 2\ln 2\, C_{OPT} = 1.38 \cdots C_{OPT}$.

While TR is analytically better than MTF, MTF is often preferable in the short run due to its rapid convergence (e.g. see [TENE78, YU79] for empirical evidence). From a combination of analytic and empirical results a rule of thumb is to prefer TR over MTF if the number of accesses is $\Omega(n^2)$ [BITN79]. There are two approaches to the problem of the relationship of convergence to the number of accesses. First a hybrid rule [BITN79] which switches from MTF to TR after an

appropriate number of accesses has been proposed. Second a "spectrum" of rules, with TR and MTF at the extremes, can be selected from depending on the number of accesses. In section 4 we smoothly combine these approaches.

The first spectrum proposed was MF_k, "move forward k" positions or to the front whichever is closer [RIVE76]. We have $C_{MF_{k-1}} \leqslant C_{MF_k}$ which implies $C_{TR} \leqslant C_{MTF}$ [GONN81]. But empirical results show that as n/m increases then k should increase as well, as expected due to the convergence rates [TENE78]. Other spectra divide the list of records at some point and use different rules, MF_{k_1} and MF_{k_2} for records before and after the dividing point (which acts like a "front" for the second half) respectively [TENE78, TENE82]. All analytic and empirical evidence indicates that we should have $k_1 \leqslant k_2$ and that the more "TR-like" a rule is the better the asymptotic cost. Of course this generalizes to many division points [TENE78] leading to the following elegant rule [SLEA85a, TAKA77]. MPF_k, "move proportionately forward", is the rule that moves the record in position *pos* forward $\lfloor pos/k \rfloor$ positions. Intuitively the advantage of this rule is that it allows records near the end of the list to move rapidly but those near the front move more slowly leading to the stability that makes TR asymptotically better. MPF_k is the basic rule we used in our experiments.

In this paper we deal with time varying rules where it becomes increasingly unlikely a move will be acted on after an access. In a similar vein the "k-in-a-row" rule was proposed [GONN81, KAN80] where a record is moved, by whatever rule, only if accessed k consecutive times. As k increases the rule becomes "more static". Let TR_k and MTF_k stand for the corresponding rules. It is known that $C_{TR_k} < C_{MTF_k}$ and $C_{TR_k} < C_{TR_{k-1}}$. Let the records be numbered so that $p_0 > p_1 > \cdots > p_{n-1}$, assuming distinct values for now. A rule is *progressively reluctant*, for parameter k, if

$$p_i^{(k)} / \sum_{j=i}^{n} p_j^{(k)} \to 1 \quad as \quad k \to \infty, \; 1 \leqslant i \leqslant n,$$

where $p_i^{(k)}$ is the probability that R_i will be moved. For the class of rules in this paper every k-in-a-row rule is progressively reluctant and it can be shown that as $k \to \infty$ the probability that R_i is in position i goes to 1, i.e. the same as OPT [KAN80]. A more reluctant variant of k-in-a-row is the "k-in-a-batch" rule which requires that the k-in-a-row occurs in predetermined intervals [GONN81]. Let MTF_k' be the corresponding rule. It is known that $C_{\text{MTF}_k'} \leqslant (\frac{\pi}{2k} \, csc \, \frac{\pi}{2k}) C_{\text{OPT}}$ and that for $k=2$ and 3 the constant is $1.110\cdots$ and $1.04\cdots$ respectively [CHUN85].

There are more direct ways to process fewer accesses. For independent access probabilities it does not matter if accesses are only periodically used for reorganization [MCCA65]. In fact if a rule is ignored with probability α, α fixed, the asymptotic results are unaffected [KAN80]. However consider the case of ignoring an access at position i with probability $\alpha_i > 0$. The analyses of TR are unaltered since the underlying Markov process is time-reversible [KAN80]. We conjecture that the expected access time is never worse for any rule in this paper if $\alpha_0 < \cdots < \alpha_{n-1}$.

When we assume the access probabilities are dependent it is typical to assume the accesses are a Markov process themselves, i.e. p_{ij}, the probability of accessing R_j after R_i, is independent of previous accesses. Let m_{ij} be the *mean first passage time* from R_i to R_j. Recall for the independent case $m_{ij} = 1/p_j$. It is known [LAM84] that

$$C_{\text{MTF}} = 1 + 2 \sum_{i<j} \frac{1}{m_{ij} + m_{ji}}$$

and

$$C_{\text{MTF}} \leqslant (1 - 2a) + 2aC_{\text{OPT}}$$

where $a = 1 - \min_i \{p_{ii}\}$. Note that we can have $C_{\text{MTF}} < C_{\text{OPT}}$. Further there are distributions for which $C_{\text{MTF}} < C_{\text{TR}} < C_{\text{OPT}}$. (There are distributions for which TR and MTF perform worse than a random organization, as well.) The intuition is that

in highly "coupled" distributions MTF should perform well since it rapidly brings together a cluster of records that TR would be slow to respond to and that OPT, being static, would not respond to at all. Further, several authors have noted that if the pattern of access is itself time varying then OPT could be worse than a dynamic rule, even for the independent case.

To complete this survey of results we should mention that for the class of rules in this paper it has been long conjectured that TR is the optimal rule for independent accesses, if indeed there is a unique optimal rule. There has been much work on this [GONN81, KAN80, RIVE76]. However it is known that if when accessing position i an arbitrary permutation, depending on i, can be applied to the list of records, then TR is not always optimal [ANDE82]. If we generalize the class further to allow arbitrary extra storage for frequency counts then rules with costs equal to C_{OPT} are trivial, e.g. a list kept in decreasing frquency order (see also [BITN79, LAM81]). If we are concerned with the *amortized cost*, the sum of the access times for m accesses, then it is known that MTF performs very well in the worst case and TR does not [BENT85, SLEA85a]. However these considerations are mainly insurance against wildly improbable access sequences.

Oommen has done some preliminary work on using self-organization to produce static list orderings [OOMM84]. However his scheme uses $O(n)$ extra storage for frequency counters which are used with a probabilistic MTF rule. Not only is the rule complicated, e.g. using periodic sorting steps, but is inferior to the trivial rule mentioned above. He mentions, in a final note, that a probabilistic "move-to-rear" rule is far simpler, but still it uses $O(n)$ extra storage.

Kapoor and Reingold have given a rule which in some sense converges to a static arrangement [KAPO85]. They use MTF but ignore the rule with probability $1 - 1/f(t)$ where $f(t)$ is an increasing unbounded function of the time t. As $t \to \infty$ the rule becomes static. The motivation for their rule is that since MTF

converges rapidly it is wasteful to continue updating considering that the MTF updates are expensive. However we will see that it not desirable to converge to the performance MTF gives. In their conclusion section, independently of the work in this paper, they raise the possiblity of a rule that varies smoothly from MTF to TR.

Sleator and Tarjan also mention a rule for producing a good static ordering [SLEA85b]. They state that a rule which has amortized cost within a constant factor of optimal will, if stopped randomly, produce a static arrangement which has expected cost within a constant factor of optimal. (They only explicitly show this for a tree algorithm but they apply it to MTF.) Their stopping rule chooses m first and halts after any access with probability $1/m$. They only deal with independent access probabilities. Recall that TR does not have good amortized performance but our experiments show that it produces better static orderings when stopped arbitrarily (though not "randomly" as above).

3. Simulated Annealing Schedules

In this section we present our annealing algorithm which shares many details with other such algorithms in the literature but is tailored to our application. The procedure begins with a temperature t which drops over time. It does not drop "continuously" but in discrete steps; at each step a certain amount of activity must occur.

The algorithm is in figure 1. Each access is considered for action by some reorganization rule *moverule*. We use $\Delta(access, t)$ to evaluate the change in "cost" (not necessarily the average access time) between the current configuration of the list and the list resulting from $moverule(access, t)$. If $\Delta < 0$ we accept the move, otherwise we accept probabilistically according to the Boltzmann distribution. Clearly as $t \to 0$ the list "freezes" since no moves will be permitted. However to anticipate this and save computation time we use the counters *att* and *succ*. If far more attempts than successes occur in some consecutive time period then the list is judged to be frozen. One stage of the algorithm tries much reorganization at a fixed temperature. Depending on the number of successes the stage is judged as being a "failure". After each stage the temperature is decremented by the factor δ, $\delta < 1$. The temperature must remain above an arbitrary *limit* to save computation time.

Filling in the details of this algorithm was done mostly by trial and error. We had $fbound = 3$, $abound = 50n$, $sbound = 10n$, $t_0 = 100$, and $limit = t_0/100$. Typically $\delta = .9$ but varying δ was useful for determining convergence rates.

Actually we never have $\Delta < 0$ so that test is vacuous. It was included to emphasize the difference of our algorithm from previous schemes. No attempt was made at evaluating the "cost" of the lists, before and after a move, due to the $O(1)$ extra storage restriction. Instead we assumed that $\Delta \geq 0$ and it grew monotonically with the search time for the accessed item. This heuristic assumes that some degree of organization is present and outlying records are less important. This

```
t ← t_0
frozen ← false
failures ← 0
while not frozen and t > limit do
        att ← succ ← 0
        while att < abound and succ < sbound do
                get(access)
                Δ ← Δ(access, t)
                if Δ < 0 or random < e^(−Δ/t) then
                        moverule(access, t)
                        succ ← succ + 1
                endif
                att ← att + 1
        endwhile
        if att ⩾ abound then
                failures ← failures + 1
        else
                failures ← 0
        endif
        frozen ← failures = fbound
        t ← δ t
endwhile
```

Annealing algorithm

Figure 1

assumption is clearly time varying, hence the second parameter to Δ.

4. Annealing Linear Lists

Our investigation began with linear lists before considering f-lists. No attempt was made at an exhaustive multivariate analysis; instead we sought to identify those parameters and other details that gave improved performance. Our statistics are the average of only 5 runs. The effect of the variance was evident but did not affect the relative rankings of the schemes. No attempt was made at determining higher moments. Our measure of performance was the percentage of C_{OPT}.

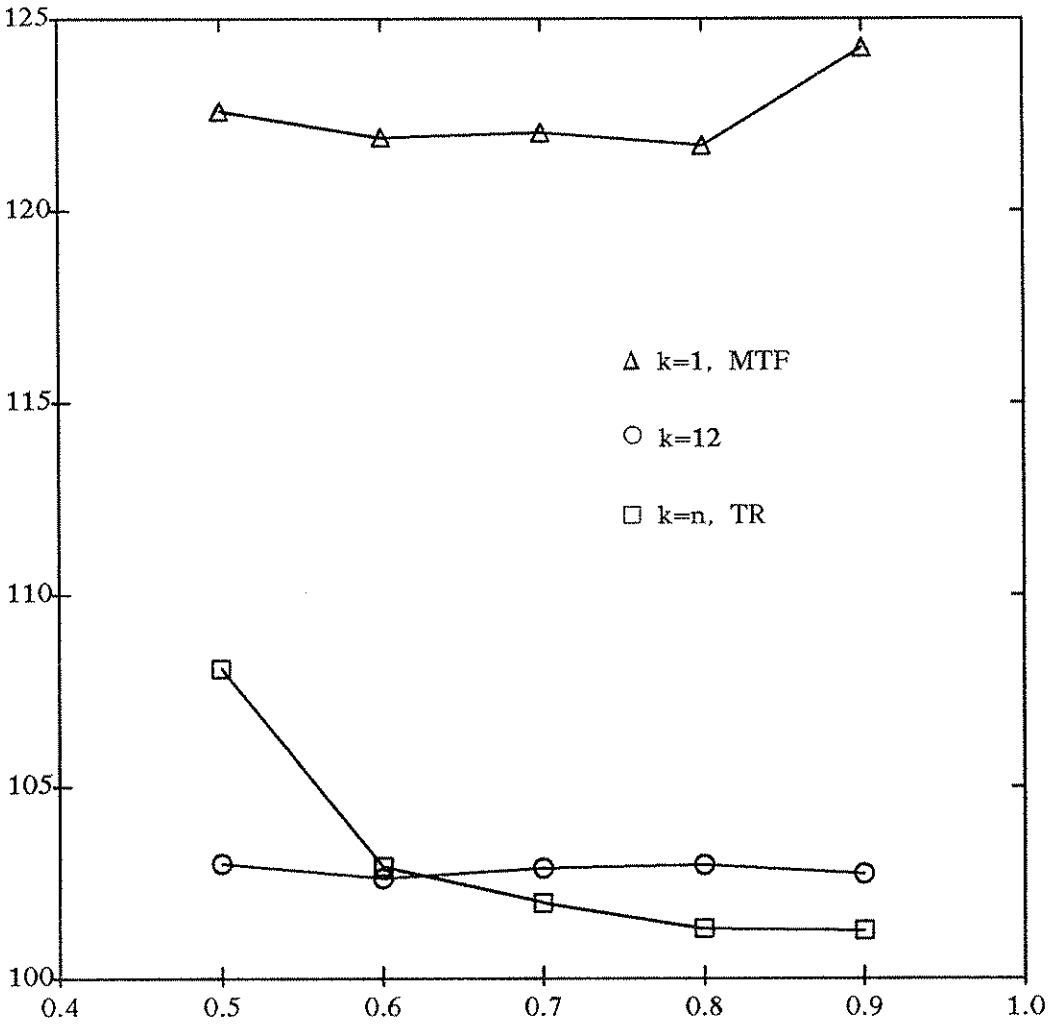
We used random access sequences for two different distributions, with $n=100$ throughout. We generated an access randomly with respect to "frequency" counts. The *linear* distribution used random frequencies from 0 to 9999. The *Zipf* distribution used frequencies equal to $\lfloor 1000/i \rfloor$, $1 \leqslant i \leqslant 100$, which were then randomly permuted. (This closely approximates the actual Zipf's distribution, $p_i = 1/iH_n$.) It is easy to predict that the linear data would be less sensitive than the Zipf data. This is because the latter contains relatively few large values, all of which must be very close to the front of the list.

Our rule, *moverule*, was to move forward a distance $\lfloor pos/k \rfloor$ when accessing a record in position *pos*, i.e. MPF_k. With $k = 1$ and n we have MTF and TR respectively. But, as discussed in section 2, it is better to have a time varying rule. Recall that the temperature can vary from t_0 to *limit* and at the ith stage $t = t_0 \delta^i$. Let

$$\sigma(t) = 1 - \frac{t - limit}{t_0}.$$

Note that $\sigma(t)$ varies from $limit/t_0$ to 1 as a function of the temperature. We chose $k = k(t) = \lfloor \sigma(t) n \rfloor$. If $limit \leqslant t_0/n$ then this rule varies smoothly from MTF to TR, with the majority of the time it being "TR-like".

Let us begin by having $\Delta = 0$, i.e. all accesses are acted on. We effectively increase the length of the experiment by increasing δ. We first considered fixed k's, $k = 1$, 12, and 100, as shown in figure 2 for linear data. We see that for MTF,



δ vs. % of optimal

Linear data with $\Delta = 0$

Figure 2

$k=1$, the performance is relatively insensitive to δ showing that it is near its steady state. Further, while its performance is well within its theoretical limits it is far worse than TR, $k=n$, indicating that the convergence "crossover" of the two methods is long past. Such a crossover is seen for MPF_{12} and TR. Similar results were obtained for the Zipf data. For reference, the number of accesses for $\delta = .5$, .6, .7, .8, and .9 were 7000, 10000, 13000, 21000, and 44000 respectively.

We compared the time varying $MPF_{k(t)}$ with TR, $k=n$, again with $\Delta = 0$. The results are shown in figure 3 for the Zipf data. The results indicate a clear cut improvement using $k(t)$. Recall that the same records are being moved in both cases, only the distance moved is different. A similar improvement was found for the linear data.

The next step is to introduce $\Delta \neq 0$ so that annealing occurs. Recall an access to a record *access* in position *pos* has been made, so we let $\Delta(access, t) = \Delta(pos, t)$. We tried several functions, notably

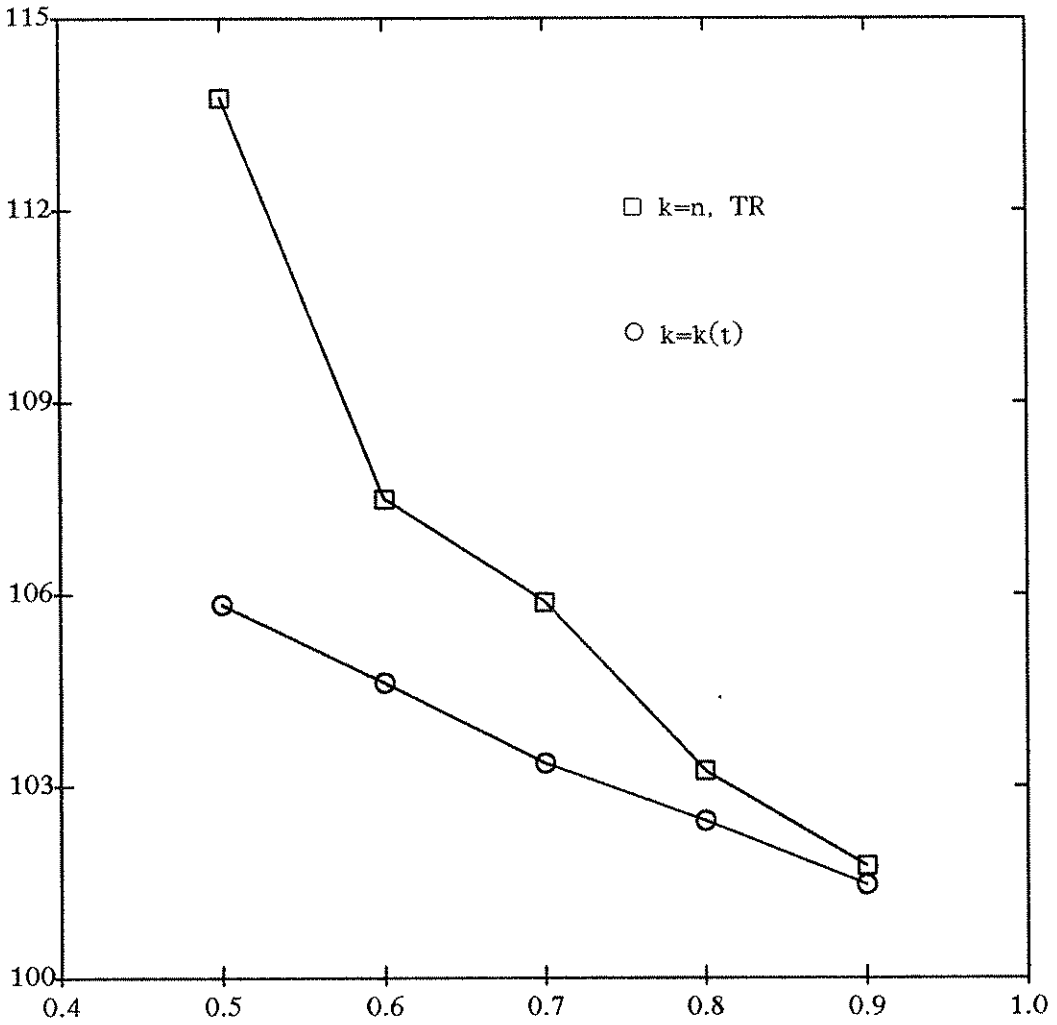
$$\Delta_1(pos, t) = a_1\, pos + a_2$$

and

$$\Delta_2(pos, t) = b_1((t_0/t)^{b_2\, pos} - 1)$$

where a_1, a_2, b_1, and b_2 are constants which may depend on n. The motivation for Δ_1 is simply that, in steady state, elements further from the front are less likely to need to go towards the front. It has the disadvantage of being independent of t and hence is applied before the steady state and, in fact, should delay convergence.

There are simpler functions that depend on *pos* and t than Δ_2. For example $c_1\, pos/t + c_2$. However Δ_2 is the simplest intuitively motivated function that gives rise to a rule that is *weakly* progressively reluctant. This means that as $t \to 0$ the record that will be moved will be in the first position with probability 1, and that among the remaining records the record that will be moved will be in the second position with probability 1, and so on. We conjecture that for the rules in this



δ vs. % of optimal

Zipf data with $\Delta = 0$

Figure 3

paper Δ_2 is in fact progressively reluctant in the stronger sense of section 2. (This a stronger conjecture than our conjecture about the performance with $\alpha_0 < \cdots < \alpha_{n-1}$.) Recall this would ensure that the scheme is probabilistically optimal as $t \to 0$. Note that Δ_2 is initially 0 and grows with both t and *pos*. Hence in the beginning most accesses are acted on which should speed convergence. It is important that Δ not grow so fast that the annealing schedule "freezes" before enough accesses are acted on. This indicates an important implicit relationship between Δ, *abound*, and *sbound*. We chose our constants so that the schedule never froze.

The most surprising result of our experiments is that when annealing was used with TR and $\text{MPF}_{k(t)}$ no improvement was found using Δ_1 or Δ_2. In fact when Δ grew rapidly the performance could be much worse. This is counterintuitive. It seems that if the same number of accesses are acted on then increasing the number of failures should not hurt, especially when the latter accesses were thought to be poor candidates for moves. (In the limit using Δ_2 should give better results; however it is very slow since we used up to 44000 accesses for 100 records!)

The resolution of this paradox is twofold. In the early stages denying any moves slows convergence. The second reason is that TR and $\text{MPF}_{k(t)}$ are very good to start with. In particular in the later stages candidates for moves will only move a short distance so it is not appropriate to "protect" against moves from the rear of the list. Such moves tend to fine-tune the arrangement.

In contrast, annealing *is* helpful for MTF. The primary reason MTF performs so poorly is that in the later stages it does not protect itself against ill-advised moves. For the linear data, with $\delta = .8, b_1 = .01, b_2 = 5/n$, we found a reduction in the percentage of optimal from 121.70 to 106.65 using Δ_2. Similarly for the Zipf data, with $\delta = .8, b_1 = 3.0, b_2 = 5/n$, we found a reduction from 140.70 to 107.99. However Δ_1 did not do as well. For example, for the latter case with

$\delta = .8, a_1 = .6, a_2 = 0$, we found a reduction from 140.70 to only 112.68. (The constants were chosen above to give the best observed performances and to involve the same number of accesses.)

We noted that the amount of processing for $\delta = .9$ was double that for $\delta = .8$ with little improvement. This, of course, is a function of the small n; as n increases δ must as well to get comparable results.

5. Annealing f-Lists

We now turn to the original problem of reorganizing the records of an f-list. Due to the expense of these experiments, which used a larger n, we relied on the nearly one thousand trials we did for linear lists to dictate what we tried for f-lists. In general our results were quite favorable, supporting the hypothesis that effective rules for the two problems should be similar.

We did not use random data since real data for an f-list was available (a disk in this case) [CARS85]. We did our experiments with a file that recorded 94834 consecutive seeks on a disk with $n = 842$ cylinders. We used the file to generate accesses, rewinding as often as necessary. With $\delta = .8$ and .9 we used, at least, 176820 and 370480 accesses respectively. The average access time was computed before and after reorganization. The initial average was, perhaps, overly large due to the placement of highly used directory records in the low numbered cylinders, as is common. Our measure of performance was the percent of improvement.

The same program was used with the following small modifications. The position *pos* of an access, used in the formulas of section 4, was the absolute distance from the previous access and *moverule* always moved a record towards the previous record as if it were the "front".

We began again with $\Delta = 0$ and varied k. For $\delta = .8$ and $k=1$ (MTF), 100, and 842 (TR) the improvement was 49.7%, 32.6%, and 47.2% respectively. The surprising result is that TR was not best after so many accesses. We feel this is due to the large number of infrequently accessed records and the large distances they must travel. With $k=100$ records move faster but it is still somewhat "TR-like" for nearby records where stability matters. Note that these findings are not in agreement with the list case as seen in figure 2, where $k=12$ is approximately in the same ratio to n as $k=100$ is here. This probably indicates a crossover has yet to be reached.

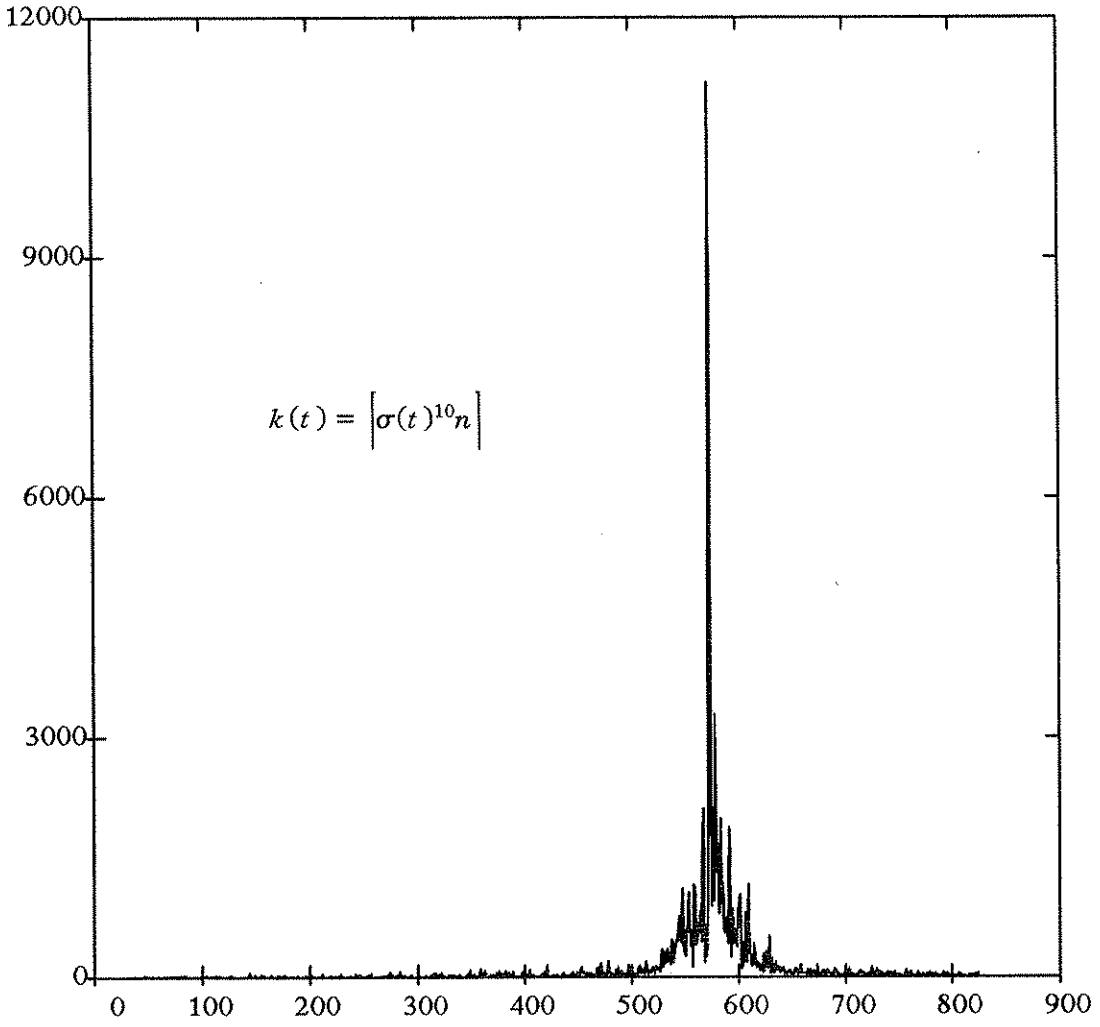
When the time varying $k = k(t)$ was tried, with $\delta = .8$, we had a 34.7% reduction. While this was much better than TR it was still a bit worse than with $k=100$. From our experience above we could predict further improvement by having our rule become "TR-like" more slowly. This motivated a *moverule* that moves a record $\lfloor pos/k(t) \rfloor$ positions, where

$$k(t) = \left\lfloor \sigma(t)^{\epsilon} n \right\rfloor, \qquad \epsilon \geq 1.$$

As ϵ increases $k(t)$ becomes more slow growing. We found for the above trial with $\epsilon=3$ we had a 31.5% improvement, narrowly beating $k=100$.

Judging from the lack of success with $\Delta>0$ for linear lists we were skeptical of their utility for f-lists. Again, in all cases for Δ_1 and Δ_2 we had a degradation of performance. The degradation increased when, by adjusting the constants, the algorithm became more selective. This supports the analysis in the previous section that argued annealing would not improve on $\mathrm{MPF}_{k(t)}$ in the short run. No attempt was made to gauge the improvement over unannealed MTF for f-lists.

Finally we tried $\delta = .9$. For $k=100$ the improvement was 26.3%. (Note that for the larger n in this section this increase in δ is significant.) For $k = k(t)$ and $\epsilon=3$ we found a 28.2% improvement, but with $\epsilon=10$ we had 25.9%. This indicates that ϵ should be time varying as well; this was not explored. The resulting arrangment from the final experiment is seen in figure 4. While it is not possible to determine the average access time from such a figure, it does show the characteristic clustering that occurs when dependent probabilities are considered.



Cylinder position vs. Frequency of access

Disk access data with $\Delta = 0$

Figure 4

6. Conclusions

The primary finding is that for self-organizing lists and f-lists when annealing is restricted to use $O(1)$ extra storage the benefits are not as favorable as when good self-organization is used alone. A preliminary comparison of our results with some annealing results, using TR only, for the same disk data, using $O(n)$ and $O(n^2)$ extra storage [CARS85] indicates that our rules are superior in time and performance. This indicates that improving self-organization techniques is more profitable than annealing. More work needs to be done to explain why pure self-organization rules are so robust that, in the short run, annealing seems to interfere with their performance. Further work is planned that will provide a full statistical comparison of our algorithms and those in [CARS85] as well as some others in the literature. Extending the model to allow several independent (or dependent) fingers has not been explored. Other algorithms for the f-list problem have been recommended that simply use local improvement [CARS85, FLOR78, RAMA71].

An open problem is to apply the techniques of this paper to data structures that support key searchs, more sophisticated than linear search. Fredrickson has thoroughly investigated a complicated class of self-organizing "implicit" data structures that have asymptotic performance equal to trees [FRED83, FRED84] but did not consider static arrangements. The case of binary search trees [ALLE78, BITN79, GONN81] is somewhat anomalous since *move-to-root* is quite good but the analogue of TR is very bad. Can time-varying rules and annealing mediate this anomaly? Using a more complicated *splaying* rule, which combines move-to-root and path compression, a "snapshot" theorem for static arrangements has been derived that promises good performance [SLEA85b].

7. Acknowledgements

I would like to acknowledge Scott Carson and Paul Reynolds Jr. for introducing this problem to me and the use of their disk data.

8. References

[ALLE78] B. Allen and I. Munro, Self-organizing binary search trees, *J. ACM*, **25**, 1978, pp. 526-535.

[ANDE82] A. E. Anderson, P. Nash and R. R. Weber, A counterexample to a conjecture on optimal list ordering, *J. Appl. Prob.*, **19**, 1982, pp. 730-732.

[BENT85] J. L. Bentley and C. C. McGeoch, Amortized analyses of self-organizing sequential search techniques, *Comm. ACM*, **28**, 1985, pp. 404-411.

[BITN79] J. R. Bitner, Heuristics that dynamically organize data structures, *SIAM J. Computing*, **8**, 1979, pp. 82-110.

[BURV73] P. J. Burville and J. F. C. Kingman, On a model for storage and search, *J. Appl. Prob.*, **10**, 1973, pp. 697-701.

[CARS85] S. Carson and P. Reynolds Jr., Adaptive reorganization of disk cylinder structures, Computer Science Tech. Rep. 85-20, Dept. Computer Science, Univ. Virginia , 1985.

[CHUN85] F. R. K. Chung, D. J. Hajela and P. D. Seymour, Self-organizing sequential search and Hilbert's inequalities, *Proc. 17th ACM Symp. Theory Comp.*, 1985, pp. 217-223.

[FLOR78] A. Flory, J. Gunther and J. Kouloumdjian, Data base reorganization by clustering methods, *Information Systems*, **3**, 1978, pp. 59-62.

[FRED83] G. N. Fredrickson, Self-organizing heuristics, implicit data structures, and unsuccessful search (Extended Abstract), *Proc. 21st Annual Allerton Conf. on Communication, Control, and Computing*, 1983, pp. 637-646.

[FRED84] G. N. Fredrickson, Self-organizing heuristics for implicit data structures, *SIAM J. Computing*, **13**, 1984, pp. 277-291.

[GARE79] M. R. Garey and D. S. Johnson, *Computers and Intractability*, Freeman, 1979.

[GONN81] G. Gonnet, J. I. Munro and H. Suwanda, Exegesis of self-organizing linear search, *SIAM J. Computing*, **10**, 1981, pp. 613-637.

[HEND76] W. J. Hendricks, An account of self-organizing systems, *SIAM J. Computing*, **5**, 1976, pp. 715-723.

[KAN80] Y. C. Kan and S. M. Ross, Optimal list order under partial memory constraints, *J. Appl. Prob.*, **17**, 1980, pp. 1004-1015.

[KAPO85] S. Kapoor and E. M. Reingold, Stochastic rearrangment rules for self-organizing data structures, Tech. Rept., University of Illinois, 1985.

[KIRK83] S. Kirkpatrick, C. D. Gelatt and M. P. Vecchi, Optimization by simulated annealing, *Science*, **220**, 1983, pp. 671-680.

[KNUT73] D. E. Knuth, in *The Art of Computer Programming: Sorting and Searching*, vol. 3, Addison-Wesley, Reading, 1973.

[KONN81] L. K. Konneker and Y. L. Varol, A note on heuristics for dynamic organization of data structures, *Info. Proc. Let.*, **12**, 1981, pp. 213-216.

[LAM81] K. Lam, M. K. Siu and C. T. Yu, A generalized counter scheme, *Theoretical Comp. Sci.*, **16**, 1981, pp. 271-278.

[LAM84] K. Lam, M. Leung and M. Siu, Self-organizing files with dependent accesses, *J. Applied Prob.*, **21**, 1984, pp. 343-359.

[MCCA65] J. McCabe, On serial files with relocatable records, *Operations Research*, **13**, 1965, pp. 609-618.

[OOMM84] B. J. Oommen, On the use of smoothsort and stochastic move-to-front operations for optimal list reorganization, *Proc. 23rd Allerton Conf.*, 1984, pp. 243-251.

[RAMA71] C. V. Ramamoorthy and P. R. Blevins, Arranging frequency dependent data on sequential memories, *AFIPS Conf. Proc.: Spring Joint Comp. Conf.*, **38**, 1971, pp. 545-556.

[RIVE76] R. Rivest, On self-organizing sequential search heuristics, *Comm. ACM*, **19**, 1976, pp. 63-67.

[SLEA85a] D. D. Sleator and R. E. Tarjan, Amortized efficiency of list update and paging rules, *Comm. ACM*, **28**, 1985, pp. 202-208.

[SLEA85b] D. D. Sleator and R. E. Tarjan, Self-adjusting binary search trees, *J. ACM*, **32**, 1985, pp. 652-686.

[TAKA77] I. Takanami and M. Fujii, On heuristic construction of self-organizing files for sequential searches, *Trans. Inst. Electron. & Commun. Eng. Japan, Section E*, **E60**(2), 1977, pp. 112.

[TENE78] A. Tenenbaum, Simulations of dynamic sequential search algorithms, *Comm. ACM*, **21**, 1978, pp. 790-791.

[TENE82] A. M. Tenenbaum and R. M. Nemes, Two spectra of self-organizing sequential search algorithms, *SIAM J. Computing*, **11**, 1982, pp. 557-566.

[VAQU80] A. Vaquero and J. M. Troya, Placement of records on linear storage devices, in *Information Processing 80 (IFIP)*, S. Lavington (ed.), 1980, 331-336.

[WONG83] C. K. Wong, *Algorithmic Studies in Mass Storage Systems*, Computer Science Press, 1983.

[YU79] C. T. Yu, M. K. Siu, K. Lam and M. Ozsoyoglu, Performance analysis of three related assignment problems, in *Proc. ACM-Sigmod Intl. Conf. on Management of Data*, P. Bernstein (ed.), 1979, 82-91.