International Journal of Engineering, Mathematical and Physical Sciences
ISSN: 2517-9934
Vol:2, No:2, 2008

Block Sorting: A New Characterization and a New Heuristic

Swapnoneel Roy, Ashok Kumar Thakur, and Minhazur Rahman

***Abstract*—The Block Sorting problem is to sort a given permutation moving blocks. A block is defined as a substring of the given permutation, which is also a substring of the identity permutation. Block Sorting has been proved to be NP-Hard. Until now two different 2-Approximation algorithms have been presented for block sorting. These are the best known algorithms for Block Sorting till date. In this work we present a different characterization of Block Sorting in terms of a $transposition\ cycle\ graph$. Then we suggest a heuristic, which we show to exhibit a 2-approximation performance guarantee for most permutations.**

***Keywords*—Block Sorting, Optical Character Recognition, Genome Rearrangements, Sorting Primitives, Approximation Algorithms**

I. Introduction

The problem of block sorting has its applications in *optical character recognition* besides being a non-trivial variantion of sorting by transpositions. Certain text regions are referred to as *zones* in optical character recognition by marking them [6]. In the zoning procedure, the zones might not be read in the correct order by the optical scanner. Thus an efficient technique is needed here to bring the zones in order. This is where block sorting comes into play. Moving the the zones in their correct places corresponds to the block sorting problem. We are interested in finding out the minimum number of steps required to bring all the zones in the correct order.

II. Preliminaries

We define a $permutation$ π of length n to be a string of length n from the set of strings which can be formed from the set of integers 1,2,...,n. We impose an additional restriction that every element should occur only once in a permutation.

Definition 1 (Block)**.** *Let π be a permutation on n elements, written as a string $\pi_1\pi_2.....\pi_n$ A block is a maximal substring of π which is also a substring of the identity permutation id_n = 1 2 . . . n [1]. For example,in the permutation 8 2 5 6 3 9 1 4 7 on 9 elements, there are eight blocks, and 5 6 is the only block containing more than a single element.*

Definition 2 (Block Move)**.** *A block move is the operation of picking a block and placing it adjacent to its predecessor or successor block so that it merges with it to form a larger block. For instance, a block move of 5 in 6 2 3 4 1 5 will result in either 5 6 2 3 4 1 or 6 2 3 4 5 1 [1].*

Definition 3 (Block Sorting Problem)**.** *The block sorting problem is to find a shortest series of block moves which, when applied in succession, sorts a given permutation π. The length of this shortest series is denoted by $bs(\pi)$ and is called the block sorting distance of π.*

Thus the block sorting problem is essentially an optimization problem in which the number of block moves required to sort a permutation is minimized.
Now we mention the nature of input permutations for the block sorting problem. The input permutations must satisfy the following conditions:

1) The input sequence cannot contain two identical numbers. For instance, it cannot contain two 8's.
2) The input sequence does not contain any negative number. That is, the input sequence is unsigned.

Definition 4 (Kernel Permutation)**.** *A kernel permutation $ker(\pi)$ of a permutation π is obtained in the following manner: The blocks in π are replaced by their ranks in π. For example, if $\pi = 825639147$, $ker(\pi) = 72538146$.*

In a kernel permutation, all the blocks are of length 1. It has been shown in [1] that $bs(\pi) = bs(ker(\pi))$. That is, block sorting the kernel permutation will take the same number of steps as block sorting the original permutation.
A transposition moves any substring of π of length k to any other location in π. A block move can thus be regarded as a transposition in which $k = 1$. In this way, the block sorting problem is a non-trivial invariant of the

Swapnoneel Roy is with the India Software Lab IBM India Pvt. Ltd, email: swapnoneel.roy@in.ibm.com
Ashok Kumar Thakur is with the India Software Lab IBM India Pvt. Ltd, email: ashok.thakur@in.ibm.com
Minhazur Rahman is with the India Software Lab IBM India Pvt. Ltd, email: minhazur_r@in.ibm.com

sorting by transposition problem.

III. Lower bounds for the block sorting problem

Definition 5 (Block Sorting Distance). *We define the block sorting distance $bs(\pi)$ as the minimum or optimal number of steps taken to sort π to id_n.*

Theorem 6. *By performing a single block move it is easy to see that the maximum number of blocks that can be reduced is 3. The identity permutation contains only 1 block. Thus if the initial permutation contains k blocks, a trivial lower bound for this problem is $(k-1)/3$.*

The block sorting problem is a nontrivial variation of another well known sorting problem, $Sorting\ by\ Transpositions$, that arises in the study of genome rearrangement [2]. A transposition is the operation of picking up any substring of the given permutation and placing it elsewhere in the permutation. The $Sorting\ by\ Transpositions$ problem is to sort the given permutation with the minimum number of transpositions. A $block\ move$ is just a transposition with the additional restriction that the substring moved should be a $block$.
In [5], Bafna and Pevzner introduced the notion of a cycle graph of a permutation, and used the cycle graph to obtain improved lower bounds for the Sorting by Transposition problem. Since, a block move is nothing but a transposition, this lower bounds and some other properties introduced in [5] also holds in case of Sorting by Block Moves. We shall use some of these properties in our analysis for block sorting.
The computational complexity of Sorting by Transposition is still open, where as Block Sorting has been proved to be NP-Hard [7]. In the report a new characterization of Block Sorting has been given using the transposition cycle graph. Next a simple heuristic is suggested which is shown to exhibit an approximation guarantee of 2 in most of the cases. We have used certain proved facts on Sorting by Transposition which also holds for Block Sorting to analyze the heuristic designed.

Definition 7 (Cycle Graph). *A $cycle\ graph$ of a permutation π, denoted by $G(\pi)$, is a directed edge color graph with vertex set 0, 1, 2, ..., n, n+1 and edge set defined as follows: For all $1 \leq i \leq n+1$, grey edges are directed from $i-1$ to i and black edges from π_i to π_{i-1}.*

As an example we show the $cycle\ graph$ for the permutation 1 4 3 2 5 the the following figure.

Definition 8 (Alternating Cycle). *An $alternating\ cycle$ of a $cycle\ graph$ is a cycle where each pair of adjacent*

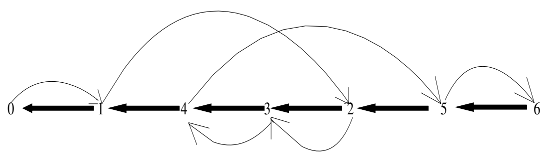


Fig. 1. A Cycle Graph

edges are of different colors. In addition to that, we call the alternating cycles with odd number of black edges as $odd\ cycles$ and the ones with even number of black edges as $even\ cycles$. As an example we show the alternating cycles of the graph of Figure 1.

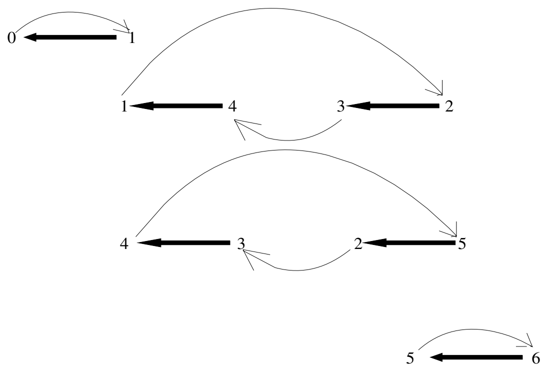


Fig. 2. Alternating Cycles

Observation 9.
- *The cycle graph $G(\pi)$ can be uniquely decomposed into alternating cycles.*
- *Each edge participates in $exactly\ one$ alternating cycle.*
- *The cycle graph of the identity permutation contains the maximum number of cycles and all of them are odd cycles (having a length 1).*
- *The number of (odd) cycles of the identity permutation of n elements is $n+1$.*

The sorting by transposition problem can be viewed as increasing the number of odd alternating cycles $C(\pi)$ in the given permutation π to $n+1$. Let the number of odd cycles reduced by a transposition ρ be $\Delta C(\rho)$.

Lemma 10. $\Delta C(\rho) \in \{2, 0, -2\}$ *[5].*

Thus from the above lemma a new lower bound for the sorting by transposition is obtained as (n+1-$C_{odd}(\pi)$)/2. Since the block sorting problem is a variantion of the sorting by transposition problem, all the above observations holds in the case of block sorting too. Hence we have the following lower bound $bs(\pi)$ for block sorting.

Theorem 11. *$bs(\pi) \geq$ (n+1-$C_{odd}(\pi)$)/2.*

IV. The Heuristic for Sorting by Block Moves

For any given permutation of n elements, we always append the elements 0 and $n+1$ to the beginning and end of the permutation respectively.

Definition 12 (k-moves)**.** *We call the block moves which increase the number of cycles by k k-moves. Clearly k* $\in \{2, 0, -2\}$.

Now we shall state and prove certain lemmas which would lead to the designed heuristic.

Lemma 13. *(Bafna and Pevzner [5]) If a transposition ρ acts on a cycle and creates more than one new cycle, in $G(\pi\rho)$ then ρ is a 2-move.*

Lemma 14. *(Bafna and Pevzner [5]) If a transposition ρ acts on edges belonging to exactly two different cycles then ρ is a 0-move.*

Since a block move is nothing other than a transposition, we can say the following:

Lemma 15. *If a block move ρ acts on a cycle and creates more than one new cycle, in $G(\pi\rho)$ then ρ is a 2-move.*

Lemma 16. *If a block move ρ acts on edges belonging to exactly two different cycles then ρ is a 0-move.*

Lemma 17. *If a block move reduces the number of blocks by 2, then it also increases the number of cycles by 2, i.e. it is a 2-move. Further two new odd cycles gets created here. In other words, the number of odd cycles gets increased by two here.*

Proof: For proving the above lemma, we first look at the cases which allow a block move which reduce the number of blocks by 2 (refer to Figure 3 on the next page). In the first case, we note that x, $x+1$, y, $y+1$, z are in the same cycle since there is a gray-black path from x to z via $x+1$, y and $y+1$.The block move to join y with $y+1$ creates at least 2 new odd cycles. Thus it is a transposition which acts on a single cycle and creates more than one new odd cycles. Hence it is a 2-move.
In the second case, we note that $x+2$, x, $x+1$ are in the same cycle since there is a black edge from $x+2$ to x, the outgoing gray edge from x is to $x+1$ and the incoming gray edge to $x+2$ is from $x+1$.Since

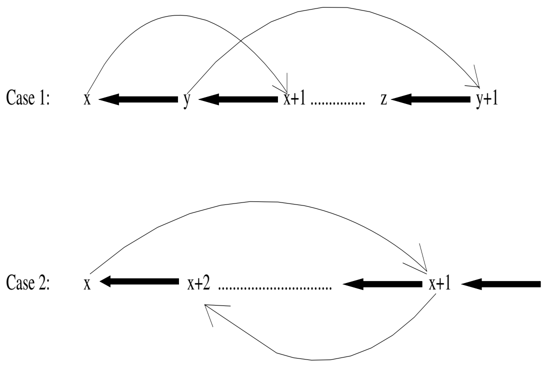


Fig. 3. Cases Allowing An Increment of 2

both the incoming and out going edges of $x+1$ is in the same cycle, both its black edges also must lie in the same cycle. The block movement of $x+1$ in between x and $x+2$ creates at least 2 new cycles. Thus it is a transposition which acts on a single cycle and creates more than one new cycles. Hence it is a 2-move. We can note that two of the new cycles created are cycles having only *one* black edges. That is, they are *odd* cycles. Hence the number of odd alternating cycles increases by two in this case. By similar arguments, we can show that a block move which reduces the number of blocks by 3 is also a 2-move. ■

Lemma 18. *If we have a vertex which repeats in a cycle, i.e if there are two adjacent black edges in the permutation graph belonging to the same cycle, we have a 2-move.*

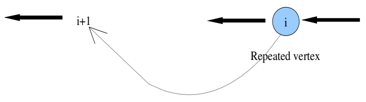


Fig. 4. A Repetition in a Cycle

Proof: If moving the repeated vertex results in the reduction of the number of blocks by 2 or 3, then the move is a 2-move (by lemma 6). Else the repeated vertex say x, can be moved to a position adjacent its successor to form a block. Clearly, the move acts on a single cycle and creates a new 1-cycle and some other cycles. Thus this is also a 2-move by lemma 4. Hence the lemma is proved. ■

Call a cycle in a permutation with k black edges as a k-cycle. A k-cycle in the breakpoint graph is called short if $k \leq 3$; otherwise, it is called long. A breakpoint graph is called *simple* if it contains only short cycles. A permutation π is called simple if $tG(\pi)$

is simple. A common technique in the genome literature is to transform a permutation into an *equivalent* simple permutation. The transformation involves inserting new elements in the permutation and splitting up the long cycles into short cycles. The technique is referred to as $(g, b) - split$ in the literature. For a detailed description of the procedure the reader is referred to the papers [6], [7], and [8]. Some of the results of these papers is used here.

Lemma 19. *(Lin and Xue [7]) Every permutation can be transformed safely into a simple one.*

Lemma 20. *(Hannenhalli and Pevzner [5]) Let $\hat{\pi}$ be a simple permutation that is equivalent to π, then every sorting of $\hat{\pi}$ mimics a sorting of π with the same number of operations.*

The heuristic given in this report first transforms the given permutation π into an equivalent simple permutation $\hat{\pi}$ in each step, then it finds a sorting sequence for $\hat{\pi}$, and, finally, the sorting of $\hat{\pi}$ is mimicked on π. Therefore, throughout most of the report we will be concerned only with simple permutations and short cycles.

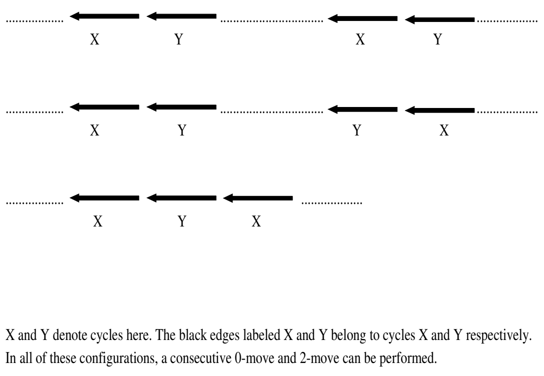
Definition 21 (Conjoined Cycles)**.** *Two cycles are called $conjoined$ if they meet at some point i.e. they have a vertex in common. The common vertex is called $conjoining$ $vertex$ for those two cycles.*

Lemma 22. *If we have two cycles conjoined at two or more vertices, we have a 0-move followed by a 2-move.*

Proof: The different cases are shown in the following figure. In the first move, we move one of the elements common to both the cycles to a place in one of the cycles, such that we have a 2-move in the next step. If we consider simple permutations, we can clearly observe that there will always be a common block i, such that in the first move it could be either moved to the left of block $i+2$ to havr the configuration $i\ i+2$, or can be moved to a position to have the following configuration: $i\ x\ i+1$. Since this move involves only two cycles, by lemma 3 it is a 0-move. The next move clearly reduces the number of blocks by at least 2. Thus it is a 2-move. ■

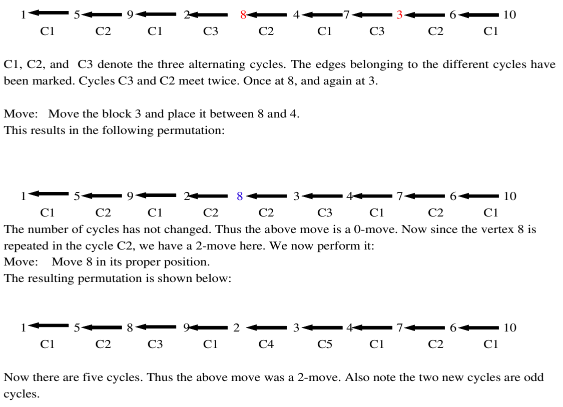
Figure 6 shows an example. We can further note that, in case of short cycles, the 2-move always increases the number of *odd* cycles by 2.

Lemma 23. *If there is a 2-cycle, then there is a 0-move which reduces the number of even cycles by two, and simultaneously increases the number of odd cycles by*



X and Y denote cycles here. The black edges labeled X and Y belong to cycles X and Y respectively.
In all of these configurations, a consecutive 0-move and 2-move can be performed.

Fig. 5. Cycles conjoined at two or more vertices


1 5 9 2 8 4 7 3 6 10
C1 C2 C1 C3 C2 C1 C3 C2 C1
1 5 9 2 8 3 4 7 6 10
C1 C2 C1 C2 C2 C3 C1 C2 C1
1 5 8 9 2 3 4 7 6 10
C1 C2 C3 C1 C4 C5 C1 C2 C1
C1, C2, and C3 denote the three alternating cycles. The edges belonging to the different cycles have been marked. Cycles C3 and C2 meet twice. Once at 8, and again at 3.

Move: Move the block 3 and place it between 8 and 4.
This results in the following permutation:

The number of cycles has not changed. Thus the above move is a 0-move. Now since the vertex 8 is repeated in the cycle C2, we have a 2-move here. We now perform it:
Move: Move 8 in its proper position.
The resulting permutation is shown below:

Now there are five cycles. Thus the above move was a 2-move. Also note the two new cycles are odd cycles.

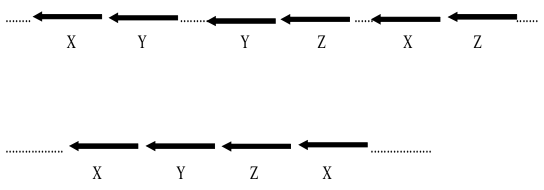
Fig. 6. An Example of a (0,2) move

two.

Proof: This is quite trivial to show. The point here is that later on it is shown that this 0-move actually is good. ■

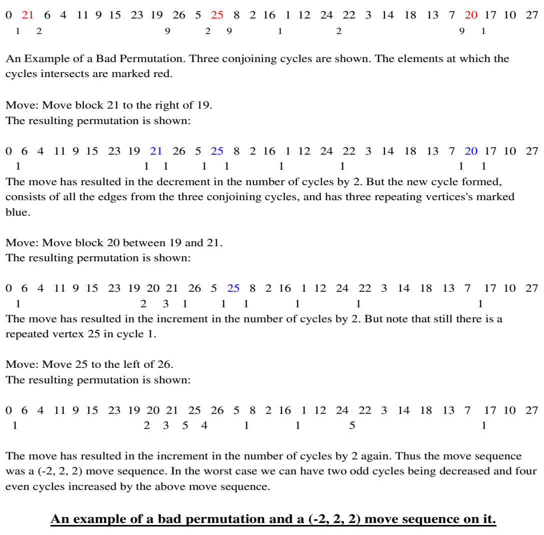
Lemma 24. *If we have three conjoined cycles, we have a -2-move followed by a couple of 2-moves.*

Proof: In cases of each of the arrangements of Figure 7, we can perform a move to tie these three cycles together to form a new cycle consisting of number of edges equal to the sum of the number of edges of the three individual cycles. This is a -2-move since the number of cycles here decreases by 2. But it can be shown that this move can be followed up by two consequtive 2-moves. The reasoning is, the number of vertices in the new cycle formed after the -2-move will be less than twice the number of edges in the cycle by $at\ least$ 2. Thus at least two vertices will be there in the new cycle which repeats in the cycle. Thus by lemma 7 we have at least two 2-moves here. For a simple



X , Y and, Z denote cycles here. The black edges labeled X, Y and, Z belong to cycles X, Y and, Z respectively. In all of these configurations, a -2-move followed by two consecutive 2-moves can be performed.

Fig. 7. Three conjoined cycles



```
0  21  6  4  11 9 15  23 19  26  5  25  8  2 16  1 12  24  22  3  14  18  13  7  20 17 10  27
 1   2                    9       2   9        1       2                       9   1
```

An Example of a Bad Permutation. Three conjoining cycles are shown. The elements at which the cycles intersects are marked red.

Move: Move block 21 to the right of 19.
The resulting permutation is shown:

```
0  6  4  11 9 15  23 19  21  26  5  25  8  2 16  1 12  24  22  3  14  18  13  7  20 17 10  27
 1                      1   1     1   1        1       1                      1   1
```

The move has resulted in the decrement in the number of cycles by 2. But the new cycle formed, consists of all the edges from the three conjoining cycles, and has three repeating vertices's marked blue.

Move: Move block 20 between 19 and 21.
The resulting permutation is shown:

```
0  6  4  11 9 15  23 19 20 21  26  5  25  8  2 16  1 12  24  22  3  14  18  13  7   17 10  27
 1                     2   3  1      1   1       1       1                       1
```

The move has resulted in the increment in the number of cycles by 2. But note that still there is a repeated vertex 25 in cycle 1.

Move: Move 25 to the left of 26.
The resulting permutation is shown:

```
0  6  4  11 9 15  23 19 20 21  25  26  5  8  2 16  1 12  24  22  3  14  18  13  7   17 10  27
 1                     2   3   5   4       1        1       5                       1
```

The move has resulted in the increment in the number of cycles by 2 again. Thus the move sequence was a (-2, 2, 2) move sequence. In the worst case we can have two odd cycles being decreased and four even cycles increased by the above move sequence.

An example of a bad permutation and a (-2, 2, 2) move sequence on it.

Fig. 8. An Example of a (-2,2,2) move sequence on a bad permutation

permutation, note that this kind of permutations contains only 3-cycles.
We would like to comment here, that such kind of simple permutations where there are no *repeated vertices*, no two cycles intersecting more than once i.e. conjoined at two or more vertices and, where there are only 3-cycles are very rare. This permutation however exists and can be termed as *bad cases* for block sorting. Further, we have no idea about the nature of the cycles that ultimately result after the three moves. That is, in the worst case, we could have four new even cycles at the cost of two odd cycles. We have considered this worst case in our analysis. An example of (-2, 2, 2) move sequence on a bad permutation has been shown in Figure 8. Here we have gained two odd cycles and two even cycles at the cost of two odd cycles in the beginning. But this might well be just one sequence of moves. There can be several other ways of performing them. ■

We now present the heuristic for block sorting.

Algorithm 1

```
Input: Two permutations $\pi$ and $id_n$
Output: The approximate block sorting distance $bs(\pi)$ between $\pi$ and $id_n$
Construct the cycle graph $G(\pi)$ of the permutation $\pi$
$bs(\pi) = 0$.
If the permutation is not a simple one, convert it into a simple permutation.
Transform the permutation to its kernel permutation.
while ($\pi \neq id$) do
  if There is a move which reduces the number of blocks by 2 or 3 then
    Perform a 2-move
    $bs(\pi) = bs(\pi) + 1$
  else if There is a vertex which repeats in a cycle then
    Perform a 2-move
    $bs(\pi) = bs(\pi) + 1$
  else if There are two cycles conjoined at two or more vertices then
    Perform the (0,2) move sequence
    $bs(\pi) = bs(\pi) + 2$
  else if There is a 2-cycle then
    Perform the 0-move to increase the number of odd cycles by 2
    $bs(\pi) = bs(\pi) + 1$
  else
    This is a bad case. Perform the (-2,2,2) move sequence
    $bs(\pi) = bs(\pi) + 3$
  end if
end while
Output the distance $bs(\pi)$
```

Theorem 25. *The above heuristic for block sorting will sort any arbitrary permutation in at most twice the number of moves taken by an optimal block sorting algorithm in most of the cases.*

Proof: In the proof, the performance of the heuristic shall be considered in all the above mentioned cases. Then we shall use a *mixed* objective function which gives different weights to odd and even cycles, to establish the performance guarantee of the heuristic. This particular technique was used to establish the performance guarantee of a *Sorting by Transposition* algorithm in [5].

The Objective Function In the identity permutation with n numbers, there are $n+1$ odd cycles and 0 even cycles. This heuristic can thus be thought to increase the number of odd cycles and decrease the number of even cycles until there are $n+1$ odd cycles and 0 even cycles. Let $C_{odd}(\pi)$ and $C_{even}(\pi)$ denote the number of odd and even cycles respectively in permutation π. Define an objective function

- $f(\pi) = xC_{odd}(\pi) + C_{even}(\pi)$

where $x \geq 1$.

Let π_i denote the $identity$ permutation. Then,

- $f(\pi_i) = xC_{odd}(\pi_i) = x(n+1)$

Now, for an arbitrary permutation, which is not the identity permutation,

- $C_{odd}(\pi) = C(\pi) - C_{even}(\pi)$

Finally by substitution we have,

- $f(\pi) = xC(\pi) - (x-1)C_{even}(\pi)$

Since $f(\pi)$ is maximized by the identity permutation, block sorting can be considered as a process of maximizing $f(\pi)$.

Now we shall consider various cases which might arise by a block sorting move and observe the changes in $f(\pi)$ in those cases. Then we shall use these observations to analyse the heuristic. Let the change in $f(\pi)$ by a move, be denoted by $\Delta f(\pi)$.

- **Case 1:** $C(\pi)$ increases by two odd cycles.

Here $\Delta f(\pi) = x(C(\pi)+2) - (x-1)C_{even}(\pi) - xC(\pi) + (x-1)C_{even}(\pi)$
Or, $\Delta f(\pi) = 2x$

- **Case 2:** $C(\pi)$ increases by two even cycles.

Here $\Delta f(\pi) = x(C(\pi)+2) - (x-1)(C_{even}(\pi)+2) - xC(\pi) + (x-1)C_{even}(\pi)$
Or, $\Delta f(\pi) = 2$

- **Case 3:** $C(\pi)$ remains the same, $C_{even}(\pi)$ decreases by two.

Here $\Delta f(\pi) = xC(\pi) - (x-1)(C_{even}(\pi)-2) - xC(\pi) + (x-1)C_{even}(\pi)$
Or, $\Delta f(\pi) = 2x-2$

- **Case 4:** $C(\pi)$ is decreased by two odd cycles.

Here $\Delta f(\pi) = x(C(\pi)-2) - (x-1)C_{even}(\pi) - xC(\pi) + (x-1)C_{even}(\pi)$
Or, $\Delta f(\pi) = -2x$

Now consider the cases of the heuristic:

- **Case 1:** There is a vertex which repeats in a cycle. Or there is a block move to reduce the number of blocks by 2 or 3.

Here there is a move which increases $C(\pi)$ by two. In the worst case $C_{even}(\pi)$ increases by two. But still there is a gain of 2 here.

- **Case 2:** There are two cycles conjoined at two or more vertices.

Here there is a move which would not change the number of cycles, followed by a move which increases $C(\pi)$ and $C_{odd}(\pi)$ by two. There can be two sub-cases. In one case, the first move increases $C_{even}(\pi)$ by two, in the other it does not. However, in both the cases $C(\pi)$ remains the same after the first move. The average gain in the first case is thus $(2x-2)/2 = x-1$ per move, while in the second it is just $2x/2 = x$ per move. Both the sub-cases shall be considered here.

- **Case 3:** There is a 2-cycle.

Here the gain out of the 0-move is $2x-2$.

- **Case 4:** There are three conjoined cycles.

Here we have a move which decreases $C(\pi)$ by two followed by two consequtive moves each of which increases $C(\pi)$ by two. In the worst case, $C_{odd}(\pi)$ decreases by two in the first move and $C_{even}(\pi)$ increases by two in each of the following moves. Thus the average gain here is $(4-2x)/3$ per move.

The performance guarantee of the heuristic is given by:
$$\frac{2x}{min\{2,x-1,x,2x-2,(4-2x)/3\}}$$

In the worst case, we have to find $min\{2, x -$

$1, x, 2x-2, (4-2x)/3\}$.We have plotted the five functions (Figure 9). The minimum value of intersection can be found out from the plot. We find $x = 0.8 = 4/5$ is the minimum. Since it is the intersection of $y = x$ and $y = (4-2x)/3$, the performance ratio can be found out to be

$$\frac{2x}{min\{2,x-1,x,2x-2,(4-2x)/3\}} = \frac{2*4/5}{(4-2*4/5)/3} = \frac{3}{2.5/4-1} = \frac{3}{3/2} = 2$$

This proves the above theorem. ■

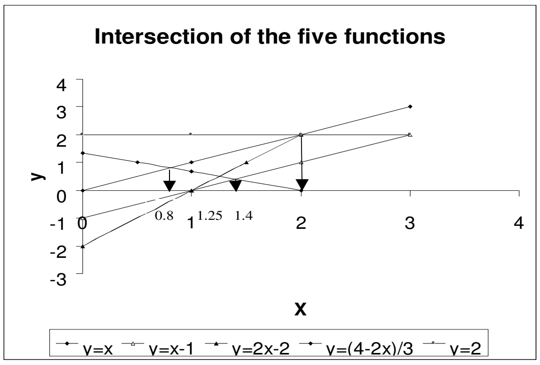


Fig. 9. The various functions determining the performance of the Heuristic

The steps of the heuristic on the permutation 5 4 3 2 1 is shown in Figure 10.

Given permutation 5 4 3 2 1

After adding 0 and 6 it becomes 0 5 4 3 2 1 6

Kernel permutation: 0 5 4 3 2 1 6

Moves Performed	Kernel Permutation	Equivalent Block Moves
0 5 4 3 2 1 6	0 5 4 3 2 1 6	5 4 3 2 1
0 5 1 4 3 2 6	0 5 1 4 3 2 6	5 1 4 3 2 (0-move)
0 1 4 3 2 5 6	1 4 3 2 5	1 4 3 2 5 (2-move)
1 4 2 3 5	1 3 2 4	1 4 2 3 5 (0-move)
1 2 3 4	1	1 2 3 4 5 (2-move)

Fig. 10. Moves performed by the heuristic to sort the permutation 5 4 3 2 1

V. Conclusion

The heuristic performs even better than that of a 2-approximation in majority of the cases, but it is not clear whether the performance guarantee can be improved by this approach. A rigorous case analysis could do it. In this work some sub-cases have been ignored, as they lead to too much complications. But we conjecture that taking them into consideration can lead to a better performance in the heuristic. Also we do not have any proof that there cannot be any permutation which do not fall into any of the cases considered by the heuristic. We only conjecture that such a permutation if at all exists must be very large.

References

[1] M. Mahajan, R.Rama, V. Raman, and S. Vijaykumar. Approximate block sorting. International Journal of Foundation of Computer Science, to appear.

[2] M. Mahajan, R.Rama, and S. Vijaykumar. Towards constructing optimal block move sequences. In proc. of the 10th International Computing and Combinatorial conference, COCOON 2004, LNCS vol.3106, pages 33-42. Springer-Verlag, Aug 2004.

[3] P. Pevzner. Computational Molecular Biology: An Algorithmic Approach. MIT Press, Cambridge, MA, USA, 2000.

[4] V.Bafna and P.Pevzner. Genome rearrangements and sorting by reversals. SIAM Journal on Computing, 25:272-289, 1996.

[5] V.Bafna and P.Pevzner. Sorting by transpositions. SIAM Journal on Discrete Mathematics 11(2):224-240, may 1998.

[6] W.W. Bein, L.L. Larmore, L. Morales, and I.H. Sudborough. A Polymomial Time 2-Approximation for Block Sorting. Unpublished manuscript, available at http://www.egr.unlv.edu/ bein/pubs/twoblock.ps.

[7] W.W. Bein, L.L. Larmore, L. Morales, and I.H. Sudborough. Block sorting is hard. Intl. Jl. of Foundations of Computer science, 14:425-437, 2003.

[8] G.H Lin and G Xue. Signed genome arrangement by reversals and transpositions: Models and approximations. *Theoretical Computer Science*, 259:513-531, 2001.