Les Cahiers du GERAD ISSN: 0711–2440

Optimal sorting with comparisons involving three elements

C. Audet

G–2025–54

August 2025

Optimal sorting with comparisons involving three elements

Dedicated to the memory of Pierre Hansen, the world's finest octagonist.

Charles Audet

GERAD & Département de mathématiques et de génie industriel, Polytechnique Montréal, Montréal, (Qc), Canada, H3T 1J4

charles.audet@gerad.ca

August 2025
Les Cahiers du GERAD
G–2025–54

Abstract : The present work studies the problem of sorting using comparisons involving three elements at a time. Each comparison only identifies the smallest, middle, and largest elements among the three. We propose an algorithm based on six heuristic strategies, aiming to minimize the number of comparisons in the worst case. We report optimal or near-optimal results for all values up to $n = 40$ elements, and for selected instances up to $n = 1000$.

Keywords : Comparison sort, optimal sorting, worst-case analysis, heuristics

1 Introduction

The problem of sorting n elements with the minimum possible number of comparisons has been extensively investigated in the context of *pairwise* comparisons. In such a comparison, given two distinct real numbers a and b, we learn whether $a < b$ or $a > b$. A *comparison sort* is an algorithm that monotonically orders the n elements; well-known examples include quicksort, heapsort, bubblesort, and mergesort.

Worst-case analyses study the largest number of comparisons required by a sorting algorithm. The objective is to minimize that maximum. Because there are $n!$ possible permutations of n distinct elements and each pairwise comparison yields only two outcomes, the value $\lceil \log_2(n!) \rceil$ is a valid lower bound. That lower bound is attained by the Ford-Johnson algorithm [3] for values of n up to 11 and for $n \in \{20, 21\}$. The minimum number of comparisons is known for $n \in \{12, 13, 14, 15, 19, 22\}$. The optimal value is unknown for $n \in \{16, 17, 18\}$ and for $n \geq 23$. This problem is studied in a series of papers by Peczarski [4, 5, 6] and in [2].

The present work considers situations where the comparisons are not pairwise, but involve three elements. A comparison of the three elements only indicates which is the smallest, the intermediate and the largest. If the three numbers are all different, then there are six possible outcomes. A lower bound on the guaranteed minimal number of comparisons for any algorithm sorting $n \geq 3$ distinct numbers is therefore $\ell_n = \lceil \log_6(n!) \rceil$, but that bound only appears to be tight for the trivial cases where $n = 3$.

We propose an algorithm to sort $n \geq 3$ numbers and analyse its worst-case behavior. The algorithm uses six heuristic methods, and takes the best of them for each value of n. Section 2 recalls a definition to help manipulate ordered sets of elements, and gives sufficient number of comparisons to order particular configurations. Section 3 describes six heuristic methods to sort n elements. Section 4 compiles and comment the results of the heuristics for values of n up to 1000.

2 Definitions and general results on three-way comparisons

We recall the following definition and a preliminary theorem from [1], which studied in detail the case of sorting 15 elements using three-way comparisons. In that setting, the problem is formulated as sorting n distinct marbles using a three-tray balance.

Definition 2.1. A *chain* is a totally ordered set of elements. A *v-chain* is a set of elements composed of two chains sharing the same least element.

The next theorem analyzes particular configurations that can be sorted efficiently.

Theorem 2.2. The following hold:

i- $n - 2$ comparisons suffice to sort two chains together with a v-chain, totaling $n \geq 3$ elements.

ii- $n - 1$ comparisons suffice to sort four chains totaling $n \geq 4$ elements.

iii- $n - 2$ comparisons suffice to sort four nonempty chains totaling $n \geq 6$ elements, provided at least one chain has length 1.

Proof.

(i-) The proof is by induction on n, the total number of elements. The result is trivial for $n = 3$ since a single comparison will sort them.

Assume the statement holds for some $n \geq 3$. Consider two chains and a v-chain containing a total of $n + 1$ elements. A single comparison between the least element of each group identifies the overall smallest element (if a group is empty, select an element from another group containing at least two elements). This comparison also reveals the relative order of the least elements of the two chains, allowing them to be merged into a v-chain. We now have two v-chains.

Consequently, the two chains may be merged into a v-chain. The $n+1$ elements are now partitioned into two v-chains.

Removing the overall least element breaks one v-chain into two chains; the induction hypothesis then sorts the remaining n elements in $n-2$ comparisons. Including the initial comparison yields $n-1$ comparisons for $n+1$ elements.

(ii-) Compare the least elements of three of the chains. This produces a configuration covered by (i), which completes the sorting.

(iii-) First compare the least elements of the three longest chains; let ℓ be the smallest and m the middle element. Let s be the next element in ℓ's chain (if ℓ's chain has length 1, s does not exist, but the proof still holds). Let the fourth chain consist of a single element a. Use a second comparison to compare s, m to the fourth chain composed of a single element a. Figure 1 illustrates the situation; the elements are represented by circles, and an edge joining two elements indicates that the lowest one is less than the other.

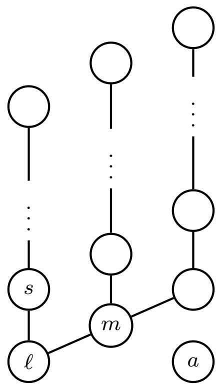

Figure 1: Case (iii-) of Theorem 2.2

If a is larger than $\min(s,m)$, then the $n-2$ elements larger than this minimum (including a) form a v-chain and two chains; case (i) sorts them in $n-4$ further comparisons.

If a is smaller, then ℓ and a are both less than $\min(s,m)$, and the remaining $n-3$ elements form three chains; case (i) sorts them in $n-5$ comparisons, plus one more to order $\{\ell, a\}$.

In either situation, the total is $2+(n-4) = n-2$ comparisons. □

By symmetry, the above results trivially hold when extending the definition of a v-chain to pairs of chains that have the same largest, instead of least, element. These will be called *reversed v-chains*.

3 Heuristic methods

The worst-case number of comparisons required by our algorithm to sort n numbers will be denoted by t_n. Trivially, we have $t_1 = 0$ and $t_2 = t_3 = 1$. This section is partitioned into six subsections, each devoted to a heuristic. For $n \geq 4$, the value t_n will be set to be the minimum value produced by the heuristics. When each heuristic is called to sort $n \geq 4$ values, it is assumed that the values $t_1, t_2, \ldots, t_{n-1}$ are available.

3.1 Single-insertion heuristic (s_n)

The worst-case number of comparisons required by the single-insertion heuristic to sort n numbers is denoted by s_n. The heuristic starts by performing t_{n-1} evaluations to sort $n-1$ elements, then it inserts the other element.

Let α_n be the number of comparisons required to insert one element into a list of $n-1$ sorted elements. Partition the $n-1$ elements into five groups of cardinality $\{a, 1, b, 1, c\}$. The a smallest, followed by a singleton denoted by i, the b next ones, another singleton denoted by j and the c largest so that $a \geq b \geq c \geq a-1$ and $a+b+c = n-3$. Figure 2 illustrates the partition; an edge joining a pair of elements indicates that the one to the right is greater than or equal to the one to the left.

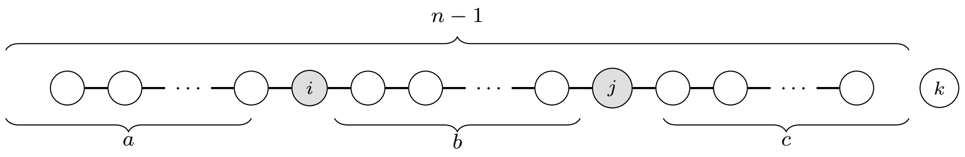

Figure 2: Partition of $n-1$ elements into five groups

To insert an element k into this ordered list, the heuristic first compares i, j and k. There are three possible outcomes: k needs to be compared with a group of either a, b or c elements. But since $a \geq b \geq c$, the worst case complexity occurs with the group of a elements. It follows that α_n and the total number of comparisons s_n are

$$\alpha_n = \begin{cases} 1 & \text{if } n \leq 3 2 \\ 1 + \alpha_{a+1} & \text{if } n \geq 4, \text{ with } a = \lceil \frac{n-3}{3} \rceil \end{cases} \qquad \text{and} \qquad s_n = t_{n-1} + \alpha_n.$$

3.2 Pair-insertion heuristic (p_n)

The worst-case number of comparisons required by the pair-insertion heuristic to sort n numbers is denoted by p_n. The heuristic starts by performing t_{n-2} evaluations to sort $n-2$ elements, then it inserts the remaining pair of elements.

Let β_n be the number of comparisons required to insert a pair of elements into a list of $n-2$ sorted elements. Partition the $n-2$ elements into three groups. The a smallest, followed by a singleton denoted by i and the b largest so that $a \geq b \geq a-1$ and $a+b = n-3$. Figure 3 illustrates the partition.

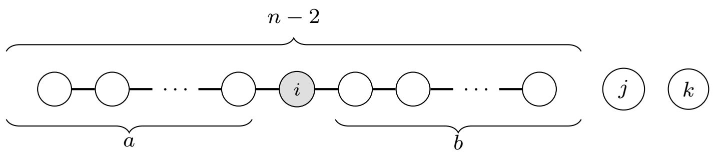

Figure 3: Partition of $n-2$ elements into three groups

To insert the pair of elements $\{j, k\}$ into this ordered list, the heuristic first compares i, j and k. By relabeling if necessary, assume that $j \leq k$. Three cases need to be considered.

- If $j \leq k \leq i$, then α_{a+1} comparisons (from Section 3.1) are sufficient to insert k into the a elements, and at most α_{a+1} comparisons to insert j into the same a elements (recall that $j \leq k$).
- If $j \leq i \leq k$, then α_{a+1} comparisons are enough to insert j in to the a elements, and α_{b+1} to insert k into the b elements.
- If $i \leq j \leq k$, then α_{b+1} comparisons are sufficient to insert j in to the b elements, and at most α_{b+1} comparisons to insert k into the same b elements (recall that $j \leq k$).

Now, since $a \geq b$, the worst case complexity occurs with the group of a elements. It follows that β_n and the total number of comparisons p_n are

$$\beta_n = \begin{cases} 1 & \text{if } n = 2 \text{ or } n = 3 \\ 1 + 2\alpha_{a+1} & \text{if } n \geq 4, \text{ with } a = \lceil \frac{n-3}{2} \rceil \end{cases} \qquad \text{and} \qquad p_n = t_{n-2} + \beta_n.$$

3.3 4chains heuristic (f_n)

The worst-case number of comparisons required by the 4chains heuristic to sort n numbers is denoted by f_n.

The second statement of Theorem 2.2 states that $n-1$ comparisons are sufficient to sort four chains totaling $n \geq 4$ elements. The situation where at least one chain is of length 1 is treated in another heuristic. An enumerative way to use this result is to examine all ways of partitioning n elements into 4 non-empty groups of size $\{a, b, c, d\}$, and then to order each group into a chain using $t_a + t_b + t_c + t_d$ comparisons. The following pseudo code shows how to compute f_n, the corresponding best way to sort n elements.

Algorithm 3.1. 4chains-insertion heuristic

```
0. Initialisation
    Set $f_n = +\infty$, where $n$ is the number of elements to sort
1. Main iterations
    for $a = 2$ to $\lfloor n/4 \rfloor$
      for $b = a$ to $\lfloor (n-a)/3 \rfloor$
        for $c = b$ to $\lfloor (n-a-b)/2 \rfloor$
          $d = n - a - b - c$
          $f_n \leftarrow \min\{f_n,\ t_a + t_b + t_c + t_d + (n-1)\}$
2. End
    Return $f_n$
```

3.4 1v-2chains heuristic (v_n)

The worst-case number of comparisons required by the 1v-2chains heuristic to sort n numbers is denoted by v_n.

The first statement of Theorem 2.2 states that $n-2$ comparisons suffice to sort two chains together with a v-chain, totaling $n \geq 3$ elements. A similar technique than the one presented in the last section is applied. The v-chain is composed of the third and fourth groups and therefore the sum of the number of elements of the four chains is $n+1$ and not n. The following pseudo code shows how to compute v_n, the corresponding best way to sort n elements.

Algorithm 3.2. 1v-2chains-insertion heuristic

```
0. Initialisation
    Set $v_n = +\infty$, where $n$ is the number of elements to sort
1. Main iterations
    for $a = 1$ to $\lfloor (n+1)/4 \rfloor$
      for $b = a$ to $\lfloor (n+1-a)/3 \rfloor$
        for $c = b$ to $\lfloor (n+1-a-b)/2 \rfloor$
          $d = n + 1 - a - b - c$
          $v_n \leftarrow \min\{v_n,\ t_a + t_b + t_c + t_d + (n-2)\}$
2. End
    Return $v_n$
```

3.5 1mono-3chains heuristic (m_n)

The worst-case number of comparisons required by the 1mono-3chains heuristic to sort n numbers is denoted by m_n.

The last statement of Theorem 2.2 states that $n-2$ comparisons suffice to sort four non-empty chains with at least one chain of length one, totaling $n \geq 6$ elements. A similar technique than the one presented in the last section is applied. The following pseudo code shows how to compute m_n, the corresponding best way to sort n elements.

Algorithm 3.3. 1mono-3chains-insertion heuristic

```
0. Initialisation
    Set $m_n = +\infty$, where $n$ is the number of elements to sort
1. Main iterations
    for $a = 1$ to $\lfloor (n-1)/3 \rfloor$
      for $b = a$ to $\lfloor (n-1-a)/2 \rfloor$
        $c = n - 1 - a - b$
        $d = 1$
        $m_n \leftarrow \min\{m_n,\ t_a + t_b + t_c + (n-2)\}$
2. End
    Return $m_n$
```

3.6 Specific heuristic to sort $n = 7, 9, 13, 15$ and 21 elements (g_n)

The worst-case number of comparisons required by the specific heuristic to sort n numbers is denoted by g_n.

The paper [1] proposes ways to sort up to 9 elements, and $n = 15$ elements. Some of these values are lower than the ones produced by the best of the above-mentioned heuristics. Let g_n denote the lest known number of comparisons to order n elements. The paper shows that $g_7 = 6$, $g_9 = 9$ and $g_{15} = 20$. These three values are less than the ones produced by the five heuristics from the previous sections. These heuristics are also suboptimal for $n = 13$ and for $n = 21$. The next proposition details a way to sort 13 elements using at most 16 comparisons. The case with $n = 21$ is studied next.

Proposition 3.1. 16 comparisons suffice to sort 13 elements.

Proof. First, take nine elements and form three chains of length three, and then compare the middle of each chain. Let a, b and c denote the least, intermediate and largest of the last comparison. Next, compare the largest element of the original chain containing a with two other elements, and denote the largest by d (it is certainly larger than a). Similarly, compare the least element of the original chain containing c with the two remaining elements, and denote the least by e (it is certainly least than c). The middle element of these last two comparisons are shaded in Figure 4.

Apply a seventh comparison on the two shaded elements with b. At this point, every element but two are known to be less than or greater than b. Use an eight comparison on b an on these two elements and label them f and g, where f is less than g. By symmetry, suppose that the number of elements greater than b exceeds or equals the number of elements less than b. Four cases need to be studied.

- Figure 5 illustrates the situation where only three elements are less than b. The nine elements larger than b form four chains with one of length 1. Theorem 2.2(iii-) guarantees that 7 more comparison suffice to sort them. The three element less than b may be sorted with a single comparison.

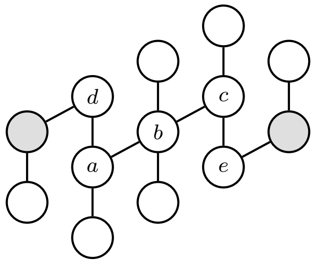

Figure 4: Disposition of 13 elements after 6 comparisons

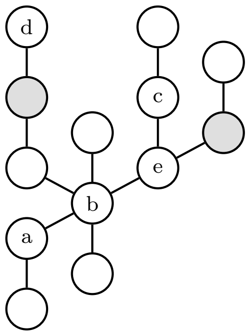

Figure 5: Three elements are less than b and 9 elements are greater than b

- If four elements are less than b, then Theorem 2.2(i-) guarantees that they may be sorted with 2 comparisons. The eight larger elements may be sorted with 6 comparisons by part (iii-) of the same theorem.
- If five elements are less than b, then these elements either contain the chain composed of f and g or the chain composed of f and a shaded element. In both cases Theorem 2.2(i-) guarantees that they may be sorted with 3 comparisons. The seven larger elements may be sorted with 5 comparisons by part (iii-) of the same theorem.
- If six elements are less and six are greater than b, then both groups contain two chains of length two and two chains of length one, and therefore Theorem 2.2(iii-) guarantees each group may be sorted with 4 comparisons.

In all four cases, 8 additional comparisons were performed for a total of 16. It follows that 16 comparisons are sufficient to sort 14 elements. □

The previous proposition describes an heuristic method with $g_{13} = 16$. The next proposition details a way to sort 21 elements using at most 34 comparisons.

Proposition 3.2. 34 comparisons suffice to sort 21 elements.

Proof. Start by creating 7 disjoint chains of length three. Then, use $t_7 = 6$ comparisons to order the central elements of the chains. Figure 6 illustrates the result.

The next step requires 3 comparisons involving the central grey element o with the three pairs $\{a, b\}$, $\{c, d\}$ and $\{e, f\}$. At this point, i.e., after 16 comparisons, it is know whether each element is greater than or less than the central element o. The analysis continues by studying the number of elements in each partition. By symmetry, assume that there are at least as many elements in the group of larger elements than in the group of least elements.

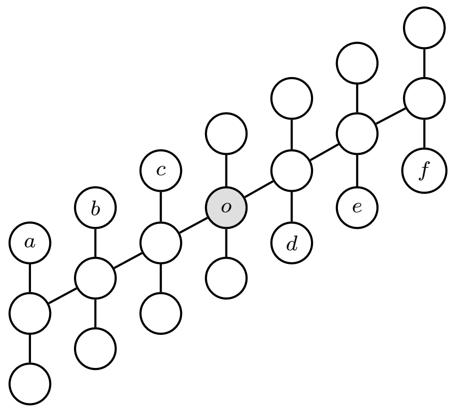

Figure 6: The central element o of the fourth chain (in grey) is less than seven elements and larger than seven others after 13 comparisons.

- Consider the case where thirteen elements are larger than than o, and seven are less. The thirteen elements contain three chains of length three, and may be sorted using $t_{13} - 3 = 13$ comparisons. The seven elements contain one chain of length three, and therefore, may be sorted with $t_7 - 1 = 5$ comparisons.
- Consider the case where twelve elements are larger than than o, and eight are less. If the twelve elements contain d, e and f, then they contain three chains of length three, and may be sorted using $t_{12} - 3 = 12$ comparisons. Otherwise, they contain a reversed v-chain composed of six elements (whose largest element is $\max\{a, b\}$), one chain of length three (whose largest element is c), one chain of length two, and two singletons. By using 2 comparisons to sort the singletons and the chain of length two, and applying Theorem 2.2(i-), it follows that the twelve elements once again can be sorted using $2 + (12 - 2) = 12$ comparisons.

  The eight elements less than o are shown in Figure 7.

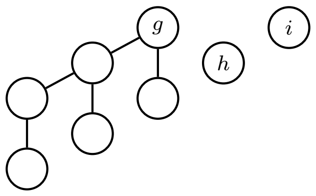

Figure 7: The eight elements less than o.

  Comparing g, h and i identifies the largest of all eight. The seven others from a reversed v-chain, a chain of length two and a singleton; Theorem 2.2(i-) ensures that they can be ordered using $7 - 2 = 5$ comparisons.

  In summary, 12 and 6 comparisons are required for each partition.
- Consider the case where eleven elements are larger than than o, and nine are less. At least one of the element among d, e or f is larger than o. Therefore, the partition of the eleven elements contains two chains of length three, and it can be sorted with $t_{11} - 2 = 11$ comparisons. The nine elements less than o are shown in Figure 8.

  Comparing j with the pairs $\{k, \ell\}$ and $\{m, n\}$ partitions the nine elements with respect to j. There can be any number between 3 to 7 elements less than j. In each case, Theorem 2.2(i-) guarantees that $2 + (7 - 2) = 7$ comparisons are enough to sort them.
- Finally, consider the case where ten elements are larger than than o, and ten are less. If at least one of the element among d, e or f is larger than o, then, the partition contains two chains of

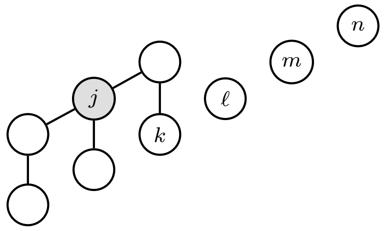

Figure 8: The nine elements less than o.

length three, and can be sorted with $t_{10} - 2 = 9$ comparisons. Otherwise, Figure 9 illustrates the ten elements larger than o (a and b form the chain of length two on the left).

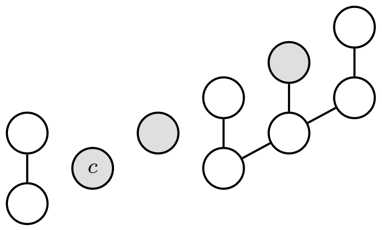

Figure 9: The ten elements larger than o.

Comparing the three shaded elements and applying Theorem 2.2(i-) ensures that $1 + (10 - 2) = 9$ comparisons are enough to sort them.

The analysis of the smaller partition is symmetrical.

In all four cases, the number of comparisons to sort the partitions is 18 ($13 + 5, 12 + 6, 11 + 7$ and $9 + 9$, respectively). Combining this with the 16 comparisons used to reach Figure 6 completes the proof. □

The previous proposition provides a heuristic method with $g_{21} = 34$.

4 Best known strategies to sort n elements

Table 1 summarizes the results of the six heuristics. The first column gives the number n of elements for all values up to 40 and for $n \in \{50, 100, 200, 500, 1000\}$. The next two columns give the number of comparisons α_n to insert a single element into a sorted list of $n - 1$ elements, and the number of comparisons β_n to insert a pair of elements into a sorted list of $n - 2$ elements. The six next columns show the result of the heuristics from the previous section. The second-to-last column ℓ_n gives the lower bound $\lceil \log_6(n!) \rceil$. The last column is the minimal value t_n returned by the six heuristics.

All heuristic values corresponding to the best-know solution appear in boldface. Boxed numbers correspond to situations where the corresponding heuristic was the only one to produce the best-known solution. The three rows at the end of the table indicate i- the number of times that each heuristic produced the best (including ties) solution for $4 \leq n \leq 40$; ii- the number of times that each heuristic produced the best (and no other heuristic did) solution for $4 \leq n \leq 40$; iii- the largest value of $n \leq 1000$ for which each heuristic produced the best solution.

For all values $23 \leq n \leq 1000$, the heuristic f_n always produce the best solution. Only the heuristic m_n produces values that are far from the best solution.

		Heuristic methods								
n	α_n	β_n	s_n	p_n	f_n	v_n	m_n	g_n	ℓ_n	t_n
4	2	2	**3**	**3**	**3**	**3**	–	–	2	3
5	2	3	5	**4**	5	**4**	–	–	3	4
6	2	3	6	6	6	6	$\boxed{\mathbf{5}}$	–	4	5
7	2	3	7	7	8	7	7	$\boxed{\mathbf{6}}$	5	6
8	2	5	**8**	10	9	9	**8**	–	6	8
9	2	5	10	11	11	10	10	$\boxed{\mathbf{9}}$	8	9
10	3	5	12	13	12	12	$\boxed{\mathbf{11}}$	–	9	11
11	3	5	14	14	14	$\boxed{\mathbf{13}}$	14	–	10	13
12	3	5	16	16	$\boxed{\mathbf{15}}$	16	16	–	12	15
13	3	5	18	18	18	18	18	$\boxed{\mathbf{16}}$	13	16
14	3	5	$\boxed{\mathbf{19}}$	20	20	20	20	–	15	19
15	3	5	22	21	22	22	23	$\boxed{\mathbf{20}}$	16	20
16	3	5	$\boxed{\mathbf{23}}$	24	24	25	25	–	18	23
17	3	5	26	$\boxed{\mathbf{25}}$	27	27	27	–	19	25
18	3	5	**28**	**28**	29	29	29	–	21	28
19	3	5	31	$\boxed{\mathbf{30}}$	31	31	32	–	22	30
20	3	7	**33**	35	**33**	34	34	–	24	33
21	3	7	36	37	36	36	36	$\boxed{\mathbf{34}}$	26	34
22	3	7	$\boxed{\mathbf{37}}$	40	38	38	38	–	28	37
23	3	7	**40**	41	**40**	**40**	41	–	29	40
24	3	7	43	44	$\boxed{\mathbf{42}}$	43	43	–	31	42
25	3	7	**45**	47	**45**	**45**	46	–	33	45
26	3	7	48	49	**47**	**47**	48	–	35	47
27	3	7	50	52	**49**	**49**	51	–	37	49
28	4	7	53	54	$\boxed{\mathbf{51}}$	52	53	–	38	51
29	4	7	55	56	**54**	**54**	56	–	40	54
30	4	7	58	58	$\boxed{\mathbf{56}}$	57	59	–	42	56
31	4	7	60	61	**59**	**59**	62	–	44	59
32	4	7	63	63	$\boxed{\mathbf{61}}$	62	64	–	46	61
33	4	7	65	66	**64**	**64**	67	–	48	64
34	4	7	68	68	$\boxed{\mathbf{66}}$	67	70	–	50	66
35	4	7	70	71	**69**	**69**	73	–	52	69
36	4	7	73	73	$\boxed{\mathbf{71}}$	72	75	–	54	71
37	4	7	75	76	$\boxed{\mathbf{74}}$	75	78	–	56	74
38	4	7	78	78	$\boxed{\mathbf{77}}$	78	81	–	58	77
39	4	7	81	81	**80**	**80**	84	–	60	80
40	4	7	84	84	$\boxed{\mathbf{82}}$	83	86	–	62	82
50	4	7	111	111	$\boxed{\mathbf{110}}$	111	118	–	83	110
100	5	9	276	276	$\boxed{\mathbf{274}}$	276	289	–	204	274
500	6	13	1943	1945	$\boxed{\mathbf{1941}}$	1944	2031	–	1458	1941
1000	7	13	4355	4355	$\boxed{\mathbf{4353}}$	4357	4593	–	3300	4353
Best occurences $n \leq 40$			9	5	21	11	3	5		
Unique best $n \leq 40$			3	2	10	1	2	5		
Largest best $n \leq 1000$			243	19	1000	83	10	21		

Table 1: Minimal number of comparisons to sort n elements using six heuristic methods

5 Discussion

This paper introduced a new class of sorting problems based on comparisons involving three elements at a time. Instances with n ranging from 6 to 15 were examined in detail by the author, and for each such n, a single heuristic emerged as the best performer, except in the case $n = 8$. Larger instances have not been studied as thoroughly, and many of the reported values of t_n are likely not optimal.

Establishing optimality is challenging even for relatively small n. Simple enumeration strategies are possible up to $n = 8$, but becomes computationally impractical by $n = 9$. Further research is required to confirm or disprove the optimality of the reported t_n values.

A natural generalization of the present work is to sort n elements using comparisons that simultaneously rank p elements. Pairwise comparisons correspond to $p = 2$, while the case studied here corresponds to $p = 3$. The analysis for larger p remains an open and potentially rich area for investigation.

References

[1] C. Audet. Ordering 15 marbles with a three-way scale. The Mathematical Gazette., 98(542):304–316, 2014.

[2] W. Cheng, X. Liu, G. Wang, and J. Liu. The results of s(15) and s(19) to minimum-comparison sorting problem. Journal of Frontiers of Computer Science and Technology, 1(3):305–313, 2007.

[3] L.R. Ford Jr. and S.M. Johnson. A tournament problem. The American Mathematical Monthly, 66(5):387–389, 1959.

[4] M. Peczarski. New results in minimum-comparison sorting. Algorithmica, 40:133–145, 2004.

[5] M. Peczarski. The Ford-Johnson algorithm still unbeaten for less than 47 elements. Information Processing Letters, 101(3):126–128, 2007.

[6] M. Peczarski. Towards optimal sorting of 16 elements. Acta Universitatis Sapientiae, Informatica, 4(2):215–224, 2012.