# Proposal: k-d Tree Algorithm for k-Point Matching

John R Hott

University of Virginia

## 1 Motivation

Every drug seeking FDA approval must go through Phase II and III clinical trial periods (trials on human participants) to determine its safety and effectiveness [1]. They are tested against a placebo and sometimes compared to the effectiveness of similar drugs. How can we guarantee that the data accurately depicts the effectiveness of the drug and not differences in physical traits–body type, age, gender, ethnicity, etc–of the patients? We would like to be able to match participants of roughly the same traits who are taking different drugs to determine each drug's effectiveness. Consider the following example. We would like to test Advil and Tylenol against a placebo to see which drug is the best pain reliever. For this study, we want to compare people of similar weight and age to test the effectiveness of each drug. Figure 1 shows a plot of sample patients based on age and weight.

This matching may be computed by hand, but is more conveniently calculated by an efficient algorithm. A naive approach for a matching algorithm would include matching all participants taking each drug, sorting them based on the quality of the match, then picking matches in order ensuring that no person is used twice. Assuming there are $n$ participants taking each drug, this will lead to $n^3$ total matches requiring $O(n^3)$ space complexity to store this data. If the trial contains 300 participants, that would be $100^3 = 1,000,000$ matches to store. As $n$ grows, current hardware becomes unable to store the data efficiently.

We plan to pursue an algorithm that performs as well as brute force in the worst case, but under a uniform density of data, will perform much better. First, let us formalize the problem.

## 2 Problem Statement

The original problem was given for 3 colors in 2-dimensions–3 drugs tested considering 2 physical traits. We propose here a generalized statement of the problem for an arbitrary number of dimensions $d$, and an arbitrary number of colors $k$. Let $K \subseteq \mathbb{R}^d$ be a disjoint union of $K_1, ..., K_k$. Define a match $m \subseteq K$ such that $|m \cap K_i| = 1, \forall K_i$. Then $M = \{m\}$ with $|M| = \prod_i |K_i|$ and $|K| = \sum_i |K_i|$. For this paper, let us assume $|K_1| = |K_2| = ... = |K_k| = n$, so that $|M| = n^k$.

To define the smallest match, let us define the match measure function

$$size : M \to \mathbb{R}.$$

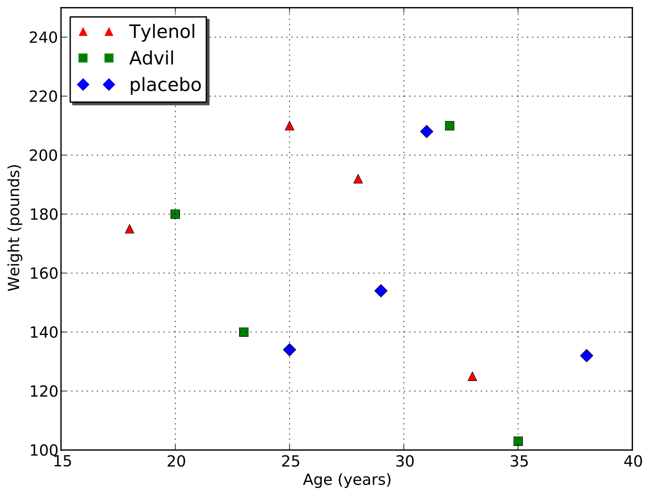

**Fig. 1.** Sample plot of patient weight verses age, where participants are taking either Advil, Tylenol, or a placebo drug.

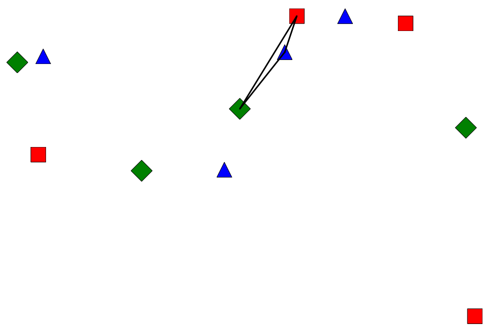

**Fig. 2.** Sample smallest match in 2 dimensions with 3 colors.

This function must not depend on the order of the points in the match, but give a consistent measurement of each match. For the case of 3 colors, let us define $K_1 = R$ (for red), $K_2 = G$ (for green), $K_3 = B$ (for blue), and $size(m)$ as the perimeter of the triangle $(r, g, b) \in R \times G \times B$. Figure 2 shows the smallest match in a grouping of points.

## 3 Background and Related Work

In order to improve upon the brute force implementation, we sought to reduce the number of matches considered for a given point. Around each point we only want to consider some small search radius for a match, therefore limiting the number of matches considered to those within that radius. For 3 colors, the intuition is to reduce a point's $n^2$-match search space to one that only contains a constant number of points. This leads us to a nearest neighbor search, which has been approached sucessfully with k-d trees [2]. Specifically, Samet in [3] shows that building the k-d tree structure with $n$ points requires $O(n \log n)$ time. More importantly to our discussion is the insertion and search time complexity, which Samet shows to be $O(\log_2 n)$ in the expected case. In the worst case, however, the running time is $O(dn^{1-1/d})$ for $d$ dimensions.

We propose leveraging k-d trees' $O(\log_2 n)$ expected case lookup cost to find points within a neighborhood, reducing the cost of creating a match for any given point to $O(\log_2 n + C)$ from $O(n^2)$, where $C$ is a constant number of matches considered in that search area.

## 4 Existing Algorithm

The brute force method described earlier for 3 colors included creating all $n^3$ matches, sorting them by perimeter, then saving the top $n$ matches in the list where the points have not been used before in another match is shown in Algorithm 1. This method requires $O(n^3 \log n)$ time in the worst case, since it requires the sorting step. The downfall of this approach, as discussed, is the $O(n^3)$ space complexity requirement, which limits the size of $n$ due to system space constraints.

An alternative brute force algorithm that reduces the space complexty to $O(n)$ increases the time complexity to $O(n^4)$. This algorithm loops over $k = 0 : n - 1$ to examine all $(n - k)^3$ matches left, only stores the smallest per iteration, and removes the used points from the pool. This approach can be seen in Algorithm 2.

Our first approach to beat these brute force algorithms loops over the $n$ red points, searching for nearest neighbor blue and green points. It then performs a nearest-neighbor search over the area within a $perimeter/2$ radius of the red point to determine if a smaller match exists for this point. It repeats this process until all $n$ matches are found. This algorithm utilizes k-d trees [2] to provide efficient nearest-neighbor lookups.

**Algorithm 1**: Brute Force Algorithm requiring $O(n^3)$ space, $O(n^3 \log n)$ time

```
Input: 3 sets of $n$ points
Output: set of $n$ ordered smallest matches of 3 points each

1  read input from file
2  foreach $red$ do
3      foreach $green$ do
4          foreach $blue$ do
5              $M \leftarrow m = \{r, g, b\}$
6  sort($M$)
7  foreach $M$ do
8      if $r_m$, $g_m$, $b_m$ clean then
9          $M_{ans} \leftarrow m$
10 return $M_{ans}$
```

**Algorithm 2**: Brute Force Algorithm requiring $O(n)$ space, $O(n^4)$ time

```
Input: 3 sets of $n$ points
Output: set of $n$ ordered smallest matches of 3 points each

1  read input from file
2  for $i = 1 : n$ do
3      $smallest = MAX$
4      $m_{smallest}$ = null
5      foreach $red$ do
6          foreach $green$ do
7              foreach $blue$ do
8                  if $size(m = \{r, g, b\} < smallest$ then
9                      $m_{smallest} = m$
10                     $smallest = size(m)$
11     $M_{ans} \leftarrow m_{smallest}$
12     remove $r_m, g_m, b_m$ from $red, green, blue$
13 return $M_{ans}$
```

## 5 Research Questions and Extensions

The purpose of this project is to extend and improve upon brute force and our first approach as defined above. We plan to do the following:

- Analyze the time complexity of the current k-d tree algorithm for both worst-case and expected case running times and prove its correctness,
- Examine other methods for defining the size of a match that are not dependent or limited by dimensionality, number of colors, or ordering of the points,
- Generalize the algorithm to an arbitrary $k$ colors in $d$ dimensions, as defined in the problem statement,
- Analyze the general algorithm under both worst-case and expected case running times and prove correctness.

## 6 Reading List

1. Jon Louis Bentley. 1975. Multidimensional binary search trees used for associative searching. Commun. ACM 18, 9 (September 1975), 509-517.
2. Hanan Samet. *Foundations of Multidimensional and Metric Data Structures*, chap. 1.5. Morgan Kaufmann Publishers, 500 Sansome Street, Suite 400, San Francisco, CA 94111. 2006.
3. Franco P. Preparata and Michael I. Shamos. 1985. Computational Geometry: an Introduction. chap. 3,5. Springer-Verlag New York, Inc., New York, NY, USA.
4. Cormen, Thomas H.; Leiserson, Charles E.; Rivest, Ronald L.; Stein, Clifford (2009). Introduction to Algorithms (3rd ed.). chaps. 2,4,12,15,22-26, and 35.2. MIT Press. ISBN 0-262-03384-4.
5. Lóvasz, László; M.D. Plummer (1986). Matching Theory. chap. 1 (Matchings in Bipartite Graphs). Akadémiai Kiadó, North Holland, Budapest (reprinted by AMS Chelsea Publishing, 2009).

## References

1. Blanchard Randall IV. The u.s. drug approval process: A primer. *Congressional Research Service*, June 1, 2001.
2. Andrew W. Moore. An intoductory tutorial on kd-trees, 1991.
3. Hanan Samet. *Foundations of Multidimensional and Metric Data Structures (The Morgan Kaufmann Series in Computer Graphics and Geometric Modeling).* Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2005.