Gift-wrapping for computing non-dominated points

Mu Tian [1][*] and Tianshu Bao [2][†]

May 6, 2015

Abstract

In this report we describe a Gift-wrapping algorithm for computing non-dominated points. We will demonstrate the $O(nh)$ complexity of this algorithm by simulation. Also, we developed a GUI animation tool for illustration of this algorithm using java. Both Mu Tian and Tianshu Bao have worked on some aspects of this project. In particular, Mu implemented the fundamental version of the Gift-wrapping algorithm and provided a simple animation, then Tianshu simplified the data structure of the algorithm, complete the entire GUI structure, and add in the mouse clicking input function. In addition, Tianshu prepared the presentation slides and Mu wrote this report.

[*] mu.tian@stonybrook.edu

[†] tianshu.bao@stonybrook.edu

Contents

1 Introduction

The term “Gift-wrapping” is mostly used to describe the algorithm developed by R. A. Jarvis[1] to compute the convex hull of a given set of points. This algorithm has $O(nh)$ time complexity where n is the number of points and h is the number of vertices on the convex hull. The fundamental idea of “Gift-wrapping” lies in its iteration step: given the current convex hull vertex p_i, we locate the next target p_{i+1}, counter clockwise such that no other points lies to the right of ray $p_i p_{i+1}$. Each of this step costs $O(n)$ because it essentially amounts to finding the maximum/minimum value according to some metric among $O(n)$ objects. Upon understanding this nature of “Gift-wrapping”, we can then implement the algorithm to find non-dominated points among n points in 2D. In next sections we will formally describe the algorithm, and then give a brief simulation to demonstrate the $O(nh)$ complexity. Java packages swing and awt offers comprehensive tools for animating the algorithm in a graphical interface. We will show our GUI finally as well.

2 Algorithm

We first define a relationship of two points: we say $p(p_x, p_y)$ is great than $q(q_x, q_y)$ in x_y, $(p > q)^x_y$ if $\{p_x > q_x\}$ OR $\{p_x = q_x$ AND $p_y > q_y\}$. We say $q = \text{argmax}^x_y S$ if no other point in S is greater than q in x_y. We say p is a candidate non-dominated point relative to q, $p \in N_d(q)$, if q is a non-dominated point and $p_x < q_x$ and $p_y > q_y$, where $N_d(q)$ is a set of all candidate non-dominated point relative to q.

The “Gift-wrapping” algorithm for non-dominated points is as follows.

Algorithm 1 Gift-wrapping for Non-dominated Points

```
1: **Input:** $S = \{p_1, ..., p_n\}$ a set of $n$ 2D points
2: **Output:** $S_N = \{q_1, ..., q_h\}$ a set of Non-dominated points from $S$
3: Initialize: $S_N \leftarrow \Phi$
4: Find the First Non-dominated point:
5:     $p \leftarrow \text{argmax}^x_y S$
6:     $S_N \leftarrow S_N \cup p$
7:     $S = S - \{p\}$
8: while $N_d(p) \neq \Phi$ do
9:     $p \leftarrow \text{argmax}^x_y N_d(p)$
10:    $S_N \leftarrow S_N \cup p$
11:    $S = S - \{p\}$
12: end while
```

An immediate inspection should give a $O(nh)$ time bound for this algorithm.

3 Simulation

We record the total number of steps for running the algorithm on various numbers of randomly generated points. The experimental results turned out to match the theoretical complexity.

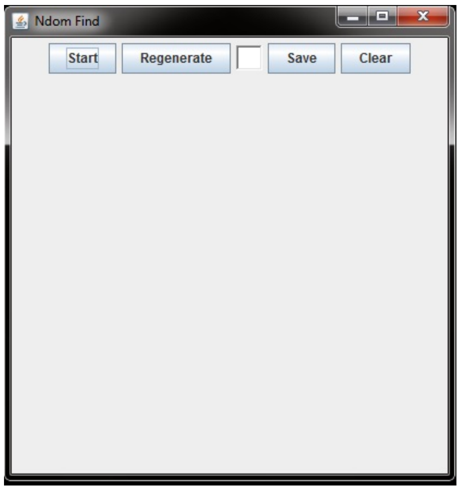
Table 1: Number of steps vs. number of points

n	10	20	40	80	160	320	640
h	4	5	5	4	1	4	9
steps	50	120	240	400	320	1600	6400

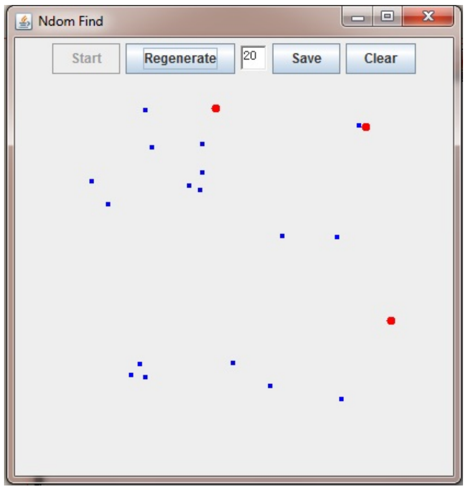
We can observe that steps$\sim O(nh)$, also, the non-dominated points remain sparse even when the total number of points increases.

4 Java GUI

The GUI is like below: we give the options to choose the number of points to be randomly sampled, as well as to input points by mouse clicking. Also, we allow combining randomly generated and mouse clicking input. As we start the algorithm, the non-dominated points will be found sequentially and labeled red color from right to left.



(a) 1



(b) 2

Figure 1: GUI