1 Algorithm Description

In this section, the algorithm is described. The population is being stored in an array of *Ranks*. Each *Rank* contains set of *Individuals* of a certain rank. This rank equals to the index of the corresponding *Rank* in the population array.

Algorithm 1 The function DetermineRank. It calculates rank of the new point p_n basing the ranks of points from $p \in P$ who dominate p_n

function DetermineRank(P, p_n)
    $R_n \leftarrow 0$
    for $p \in P$ **do**
        if $x_p \leq x_{p_n} \land y_p \leq y_{p_n}$ **then**
            $R_n \leftarrow max(R_n, Rg(p) + 1)$
        end if
    end for
    return R_n
end function

Algorithm 2 The procedure AddPoint. On each step it splits tree of current rank into two parts: points, that should change rank (C_i) and points that should not. Then points, that have changed their rank on the previous steps, are being added to the remainder. The proof is given in Theorem 1

procedure AddPoint(P, p_n)
    $R_n \leftarrow$ DetermineRank(P, p_n)
    if $p_n \in P[R_n]$ **then return**
    else
        $i \leftarrow 0$
        $p_0 \leftarrow p_n$
        $C_{-1} \leftarrow \{p_n\}$
        $C_0 \leftarrow \{p : Rg(p) = R_n \land p_n \prec p\}$
        while $|C_i| \neq 0$ **do**
            $P[R_n + i] \leftarrow$ CutTree($P[R_n + i], C_i$)
            $P[R_n + i] \leftarrow$ AddTree($P[R_n + i], C_{i-1}$)
            $p_{i+1} \leftarrow (\min c \in C_i x_c, \min c \in C_i y_c)$
            $i \leftarrow i + 1$
            $C_i \leftarrow \{p : Rg(p) = R_n + i \land p_i \prec p\}$
        end while
        $P[R_n + i] \leftarrow$ AddTree($P[R_n + i], C_{i-1}$)
    end if
end procedure

2 Proof

Lemma 1. *If:*

$$C = \{c : Rg(c) = R\}, \tag{1}$$

$$p_0 = (min_{c \in C} c_x; min_{c \in C} c_y), \tag{2}$$

$$\nexists p' : Rg(p') = R \quad and \quad x_{p'} \in [min_{c \in C} c_x; max_{c \in C} c_x]. \tag{3}$$

Then:

$$D_{p_0} = \{d : p_0 \prec d \wedge Rg(d) > R\} = D_C = \cup_{c \in C}\{d : c \prec d\} \tag{4}$$

Where D_e is a set of elements, dominated by e (or by any member of e, if this is a set).

Proof. 1. By definition of p_0,

$$p_0 \preceq c, \forall c \in C$$

That leads, by the transitivity of $\preceq$, to the following:

$$c \in C \prec d \Rightarrow p_0 \prec d$$

2. Let's suggest the following:

$$p_0 \prec d, \tag{5}$$

$$\nexists c \in C : c \prec d. \tag{6}$$

(5) can lead to the following cases:

1. $x_d = x_{p_0}$, $y_d > y_{p_0}$;

   According to (2),

   $$\exists c_1 \in C : x_{c_1} = x_{p_0}$$

   That means, that $x_{c_1} = x_d$. According to (6), $y_{c_1} >= y_d$. That means, that

   $$d \preceq c_1 \Rightarrow Rg(c_1) >= Rg(d) >^{(4)} R$$

   This contradicts with (1).

2. $x_d > x_{p_0}$, $y_d = y_{p_0}$;

   The proof is the same as for the previous case.

3. $x_d > x_{p_0}$, $y_d > y_{p_0}$;

   $$Rg(d) >^{(4)} R \Rightarrow \exists p \notin C : Rg(p) = R, p \prec d$$

   Let's state the following:

   $$c_1 : c_1 \in C, x_{c_1} = x_{p_0}$$

   $$c_2 : c_2 \in C, y_{c_1} = y_{p_0}$$

According to (3),

$$x_p < x_0 \vee y_p < y_0$$

which means that $x_p < x_{c_1}$, and, according to the definition of rank, $y_p > y_{c_1}$. But $x_d > x_{p_0} = x_{c_1}$. Therefore:

$$p \prec d \Rightarrow c_1 \prec d$$

It contradicts with (6). The similar proof is applicable to c_2.

$\square$

Let's define $Rg(p)$ as rank of p before point addition, and $Rg'(p)$ as rank of p after point addition. Let's define R_i:

$$F_i : \{\forall f \in F_i : Rg(f) = F_i, p_i \not\prec f\}$$

Theorem 1. *Point n was added. $Rg'(n) = R_0$. The following statements are applicable for any iteration of point addition algorithm:*

1. $\forall i > 0 : R_i = R_0 + i$;
2. $\exists p_i, C_i : \{\forall c \in C_i : Rg(c) = R_i, p_i \prec c\}$;
3. $Rg'(C_i) = R_i + 1$;
4. $Rg'(F_i) = R_i$;
5. $p_{i+1} = (min_{c\in C_i} c_x; min_{c\in C_i} c_y)$.

Proof. The proof will be by induction.

1. Base.
   (a) $p_0 = n$;
   (b) $\exists C_0 : \{\forall c \in C_0 : Rg(c) = R_0, n \prec c\}$;
   (c) $Rg'(C_0) = R_0 + 1 = R_i + 1$;
   (d) $Rg'(F_0)$ won't be changed;
   (e) $p_{i+1} = (min_{c\in C_0} c_x; min_{c\in C_0} c_y)$.
2. Induction step.
   (a) $p_{i+1} = (min_{c\in C_i} c_x; min_{c\in C_i} c_y)$;
   (b) $Rg'(C_i) = R_i \Rightarrow^{lemma1} \forall d : Rg(d) = R_i, p_i \prec d : Rg'(d) = Rg(C_i) + 1 = R_i + 1$;
   (c) $Rg'(F_i)$ won't be changed.

$\square$