Algorithm 1: Non-dominated Sorting

Input : list $P[N, C_{rate}, E]$
Output: list F

```
for each p ∈ P do
    S_p = ∅
    n_p = 0
    for each q ∈ P do
        if q[C_rate] < p[C_rate] then
            S_p = S_p ∪ q
        else if q[C_rate] = p[C_rate] then
            if (q[N] < p[N] and q[E] < p[E]) or (q[N] < p[N] and
              q[E] = p[E]) or (q[N] = p[N] and q[E] < p[E]) then
                S_p = S_p ∪ q
            else
                n_p = n_p + 1
            end if
        else
            n_p = n_p + 1
        end if
    end for
end for
if n_p = 0 then
    p_rank = 1
    F_1 = F_1 ∪ p
end if
i = 1
while F_i ≠ 0 do
    Q = ∅
    for each q ∈ S_p do
        n_q = n_q − 1
        if n_q = 0 then
            q_rank = i + 1
            Q = Q ∪ q
        end if
        i = i + 1
        F_i = Q
    end for
end while
return F
```

Algorithm 2: Crowding distance

Input : list $P[N, C_{rate}, E], F$
Output: list $P_{distance}$

$l = |P|$
for *each* $i \in P$ **do**
  $P[i]_{distance} = 0$
end for
for *each object* o **do**
  $P = sort(P, o)$
  $P[0]_{distance} = P[l-1]_{distance} = \infty$
  for $i = 1$ *to* $(l-1)$ **do**
    $P[i]_{distance} = P[i]_{distance} + (P[i+1]_o - P[i-1]_o)/(P_o^{max} - P_o^{min})$
  end for
end for
return $P_{distance}$

Algorithm 3: modified NSGA-II

Input : $population, r_c, r_m, iter$
Output: $final$

$i = 1$
$parents = Initation(population)$
while $i \neq iter$ **do**
  $offspring = Crossover(parents, r_c)$
  $offspring = Mutation(offspring, r_m)$
  $family = parents + offspring$
  $P[N, C_{rate}, E] = Evaluation(family)$
  $F = NondominatedSorting(P)$
  $CrowdingDistance(F, P)$
  Select the first 50% of chromosomes as the next generation after CrowdingDistance
  $i = i + 1$
end while
Choose the smallest N as the final answer
return $final$