Algorithm 1 NEH Heuristic

Calculate the total processing time of each job as $S[1, \cdots n]$
Sort $S[1, \cdots n]$ in the descending order

Require: $n \geq 0 \vee x \neq 0$
Ensure: $y = x^n$
  $y \leftarrow 1$
  if $n < 0$ **then**
    $X \leftarrow 1/x$
    $N \leftarrow -n$
  else
    $X \leftarrow x$
    $N \leftarrow n$
  end if
  while $N \neq 0$ **do**
    if N is even **then**
      $X \leftarrow X \times X$
      $N \leftarrow N/2$
    else $\{N$ is odd$\}$
      $y \leftarrow y \times X$
      $N \leftarrow N - 1$
    end if
  end while

Algorithm 2 NEH Heuristic

Calculate the total processing time of each job as $S[1, \cdots n]$
Sort $S[1, \cdots n]$ in the descending order

Require: $n \geq 0 \vee x \neq 0$
Ensure: $y = x^n$
  $y \leftarrow 1$
  if $n < 0$ **then**
    $X \leftarrow 1/x$
    $N \leftarrow -n$
  else
    $X \leftarrow x$
    $N \leftarrow n$
  end if
  while $N \neq 0$ **do**
    if N is even **then**
      $X \leftarrow X \times X$
      $N \leftarrow N/2$
    else {N is odd}
      $y \leftarrow y \times X$
      $N \leftarrow N - 1$
    end if
  end while

Algorithm 3 Hill Climbing

  π := NEH_Heuristic
  π := Local_Search(π)
  iter := 0
  while iter < MAX_ITERATIONS **do**
    i, j := Random_Integers$(0, n)$
    Swap (π_i, π_j)
    π' := Local_Search(π)
    if $C_{\pi'} < C_\pi$ **then**
      $\pi := \pi'$
    end if
    iter := iter + 1
  end while

Algorithm 4 Simulated Annealing

```
π := NEH_Heuristic
π := Local_Search(π)
iter := 0
temperature := Osman_Initialization(π)
while iter < MAX_ITERATIONS do
  i, j := Random_Integers(0, n)
  Swap (π_i, π_j)
  π' := Local_Search(π)
  diff := C_π' − C_π
  acceptance_criterion := exp(−diff/temperature)
  probablity := Random_Real (0, 1)
  if C_π' < C_π then
    π := π'
  else if probablity < acceptance_criterion then
    π := π'
  end if
  iter := iter + 1
  if iter % ITER_PER_EPOCH == 0 then
    temperature := temperature * annealing_rate
  end if
end while
```

Algorithm 5 Genetic Algorithm

```
population := Initialize_Chromes()
population := Local_Search (population)
generation := 0
while generation < MAX_GENERATION do
  new_population := Select_Chromes(population)
  new_population := CrossOver_Chromes(new_population)
  new_population := Mute_Chromes (new_population)
  new_population := Local_Search (new_population)
  if max (C_new_population) < max (C_population) then
    population := new_population
  end if
  generation := generation + 1
end while
```