I. PROBLEM PARAMETERS

$\boldsymbol{n}$	The total number of fruit trees in the orchard
$\boldsymbol{t}$	Number of mature fruit trees, that is, the total number of subtasks
$\boldsymbol{Ratio}$	The maturity rate of fruit trees
$\boldsymbol{y}$	Coordinates of mature fruit trees
$\boldsymbol{Yield}$	The yield of mature fruit trees, that is, the task amount of subtasks
$\boldsymbol{s}$	$= \{1, 2, ..., t\}$. Serial number of mature fruit trees
$\boldsymbol{r}$	The number of robots
$\boldsymbol{Cr}$	Congestion degree in the current environment
$\boldsymbol{LEN, WID}$	The length and width of the orchard
$\boldsymbol{len, wid}$	The length and width of the robot model used in this paper are 1.5 and 1 respectively
$\boldsymbol{Wm}$	Maximum load of a robot, which is 300 kilograms (kg) in this paper
$\boldsymbol{w}$	Robot's load
$\boldsymbol{Sw}$	Self-weight of a robot
$\boldsymbol{Dp}$	The driving power of the robot, whose maximum is 2000 watts in this paper
$\boldsymbol{v}$	Average moving speed of a robot
$\boldsymbol{T}$	A task, which consists of multiple subtasks
$\boldsymbol{TS}$	A task set, which consists of multiple tasks
$\boldsymbol{AT}$	A collection of all tasks
$\boldsymbol{Tp}$	Time of harvesting preprocessing of subtasks
$\boldsymbol{Wt}$	The task amount of a task
$\boldsymbol{Pr}$	The proportion of a subtask's completion in a task to its total task amount
$\boldsymbol{k}$	The number of subtasks in a task
$\boldsymbol{N}$	Number of tasks in a TS
$\boldsymbol{TN}$	Total number of tasks
$\boldsymbol{time}$	$= \{time_1, time_2, ..., time_{TN}\}$. The task duration to complete tasks
$\boldsymbol{TIME}$	$= \{TIME_1, TIME_2, ..., TIME_r\}$. The TS duration to complete TSs
$\boldsymbol{Na}$	Number of adjacent fruit trees, whose maximum value is 4
$\boldsymbol{i, j}$	Index

II. Pseudocode of algorithms

Algorithm 2 Initial solution generation

Input: $Yield$, Wm, y, r, cr, $Pnum$
Output: AT, FIT
1: $AT \leftarrow \emptyset$
2: **for** $i = 1 \rightarrow r - 1$ **do**
3: $tempr = r - i$
4: $Cluster_i \leftarrow$ **Kmeans**$(y, tempr)$ // The mature fruit trees are clustered into $tempr$ clusters, and a cluster $Cluster_i$ closest to the warehouse is obtained
5: $[rem, [T_1, T_2, ..., T_k]] \leftarrow Cluster_i$ // rem is the remaining subtasks, whose task weight is much less than Wm. It is assumed that the subtasks in the $Cluster_i$ can be divided into k tasks except rem
6: Update y with y, $Cluster_i$, and rem // Recalculate the subtasks that are not assigned
7: $AT \leftarrow AT \cup [T_1, T_2, ..., T_k]$ // Update AT
8: **end for**

Algorithm 3 Proposed SSA

Input: AT, $Yield$, Wm, y, cr, $Pnum$
Output: AT
1: $[T_1, T_2, ..., T_s] \leftarrow AT$ // Suppose a solution contains s tasks
2: $i = 1$
3: **for** $i = 1 \rightarrow s$ **do**
4: $Seq_1 \leftarrow T_i$ // Get the subtask sequence
5: Randomly generate Seq_2 // Randomly generated sequence
6: $Newpop \leftarrow Seq_1 \cup Seq_2$
7: $Pop \leftarrow Newpop$
8: **while** Size of $Pop \leq Pnum$ **do**
9: $gbest \leftarrow$ **Select**(Pop) // Select the sequence with the shortest task duration
10: $Newpop \leftarrow$ **IDGA1**$(gbest, Newpop)$ // Generate new sequences
11: $Pop \leftarrow Pop \cup Newpop$ // Store the newly generated sequences in the population
12: **end while**
13: $gbest \leftarrow$ **Select**(Pop)
14: $T_i \leftarrow gbest$ // Update subtask sequence
15: **end for**
16: $AT \leftarrow [T_1, T_2, ..., T_s]$ // Update the solution
17: Return AT

Algorithm 4 Proposed TBM

Input: AT, $Yield$, Wm, y, r, Cr, $Pnum$, TN, difference threshold THD, etc
Output: $Archive$
1: $z \leftarrow$ **max**(ceil(sum($Yield$)/Wm), ceil(TN/r)*r) // Get the number of unoccupied tasks, where ceil(*) means round up and sum(*) means add together
2: $AT \leftarrow [T_1, ..., T_{TN}, T_{TN+1}, ..., T_{TN+z}]$ //Suppose the solution AT contains TN tasks and add z empty tasks to it
3: $[MTC, LTC, thd] \leftarrow AT$ // Choose the most time-consuming task and the most time-saving task from AT, and get the time difference thd
4: $archive \leftarrow AT$ // Initialize $archive$
5: **while** $thd \geq THD$ or overloaded tasks exist **do**
6: $[MTC, LTC] \leftarrow$ **exchange**(MTC, LTC) // Algorithm 5
7: $AT \leftarrow [AT, MTC, LTC]$ // Update AT
8: $archive \leftarrow archive \cup AT$
9: $[MTC, LTC, thd] \leftarrow AT$ // Update MTC, LTC, and thd
10: **end while**
11: $Archive \leftarrow$ **Selection**$(archive, Pnum)$ // Screen out $Pnum$ solutions with the minimum TEC

Algorithm 5 Proposed exchange operator

Input: MTC, LTC
Output: $TempC, TempS$
/*Transfer subtasks from MTC header***/**
$tt_1 =$ Inf // Initialize total time tt_1
$tt_2 \leftarrow [MTC, LTC]$ // Get total time tt_2 of two tasks
$[TempC_1, TempS_1] = [MTC, LTC]$
while $tt_2 < tt_1$ **do**
  $[TC_1, TS_1] = [TempC_1, TempS_1]$ //Update TC_1 and TS_1
  $tt_1 = tt_2$ // Update tt_1
  $[TempC_1, TempS_1] \leftarrow$ **Transfer1**(TC_1, TS_1) // Gradually transfer subtasks from TC_1 **header** to TS_1
  $tt_2 \leftarrow [TempC_1, TempS_1]$ // Update tt_2
end while
/*Transfer subtasks from MTC tail***/**
$tt_1 =$ Inf // Initialize tt_1
$tt_2 \leftarrow [MTC, LTC]$ // Initialize $tt2$
$[TempC_2, TempS_2] = [MTC, LTC]$
while $tt_2 < tt_1$ **do**
  $[TC_2, TS_2] = [TempC_2, TempS_2]$
  $tt_1 = tt_2$
  $[TempC_2, TempS_2] \leftarrow$ **Transfer2**(TC_2, TS_2) // Gradually transfer subtasks from TC_2 **tail** to TS_2
  $tt_2 \leftarrow [TempC_2, TempS_2]$ // Update tt_2
end while
$[TempC, TempS] \leftarrow$ **Select**$([TC_1, TS_1], [TC_2, TS_2])$ // Filter out the group with the shortest total task duration as output

Algorithm 6 Proposed NTOM

Input: AT, $Yield$, Wm, y, r, cr, $Pnum$, THD, etc
Output: $Archive$
$[Cor] \leftarrow$ **Locate**(AT, Y) // Get the center coordinates, Cor, of each task in the current solution
if rand $\leq \frac{1}{3}$ **then**
  $op_1 \leftarrow MTC$
else if $\frac{2}{3} \leq$ rand $\leq \frac{2}{3}$ **then**
  $op_1 \leftarrow LTC$
else
  Random select a task as op_1
end if// Initialize op_1
$Cor_{op_1} \leftarrow [Cor, op_1]$ // Get the center coordinate of op_1, Cor_{op_1}
$sort1 \leftarrow [Cor, Cor_{op_1}]$ // Get the ranking of the distance from each center coordinate to the op_1 center coordinate
$sort2 \leftarrow AT$ // Get the ranking of task duration of each task
$sort \leftarrow$ **sum**$(sort1, sort2)$ // Get the comprehensive ranking of each task
$op_2 \leftarrow AT(sort)$ // Select the task with the first comprehensive ranking, op_2
$thd \leftarrow [op_1, op_2]$ // Get the time difference thd
while $thd \geq THD$ **do**
  $[TempF, TempS] \leftarrow$ **KGLS**(op_1, op_2) // Knowledge-guided local search operation
  $AT \leftarrow [AT, TempF, TempS]$ // Update AT
  $archive \leftarrow archive \cup AT$
  $[op_1, op_2, thd] \leftarrow AT$ // Update op_1, op_2, and thd
end while
$Archive \leftarrow$ **Selection**$(archive, Pnum)$ // Screen out $Pnum$ solutions with the minimum TEC

Algorithm 7 Proposed PTAM

Input: $Yield, archive, r, Pnum$
Output: $ARCHIVE$, FIT

$d \leftarrow archive$ // Suppose the archive contains d solutions
$ARCHIVE \leftarrow \emptyset$ // Initialize the output archive
$FIT \leftarrow \emptyset$ // Initialize the fitness value corresponding to $ARCHIVE$
for $i = 1 \rightarrow d$ **do**
    $AT \leftarrow archive(i)$
    $[T_1, T_2, ..., T_{TN}] \leftarrow AT$ // Suppose a solution contains TN tasks
    $[seq1, seq2] =$ **randi**(TN) // Randomly generate two sequences from 1 to TN, which are used to disrupt the tasks in AT
    $[sol1, TIME1] \leftarrow$ **IDGA2**$([T_1, T_2, ..., T_{TN}], idx_1, r)$ // Using IDGA2 to generate the task assignment result $sol1$. And the task duration vector $TIME1$ of the robot is obtained
    $fit1 \leftarrow$ **max**$(TIME1)$ // Get the MCT value of $sol1$
    $[sol2, TIME2] \leftarrow$ **IDGA2** $([T_1, T_2, ..., T_{TN}], seq2, r)$
    $fit2 \leftarrow$ **max**$(TIME2)$
    $Newpop \leftarrow seq1 \cup seq2$
    $Pop \leftarrow Newpop$
    $Fit \leftarrow fit1 \cup fit2$
    while Size of $Pop \leq Pnum$ **do**
        $gbest \leftarrow$ **Select**(Pop, Fit) // Select the sequence with the smallest MCT
        $Newpop \leftarrow$ **IDGA2**$(gbest, Newpop)$ // Generate new sequences
        $[sol, TIME] \leftarrow$ **IDGA2**$([T_1, T_2, ..., T_{TN}], Newpop, r)$
        $fit \leftarrow$ **max**$(TIME)$
        $Pop \leftarrow Pop \cup Newpop$ // Store the newly generated sequences in the population
        $Fit \leftarrow Fit \cup fit$
    end while
    $[sol, TIME] \leftarrow$ **IDGA2**$([T_1, T_2, ..., T_{TN}], gbest, r)$
    $[ARCHIVE, FIT] \leftarrow [ARCHIVE, FIT] \cup [sol,$ **[sum**$(TIME)$, **max**$(TIME)]]$ // Calculate TEC and MCT. Update the output archive and its fitness value
end for

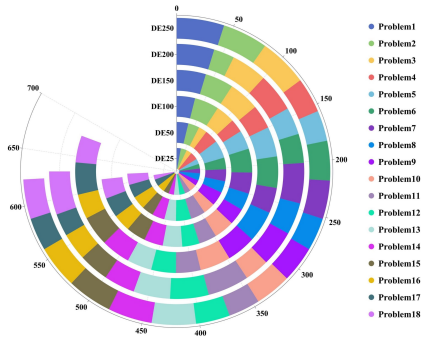
III. Parameter validation

This section calibrated the main parameters of the RNSGA algorithm. The parameter under validation is the THD, which directly impacts the termination criteria of the TBM and NTOM, thereby affecting the search process and efficiency of the entire algorithm. Based on past experimental results and empirical analysis, we carefully determine the range of THD values: $25, 50, 100, 150, 200, 250$. Each of these six parameter values corresponds to a variant of the algorithm THD25, THD50, THD100, THD150, THD200, THD250, with all variants identical except for the THD parameter. Each variant underwent comparative experiments according to the experimental settings outlined in the first section.

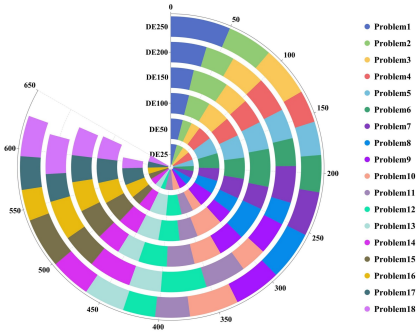
To visually assess which parameter is most suitable for the algorithm, we first compute the HV and IGD values for each of the six variants across 180 test instances. Subsequently, we obtain the rankings of each variant for these metrics in each test instance. Finally, we aggregate the rankings of each variant across ten test instances for each test problem.

Fig. 1 presents the statistical results of each variant in a runway format, with each test problem represented by a different color. A shorter runway indicates a higher overall ranking, suggesting that the corresponding variant's parameter is more applicable. Overall, the differences in rankings between each variant are not significant, indicating that the value of parameter THD do not extremely impact the algorithm. Considering both Fig. 1(a) and Fig. 1(b), THD50 shows relatively higher overall rankings; therefore, we select $THD = 50$ as the final parameter setting.

In addition, in order to reduce the complexity of calculation, some parameters are set to fixed values. For example, the P value in Eq. 4 and the POW value in Eq. 6 are both set to 1. The maximum load Wm of the robot is set to 300.



(a) The ranking of each variant on IGD



(b) The ranking of each variant on HV

Fig. 1. Sum of rankings for different THD values

IV. Verification of RNSGA components

In the RNSGA algorithm, several important components include the solution initialization methods, SSA, TBM, NTOM, and PTAM. To enhance the clarity of the paper, these components are verified and analyzed together in this section. The paper proposes five initialization methods: underloaded initialization, uniform initialization, fully loaded initialization, overloaded initialization, and random initialization. These correspond to variants V1, V2, V3, V4, and V5, respectively. Experimental results show that V1 performs the best, as illustrated in Fig. 2. Therefore, the underloaded initialization method is used in both the RNSGA and subsequent comparative experiments. Variant V6 corresponds to the RNSGA without SSA, used to validate the effectiveness of SSA. Variants V7 and V8 correspond to the RNSGA without TBM and NTOM, respectively, to verify the effectiveness of these two mechanisms. Finally, the variant V9 replaces the PTAM with a random task allocation method to verify the effectiveness of PTAM.

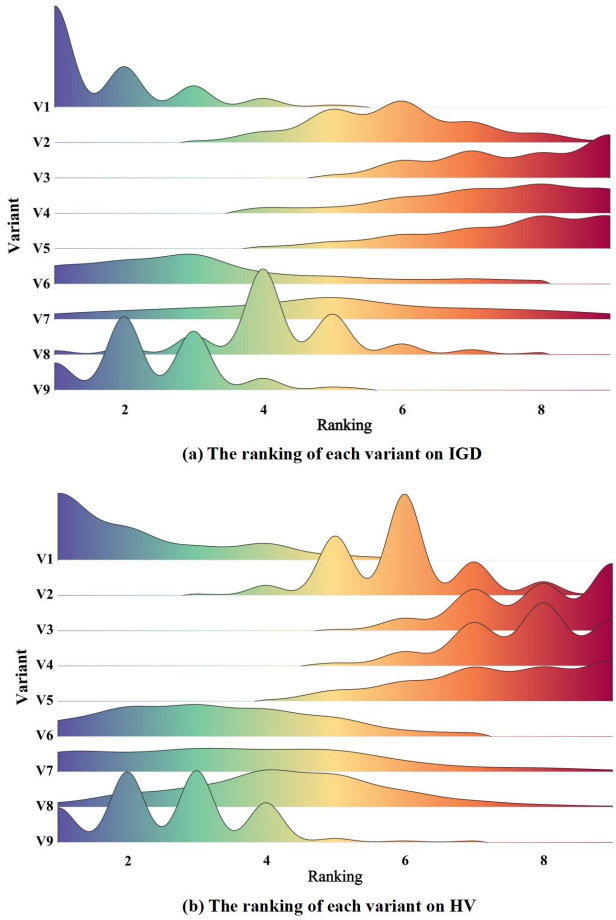


Fig. 2. Verification of RNSGA components

The figure shows the approximate ranking distribution of each variant across all test instances in the form of peaks. Whether considering the IGD and HV indicators, V1 consistently ranks first more frequently than the other variants. Among the initialization methods, V2's uniform initialization results in each task's load being close to the average load during initialization. This results in a large number of TN, causing subsequent TBM to be ineffective and the initialization solution to be deemed unnecessary for further optimization. Thus, the initialized result is output directly. V3's fully loaded initialization results in all tasks being fully loaded except for the last task. This solution completes all tasks with fewer TN. Even with the addition of some empty tasks in TBM to distribute the load, most tasks remain fully loaded. The same applies to V4's overloaded initialization method. These tasks' slow movement speeds significantly impact task duration, and this effect becomes more pronounced as the map size increases, leading to lower rankings. The random initialization of V5 can be adjusted greatly through TBM and NTOM, but the large number of fully loaded and overloaded tasks still makes complete optimization difficult.

In summary, with a fixed $Yield$, the $time$ robots spend on harvesting remains constant. According to Eq. 4, the significant factor affecting the objective value is the duration all robots spend moving. Therefore, it is necessary to balance the number of trips each robot makes and their speed for each trip. In V1's results, the speed of each task is not lowered immensely, and the number of tasks is well-controlled. Most importantly, the results allow TBM and NTOM to function effectively. Hence, the underloaded initialization method is more suitable for RNSGA.

As shown in the figure, the results of variants V6-V9 rank higher overall compared to variants V2-V5, demonstrating the significant impact of the initialization method. However, there is still a considerable gap in ranking distribution compared to V1. Therefore, SSA, TBM, NTOM, and PTAM play crucial and irreplaceable roles in RNSGA. SSA is used to adjust the sequence of subtasks within each task, ensuring smooth transitions between subtasks and reducing unnecessary travel time. A good subtask order can also reduce harvesting preprocessing time. TBM relieves the load of heavily loaded tasks by filling empty tasks, reducing TEC through reasonable subtask transfers, and balancing the load across tasks. NTOM further improves solution quality by optimizing the task structure between adjacent tasks, preventing excessive dispersion of subtasks. PTAM offers greater robustness compared to random task allocation, using IDGA2 to guide the search toward better allocation solutions.

V. Results on more problems

To validate the robustness of RNSGA, we conduct further tests on the model established by Dai et al. [1]. The termination for all problems are the maximum elapsed CPU time for $n \times 0.05$ seconds. Considering the different characteristics of the model, we remove the premature termination for optimization in RNSGA. This allows RNSGA to further refine solutions even for small-scale problems. The experimental results are presented in Table I.

From the experimental results, RNSGA demonstrated strong robustness and superiority, particularly in terms of HV results, where it significantly outperformed other algorithms. However, due to the limited number of tasks in this test problem, it was challenging to fully showcase RNSGA's capability in solving large-scale problems. Furthermore, setting the total distance traveled by the robots as an optimization objective is less meaningful than considering the total energy consumption of the robots. This also validates the reasonableness of the model proposed in this paper.

TABLE I
Test results on more prolems

IGD							
Algorithm/Problem	**AMOEA**	**CDABC**	**MODABC**	**MOEA/D**	**NSGA-II**	**MOTS**	**RNSGA**
50×4	1.2543e+02 (4.31e+01) -	**4.6131e+01 (2.73e+01) =**	4.7436e+01 (2.12e+01) =	5.0417e+02 (5.24e+01) -	1.9322e+02 (5.05e+01) -	2.5470e+02 (3.58e+01) -	5.7275e+01 (3.86e+01)
50×5	1.5417e+02 (3.63e+01) -	9.0770e+01 (3.21e+01) =	9.8736e+01 (3.08e+01) =	6.4788e+02 (8.50e+01) -	2.5606e+02 (4.82e+01) -	3.6054e+02 (3.51e+01) -	**7.7126e+01 (3.73e+01)**
50×6	2.1189e+02 (7.61e+01) -	1.4456e+02 (4.58e+01) =	1.5242e+02 (4.59e+01) =	7.1931e+02 (1.06e+02) -	3.7920e+02 (5.97e+01) -	4.6988e+02 (3.94e+01) -	**1.2934e+02 (3.96e+01)**
50×7	1.7983e+02 (6.32e+01) -	**9.3208e+01 (3.00e+01) =**	9.7906e+01 (3.37e+01) =	7.3334e+02 (1.12e+02) -	4.1102e+02 (6.46e+01) -	4.9649e+02 (6.42e+01) -	9.5239e+01 (6.30e+01)
60×4	1.9249e+02 (5.17e+01) -	**1.1334e+02 (4.30e+01) =**	1.1342e+02 (4.50e+01) =	7.7069e+02 (8.93e+01) -	4.6797e+02 (5.61e+01) -	6.1879e+02 (4.45e+01) -	1.2773e+02 (5.32e+01)
60×5	2.7236e+02 (7.29e+01) -	1.3350e+02 (3.82e+01) =	1.4714e+02 (4.74e+01) -	8.6292e+02 (7.58e+01) -	5.7144e+02 (6.67e+01) -	7.0623e+02 (4.93e+01) -	**1.0041e+02 (4.26e+01)**
60×6	2.5928e+02 (4.10e+01) -	1.1542e+02 (2.60e+01) =	1.5299e+02 (3.22e+01) -	5.5922e+02 (6.17e+01) -	2.5565e+02 (2.85e+01) -	2.9204e+02 (4.33e+01) -	**1.0383e+02 (1.64e+01)**
60×7	3.4747e+02 (3.67e+01) -	**1.7603e+02 (3.70e+01) =**	1.8449e+02 (4.00e+01) =	7.0518e+02 (8.59e+01) -	3.4358e+02 (5.92e+01) -	3.6330e+02 (5.57e+01) -	1.8347e+02 (1.81e+01)
70×4	3.6400e+02 (3.61e+01) -	2.1474e+02 (2.74e+01) =	2.1488e+02 (2.73e+01) =	7.5686e+02 (8.95e+01) -	3.7530e+02 (4.29e+01) -	4.6257e+02 (4.96e+01) -	**2.0075e+02 (1.51e+01)**
70×5	4.5231e+02 (4.36e+01) -	1.5298e+02 (2.90e+01) =	1.5993e+02 (2.83e+01) =	8.8857e+02 (8.08e+01) -	4.8285e+02 (6.07e+01) -	5.5227e+02 (3.81e+01) -	**1.4269e+02 (3.33e+01)**
70×6	4.2525e+02 (6.85e+01) -	**2.6029e+02 (6.26e+01) =**	2.6568e+02 (6.82e+01) =	1.0834e+03 (6.24e+01) -	7.0370e+02 (5.89e+01) -	8.0064e+02 (4.92e+01) -	2.7461e+02 (4.85e+01)
70×7	4.9521e+02 (5.85e+01) -	3.2363e+02 (4.09e+01) =	3.2582e+02 (4.67e+01) =	1.1298e+03 (1.02e+02) -	8.1028e+02 (5.95e+01) -	9.3588e+02 (5.84e+01) -	**2.9760e+02 (5.27e+01)**
80×4	4.4620e+02 (6.43e+01) -	1.3886e+02 (3.11e+01) =	1.5132e+02 (3.04e+01) =	5.4513e+02 (6.39e+01) -	2.8074e+02 (7.27e+01) -	2.7844e+02 (6.18e+01) -	**1.4500e+02 (2.40e+01)**
80×5	4.5917e+02 (7.18e+01) -	2.0828e+02 (3.16e+01) =	2.2187e+02 (3.94e+01) =	7.2534e+02 (9.91e+01) -	4.3192e+02 (5.37e+01) -	4.7508e+02 (6.52e+01) -	**1.8998e+02 (3.53e+01)**
80×6	5.8631e+02 (3.54e+01) -	2.1493e+02 (2.77e+01) =	2.2711e+02 (2.74e+01) =	8.4912e+02 (5.58e+01) -	4.1334e+02 (4.73e+01) -	4.4977e+02 (5.15e+01) -	**1.9184e+02 (1.64e+01)**
80×7	6.6698e+02 (4.60e+01) -	1.6059e+02 (3.48e+01) =	1.9211e+02 (4.09e+01) -	9.0280e+02 (8.57e+01) -	4.7598e+02 (5.80e+01) -	5.2658e+02 (2.73e+01) -	**1.4845e+02 (3.95e+01)**
100×4	6.1722e+02 (5.54e+01) -	3.6515e+02 (4.41e+01) =	3.7744e+02 (4.02e+01) =	1.2267e+03 (1.22e+02) -	7.8514e+02 (5.07e+01) -	8.6911e+02 (6.40e+01) -	**3.4877e+02 (6.24e+01)**
100×5	8.1815e+02 (5.62e+01) -	**2.7269e+02 (4.43e+01) =**	2.8008e+02 (3.79e+01) =	1.2749e+03 (1.06e+02) -	8.2017e+02 (6.57e+01) -	8.9516e+02 (5.20e+01) -	2.9220e+02 (5.66e+01)
100×6	4.8977e+02 (3.97e+01) -	2.1104e+02 (2.68e+01) -	2.3394e+02 (3.01e+01) -	6.1009e+02 (1.14e+02) -	3.1515e+02 (4.88e+01) -	3.3203e+02 (5.10e+01) -	**1.4427e+02 (2.12e+01)**
100×7	6.2603e+02 (5.44e+01) -	1.9727e+02 (3.31e+01) =	2.0682e+02 (3.75e+01) =	7.3981e+02 (7.57e+01) -	3.9062e+02 (6.58e+01) -	4.1746e+02 (5.84e+01) -	**1.8641e+02 (2.32e+01)**
120×4	5.3799e+02 (2.36e+01) -	**3.0923e+02 (3.79e+01) =**	3.1541e+02 (2.89e+01) =	9.1745e+02 (6.61e+01) -	5.3733e+02 (5.43e+01) -	5.8857e+02 (2.82e+01) -	3.1278e+02 (2.20e+01)
120×5	5.8426e+02 (4.50e+01) -	**3.8821e+02 (2.24e+01) =**	4.0111e+02 (2.70e+01) =	1.0920e+03 (8.25e+01) -	6.9216e+02 (4.88e+01) -	7.4782e+02 (4.25e+01) -	4.0086e+02 (1.56e+01)
120×6	9.8342e+02 (6.94e+01) -	**2.3998e+02 (4.62e+01) =**	2.5008e+02 (4.94e+01) =	1.1311e+03 (9.84e+01) -	6.6944e+02 (8.87e+01) -	7.6243e+02 (6.73e+01) -	2.8362e+02 (5.09e+01)
120×7	**4.8222e+02 (5.99e+01) +**	8.6627e+02 (5.07e+01) =	8.6460e+02 (5.79e+01) =	1.8556e+03 (8.36e+01) -	1.3813e+03 (5.91e+01) -	1.4764e+03 (5.74e+01) -	8.4512e+02 (4.38e+01)
+/-/=	**1/23/0**	**0/1/23**	**0/4/20**	**0/24/0**	**0/24/0**	**0/24/0**	

HV							
Algorithm/Problem	**AMOEA**	**CDABC**	**MODABC**	**MOEA/D**	**NSGA-II**	**MOTS**	**RNSGA**
50×4	1.2608e-01 (3.08e-02) -	1.8552e-01 (1.47e-02) =	1.8266e-01 (1.18e-02) -	1.0910e-01 (1.19e-02) -	1.7115e-01 (1.54e-02) -	1.5446e-01 (1.33e-02) -	**1.9411e-01 (1.51e-02)**
50×5	9.5661e-02 (2.03e-02) -	1.5670e-01 (1.27e-02) -	1.5228e-01 (1.10e-02) -	8.9664e-02 (1.20e-02) -	1.4922e-01 (1.29e-02) -	1.3093e-01 (1.07e-02) -	**1.7088e-01 (1.33e-02)**
50×6	1.3063e-01 (3.32e-02) -	1.9241e-01 (1.04e-02) -	1.8913e-01 (1.03e-02) -	1.1521e-01 (1.70e-02) -	1.7353e-01 (9.91e-03) -	1.5437e-01 (7.40e-03) -	**2.0413e-01 (1.04e-02)**
50×7	1.8958e-01 (2.18e-02) -	2.4148e-01 (9.04e-03) =	2.4008e-01 (1.07e-02) =	1.5393e-01 (1.51e-02) -	2.1056e-01 (1.06e-02) -	1.9658e-01 (1.02e-02) -	**2.5070e-01 (1.32e-02)**
60×4	9.4104e-02 (1.81e-02) -	1.4437e-01 (9.31e-03) -	1.4367e-01 (1.06e-02) -	9.4665e-02 (7.70e-03) -	1.2725e-01 (7.79e-03) -	1.1116e-01 (6.87e-03) -	**1.5304e-01 (9.46e-03)**
60×5	9.2603e-02 (2.17e-02) -	1.3889e-01 (1.15e-02) -	1.3726e-01 (1.24e-02) -	9.0183e-02 (9.88e-03) -	1.2107e-01 (1.06e-02) -	1.0813e-01 (8.25e-03) -	**1.5222e-01 (1.02e-02)**
60×6	1.4066e-01 (2.97e-02) -	**2.7660e-01 (1.39e-02) =**	2.6883e-01 (1.50e-02) =	1.3905e-01 (2.32e-02) -	2.3881e-01 (1.08e-02) -	2.1854e-01 (1.38e-02) -	**2.7559e-01 (9.47e-03)**
60×7	1.1119e-01 (2.56e-02) -	2.5256e-01 (1.35e-02) =	2.4726e-01 (1.17e-02) -	1.4513e-01 (1.34e-02) -	2.1822e-01 (1.45e-02) -	2.1144e-01 (1.40e-02) -	**2.6175e-01 (1.30e-02)**
70×4	1.1698e-01 (2.73e-02) -	2.5581e-01 (1.10e-02) =	2.4828e-01 (1.25e-02) -	1.3889e-01 (1.43e-02) -	2.2723e-01 (7.91e-03) -	2.0605e-01 (9.80e-03) -	**2.6298e-01 (8.05e-03)**
70×5	1.1887e-01 (2.53e-02) -	2.6098e-01 (9.18e-03) -	2.5778e-01 (9.13e-03) -	1.6034e-01 (1.27e-02) -	2.2001e-01 (9.90e-03) -	2.0893e-01 (7.17e-03) -	**2.7100e-01 (9.54e-03)**
70×6	1.1226e-01 (2.70e-02) -	2.4029e-01 (1.44e-02) =	2.3797e-01 (1.58e-02) -	1.5146e-01 (1.17e-02) -	2.0412e-01 (1.18e-02) -	1.9179e-01 (1.03e-02) -	**2.5165e-01 (1.19e-02)**
70×7	1.0509e-01 (2.32e-02) -	2.2808e-01 (8.08e-03) -	2.2576e-01 (7.98e-03) -	1.5572e-01 (1.41e-02) -	1.9457e-01 (8.77e-03) -	1.8027e-01 (8.12e-03) -	**2.3969e-01 (8.97e-03)**
80×4	2.4311e-01 (4.48e-02) -	3.7373e-01 (1.89e-02) =	3.6512e-01 (1.93e-02) =	2.1501e-01 (1.86e-02) -	3.1193e-01 (2.85e-02) -	3.0468e-01 (2.34e-02) -	**3.7604e-01 (2.33e-02)**
80×5	2.7243e-01 (4.04e-02) -	4.1534e-01 (1.89e-02) =	4.0668e-01 (2.08e-02) =	2.7027e-01 (3.35e-02) -	3.5641e-01 (1.74e-02) -	3.4175e-01 (2.16e-02) -	**4.1667e-01 (1.55e-02)**
80×6	2.2561e-01 (1.74e-02) -	3.6809e-01 (1.42e-02) =	3.6132e-01 (1.35e-02) -	2.2075e-01 (1.13e-02) -	3.1827e-01 (1.29e-02) -	3.0893e-01 (1.24e-02) -	**3.7844e-01 (8.49e-03)**
80×7	1.2115e-01 (2.92e-02) -	3.0722e-01 (8.71e-03) -	3.0027e-01 (1.15e-02) -	1.7297e-01 (1.58e-02) -	2.5931e-01 (1.26e-02) -	2.4933e-01 (6.28e-03) -	**3.1862e-01 (8.17e-03)**
100×4	1.2470e-01 (2.73e-02) -	3.0496e-01 (9.36e-03) -	2.9982e-01 (9.07e-03) -	1.8616e-01 (1.85e-02) -	2.5916e-01 (7.60e-03) -	2.4861e-01 (1.02e-02) -	**3.1652e-01 (1.27e-02)**
100×5	2.5795e-01 (1.43e-02) -	3.6405e-01 (9.56e-03) =	3.6013e-01 (7.71e-03) =	2.3067e-01 (1.52e-02) -	2.9737e-01 (1.09e-02) -	2.8835e-01 (8.82e-03) -	**3.7079e-01 (1.25e-02)**
100×6	3.1170e-01 (2.78e-02) -	**4.3835e-01 (1.77e-02) =**	4.2382e-01 (1.96e-02) =	2.4928e-01 (3.29e-02) -	3.7084e-01 (2.16e-02) -	3.5151e-01 (1.91e-02) -	4.3760e-01 (1.16e-02)
100×7	2.7200e-01 (3.04e-02) -	4.0693e-01 (1.60e-02) =	3.9915e-01 (1.50e-02) =	2.2988e-01 (2.05e-02) -	3.3688e-01 (2.10e-02) -	3.2614e-01 (1.83e-02) -	**4.1540e-01 (1.73e-02)**
120×4	2.8252e-01 (1.73e-02) -	**4.1550e-01 (1.20e-02) =**	4.0818e-01 (1.14e-02) =	2.3906e-01 (1.33e-02) -	3.5190e-01 (1.19e-02) -	3.3383e-01 (9.91e-03) -	4.1032e-01 (1.42e-02)
120×5	2.7240e-01 (1.80e-02) -	4.0356e-01 (5.86e-03) -	3.9658e-01 (7.08e-03) -	2.4467e-01 (1.52e-02) -	3.3659e-01 (9.37e-03) -	3.2544e-01 (8.75e-03) -	**4.1200e-01 (6.63e-03)**
120×6	2.6296e-01 (2.48e-02) -	3.9885e-01 (1.32e-02) =	3.9397e-01 (1.37e-02) =	2.4522e-01 (1.65e-02) -	3.3188e-01 (1.63e-02) -	3.1458e-01 (1.22e-02) -	**4.0307e-01 (1.28e-02)**
120×7	1.2723e-01 (2.24e-02) -	3.3211e-01 (7.75e-03) -	3.2819e-01 (7.79e-03) -	2.0107e-01 (1.21e-02) -	2.7418e-01 (7.67e-03) -	2.6351e-01 (8.81e-03) -	**3.4727e-01 (8.35e-03)**
+/-/=	**0/24/0**	**0/10/14**	**0/15/9**	**0/24/0**	**0/24/0**	**0/24/0**	

VI. TABLES

TABLE II
MULTI-PROBLEM WILCOXON'S TEST RESULTS

IGD				
RNSGA VS.	R^+	R^-	$P-$**value**	$P-$**value** $\leq$ **0.05**
AMOEA	171.0	0.0	0.00018	Yes
CDABC	171.0	0.0	0.00018	Yes
MODABC	171.0	0.0	0.00018	Yes
MOEA/D	171.0	0.0	0.00018	Yes
NSGA-II	171.0	0.0	0.00018	Yes
MOTS	171.0	0.0	0.00018	Yes

HV				
RNSGA VS.	R^+	R^-	$P-$**value**	$P-$**value** $\leq$ **0.05**
AMOEA	171.0	0.0	0.00018	Yes
CDABC	144.5	26.5	0.00739	Yes
MODABC	153.0	18.0	0.001632	Yes
MOEA/D	171.0	0.0	0.00018	Yes
NSGA-II	171.0	0.0	0.000166	Yes
MOTS	171.0	0.0	0.00018	Yes

TABLE III
COMPARISON OF DEFAULT OUTPUT RESULTS OF EACH ALGORITHM

IGD

Problem	**AMOEA**		**CDABC**		**MODABC**		**MOEA/D**		**NSGA-II**		**MOTS**		**RNSGA**	
	Value	**Rank**	**Value**	**Rank**	**Value**	**Rank**	**Value**	**Rank**	**Value**	**Rank**	**Value**	**Rank**	**Value**	**Rank**
100×0.4	5.1518e+02	6	4.6690e+02	4	**4.1894e+02**	**1**	4.3953e+02	3	5.3235e+02	7	5.0704e+02	5	4.2888e+02	2
100×0.6	8.0482e+02	5	7.0743e+02	2	7.4638e+02	4	7.4254e+02	3	8.3056e+02	7	8.0806e+02	6	**5.5656e+02**	**1**
100×0.8	1.0965e+03	5	1.0527e+03	4	**9.8477e+02**	**1**	1.1661e+03	7	1.0180e+03	3	1.1132e+03	6	9.9520e+02	2
225×0.4	1.1742e+03	5	1.1380e+03	3	**1.0902e+03**	**1**	1.2611e+03	6	1.1403e+03	4	1.3817e+03	7	1.1200e+03	2
225×0.6	**1.8060e+03**	**1**	2.1786e+03	5	2.0148e+03	4	2.3529e+03	7	1.9670e+03	2	2.2634e+03	6	1.9900e+03	3
225×0.8	3.0859e+03	2	3.4801e+03	6	**3.0036e+03**	**1**	3.3570e+03	5	3.3459e+03	4	4.3612e+03	7	3.2697e+03	3
400×0.4	2.3327e+03	3	2.3685e+03	4	2.2472e+03	2	2.5230e+03	6	2.4687e+03	5	2.9843e+03	7	**2.2014e+03**	**1**
400×0.6	4.8605e+03	5	4.6043e+03	2	4.7519e+03	4	5.0525e+03	6	4.6809e+03	3	5.4123e+03	7	**4.5604e+03**	**1**
400×0.8	7.1626e+03	6	6.4450e+03	2	6.7967e+03	5	7.6711e+03	7	6.7228e+03	4	6.6859e+03	3	**6.4306e+03**	**1**
625×0.4	**4.1705e+03**	**1**	4.3305e+03	2	4.7312e+03	4	5.0325e+03	6	4.7497e+03	5	6.3396e+03	7	4.6782e+03	3
625×0.6	1.1179e+04	5	1.1337e+04	6	1.0614e+04	2	1.1631e+04	7	1.0881e+04	4	**1.0492e+04**	**1**	1.0833e+04	3
625×0.8	1.7924e+04	6	1.4381e+04	5	**1.3340e+04**	**1**	1.3693e+04	2	1.3883e+04	4	2.1538e+04	7	1.3866e+04	3
900×0.4	**9.1196e+03**	**1**	9.4849e+03	3	9.4790e+03	2	1.1265e+04	6	1.0168e+04	4	1.1367e+04	7	1.0373e+04	5
900×0.6	1.8096e+04	6	1.7326e+04	4	1.7251e+04	3	1.7408e+04	5	**1.6037e+04**	**1**	2.6006e+04	7	1.6507e+04	2
900×0.8	3.8528e+04	6	3.1771e+04	5	**2.9394e+04**	**1**	2.9888e+04	2	3.0073e+04	3	4.3463e+04	7	3.1029e+04	4
1225×0.4	1.1894e+04	6	7.6778e+03	3	7.8513e+03	5	7.7506e+03	4	7.3753e+03	2	2.6556e+04	7	**6.8136e+03**	**1**
1225×0.6	8.7504e+04	6	6.4954e+04	3	7.3035e+04	5	6.4850e+04	2	6.4976e+04	4	1.0829e+05	7	**8.4838e+03**	**1**
1225×0.8	1.1045e+05	6	9.5029e+04	3	9.4998e+04	2	1.0545e+05	5	9.8886e+04	4	1.5509e+05	7	**8.3481e+03**	**1**
Total rank		81		66		48		89		70		111		39

HV

Problem	**AMOEA**		**CDABC**		**MODABC**		**MOEA/D**		**NSGA-II**		**MOTS**		**RNSGA**	
	Value	**Rank**	**Value**	**Rank**	**Value**	**Rank**	**Value**	**Rank**	**Value**	**Rank**	**Value**	**Rank**	**Value**	**Rank**
100×0.4	9.2072e-02	7	1.1101e-01	2	**1.1115e-01**	**1**	1.0788e-01	3	9.3004e-02	6	9.4613e-02	5	1.0599e-01	4
100×0.6	1.3408e-01	5	1.5384e-01	2	1.5332e-01	3	1.4719e-01	4	1.3051e-01	6	1.2606e-01	7	**1.5632e-01**	**1**
100×0.8	1.9080e-01	6	2.1134e-01	2	2.1120e-01	3	1.9256e-01	4	1.9227e-01	5	1.6616e-01	7	**2.1809e-01**	**1**
225×0.4	1.9349e-01	5	2.0982e-01	3	2.1438e-01	2	1.9747e-01	4	1.8609e-01	6	1.7096e-01	7	**2.1554e-01**	**1**
225×0.6	2.2522e-01	6	2.4233e-01	4	2.4528e-01	2	2.4324e-01	3	2.3551e-01	5	1.7473e-01	7	**2.5924e-01**	**1**
225×0.8	2.7956e-01	6	2.9136e-01	3	2.9385e-01	2	2.9091e-01	4	2.8509e-01	5	2.0640e-01	7	**3.1337e-01**	**1**
400×0.4	2.2849e-01	6	2.5710e-01	3	2.5206e-01	4	2.5799e-01	2	2.5103e-01	5	1.9813e-01	7	**2.7339e-01**	**1**
400×0.6	3.4018e-01	6	3.6676e-01	2	3.5447e-01	4	3.4359e-01	5	3.5993e-01	3	3.0509e-01	7	**3.7726e-01**	**1**
400×0.8	3.5531e-01	7	4.1194e-01	2	3.9913e-01	3	3.9203e-01	5	3.9697e-01	4	3.8386e-01	6	**4.3255e-01**	**1**
625×0.4	3.1750e-01	6	3.4418e-01	2	3.3044e-01	3	3.2751e-01	4	3.2231e-01	5	2.4102e-01	7	**3.5330e-01**	**1**
625×0.6	2.8701e-01	7	3.6596e-01	2	3.5035e-01	4	3.5994e-01	3	3.5016e-01	5	3.2322e-01	6	**3.7848e-01**	**1**
625×0.8	3.1526e-01	6	4.3399e-01	2	4.1650e-01	5	4.2453e-01	3	4.1918e-01	4	3.0044e-01	7	**4.4769e-01**	**1**
900×0.4	3.9854e-01	6	4.3324e-01	2	4.0184e-01	5	4.2801e-01	3	4.2295e-01	4	3.2702e-01	7	**4.4347e-01**	**1**
900×0.6	4.1559e-01	6	5.1049e-01	2	5.0494e-01	3	4.9850e-01	4	4.9695e-01	5	3.7420e-01	7	**5.1289e-01**	**1**
900×0.8	4.3816e-01	7	5.9178e-01	4	5.9424e-01	3	5.7588e-01	5	6.0011e-01	2	4.4801e-01	6	**6.0716e-01**	**1**
1225×0.4	3.7598e-01	6	4.2731e-01	3	4.2649e-01	4	4.2433e-01	5	4.2991e-01	2	2.8691e-01	7	**4.3877e-01**	**1**
1225×0.6	4.0395e-01	6	5.2346e-01	3	4.8155e-01	5	5.1292e-01	4	5.2694e-01	2	3.8549e-01	7	**8.1811e-01**	**1**
1225×0.8	4.6754e-01	6	5.6274e-01	2	5.4930e-01	3	5.1876e-01	5	5.3180e-01	4	4.1948e-01	7	**8.1123e-01**	**1**
Total rank		110		45		59		70		78		121		21

VII. FIGURES

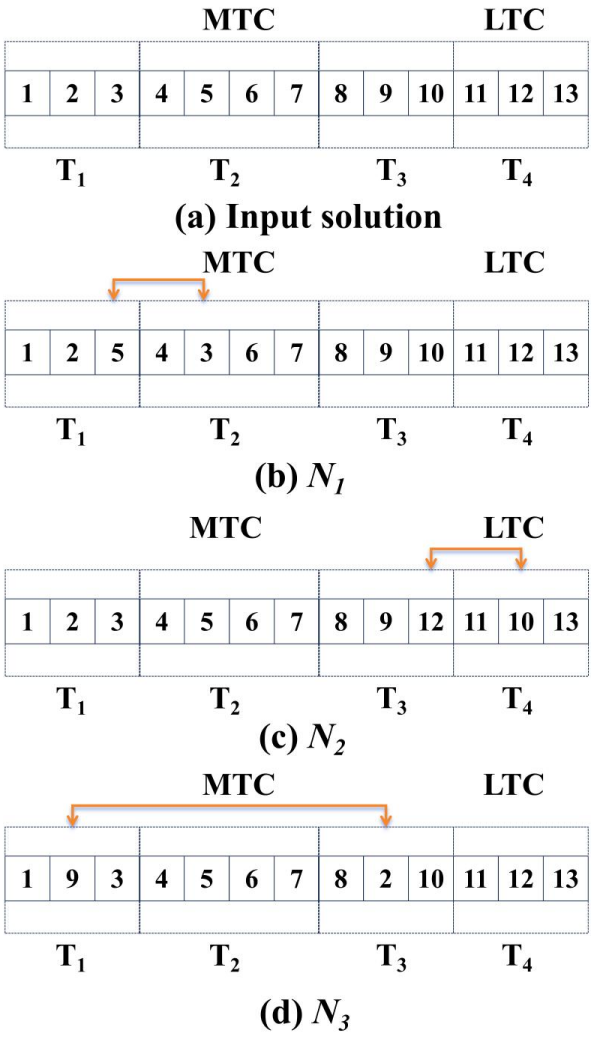


Fig. 3. Near task optimization mechanism

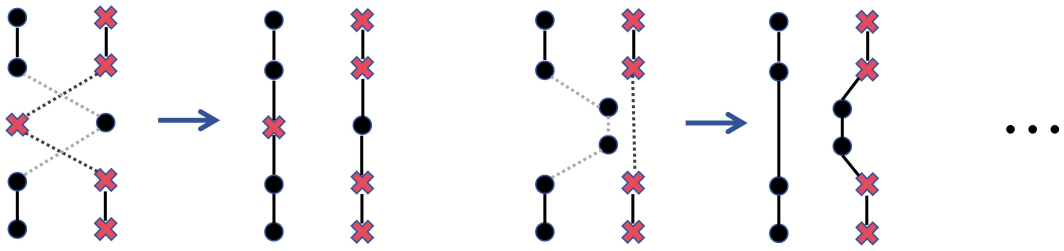


Fig. 4. Some simple examples of rationalizing the path through NTOM

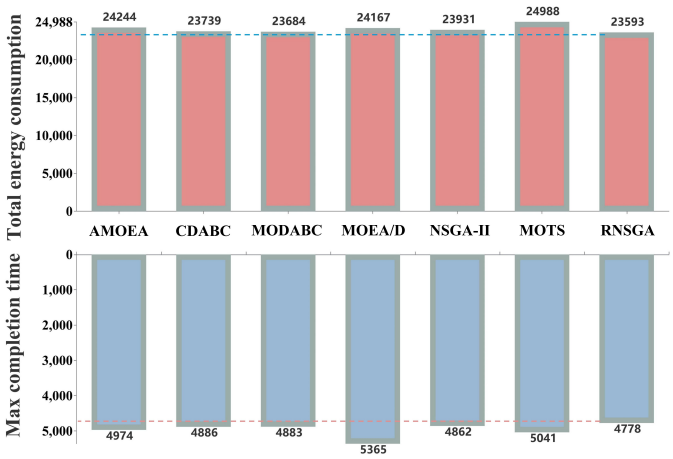
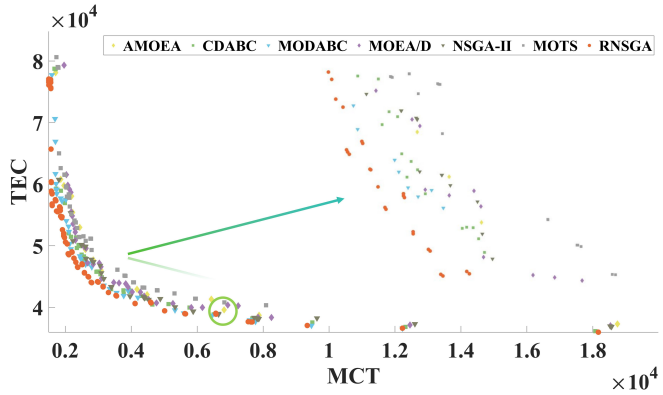
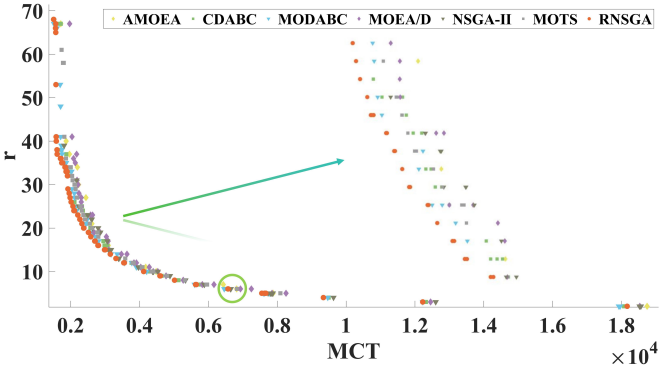


Fig. 5. Objective values for algorithms in a instance



(a) PF distribution on *MCT* and *TEC*



(b) PF distribution on *MCT* and *r*

Fig. 6. PF distribution on 625×0.6 scenario

REFERENCES

[1] Lou-Lei Dai, Quan-Ke Pan, Zhong-Hua Miao, Ponnuthurai Nagaratnam Suganthan, and Kai-Zhou Gao. Multi-objective multi-picking-robot task allocation: Mathematical model and discrete artificial bee colony algorithm. *IEEE Transactions on Intelligent Transportation Systems*, 2023.