Particle Swarm Optimization for Cooperative Multi-Robot Task Allocation: A Multi-Objective Approach

Changyun Wei[1], Ze Ji[2] and Boliang Cai[1]

Abstract— **This paper presents a new Multi-Objective Particle Swarm Optimization (MOPSO) approach to a Cooperative Multi-Robot Task Allocation (CMRTA) problem, where the robots have to minimize the total team cost and, additionally, balance their workloads. We formulate the CMRTA problem as a more complex variant of multiple Travelling Salesman Problems (mTSP) and, in particular, address how to minimize the total travel distance of the entire robot team, as well as how to minimize the highest travel distance of an individual robot. The proposed approach extends the standard single-objective Particle Swarm Optimization (PSO) to cope with the multiple objectives, and its novel feature lies in a Pareto front refinement strategy and a probability-based leader selection strategy. To validate the proposed approach, we first use three benchmark functions to evaluate the performance of finding the true Pareto fronts in comparison with four existing well-known algorithms in continuous spaces. Afterwards, we use six datasets to investigate the task allocation mechanisms in dealing with the CMRTA problem in discrete spaces.**

Index Terms— **Multi-robot Systems; Optimization and Optimal Control; Cooperating Robots.**

I. Introduction

In many practical applications, robots are seldom stand-alone systems, but they need to coexist and coordinate with each other so as to achieve a team goal more efficiently [1]. In order to obtain teamwork benefit, multiple robots require more variables to be considered in a cohesive manner [2]. Multi-robot task allocation addresses how to assign a set of tasks to a set of robots with various objectives. This paper studies a Cooperative Multi-Robot Task Allocation (CMRTA) problem, where the robots are required to balance their workloads when completing the overall task as a team in an efficient manner. To be specific, the robots have to minimize the total cost over the team and the highest cost of any individual robot.

In this work, the CMRTA problem is modelled as a more complex variant of multiple Travelling Salesman Problem (mTSP), where a team of robots has to collect a set of targets dispersed in the environment, and then retrieve them back to a nest base. Such a scenario can be motivated by many practical applications, e.g., package collection and transport in intelligent warehouses [3], and search and rescue after an earthquake [4]. This paper aims at providing a novel task allocation approach to the CMRTA problem with multiple objectives. Thus, we do not focus on low-level of action execution, e.g., how to navigate to a target or to avoid collisions among each other in a cluster situation.

The main contributions of this work include:

1) a novel task allocation approach to the CMRTA problem, taking into account both the overall team cost and any individual workload,
2) an alternative multi-objective PSO algorithm, with the feature of Pareto front refinement and probability-based leader selection strategies, and
3) a competitive and generic solution to Multi-Objective Optimization problems in finding the true Pareto fronts in continuous spaces.

We organize the paper as follows. In the next section, we discuss the state of the art to typical multi-robot task allocation problems, and Section III formulates the CMRTA problem as a more complex variant of mTSP. Section IV details the proposed approach, and Section V evaluates the proposed approach in both continuous and discrete spaces. Finally, we conclude this work in Section VI.

II. Related Work

Multi-robot task allocation addresses the problem of finding the optimal task-to-robot assignment so as to achieve a good team performance [5]. It has been extensively studied in the agent/robot domain, and usually formulated as an optimization problem, where a set of robots has to complete a set of goals in such a way that optimizes the overall team performance.

A. Multiple Travelling Salesman Problem

If there is only one robot involved in the environment, the task allocation can be modelled as a Travelling Salesman Problem (TSP), where the salesman has to visit all the nodes with minimal travel cost. If there are multiple robots working in a shared space, the problem can be considered as a multiple Travelling Salesman Problem (mTSP) [6], where multiple salesmen have to visit all the nodes and each node has to be visited exactly once, with the objective of minimizing the total travel cost. A considerable literature has studied the mTSP but mainly focused on how to minimize the total travel distance [7], [8]. Thus, as in [9], [10], [11], this work seeks to minimize the cost of an individual robot, additionally.

Recent advances in evolutionary optimization have also attempted to find the solutions to the mTSP, e.g., genetic

*This work was supported by the National Natural Science Foundation of China (Grant No. 61703138), and Natural Science Foundation of Jiangsu Province (Grant No. BK20170307)

[1] Changyun Wei and Boliang Cai are with the College of Mechanical and Electrical Engineering, Hohai University, Changzhou, 213022, China, c.wei@hhu.edu.cn.

[2] Corresponding Author: Ze Ji is with the School of Engineering, Cardiff University, Cardiff, CF24 3AA, United Kingdom, jiz1@cardiff.ac.uk.

optimizer [12], [13], ant colony and decomposition optimizer [14], [15], ant lion optimizer [16], grey wolf optimizer [17], physarum optimizer [18], and particle swarm optimizer [19], [20]. In the standard mTSP, the salesmen are required to visit all the nodes exactly once from a starting node and return back to it after a round trip. The typical objective function is to minimize the total travel distance. As mentioned previously, this work focuses on a cooperative robot team, where the robots are expected to balance their workloads as well as to maintain good team performance in allocating the nodes to be visited.

B. Multi-Objective Optimization

Multi-Objective Optimization (MOO) techniques are popular for solving the task allocation problems. The work [21] presents an auction-based approach, in which the robots calculate the bids for each task by means of MOO. In order to improve the performance of a scheduling problem, three aspects (i.e., task execution time, task transferring time and task execution cost) have been considered in [22]. The non-dominated sorting genetic algorithm (NSGA-II) [23] has shown its advantage in solving many MOO tasks, e.g., optimizing the beam structures [24] and optimizing the laser welding process [25]. In NSGA-II, an elitist strategy is employed in each generation to avoid the loss of non-dominated solutions. The work [26] introduces four mutation operations to improve the search ability. To ensure diversity and convergence, the work [27] presents a strategy to select the leader for each particle. Several metrics are combined into a single fuzzy metric to convert multiple objectives into a single objective in [28].

Among the above evolutionary algorithms, PSO is popular and can ensure fast convergence, but the standard PSO is originally designed for handing single objective optimization problems. Moreover, existing extensions for multiple objectives are usually ad hoc, and a few of them can be used to solve a generic MOO problem in both discrete and continuous spaces [29].

III. Problem Formulation

In this work, we relate the CMRTA problem with multiple objectives to a more complex variant of the mTSP. We use a complete graph $G(V, E)$ to define the problem, where V denotes the set of nodes (e.g., targets dispersed in the environment), and E represents the set of edges linking the nodes. Each edge $e_{i,j} \in E$ is associated with a weight $d_{i,j} \geq 0$ that defines the travel distance from node i to j, and the travel distance between two nodes is assumed to be symmetric, i.e., $d_{i,j} = d_{j,i}$.

A. Cooperative Multi-Robot Task Allocation (CMRTA)

In the CMRTA problem, each robot should be allocated with a set of subtasks to complete. We use m to denote the number of robots, and $\boldsymbol{T}_k$ to represent the set of subtasks allocated to the k-th robot, $k \in [1, m]$. Here a subtask indicates a sequence of nodes that forms a round trip for a robot to visit. Thus, the robot team needs to complete the overall task $\boldsymbol{T} = [\boldsymbol{T}_1, \boldsymbol{T}_2 \ldots, \boldsymbol{T}_m]$. We use $Cost(\boldsymbol{T}_k)$ to represent the tour cost of k-th robot to visit its allocated subtasks,

$$Cost(\boldsymbol{T}_k) = \sum_{e_{i,j} \in E} w_{ijk} d_{i,j}, \tag{1}$$

where the binary variable w_{ijk} indicates whether the tour of the k-th robot includes the edge $e_{i,j}$ or not. To perform such a task allocation problem, we need to find m Hamiltonian tours that should satisfy two objectives, i.e., minimizing the total tour cost for the overall robot team and the longest tour for any individual robot. In order to find Hamiltonian tours, each node in the graph has to be visited exactly once by only one robot. Thus, the goal of the cooperative MRTA problem is to minimize the following function:

$$f(\boldsymbol{T}) = (\Sigma_{k=1}^{m} Cost(\boldsymbol{T}_k), \max_{k \in [1,m]} Cost(\boldsymbol{T}_k)), \tag{2}$$

subject to :

$$\begin{cases} \bigcap\limits_{k \in [1,m]} \{\boldsymbol{T}_k \backslash \{o\}\} = \phi \\ \bigcup\limits_{k \in [1,m]} \boldsymbol{T}_k = V, \end{cases} \tag{3}$$

where o denotes the nest node. According to the above objective function, the robots are expected to minimize the entire team cost and the longest individual travel cost.

B. Multi-Objective Optimization Model

The CMRTA problem formulated above can be abstracted into a Multi-Objective Optimization (MOO) problem, where several objectives are required to be achieved, and each objective depends on a vector of decision variables. Without loss of generality, a MOO problem can be defined as

$$minimize\ f(\boldsymbol{x}) = \{f_1(\boldsymbol{x}), f_2(\boldsymbol{x}), \ldots f_l(\boldsymbol{x})\}, \tag{4}$$

subject to:

$$\begin{cases} g_i(\boldsymbol{x}) \leq 0, \text{for } i \in [1, r] \\ h_i(\boldsymbol{x}) = 0, \text{for } i \in [1, s], \end{cases}$$

where $\boldsymbol{x}$ is the vector of decision variables, $f_i(\boldsymbol{x})$ represents the i-th objective function, and $g_i(\boldsymbol{x})$, $h_i(\boldsymbol{x})$ are the constraint functions.

In a MOO problem, we can evaluate a solution with respect to different objectives. Let $\boldsymbol{u}$ and $\boldsymbol{v}$ be two solutions of the problem defined in Equation 4. We say that $\boldsymbol{v}$ is dominated by $\boldsymbol{u}$ (denoted by $\boldsymbol{v} \preceq \boldsymbol{u}$), if the following conditions are satisfied

$$\forall i : f_i(\boldsymbol{u}) \leq f_i(\boldsymbol{v}) \text{ and } \exists j : f_j(\boldsymbol{u}) < f_j(\boldsymbol{v}), \tag{5}$$

where $i, j \in [1, l]$. If a solution is not dominated by any other ones, it is called a Pareto optimal solution. The Pareto front represents the set of all Pareto optimal solutions. Thus, an algorithm to a MOO problem seeks to search for a set of non-dominated solutions.

IV. Multi-Objective PSO Approach

In this section, we will first discuss the standard PSO algorithm, and then an extended novel algorithm will be proposed to address multiple objectives. Afterwards, an adapted version for discrete spaces is presented to deal with the CMRTA problem.

A. Standard Particle Swarm Optimization

PSO is originally inspired by collective behaviours of birds, fishes or insects. With regard to a MOO, a solution to such a problem can be represented by a vector of particles $\boldsymbol{x}_i = (x_{i1}, x_{i2}, \ldots, x_{iD})$, where D is the dimension of the search space. Thus, each particle is endowed with a position in the search space, and it can move around with a velocity $\boldsymbol{v}_i = (v_{i1}, v_{i2}, \ldots, v_{iD})$ so as to find the optimal solutions. We can use the following equation to calculate the coming location of a particle for the next timestep,

$$\boldsymbol{x}_i^{t+1} = \boldsymbol{x}_i^t + \boldsymbol{v}_i^{t+1}, \tag{6}$$

where t denotes the t-th timestep of iterations.

To decide the moving direction, each particle should consider its own trajectory and the behaviours of the others. Specifically, two best solutions are used to update the position of each particle, i.e., the personal best position $pbest$ and the global best position $gbest$. Thus, each particle can update its own position based on the information of $pbest$ and $gbest$ as follows,

$$\boldsymbol{v}_i^{t+1} = \omega \boldsymbol{v}_i^t + r_1\phi_1(gbest^t - \boldsymbol{x}_i^t) + r_2\phi_2(pbest_i - \boldsymbol{x}_i^t), \tag{7}$$

where ω denotes the inertia weight that controls to what extent the previous velocity affects the coming velocity. ϕ_1 and ϕ_2 are constants that control the influences of the personal and global guides. r_1 and r_2 are random numbers uniformly distributed in $[0, 1]$. The velocity has a range in $[-\boldsymbol{v}_{max}, \boldsymbol{v}_{max}]$, which defines the limit of the velocity. PSO algorithms will terminate when a good fitness value is achieved or the maximum number of iterations is reached.

B. Extended Multi-Objective PSO Approach

To cope with multiple objectives, the standard PSO has been modified for multi-dimensional spaces, such as aggregating and lexicographic ordering [30]. The typical idea of aggregating is to integrate multiple objectives into a single objective so that the standard PSO is still applicable to find a solution. In lexicographic ordering, multiple objectives must be ranked based on their priorities, so each objective will be considered as a single PSO problem.

Comparatively, we are interested in a Pareto front-based approach in this work. To this end, we have to modify Equation 7, because it is impossible to find the $pbest$ and $gbest$ in multi-dimensional spaces for MOO problems. Instead, we need to find the Pareto front, so the $gbest$ will be replaced by a $\boldsymbol{leader}$. It is important to maintain the diversity of the Pareto front and choose the $\boldsymbol{leader}$ for fast convergence. Several methods have been proposed to address this issue, e.g., the work [31] selects the leader randomly from all non-dominated solutions. OMOPSO [32] utilizes an external archive, based on the crowding distance from NSGA-II, to filter out leader solutions and divides the particle swarms into three groups. To accelerate the convergence of the swarm, it applies uniform, non-uniform and no mutation operators to those groups, respectively. SMPOS [33], extended from OMOPSO, employs a construction factor with random social and cognition learning factors. In our work, we will introduce a Pareto front refinement strategy and a probability-based mechanism for selecting the $\boldsymbol{leader}$.

1) Main Procedure: The main procedure of the proposed MOPSO approach is summarized as follows:

- Step 1. A group of particles P and a global Pareto front set P_f are initialized. Each particle $p_i \in P$ is born with a random solution $\boldsymbol{x}_i$ and a private Pareto front set P_f^i. The solution size of all private/global Pareto fronts is set to n.
- Step 2. At timestep t, each particle's solution $\boldsymbol{x}_i^t$ will be evaluated using $f(\boldsymbol{x})$ and added into P_f^i.
- Step 3. If the solution size of the private Pareto front set for particle p_i exceeds the limit n (i.e., $\|P_f^i\| > n$), it needs to be refined based on the PARETO FRONT REFINEMENT STRATEGY (see Algorithm 1), which will delete redundant or inferior solutions.
- Step 4. The global Pareto front set P_f is updated using $P_f \leftarrow \bigcup P_f^i$, and then it also needs to be refined based on the PARETO FRONT REFINEMENT STRATEGY (see Algorithm 1).
- Step 5. The refined global Pareto front will be stored into a Elite Set B (defined to avoid population degradation) using $B \leftarrow B \bigcup P_f$.
- Step 6. A $\boldsymbol{leader}$ will be selected from the global Pareto front set P_f based on the PROBABILITY-BASED SELECTION STRATEGY, which will be discussed latter.
- Step 7. The non-dominated solutions in each particle's private Pareto front set are updated using the following equation,

$$\begin{cases} \boldsymbol{v}_{f,i}^{t+1,k} = \boldsymbol{v}_i^t + r_1\phi_1(\boldsymbol{leader} - \boldsymbol{x}_i^t) + r_2\phi_2(\boldsymbol{x}_{f,i}^{t,k} - \boldsymbol{x}_i^t) \\ \boldsymbol{x}_{f,i}^{t+1,k} = \boldsymbol{x}_i^t + \boldsymbol{v}_{f,i}^{t+1,k}, \end{cases} \tag{8}$$

  where $\boldsymbol{x}_i^t$ denotes the current solution of particle p_i at timestep t, and $\boldsymbol{x}_{f,i}^{t,k} \in P_f^i$ denotes the k-th solution in the private Pareto front set of particle p_i at timestep t.
- Step 8. Each particle p_i randomly chooses a new solution $\boldsymbol{x}_{f,i}^{t+1,k}$ from its private Pareto front set P_f^i to update its position $\boldsymbol{x}_i^{t+1}$ for the next timestep, and the private Pareto front set will be updated accordingly,

$$P_f^i \leftarrow P_f^i \bigcup \{\boldsymbol{x}_{f,i}^{t+1,k}\}. \tag{9}$$

- Step 9. If the next timestep $t+1$ does not exceed the limit of iterations, switch to Step 2.
- Step 10. The termination condition is reached, and the Elite set B will be refined based on the PARETO FRONT REFINEMENT STRATEGY (see Algorithm 1).

2) Pareto Front Refinement Strategy: As mentioned above, we have to refine the private Pareto front set, the global Pareto front set and the Elite set in step 3, 4 and 10, respectively. Herein, we will introduce a PARETO FRONT REFINEMENT STRATEGY to delete redundant or inferior solutions.

It should be noted that the Elite set is used to avoid population degradation, which is a troublesome issue to

Algorithm 1: Pareto Front Refinement Strategy.

Input: Any unrefined Pareto set P_r.
Output: Refined Pareto front set $P_{r'}$.

```
1  for x_i, x_j ∈ P_r, x_j ≠ x_i do
2      if x_i ⪯ x_j then
3          Remove x_i from P_r.            ▷ remove inferior solutions.
4      end
5  end
6  while ‖P_r‖ ≥ n do
7      Calculate C_{P_r} using CROWDED VALUE(P_r);
8                                          ▷ see Algorithm 2.
9      Find x such that arg min (C_{P_r});
10                 ▷ find x with the minimum crowded value.
11     Remove x from P_r.
12 end
```

MOO problems. This is because when refining the private or global Pareto front sets, some non-dominated solutions may be deleted or replaced by others, due to the limit of Pareto front sets. Such a case can cause premature convergence and loss of optimal solutions. Thus, in order to cope with this issue, the Elite set B is employed to store all the global optimal solutions in each iteration (see Step 5). After the terminate condition is reached, we can use the refinement strategy to reduce the size of the Elite set (Step 10). In such a way, we can make sure that all the non-dominated solutions are kept during the iterations.The PARETO FRONT REFINEMENT STRATEGY is described in Algorithm 1.

We can see that when refine a Pareto front set, we first remove all the inferior solutions (line 1-4). If the size of the set still exceeds the limit n, we need to remove some of the non-dominated solutions to compress the set. To decide which solution to remove, we develop a CROWDED VALUE function (see Algorithm 2) for evaluation, and then select the one with the minimum crowed value (line 7, 9).

Algorithm 2: Crowded Value Calculation.

Input: Pareto set P_r.
Output: Set of crowded value C_{P_r}.

```
1  for x ∈ P_r do
2      Calculate f̂(x)   ▷ normalization using Equation 10.
3  end
4  C_{P_r} = φ                          ▷ initialize crowded value set.
5  K[x_i, x_j] = ‖f̂(x_i) − f̂(x_j)‖_2, x_i, x_j ∈ P_r
6        ▷ adjacency score matrix of each solution in f̂(x).
7  for x ∈ P_r do
8      x^a, x^b = arg min{K[x :], 2}     ▷ find two solutions with the smallest values.
9      c_x = min{|f̂_i(x^a) − f̂_i(x^b)|}, i ∈ [1, l]     ▷ obtain crowded value.
10     C_{P_r} ← C_{P_r} ∪ {c_x}  ▷ update the crowded value set.
11 end
12 return C_{P_r}
```

In line 2 of Algorithm 2, for each solution of the input Pareto set P_r, a normalization process is used to standardize the objective vector $f(\boldsymbol{x}) = (f_1(\boldsymbol{x}), f_2(\boldsymbol{x}) \dots f_l(\boldsymbol{x}))$ by the below equation,

$$\hat{f}_i(\boldsymbol{x}) = \frac{f_i(\boldsymbol{x}) - \min\limits_{k\in[1,n]}(f_i(\boldsymbol{x}_f^k))}{\max\limits_{k\in[1,n]}(f_i(\boldsymbol{x}_f^k)) - \min\limits_{k\in[1,n]}(f_i(\boldsymbol{x}_f^k))}, \tag{10}$$

where $i \in [1, l]$, $k \in [1, n]$ and $\boldsymbol{x}_f^k$ denotes the k-th solution of the input Pareto front set P_r. The normalized objective vector is denoted by $\hat{f}(\boldsymbol{x}) = [\hat{f}_1(\boldsymbol{x}) \dots \hat{f}_l(\boldsymbol{x})]$.

As mentioned above, the global Pareto front set P_f, the private Pareto set P_f^i for each particle p_i, and the Elite set B need to be refined based on the refinement strategy. For the sake of simplicity and portability, all the input values (e.g., P_f, P_f^i and B) are denoted as P_r in Algorithm 1.

3) Probability-Based Selection Strategy: In the main procedure of the proposed MOPSO approach, we have mentioned that the $\boldsymbol{leader}$ needs to be found so as to update a particle's position according to Equation 8. The $\boldsymbol{leader}$' selection is crucial for faster convergence.

In this work, the $\boldsymbol{leader}$ is selected from the global Pareto front set P_f with a probabilistic distribution calculated by the CROWDED VALUE in Algorithm 2. Thus, we have to calculate the probabilities for all the solutions in the global Pareto set using the following formula,

$$Prob(\boldsymbol{x}_f^k) = \frac{c_{\boldsymbol{x}_f^k}}{\Sigma c_{\boldsymbol{x}_f^k}}, \boldsymbol{x}_f^k \in P_f, \tag{11}$$

where $\boldsymbol{x}_f^k$ denotes the k-th solution in the global Pareto front set P_f. Afterwards, the solution with better expectation can be selected as the $\boldsymbol{leader}$ with a high probability to update all the private Pareto front set of each particle in Equation 8.

C. Adapted Version for Discrete Spaces

So far the presented MOPSO approach can solve the MOO problems in continuous spaces, and it still needs to be adapted so as to deal with the CMRTA problem in discrete spaces. To this end, Equation 8 should be replaced to update the solutions of each particle's private Pareto front set by

$$\boldsymbol{x}_{f,i}^{t+1,k} = (\boldsymbol{x}_i^t \circledast \boldsymbol{leader}) \circledast \boldsymbol{x}_{f,i}^{t,k}, \tag{12}$$

where the operator $\circledast$ is used to merge two discrete solutions, and Algorithm 3 details the corresponding merge algorithm.

In Algorithm 3, we introduce a new operator $\oplus$ to swap two nodes in a discrete sequence. To be specific, Op $= \{j, q\}$ (lines 5, 7) means that the node in position j needs to be exchanged with the node in position q. For example, if the operator $\oplus$ acts on the discrete sequence $\hat{\boldsymbol{x}}_i' = (6, 3, 8, 5, 1)$ with the exchange pair Op $= \{2, 4\}$, after the operation $\hat{\boldsymbol{x}}_i' \oplus$ Op, the discrete sequence will become $\hat{\boldsymbol{x}}_i' = (6, 5, 8, 3, 1)$. In lines 4-8, we first record indexes that differentiate two discrete solutions. Afterwards, some nodes in one solution will be exchanged according to a probability function $Prob_{SA}$ (see line 10-15).

Algorithm 3: Updating discrete solutions by merging.

Input: Two discrete solutions $\boldsymbol{x}_i'$ and $\boldsymbol{x}_i''$, with regard to particle p_i.
Output: updated solution $\boldsymbol{x}_i^*$.

Ops = ϕ ▷ initialize operator sets for storing position pairs that need to be swapped in a solution.
$\hat{\boldsymbol{x}}_i' \leftarrow \boldsymbol{x}_i'$ ▷ make a temp copy for recording differences between $\boldsymbol{x}_i'$ and $\boldsymbol{x}_i''$.
for $j \in [1 \dots \|\hat{\boldsymbol{x}}_i'\|]$ **do**
  if $\hat{\boldsymbol{x}}_i'[j] \neq \boldsymbol{x}_i''[j]$ **then**
    Op $\leftarrow \{j, q\}$, where $\boldsymbol{x}_i''[q] = \hat{\boldsymbol{x}}_i'[j]$)
    Ops $\leftarrow$ Ops $\cup$ {Op} ▷ record the location.
    $\hat{\boldsymbol{x}}_i' \leftarrow \hat{\boldsymbol{x}}_i' \oplus$ Op ▷ swap the nodes in $\hat{\boldsymbol{x}}_i'$.
  end
end
for Op $\in$ Ops **do**
  if $rand() \leq Prob_{SA}(t)$ **then**
    $\boldsymbol{x}_i' \leftarrow \boldsymbol{x}_i' \oplus$ Op ▷ swap the nodes in $\boldsymbol{x}_i'$.
  end
  $\boldsymbol{x}_i^* \leftarrow \boldsymbol{x}_i'$
end
return $\boldsymbol{x}_i^*$

In order to improve the search performance and guarantee convergence, we apply a Simulated Anneal strategy to define the probability function as follows,

$$\begin{cases} Prob_{SA}(t) = \mathcal{N}(\mu(t), \sigma) \\ \sigma = (\max\{\gamma_1 - \mu(t), \mu(t) - \gamma_2\})/3 \\ \mu(t) = \alpha + \beta \cos(\pi \times \frac{t}{t_{max}}), \end{cases} \tag{13}$$

where α and β control the shapes of the centre curve, and they must obey $\alpha+\beta \leq 1$, and $\alpha-\beta \geq 0$. γ_1 and γ_2 are used to control the size of the probability interval by changing the σ in normal distribution $\mathcal{N}$.

With regard to the CMRTA problem, each robot needs to be allocated with a tour to visit. Thus, we need to find m Hamiltonian tours or cycles, where each node has to be visited only once. In order to minimize the tour cost for each robot, a Hamiltonian Tour improvement algorithm is presented in Algorithm 4, which will be called after a new solution is generated by Algorithm 3. Here we also apply the operator $\oplus$ to exchange two nodes in a tour, with the aim of minimizing the cost for each tour.

Algorithm 4: Hamiltonian Tour Improvement.

Input: Subtask of k-th robot $\boldsymbol{T}_k, k \in [1 \dots m]$.
Output: Improved subtask $\boldsymbol{T}_k'$ of k-th robot.

for $i \in [1 \dots \|\boldsymbol{T}_k\|]$ **do**
  for $j \in [i \dots \|\boldsymbol{T}_k\|]$ **do**
    if $d_{\boldsymbol{T}_k[i],\boldsymbol{T}_k[j]} + d_{\boldsymbol{T}_k[i+1],\boldsymbol{T}_k[j+1]} \leq d_{\boldsymbol{T}_k[i],\boldsymbol{T}_k[i+1]} + d_{\boldsymbol{T}_k[j],\boldsymbol{T}_k[j+1]}$ **then**
      Op $\leftarrow \{i+1, j\}$ ▷ recode the nodes.
      $\boldsymbol{T}_k \leftarrow \boldsymbol{T}_k \oplus$ Op ▷ swap the nodes in $\boldsymbol{T}_k$.
    end
  end
end
$T_k' \leftarrow T_k$;
return T_k'

V. Experiments and Results

In this section, in order to evaluate the proposed approach, we first employ three benchmark functions to assess the performance of finding the true Pareto fronts in continuous spaces. Then, we use six datasets to investigate the task allocation mechanism in dealing with the CMRTA problem in discrete spaces.

A. Results of Finding True Pareto Front

As in [23], [27], the benchmark ZDT functions have been used to evaluate the performance of finding the true Pareto fronts. The definition of ZDT functions is described in Table I. We can see that ZDT1 and ZDT3 have convex Pareto fronts, but the convex Pareto front of ZDT3 is not contiguous, and ZDT2 has nonconvex Pareto front.

To evaluate the performance of finding the Pareto fronts, several MOO algorithms are available to test the ZDT functions, but the work [34] found that OMOPSO [32], SPEA2 [35], NSGA-II [23], and SMPSO [33] can yield the best results. Thus, in this work we will compare the proposed approach with those four well-known algorithms.

Although many metrics are available to evaluate two Pareto fronts, we cannot find one that is absolutely reliable. Thus, as introduced in [36], [27], we also employ two quality indicators I_H and $I_{\epsilon+}$ to comparing the performance of algorithms, and the values of these two indicators express the closeness to the true Pareto fronts. The values can be between zero and one, and the higher is the value of I_H (or the smaller is the value of $I_{\epsilon+}$), the closer it is to the true Pareto front. Table II shows the means and standard deviations for the ZDT functions, and the bold values represent the best results among the four well-known algorithms and our proposed approach. Moreover, Figure 1 can demonstrate that the our proposed approach can outperforms the other four algorithms .

B. Results of CMRTA in Discrete Spaces

To validate the proposed approach to the CMRTA problem, six datasets from TSPLIB[1] are used to compare the performance between NSGA-II and our proposed approach. The reason why we choose TSPLIB as the test benchmark is because the CMRTA problem is modelled as a more complex variant of mTSP. In the environment of TSPLIB, each robot needs to be allocated with a set of nodes to visit, with the goal of minimizing the overall team cost and any individual cost. With regard to the parameter configurations, the population of two approaches is set to 100, and the Pareto front size is 15. The number of iterations is 200, and the mutation rate is set to 10 in NSGA-II.

In this experiment, we evaluate two algorithms in two maps (labelled KroA and KroB), scaled nodes/cities (i.e.,

[1] http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/.

TABLE I

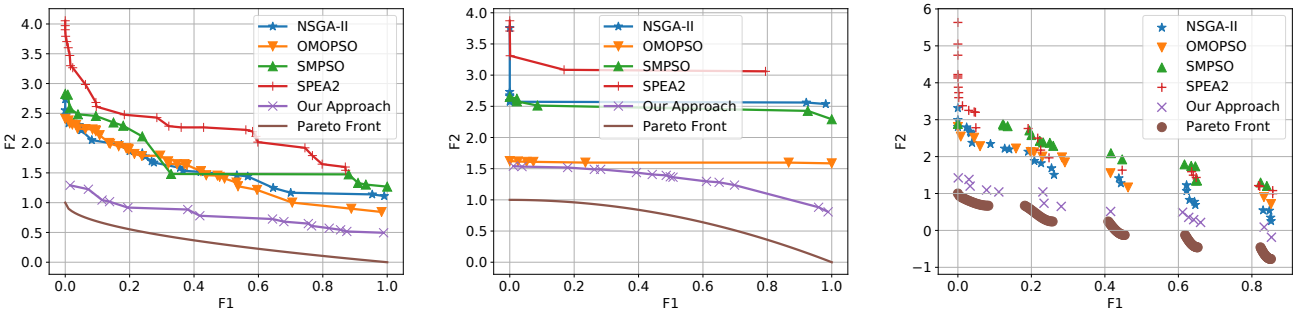
DEFINITION OF THE BENCHMARK ZDT FUNCTIONS.

Problem	Objective Functions	Optimal Solutions	comments
ZDT1	$f_1(\mathbf{x}) = x_1$ $f_2(\mathbf{x}) = g(\mathbf{x})\left[1 - \sqrt{x_1/g(\mathbf{x})}\right]$ $g(\mathbf{x}) = 1 + 9\left(\sum_{i=2}^{n} x_i\right)/(n-1)$	$x_1 \in [0,1]$ $x_i = 0$ $i = 2, \ldots, n$	convex
ZDT2	$f_1(\mathbf{x}) = x_1$ $f_2(\mathbf{x}) = g(\mathbf{x})\left[1 - (x_1/g(\mathbf{x}))^2\right]$ $g(\mathbf{x}) = 1 + 9\left(\sum_{i=2}^{n} x_i\right)/(n-1)$	$x_1 \in [0,1]$ $x_i = 0$ $i = 2, \ldots, n$	non-convex
ZDT3	$f_1(\mathbf{x}) = x_1$ $f_2(\mathbf{x}) = g(\mathbf{x})\left[1 - \sqrt{x_1/g(\mathbf{x})} - \frac{x_1}{g(\mathbf{x})}\sin(10\pi x_1)\right]$ $g(\mathbf{x}) = 1 + 9\left(\sum_{i=2}^{n} x_i\right)/(n-1)$	$x_1 \in [0,1]$ $x_i = 0$ $i = 2, \ldots, n$	convex disconnected

TABLE II

RESULTS OF BENCHMARK ZDT FUNCTIONS AFTER 30 RUNS.

Problem		OMOPSO		SPEA2		NSGA-II		SMPSO		Our Approach	
		I_H	$I_{\epsilon+}$	I_H	$I_{\epsilon+}$	I_H	$I_{\epsilon+}$	I_H	$I_{\epsilon+}$	I_H	$I_{\epsilon+}$
ZDT1	Mean	0.739	**0.480**	0.610	0.511	0.856	0.531	0.554	0.506	**0.883**	0.484
	Std.	0.140	0.065	0.121	0.070	0.150	0.074	0.164	0.058	0.086	0.027
ZDT2	Mean	**0.739**	0.898	0.473	0.796	0.583	0.949	0.527	0.894	0.602	**0.613**
	Std.	0.268	0.101	0.202	0.088	0.222	0.039	0.290	0.141	0.044	0.062
ZDT3	Mean	0.843	0.552	0.711	0.499	0.851	0.461	0.693	0.543	**0.858**	**0.384**
	Std.	0.126	0.077	0.144	0.068	0.107	0.061	0.136	0.093	0.022	0.025



(a) Pareto fronts for ZDT1. (b) Pareto fronts for ZDT2. (c) Pareto fronts for ZDT3.

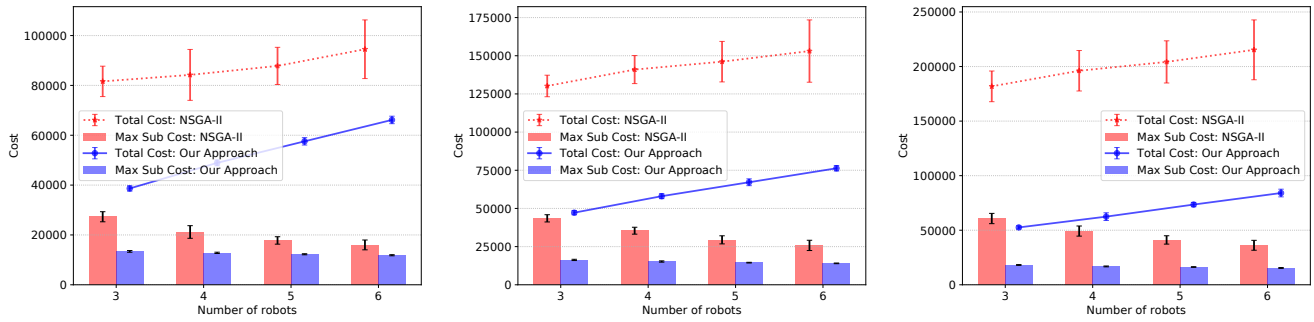
Fig. 1. Comparison of the performance with regard to the ZDT functions.

100, 150, and 200), and scaled robot teams (i.e., 3, 4, 5, and 6 robots), as shown in the results in Figure 2. In the figure, *Total Cost* represents the overall travel distances among the robot team, and *Max Sub Cost* indicates the maximum travel distance of any individual robot. For a cooperative robot team, we seek to minimize both of them. As can be seen from the results, in comparison with the NSGA-II, our proposed approach can reduce the *Total Cost* at least 26%, and reduce the *Max Sub Cost* at least 21%. Moreover, the *Max Sub Cost* is close to $1/m$ of the Total Cost according to our proposed approach. It means that the tour cost for each robot is almost the same, which also implies that the robots can balance their workloads in a cooperative manner.

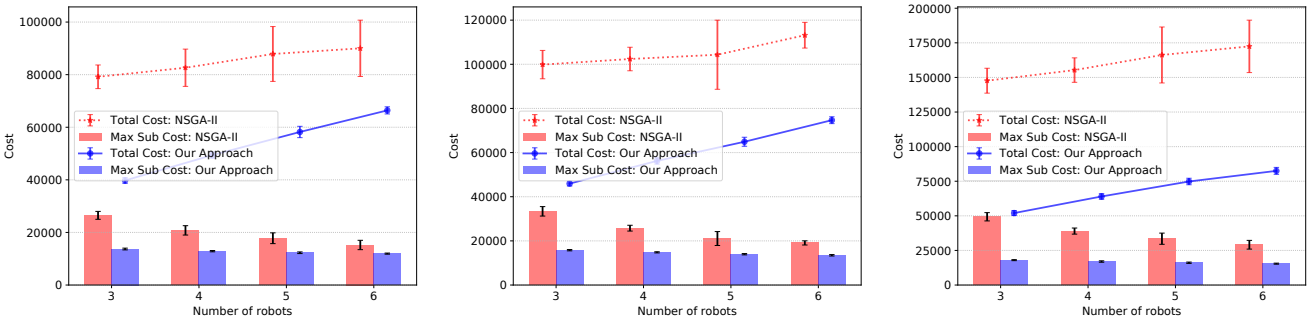
Figure 3 depicts the allocated tours in the map of KroB by the proposed approach. Here, *TC* means the total cost of all the tours, and *MSC* represents the maximum cost of an individual tour. We can also directly observe that multiple robots indeed can improve their team performance. In other words, the individual cost reduces along with the increase of the robot size. However, the total travel cost will be increased if more robots engage in the teamwork.

VI. CONCLUSIONS AND FUTURE WORK

In this work, a novel Multi-Objective Particle Swarm Optimization approach has been proposed to solve the CM-RTA problem. As the problem involves multiple objectives, the standard single particle swarm optimization cannot be directly applied to resolve this problem. We present a new Pareto front-based approach, which features a Pareto front refinement strategy and a probability-based leader selection strategy. In comparison with other well-known algorithms, the experimental results show that the proposed approach can provide a competitive solution to multi-objective optimization problems in continuous spaces. Moreover, it can take into account both the overall team cost and any individual workload when resolving the CMRTA problem. In future work, we are interested in modelling the interferences among the robots and capability of continuously completing a set of



(a) Map: KroA, Number of nodes: 100 (b) Map: KroA, Number of nodes: 150 (c) Map: KroA, Number of nodes: 200



(d) Map: KroB, Number of nodes: 100 (e) Map: KroB, Number of nodes: 150 (f) Map: KroB, Number of nodes: 200

Fig. 2. Performance comparison using the map KroA and KroB(30 runs).

allocated subtasks in real robots. This is because adding more robots cannot always yield good team performance due to potential interferences between each other in physical robot teams. The subtasks might be distinguished with various types that need different capabilities to complete. Thus, multiple constraints should be considered in order to achieve multiple objectives in a specific task.

REFERENCES

[1] B. P. Gerkey and M. J. Matarić, "A formal analysis and taxonomy of task allocation in multi-robot systems," *The International Journal of Robotics Research*, vol. 23, no. 9, pp. 939–954, 2004.

[2] C. Wei, K. V. Hendriks, M. van Riemsdijk, and C. Jonker, "A formal graphical language of interdependence in teamwork," *IEEE Intelligent Systems*, vol. 34, no. 5, pp. 25–34, 2019.

[3] V. Digani, L. Sabattini, C. Secchi, and C. Fantuzzi, "Ensemble coordination approach in multi-agv systems applied to industrial warehouses," *IEEE Transactions on Automation Science and Engineering*, vol. 12, no. 3, pp. 922–934, 2015.

[4] D. Calisi, A. Farinelli, L. Iocchi, and D. Nardi, "Multi-objective exploration and search for autonomous rescue robots," *Journal of Field Robotics*, vol. 24, no. 8-9, pp. 763–777, 2007.

[5] K. Lerman, C. Jones, A. Galstyan, and M. J. Matarić, "Analysis of dynamic task allocation in multi-robot systems," *The International Journal of Robotics Research*, vol. 25, no. 3, pp. 225–241, 2006.

[6] T. Bektas, "The multiple traveling salesman problem: an overview of formulations and solution procedures," *Omega*, vol. 34, no. 3, pp. 209–219, 2006.

[7] K. Braekers, K. Ramaekers, and I. Van Nieuwenhuyse, "The vehicle routing problem: State of the art classification and review," *Computers & Industrial Engineering*, vol. 99, pp. 300–313, 2016.

[8] J. R. Montoya-Torres, J. L. Franco, S. N. Isaza, H. F. Jiménez, and N. Herazo-Padilla, "A literature review on the vehicle routing problem with multiple depots," *Computers & Industrial Engineering*, vol. 79, pp. 115–129, 2015.

[9] J. Thunberg, D. A. Anisi, and P. Ögren, "A comparative study of task assignment and path planning methods for multi-ugv missions," in *Optimization and Cooperative Control Strategies*, pp. 167–180, Springer, 2009.

[10] G. N. Frederickson, M. S. Hecht, and C. E. Kim, "Approximation algorithms for some routing problems," in *17th annual symposium on foundations of computer science*, pp. 216–227, IEEE, 1976.

[11] N. Karapetyan, K. Benson, C. McKinney, P. Taslakian, and I. Rekleitis, "Efficient multi-robot coverage of a known environment," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 1846–1852, IEEE, 2017.

[12] J. Li, M. Zhou, Q. Sun, X. Dai, and X. Yu, "Colored traveling salesman problem," *IEEE Transactions on Cybernetics*, vol. 45, no. 11, pp. 2390–2401, 2014.

[13] L. W. Jer, A. B. Jambek, and N. S. Chin, "Kursawe function optimisation using hybrid micro genetic algorithm (hmga)," *Soft Computing*, 2015.

[14] L. Ke, Q. Zhang, and R. Battiti, "Moea/d-aco: A multiobjective evolutionary algorithm using decomposition and antcolony," *IEEE Transactions on Cybernetics*, vol. 43, no. 6, pp. 1845–1859, 2013.

[15] L.-C. Lu and T.-W. Yue, "Mission-oriented ant-team aco for min–max mtsp," *Applied Soft Computing*, vol. 76, pp. 436–444, 2019.

[16] S. Mirjalili, P. Jangir, and S. Saremi, "Multi-objective ant lion optimizer: a multi-objective optimization algorithm for solving engineering problems," *Applied Intelligence*, vol. 46, no. 1, pp. 79–95, 2017.

[17] S. Mirjalili, S. Saremi, S. M. Mirjalili, and L. d. S. Coelho, "Multi-objective grey wolf optimizer: a novel algorithm for multi-criterion optimization," *Expert Systems with Applications*, vol. 47, pp. 106–119, 2016.

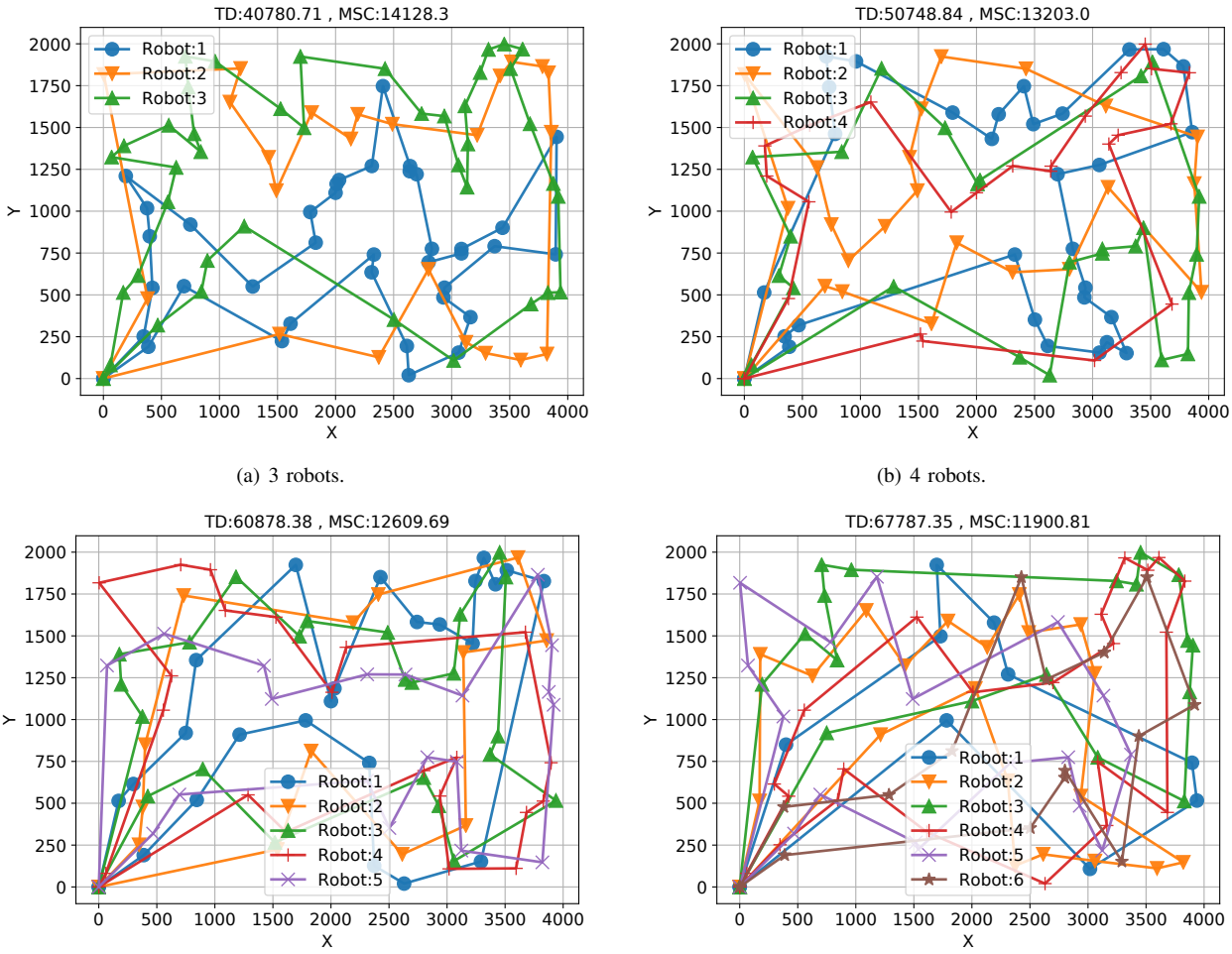
[18] L. Liu, Y. Song, H. Zhang, H. Ma, and A. V. Vasilakos, "Physarum optimization: A biology-inspired algorithm for the steiner tree problem in networks," *IEEE Transactions on Computers*, vol. 64, no. 3, pp. 818–831, 2013.

[19] X. Liang, W. Li, Y. Zhang, and M. Zhou, "An adaptive particle swarm optimization method based on clustering," *Soft Computing*, vol. 19, no. 2, pp. 431–448, 2015.

[20] M. Nouiri, A. Bekrar, A. Jemai, S. Niar, and A. C. Ammari, "An effective and distributed particle swarm optimization algorithm for flexible job-shop scheduling problem," *Journal of Intelligent Manufacturing*, vol. 29, no. 3, pp. 603–615, 2018.

[21] A. T. Tolmidis and L. Petrou, "Multi-objective optimization for dynamic task allocation in a multi-robot system," *Engineering Applications of Artificial Intelligence*, vol. 26, no. 5-6, pp. 1458–1468, 2013.

[22] F. Ramezani, J. Lu, and F. Hussain, "Task scheduling optimization



(a) 3 robots.

(b) 4 robots.

(c) 5 robots.

(d) 6 robots.

Fig. 3. Allocated tour for the robots with 200 nodes in the map of KroB.

in cloud computing applying multi-objective particle swarm optimization," in *International Conference on Service-oriented computing*, pp. 237–251, Springer, 2013.

[23] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: Nsga-ii," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 2, pp. 182–197, 2002.

[24] T. Vo-Duy, D. Duong-Gia, V. Ho-Huu, H. Vu-Do, and T. Nguyen-Thoi, "Multi-objective optimization of laminated composite beam structures using nsga-ii algorithm," *Composite Structures*, vol. 168, pp. 498–509, 2017.

[25] P. Jiang, C. Wang, Q. Zhou, X. Shao, L. Shu, and X. Li, "Optimization of laser welding process parameters of stainless steel 316l using fem, kriging and nsga-ii," *Advances in Engineering Software*, vol. 99, pp. 147–160, 2016.

[26] H. Zhou, M. Song, and W. Pedrycz, "A comparative study of improved ga and pso in solving multiple traveling salesmen problem," *Applied Soft Computing*, vol. 64, pp. 564–580, 2018.

[27] V. Mokarram and M. R. Banan, "A new pso-based algorithm for multi-objective optimization with continuous and discrete design variables," *Structural and Multidisciplinary Optimization*, vol. 57, no. 2, pp. 509–533, 2018.

[28] S. Trigui, O. Cheikhrouhou, A. Koubaa, U. Baroudi, and H. Youssef, "Fl-mtsp: a fuzzy logic approach to solve the multi-objective multiple traveling salesman problem for multi-robot systems," *Soft Computing*, vol. 21, no. 24, pp. 7351–7362, 2017.

[29] W. Tong, S. Chowdhury, and A. Messac, "A multi-objective mixed-discrete particle swarm optimization with multi-domain diversity preservation," *Structural and Multidisciplinary Optimization*, vol. 53, no. 3, pp. 471–488, 2016.

[30] C. A. C. Coello, G. B. Lamont, D. A. Van Veldhuizen, *et al.*, *Evolutionary algorithms for solving multi-objective problems*, vol. 5. Springer, 2007.

[31] C. C. Coello and M. S. Lechuga, "Mopso: A proposal for multiple objective particle swarm optimization," in *Proceedings of the 2002 Congress on Evolutionary Computation*, vol. 2, pp. 1051–1056, IEEE, 2002.

[32] M. R. Sierra and C. A. C. Coello, "Improving pso-based multi-objective optimization using crowding, mutation and-dominance," in *International conference on evolutionary multi-criterion optimization*, pp. 505–519, Springer, 2005.

[33] A. J. Nebro, J. J. Durillo, J. Garcia-Nieto, C. C. Coello, F. Luna, and E. Alba, "Smpso: A new pso-based metaheuristic for multi-objective optimization," in *IEEE Symposium on Computational Intelligence in Multi-Criteria Decision-Making*, pp. 66–73, IEEE, 2009.

[34] J. J. Durillo and A. J. Nebro, "jmetal: A java framework for multi-objective optimization," *Advances in Engineering Software*, vol. 42, no. 10, pp. 760–771, 2011.

[35] E. Zitzler, M. Laumanns, and L. Thiele, "Spea2: Improving the strength pareto evolutionary algorithm," *TIK-report*, vol. 103, 2001.

[36] A. Liefooghe, M. López-Ibáñez, L. Paquete, and S. Verel, "Dominance, epsilon, and hypervolume local optimal sets in multi-objective optimization, and how to tell the difference," in *Proceedings of the Genetic and Evolutionary Computation Conference*, pp. 324–331, ACM, 2018.