Research Article

Biobjective UAV/UGV Collaborative Rendezvous Planning in Persistent Intelligent Task-Based Wireless Communication

Qin Wang, Hua Chen, Junwei Tian, Jia Wang, and Yu Su

School of Mechanical and Electronic Engineering, Xi’an Technological University, Xi’an, China

Correspondence should be addressed to Hua Chen; hc156131@163.com

Received 25 June 2021; Revised 26 July 2021; Accepted 3 August 2021; Published 15 August 2021

Academic Editor: Balakrishnan Nagaraj

Copyright © 2021 Qin Wang et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

This paper addresses a multiobjective Unmanned Aerial Vehicle (UAV) and Unmanned Ground Vehicle (UGVs) collaboration scheduling problem in which UAVs must be recharged periodically during a persistent task, such as wireless communication and making aerial panoramic VR video by UAV. The proposal is to introduce a UGV/UAV cooperative system in which UAVs should be recharged by UGVs periodically, and also, the UGVs have to visit their task points except for the recharge points. The objective of the problem is to schedule and plan paths with the tradeoff of path length and makespan for UAVs and UGVs. The mathematical model which can be considered a combinatorial multiobjective optimization problem is presented firstly, and the solution of the problem is composed of presubsolution and postsolution. The multiobjective adaptive large neighborhood is hybrid with the Pareto local search (MOALP) to resolve the problem. The experimental result shows that the proposed algorithm outperforms the compared algorithms on the rendezvous planning problem for UAVs and UGVs working collaboratively in intelligent tasks.

1. Introduction

Unmanned Aerial Vehicles (UAVs) have been successfully used for many applications in both civil and military scenarios, such as surveillance, precision agriculture, package delivery, and facility inspection. Multiple sensors can be equipped to obtain various information because UAVs are flexible; however, the endurance capability of UAV greatly limits its operation time. Recently, researches which try to enable UAVs working in persistent tasks have been drawn much attention.

It is a great challenge for the UAV/UGV collaboration system working in persistent tasks since UAVs have a short endurance. Previously, two kinds of methods are widely studied to improve the operation time of the UAV/UGV collaboration system in the literature; the first kind of method is to set recharging stations for UAVs in the environment to improve the endurance of UAVs; in this case, UAVs are required to visit recharging stations periodically and be replenished by battery swapping systems; for the second type of method, UGVs are used as mobile recharging stations, and UAVs need to be replenished by rendezvousing with a UGV periodically.

The problem of path planning for UAV/UGV collaboration systems has been widely investigated in existing literatures. Sunder and Rathinam [1] deployed multiple depots (charging stations) in the environment to refuel one UAV working in monitoring applications, the mixed integer linear programming (MILP) model is presented, and a local search heuristic was proposed for resolving the problem. [2] considered multi-UAVs and charging stations in the working scenario, and the problem is transformed into a generalized traveling salesman problem (GTSP) and then resolved by a GTSP solver.

Different from [1, 2], mobile robots were utilized in some literatures instead of using charging depots, and Maini and Sujit [3] proposed a model named FCURP-MRS which ensures that UAVs can be recharged by UGVs by generating UGV paths and UAV paths, respectively. [4–6] studied the path planning problem for mobile charging robots. In [4, 5], multiple working robots and one charging robot were considered in the problem, and the problem is resolved by using both dynamic programming and local search heuristics. In [6], the authors studied the single charging cycle problem and multirobot charging problem, and they
proposed the integer linear programming method and multirobot Noon-Bean transformation to solve the problem, respectively.

Multiobjective path planning for the UAV/UGV collaboration system working in persistent tasks has been seldom studied in the previous related works while there exist a lot of researches on multiobjective path planning for robots, such as [7–12]. In [7, 10], NSGA-II which was introduced in [13] was used to obtain the Pareto front of the multiobjective path planning problem. Zhu et al. [8] proposed two multiobjective memetic algorithms (MOMAs) for the wheeled robot path planning problem, and the algorithm was evaluated in both simulation and real environment. In [9], the authors developed a new multiobjective optimization algorithm which is called the multiobjective firefly algorithm (MO-FA) to resolve a path planning problem which has three objectives. In [12], a local search heuristic was hybrid with SPEA2 to find the efficient solutions of the presented problem.

Researches on the combinatorial multiobjective problems (CMOPs) have a significant contribution to the study of multiobjective path planning, especially for scheduling problems. Heuristic algorithms have been widely studied for approximating the Pareto front of CMOPs because it is usually NP-hard to find the exact Pareto front of a CMOP. The Pareto local search (PLS) is one of the efficient algorithms for resolving CMOPs, and it is an extension of a single-objective local search [14]. Lust and Teghem [15] proposed 2PPLS, and it is a variant of PLS which consists of two phases of PLS in the algorithm. Based on PLS, various meta-heuristics have been proposed; in [16], the authors combined the idea of PLS and MOEA/D, and the proposed algorithm MOMAD contains three populations which represent subproblem solutions, PL solutions, and external solutions, respectively. [17] proposed the Grid Weighted Sum Pareto Local Search (GWS-PLS), its main contribution is using the grid Pareto dominance instead of the Pareto dominance, and the grid Pareto dominance method enables the algorithm to adapt to both concave and convex problems; results show that GWS-PLS performs well in the multiobjective traveling salesman problem and multiobjective knapsack problem. Recently, [18–20] studied the multiobjective adaptive large neighborhood search (MOALNS), which is a heuristic algorithm based on the single objective of the adaptive large neighborhood search. In [21], the authors proposed MOALNS/D which is an improved version of MOALNS by combining the idea of decomposition with MOALNS.

In this paper, we mainly focus on multiobjective path planning for UAV/UGV collaboration systems working in persistent tasks; as mentioned above, the present related works often have the following shortcomings: firstly, path planning for charging robots was focused on while seldom works considered both replenishment robots and UAVs; secondly, the charging robots were only considered mobile charging stations while these robots can also complete other tasks; and thirdly, the single objective was considered in previous works while the decision-maker may need to tradeoff multiobjectives when planning paths for path planning for UAV/UGV collaboration systems.

In this paper, we firstly present a new scheduling problem and then propose an efficient algorithm for resolving the problem. The contributions of our work are shown as follows:

(1) The problem is formulated as mixed integer linear programming, and planning paths for UAVs and UGVs simultaneously as well as UGVs have multi-task points to visit

(2) The presented mathematical model is a biobjective optimization problem, which is firstly considered

(3) A memetic algorithm which combines PLS and MOALNS is proposed for resolving the presented

(4) To resolve the problem efficiently, several heuristics and the local search procedure are developed for the algorithm

The remainder of the paper is organized as follows. In Section 2, we formally define the problem, and the corresponding mathematic model is presented. In Section 3, we present the framework of the proposed algorithm as well as the initialization method, solution representation, operators for the presubsolution, and proposed Hybrid Pareto Local Search (HPLS). In Section 4, we give details on MOALNS in HPLS, which contains five removal heuristics and three insertion heuristics. In Section 5, we implement the proposed algorithm for the presented problem and compare it with three other algorithms. In Section 6, we provide conclusions and avenues for future research.

2. Problem Statement

The problem we addressed in this paper is a combinational optimization problem, and we define and formulate it as follows.

Given a set of UAV and UGV working points, denote them as $V_u$ and $V_g$, respectively; the number of UAVs and UGVs are $M$ and $N$, respectively; UAVs have a fixed velocity of $v_u$, while the maximum velocity of UGVs is $v_{max}$, and UAVs have the maximum endurance of $T_u$, which means that all UAVs need to be replenished by UGVs periodically. A solution of the problem consists of paths of UAVs and UGVs, and two objectives, total length and makespan, should be optimized simultaneously to obtain an external set which contains a set of efficient solutions. The UAV and UGV paths should meet the following conditions: each UAV working point should be visited exactly once by any UAV, and according to the planned UAV path, all recharging points and the corresponding time will be generated, UGVs need to visit each UAV working point exactly, and also all recharging points should be visited by UAV before the UAV runs out of energy. Other notations used for the mathematical model are given as follows:

- $v_{uk}^k$: the $k$th UAV working point, where $k \in \{1, \cdots, K\}$
- $V_u = \bigcup_{k=1}^K v_{uk}$ is the set of all UAV working points
where \( \nu_h \) is the \( h \)th UGV working point, where \( h \in \{1, \cdots, H\} \) and \( V_g = \bigcup_{h \in \mathbb{H}} V_{gh} \) is the set of all UGV working points.

\( x_{ijm} \) is a binary variable which equals to 1 if the \( m \)th UAV travels from the \( i \)th UAV working point to the \( j \)th UAV working point and 0 if otherwise.

\( x_{ijm}^{j'} \) is a binary variable which equals to 1 if the \( j' \)th UGV travels from the \( i' \)th UGV working point to the \( j' \)th UGV working point and 0 if otherwise.

\( c_{ij} \) is the Euclidean distance between the \( i \)th UAV working point and \( j \)th UAV working point.

\( c_{ij}^{j'} \) is the Euclidean distance between the \( i' \)th UGV working point and \( j' \)th UGV working point.

\( p_u^m \) is the \( m \)th UAV working point, where \( m \in \{1, \cdots, M\} \) is the index of UAV

\( p_g^m \) is the \( m \)th UGV working point, where \( n \in \{1, \cdots, N\} \) is the index of UGV

\( t_0(\nu_i); t_1(\nu_i) \): the recharging time of \( \nu_i \). \( t_m(\nu_i) \): the recharging point in the \( m \)th UGV path, where \( m \in \{1, \cdots, L(n)\} \), \( L(n) \) is the number of recharging points in \( p_g^m \), and \( V_c = \bigcup_{i \in \mathbb{H}} \bigcup_{j \in \mathbb{L}} \bigcup_{l \in \mathbb{I}} \bigcup_{j' \in \mathbb{I}'} t_i \) is the set of all recharging points.

The formal definition of the problem is as follows: let \( G_1 = (V_u, E_u, c) \) be an undirected graph, the problem is to find a path \( p_u \), which contains \( M \) subpaths: \( p_u^m \), and each subpath has the same start position: \( v_{i1}^u \); meanwhile, for each vertex in \( V_u \setminus v_{i1}^u \), it should be visited exactly once by \( p_u \). For the \( M \) subpaths, the set of recharging vertices \( V_c \) can be obtained by considering \( T_c \), and get \( G_2 = (V_g \cup V_c, c') \) be another undirected graph, finding a path \( p_g \), which contains \( N \) subpaths: \( p_g^n \), and the start vertex of each subpath is \( v_{i1}^g \); in addition, each vertex in \( V_g \cup V_c \setminus v_{i1}^g \) should be visited by \( p_g \) exactly once, and the arrival time of each vertex in \( V_c \) should be less than its corresponding recharging time. A solution of the problem is composed of \( p_u \) and \( p_g \). The optimization problem is biobjective; furthermore, the minimization of total length implies reducing the sum of the length of \( p_u \) and \( p_g \), and the minimization of makespan implies reducing the time consumption of the UAV/UGV collaboration system.

The subsegment in one postsolution can be described as follows: for a postsolution \( p_g^m \), its subsegments have two cases: (1) the path formed between two adjacent recharging points and (2) the path between the last recharging point and the end of \( p_g^m \). Figure 1 shows an example of the subsegment, in which \( s_{n2} \) denotes the \( 2 \)th subsegment in the \( n \)th UGV path; then, the length of subsegment \( s_{n2} \) is

\[
P_{s_{n2}} = \sum_{i=1}^{j_{n2}-1} c(s_{n2}^{i}, s_{n2}^{i+1}).
\]

There have been two objectives in the proposed problem: path length and makespan. For the objective function of length cost \( L \), it contains two components: \( L_u \) and \( L_g \), which represent the length of UAV paths and the length of UGV paths, respectively, and the effect of the objective of makespan is to minimize the maximum working time of UAVs and UGVs. The two objectives can be, respectively, formulated as the following two equations.

\[
\min \left( \sum_{m=1}^{M} L_u^m + \sum_{n=1}^{N} L_g^n \right),
\]

\[
\min \max \left\{ T_u^m, T_g^n \right\},
\]

where \( L_u \) and \( L_g \) represent the length of \( p_u^m \) and \( p_g^n \) respectively, and \( T_u \) and \( T_g \) denote the makespan of \( p_u^m \) and \( p_g^n \) respectively. \( L_u, L_g, T_u, \) and \( T_g \) can be calculated according to

\[
L_u^m = \sum_{i=1}^{K} \sum_{j=1}^{K} c_{ij} x_{ijm},
\]

\[
L_g^n = \sum_{i=1}^{H} \sum_{j=1}^{H} c_{ij} x_{ij'n},
\]

\[
T_u = L_u^m / v_u,
\]

\[
T_g = t_{n2}^n + \sum_{e \in E_p(n)} c(e).
\]

The objective functions are subjected to the following constraints:

\[
\sum_{m=1}^{M} \sum_{i=1}^{K} x_{ijm} = 1, \quad \forall j \in \{2, \cdots, K\},
\]

\[
\sum_{m=1}^{M} \sum_{j=1}^{K} x_{ijm} = 1, \quad \forall j \in \{1, \cdots, K\},
\]

\[
\sum_{m=1}^{M} \sum_{i=1}^{J} x_{ijm}' = 1, \quad \forall j' \in \{2, \cdots, H + L\},
\]

\[
\sum_{m=1}^{M} \sum_{j=1}^{J} x_{ijm}' = 1, \quad \forall i' \in \{1, \cdots, H + L\},
\]

\[
t_n - t_n^{-1} \leq T_c, \quad \forall n \in \{1, \cdots, N\}, \forall s \in \{1, \cdots, s(n)\},
\]
\[ x_{ijm} = 1, \quad \forall i \in \{1, \ldots, K\}, \forall j \in \{1, \ldots, K\}, \forall m \in \{1, \ldots, M\}, \]  
\[ x_{ij} = 1, \quad \forall i \in \{1, \ldots, K\}, \forall j \in \{1, \ldots, K\}, \forall n \in \{1, \ldots, N\}. \]

where equations (8) and (9) can guarantee that the edge that enters each node in \( V_u \) is selected and the edge that leaves each node in \( V_g \) is selected, respectively. Equations (10) and (11) make sure that every node in \( V_u \cup V_g \) is selected to be visited exactly. Equation (12) implies that the solution should meet the constraint of recharging time windows; i.e., the time interval between two adjacent recharging points in the path shall not exceed the maximum interval \( T_c \). Equation (13) formulates the domain of variable \( x_{ijm} \), which expresses whether the \( nth \) UAV travels from \( v^g_i \) to \( v^g_j \) or not; similarly, equation (14) expresses whether the \( nth \) UGV travels from \( v^g_i \) to \( v^g_j \) or not.

3. Proposed Approach

In this section, the structure of the solution is presented firstly, and then, the framework of MOALP and operators for the presubsolution are given.

3.1. Solution Representation. The structure of the solution is shown in Figure 2, and it can be seen from the figure that the solution of the problem consists of two components: UAV path and UGV path, and \( v^u_i \) and \( v^g_i \) are the UAV start point and UGV start point, respectively; both UAV and UGV paths are composed of several subpaths, and every subpath is corresponding to a robot path; furthermore, the length of the UAV path is fixed at \( M + K - 1 \), and the UGV path length is not fixed since the recharging point set is determined by the UAV path.

3.2. Framework of MOALP. The proposed algorithm in this paper integrates the multiobjective adaptive large neighborhood search (MOALNS) into the PLS framework, and it is called multiobjective ALNS hybrid with PLS (MOALP); the detail of MOALP is shown in Algorithm 1.

Similar to MOMAD [16], MOALP maintains three populations during the optimization process, which are as follows:

1. \( P_p \): the population will be undergone PLS
2. \( P_i \): this population contains the solutions corresponding to subproblems
3. \( P_e \): the external population contains all nondominated solutions found so far

3.3. Initialization. The initialization method for the postsolution is as follows: first of all, a temporary initial postsolution is set by random initialization; then, a reinforcement procedure for the postsolution which is shown in Figure 3 is applied in the temporary set to improve the number of feasible solutions in the initial set. The solution reinforcement proposed in this paper is based on the following two facts: firstly, given a postsolution, recharging points are sequential for each UGV path, and secondly, in a UGV path, the distance of two adjacent recharging points is less than \( D_{max} \), where \( D_{max} = v_{max}(t(r^g_{j+1}) - t(r^g_j)) \).

The operators which are proposed for resolving the multiple travelling salesman problem are used to generate pre-solutions, and in this paper, we use the PMX crossover operator and single-point mutation operator as the variation operators to generate presolutions.

3.4. PLS Hybrids with MOALNS (HPLS). MOALNS is hybrids with PLS to resolve the presented combinatorial multiobjective optimization problem in this paper, and it is called Hybrid PLS (HPLS). The framework of HPLS is shown in Algorithm 2.

In HPLS, \( P_p \) refers to the population which will undergo HPLS; different from PLS, here use the presented MOALNS as the local search heuristics. Once a postsolution \( p^g_j \) was input into MOALNS, a temporary set called \( P_T \) will be produced, where \( P_T \) is an efficient set for the input postsolution \( p^g_j \); the external set \( P_E \) will be updated by \( P_T \) during every iteration. MOALNS can be regarded as the problem-specific multiobjective local search for HPLS, while PLS often adapts a problem-specific single-objective local search heuristic.

4. MOALNS for the Postsolution

MOALNS is adopted as the solver for obtaining the postsolution, and its framework is shown in Algorithm 3. It is obvious that MOALNS is an extension of a single-objective ALNS [21].

Similar to ALNS, several different removal and insertion heuristics can be applied in the algorithm; in this work, we propose the following removal and insertion heuristics for MOALNS.

4.1. Removal Heuristics. The removal heuristics developed in this paper can be divided into two different types: removal heuristics for UGV working points and removal heuristics for recharging points. UGV working points and recharging points will be omitted to WP removal and CP removal, respectively, in the rest of the paper.

(1) WP worst removal

Given a UGV working point \( u_j \in P_E \setminus (V_u \cup V^g) \) which is served by the \( nth \) UGV in a postsolution \( p^g_j \), we define the cost of the vertex \( u_j \) as \( r_j \), and \( r_j \) can be calculated by the following equation:

\[ r_j = c(u_{j-1}) - c(u_{j-1}, u_{j+1}) - c(u_{j-1}, u_j). \]  

where \( u_{j+1} \neq v^r_a \) and \( r_j = c(u_{j-1}) \) when \( u_{j+1} = v^r_a \); then, the WP which has the highest cost will be removed from the postsolution according to the following equation:
was used for resolving the generalized traveling salesman problem. This heuristic is inspired by the segment removal developed in [22], which was used for resolving the generalized traveling salesman problem.

$$\tau_j$$ and $$j$$ are used in the postsubsolution removed from the current postsubsolution point at random and removes it from the postsubsolution.

$$\text{arg min}_{1 \leq j \leq |S_p|} r_j; \quad (16)$$

(2) WP random removal

The WP random removal simply selects a UGV working point at random and removes it from the postsolution.

(3) WP segment removal

In this heuristic, a segment is randomly selected and removed from the current postsolution $$p_{g'}$$; this heuristic is inspired by the segment removal developed in [22], which was used for resolving the generalized traveling salesman problem.

(4) CP worst removal

Given a postsolution $$p_{g'}$$, considering the $$j$$ th recharging point in the postsolution $$p_{g'}^n$$, where $$n \in \{1, \cdots, N\}$$ and $$j \in \{1, \cdots, s(n)\}$$, the removal cost of $$u_{g}$$ is defined as the sum of $$r_{g}^j$$ and $$d_{g}^j$$, where $$r_{g}^j$$ and $$d_{g}^j$$ represent the cost of length and time constraint penalty, respectively. For a recharging point $$r_{g}^j$$ in $$p_{g'}^n$$, its cost of length is defined as $$r_{g}^j = c(r_{g}^{j-1}, r_{g}^j)$$.

$$\text{arg min}_{1 \leq j \leq s(n)} \beta d_{g}^j + r_{g}^j, \quad (18)$$

where $$L_{g}^n < \beta < +\infty$$ is a weight parameter. The positive effects of the calculation method of the cost for this heuristic are twofold; firstly, the existence of $$\beta d_{g}^j$$ can make sure that the adjacent recharging points meet the recharging time window constraint, so the feasibility of the postsubsolution can be ensured during iterations. Furthermore, by considering $$r_{g}^j$$ in the removal cost, not only can the total length of
postsubsolution be decreased but also the makespan of it can be reduced since there will be more time redundancy for UGVs to visit UGV working points.

(5) CP random removal

Similar to the WP random removal, one recharging point in $p^u_g$ will be selected and removed randomly in the CP random removal.

4.2. Insertion Heuristics. Two insertion heuristics for WP insertion and one insertion heuristic for CP insertion are proposed to insert WP or CP into the postsubsolution.

(1) WP cheapest insertion

Given a partial postsubsolution $p^n_g$, the vertex $u \in V_g \setminus P_g$ which has the minimum insertion cost and insertion position can be obtained by minimizing

$$I^{nzj}(u) = c(s_{nz}^j, u) + c(u, s_{nz}^{j+1}) - c(s_{nz}^j, s_{nz}^{j+1}).$$

where $s_{nz}$ means the $z$th subsegment in the $n$th UGV path and $z \in \{1, \cdots, L(n) + 1\}$.

(2) WP nearest insertion

Given a partial postsubsolution $p^n_g$, we first calculate the distance between every $u \in V_g \setminus P_g$ and each subsegment $s_{nz}$ according to

$$\text{dist}(u, s_{nz}) = \min_{v \in u} c(u, v).$$

And then, the nearest insertion can be formulated as

$$\arg \min_{u \in V_g \setminus P_g} \min_{1 \leq z \leq L(n)} \min_{s_{nz}} \text{dist}(u, s_{nz}).$$

When the insertion vertex $u$ and insertion subsegment $s_{nz}$ were obtained by using equation (21), the vertex would be inserted into the selected subsegment $s_{nz}$ by using the cheapest insertion.

(3) CP uniform worst insertion

Given a partial postsubsolution $p^n_g$, the insertion cost in this heuristic is the weighted sum of penalty for the recharging time window constraint $I_{nzj}^{1g}$, recharging point insertion cost $I_{nzj}^{2g}$, and total length cost $I_{nzj}^{3g}$, where $n, z, j$ represent the index of UGV, subsegment, and point in the subsegment, respectively.

The insertion cost for the recharging time window constraint is defined as $I_{nzj}^{1g}(u) = p(\tau_n^z, u) + p(u, \tau_n^z)$, where $z \neq L(n) + 1, \forall j \in 1, \cdots, |s_{nz}|$, and $I_{nzj}^{1g}(u) = p(\tau_n^z, u)$ when $z = L(n)$. The factor of the recharging point insertion cost is $I_{nzj}^{2g}(u) = c(s_n^j, u) + c(u, s_n^{j+1}) - c(s_n^j, s_n^{j+1})$, where $j \neq |s_{nz}|$, and $I_{nzj}^{2g}(u) = c(\tau_n^j, u)$ when $j = |s_{nz}|$. The total insertion cost for a charging point in this heuristic can be calculated by

$$I_{nzj}^{3g}(u) = \beta I_{nzj}^{1g}(u) + \lambda_1 I_{nzj}^{2g}(u) + \lambda_2 I_{nzj}^{3g}(u),$$

where $\lambda_1$ and $\lambda_2$ are the weight of CP insertion length cost and path length cost, respectively, and the smaller $\lambda_1$ is, the more minimization of makespan is considered, and the smaller $\lambda_2$ is, the more minimization of path length is considered. In addition, $L(n) < \beta + \infty$, and $\beta I_{nzj}^{1g}(u)$ represents the constraint of recharging time windows. The CP which is to be inserted and the corresponding insertion position can be obtained according to the following equation.

$$\arg \min_{u \in V_g \setminus P_g} \min_{1 \leq z \leq L(n)} \min_{s_{nz}} I_{nzj}^{3g}(u).$$

4.3. Local Optimization. Except for removal and insertion heuristics given above, a local optimization method is also developed to improve the performance of the local solution during every iteration, and it can be regarded as a procedure for recharging points to be reallocated based on the current postsubsolution. Given a postsubsolution $p_g$, the basic idea of local optimization is solution reconstruction which firstly removes all recharging points from the postsubsolution $p_g^n$ and then inserts them into $p_g$ again by means of the local reallocation method.
The insertion cost for recharging points of local optimization contains three components: recharging time window constraint cost $I^{T_{1j}}_w$, path length cost $I^{T_{1j}}_L$, and makespan cost $I^{T_{1j}}_M$, where $n, z, j$ represent the index of UAV, subsegment, and point in the subsegment, respectively. In detail, $I^{T_{1j}}_w$ is defined as equation (24); $I^{T_{1j}}_L = c(t^{\tau_{n_{1j}}} - t^{\tau_{n}}) + c(t^{\tau_{n}} - t^{\tau_{n+1}})$; $I^{T_{1j}}_M = c(t^{\tau_{n_{1j}}} - t^{\tau_{n}})$, where $j \neq L(n)$ and $\tau_{n} = c(t^{\tau_{n-1}} - t^{\tau_{n}})$ when $j = L(n)$; $I^{T_{1j}}_w = (t^{T_{1j}}(u))/v_{\max}$ where $z = L(n) + 1$, $T_{n} + (t^{T_{1j}}(u))/v_{\max} > T_{\max}$, and otherwise $I^{T_{1j}}_w = 0$.

The total insertion cost of local optimization is $I^{T_{1j}}_n = I^{T_{1j}}_w(u) + cI^{T_{1j}}_L(u) + (1 - c)I^{T_{1j}}_M(u)$, where $0 \leq c \leq 1$ is a coefficient which is used to tradeoff the path length and makespan by changing the value of $c$. The total insertion cost is minimized to select the insertion point and position as defined in the following equation.

$$\arg \min_{u \in V, \varphi_1} \min_{1 \leq \kappa \leq ||E||} \min_{1 \leq s \leq |S|} \min_{1 \leq t \leq |T|} I^{T_{1j}}_n.$$  (25)

### Algorithm 3: MOALNS for the postsubsolution

```plaintext
Require: A solution p.
Ensure: A temporary efficient set $P_T$
1: $x = p$
2: initialize($T\cup c\cup w\cup t$)
3: repeat
4: $x' = x$
5: Randomly select a removal heuristic R, and remove several vertices from x by using R.
6: Randomly select an insertion heuristic I, and repair x by using I.
7: local optimization(x)
8: acceptance_criteria($x' \cup x\cup accept$)
9: if accept == 0 then
10: $P_T = [P_T, x]$
11: end if
12: until stop criteria was met
```

### Table 1: Relation between $m$ and $S$.  
| Parameter | Description |
|-----------|-------------|
| $\delta_1$ | $T_{\text{new}}$ is a nondominated solution in the solution set |
| $\delta_2$ | $T_{\text{new}}$ is not dominated in the solution set but is nondominated in the current neighborhood |
| $\delta_3$ | $T_{\text{new}}$ is not a nondominated solution, but it dominates the current solution |
| $\delta_4$ | $T_{\text{new}}$ is not dominated or dominates the current solution and is accepted by the algorithm |
| $\delta_5$ | $T_{\text{new}}$ is dominated by the current solution and is accepted by the algorithm |

4.4. Weight-Adaptive Approach, Acceptance Criteria, and Stopping Criteria. In MOALNS, the score of heuristics is updated during every iteration, and in this paper, we use the weight-adaptive approach in accordance with [23], and particularly, five different senses are considered when updating the score of insertion and removal heuristics, which is shown in Table 1, and $\delta_1 > \delta_2 > \delta_3 > \delta_4 > \delta_5$ is ensured. The standard simulated annealing is adopted as the search framework of the proposed MOALNS, and the acceptance criterion is as follows:

$$P_{\text{accept}} = \min \left\{ \frac{(t^{\tau_1})_{\text{new}}(u)}{e^{\frac{c(t^{\tau_1})_{\text{new}}(u)}} - 1}, 1 \right\},$$  (26)

where $T\tau$ is the temperature and its initial value is set to be $T_{\min}$; $T\tau$ is updated in every iteration of the algorithm as shown in equation (27), where $c$ is the cooling rate.

$$T_{\tau+1} = T_{\tau} - cT_{\tau}.$$  (27)

The stopping criterion of MOALNS is as follows: the temperature will be reheating if no improvement is made to the current solution after the first $R_1$ iterations, and MOALNS ends if no improvement is made to the current solution after the first $R_2$ iterations. And no improvement is that the new solution cannot dominate the current solution.

### 5. Experimental Results

#### 5.1. Performance Metrics

(1) Set coverage (C-metric) [16]: let $A$ and $B$ be two efficient sets of an MOP; $C(A, B)$ is defined as the
Table 3: Parameter configuration of MOALP and the compared algorithms.

| Parameter | \( \epsilon \) | \( \lambda_1 \) | \( \lambda_2 \) | \( \gamma_1 \) | \( \gamma_2 \) | \( R_1 \) | \( R_2 \) | \( \sigma_1 \) | \( \sigma_2 \) | \( \sigma_3 \) | \( \sigma_4 \) | \( \sigma_5 \) |
|-----------|----------------|----------------|----------------|----------------|----------------|----------------|----------------|----------------|----------------|----------------|----------------|----------------|
| Value     | 0.1            | (0.3, 0.4)     | 0.3            | 0.5            | 0.05           | 2\( N_c \)     | 5\( N_c \)     | 10             | 7              | 5              | 3              | 2              |

Table 4: Coverage values (%) between MOALP and MOALNS, MOMAD, and PLS on the problem.

| Instance | MOALNS | MOMAD | PLS |
|----------|--------|-------|-----|
| C(A, B)  | C(B, A) | C(A, B) | C(B, A) |
| 1-1-2-2  | 76.7    | 20.0   | 65.0 |
| 1-2-2-2  | 72.0    | 28.0   | 20.0 |
| 1-3-3-4  | 60.0    | 16.1   | 65.7 |
| 1-4-2-3  | 49.3    | 27.5   | 51.3 |
| 1-5-2-3  | 67.1    | 15.4   | 63.0 |
| 2-1-2-3  | 63.3    | 13.3   | 80.0 |
| 2-2-2-2  | 26.7    | 30.0   | 50.0 |
| 2-3-3-4  | 24.7    | 20.0   | 69.3 |
| 2-4-3-4  | 71.8    | 10.7   | 76.7 |
| 2-5-2-4  | 73.3    | 25.0   | 90.0 |
| 3-1-2-3  | 56.5    | 26.7   | 100.0 |
| 3-2-2-4  | 25.0    | 20.7   | 90.0 |
| 3-3-2-4  | 44.0    | 50.0   | 100.0 |
| 3-4-2-3  | 60.0    | 33.0   | 100.0 |
| 3-5-3-5  | 40.0    | 13.6   | 100.0 |

Table 5: Hypervolume values of MOALP, MOALNS, MOMAD, and PLS.

| Instance | MOALP | MOALNS | MOMAD | PLS |
|----------|-------|--------|-------|-----|
| C(A, B)  | C(B, A) | C(A, B) | C(B, A) |
| 1-1-2-2  | 1.526E+07 | 1.126E+07 | 1.876E+07 | 1.034E+07 |
| 1-2-2-2  | 2.323E+07 | 0.964E+07 | 1.720E+07 | 1.433E+07 |
| 1-3-3-4  | 3.312E+07 | 3.182E+07 | 3.284E+07 | 2.290E+07 |
| 1-4-2-3  | 2.769E+07 | 2.735E+07 | 1.869E+07 | 1.855E+07 |
| 1-5-2-3  | 2.417E+07 | 2.183E+07 | 1.863E+07 | 0.986E+07 |
| 2-1-2-3  | 0.840E+08 | 1.265E+08 | 0.408E+08 | 0.357E+08 |
| 2-2-2-2  | 1.617E+08 | 0.955E+08 | 0.937E+08 | 0.295E+08 |
| 2-3-3-4  | 2.810E+08 | 2.247E+08 | 0.907E+08 | 1.218E+08 |
| 2-4-3-4  | 1.952E+08 | 1.700E+08 | 0.796E+08 | 0.629E+08 |
| 2-5-2-4  | 2.112E+08 | 1.421E+08 | 0.351E+08 | 0.794E+08 |
| 3-1-2-3  | 4.848E+08 | 6.202E+08 | 0.563E+08 | 0.655E+08 |
| 3-2-3-4  | 8.900E+08 | 5.962E+08 | 1.090E+08 | 1.549E+08 |
| 3-3-2-4  | 4.435E+08 | 3.711E+08 | 0.762E+08 | 0.593E+08 |
| 3-4-2-3  | 7.478E+08 | 5.425E+08 | 0.895E+08 | 1.241E+08 |
| 3-5-3-5  | 4.353E+08 | 4.451E+08 | 1.209E+08 | 0.948E+08 |

Table 6: Running time (s) of MOALP, MOALNS, MOMAD, and PLS.

| Instance | MOALP | MOALNS | MOMAD | PLS |
|----------|-------|--------|-------|-----|
|          | C(A, B) | C(B, A) | C(A, B) | C(B, A) |
| 1-1-2-2  | 16     | 16     | 7     | 28  |
| 1-2-2-2  | 19     | 17     | 7     | 31  |
| 1-3-3-4  | 30     | 21     | 11    | 45  |
| 1-4-2-3  | 30     | 23     | 10    | 53  |
| 1-5-2-3  | 36     | 29     | 14    | 62  |
| 2-1-2-3  | 47     | 44     | 36    | 137 |
| 2-2-2-2  | 47     | 38     | 17    | 103 |
| 2-3-3-4  | 77     | 59     | 47    | 173 |
| 2-4-3-4  | 72     | 67     | 45    | 182 |
| 2-5-2-4  | 98     | 92     | 67    | 292 |
| 3-1-2-3  | 100    | 86     | 217   | 475 |
| 3-2-3-4  | 130    | 100    | 110   | 508 |
| 3-3-2-4  | 157    | 121    | 211   | 818 |
| 3-4-2-3  | 146    | 110    | 172   | 555 |
| 3-5-3-5  | 211    | 168    | 558   | 954 |

percentage of the solutions in B that are dominated by at least one solution in A; that is,

\[
C(A, B) = \left| \{ u \in B | \exists v \in A : v \text{ dominates } u \} \right|. \tag{28}
\]

(2) Hypervolume (HV) [16]: let \( r = (r_1, r_2, \ldots, r_m)^T \) be a reference point in the objective space which is dominated by all solutions in a PF approximation \( S \). HV(S) indicates the volume of the region in the objective space which is dominated by \( S \) and dominates \( r \).

\[
HV(S) = \text{volume} \left( \bigcup_{x \in S} [f_1(x), r_1] \times \cdots \times [f_m(x), r_m] \right). \tag{29}
\]

5.2 Test Instances. Since the problem presented in this paper is a new problem in the literature and there are no established benchmark sets yet, we developed some instances for computational studies to test our proposed algorithm. Three different UAV working senses and five UGV working senses were generated, and the details of the tested instances are shown in Table 2.

15 instances were developed by combining UAV and UGV scenarios, and based on these instances, velocity of UAV (\( v_u \)), maximum velocity of UGVs (\( v_{max} \)), number of UAVs (\( M \)), and number of UGVs (\( N \)) were set, and
Figure 4: Continued.
Figure 4: Nondominated solutions resolved by MOALP: (a) nondominated solution 1, (b) nondominated solution 2, (c) nondominated solution 3, and (d) nondominated solution 4.
particularly, \( v_u = 2 \text{ m/s} \) and \( v_{\text{max}} = 2 \text{ m/s} \) were ensured. An example of instance is like 1-5-2-3, which means using UAV scenario 1 and UGV scenario 5; the number of UAVs and UGVs is 2 and 3, respectively.

5.3. Compared Algorithms. To show the performance of MOALP, we compare it with three other algorithms.

1. PLS [15]: this algorithm is designed for combinatorial MOP, and an external set and search set are maintained in the algorithm. In this paper, we use PMX to generate the neighborhood of the presubsolution, and then, local search heuristics proposed in this paper are integrated with PLS to find the efficient set of postsubsolutions.

2. MOMAD [16]: this algorithm is a memetic algorithm which is based on the framework of PLS, and it uses the idea of decomposition to improve the diversity of the attained PF. In the simulation experiments of this work, we implemented the framework of MOMAD and integrated local search heuristics with MOMAD to be adaptive to the problem presented in this paper.

3. MOALNS [20]: multiobjective adaptive large neighborhood search (MOALNS) is an extension of ALNS in multiobjective optimization, its major components include solution destroy, solution repair, weight adaptive, solution acceptance. In this paper, we use a standard Simulated Annealing (SA) as the optimization framework, as well as the proposed removal and insertion heuristics, and also, PMX is implemented to generate the neighborhood of presubsolution.

5.3.1. Parameter Setting. For MOALP, PLS, MOMAD, and MOALNS, we set the size of PL to be 50, and the number of working points to be removed is \( \text{rand}[0.2H, 0.3H] \), where \( H \) is the total number of working points. The number of recharging points to be removed is \( \text{rand}[0.2H, 0.4H] \), where \( N_c \) is the total number of recharging points in the current solution. Table 3 shows the parameter setting of MOALP, where \( \lambda_l \) has two values: 0.3 and 0.4, which means there exists two insertion heuristics for recharging points, while there have been two insertion heuristics for working points; four insertion heuristics are implemented in the algorithm since we combine a working point insertion with a recharging point insertion heuristic as one integral insertion heuristic; similarly, there have been 6 removal heuristics since three working removal and two recharging removal heuristics are implemented in the algorithm.

5.4. Results. Each algorithm runs independently for 5 times, and we use the average of indicators as the compared data resource. The C-metric values of all the comparisons on each instance are reported in Table 4. The hypervolume values of the efficient sets produced by these algorithms are plotted in Table 5. The average running time consumed by these algorithms is given in Table 6.

MOALP outperforms PLS on all test instances in the C-metric, and on 11 out of 15 test instances, \( C(A, B) \) equals to 0.0% which means that solutions in the efficient set obtained by PLS cannot dominate that of MOALP at least once. MOALP has the better C-metric than MOMAD except for 1-2-2-2, and when compared with MOALNS, MOALP performs better than MOALNS on 13 out of 15 instances; furthermore, on 11 test instances, the C-metric of MOALP is more than 10% higher than that of MOALNS.

On 11 out of 15 test instances, the hypervolume values of MOALP are shown in Table 5; it can be seen that MOALP performs better than all compared algorithms, and except for instances 2-1-2-3, 3-1-3-2, and 3-5-3-5, MOALP outperforms MOALNS in the hypervolume indicator.

Regarding the running time, the average running time consumed by MOALP is lower than that of PLS, and compared with MOMAD, the running time of MOALP is shorter on large-scale instances, and PLS runs for the longest time in all test instances.

In Figure 4, four nondominated solutions resolved by our proposed algorithm MOALP are presented, and it can be seen that UAVs and UGVs can work collaboratively to enable them work in persistent tasks.

5.5. Result Visualization. Figure 4 shows four nondominated solutions of instance 1-4-2-3, obtained by using MOALP; we can see that there exist two UAV paths and three UGV paths in the solution; also, the start point of UGV is \((1000, 1500)\), the objectives of the four solutions are \([2.548E + 04, 4439], [2.696E + 04, 4433], [2.798E + 04, 4264]\), and \([2.969E + 04, 3505]\), respectively.

6. Conclusions and Future Work

In this paper, we propose a multiobjective UAV/UGV collaboration path planning problem, which firstly considers multitasks for UGVs in persistent tasks. To resolve the presented problem, MOALP is proposed. The core of the algorithm is the hybridization of the Pareto local search and multiobjective large neighborhood search, and for the MOALNS in MOALP, we develop a mechanism for removal and insertion heuristics, which contains several removal and insertion heuristics for UGV working points and recharging points; besides, a local optimization method is also implemented after each iteration of MOALNS to improve the performance of the algorithm. The experiments have been conducted to verify MOALP. The comparisons of MOALP with three other algorithms (PLS, MOMAD, and MOALNS) show the efficiency of MOALP in 15 instances.

An important avenue of the proposed algorithm for future work is to extend MOALP to resolve more CMOPs, such as mTSP and mKP, and it also can be used to resolve more problems in applications of UAV/UGV collaborative systems.

Data Availability

Data sharing is not applicable to this article as no datasets were generated or analysed during the current study.
Conflicts of Interest

The authors declare no conflict of interest in preparing this article.

Acknowledgments

The authors acknowledge the Key Research and Development Projects in Shaanxi Province (Grant Nos. 2015KTZDGY-02-01, 2020GY-188, and 2020NY-148).

References

[1] K. Sunder and S. Rathinam, "Route planning algorithms for unmanned aerial vehicles with refueling constraints," in 2012 American Control Conference (ACC), pp. 3266–3271, Montreal, QC, CA, 2012.

[2] K. Yu, A. K. Budyhija, and P. Tokekar, "Algorithms for routing of unmanned aerial vehicles with mobile recharging stations," in 2018 IEEE International Conference on Robotics and Automation (ICRA), pp. 1–5, Brisbane, QLD, Australia, 2018.

[3] P. Maini and P. Sujit, "On cooperation between a fuel constrained UAV and a refueling UGV for large scale mapping applications," in 2015 International Conference on Unmanned Aircraft Systems (ICUAS), pp. 1370–1377, Denver, CO, USA, 2015.

[4] Z. Jin, T. Shima, and C. J. Schumacher, "Optimal scheduling for refueling multiple autonomous aerial vehicles," IEEE Transactions on Robotics, vol. 22, no. 4, pp. 682–693, 2006.

[5] Z. Jin, T. Shima, and C. J. Schumacher, "Scheduling and sequence reschedule for autonomous aerial refueling of multiple UAVs," in 2006 American Control Conference, p. 6, Minneapolis, MN, USA, 2006.

[6] N. Mathew, S. L. Smith, and S. L. Waslander, "Multirobot rendezvous planning for recharging in persistent tasks," IEEE Transactions on Robotics, vol. 31, no. 1, pp. 128–142, 2015.

[7] F. Ahmed and K. Deb, "Multi-objective optimal path planning using elitist non-dominated sorting genetic algorithms," Soft Computing, vol. 17, no. 7, pp. 1283–1299, 2013.

[8] Z. Zhu, J. Q. Li, F. Wang, and Q. Zhang, "Global path planning of wheeled robots using multi-objective memetic algorithms," Integrated Computer-Aided Engineering, vol. 22, no. 4, pp. 387–404, 2015.

[9] A. Hidalgo-Paniagua, M. A. Vega-Rodriguez, A. Miguel, J. Ferruz, and N. Pavón, "Solving the multi-objective path planning problem in mobile robotics with a firefly-based approach," Soft Computing, vol. 21, no. 4, pp. 949–964, 2017.

[10] Y. Xue, "Mobile robot path planning with a non-dominated sorting genetic algorithm," Applied Sciences, vol. 8, no. 11, article 2253, 2018.

[11] Y. Xue and J. Sun, "Solving the path planning problem in mobile robotics with the multi-objective evolutionary algorithm," Applied Sciences, vol. 8, no. 9, article 1425, 2018.

[12] M. Xia, L. Weng, J. Liu, and Y. Wang, "Robot path planning based on multi-objective optimization with local search," Journal of Intelligent Fuzzy Systems, vol. 35, no. 2, pp. 1755–1764, 2018.

[13] 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.

[14] L. Paquette and T. Stützle, "A two-phase local search for the biobjective traveling salesman problem," in Lecture Notes in Computer Science, pp. 479–493, Springer, 2003.

[15] T. Lust and J. Teghem, "Two-phase Pareto local search for the biobjective traveling salesman problem," Journal of Heuristics, vol. 16, no. 3, pp. 475–510, 2010.

[16] L. Ke, Q. Zhang, and R. Battiti, "Hybridization of decomposition and local search for multiobjective optimization," IEEE Transactions on Cybernetics, vol. 44, no. 10, pp. 1808–1820, 2014.

[17] X. Cai, H. Sun, Q. Zhang, and Y. Huang, "A grid weighted sum Pareto local search for combinatorial multi and many-objective optimization," IEEE Transactions on Cybernetics, vol. 49, no. 9, pp. 3586–3598, 2019.

[18] P. Schaus and R. Hartert, "Multi-objective large neighborhood search," in Lecture Notes in Computer Science, pp. 611–627, Springer, 2013.

[19] F. Lehuédé, R. Masson, S. Parragh, O. Péton, and F. Tricoire, "A multi-criteria large neighbourhood search for the transportation of disabled people," Journal of the Operational Research Society, vol. 65, no. 7, pp. 983–1000, 2013.

[20] A. Rifai, H. Nguyen, S. Z. M. Dawal, O. Péton, and F. Tricoire, "Multi-objective adaptive large neighborhood search for distributed reentrant permutation flow shop scheduling," Applied Soft Computing, vol. 40, pp. 42–57, 2016.

[21] L. P. Cota, F. G. Guimarães, S. Z. M. Dawal et al., "An adaptive multi-objective algorithm based on decomposition and large neighborhood search for a green machine scheduling problem," Swarm and Evolutionary Computation, vol. 51, article 100601, 2019.

[22] S. L. Smith and F. Imeson, "GLNS: an effective large neighborhood search heuristic for the generalized traveling salesman problem," Computers and Operations Research, vol. 87, pp. 1–19, 2017.

[23] S. Ropke and D. Pisinger, "An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows," Transportation Science, vol. 40, no. 4, pp. 455–472, 2006.