Scheduling unmanned aerial vehicle and automated guided vehicle operations in an indoor manufacturing environment using differential evolution-fused particle swarm optimization

Khosiawan, Yohanes; Khalfay, Amy; Nielsen, Izabela Ewa

Published in:
International Journal of Advanced Robotic Systems

DOI (link to publication from Publisher):
10.1177/1729881417754145

Publication date:
2018

Document Version
Publisher's PDF, also known as Version of record

Research Article

INTERNATIONAL JOURNAL OF ADVANCED ROBOTIC SYSTEMS

International Journal of Advanced Robotic Systems
January-February 2018: 1–15

DOI: 10.1177/1729881417754145
journals.sagepub.com/home/arx
SAGE

Scheduling unmanned aerial vehicle and automated guided vehicle operations in an indoor manufacturing environment using differential evolution-fused particle swarm optimization

Yohanes Khosiawan[1], Amy Khalfay[2] and Izabela Nielsen[1]

Abstract

Intelligent manufacturing technologies have been pursued by the industries to establish an autonomous indoor manufacturing environment. It means that tasks, which are comprised in the desired manufacturing activities, shall be performed with exceptional human interventions. This entails the employment of automated resources (i.e. machines) and agents (i.e. robots) on the shop floor. Such an implementation requires a planning system which controls the actions of the agents and their interactions with the resources to accomplish a given set of tasks. A scheduling system which plans the task executions by scheduling the available unmanned aerial vehicles and automated guided vehicles is investigated in this study. The primary objective of the study is to optimize the schedule in a cost-efficient manner. This includes the minimization of makespan and total battery consumption; the priority is given to the schedule with the better makespan. A metaheuristic-based methodology called differential evolution-fused particle swarm optimization is proposed, whose performance is benchmarked with several data sets. Each data set possesses different weights upon characteristics such as geographical scale, number of predecessors, and number of tasks. Differential evolution-fused particle swarm optimization is compared against differential evolution and particle swarm optimization throughout the conducted numerical simulations. It is shown that differential evolution-fused particle swarm optimization is effective to tackle the addressed problem, in terms of objective values and computation time.

Keywords

Unmanned aerial vehicle, automated guided vehicle, scheduling, metaheuristic, DE and PSO hybrid

Date received: 28 August 2017; accepted: 19 December 2017

Topic: Special Issue—Robot Path Planning Design and Implementation in Manufacturing, Healthcare and Service Systems
Topic Editor: Nak-Young Chong
Associate Editor: Hamed Fazlollahtabar

Introduction

Intelligent manufacturing environment has been an emerging topic in regard to the rise of Industry 4.0 concept across various domains.[1–3] It creates a smart factory, where automation of the manufacturing operations is the key factor. This automation enables the minimization of human–labor interventions on tedious, time-consuming, and sometimes hazardous jobs. Machines (resources) and

[1] Department of Materials and Production, Aalborg University, Aalborg, Denmark
[2] School of Computing, Mathematics and Digital Technology, Manchester Metropolitan University, Manchester, UK

Corresponding author:
Yohanes Khosiawan, Department of Materials and Production, Aalborg University, Fibigerstræde 16, Aalborg 9220, North Jutland, Denmark.
Email: yok@mp.aau.dk

robots (agents) can autonomously perform the given tasks in an efficient manner. This could mean the optimization in terms of time, energy consumption, or other objectives.

A pilot study by Khosiawan et al.[4] on the task scheduling system for unmanned aerial vehicle (UAV) operations in indoor environment has opened the gate toward the executions of material handling and visual inspection by UAVs. Task automation by employing UAVs gives a remarkable value in terms of time efficiency and capability. A UAV has more freedom in gathering images from various angles for an inspection task. On the other hand, a heavy material handling task is suitable for an automated guided vehicle (AGV) to perform. This becomes the motivation of the collaborative operations among UAVs and AGVs to perform multiple tasks in an indoor manufacturing environment, which is addressed in this article.

In the past, gradient-based optimization methods were used, and they only guarantee convergence toward local optima.[5] Furthermore, non-convex problems cannot be solved easily by those methods. In contrast, metaheuristic algorithms can explore regions in the search space at an affordable computation time, and does not tend to get trapped at a local optimum due to inbuilt escape mechanisms.

Particle swarm optimization (PSO) and differential evolution (DE) are two prominent metaheuristic algorithms which are popularly used by researchers in various optimization fields. Nilakantan et al.[6] implemented both approaches and found that DE could improve the energy efficiency of the robotic assembly lines in the manufacturing environment. A comprehensive study by Price et al.[7] shows that DE usually gives the best result with a longer computation time compared to PSO. Supported with a priori studies by Khosiawan et al.[4,8] there is a room of improvement for the explorative characteristic in the body of PSO framework. This is aligned with the reported works in the existing literatures. In this study, an approach of fusing DE's explorative characteristic into PSO is proposed, and this gives birth to DE-fused PSO (DEFPSO).

In regard to the addressed problem, DEFPSO is compared against DE and PSO in three aspects: makespan of the produced schedule, total battery consumption of the produced schedule, and the computation time. Correspondingly, the benchmark is done by using task data sets (which comprise tasks for UAVs and AGVs) with different weights on the following characteristics: geographical scale, number of predecessors, and number of tasks. These data sets are generated by the authors based on test flights at the laboratory and a field study of an indoor industry site. The investigation is remarked with a satisfactory performance of DE in terms of the pursued objectives: makespan and total battery consumption (of the optimized schedule), and the respective computational time.

The main contributions of this study are described as follows:

(i) This study developed a mathematical formulation of the addressed problem of collaborative UAV–AGV operations in an indoor manufacturing environment.
(ii) This study developed a DEFPSO which is measured as a methodology which generally outperforms DE and PSO. The measured characteristics are the solution's makespan and total battery consumption, together with its computation time.

The content of the article is organized as follows. The "Literature review" section describes the literature on metaheuristic-based approaches toward optimization problems, focusing on scheduling. The "Problem definition" section outlines the formal description of the problem, accompanied with the representative mathematical formulation. In The "Methodology" section, the proposed approach with heuristic and metaheuristic algorithms are described in detail. The "Numerical simulations" section presents the results and analysis of the benchmark of DE, PSO, and DEFPSO upon different data sets. A summary of the conducted study is then provided in the "Conclusion" section.

Literature review

The field of automation has been continuously explored in the area of healthcare and manufacturing facilities.[4,9–11] The desired autonomous operations demand a planning system to generate a schedule of task executions in a cost-efficient manner.[12] In such a system, numerous constraints according to the operational environment need to be taken into account—in connection with the respective objective function. As the scale of the problem increases—in terms of the number of agents, tasks, and constraints—the computation time grows exponentially. This entails the employment of a scheduling methodology which is able to search the optimum solution in the solution space whilst balancing time efficiency.

The paradigm of such an effort is known as optimization, and heuristic & metaheuristic algorithms have come to researchers' and practitioners' rescue to tackle various forms of optimization problem in the last two decades.[5] Among others, PSO and DE are the prominent metaheuristic algorithms in the optimization field.

Particle swarm optimization

PSO[13] is a metaheuristic-based optimization method which emphasizes the collaborative learning through the individual experience and social interactions among the particles during the search. The algorithm allows particles to go through different directions in the search space, while enabling them to adjust the direction to some extent toward the (global) best one so far. This behavior is enabled by the role of local and global best particles, respectively.

Two major updates in the PSO procedure are velocity and position updates, as depicted in equations (1) and (2). Velocity updates utilize parameters learning coefficients c_1 and c_2, together with velocity coefficients u_1 and u_2. They will determine the degree of learning toward the local and global best particles ($^{lo}P_i^t$ and G^t). Correspondingly, the updated velocity will be used to modify the position. This process is done in every generation t and every particle i.

$$v_i^{t+1} = v_i^t + \underbrace{c_1 * [u_1 * (^{lo}P_i^t - P_i^t)]}_{\text{cognitive part}} + \underbrace{c_2 * [u_2 * (G^t - P_i^t)]}_{\text{social part}} \quad (1)$$

$$P_i^{t+1} = P_i^t + v_i^{t+1} \quad (2)$$

The initial swarm plays a significant role in giving good starting points.[14] The comprised initial particles are generated through intuitive heuristics, which are inspired by the characteristics of the problem. The more distinct the initial particles produced, the more explorative the search will be— right from the beginning of the search.

Zhu et al.[15] and Mahmoudzadeh et al.[16] have investigated that the incorporation of PSO is efficient for the problem of task assignment toward multiple robots. In connection with this article, such a problem is NP-hard natured. Deciding whether there exists a schedule where all tasks are executed is an NP-complete problem. However, when one looks for an optimum schedule (e.g. with the minimum makespan), it becomes an NP-hard problem—NP-hard problem is at least as hard as NP-complete problem.

Differential evolution

Along with PSO, DE belongs to the family of swarm intelligence algorithms. With the same principle of initial swarm, the performance characteristics can be different. While DE's computation time may not be the fastest one, it usually produces the best result among several other algorithms.[6,7]

Krömer et al.[17] has applied DE to the independent tasks scheduling problem on heterogeneous distributed environments, which is an NP-complete problem. Without any tuning, it managed to optimize schedules to a certain degree. The authors found that using scheduling heuristics for generating initial population for DE delivers good results. Moreover, Nearchou and Omirou[18] employed DE for solving NP-hard scheduling problems. The authors concluded that with a slight modification of encoding scheme within DE, the performance was found substantially superior than the original form's.

The existing works mentioned above suggested the insights of approaching scheduling problems with metaheuristic algorithms, and they can be effective to tackle different combinatorial optimization problems. Furthermore, some reported studies[19,20] have performed a comparative evaluation of several metaheuristic algorithms (e.g. genetic algorithm, ant colony optimization, artificial bee colony, and PSO) on tackling different optimization problems, which once again exhibits metaheuristics as the major player in the game.

Metaheuristic algorithms are well known for their adaptability, in connection with the type of the problem, and its simplicity during implementation. Metaheuristics are general purpose, they are not problem specific. Their role is to guide a lower level heuristic, encompassing both intensification (to concentrate on high quality areas of solution space) and diversification (to allow the freedom to explore unvisited areas of search space) characteristics.

Table 1. Task type.

Task type	Description	Payload (g)	Agent
0	Ground material delivery	201–2000	AGV
1	Air material delivery	1–200	UAV
2	Visual inspection	—	UAV

Table 2. Example of task data set with 10 tasks.

Task ID	Start position	End position	Task type	Payload	Predecessors (Task IDs)
1	e1	a1	1	58	-
2	b3	b3	2	0	1
3	c4	c4	2	0	4
4	d2	b2	1	11	7
5	d4	d4	2	0	-
6	f2	b3	1	66	7
7	a1	a1	2	0	1;5;9
8	f0	d0	0	455	9
9	b0	a0	0	1396	-
10	b3	f1	1	6	-

Problem definition

Automation in the manufacturing environment has been a consistently growing interest in both research and implementation. One of the technological advancements is the usage of robot agents such as UAV and AGV on the shop floor. In this study, the problem focuses on scheduling the task executions by multiple UAVs and AGVs in an indoor manufacturing environment. The types of task during the operations are listed in Table 1.

An instance of a task data set is depicted in Table 2. It comprises attributes such as task identifier, start position, end position, task type, payload, and predecessor(s).

The given tasks in the data set shall be performed by the agents (i.e. UAVs and AGVs) in an efficient way (according to the defined objective, e.g. makespan). The execution manner in regards to the available agents, time, and other resources (e.g. machine at a particular position) is planned

```
Schedule: {
      101=[R1_TO_to_e1(0-19), 1(19-66), 7(66-76), a1_to_d2(76-91),
4(91-132), b2_to_b3(132-134), 2(134-144)],
      901=[G1_OUT_to_b0(0-7), 9(7-46)],
      102=[R1_TO_to_d4(0-19), 5(19-29), d4_to_b3(29-35), 10(35-
81), f1_to_f2(81-82), 6(82-125), b3_to_c4(125-132), 3(132-142)],
      902=[WoG(0-34), G2_OUT_to_f0(34-46), 8(46-83)]
}
```

Figure 1. An instance of a schedule.

in a schedule. The example of a schedule representation is depicted in Figure 1.

The objective of the optimization problem in this study is to generate a schedule of the given tasks which optimizes the makespan and the battery consumption, whilst balancing time efficiency. The optimization process is driven by this objective, while being bounded to the defined constraints (e.g. agent's battery capacity, precedence relationship between tasks, and geographical space limitation).

The integer programming formulation of the problem in this article is described as follows.

Mathematical formulation

In this section, the mathematical formulation of the novel problem of collaboratively scheduling UAV and AGV operations is described. We begin with a set of UAVs $U = \{1, ...u\}$ and AGVs $A = \{1, ...a\}$. The UAVs and AGVs will be scheduled to complete a set of tasks $N = \{1, ...n\}$. Each UAV u and AGV a has a level of energy which can be replenished by a recharge station denoted as the set $R = \{1, ...r\}$. Each recharge station r has a number of slots/maximum capacity given by r_i.

The objective function is to minimize the makespan of the schedule whilst serving all jobs as described in equation (3)

$$\text{Minimize } T \tag{3}$$

Subject to the following constraints

$$\sum_{u \in U} x_{u,n} + \sum_{a \in A} x_{a,n} \geq 1 \quad \forall n \in N \tag{4}$$

Equation (4) illustrates that each task is executed only once, by either a UAV or AGV. Here, $x_{u,n}$ equals one if task n is allocated to UAV u, and $x_{a,n}$ equals one if task n is allocated to AGV a.

Equation (5) guarantees that a recharge station comprises at least one recharge slot or more, where r_i equals the number of recharge slots available at r

$$r_i \geq 1 \quad \forall r \in R \tag{5}$$

Equation (6) states that a recharge slot can only be occupied by a UAV or AGV at any time. Here, $r_{t,u}$ and $r_{t,a}$ are equal to one if recharge slot r is occupied by either u or a at time t

$$r_{t,u} + r_{t,a} \leq 1 \quad \forall u, \in U, a \in A \tag{6}$$

Equation (7) ensures that a UAV or AGV may wait on ground if the recharge slot r is occupied. We use $w_{t,u,r}$ and $w_{t,a,r}$ equal to one if at time t a UAV u or AGV a is waiting for recharge slot r

$$w_{t,u,r} + w_{t,a,r} \leq r_{t,u} + r_{t,a} \forall u \in U, a \in A, t \in T \tag{7}$$

Equation (8) shows that each task n has a begin time a_n and a duration w_n which when summed equals the end time of the task z_n

$$z_n = a_n + w_n \quad \forall n \in N \tag{8}$$

Equation (9) shows that each UAV or AGV can only execute one task at a time. Let $x'_{n,n',u}$ or $x'_{n,n',a}$ equals one if both task n and n' are allocated to either UAV u or AGV a

$$a_n \geq z_{n'} || z_n \leq a_{n'} \quad \forall n, n' \in N, n \neq n', x'_{n,n',u} + x'_{n,n',a} = 1 \tag{9}$$

Equations (10) and (11) state that if two tasks are both scheduled, $x'_{n,n',u} = 1$ or $x'_{n,n',a} = 1$ to be completed by UAV u or by AGV a, then both tasks are individually allocated to u and a also

$$x'_{n,n',u} \geq x_{n,u} + x_{n',u} - 1 \quad \forall n \in N, u \in U \tag{10}$$

$$x'_{n,n',a} \geq x_{n,a} + x_{n',a} - 1 \quad \forall n \in N, a \in A \tag{11}$$

Equations (12) and (13) show that if two tasks are scheduled to the same UAV or AGV, then each task belonging to this pairing is also scheduled

$$x'_{n,n',u} \leq x_{n,u} \quad \forall n, n' \in N, u \in U \tag{12}$$

$$x'_{n,n',a} \leq x_{n,a} \quad \forall n, n' \in N, a \in A \tag{13}$$

Equation (14) shows that if two tasks are allocated to the same resource; that is, $q_{n,n'} = 1$, then both tasks are allocated either to UAV u or to AGV a

$$q_{n,n'} = \sum_{u \in U} x'_{n,n',u} + \sum_{a \in A} x'_{n,n',a} \quad \forall n, n' \in N, n \neq n' \tag{14}$$

Equation (15) illustrates that a recharge can only happen to UAV u before executing task n if task n is allocated to u

$$y_{n,u} \leq x_{n,u} \quad \forall n \in N, u \in U \tag{15}$$

Equation (16) states that a recharge can only happen to AGV a before executing task n if task n is allocated to a

$$y_{n,a} \leq x_{n,a} \quad \forall n \in N, a \in A \tag{16}$$

Constraints (17) and (18) ensure that the start time a_n and the finish time z_n lie within the boundaries of the time window in which the job must be completed $[n_{\min}, n_{\max}]$

$$n_{\min} \leq a_n \quad \forall n \in N \tag{17}$$

$$z_n \leq n_{\max} \quad \forall n \in N \tag{18}$$

Constraint (19) guarantees that a task n may only begin once all of its predecessors have been completed.

$$a_n \geq z_{n'} \quad \forall n \in N, n' \in \sigma_n \tag{19}$$

Constraints (20)–(23) describe the conditions which trigger a recharge for each UAV and AGV. Here, $b_{n,u}$ and $b_{n,a}$ denote the level of battery remaining before executing task n, and w_n is the duration of task n. In addition, c_{r,e_n} is the travel time to recharge station r from position e_n which is the end position of task n

$$b_{n,u} - (w_n + c_{r,e_n})x_{n,u} \geq 0 \quad \forall n \in N \tag{20}$$

$$b_{n,a} - (w_n + c_{r,e_n})x_{n,a} \geq 0 \quad \forall a \in A \tag{21}$$

$$b_{n',u} - (z_n - a_{n'}) - c_{r,e_n} \geq 0 \quad \forall n \in N, u \in U, v_{n,n',u} = 1 \tag{22}$$

$$b_{n',a} - (z_n - a_{n'}) - c_{r,e_n} \geq 0 \quad \forall n \in N, a \in A, v_{n,n',a} = 1 \tag{23}$$

Constraint (24) ensures that no tasks are executed at the same time in the same place

$$a_n \geq z_{n'} || z_n \leq a_{n'} \quad \forall n, n' \in N, n \neq n', \frac{min(s_n, s_{n'})}{max(s_n, s_{n'})} = 1 \tag{24}$$

Constraints (25) and (26) show what recharge does to the battery level and how the battery is consumed. Here, the battery level is equal to α minus the travel time from the end position of task n, s_n to recharge station r.

$$b_{n,u} = (\alpha - c_{s_n,r}) \quad \forall n, n' \in N, U_{n,n',u} = 1 \tag{25}$$

$$b_{n,a} = (\alpha - c_{s_n,r}) \quad \forall n, n' \in N, U_{n,n',a} = 1 \tag{26}$$

Constraints (27) and (28) illustrate that the battery level before executing n is equal to the battery level before executing n' minus the difference between the start times of n and n'. This is subject to the condition that either $V_{n,n',u} = 1$ or $V_{n,n',a} = 1$, which states that no recharge occurs between the execution of these tasks

$$b_{n,u} = b_{n',u} - (a_n - a_{n'}) \quad \forall n, n' \in N, V_{n,n',u} = 1 \tag{27}$$

$$b_{n,a} = b_{n',a} - (a_n - a_{n'}) \quad \forall n, n' \in N, V_{n,n',a} = 1 \tag{28}$$

Conversely, constraints (29) and (30) describe the condition where a recharge does occur between the execution of two tasks. If a recharge occurs then either $U_{n,n',u} = 1$ or $U_{n,n',a} = 1$. Here the variable $O_{n,n'} = 1$ only if task n' precedes task n, where $y_{n,u}$ and $y_{n,a}$ specifies a recharge before n is executed

$$U_{n,n',u} \geq O_{n,n'} + y_{n,u} - 1 \quad \forall n, n' \in N, u \in U \tag{29}$$

$$U_{n,n',a} \geq O_{n,n'} + y_{n,a} - 1 \quad \forall n, n' \in N, a \in A \tag{30}$$

Constraints (31) and (32) ensure that either a recharge happens or it does not before task n is executed

$$U_{n,n',u} + V_{n,n',u} \leq 1 \quad \forall n, n' \in N, u \in U \tag{31}$$

$$U_{n,n',a} + V_{n,n',a} \leq 1 \quad \forall n, n' \in N, a \in A \tag{32}$$

Constraints (33) and (34) ensure that if no recharge happens before the execution of task n, then the value that either $V_{n,n',u}$ or $V_{n,n',a}$ takes is equal to one. This value is greater than or equal to the sum of $O_{n,n'} = 1$ (if task n is executed after another task), plus $x_{n,u} = 1$ or $x_{n,a} = 1$ (if n is allocated to u or a respectively), minus $y_{n,u} = 0$ (if a recharge has occurred), minus 1

$$V_{n,n',u} \geq O_{n,n'} + x_{n,u} - y_{n,u} - 1 \quad \forall n, n' \in N, u \in U \tag{33}$$

$$V_{n,n',a} \geq O_{n,n'} + x_{n,a} - y_{n,a} - 1 \quad \forall n, n' \in N, a \in A \tag{34}$$

Constraints (35) and (36) set the battery level of the UAVs and AGVs at the beginning of the planning horizon to be fully charged. Here α is the maximum level of charge, and t_{s_o,s_n} is the travel time from position s_o to the start position of task n, s_n

$$b_{n,u} = \alpha - t_{s_o,s_n} \quad \forall n \in N, p_n = 0, x_{n,u} = 1, u \in U \tag{35}$$

$$b_{n,a} = \alpha - t_{s_o,s_n} \quad \forall n \in N, p_n = 0, x_{n,u} = 1, a \in A \tag{36}$$

Constraints (37) and (38) determine the start time of the first task to be executed, which is equal to the travel time from the starting position s_o to the start position of task n, s_n

$$t_{s_o,s_n} \leq a_n \quad \forall n \in N, p_n = 0, x_{n,u} = 1 u \in U \tag{37}$$

$$st_{s_o,s_n} \leq a_n \quad \forall n \in N, p_n = 0, x_{n,a} = 1 a \in A \tag{38}$$

Operational precedence constraints are stated in equations (39) and (40). Here f_n denotes the start time of a task that is operationally preceded by task n and $f''_{n,n'}$ is equal to one if the start time of task n is less than the end time of task n' and M is a large constant value

$$O_{n,n'} \geq -a_n + f_{n'} - f''_{n,n'} 2M - 2M(2 - x_{n,u} - x_{n',u}) + 1 \quad \forall n, n' \in N, u \in U \tag{39}$$

$$O_{n,n'} \geq -a_n + f_{n'} - f''_{n,n'} 2M - 2M(2 - x_{n,a} - x_{n',a}) + 1 \quad \forall n, n' \in N, a \in A \tag{40}$$

Constraint (41) illustrates that two tasks may only happen sequentially if they are scheduled to the same UAV or AGV

$$O_{n,n'} \leq q_{n,n'} \quad \forall n, n' \in N \tag{41}$$

Constraints (42) and (43) state that the start time of a task preceded by n is equal to the minimum value of $f'_{n,n',u}$ which is equal to $a_{n'}$ unless $a_{n'} \leq z_n$

$$f_n = \min_{n' \in N, u \in U} f'_{n,n',u} \quad \forall n \in N \tag{42}$$

$$f_n = \min_{n' \in N, a \in A} f'_{n,n',a} \quad \forall n \in N \tag{43}$$

Constraints (44)–(46) state that if two tasks are assigned to the same UAV u or AGV a, then the start time of task n will be equal to the end time of task n', this allows for time continuity to ensure a task n cannot begin until the task that operationally precedes it, n', has been completed.

$$f'_{n,n',u} = a_n + f''_{n,n'}M + (2 - x_{n,u} - x_{n',u})M \quad \forall n, n' \in N, u \in U \tag{44}$$

$$f'_{n,n',a} = a_n + f''_{n,n'}M + (2 - x_{n,a} - x_{n',a})M \quad \forall n, n' \in N, a \in A \tag{45}$$

$$f''_{n,n'} = 1 - (a_n \geq z_{n'}) \quad \forall n, n' \in N \tag{46}$$

Constraints (47)–(49) state that a task can be operationally preceded by at most one task, where p_i

$$p_n \leq 1 \quad \forall n \in N \tag{47}$$

$$\sum_{n \in N} O_{n,n'} \leq 1 \quad \forall n' \in N \tag{48}$$

$$p_n = \sum_{n' \in N} O_{n,n'} \leq 1 \quad \forall n \in N \tag{49}$$

Equation (50) and (51) illustrate that there can be no self operational precedence and no cyclic operational precedence

$$O_{n,n} = 0 \quad \forall n \in N \tag{50}$$

$$O_{n,n'} + O_{n',n} \leq 1 \quad \forall n, n' \in N \tag{51}$$

Equation (52) states that no task is completed after the total makespan of the solution

$$z_n \leq T \quad \forall n \in N \tag{52}$$

The UAV–AGV operations are conducted in the indoor environment, where the map is provided as an input for the scheduling process. A tractable yet realistic position mapping is used, where waypoints are connected with (one-way) directed paths and the whole graph is fully connected. The waypoints in the air are for the UAV operations, while the ones on the ground are mainly for the AGV operations (except the UAV recharge station). In this study, the transmission of the command to the agent is done through a component which verifies that the command does not yield a geographical conflict (path collision) with the currently being executed ones. Otherwise, the transmission is delayed until this constraint is satisfied.

To briefly summarize equations (5)–(52), there are a set of tasks which must be completed, some of which require an AGV (visual inspection) and others a UAV (ground material delivery) to satisfy each task's demand. The objective of the problem is to service all tasks (just once) whilst minimizing the scheduling horizon makespan.

The UAVs and AGVs have levels of charge which can be replenished at a recharge station. At the beginning of the planning horizon, it is assumed that each UAV and AGV has maximum charge. The level of charge that a UAV or AGV has is reduced when travelling to perform tasks. Each recharge station has a maximum number of ports, and therefore, sometimes a UAV or AGV may have to wait until a port becomes available in order to recharge. A recharge must happen if the UAV or AGV does not have enough power to serve the next task.

Each UAV or AGV may only serve a single task at a time, and the end time of a task is equal to the beginning time plus the task duration. In addition, a task has a time window in which the service of a task must begin, and a set of predecessor tasks which must be completed before the service of a task begins. Travel time between locations of successive tasks is also accounted for to ensure time continuity, so the beginning time of a task, is equal to the end time of the previous task, plus the travel time between the locations (if a recharge is not needed). There are also geographical constraints associated with the UAVs and AGVs, such that at no time during the schedule execution are two UAVs or AGVs situated at the same location.

Methodology

Constructive heuristic to create a schedule from a task sequence

A constructive heuristic has been introduced in a study by Khosiawan et al.[4] for creating a schedule of UAV operations from a task sequence. Given a task sequence, each comprised task is put into the schedule sequentially in the earliest available time manner. This constructive heuristic has been modified in this study to be able to schedule the tasks in cooperative UAV–AGV operations. As depicted in Table 1, a task needs to be executed by either a UAV or an AGV. The heuristic is modified to construct a schedule with the awareness of the type of the available agents (i.e. UAV or AGV) and the corresponding type of task (i.e. visual inspection by UAV, light material handling by UAV, and heavy handling by AGV). In principle, the available agents for a particular task are filtered based on the type of the task; for example, only UAVs will be checked as the prospective performing agents for the air material delivery. Furthermore, the recharge station selection is also

Table 3. Priority rules.

No	Priority rule	Task sequence (Task ID)									
1	Minimum number of cumulative predecessors	1	5	9	10	2	8	7	4	6	3
2	Minimum number of immediate predecessors	1	5	9	10	2	3	4	6	8	7
3	Maximum number of cumulative successors	1	9	5	7	4	2	3	6	8	10
4	Maximum number of immediate successors	1	7	9	4	5	2	3	6	8	10
5	Maximum task execution time	1	10	6	4	9	8	2	3	5	7
6	Minimum task execution time	2	3	5	7	8	9	4	6	10	1
7	Maximum ranked positional weight	9	1	5	7	4	2	3	6	8	10
8	Minimum inverse positional weight	1	5	9	10	8	2	7	4	6	3
9	Tasks with less occupied position	5	7	3	8	9	4	6	1	2	10
10	Tasks with most occupied position	2	10	1	6	4	9	8	3	7	5

conducted in a similar manner; for example, only recharge stations on the ground will be considered for recharging AGVs. Due to the implementation-wise nature of the modification toward the a priori algorithm, the constructive heuristic to create a schedule from a task sequence is not elaborated further in this article.

DE-fused PSO

The heuristic-based approach is viable for solving the problem of scheduling collaborative UAV–AGV operations, whose nature is NP-hard. There are heuristics for constructing a solution and for exploring the solution space (searching). Constructive heuristics can be used for producing initial solutions prior to the search. Since heuristics are designed based on intuitive rules according to the constraints at hand, the produced solutions are good starting points for the search. These rules are well known as the priority rules. In this study, 10 priority rules have been utilized, which are addressed in the study by Khosiawan and Nielsen.[8] The priority rules are depicted in Table 3. A solution is represented as a sequence of tasks, which correspond to a schedule. For instance, a task sequence [1, 5, 10, 9, 7, 6, 4, 8, 3, 2] corresponds to the schedule depicted in Figure 1.

In the study by Khosiawan et al.,[4,8] PSO shows that the algorithm exposes a room for improvements on the explorative side. The position update during the search is guided by the local and global best particles. The global best particle is formed through the efforts of all particles in the swarm, but it is postulated to be marginal. Through an investigation in this study, DE is able to explore the search space, where more optimum solutions lie—where PSO generally does not reach. On the other hand, there is a trade-off between the optimality of the solution and the computation time which is quite significant in DE.

This trade-off is the challenging gap which this study tries to bridge. The proposed DEFPSO is aimed to produce a high quality near optimum solution whilst balancing time efficiency. In the place of the local and global best particles, a random particle from the current swarm is used for the position update. This treatment is realistic because the initial swarm is generated based on the priority rules. These rules employ heuristics which are believed to be sensible to produce a good quality schedule. Furthermore, a crossover operation with the global best particle is performed after the position update. This allows both rapid (position) information absorption from the global best solution and search space exploration (i.e. through the recombinant[7] particle) simultaneously. The detailed procedure of DEFPSO is depicted in Figure 2 and described as follows.

Step 1. Initialize DEFPSO parameters. They include F (degree of velocity update), CR (possibility of crossover), N (size of population), maximum number of iterations, and maximum number of iterations without improvement.

Step 2. Generate initial swarm based on the given priority rules. If the size of population exceeds the number of unique task sequences, random mutations will be performed to the existing particles and the newly formed ones are added into the swarm. If the number of all possible combinations of the sequence ξ is less than the required size of population, then the size of population is set to ξ.

Step 3. If the maximum number of generations is reached, select the global best particle as the final solution. Otherwise, go to step 4.

Step 4. If the maximum number of generations without any improvement is reached, select the global best particle as the final solution. Otherwise, go to step 5.

Step 5. If every individual in the population has been evolved in the current generation, go to step 3. Otherwise, get the next unevolved particle, evaluate its fitness value, and update the global best particle if a better solution is found.

Step 6. Update the position based on the current velocity.

Step 7. Crossover with the global best particle is conducted based on the value of CR. This action is performed to promote the generation of high quality offspring. This allows a great step of search, while inducing a good drive (direction) throughout the search. On the other hand, it evens out the absence of the local best particle influence (see step 8 for more elaboration).

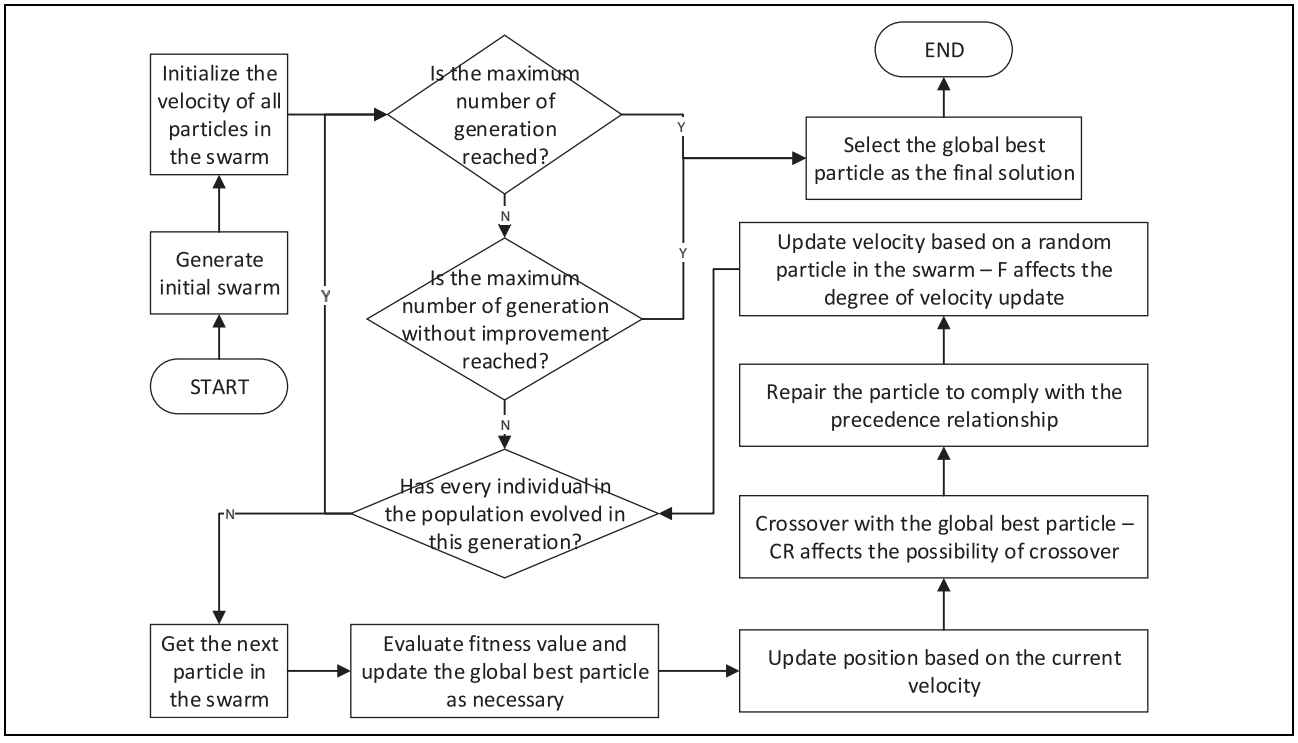

Figure 2. Flowchart of DEFPSO algorithm. DEFPSO: differential evolution-fused particle swarm optimization.

Step 8. Update the velocity based on a random particle (other than itself) from the swarm. The degree of velocity update is affected by F. It acts similar to the role of social learning coefficient c_2 in the traditional PSO. In contrast with the traditional PSO (and as briefly mentioned in step 7), DEFPSO does not take the distance of the local best particle with the current one. This treatment is performed to allow more explorative behavior during the search. Afterwards, go to step 5.

In this study, the optimization with DEFPSO is aimed to minimize the makespan and total battery consumption. The fitness evaluation will be done with makespan as the top priority. When the makespan of two schedules are the same, the total battery consumption will be used as a tie breaker. The schedule with less makespan and total battery consumption is preferred. The results of the numerical simulations are shown in the following section.

Numerical simulations

The proposed methodology has been benchmarked with 12 data sets, each with different weighted characteristics, that is, geographical scale (laboratory scale and industrial scale), number of predecessors, and number of tasks. The simulations are run on an Intel Core i7-4910MQ processor (2.9 GHz) with 32 GB of RAM. They involve numerous scheduling attempts in the following manner.

- There are two geographical scales: laboratory and industrial scale.
- For each scale, there are three different data set classifications based on the mean number of predecessors: 0, 1, and 2.

The exact number of predecessors of a task will be normally distributed with $\overline{x} = 0 \vee 1 \vee 2$ and $\sigma = min\,(1, \overline{x})$.

- For each mean number of predecessor, there are two data set classifications based on the number of tasks: 50 and 100.

In the end, there are $2 \times 3 \times 2 = 12$ data sets with different weights of the aforementioned characteristics (i.e. geographical scale, number of predecessors, and number of tasks).

- For each task data set, 20 scheduling runs are performed.

With numerous runs on the same data set, the analysis results are based on reproducible behavior of the tested algorithms. In total, there are $12 \times 20 = 240$ runs for each algorithm. Since there are three benchmarked algorithms: DE, PSO, and DEFPSO; $240 \times 3 = 720$ runs are performed.

Parameter values

The selected set of parameters through the simulations, in respect to the investigated three methodologies (i.e. DE, PSO, DEFPSO), are listed as follows.

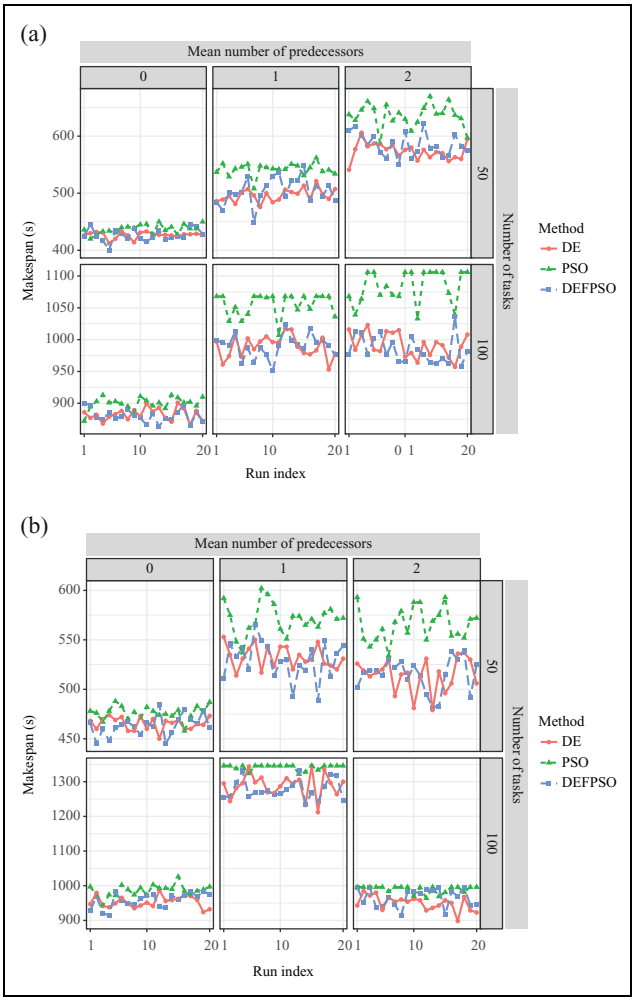

Figure 3. Makespan of schedules for laboratory scale (a) and industrial scale (b) data sets with DE, PSO, and DEFPSO. DE: differential evolution; PSO: particle swarm optimization; DEFPSO: DE-fused PSO.

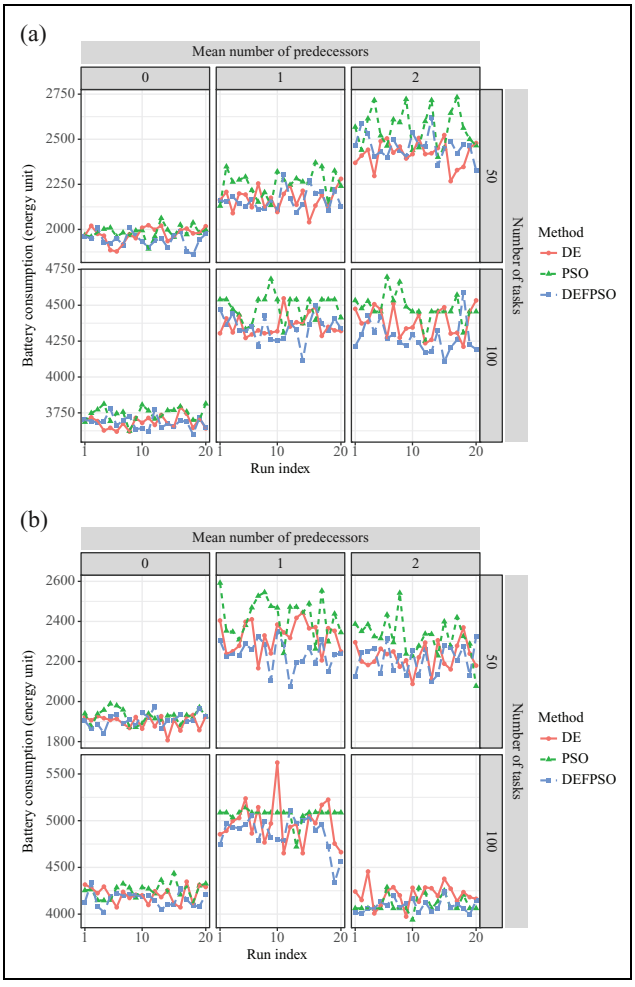

Figure 4. Total battery consumption of schedules for laboratory scale (a) and industrial scale (b) data sets with DE, PSO, and DEFPSO. DE: differential evolution; PSO: particle swarm optimization; DEFPSO: DE-fused PSO.

- DE

  The values of $F = 0.8$ (weighting factor which controls mutation) and $CR = 0.5$ (crossover control parameter).

- PSO

  The values of $c_1 = 1$ (cognitive learning coefficient) and $c_2 = 2$ (social learning coefficient), while u_1 and u_2 are randomly (following a uniform distribution) set in the range of [0, 0.5].

- DEFPSO

  The values of $F = 0.5$ (F acts similar to c_2 in the traditional PSO) and $CR = 0.5$.

- DE, PSO, and DEFPSO

  The values of $N = 40$ (size of population), $I = 40$ (maximum number of iterations), and $\gamma = 10$ (maximum number of iterations without improvement).

In this section, the simulations are done upon the operations of five agents (3 UAVs and 2 AGVs), while the ones with six agents (3 UAVs and 3 AGVs) are depicted in Appendix 1 for further reading. This study is a pilot investigation on UAV–AGV operations which is originated from a work on UAV operations in indoor environment.[21] As a minimum working instance for multi-agent operations (multiple UAVs and AGVs) which is dominated by UAV, three UAVs and two AGVs are used. Furthermore, simulations on three UAVs and three AGVs are also performed to see more results.

Figure 3 depicts the makespans of the schedules generated by DE, PSO, and DEFPSO. The makespans yielded by DEFPSO clearly outperform those from PSO and are on par with the ones from DE. Furthermore, the proposed DEFPSO consistently maintains its position relative to DE and PSO regardless of the target data set.

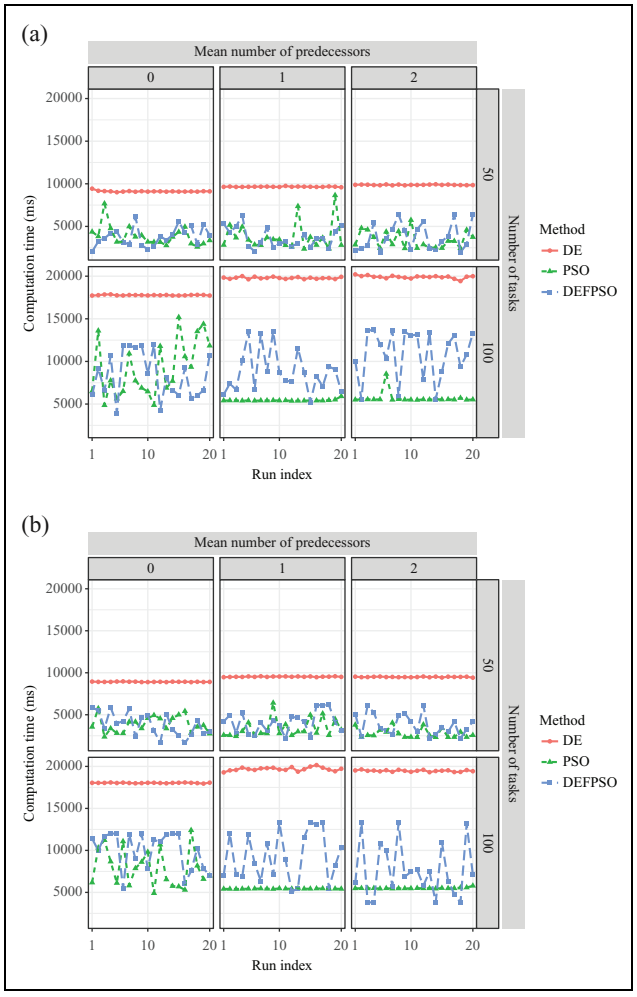

Figure 5. Computation time of schedules for laboratory scale (a) and industrial scale (b) data sets with DE, PSO, and DEFPSO. DE: differential evolution; PSO: particle swarm optimization; DEFPSO: DE-fused PSO.

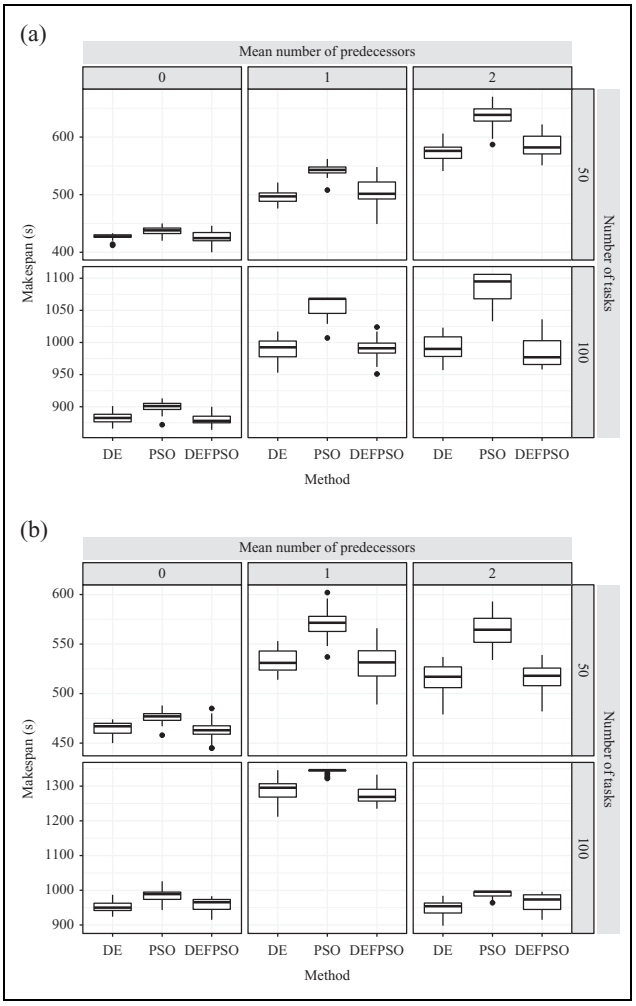

Figure 6. Boxplot of makespan of schedules for laboratory scale (a) and industrial scale (b) data sets in connection with the results in Figure 3.

From the perspective of the total battery consumption, as the lower-priority objective (compared to makespan), DEFPSO generally outperforms PSO and is on par with DE. The results show DEFPSO to be even better than DE in many runs. One can see this situation as finding a diamond among other minerals and rocks. This indicates that DEFPSO explores various areas (getting stuck in the same area less), and allow it to find better solutions in the promising area.

With the obtained objective values depicted in Figures 3 and 4, the computation time of the three methods plays an important role to make a remark. In Figure 5, the computation time of DEFPSO is slightly higher than PSO and significantly lower than DE. DEFPSO's appeal is then formed by the high quality objective value (better than PSO and on par with DE) that it can achieve within less time than what DE needs. On a further discussion, the computation time graph of DEFPSO is oscillating due to its convergence in various high quality local optima. With the trade-off of a significantly higher computation time, DE offers the tendency to search further and get better objective values than DEFPSO.

Analysis

To pull out a tractable numerical analysis, the quartiles of the makespan, battery consumption, and computation time data are shown in Figures 6 to 8.

The mean numbers of the characteristics being observed are put into Figure 9. DE and DEFPSO are compared toward PSO to show the better results they gained. It is followed by calculating the gain ratio to quantify the excellence of the proposed DEFPSO. The ratios show that in terms of objective values (makespan and total battery consumption), DEFPSO gains as much as 83–140% of improvement from what DE gets against PSO. This means that DEFPSO's performance is nearly as good as DE's or even better. From the perspective of the computation time,

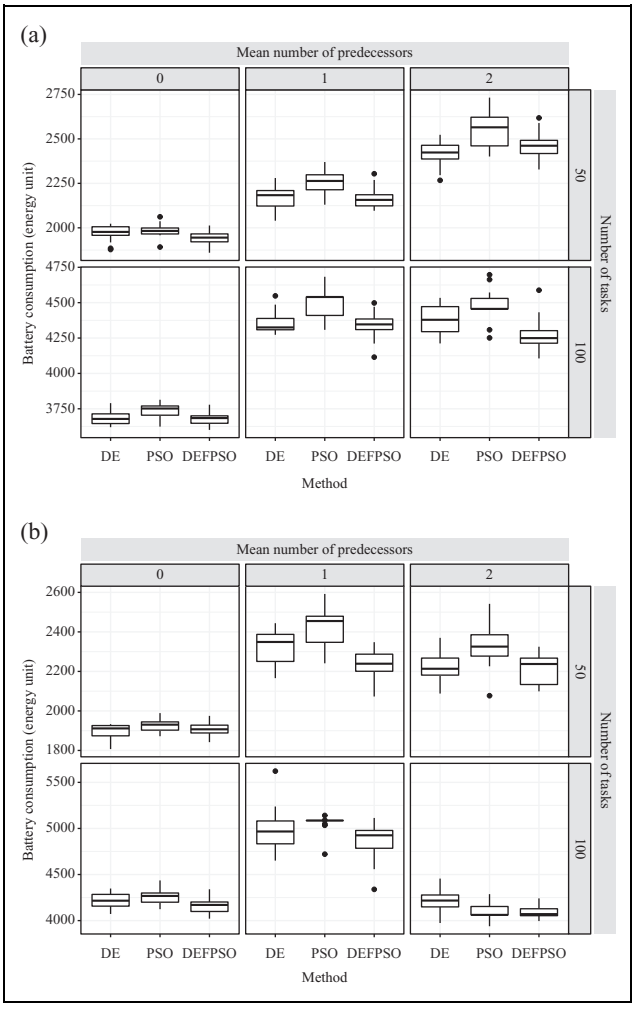

Figure 7. Boxplot of battery consumption of schedules for laboratory scale (a) and industrial scale (b) data sets in connection with the results in Figure 4.

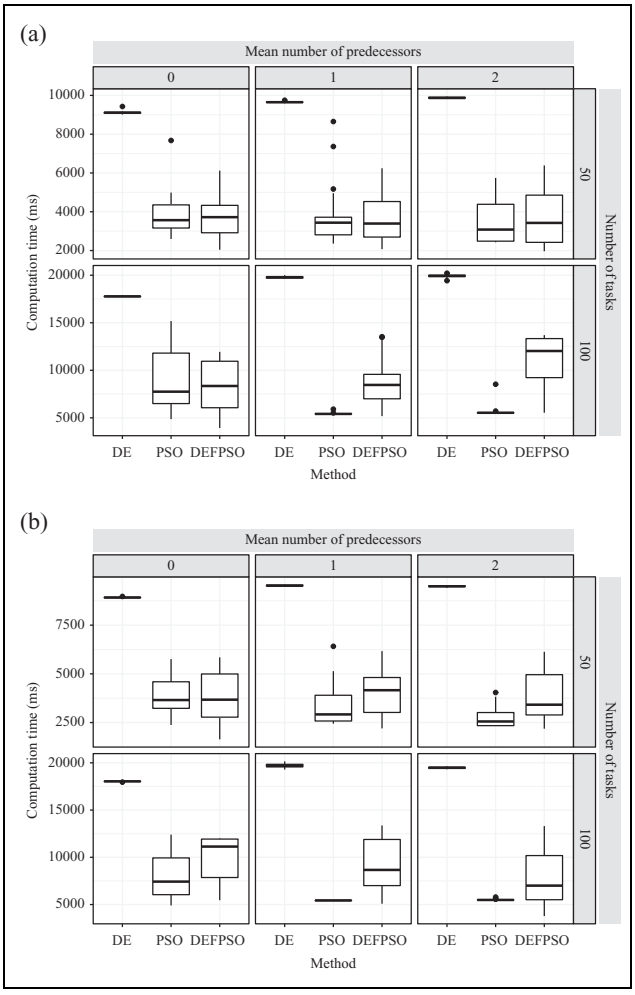

Figure 8. Boxplot of computation time of schedules for laboratory scale (a) and industrial scale (b) data sets in connection with the results in Figure 5.

DEFPSO needs at most 21% of DE's computation time, which is on par with PSO's (refer Figure 8). Hence, DEFPSO is shown as an effective methodology to solve the problem of task executions by multiple UAVs and AGVs in indoor manufacturing environment.

In the calculated gained ratio, it is depicted that the superiority of makespan yielded by DE and DEFPSO are on par, it is almost as good (with a ratio a bit less than 1.0) or even better (with a ratio of > 1.0) in regard to the various task data sets. In addition, the gain ratio in respect to the computation time is low (< 0.21), which signifies the additional time (against PSO's computation time) required by DEFPSO is not as long as DE's. This additional time represents the trade-off of having a longer computation time to get a higher quality near optimum solution.

Additionally, a paired *t*-test analysis of the proposed method is presented in Table 4. It depicts the certainty of superiority or inferiority of DEFPSO over each of its parents: DE and PSO. With 95% confidence interval, a *p* value less than 0.05 indicates that the results from DEFPSO has statistically significantly lower makespan, battery consumption or computation time than the ones from DE or PSO. Each paired *t* test has the same one-sided alternative hypothesis ($H_a = \mu_o - \mu_n > 0$, where o is another algorithm's observation and n is DEFPSO's observation) and degrees of freedom (df = 239). Table 5 lists the cases in the *t* test analysis in Table 4. More variations of DEFPSO parameters are used to conduct more simulations to be used in the statistical test. When not mentioned, the parameters conform to the values described in the "Parameter values" subsection.

In cases C1–C4, DE is statistically tested against DEFPSO variants (with different configurations). The makespans of schedules from DEFPSO are definitely not less than the ones from DE. The *p* values are quite greater than 0.05, even though C1 has a slightly lower value than the others. For the secondary objective, all DEFPSO variants have statistically significantly lower battery

No	No. of tasks	Method	Makespan average (s)	Battery consumption average (energy unit)	Computation time average (ms)	Gain against PSO			Gain ratio of DEFPSO:DE		
						Makespan	Battery	Comp. time	Makespan	Battery	Comp. time
(result summary in regard to the lab. scale task datasets)											
1	50	DE	498.9167	2187.417	9549.233	39.4833	79.083	-5863.95			
2	50	PSO	538.4	2266.5	3685.283	-	-	-	0.834107	0.982929	0.012077
3	50	DEFPSO	505.4667	2188.767	3756.1	32.9333	77.733	-70.817			
4	100	DE	954.6167	4141	19158.567	58.2166	95.383	-12398.25			
5	100	PSO	1012.8333	4236.383	6760.317	-	-	-	1.050673	1.439628	0.210137
6	100	DEFPSO	951.6667	4099.067	9365.65	61.1666	137.316	-2605.333			
(result summary in regard to the industrial scale task datasets)											
7	50	DE	504.1	2149.4	9316.1	33.4833	75.117	-5963.033			
8	50	PSO	537.5833	2224.517	3353.067	-	-	-	1.047286	1.399151	0.092682
9	50	DEFPSO	502.5167	2119.417	3905.733	35.0666	105.1	-552.666			
10	100	DE	1064.1667	4463.95	19066.967	41.7166	12.05	-12775.45			
11	100	PSO	1105.8833	4476	6291.517	-	-	-	0.925291	8.643154	0.209154
12	100	DEFPSO	1067.2833	4371.85	8963.55	38.6	104.15	-2672.033			

Figure 9. Result summary of the proposed DEFPSO in connection with DE and PSO. DE: differential evolution; PSO: particle swarm optimization; DEFPSO: DE-fused PSO.

Table 4. *p* Values of one-sided paired *t* tests of DE or PSO against DEFPSO with different configurations.

	Observation criterion		
Case	Makespan	Battery consumption	Computation time
C1	0.779	2.946e-06	1.741e-103
C2	0.991	4.445e-05	1.774e-99
C3	0.999	0.026	2.554e-98
C4	0.938	1.304e-07	2.120e-107
C5	9.930e-48	3.229e-28	0.999
C6	5.732e-41	6.244e-26	0.999
C7	2.472e-38	5.576e-19	0.999
C8	5.829e-46	1.428e-34	0.999

DE: differential evolution; PSO: particle swarm optimization; DEFPSO: DE-fused PSO.

Table 5. Cases for statistical significance analysis in Table 4.

Case	Description
C1	DE against DEFPSO
C2	DE against DEFPSO whose *F* modified to 0.8
C3	DE against DEFPSO whose *F* and *CR* modified to 0.8 and 0.3, respectively
C4	DE against DEFPSO whose *F* and *CR* modified to 0.8 and 0.8, respectively
C5	PSO against DEFPSO
C6	PSO against DEFPSO whose *F* modified to 0.8
C7	PSO against DEFPSO whose *F* and *CR* modified to 0.8 and 0.3, respectively
C8	PSO against DEFPSO whose *F* and *CR* modified to 0.8 and 0.8, respectively

DE: differential evolution; PSO: particle swarm optimization; DEFPSO: DE-fused PSO.

consumption than DE. The computation time of all variants are also significantly lower than DE. From here, the performance of C1 in pursuing the objectives during the search cannot outperform DE. It is only on par with DE as depicted in Figure 3.

In cases C5–C8, PSO is statistically tested against DEFPSO variants (with different configurations). All four cases show that DEFPSO variants have statistically significantly lower makespan and battery consumption than PSO. In terms of computation time, all four cases statistically not lower than PSO (with the *p.value* being 0.999). On a thorough observation, C5 and C8 are on par, which indicates the essence of the proportion of F and CR (mutation and crossover rate) and the balance between them to have a good search experience.

Based on the aforementioned statistical significance analysis, the superiority of DEFPSO with $F = 0.5$ and $CR = 0.5$ among others can be postulated. Furthermore, DEFPSO has statistically significantly lower makespan and objective compared to PSO, and statistically significantly lower computation time compared to DE.

On the robustness of the system

In the UAV operations, there are uncertain events that may happen during the flight or the task execution. It can expose a delay to the originally scheduled completion time of a task. To some extent, the exposed delay will still yield the same schedule to achieve well optimized operations. When the delay occurs frequently with a significant amount of time, an efficient method of producing a fault-tolerant schedule is required. A straightforward rescheduling is what is within the capability of the current study. Hence, a fault-tolerant scheduling system which focuses on the

robustness of the scheduling system is the next goal to pursue.

Conclusion

The problem of scheduling task executions by multiple robots is NP-hard natured, which demands the involvement of heuristic and metaheuristic algorithms to get a high quality feasible solutions whilst balancing time efficiency. Researchers have been investigating prominent metaheuristic algorithms such as DE and PSO to tackle problems with such a nature. The quality of the solution produced by DE is found to be usually better, while the computation time is generally longer compared to PSO. In this article, a mathematical formulation of the addressed problem is developed. A metaheuristic algorithm called DEFPSO is proposed to solve it, where the explorative property of DE is fused into PSO, and the performance is then benchmarked through several data sets. They have different weighted characteristics including geographical scale, number of predecessors, and number of tasks. DEFPSO obtains at least 83% of improvement in terms of objective values, and needs at most only 21% of the computation time compared to what DE gains against PSO. The results are also analyzed through paired *t* test, and DEFPSO statistically significantly outperforms DE and PSO in terms of computation time and objective values, respectively. Future researchers in the optimization area may conduct further studies for different optimization fields, not only scheduling, and perform different utilization ways of the parameters, operators, and the overall optimization framework.

Declaration of conflicting interests

The author(s) declared no potential conflicts of interest with respect to the research, authorship, and/or publication of this article.

Funding

The author(s) disclosed receipt of the following financial support for the research, authorship, and/or publication of this article: This work has been partly supported by Innovation Fund Denmark under project UAWorld, grant agreement number 9-2014-3.

References

1. Li D, Tang H, Wang S, et al. A big data enabled load-balancing control for smart manufacturing of industry 4.0. *Cluster Computing J Networks Software Tools Appl* 2017; 20(2): 1855–1864. DOI: 10.1007/s10586-017-0852-1.
2. Qian F, Zhong W, and Du W. Fundamental theories and key technologies for smart and optimal manufacturing in the process industry. *Engineering* 2017; 3(2): 154–160. DOI: 10.1016/J.ENG.2017.02.011.
3. Ang JH, Goh C, Saldivar AAF, et al. Energy-efficient through-life smart design, manufacturing and operation of ships in an industry 4.0 environment. *Energies* 2017; 10(5): 610. DOI: 10.3390/en10050610.
4. Khosiawan Y, Park YS, Moon I, et al. Task scheduling system for UAV operations in indoor environment. *Artif Intell arXiv preprint arXiv:160406223* 2016.
5. Kaveh A. *Advances in metaheuristic algorithms for optimal design of structures*. Switzerland: Springer International Publishing, 2017. ISBN 9783319461724.
6. Nilakantan MJ, Ponnambalam S, and Jawahar N. Design of energy efficient RAL system using evolutionary algorithms. *Eng Comput* 2016; 33(2): 580–602. DOI: 10.1108/EC-11-2014-0232.
7. Price K, Storn RM, and Lampinen JA. *Differential evolution: a practical approach to global optimization*. Germany: Springer Science & Business Media, 2006.
8. Khosiawan Y and Nielsen I. Indoor UAV scheduling with restful task assignment algorithm. *Artif Intell arXiv preprint arXiv:170609737* 2017.
9. Das GP, McGinnity TM, Coleman SA, et al. A distributed task allocation algorithm for a multi-robot system in healthcare facilities. *J Intell Rob Syst* 2015; 80(1): 33–58. DOI: 10.1007/s10846-014-0154-2.
10. Hsu PE, Hsu YL, Chang KW, et al. Mobility assistance design of the intelligent robotic wheelchair. *Int J Adv Rob Syst* 2012; 9(6): 244. DOI: 10.5772/54819.
11. Zhang R. A simulated annealing-based heuristic algorithm for job shop scheduling to minimize lateness. *Int J Adv Rob Syst* 2013; 10(4): 214. DOI: 10.5772/55956.
12. Nielsen I, Dang QV, Nielsen P, et al. *Scheduling of mobile robots with preemptive tasks*. In: *Distributed computing and artificial intelligence, 11th international conference*, pp. 19–27. Switzerland: Springer.
13. Poli R, Kennedy J, and Blackwell T. Particle swarm optimization. *Swarm Intell* 2007; 1(1): 33–57. DOI: 10.1007/s11721-007-0002-0.
14. Kirsch U. *Structural optimization: Fundamentals and applications*. 1st ed. Berlin, Heidelberg: Springer-Verlag, 1993. ISBN 978-3-540-55919-1,978-3-642-84845-2.
15. Zhu Z, Tang B, and Yuan J. Multirobot task allocation based on an improved particle swarm optimization approach. *Int J Adv Rob Syst* 2017; 14(3): 1–22. DOI: 10.1177/1729881417710312.
16. Mahmoudzadeh, Powers DM, Sammut K, et al. Toward efficient task assignment and motion planning for large scale underwater missions. *Int J Adv Rob Syst* 2016; 13(5): 1–13. DOI: 10.1177/1729881416657974.
17. Krömer P, Abraham A, Snášel V, et al. Differential evolution for scheduling independent tasks on heterogeneous distributed environments. In: Snášel V, Szczepaniak PS, Abraham A, et al. (eds), *Advances in intelligent web mastering – 2: proceedings of the 6th Atlantic web intelligence conference - AWIC'2009, Prague, Czech Republic, September, 2009*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 127–134. ISBN 978-3-642-10687-3. DOI: 10.1007/978-3-642-10687-3_12.
18. Nearchou AC and Omirou SL. Differential evolution for sequencing and scheduling optimization. *J Heurist* 2006; 12(6): 395–411. DOI: 10.1007/10732-006-3750-x.
19. Odili J, Kahar MNM, Noraziah A, et al. A comparative evaluation of swarm intelligence techniques for solving

combinatorial optimization problems. *Int J Adv Rob Syst* 2017; 14(3): 1–11. DOI: 10.1177/1729881417705969.
20. Li Z, Janardhanan MN, Tang Q, et al. Co-evolutionary particle swarm optimization algorithm for two-sided robotic assembly line balancing problem. *Adv Mechanic Eng* 2016; 8(9): 1–14. DOI: 10.1177/1687814016667907.
21. Khosiawan Y and Nielsen I. A system of UAV application in indoor environment. *Prod Manuf Res* 2016; 4(1): 2–22.

Appendix 1

Simulation results for the operations of six agents generated by DE, PSO, and DEFPSO

Figures 1A to 1C depict the makespan, battery consumption, and computation time of the schedules from simulations with six agents (i.e. 3 UAVs and 3 AGVs).

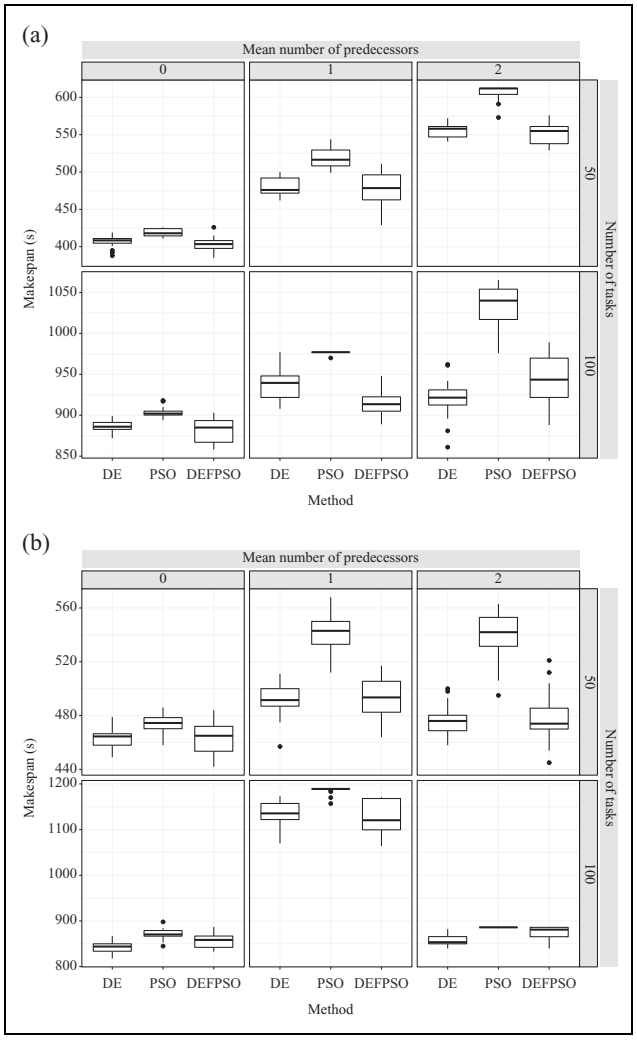

Figure 1A. Makespan of schedules for laboratory scale (a) and industrial scale (b) data sets with DE, PSO, and DEFPSO. DE: differential evolution; PSO: particle swarm optimization; DEFPSO: DE-fused PSO.

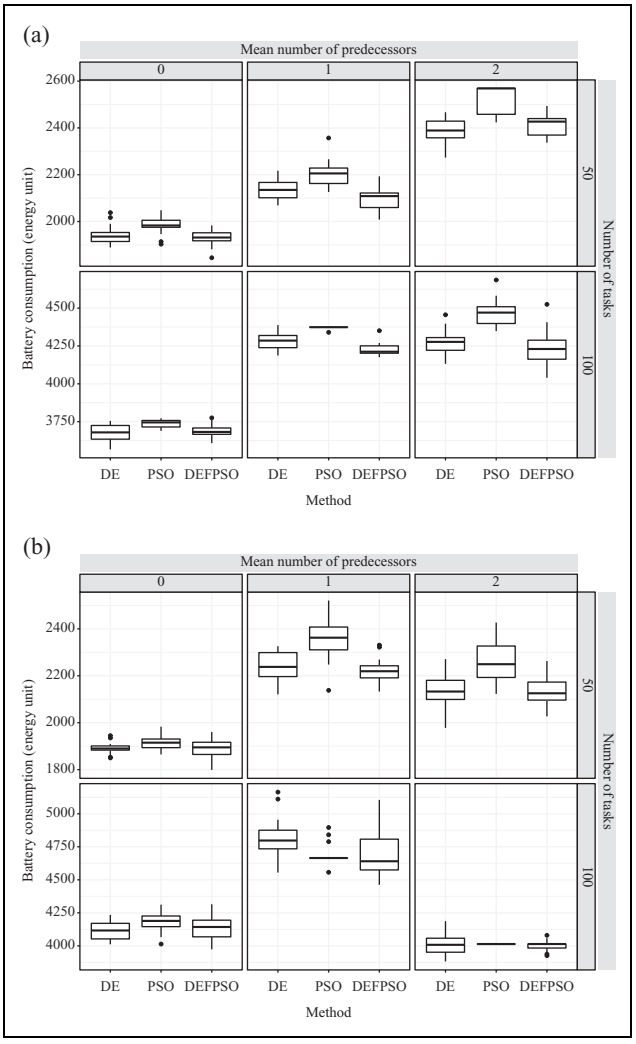

Figure 1B. Battery consumption of schedules for laboratory scale (a) and industrial scale (b) data sets with DE, PSO, and DEFPSO. DE: differential evolution; PSO: particle swarm optimization; DEFPSO: DE-fused PSO.

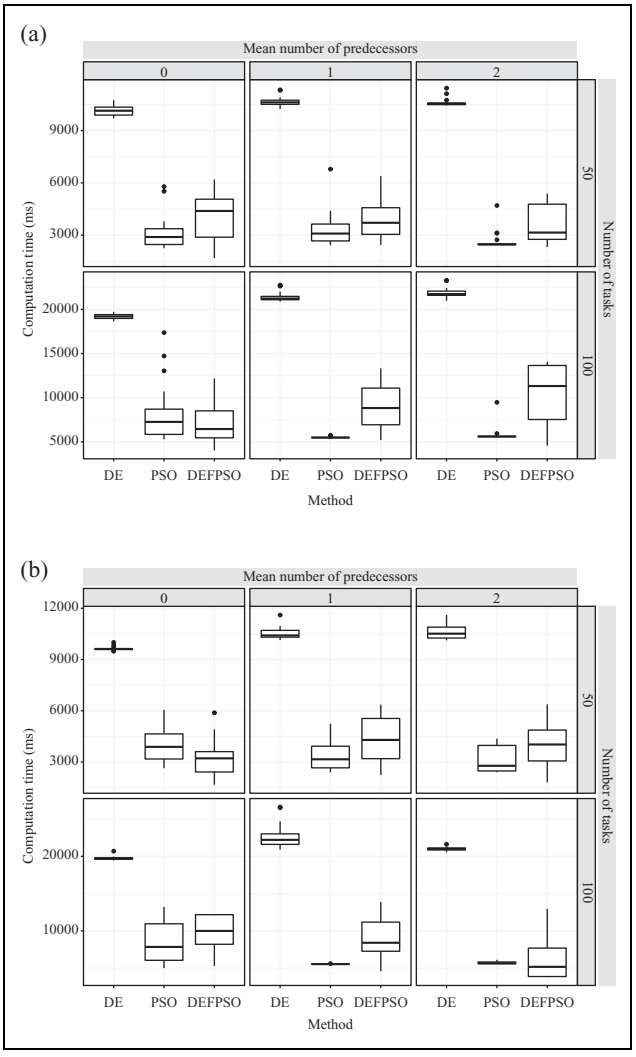

Figure 1C. Computation time of schedules for laboratory scale (a) and industrial scale (b) data sets with DE, PSO, and DEFPSO. DE: differential evolution; PSO: particle swarm optimization; DEFPSO: DE-fused PSO.