The Knowledge Engineering Review, Vol. 34, e19, 1 of 25.
doi:10.1017/S0269888919000134

A multi-objective evolutionary hyper-heuristic algorithm for team-orienteering problem with time windows regarding rescue applications

HADI S. AGHDASI[1], SAEED SAEEDVAND[1], and JACKY BALTES[2]

[1] *Faculty of Electrical and Computer Engineering, University of Tabriz, Tabriz, Iran*
e-mails: aghdasi@tabrizu.ac.ir, saeedvand@tabrizu.ac.ir
[2] *Department of Electrical Engineering, National Taiwan Normal University, Taipei, Taiwan*
e-mail: jacky.baltes@ntnu.edu.tw

Abstract

The team-orienteering problem (TOP) has broad applicability. Examples of possible uses are in factory and automation settings, robot sports teams, and urban search and rescue applications. We chose the rescue domain as a guiding example throughout this paper. Hence, this paper explores a practical variant of TOP with time window (TOPTW) for rescue applications by humanoid robots called TOPTWR. Due to the significant range of algorithm choices and their parameters tuning challenges, the use of hyper-heuristics is recommended. Hyper-heuristics can select, order, or generate different low-level heuristics with different optimization algorithms. In this paper, first, a general multi-objective (MO) solution is defined, with five objectives for TOPTWR. Then a robust and efficient MO and evolutionary hyper-heuristic algorithm for TOPTW based on the humanoid robot's characteristics in the rescue applications (MOHH-TOPTWR) is proposed. MOHH-TOPTWR includes two MO evolutionary metaheuristics algorithms (MOEAs) known as non-dominated sorting genetic algorithm (NSGA-III) and MOEA based on decomposition (MOEA/D). In this paper, new benchmark instances are proposed for rescue applications using the existing ones for TOPTW. The experimental results show that MOHH-TOPTWR in both MOEAs can outperform all the state-of-the-art algorithms as well as NSGA-III and MOEA/D MOEAs.

1 Introduction

The team-orienteering problem (TOP) has broad applicability. Examples of possible uses are in factory and automation settings, robot sports teams (Baltes *et al.*, 2017), and urban search and rescue applications. Throughout this paper, we chose the rescue domain as a guiding example. After a natural or man-made disaster, many tasks need to be accomplished efficiently and quickly. So task allocation is a crucial problem to avoid unnecessary casualties and economic losses (Gunn & Anderson, 2015). There are many challenges for task allocation in rescue scenarios.

In disaster rescue, tasks include searching for survivors, extinguishing fires, cutting electrical wires, turning a valve to disconnect gas or water flow, transporting victims, breaking through a wall, connecting a fire hose, checking an area for gas leaks, etc. Most of these tasks have deadlines, for example, a time after which a victim's condition deteriorates or a fire consumes a building. These kinds of complex scenarios are impossible to accomplish for a single agent, so teams of agents must be used. Furthermore, in disaster rescue, various tasks have different allocation importance with varying time and energy requirements (Yin *et al.*, 2007; Su *et al.*, 2018).

Multi-robot systems (Zhang & Ueno, 2007) due to their distributed parallel processing abilities have been used in various missions that require robust and fast performance such as monitoring (Saeedvand & Aghdasi, 2016), tracking (Chang *et al.*, 2016), surveillance (Khamis *et al.*, 2015), hazardous industries (Jose & Pratihar, 2016), etc. The multi-robot task allocation (MRTA) problem is one of the

core issues in multi-robot systems (Nunes *et al.*, 2017; Schwarzrock *et al.*, 2018; Saeedvand *et al.*, 2019). In MRTA, the solutions for different applications also depend on the criteria used for optimization, such as distance traveled, time to completion, and energy consumed. The MRTA problem has been investigated extensively with several proposed approaches in the literature (Farinelli *et al.*, 2017). The most recent taxonomy for MRTA problems proposes a comprehensive comparison (Nunes *et al.*, 2017). In this taxonomy, the authors also discuss the relationship between the TOP with time windows (TOPTWs) and the job-shop scheduling problem (JSP). The JSP (similar to TOPTW) is sequencing the jobs and assigning their start and end times by performing scheduling (Nunes *et al.*, 2017).

The orienteering problem (OP), initially defined by Tsiligirides, is the problem of selecting and ordering a set of tasks to maximize the total profit within a given time budget (Tsiligirides, 1984). To date, some common extensions such as the team OP (TOP), the team OP with time windows (TOPTWs), and the time-dependent OP have been studied (Gunawan *et al.*, 2016). In the context of the team OP, instead of one path, a certain number of paths are allowed, each allocated to one agent. In TOPTW, each node may have a time window associated with it, and agents can only visit the node and receive the reward during this time window (Labadie *et al.*, 2012). Among the many extensions of TOP, the TOPTW matches closely to the task allocation problem in disaster rescue domains (Nunes *et al.*, 2017).

Consequently, for rescue applications, in this paper, our primary focus is on the TOPTW with some additional differences. The major difference is that in TOPTW, most approaches only solve for a single-objective function, whereas in rescue domains several possibly conflicting objective functions must be optimized. For example, both the sum of the accumulated profits (e.g., tasks completed) and the sum of the task completion times are important features in the rescue domain. The second difference is that in our paper we also include a cost (energy consumption) associated with visiting a node. We included this factor because energy consumption is an important aspect of rescue domains. Third, in the TOPTW, the problem is assuming homogeneous robots and therefore applies the same time constraints to all robots. In rescue domains, mostly the tasks require a different duration of time with constraints, which may depend on the agents' state and their current energy consumption.

Several algorithms have been proposed to solve the TA and TOP with extensions. These include the brute-force method (Park *et al.*, 2017; Huang *et al.*, 2018) and heuristics methods (Gunawan *et al.*, 2017; Lin & Vincent, 2017; Vincent *et al.*, 2017; Koubaa *et al.*, 2018; Bottarelli *et al.*, 2019). Many different heuristics and evolutionary computing methods have been applied in order to find good or optimal solutions in a reasonable amount of time. Metaheuristic approaches, such as tabu search (Tang & Miller-Hooks, 2005), particle swarm optimization (Vincent *et al.*, 2017), ant colony optimization (Montemanni & Gambardella, 2009), variable neighborhood search (Campbell *et al.*, 2011), iterated local search heuristic (ILS) (Lin & Vincent, 2012), simulated annealing (SA) iterated local search heuristic (SAILS) (Gunawan *et al.*, 2017), artificial bee colony (ABC) algorithm (Cura, 2014), a hybrid approach (Gunawan *et al.*, 2018), biologically inspired approaches (Kube & Bonabeau, 2000), have all been used and worked well for some specific problems. Among the existing studies on TOPTW, most approaches are limited to single-objective functions. As for TOPTW with multiple objectives, the most common approaches transform it into a single-objective optimization problem and then use single-objective approaches to solve it. On the other hand, some researchers convert it into a bi-objective problem (Martín-Moreno & Vega-Rodríguez, 2018). To the best of our knowledge, the original TOPTW has not been solved yet for MO problems.

Golden proved that the TOP is NP-hard (Golden *et al.*, 1987). Furthermore, task allocation problems are NP-hard (non-deterministic polynomial-time hard), (Solomon, 1986; Savelsbergh, 1985). Therefore, it is challenging and usually requires long computational time to find an optimal solution for non-trivial problems. In real-world problems, which are usually large instances of the TOP problem, optimal solutions are therefore intractable. Hence, evolutionary algorithms (EAs) have been used to find good, albeit non-optimal, solutions to MRTA and TOPTW. However, generally due to the significant variety of parameter choices, the use of the EAs can be problematic. On the other hand, in a rescue environment, MO solutions are often advantageous. MOEAs are based on Pareto-dominance concepts in which a set of solutions should represent suitable trade-offs between different objectives (Dong & Dai, 2018). As

suggested in the future works section of the recent survey (Gunawan *et al.*, 2016), and to the best of our knowledge, only a few MO solutions are available for both MRTA and TOPTW in rescue applications. So far, the related literature on the TOPTW is limited to dealing with a maximum of two relevant conflicting objectives (Schilde *et al.*, 2009; Vansteenwegen *et al.*, 2011; Jiang, 2016; Bederina & Hifi, 2017; Nunes *et al.*, 2017).

It is recommended that the MOEAs can dealt with using hyper-heuristics (Burke *et al.*, 2010; Guizzo *et al.*, 2017; Alkhanak & Lee, 2018). Hyper-heuristics primarily choose or generate low-level heuristics (LLHs), while optionally selected optimization algorithms are applied afterward. Hence, the hyper-heuristics are independent of the underlying algorithms. The idea of employing MO hyper-heuristics comparing conventional algorithms is to create a robust and versatile algorithm. Opposite to EAs such as genetic or memetic algorithms (Abbaszadeh *et al.*, 2012; Abbaszadeh & Saeedvand, 2014), hyper-heuristic operates over the LLHs space, whereas metaheuristics (Yang, 2010) operate on the solution space. So hyper-heuristic, instead of solving the problem directly, is a technique to select the best existing heuristics or generate a new heuristic (Burke *et al.*, 2013). As a result, we can state a hyper-heuristic guides the search process on the LLHs such as metaheuristics and genetic operators. However, based on the advantages of the hyper-heuristic method, few attempts are made at applying hyper-heuristic to the OP (Toledo & Riff, 2015), and as we are aware, there is no research on TOPTW or MRTA problems. Motivated by this, we introduce our solution based on hyper-heuristics.

Robots are used in hazardous industries with a great risk for human operators such as defense projects (Diftler *et al.*, 2003), rescue applications, and disaster response (DeDonato *et al.*, 2015). The use of humanoid robots in different applications is increasing (Jin *et al.*, 2017; Saeedvand *et al.*, 2018; Spenko *et al.*, 2018). Humanoid robots still suffer from a lack of scientific theories, practical guidelines, and mechanical development constraints. However, the development of better and more capable humanoid robots in the future is inevitable. Rescue applications for robots, especially for humanoid robots are very interesting (Spenko *et al.*, 2018). Over the last decade, different robotic challenges focused on disaster or emergency-response scenarios. For instance, the United States Defense Advanced Projects Research Agency (DARPA) proposed the DARPA robotic challenge to accelerate the development of disaster response robots. These robots are aimed to have the capability for early response and decreasing disasters breakdowns (Kaneko *et al.*, 2015; Kohlbrecher *et al.*, 2015). So a lot of the humanoid robots' development projects are focused on these challenges (Diftler *et al.*, 2003; Feng *et al.*, 2015).

Although research on TOPTW and MRTA applications is popular, few algorithms are able to solve MO problems in disaster response for rescue environments. Thus, in this paper, we propose a robust and efficient MO and evolutionary hyper-heuristic algorithm for TOPTWs based on the humanoid robot's characteristics in the rescue applications (MOHH-TOPTWR).

The theoretical contributions of this paper are as follows: (1) introducing a five-objective variant of TOPTW, (2) introducing for the first time a hyper-heuristic for a variant of TOPTWR, (3) combining 24 state-of-the-art algorithms as LLHs to propose the MOHH-TOPTWR algorithm with an efficient learning approach, (4) performing extensive experiments and statistical analysis to evaluate the proposed MOHH-TOPTWR algorithm regarding the humanoid robot characteristics for the first time (we consider humanoid robots' variant walk energy consumptions for MOHH-TOPTWR), (5) expanding new benchmark instances based on the existing benchmark instances for TOPTW at rescue applications including humanoid robots' heterogeneous energy consumptions, and (6) implementing and comparing two MOEAs as NSGA-III (Deb & Jain, 2014) and MOEA based on decomposition (MOEA/D) (Zhang & Li, 2007). We chose those algorithms because of their robustness and popularity.

This paper is organized as follows: Section 2 describes the problem based on robot's resources application and its model as a mixed integer programming (MIP) problem. Then the MO optimization and the proposed objective functions are formulated. Section 3 discusses hyper-heuristics; it explains the upper confidence bound (UCB) selection strategy and introduces the main processes of the proposed approach and LLHs. The simulation results and performance of the proposed approach are discussed in Section 4, and concluding remarks are presented in Section 5.

2 Problem description

In this section, the MOHH-TOPTWR is described. To have more clear descriptions, we modeled the problem as an MIP formulation. Therefore, we first explain the general problem and then describe MO optimizations and their objective functions.

2.1 Problem and notation description

When allocating agents to tasks, in addition to the deadline, the task also includes the profit per accomplishing each task. In tasks with a deadline, for instance, when agents (robots) allocating to extinguish a fire of a building, the accomplishment time is the most critical aspect of the task allocation. Due to limited resources of available robots, a limited number of robots can enter the disaster environment. Thus, in a disaster environment, the number of robots is usually much less than the number of tasks (Su *et al.*, 2018). Based on these characteristics, the TOPTWR is closer to TOPTW (Nunes *et al.*, 2017). In general, TOPTW can be considered as hard time windows and soft time windows. In a TOPTW with hard time windows, commencing after the latest time constraint is not allowed (Labadie *et al.*, 2012). In practice, because the robots' travel time is usually stochastic and cannot be precisely predicted, the hard time windows are often relaxed (Wang *et al.*, 2016). TOPTW with soft time windows is a generalization of TOPTW with hard time windows, which has many practical applications. In many rescue cases, relaxing the time windows may be more applicable (Wang *et al.*, 2016) because it may result in better solutions requiring movement for shorter distance and fewer movements between task time. Hence, in this paper and in the TOPTWR, the soft time windows are considered as in Wang *et al.* (2016). Note that in the MO platform, the user can choose the desired solution on the Pareto front (PF) whererin the user usually is able to select the desired outlines including solutions with hard time window choices.

Formally, the TOPTWR can be defined by stating a directed network $G = (T, L)$, where $T = \{1, 2, ..., n\}$ is the set of tasks' locations. Also, $L = \{(t, t') : t \neq t' \in T\}$. is the set of arcs path between locations. Each task $t = 1, ..., n$ is associated with a profit P_t, a time duration S_t, and a closing time C_t. All paths must start from a starting location and end at the same location which usually is a location for starting a mission. The shortest travel time $TT_{t,t'}$ which is needed to travel from location t to t' is known for each pair of locations t and t'. In the real world, $TT_{t,t'}$ can be computed by different path planning algorithms such as the jump point search algorithm (Duchoň *et al.*, 2014). The time of a route for each robot cannot exceed the given time constraint T_{max}. Each robot must finish its allocated tasks before finishing its energy. Each task performing requirement is at most one robot (no need for simultaneous performing on one task). The visit is preferred to start before the location's closing time C_i.

Based on the given taxonomy in Nunes *et al.* (2017), the TOPTWR could divide as a deterministic allocation. In the deterministic problems, typically we assume there are known tasks with time window constraints in advance which are more than robots. The TOPTWR problem is composed of two intertwined sub-problems: (i) finding best tasks' assignment to robots that optimizes objective function $f(.)$ and (ii) finding best feasible order of tasks that result in optimal assignments. Table 1 shows the notation used in the paper.

For TOPTWR problem using Table 1, an MIP formulation is proposed (Equations (1)–(11)).

$$\forall a \in A, \textit{ and } t = \textit{start of } a \qquad x_t^a = 1 \tag{1}$$

$$\forall t \in T \qquad \sum a \in A^{x_a^t} = 1 \tag{2}$$

$$\forall a \in A \qquad \sum t \in T^{W_t^a x_t^a} \leq E_a^{Max} \tag{3}$$

$$\forall a \in A \qquad \sum t \in T^{DUR_t x_t^a} \leq T^{Max} \tag{4}$$

$$\forall a \in A, t' \in T \qquad \sum t' \in T^{O_{t,t'}^a + z_{t'}^a} = x_t^a \tag{5}$$

Table 1 The notation used in the paper

Robots	
A	Set of robots
a	Robot in set A
E_a^{Max}	Capacity, or maximum energy, of the robot a
E_a^R	Energy consumption of robot a when robot rotating
E_a^S	Energy consumption of robot a when robot walking straight
Tasks	
T	Set of tasks
t	Task in set T
C_t	The closing time of the task t
O_t	Opening time of the task t
P_t	Profit associated with the task t
T_a^+	Set of all the allocated tasks to robot's plus the starting task to a
t_a^+	The task in the set T_a^+ which allocated to a robots a
T^-	Set of all the non-allocated tasks to robot's set A (reserve list)
t^-	The task in the set T^- which non-allocated to any robots in set A
T^{Max}	Total given time constraint for all tasks
DUR_t	Duration of task t
$TT_{t,t'}$	Travel time between tasks t and t'
$RT_{t,t'}$	Rotation time between tasks t and t'
W_a^t	The workload for the task t when performed by robot a
Optimization	
$f(\cdot)$	Generic optimization function
x_a^t	An indicator of assignment of a task t to robot a
$o_{t,t'}^a$	An indicator that robot a performs a task t' directly after t
z_t^a	An indicator that robot a performs task t last

$$\forall a \in A \qquad \sum t \in T_t^a z_t^a = 1 \tag{6}$$

$$\forall t \in T \qquad T^{Max} \geq DUR_t \tag{7}$$

$$\forall a \in A,\, t \in T_a^+,\, and\ t' \in T \qquad o_{t,t'}^a [F_t - TT(t, t') + RT(t, t')] \geq 0 \tag{8}$$

$$\forall a \in A,\, t \in T \qquad x_t^a \in \{0, 1\} \tag{9}$$

$$\forall a \in A,\, t \in T_a^+,\, and\ t' \in T \qquad o_t^a \in \{0, 1\} \tag{10}$$

$$\forall a \in A,\, k \in T_a^+ \qquad z_k^a \in \{0, 1\} \tag{11}$$

where in Equation (1) every robot a has an initial starting location, in Equation (2) every task gets at most a robot. In Equation (3), no robot surpasses its energy constraints. In Equation (4), no robot exceeds the total surpasses' time constraints. In Equation (5), every task has just one predecessor. In Equation (6), every last task has just one successor. In Equation (7), every robot has a last task. In Equation (8), task's time between two successive tasks is long enough to allow for travel time including robots' rotation time.

Equation (9) indicator: robot a performs the task t. Equation (10) indicator: robot a performs the task t' after task t. Equation (11) indicator: t is robot's last task.

In the optimization formulation, the objective function $f(.)$ as a cost function can be minimized or maximized. It can also be single or MO. Thus, as we introduce an MO problem, first the MO problems are briefly described and then the considered objectives are described in the rest of this section.

2.2 Multi-objective optimization and proposed objective functions

In multi-objective problems (MOPs), the favorable objectives are in conflict with each other. Hence, it is not possible to optimize all of the objectives simultaneously. For MOPs, the general formulation is minimizing the objective as below:

$$\min f(x) = \{f_1(x), f_2(x), \ldots, f_n(x)\}, x \in Q, n \geq 2 \tag{12}$$

where $f(x)$ represents the set of objectives. Variable x is the n-dimensional decision variable. x is constrained to feasible region Q which is limited by K-equality and J-inequality constraints (Coello *et al.*, 2007). The PF is a set of non-dominated chosen optimal solutions. The objectives of PF cannot be improved without losing at least one other objective. Let's assume the solution x^*. The solution x^* is referred to dominated solution by another solution s, if and only if s is equal to or better than x^* with regard to all objectives $\forall i : f_i(x^*) \leq f_i(x)$, $i \in \{1, 2, .., n\}$. In addition, x^* should have at least one objective better than x in one objective $\exists i : f_j(x^*) < f_j(x), j \in \{1, 2, .., n\}$.

As described before, various objectives of a multi-objective optimization problem (MOOP) conflict with each another. Thus, classical optimization problems usually fail to attain true Pareto-optimal solutions. Different existing MO algorithms have been successfully used to solve MOOPs including non-dominated sorting genetic algorithm (NSGA-II) (Deb *et al.*, 2002), state-of-the-art NSGA-III (Deb & Jain, 2014), and MOEA/D (Zhang & Li, 2007). Recent works on MOEAs have shown that the NSGA-III and MOEA/D algorithm performance are two of the best MOEAs. Thus, in this paper, the provided algorithm's MOEA is modified based on NSGA-III and MOEA/D algorithms.

Based on the described MO structure of TOPTWR, as mentioned earlier, in this paper we consider the following five objectives for MOHH-TOPTWR.

(1) Maximizing profit collected for all tasks (f_1)

$$\max f_1 = \sum_{a \in A} \sum_{t_a^+ \in T_a^+} P_{t_a^+} \tag{13}$$

(2) Minimizing the sum of the tasks' completion times (f_2)

$$\min f_2 = \sum_{a \in A} \sum_{t_a^+ \in T_a^+} DUR_{t_a^+} + TT_{t_a^+, t_a^+}, + RT_{t_a^+, t_a^+}, \tag{14}$$

(3) Minimizing robots' total energy consumption (including robots' physical rotations between tasks along with their walking distance), (f_3)

$$\min f_3 = \sum_{a \in A} \sum_{t_a^+ \in T_a^+} W_a^t DUR_{t_a^+} + E_a^S TT_{t_a^+, t_a^+}, + E_a^R RT_{t_a^+, t_a^+}, \tag{15}$$

(4) Minimizing robots' fair energy consumptions (f_4)

$$\min f_4 = \sqrt{\frac{1}{|A|} \sum_{a \in A} (Total_a - \mu)^2}, \mu = \frac{1}{|A|} \sum_{a \in A} Total_a$$

$$Total_a = \sum_{t_a^+ \in T_a^+} W_a^t \, DUR_{t_a^+} + E_a^S TT_{t_a^+, t_a^+}, + E_a^R RT_{t_a^+, t_a^+}, \tag{16}$$

(5) Minimizing tasks delays after their closing time (f_5)

$$\min f_5 = \sum_{a \in A} \sum_{t_a^+ \in T_a^+} Delay(C_{t_a^+}) \tag{17}$$

Maximizing f_1 aims to maximize the total profit in rescue application regarding the constraints defined in the proposed MIP formulation. Minimization of f_2 aims to reduce the total tasks' accomplishment time due to rescue application. Minimization of f_3 aims to reduce the total consumed energy of the robots. This is useful when (i) the number of robots and their tasks' maximum time T^{Max} is not restricted and robots can easily collect maximum profit in f_1, (ii) the robots have limited energy, and (iii) we need robots on another rescue mission, for instance, by completing a mission first on one building with some tasks and then completing another mission on the next building. Also, it is different from f_2, because the energy consumption of each robot x is different for each task t. Objective f_4 aims to minimize fair energy consumption between all robots. This is useful when robots have limited energy, and we prefer not to have an out-of-battery robot for later missions. Finally, Objective f_5 aims to minimize the total delay in performing assigned tasks regarding their closing time $C_{t_a^+}$ for all robots, where $Delay\ (C_{t_a^+})$ is a function that computes delay in performing an assigned task. In other words, this objective aims to start all tasks before their closing time.

3 Proposed algorithm

In this section, we describe the proposed MOHH-TOPTWR. In this regard, first, we describe the hyper-heuristics methodology and UCB selection strategy, followed by an explanation on the procedures and operators of the proposed MOHH-TOPTWR.

3.1 Hyper-heuristics

In the last two decades, different heuristic algorithms and other search techniques have been proposed successfully. One of the stimulating issues to solve a problem is to select a suitable approach among a major range of algorithm choices. A hyper-heuristic is a high-level heuristic that uses a set of LLHs. Hyper-heuristics usually select or generate heuristics (Burke *et al.*, 2010). So a hyper-heuristic can be utilized to select the most appropriate heuristics and find the best method or sequence of LLHs (Chakhlevitch & Cowling, 2008; Burke *et al.*, 2013). This was motivated by bringing less work because of the difficulties regarding the application of conventional search techniques, in which significant number of parameters must tune. Therefore, in this paper, hyper-heuristic is defined (Burke *et al.*, 2010) as an automated methodology for selecting or generating heuristics to solve hard computational search problems. In Burke *et al.* (2010), the authors also proposed a classification for hyper-heuristics consisting of two main dimensions including (i) the heuristic search space nature and (ii) the sources of heuristic feedback as follows. (i) construction heuristics starting with empty solutions or (ii) perturbation heuristics, which start with complete solutions (then iteratively improve solutions).

In this paper, hyper-heuristic with an online selection of perturbation heuristics is used in which search operators for MOEA is selected (Coello *et al.*, 2007). Moreover, hyper-heuristic presented here uses a profit-based heuristic selection method: sliding multi-armed bandit (SlMAB) proposed in Fialho *et al.* (2010).

3.2 Upper confidence bound

The MAB problem consists of a set of K-independent arms in which each arm has an unknown probability to obtain a reward (Mahajan & Teneketzis, 2008). So the goal is to maximize the accumulated reward by moving the arms in an optimal sequence. Inspired by the MAB, UCB selection strategy is proposed in Auer *et al.* (2002). In the UCB selection strategy, the *i*th given operator is associated with two values: (i) an estimated reward $\hat{r}_i$ for measuring the empirical reward of the *i*th operator over time and (ii) an

operator executed times-dependent confidence interval. Equation (18) shows the UCB method's operator selection with the best value:

$$\max_{i \in I}(\hat{R}_{i,t} + S\sqrt{2\log N/n_{i,t}}) \tag{18}$$

where i indicates the ith operator in a set of predefined operators I, S is the scaling factor to control the trade-off between the exploration and exploitation, and $\hat{R}_{i,t}$ is the gained average reward by the ith operator to that moment. Note that to have more exploitation the parameter S must be decreased and for exploration vice versa. $n_{i,t}$ indicates the number of times the ith operator was executed to that moment and N is the overall number of operator executions at that moment.

The SlMAB algorithm is an original dynamic variant of the standard MAB. The SlMAB is used to solve the exploitation versus exploration trade-off in static settings optimally (Fialho *et al.*, 2010). The SlMAB introduces a memory length adjustment and a credit assignment function. These allow a faster distinction of changes in the performance of LLHs. The SlMAB uses two main functions to compute the profit of each operator in Equation (19):

$$\hat{R}_{i,t+1} = \hat{R}_{i,t}(W/W + (ts - ts_i)) + r_{i,t}(1/n_{i,t} + 1) \tag{19}$$

Equation (19) shows how SlMAB computes the reward estimate $\hat{R}_{i,t+1}$. In this Equation i is the ith operator of a set of defined operators; ts is the current time step or iteration, ts_i is the last time step that operator i was applied, $r_{i,t}$ is the the ith operator's given immediate or raw reward at the time step ts, W indicates the size of the sliding window (SL), and $n_{i,t}$ is the ith operator's number of applied times until the time step ts. The SL is a type of memory that stores the last W rewards obtained by all operators. W can be obtained in several ways, but in Fialho *et al.* (2010) the authors concluded that extreme credit assignment, which defines the reward of an operator as the best reward found in the SL, is the most robust operator selection method in combination with SlMAB.

Function includes the credit assignment component of SlMAB. This component is to maximize the chosen operator in Equation (18).

$$n_{i,t+1} = n_{i,t}(W/W + (ts - ts_i)) + (1/n_{i,t} + 1) \tag{20}$$

where $n_{i,t}$ is the counter of ith operator until the time step ts (application frequency). We adapted the SlMAB implementation to make it more well matched with the MOHH-TOPTWR proposed in this paper. Our main motivations for using SlMAB was the good learning capability of SlMAB and its excellent results in other works (Fialho *et al.*, 2010; Guizzo *et al.*, 2017).

3.3 MOHH-TOPTWR

As described before in this paper, an MOHH-TOPTWR. The MOHH-TOPTWR is using online hyper-heuristic for selecting different LLHs. The LLHs used by MOEAs in order to solve the TOPTWR. The primary goal of MOHH-TOPTWR is to select the best combination of the state-of-the-art heuristics in the literature for each mating, and best existing mutation and crossover operators for similar problems in TOPTW. The hyper-heuristic used in this paper profits each LLH, updates and selects the best one in each mating. In this regard, it uses parents for generating offspring. The main steps and components of the MOHH-TOPTWRs are shown in Figure 1.

In the first step of the proposed MOHH-TOPTWR, hyper-heuristic is initialized using some inputs given by the user as follows:

(i) An MOEA and its parameters to perform the optimization using the objective functions.
(ii) A selection method to select the next best LLH in each mating of the MOEA.
(iii) A set of LLHs to generate solutions compatible with the problem representation.
(iv) The objective functions to evaluate the solutions.

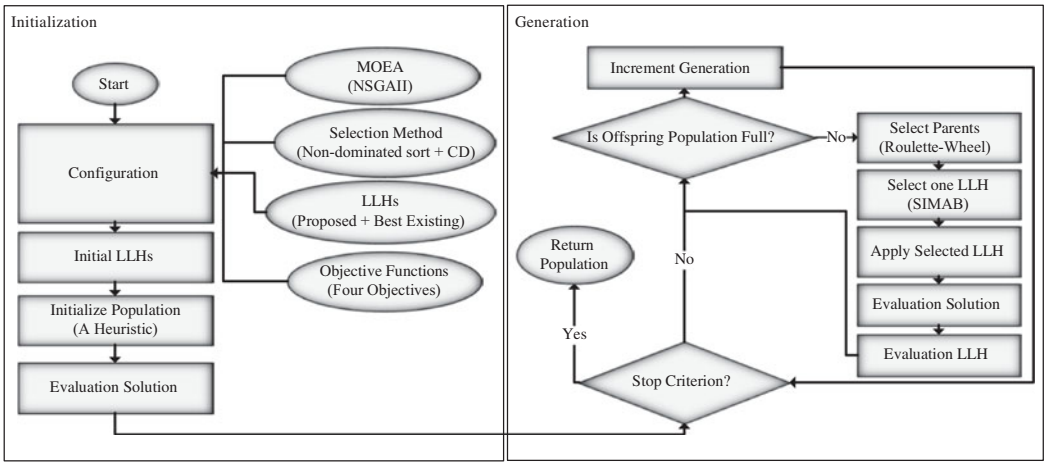

Figure 1 Multi-objective and evolutionary hyper-heuristic algorithm for TOPTW based on the humanoid robot's characteristics in the rescue applications (MOHH-TOPTWR) workflow

In this paper, an initial solution is constructed by inserting tasks subsequently into one of the paths based on Roulette-Wheel Selection method (Goldberg, 1989). We use both NSGA-III (Deb & Jain, 2014) and MOEA/D (Zhang & Li, 2007) as the MOEAs to solve MOHH-TOPTWR.

The objective functions described in Section 2.2, though are not necessarily conflicting, may be optimized simultaneously and independently. Thus, the optimization process may find several non-dominated solutions regarding the five objectives described. The advantage is enabling the user to choose the non-dominated solution based on the rescue environment that best fits the PF purposes. The problem is represented as an array of tasks and optimized as a permutation problem. Due to this, we used the best existing new LLHs specific for this kind of problem. Moreover, we also implemented and adapted SlMAB as the selection method (see Sections 3.2 (Fialho *et al.*, 2010)

The second step of MOHH-TOPTWR is initializing LLHs. Each LLH has an associated value for the two measures as $r_{i,t}$, and ts. These values are updated in each genetic mating, which is used in Section 3.2. To compute $r_{i,t}$ and ts, the following steps were followed:

(i) Reward measure ($r_{i,t}$): $r_{i,t}$ is an immediate reward given to LLH's performance. The greater reward is a better recent performance by evaluating both parent and offspring domination states (between 0 and 1). For each LLH, the $r_{i,t}$ value is computed by applying the ith LLH (such as mating parents and the generated offspring) as shown in Equation (21).

$$r_{i,t} = \frac{|\{o \in O | \exists p \in P : p > O\}|}{|P| \times |O|} + \frac{|\{o \in O | \exists p \in P : p = O\}|}{|P| \times |O| \times 3} \tag{21}$$

where $r_{i,t}$ is the computed reward value of the ith applied LLH. p is a parent in selected parents mating P, o is the offspring generated at the same mating O. This measure has a straightforward implementation using the explicitly well-known Pareto-dominance concept.

(ii) Time measure (ts): Time measure ts indicates the count of the past matings since the last mating using that LLH. So the higher value of the ts indicates that LLH has been idle (ts is within the interval $[0, \infty]$). The update rules for ts are straightforward at each mating, and the ts value for the LLHs that are not applied increases in that mating (for the applied one the counter set to 0).

According to MOEA initialization strategy, (usually this is random), the population is initialized. However, based on the literature review, we use a semi-random insert operator (greedy construction heuristic), which is a multi-start version of the ILS operator proposed in Gunawan *et al.* (2017) (operators will be described in Section 3.4 in detail). At each generation, the offspring population is generated after evaluating the initial population using the proposed objective functions. This generation procedure is performed until the stop criterion is met. In this regard, one parent or two parents are selected based on

Table 2 Operators used in the hyper-heuristic proposed in this paper

OP#	Crossover/local search	OP#	Mutation operators
1	PMX (Zhu *et al.* 2018)	1	Reverse (Vincent *et al.*, 2017), (Lin & Vincent, 2017), (Lin & Vincent 2012),
2	Single point (Zheng *et al.*, 2017)	2	Swap 1 (Vincent *et al.*, 2017), (Gunawan *et al.*, 2017), (Lin & Vincent, 2012), (Lin & Vincent, 2017),
3	Iterated local search (ILS) (Gunawan *et al.*, 2017)	3	Swap 2 (Vincent *et al.*, 2017), (Gunawan *et al.*, 2017), (Lin & Vincent, 2012)
4	Simulated annealing ILS (SAILS) (Gunawan *et al.*, 2017)	4	Insert (Vincent *et al.*, 2017), (Lin & Vincent, 2017), (Gunawan *et al.*, 2017), (Lin & Vincent, 2012)
		5	Replace (Huang *et al.*, 2018), (Vincent *et al.*, 2017), (Gunawan *et al.*, 2017), (Lin & Vincent, 2012)

PMX = partially matched crossover; SAILS = simulated annealing (SA) iterated local search heuristic.

the nature of LLH to generate a new offspring. The selection strategy used in this work is Roulette-Wheel Selection method (Goldberg, 1989). Thereafter, using the hyper-heuristic method, an LLH is selected to apply on selected parents and to produce new offspring. In this vein, if new solutions violate any constraint, a recovery algorithm is applied. If not, produced solutions are evaluated by the proposed objective functions, and this evaluation indicates the performance of the selected LLH (note that the produced offspring are added to the offspring pool). The *ts* of each selected LLH is set to 0, and the other non-selected LLHs' *ts* are incremented by one. This routine continues until the new offspring population pool is not full.

When the offspring population generations end, MOHH-TOPTWR returns the non-dominated solutions of the current population. The non-dominated solutions returned by MOHH-TOPTWR are the solutions that present the best trade-off between the introduced objectives. This is achieved by the defined MOEA, which in this paper are based on the NSGA-III method (Deb & Jain, 2014) or the MOEA/D (Zhang & Li, 2007) structure.

The following subsection presents the LLHs implemented in this work.

3.4 Low-level heuristics

MOHH-TOPTWR can use any type of operators as LLH. In this work, the recent state-of-the-art approaches for TOPTW such as local searches are implemented as LLHs. Hence, the crossover operator, local searches, or a combination of a mutation operator and a crossover operator is used.

In this paper, we adapted and used four types of operators as crossover or local search operators and five mutation operators, all for permutation encoding based on the TOPTWR definitions. In this regard, the best efficient operators are extracted from the recent related algorithms on TOPTW. Most of these operators are common in most of the proposed approaches in the literature (Vansteenwegen *et al.*, 2009; Cura, 2014; Wang *et al.*, 2016; Guizzo *et al.*, 2017; Gunawan *et al.*, 2017; Lin & Vincent, 2017; Nunes *et al.*, 2017; Vincent *et al.*, 2017). Table 2 presents the operators used in this work.

In the rest of this section, we first describe the crossover operators utilized, then explain mutation operators, and, finally, show how LLHs in the proposed hyper-heuristic are implemented.

In this paper, we utilized two crossovers partially matched crossover (PMX) and two-point crossover, which use two parents and produce two offspring. This was because of their popularity and extensive use in different MOEA algorithms (Zhu *et al.*, 2018).

On the other hand, we utilized two best local search operators, which are the core proposed metaheuristics in the literature. Based on the surveys in TOPTW, the proposed algorithms such as ILS obtained the best results compared to all other algorithms (Gunawan *et al.*, 2016). These local search approaches use one parent and produce one offspring. In the most recent paper an ILS and SAILS (Gunawan *et al.*, 2017) are proposed. These metaheuristic approaches are based on efficient local searches. The results of the SAILS, the best existing one in the literature, dominated most other proposed algorithms in TOPTW such as ant colony system (Montemanni & Gambardella, 2009), variable neighborhood search (Tricoire *et al.*, 2010), slow SA (Lin & Vincent, 2012), granular variable neighborhood search (Labadie *et al.*, 2012), iterative three-component heuristic (Hu & Lim, 2014), and hybridized greedy randomized iterated local search (Souffriau *et al.*, 2013). As the authors explained in Gunawan *et al.* (2017), the order of the used operators is decided after conducting some preliminary experiments; whereas in this paper, we aimed to find the best order in all possible best operators. Thus, we used its core local searches as a multi-start ILS and considered it as LLH. Despite the use of mutation operators such as reverse, etc., their metaheuristics platform is not capable of being used in the best combinations.

All of the five mutation operators shown in Table 2 are applied to one given solution approach (chromosome). These mutation operators are described as follows:

Reverse operator (Vincent *et al.*, 2017):

Step 1: Randomly select a robot a in the set t^- and a task t_a^+ in the set T_a^+, then select a random reverse count RC, $0 < RC < |T_a^+|$, and reverse RC count of the order of visiting tasks starting t_a^+.
Step 2: Evaluate the feasibility of the modified path and compute the objective functions.
Step 3: If the feasibility is maintained and the objective of the modified path is improved, then keep the modified solution; otherwise, return the solution to the Step 1.

Swap 1 operator (Vincent *et al.*, 2017):

Step 1: Randomly select a robot a in the set.
Step 2: Randomly select two tasks t_a^+ and $t_a^{+'}$ in the same T_a^+.
Step 3: Swap task t_a^+ with $t_a^{+'}$.
Step 4: Evaluate the feasibility of the modified task's order for the selected robot a and the objective functions.
Step 5: If the feasibility is maintained and the objective functions of the modified path are improved, then keep the modified path; otherwise, return the path to Step 1.

Swap 2 operator (Vincent *et al.*, 2017):

Step 1: Randomly select a robot a in set A and tasks t_a^+ in T_a^+.
Step 2: Randomly select another robot a' in set A and tasks $t_a^{+'}$ in T_a^+.
Step 3: Swap task t_a^+ with the task $t_a^{+'}$.
Step 4: Evaluate the feasibility of both the modified task's order in a, a', and their objective functions.
Step 5: If the feasibility is maintained and the objective functions of the modified path are improved, then keep the modified path; otherwise, return the path to Step 1.

Insertion operation (Lin & Vincent, 2017):

Step 1: Randomly select a robot a in set A and tasks t_a^+ in T_a^+.
Step 2: Randomly select a task t^- in T^-(reserve list).
Step 3: Insert t^- next to t_a^+.

Step 4: Evaluate the feasibility of the modified task's order and objective functions.
Step 5: If the feasibility is maintained and the objective functions of the modified path are improved, then keep the modified path; otherwise, return the path to Step 1.

Replace operator (Vincent *et al*., 2017):

Step 1: Randomly select a robot a in set A and tasks t_a^+ in T_a^+.
Step 2: Randomly select a task t^- in T^-(reserve list).
Step 3: Swap task t_a^+ with t^-.
Step 4: Evaluate the feasibility of the modified task's order and objective functions.
Step 5: If feasibility is maintained and the objective functions of the modified task's order in a is improved, then keep the modified path; otherwise, return the path to step 1.

In the following section, first we briefly describe the crossover and local search approaches used in this work and then the proposed SAILS.

Partially matched crossover (Zhu *et al*., 2018):

One of the most common types of crossover with this characteristic is PMX (Michalewicz, 2013; Karakatič & Podgorelec, 2015). In PMX, two chromosomes and two different crossing points are randomly chosen. Then each section is transferred from one chromosome into another chromosome (proceeds by position-wise exchanges). In other words, a portion of one parent's genes is exchanged with a portion of the other parent's genes and the remaining genes are copied or regenerated by mapping.

Single-point crossover (Zheng *et al*., 2017):

In Zheng *et al.* (2017), an adaptive heuristic search algorithm GA (genetic algorithm) was used, which is commonly used to deal with optimization and search problems by relying on bio-inspired operators. For the route optimization process, authors adopt a single-point crossover to generate a high-quality route set. This process involves three steps: (1) two routes are selected randomly to be the parents, (2) a cut point is randomly determined, and (3) the new routes (named the offspring) are determined as concatenations of parts from the two parents.

Iterated local search (Gunawan *et al*., 2017):

In this work, three components of ILS, namely, perturbation, local search, and acceptance criterion are taken into consideration.

Step 1 (initialization): The initial solution that is generated by the greedy construction heuristic is treated as the current solution in ILS.
Step 2 (perturbation): Two different steps including exchange path and shake are implemented in perturbation. In this vein, if the number of iterations without improvement is larger than a predefined specific threshold, the exchange path is executed; otherwise, the shake is selected.

Exchange: Changes the order of the paths in the solution by swapping two adjacent paths every time.
Shake: Shake step is adopted from the one proposed in Vansteenwegen *et al.* (2009) with some modifications in which two integer values, CONS and POST, are used. The value POST indicates the first position of the removing process in a particular path, while the value CONS indicates how many consecutive tasks remove a particular path.

Table 3 Low-level heuristics composition in MOHH-TOPTWR

		Mutation				
Local search/crossover	**–**	**Reverse**	**Swap 1**	**Swap 2**	**Insert**	**Replace**
ILS	H1	H2	H3	H4	H5	H6
SAILS	H7	H8	H9	H10	H11	H12
PMX	H13	H14	H15	H16	H17	H18
Two point	H19	H20	H21	H22	H23	H24

MOHH-TOPTWR = Multi-objective and evolutionary hyper-heuristic algorithm for TOPTW based on the humanoid robot's characteristics in the rescue applications; ILS = Iterated local search; PMX = partially matched crossover; SAILS = simulated annealing (SA) iterated local search heuristic.

Step 3 (local search): In local search, the authors used six different operations consecutively. In which two other operators, insert and replace, contribute to improving the quality of the solution. Thus, we just used these two operators as the local search core in ILS. (Note that other operators will be used in the proposed hyper-heuristic platform.)

Insert: The purpose of insert is to insert one unscheduled node (reserve task) to a particular path with its improvement probability. It is started by generating a feasible insertion points probability list per reserve node. Then it inserts a node with Roulette-Wheel selection (for details refer to Gunawan *et al.*, 2017).

Replace: The replace operation is started by selecting the path with the highest remaining travel time and selecting one node with the highest profit. It is started by searching a feasible replace points per reserve node. Hence, once this operation is successful, the process will continue to the next reserved node and repeat the operation. This will continue until there is no possible replacement.

Step 4 (acceptance criterion): If the current solution is better than the previous solution, it updates the best-found solution so far.

Step 5 (intensification strategy): This work included the intensification strategy in ILS, which is the main difference with the standard ILS. The idea of the intensification strategy is that if improvement is not achieved for a certain number of iterations, it restarts the search from the best-found solution.

Simulated annealing iterated local search (Gunawan *et al.*, 2017):

SAILS is a hybridization of ILS and SA, which helps SA to escape from local optima. The SA escapes from a local optimum by accepting a worse solution with a probability that changes over time. This SA algorithm works with three parameters T_0 to the initial temperature, α as a coefficient used to control the speed of the cooling schedule ($0 < \alpha < 1$), and *InnerLoop* as a number of iterations at a particular temperature.

In the beginning, the current temperature *Temp* is defined as equal to T_0 and decreases after *InnerLoop* iterations by using the following formula: $Temp = Temp \times \alpha$. In this work at a particular value of temperature (Temp), two components of ILS (perturbation, local search) with specific tuned parameters are applied. Other steps of SAILS are the same as that described for ILS in the previous paragraph.

Note that for more details of implementation, parameter tuning, and algorithm of both ILS and SAILS refer to (Gunawan *et al.*, 2017).

Table 3 presents the LLHs used in this paper, which is composed of the operators described in subsection 3.4. As shown in this table, we included LLHs composed of local search or crossover operators, because this kind of operator is a distinguishing feature of evolutionary.

Table 4 Benchmark instances used in the study

Set#	Reference	Reference name	Instance set	Number of instances	Number of tasks (nodes)	Number of robots (paths)	Tasks types
1	Righini and Salani (2009)	Solomon	c100_100_2, r100_100_2, rc100_100_2	29	100	2–4	4
2		Cordeau	pr10_1	10	48-228	2–4	4
3	Montemanni and Gambardella (2009)	Solomon	c200_100, r200_100, rc200_100	27	100	2–4	4
4		Cordeau	pr11_20	10	48-228	2–4	4

4 Experimental results

In order to evaluate the performance of the proposed hyper-heuristic algorithm, experiments were implemented in C# programming language on a laptop (i5-6360U 2.00 GHz with 8 GB RAM). Average running time (seconds) of the algorithms of over 20 runs for each instance was 600 seconds. We know unlike the population-based EA, there is no explicit concept of generation in ILS approaches. Thus, the running time of the proposed hyper-heuristic algorithm was set as the stopping criterion for other compared algorithms in same time durations for a fair comparison.

4.1 Benchmark instances

The main aim of this paper is to propose new algorithms for rescue and general humanoid robots' energy consumption instances. In order to evaluate the performance of all the discussed algorithms, the algorithms were tested on the extended existing benchmark instances on TOPTW, which we presented with some modifications and supplementary. Hence, all algorithms were evaluated on the two well-known adopted sets of benchmark instances with supplemental package described below.

The benchmark TOPTW instances were initially proposed in Righini and Salani (2009) in which 39 problem instances generated from Solomon's (Solomon, 1987) and Cordeau's instances (Cordeau *et al.*, 1997). Also, in Montemanni and Gambardella (2009) another set of 37 instances was proposed for the OPTW (see Table 4). These benchmark instances can be downloaded from Tang and Miller-Hooks (2005). Based on the described TOPTWR in this paper, it is slightly different from TOPTW as follows. (i) Starting time window of tasks is considered open (in a rescue environment all tasks need to start and help victims) and (ii) robots are considered heterogeneous and consumed heterogeneous energy, whereas in the existing TOPTW benchmark instances heterogeneous energy consumptions per robots are not introduced. (iii) In addition to the maximum time constraints, T^{Max}, energy constraints are observed for each robot E_a^{Max}. Thus, we produced a supplemental package for these benchmark instances including energy consumptions of heterogeneous humanoid robots per task. Also, we provided the energy consumption of robots by traveling a distance between two tasks. Table 4 provides the summary of the modified benchmark instances.

In addition to the standard benchmark, we defined four types of tasks in the last column. This column indicates that in each instance four types of task are included in which they are classified by task numbers. As a result, for instance, set with 100 number of tasks, tasks with number 1–25 will be in task type 1, tasks with numbers between 26 and 50 will be in task type 2, etc. For instance, a set with 48 number of tasks, the tasks with numbers 1–12 will be in task type 1, tasks with numbers 13–24 will be in task type 2, etc. Finally, for instance, set with 228 number of tasks, tasks with numbers 1–57 will be in task type 1, tasks with numbers 58–114 will be in task type 2, etc.

Table 5 Robots' energy consumption per robots' walk and task type

	Energy consumption						
	Straightforward walking average (J/s) E_a^s	**Rotating average (J/s)** E_a^R	**Tasks execution (J/s)** W_t^a				F_a^{Max}
Robot #	–	–	**Type A**	**Type B**	**Type C**	**Type D**	
1	2	3	1	3	5	6	$T^{Max} \times 6$
2	3	4	1	3	6	7	$T^{Max} \times 7$
3	3	4	2	4	6	8	$T^{Max} \times 8$
4	2	3	3	5	7	8	$T^{Max} \times 9$

Table 5 presents robots' energy consumption per robots' straightforward walk, rotation, and the task workload ($T^{W_t^a x_t^a}$) for each task type in joule/second unit (note that data are an approximation of our existing humanoid robots energy consumptions with Dynamixel actuators and 20 degrees of freedom. Using this table, we can compute each robot's energy consumptions when the robot walks between tasks, multiplying the energy consumed during of the related robot by traveling the distance in the required time, which can be computed from the given benchmark instances presented in Table 4. To obtain the energy consumption of each robot for each task type, we need to multiply the time required per task ($C_t - O_t$) for a given benchmark instances (Table 4) by each task execution energy consumption. Also, in the last column of this table provides the maximum heterogeneous energy of each robot $E_a^{\max}$. These values are computed by the related given $T_{\max}$ per benchmark instances.

4.2 Experiment organization

In order to evaluate the proposed algorithm, we configured the recent state-of-the-art algorithms in an MO platform and adapted them to the described problem and then the performance of each was compared. In this regard, we compared the following algorithms.

(I) SAILS (Gunawan *et al.*, 2017): This algorithm outperformed all six recent algorithms in TOPTW. In addition, in the same research paper authors proposed one ILS algorithm.
(II) Multi-start SA (MSA) (Lin & Vincent, 2017): One recently proposed algorithm on TOPTW, which outperformed ABC (Cura, 2014), and SA (Lin & Vincent, 2017).
(III) Multi-objective local search (MOLS) (Wang *et al.*, 2016): Most recently proposed algorithm on vehicle routing problem simultaneous delivery and pickup and time window, which outperformed multi-objective memetic algorithm (Wang *et al.*, 2016).
(IV) MOHH-TOPTWR-NSGA-III: The proposed hyper-heuristic algorithm that applies NSGA-III as MOEA in the proposed hyper-heuristic.
(V) MOHH-TOPTWR-MOEA/D: The proposed hyper-heuristic algorithm that applies MOEA/D as MOEA in the proposed hyper-heuristic.

Each algorithm was executed 20 times for each instance. For first three experiments, we defined both NSGA-III (Deb & Jain, 2014) and MOEA/D (Zhang & Li, 2007) as the MOEAs because of their popularity, superiority, and shortcoming against each other. Table 6 provides the essential parameters of algorithms. The parameters of each first three algorithms are set as in the related paper (for more detail refer to the related paper). Also, the MOEA/D parameters were set according to Wang *et al.* (2016), and the NSGA-III parameters were set according to Deb and Jain (2014). The parameters required by Sliding Multi-Armed Bandit (SIMAB) were set after an empirical tuning. This tuning was done to properly balance the different intervals between the metrics $r_{i,t}$ and ts.

Table 6 Important parameters used in the MOHH-TOPTWR

Common parameters	SIMAB	MOEA/D	MOLS	SAILS	MSA
Population size: 210 Max generation time: 600 seconds	W: 140 C: 5	ϵ: 0.05 T: 5	MaxDepth: 10 ϵ: 0.05 T: 5 H: 8	f: 5 THRESHOLD2: 20 THRESHOLD3: 3	T0: 3 A: 0.9 $N_{\text{non-improving}}$: 15 *Iiter: 5000*

SIMAB = Sliding Multi-Armed Bandit; MOEA/D = MO evolutionary metaheuristics algorithm based on decomposition; MOLS = Multi-objective local search; SAILS = simulated annealing iterated local search heuristic; MSA = multi-start simulated annealing.

For both NSGAII and MOEA/DD, usually, a high crossover probability and a very low mutation probability should be defined. For proposed MOHH-TOPTWR-NSGA-III and MOHH-TOPTWR-MOEA/D, we did not use such probabilities because if the hyper-heuristic algorithm selects an LLH, its operators are mandatorily applied. Hence, there is no need for expert user to select the operators' configurations manually. So, elimination of manually configuration process is the main advantage of using a hyper-heuristic algorithm included with a learning part for the LLH's selections (Guizzo *et al.*, 2017).

4.3 Performance metrics

To measure the performance of an MO optimization algorithm, it is important to select a performance metric that can give a comprehensive comparison and that traditional single-objective approaches cannot be used (Zitzler *et al.*, 2003). Based on this fact, in this paper, like most of the MOP algorithms comparisons (Wang *et al.*, 2016), we used the following three metrics.

(i) Inverted generational distance (IGD) (Zhang *et al.*, 2008) was used to measure both the convergence and diversity of the non-dominated solutions obtained. The IGD calculates the distance of the true PF to another front. Thus, the lower the IGD of a front, the better the front is. Assume SL^* as non-dominated solutions of an MO algorithm and PF as a true Pareto-optimal set, then we can compute IGD in Equation (22).

$$D(SL^*, PF) = \frac{1}{|SL^*|} \sum_{v \in SL^*} Dis(v, PF) \tag{22}$$

where $Dis(v, PF)$ is the minimum Euclidean distance between v and the points in PF

(ii) Hypervolume (HV) (Zitzler & Thiele, 1999) is used to specify the area in the objective space that was dominated at least by one of the non-dominated solutions. The larger the HV, the closer the corresponding non-dominated solutions toward the PF. Therefore, it reflects the closeness of the non-dominated solutions to the PF. HV also measures both the convergence and diversity.

(iii) Coverage metric (C-metric) (Zitzler, 1999): C-metric or simply coverage is a commonly used metric for comparing two sets of non-dominated solutions. C-metric is a binary indicator. Let A and B be two approximation sets. $C(A, B)$ gives the fraction of solutions in B that are dominated by at least one solution in A. Hence, $C(A, B) = 1$ means that all solutions in B are dominated by at least one solution in A, while $C(A, B) = 0$ implies that no solution in B is dominated by a solution in A, see the Equation (23).

$$C(A, B) = \frac{|\{b \in B \mid \exists a \in A : a \geq b\}|}{|B|} \tag{23}$$

where $a \geq b$ suggests that solution a dominates or equals to solution b.

Note that since the true Pareto-optimal set is unknown in this study, to compute the performance metrics, Pareto-optimal is approximated by the overall best approximately seen non-dominated solutions obtained by all algorithms in all experimental runs.

4.4 Experimental evaluation

To evaluate the algorithms referred to in Tables 7–10, we show the average values of IGD, HV, and C-metric for all 76 benchmark instances for all algorithms with two types of MOEAs (boldface values indicate superiority). Below the tables, we comprehensively discuss their results and compare their performances for three different numbers of robots.

A significant feature of the TOPTWR data set is that it may not hold for the different situations in rescue applications. Thus, previous single-objective or bi-objective optimization approaches for the benchmark instances cannot be directly used to solve the real-world TOPTWR instances (Martín-Moreno & Vega-Rodríguez, 2018). Thus, they cannot effectively solve the many objective TOPTWR considered in this paper.

Two proposed algorithms based on the hyper-heuristic method proposed in this paper were compared with the best existing literature algorithms on the MO platform and each other since there is no previous benchmark algorithm. We extended the existing benchmark instances for TOPTW, hence our algorithms can be seen as the benchmark algorithms for TOPTWR instances and, as a result, can be compared with the future research.

Average values of the described metrics over 20 independent runs of all algorithms on the benchmark instances were calculated and are shown in Tables 7–10. We should note that due to space limitations, the values are shown as average for the same benchmark series, and standard deviations of the described metrics are not presented in the tables. From Tables 7–10, the following can be observed.

(I) For 76 instances, MOHH-TOPTWR-NSGA-III outperforms all other algorithms in most instances. Specifically, in all three robots MOHH-TOPTWR-NSGA-III outperforms other algorithms in 70 instances and is outperformed by MOHH-TOPTWR-MOEA/D in 6 instances in terms of IGD. Regarding HV, MOHH-TOPTWR-NSGA-III outperforms all other algorithms in 72 instances and is outperformed by MOHH-TOPTWR-MOEA/D in 4 instances. This means that MOHH-TOPTWR-NSGA-III can obtain better results than other algorithms in terms of both convergence and diversity.

(II) MOHH-TOPTWR-NSGA-III significantly outperforms other algorithms in terms of *C*-metric as follows. In MOEA/D platform: 23 instances against SAILS, 23 instances against MSA, 24 instances against MOLS, and 15 instances against MOHH-TOPTWR-MOEA/D; and in NSGA-III platform 24 instances against SAILS, and MSA and 23 instances against MOLS. This means that MOHH-TOPTWR-NSGA-III can obtain better results than other algorithms in terms of both convergence and diversity. Consequently, in this problem, MOHH-TOPTWR-NSGA-III shows better convergence performance.

(III) Excluding MOHH-TOPTWR-NSGA-III, MOHH-TOPTWR-MOEA/D outperforms other algorithms regarding IGD, HV, and *C*-metric in most of the instances, which shows its hyper-heuristic superiority against other state-of-the-art algorithms.

Finally, the relative benefits of the proposed hyper-heuristic are briefly discussed. In both MOHH-TOPTWR-NSGA-III and MOHH-TOPTWR-MOEA/D, a single-objective local search can be easily adapted for MO-TOPTWR. However, combining objective-wise local search with genetic operators preserved the diversity among subproblems along with adopting optimization of each objective, which promotes the convergence and diversity of search. This is the reason that both MOHH-TOPTWR-NSGA-III and MOHH-TOPTWR-MOEA/D are better than other algorithms in terms of both convergences and diversity in most of the instances.

Table 7 Average values of IGD on the described benchmark instances; MOHH-TOPTWR-NSGA-III denoted as MTN and MOHH-TOPTWR-MOEA/D denoted as MTM (detailed results for Solomon and Cordeau instances with $|A| = 2$, $|A| = 3$, $|A| = 4$)

		IGD							
		NSGA-III				MOEA/D			
Robot #	**Instance (76)**	**SAILS**	**MSA**	**MOLS**	**MTN**	**SAILS**	**MSA**	**MOLS**	**MTM**
$\|A\| = 2$	c101-c109	0.19741	0.19701	0.19686	0.1931	0.19751	0.19779	0.19751	**0.19228**
	c201-c208	0.17924	0.17924	0.17909	**0.16634**	0.17792	0.17816	0.17792	0.17701
	pr1-pr10	0.21755	0.21755	0.21755	**0.17986**	0.21727	0.21723	0.21588	0.20995
	pr11-pr20	0.21872	0.21873	0.21874	**0.18029**	0.21787	0.21786	0.21736	0.20369
	r101-r112	0.24964	0.25033	0.24981	**0.22647**	0.25091	0.25156	0.25011	0.24729
	r201-r211	0.23782	0.23801	0.23801	**0.17814**	0.23745	0.23764	0.23764	0.23336
	rc101-rc108	0.17913	0.17923	0.17923	**0.15564**	0.17954	0.17953	0.17957	0.17576
	rc201-rc208	0.17965	0.17965	0.17965	0.13847	0.17969	0.17965	0.17969	**0.12676**
$\|A\| = 3$	c101-c109	0.19558	0.19583	0.19558	**0.18636**	0.1964	0.1965	0.19661	0.19275
	c201-c208	0.18145	0.18228	0.18475	**0.17398**	0.17969	0.17979	0.1799	0.18052
	pr1-pr10	0.21975	0.21979	0.21975	**0.20046**	0.21536	0.21541	0.2149	0.21608
	pr11-pr20	0.21944	0.21942	0.21944	**0.20542**	0.21862	0.21867	0.21826	0.21939
	r101-r112	0.25589	0.25605	0.25615	**0.24048**	0.25609	0.25616	0.25642	0.25615
	r201-r211	0.23912	0.23912	0.23912	0.23792	0.23923	0.23923	0.23923	**0.23776**
	rc101-rc108	0.17891	0.1789	0.17891	**0.1652**	0.17892	0.17917	0.17892	0.17807
	rc201-rc208	0.17967	0.17988	0.17967	0.17709	0.17945	0.17947	0.17942	**0.17565**
$\|A\| = 4$	c101-c109	0.19315	0.19357	0.19358	**0.18766**	0.19418	0.19439	0.19439	0.19283
	c201-c208	0.19808	0.18901	0.18525	0.18015	0.18213	0.18245	0.18526	**0.17941**
	pr1-pr10	0.21991	0.21983	0.22	**0.18365**	0.21976	0.21967	0.21976	0.21414
	pr11-pr20	0.21992	0.2198	0.21997	**0.17774**	0.21936	0.21924	0.21941	0.21199
	r101-r112	0.25178	0.25134	0.25172	**0.22981**	0.25687	0.2569	0.25724	0.25172
	r201-r211	0.23976	0.23976	0.23976	**0.23656**	0.23988	0.23991	0.23988	0.23884
	rc101-rc108	0.17949	0.17949	0.17949	0.1625	0.1799	0.17994	0.17994	**0.15937**
	rc201-rc208	0.18141	0.18124	0.18648	**0.17438**	0.18231	0.18147	0.18645	0.17853

MOHH-TOPTWR = multi-objective and evolutionary hyper-heuristic algorithm for TOPTW based on the humanoid robot's characteristics in the rescue applications; IGD = inverted generational distance; NSGA = non-dominated sorting genetic algorithm MOEA/D = MO evolutionary metaheuristics algorithm based on decomposition; MOLS = Multi-objective local search; SAILS = simulated annealing iterated local search heuristic; MSA = multi-start simulated annealing.

Table 8 Average values of HV on the described benchmark instances; MOHH-TOPTWR-NSGA-III denoted as MTN and MOHH-TOPTWR-MOEA/D denoted as MTM (detailed results for Solomon and Cordeau instances with $x_t^a = 1$)

		HV							
		NSGA-III				MOEA/D			
Robot #	**Instance (76)**	**SAILS**	**MSA**	**MOLS**	**MTN**	**SAILS**	**MSA**	**MOLS**	**MTM**
$\|A\| = 2$	c101-c109	0.023	0.1458	0.0432	**0.6438**	0.032	0.1352	0.1327	0.2255
	c201-c208	0.023	0.0926	0.0533	**0.6835**	0.07	0.1262	0.1379	0.3532
	pr1-pr10	0.03	0.1164	0.0437	**0.3766**	0.041	0.2007	0.2095	0.1922
	pr11-pr20	0.027	0.1186	0.1309	0.4171	0.0104	0.3566	0.2388	**0.4527**
	r101-r112	0.0553	0.2173	0.2196	**0.7881**	0.0513	0.2324	0.2275	0.3133
	r201-r211	0.097	0.0835	0.1009	**0.3992**	0.0104	0.0369	0.0615	0.1103
	rc101-rc108	0.0223	0.3126	0.1338	**0.6251**	0.0257	0.1521	0.218	0.3824
	rc201-rc208	0.029	0.0923	0.0799	0.2115	0.053	0.0355	0.0333	**0.2229**
$\|A\| = 3$	c101-c109	0.0383	0.0727	0.1429	**1.1821**	0.0416	0.1421	8.246	0.077
	c201-c208	0.063	0.1303	0.078	**1.1025**	0.0709	0.5344	13.758	0.1183
	pr1-pr10	0.037	0.0788	0.0271	**1.1667**	0.0334	0.1814	0.2286	0.0709
	pr11-pr20	0.0306	0.1046	0.0859	**1.1092**	0.0264	0.4031	0.3672	0.0847
	r101-r112	0.0948	0.1101	0.1849	**1.0449**	0.0954	0.1895	0.0797	0.1169
	r201-r211	0.0172	0.0297	0.0371	**2.0254**	0.0209	0.0447	0.0163	0.0322
	rc101-rc108	0.0622	0.3331	0.2214	**1.2702**	0.0686	0.4508	0.139	0.3295
	rc201-rc208	0.0164	0.0821	0.0802	0.8638	0.0232	0.3524	0.0677	**0.9782**
	c101-c109	0.1257	0.1766	0.2672	**1.2568**	0.1265	0.1746	0.1437	0.1794
$\|A\| = 4$	c201-c208	0.0053	0.2996	0.1166	**1.9827**	0.004	0.2484	0.2947	0.2927
	pr1-pr10	0.0525	0.2024	0.0429	**1.193**	0.0505	0.1613	0.4037	0.1959
	pr11-pr20	0.057	0.2392	0.2445	**1.3464**	0.0571	0.2092	0.5817	0.2332
	r101-r112	0.0878	0.3661	0.708	0.5545	0.0869	0.6678	0.5777	**0.6916**
	r201-r211	0.0285	0.4344	0.2689	**2.6127**	0.0335	0.5766	0.8388	0.2566
	rc101-rc108	0.0837	0.6729	0.8749	**1.1473**	0.0901	0.9097	0.3981	0.6037
	rc201-rc208	0.0228	0.1071	0.1328	**2.4111**	0.0288	0.1777	0.5421	0.9868

MOHH-TOPTWR = multi-objective and evolutionary hyper-heuristic algorithm for TOPTW based on the humanoid robot's characteristics in the rescue applications; HV = hypervolume; NSGA = non-dominated sorting genetic algorithm; MOEA/D = MO evolutionary metaheuristics algorithm based on decomposition; MOLS = Multi-objective local search; SAILS = simulated annealing iterated local search heuristic; MSA = multi-start simulated annealing.

Table 9 Average values of C-Metric; MOHH-TOPTWR-NSGA-III denoted as MTN and MOHH-TOPTWR-MOEA/D denoted as MTM on the Solomon and Cordeau instances benchmark instances, A1: MTN, A2: MTM, B_1: SAILS, B_2: MSA, B_3: MOLS (B_1, B_2, B_3 are with MOEA/D and detailed results for $\sum t \in T^{W_i^a x_i^a} \leq E_a^{Max}$)

		C metric													
		A1 : MTN, B_1 : SAILS		**A_1 : MTN, B_2 : MSA**		**A_1 : MTN, B_3 : MOLS**		**A_2 : MTM, B_1 : SAILS**		**A_2 : MTM, B_2 : MSA**		**A_2 : MTM, B_3 : MOLS**		**A_1 : MTN, A_2 : MTM**	
	Instances	**C (A_1,B_1)**	**C (B_1,A_1)**	**C (A_1,B_2)**	**C (B_2,A_1)**	**C (A_1,B_3)**	**C (B_3,A_1)**	**C (A_2,B_1)**	**C (B_1,A_2)**	**C (A_2,B_2)**	**C (B_2,A_2)**	**C (A_2,B_3)**	**C (B_3,A_2)**	**C (A_1,A_2)**	**C (A_2,A_1)**
$\|A\| = 2$	c101-c109	**0.595**	0.27	0.275	**0.33**	**0.735**	0.335	0.012	**0.46**	0.1001	**0.2725**	0.1124	**0.1925**	**0.695**	0.047
	c201-c208	**0.875**	0.145	**0.835**	0.01	**0.16**	0.01	**0.105**	0.042	**0.0975**	0.0675	**0.015**	0.0125	**0.915**	0.15
	pr1-pr10	**0.87**	0.252	**0.715**	0.078	**0.18**	0.005	**0.185**	0.1125	0.035	**0.0425**	0.005	**0.1825**	**0.815**	0.143
	pr11-pr20	**0.425**	0.073	**0.61**	0.221	**0.53**	0.006	**0.0075**	0.0025	**0.2125**	0.01	**0.035**	0.0175	**0.775**	0.252
	r101-r112	**0.83**	0.005	**0.875**	0.02	**0.02**	0.065	0.170	**0.0175**	**0.1875**	0.0225	0.1245	**0.2125**	**0.935**	0.215
	r201-r211	**0.82**	0.252	**0.82**	0.015	**0.91**	0.012	**0.295**	0.0225	**0.18**	0.0125	**0.3325**	0.2225	**0.755**	0.362
	rc101-rc108	**0.915**	0.154	**0.89**	0.244	**0.955**	0.171	0.1225	**0.3775**	**0.1225**	0.095	**0.255**	0.1925	**0.915**	0.227
	rc201-rc208	**0.55**	0.142	**0.755**	0.075	**0.79**	0.04	0.0725	**0.42**	0.055	**0.23**	0.0775	**0.2875**	**0.94**	0.352
$\|A\| = 3$	c101-c109	0.275	**0.33**	**0.275**	0.33	**0.38**	0.285	0.231	**0.2725**	0.123	**0.2725**	0.1247	**0.2425**	**0.695**	0.3245
	c201-c208	**0.92**	0.04	**0.835**	0.01	**0.955**	0.127	0.08	**0.415**	**0.0975**	0.0675	**0.1425**	0.065	**0.915**	0.035
	pr1-pr10	**0.77**	0.005	**0.715**	0.236	**0.815**	0.0364	0.1675	**0.4075**	0.035	**0.0425**	**0.165**	0.0225	**0.815**	0.1275
	pr11-pr20	**0.855**	0.045	**0.61**	0.124	**0.865**	0.1647	0.2975	**0.355**	**0.2125**	0.01	**0.2725**	0.0075	**0.775**	0.015
	r101-r112	**0.765**	0.01	**0.875**	0.02	**0.92**	0.01	0.205	**0.2725**	**0.1875**	0.0225	**0.255**	0.005	**0.935**	0.015
	r201-r211	**0.8**	0.01	**0.82**	0.015	**0.75**	0.214	0.335	**0.355**	**0.18**	0.0125	**0.2825**	0.015	**0.755**	0.245
	rc101-rc108	**0.88**	0.224	**0.89**	0.245	**0.92**	0.1347	0.1825	**0.435**	**0.1225**	0.095	**0.15**	0.08	**0.915**	0.1458
	rc201-rc208	**0.735**	0.08	**0.755**	0.075	**0.87**	0.035	0.025	**0.46**	0.055	**0.23**	0.1475	**0.2075**	**0.94**	0.035
$\|A\| = 4$	c101-c109	**0.545**	0.4842	**0.725**	0.231	**0.625**	0.324	0.178	**0.295**	0.095	**0.0125**	0.08	**0.005**	**0.56**	0.041
	c201-c208	**0.565**	0.1231	**0.61**	0.1283	**0.45**	0.124	0.04	**0.2375**	**0.065**	0.0025	0.015	**0.0325**	**0.63**	0.075
	pr1-pr10	**0.63**	0.565	**0.675**	0.086	**0.59**	0.034	**0.0525**	0.015	**0.0175**	0.015	**0.0225**	0.19	**0.575**	0.135
	pr11-pr20	**0.525**	0.151	**0.565**	0.187	**0.435**	0.845	**0.3**	0.045	**0.0125**	0.015	**0.14**	0.045	**0.595**	0.35
	r101-r112	**0.535**	0.315	**0.575**	0.258	**0.535**	0.4567	**0.06**	0.04	**0.055**	0.054	0.0525	**0.065**	**0.58**	0.17
	r201-r211	**0.68**	0.1258	**0.735**	0.4521	**0.56**	0.247	**0.185**	0.05	**0.1875**	0.005	**0.1875**	0.0975	**0.57**	0.025
	rc101-rc108	**0.685**	0.2354	**0.745**	0.584	**0.64**	0.098	**0.095**	0.036	**0.0775**	0.052	**0.2325**	0.0425	**0.645**	0.21
	rc201-rc208	**0.565**	0.368	**0.71**	0.1345	**0.63**	0.1285	**0.0525**	0.02	**0.025**	0.0025	**0.1525**	0.12	**0.59**	0.32
		23/1		23/1		24/0		10/14		17/7		15/9		24/0	

MOHH-TOPTWR = multi-objective and evolutionary hyper-heuristic algorithm for TOPTW based on the humanoid robot's characteristics in the rescue applications; NSGA = non-dominated sorting genetic algorithm; MOEA/D = MO evolutionary metaheuristics algorithm based on decomposition; MOLS = Multi-objective local search; SAILS = simulated annealing iterated local search heuristic; MSA = multi-start simulated annealing.

Table 10 Average values of C-Metric; MOHH-TOPTWR-NSGA-III denoted as MTN and MOHH-TOPTWR-MOEA/D denoted as MTM on the Solomon and Cordeau instances benchmark instances, A1: MTN, A2: MTM, B_1: SAILS, B_2: MSA, B_3: MOLS (B_1, B_2, B_3 are with NSGA-III and detailed results for $\sum t' \in T^{o^a_{t,t'}+z^a_{t'}} = x^a_t$)

		C metric											
		A1 : MTN, B_1 : SAILS		**A_1 : MTN, B_2 : MSA**		**A_1 : MTN, B_3 : MOLS**		**A_2 : MTM, B_1 : SAILS**		**A_2 : MTM, B_2 : MSA**		**A_2 : MTM, B_3 : MOLS**	
	Instances	**C (A_1,B_1)**	**C (B_1,A_1)**	**C (A_1,B_2)**	**C (B_2,A_1)**	**C (A_1,B_3)**	**C (B_3,A_1)**	**C (A_2,B_1)**	**C (B_1,A_2)**	**C (A_2,B_2)**	**C (B_2,A_2)**	**C (A_2,B_3)**	**C (B_3,A_2)**
$\|A\| = 2$	c101-c109	**0.44**	0.03	**0.315**	0.305	**0.7**	0.245	**0.16**	0.015	0.01	**0.1725**	**0.305**	0.2225
	c201-c208	**0.68**	0.354	**0.88**	0.254	**0.79**	0.128	**0.0025**	0.005	**0.005**	0.12	**0.015**	0.021
	pr1-pr10	**0.78**	0.462	**0.81**	0.102	**0.78**	0.458	**0.05**	0.0125	0.0025	**0.1175**	0.04	**0.005**
	pr11-pr20	**0.805**	0.6325	**0.85**	0.2413	**0.79**	0.4541	**0.005**	0.2	0.0925	**0.1475**	0.01	**0.16**
	r101-r112	**0.735**	0.3256	**0.77**	0.11	**0.775**	0.3145	0.0025	**0.1525**	0.035	**0.065**	0.0075	**0.2875**
	r201-r211	**0.795**	0.325	**0.75**	0.251	**0.78**	0.561	0.135	**0.225**	0.0195	**0.04**	0.025	**0.1675**
	rc101-rc108	**0.795**	0.154	**0.835**	0.2144	**0.815**	0.2154	**0.2025**	0.1325	0.0231	**0.0475**	0.2025	**0.205**
	rc201-rc208	**0.775**	0.614	**0.71**	0.42	**0.745**	0.1256	0.0375	**0.295**	0.0275	**0.0575**	**0.1975**	0.125
$\|A\| = 3$	c101-c109	**0.67**	0.005	**0.315**	0.305	**0.54**	0.145	**0.065**	0.0175	0.061	**0.1725**	0.032	**0.1225**
	c201-c208	**0.79**	0.564	**0.88**	0.321	**0.81**	0.5478	0.06	**0.2175**	0.003	0.0128	0.001	**0.005**
	pr1-pr10	**0.72**	0.641	**0.81**	0.5827	**0.82**	0.3246	0.0725	**0.18**	0.0025	**0.1175**	0.0025	**0.12**
	pr11-pr20	**0.815**	0.3554	**0.85**	0.3452	**0.875**	0.4123	0.025	**0.2875**	0.0815	**0.1475**	0.0225	**0.16**
	r101-r112	**0.79**	0.026	**0.77**	0.321	**0.79**	0.156	**0.1425**	0.115	0.035	**0.065**	0.0525	**0.1**
	r201-r211	**0.74**	0.642	**0.75**	0.541	**0.85**	0.245	0.0775	**0.305**	0.007	**0.04**	0.0475	**0.0775**
	rc101-rc108	**0.855**	0.645	**0.835**	0.356	**0.835**	0.523	0.17	**0.3**	0.031	**0.0475**	0.06	**0.0425**
	rc201-rc208	**0.72**	0.425	**0.71**	0.624	**0.735**	0.451	0.0475	**0.385**	0.0275	**0.0575**	0.0375	**0.1125**
$\|A\| = 4$	c101-c109	**0.62**	0.22	**0.725**	0.621	**0.695**	0.0213	0.0875	**0.24**	0.13	**0.0125**	**0.02**	0.0011
	c201-c208	**0.535**	0.215	**0.61**	0.3255	**0.61**	0.263	0.1475	**0.2425**	**0.065**	0.0025	**0.08**	0.005
	pr1-pr10	**0.545**	0.2156	**0.675**	0.1633	**0.675**	0.151	0.0975	**0.17**	**0.0175**	0.015	**0.025**	0.0075
	pr11-pr20	**0.485**	0.251	**0.565**	0.0699	**0.555**	0.0151	0.08	**0.1525**	0.0125	**0.015**	0.01	**0.03**
	r101-r112	**0.54**	0.354	**0.575**	0.214	0.575	**0.598**	0.065	**0.16**	**0.055**	0.02	**0.0175**	0.006
	r201-r211	**0.585**	0.564	**0.735**	0.1549	**0.725**	0.5012	**0.1575**	0.1175	**0.1875**	0.005	**0.07**	0.005
	rc101-rc108	**0.655**	0.318	**0.745**	0.631	**0.735**	0.351	0.0875	**0.1975**	**0.0775**	0.014	**0.16**	0.005
	rc201-rc208	**0.6**	0.251	**0.71**	0.245	**0.705**	0.525	0.09	**0.305**	**0.025**	0.0025	**0.0225**	0.0075
		24/0		24/0		23/1		8/16		7/16		10/14	

MOHH-TOPTWR = multi-objective and evolutionary hyper-heuristic algorithm for TOPTW based on the humanoid robot's characteristics in the rescue applications; NSGA = non-dominated sorting genetic algorithm; MOEA/D = MO evolutionary metaheuristics algorithm based on decomposition; MOLS = Multi-objective local search; SAILS = simulated annealing iterated local search heuristic; MSA = multi-start simulated annealing.

5 Conclusion

This paper has introduced an MO variant of TOPTW for rescue applications (TOPTWR) based on the humanoid robots' characteristics. In this regard, we considered heterogeneous tasks and robots with different energy consumptions per task. Also, we introduced five objectives in TOPTWR to give a user a chance to select a desired suitable solution based on the rescue situation. To achieve this, hyper-heuristic algorithms were designed for TOPTWR and named MOHH-TOPTWR. The MOHH-TOPTWR automatically selects LLHs (a combination of the state-of-the-art operators) during the MO evolutionary optimization. MOHH-TOPTWR with the hyper-heuristic algorithm in this paper reduces algorithm choice challenges and parameter tuning efforts on TOPTWR dramatically for the first time. We implemented MOHH-TOPTWR with two metaheuristics as NSGA-III and MOEA/D. Extensive experiments showed the effectiveness of the proposed algorithms in comparison to the state-of-the-art algorithms. The proposed algorithms can be seen as benchmark algorithms for MO-TOPTWR, which can be used for comparison by the future research.

In the future work, a different selection strategy can be implemented and evaluated. Moreover, other different MOEAs and hyper-heuristic techniques can be tested and compared on the MO-TOPTWR.

Acknowledgments

This work was financially supported by the 'Chinese Language and Technology Center' of National Taiwan Normal University (NTNU) from The Featured Areas Research Center Program within the framework of the Higher Education Sprout Project by the Ministry of Education (MOE) in Taiwan, and Ministry of Science and Technology, Taiwan, under Grant Nos. MOST 108-2634-F-003-002, MOST 108-2634-F-003-003, and MOST 108-2634-F-003-004 (administered through Pervasive Artificial Intelligence Research (PAIR) Labs) as well as MOST 107-2811-E-003-503. We are grateful to the National Center for High-performance Computing for computer time and facilities to conduct this research.

References

Abbaszadeh, M. & Saeedvand, S. 2014. A fast genetic algorithm for solving university scheduling problem. *IAES International Journal of Artificial Intelligence* **3**, 7.

Abbaszadeh, M., Saeedvand, S. & Mayani, H. A. 2012. Solving university scheduling problem with a memetic algorithm. *IAES International Journal of Artificial Intelligence* **1**, 79.

Alkhanak, E. N. & Lee, S. P. 2018. A hyper-heuristic cost optimisation approach for scientific workflow scheduling in cloud computing. *Future Generation Computer Systems* **86**, 480–506.

Auer, P., Cesa-Bianchi, N. & Fischer, P. 2002. Finite-time analysis of the multiarmed bandit problem. *Machine Learning* **47**, 235–256.

Baltes, J., Tu, K.-Y., Sadeghnejad, S. & Anderson, J. 2017. HuroCup: Competition for multi-event humanoid robot athletes. *The Knowledge Engineering Review* **32**, 1–14.

Bederina, H. & Hifi, M. 2017. A hybrid multi-objective evolutionary algorithm for the team orienteering problem. In *2017 4th International Conference on Control, Decision and Information Technologies* (CoDIT), 0898–0903. IEEE.

Bottarelli, L., Bicego, M., Blum, J. & Farinelli, A. 2019. Orienteering-based informative path planning for environmental monitoring. *Engineering Applications of Artificial Intelligence* **77**, 46–58.

Burke, E. K., Gendreau, M., Hyde, M., Kendall, G., Ochoa, G., Özcan, E. & Qu, R. 2013. Hyper-heuristics: a survey of the state of the art. *Journal of the Operational Research Society* **64**, 1695–1724.

Burke, E. K., Hyde, M., Kendall, G., Ochoa, G., Özcan, E. & Woodward, J. R. 2010. A classification of hyper-heuristic approaches. In *Handbook of Metaheuristics*, Gendreau, M. & Potvin, JY. (eds). Springer.

Campbell, A. M., Gendreau, M. & Thomas, B. W. 2011. The orienteering problem with stochastic travel and service times. *Annals of Operations research* **186**, 61–81.

Chakhlevitch, K. & Cowling, P. 2008. Hyperheuristics: Recent developments. In *Adaptive and Multilevel Metaheuristics*, Cotta, C., Sevaux, M. & Sörensen, K. (eds). Springer.

Chang, C.-H., Wang, S.-C. & Wang, C.-C. 2016. Exploiting moving objects: multi-robot simultaneous localization and tracking. *IEEE Transactions on Automation Science and Engineering* **13**, 810–827.

Coello, C. A. C., Lamont, G. B. & Van Veldhuizen, D. A. 2007. *Evolutionary Algorithms for Solving Multi-Objective Problems*. Springer.

Cordeau, J.-F., Gendreau, M. & Laporte, G. 1997. A tabu search heuristic for periodic and multi-depot vehicle routing problems. *Networks: An International Journal* **30**, 105–119.

Cura, T. 2014. An artificial bee colony algorithm approach for the team orienteering problem with time windows. *Computers & Industrial Engineering* **74**, 270–290.

Deb, K. & Jain, H. 2014. An evolutionary many-objective optimization algorithm using reference-point-based non-dominated sorting approach, part I: Solving problems with box constraints. *IEEE Transactions on Evolutionary Computation* **18**, 577–601.

Deb, K., Pratap, A., Agarwal, S. & Meyarivan, T. A. M. T. 2002. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation* **6**, 182–197.

DeDonato, M., Dimitrov, V., Du, R., Giovacchini, R., Knoedler, K., Long, X., Polido, F., Gennert, M. A., Padır, T. & Feng, S. 2015. Human-in-the-loop control of a humanoid robot for disaster response: a report from the DARPA robotics challenge trials. *Journal of Field Robotics* **32**, 275–292.

Diftler, M. A., Culbert, C. J., Ambrose, R. O., Platt, R. & Bluethmann, W. J. 2003. Evolution of the NASA/DARPA robonaut control system. In '2003 Proceedings of IEEE International Conference on Robotics and Automation ICRA'03, 2543–2548. IEEE.

Dong, N. & Dai, C. 2018. An improvement decomposition-based multi-objective evolutionary algorithm using multi-search strategy. *Knowledge-Based Systems* **163**, 572–580.

Duchoň, F., Babinec, A., Kajan, M., Beňo, P., Florek, M., Fico, T. & Jurišica, L. 2014. Path planning with modified a star algorithm for a mobile robot. *Procedia Engineering* **96**, 59–69.

Farinelli, A., Zanotto, E. & Pagello, E. 2017. Advanced approaches for multi-robot coordination in logistic scenarios. *Robotics and Autonomous Systems* **90**, 34–44.

Feng, S., Whitman, E., Xinjilefu, X. & Atkeson, C. G. 2015. Optimization-based full body control for the DARPA robotics challenge. *Journal of Field Robotics* **32**, 293–312.

Fialho, Á., Da Costa, L., Schoenauer, M. & Sebag, M. 2010. Analyzing bandit-based adaptive operator selection mechanisms. *Annals of Mathematics and Artificial Intelligence* **60**, 25–64.

Goldberg, D. E. 1989 *Genetic Algorithms in Search, Optimization, and Machine Learning, Addison-Wesley, Reading, Ma, 1989*. Addison-Wesley Longman Publishing.

Golden, B. L., Levy, L. & Vohra, R. 1987. The orienteering problem. *Naval Research Logistics (NRL)* **34**, 307–318.

Guizzo, G., Vergilio, S. R., Pozo, A. T. & Fritsche, G. M. 2017. A multi-objective and evolutionary hyper-heuristic applied to the integration and test order problem. *Applied Soft Computing* **56**, 331–344.

Gunawan, A., Lau, H. C. & Lu, K. 2018. ADOPT: combining parameter tuning and adaptive operator ordering for solving a class of orienteering problems. *Computers & Industrial Engineering* **121**, 82–96.

Gunawan, A., Lau, H. C. & Vansteenwegen, P. 2016. Orienteering problem: a survey of recent variants, solution approaches and applications. *European Journal of Operational Research* **255**, 315–332.

Gunawan, A., Lau, H. C., Vansteenwegen, P. & Lu, K. 2017. Well-tuned algorithms for the team orienteering problem with time windows. *Journal of the Operational Research Society* **68**, 861–876.

Gunn, T. & Anderson J. 2015. Dynamic heterogeneous team formation for robotic urban search and rescue. *Journal of Computer and System Sciences* **81**, 553–567.

Hu, Q. & Lim, A. 2014. An iterative three-component heuristic for the team orienteering problem with time windows. *European Journal of Operational Research* **232**, 276–286.

Huang, L., Ding, Y. & Jin, Y. 2018. Multiple-solution optimization strategy for multi-robot task allocation. *IEEE Transactions on Systems, Man and Cybernetics: Systems*.

Jiang, Y. 2016. A survey of task allocation and load balancing in distributed systems. *IEEE Transactions on Parallel and Distributed Systems* **27**, 585–599.

Jin, M., Lee, J. & Tsagarakis, N. G. 2017. Model-free robust adaptive control of humanoid robots with flexible joints. *IEEE Transactions on Industrial Electronics* **64**, 1706–1715.

Jose, K. & Pratihar, D. K. 2016. Task allocation and collision-free path planning of centralized multi-robots system for industrial plant inspection using heuristic methods. *Robotics and Autonomous Systems* **80**, 34–42.

Kaneko, K., Morisawa, M., Kajita, S., Nakaoka, S. I., Sakaguchi, T., Cisneros, R. & Kanehiro, F. 2015. Humanoid robot HRP-2Kai—Improvement of HRP-2 towards disaster response tasks. In '2015 IEEE-RAS 15th International Conference on Humanoid Robots (Humanoids), 132–139. IEEE.

Karakatič, S. & Podgorelec, V. 2015. A survey of genetic algorithms for solving multi depot vehicle routing problem. *Applied Soft Computing* **27**, 519–532.

Khamis, A., Hussein, A. & Elmogy, A. 2015. Multi-robot task allocation: a review of the state-of-the-art. In *Cooperative Robots and Sensor Networks*. Springer.

Kohlbrecher, S., Romay, A., Stumpf, A., Gupta, A., Von Stryk, O., Bacim, F., Bowman, D. A., Goins, A., Balasubramanian, R. & Conner, D. C. 2015. Human-robot teaming for rescue missions: team ViGIR's approach to the 2013 DARPA robotics challenge trials. *Journal of Field Robotics* **32**, 352–377.

Koubaa, A., Bennaceur, H., Chaari, I., Trigui, S., Ammar, A., Sriti, M. F., Alajlan, M., Cheikhrouhou, O. & Javed, Y. 2018. Different approaches to solve the MRTA problem. In *Robot Path Planning and Cooperation*, Kacprzyk, J. (ed.). Springer.

Kube, C. R. & Bonabeau, E. 2000. Cooperative transport by ants and robots. *Robotics and Autonomous Systems* **30**, 85–101.

Labadie, N., Mansini, R., Melechovský, J. & Calvo, R. W. 2012. The team orienteering problem with time windows: an lp-based granular variable neighborhood search. *European Journal of Operational Research* **220**, 15–27.

Lin, S.-W. & Vincent F. Y. 2012. A simulated annealing heuristic for the team orienteering problem with time windows. *European Journal of Operational Research* **217**, 94–107.

Lin, S.-W. & Vincent, F. Y. 2017. Solving the team orienteering problem with time windows and mandatory visits by multi-start simulated annealing. *Computers & Industrial Engineering* **114**, 195–205.

Mahajan, A. & Teneketzis, D. 2008. Multi-armed bandit problems. In *Foundations and Applications of Sensor Management*, Hero, AO., Castañón, D., Cochran, D. & Kastella, K. (eds). Springer.

Martín-Moreno, R. & Vega-Rodríguez, M. A. 2018. Multi-objective artificial bee colony algorithm applied to the bi-objective orienteering problem. *Knowledge-Based Systems* **154**, 93–101.

Michalewicz, Z. 2013. *Genetic Algorithms+ Data Structures= Evolution Programs*. Springer Science & Business Media.

Montemanni, R. & Gambardella L. M. 2009. An ant colony system for team orienteering problems with time windows. *Foundation Of Computing And Decision Sciences* **34**, 287.

Nunes, E., Manner, M., Mitiche, H. & Gini, M. 2017. A taxonomy for task allocation problems with temporal and ordering constraints. *Robotics and Autonomous Systems* **90**, 55–70.

Park, J., Lee, J., Ahn, S., Bae, J. & Tae, H. 2017. Exact algorithm for the capacitated team orienteering problem with time windows. *Mathematical Problems in Engineering* **2017**, 1191–1203.

Righini, G. & Salani, M. 2009. Decremental state space relaxation strategies and initialization heuristics for solving the orienteering problem with time windows with dynamic programming. *Computers & Operations Research* **36**, 1191–1203.

Saeedvand, S. & Aghdasi, H. S. 2016. An energy efficient metaheuristic method for micro robots indoor area coverage problem. In '2016 6th International Conference on Computer and Knowledge Engineering (ICCKE), 88–93. IEEE.

Saeedvand, S., Aghdasi, H. S. & Baltes, J. 2018. Novel lightweight odometric learning method for humanoid robot localization. *Mechatronics* **55**, 38–53.

Saeedvand, S., Aghdasi, H. S. & Baltes, J. 2019. Robust multi-objective multi-humanoid robots task allocation based on novel hybrid metaheuristic algorithm. *Applied Intelligence* **49**, 4097–4127.

Savelsbergh, M. W. P. 1985. Local search in routing problems with time windows. *Annals of Operations Research* **4**, 285–305.

Schilde, M., Doerner, K. F., Hartl, R. F. & Kiechle, G. 2009. Metaheuristics for the bi-objective orienteering problem. *Swarm Intelligence* **3**, 179–201.

Schwarzrock, J., Zacarias, I., Bazzan, A. L., de Araujo Fernandes, R. Q., Moreira, L. H. & de Freitas, E. P. 2018. Solving task allocation problem in multi unmanned aerial vehicles systems using swarm intelligence. *Engineering Applications of Artificial Intelligence* **72**, 10–20.

Solomon, M. M. 1986. On the worst-case performance of some heuristics for the vehicle routing and scheduling problem with time window constraints. *Networks* **16**, 161–174.

Solomon, M. M. 1987. Algorithms for the vehicle routing and scheduling problems with time window constraints. *Operations Research* **35**, 254–265.

Souffriau, W., Vansteenwegen, P., Vanden Berghe, G. & Van Oudheusden, D. 2013. The multiconstraint team orienteering problem with multiple time windows. *Transportation Science* **47**, 53–63.

Spenko, M., Buerger, S. & Iagnemma, K. 2018. *The DARPA Robotics Challenge Finals: Humanoid Robots To The Rescue*. Springer.

Su, X., Wang, Y., Jia, X., Guo, L. & Ding, Z. 2018. Two innovative coalition formation models for dynamic task allocation in disaster rescues. *Journal of Systems Science and Systems Engineering* **27**, 215–230.

Tang, H. & Miller-Hooks, E. 2005. A tabu search heuristic for the team orienteering problem. *Computers & Operations Research* **32**, 1379–1407.

Toledo, A. & Riff, M. C. 2015. HOPHS: a hyperheuristic that solves orienteering problem with hotel selection. In '2015 Fifth International Conference on Digital Information Processing and Communications (ICDIPC), 148–152. IEEE.

Tricoire, F., Romauch, M., Doerner, K. F. & Hartl, R. F. 2010. Heuristics for the multi-period orienteering problem with multiple time windows. *Computers & Operations Research* **37**, 351–367.

Tsiligirides, T. 1984. Heuristic methods applied to orienteering. *Journal of the Operational Research Society* **35**, 797–809.

Vansteenwegen, P., Souffriau, W., Berghe, G. V. & Van Oudheusden, D. 2009. Iterated local search for the team orienteering problem with time windows. *Computers & Operations Research* **36**, 3281–3290.

Vansteenwegen, P., Souffriau, W. & Van Oudheusden, D. 2011. The orienteering problem: a survey. *European Journal of Operational Research* **209**, 1–10.

Vincent, F. Y., Jewpanya, P., Ting, C. J. & Redi, A. P. 2017. Two-level particle swarm optimization for the multi-modal team orienteering problem with time windows. *Applied Soft Computing* **61**, 1022–1040.

Wang, J., Zhou, Y., Wang, Y., Zhang, J., Chen, C. P. & Zheng, Z. 2016. Multiobjective vehicle routing problems with simultaneous delivery and pickup and time windows: formulation, instances, and algorithms. *IEEE Transactions on Cybernetics* **46**, 582–594.

Yang, X.-S. 2010. *Nature-Inspired Metaheuristic Algorithms*. Luniver Press.

Yin, P.-Y., Yu, S. S., Wang, P. P. & Wang, Y. T. 2007. Multi-objective task allocation in distributed computing systems by hybrid particle swarm optimization. *Applied Mathematics and Computation* **184**, 407–420.

Zhang, Q. & Li, H. 2007. MOEA/D: a multiobjective evolutionary algorithm based on decomposition. *IEEE Transactions on Evolutionary Computation* **11**, 712–731.

Zhang, Q., Zhou, A. & Jin, Y. 2008. RM-MEDA: a regularity model-based multiobjective estimation of distribution algorithm. *IEEE Transactions on Evolutionary Computation* **12**, 41–63.

Zhang, T. & Ueno, H. 2007. Knowledge model-based heterogeneous multi-robot system implemented by a software platform. *Knowledge-Based Systems* **20**, 310–319.

Zheng, W., Liao, Z. & Qin, J. 2017. Using a four-step heuristic algorithm to design personalized day tour route within a tourist attraction. *Tourism Management* **62**, 335–349.

Zhu, W., Li, L., Teng, L. & Yonglu, W. 2018. Multi-UAV reconnaissance task allocation for heterogeneous targets using an opposition-based genetic algorithm with double-chromosome encoding. *Chinese Journal of Aeronautics* **31**, 339–350.

Zitzler, E. 1999. *Evolutionary Algorithms for Multiobjective Optimization: Methods and Applications*. Citeseer.

Zitzler, E. & Thiele, L. 1999. Multiobjective evolutionary algorithms: a comparative case study and the strength Pareto approach. *IEEE Transactions on Evolutionary Computation* **3**, 257–271.

Zitzler, E., Thiele, L., Laumanns, M., Fonseca, C. M. & Da Fonseca, V. G. 2003. Performance assessment of multiobjective optimizers: an analysis and review. *IEEE Transactions on Evolutionary Computation* **7**, 117–132.