Applied Soft Computing 63 (2018) 249–267

Contents lists available at ScienceDirect

Applied Soft Computing

journal homepage: www.elsevier.com/locate/asoc

A bi-objective load balancing model in a distributed simulation system using NSGA-II and MOPSO approaches

Shuxin Ding[a,b,c], Chen Chen[a,b,*], Bin Xin[a,b], Panos M. Pardalos[c]

[a] School of Automation, Beijing Institute of Technology, Beijing 100081, China
[b] State Key Laboratory of Intelligent Control and Decision of Complex System, Beijing 100081, China
[c] Center for Applied Optimization, Department of Industrial and Systems Engineering, University of Florida, Gainesville, FL 32611, USA

ARTICLE INFO

Article history:
Received 13 April 2017
Received in revised form 6 August 2017
Accepted 5 September 2017

Keywords:
Load balancing
High level architecture
Multi-objective optimization
Non-dominated sorting genetic algorithm with elitist strategy
Multi-objective particle swarm optimization

ABSTRACT

High level architecture (HLA) is a software-architecture specification of a distributed simulation system which does not involve load balancing. As a result, problems of long simulation time and distortion caused by unequally distributed simulation tasks cannot be solved in the simulation process of HLA. In this paper, the simulation system based on the component level will be studied, and problems of load imbalances will be tackled. The goal is to find a set of Pareto optimal solutions to minimize the imbalance of the computation load and the total communication load with load limitation constraints in the model. To formulate this problem, a new integer programming model is presented. Both a non-dominated sorting genetic algorithm with elitist strategy (NSGA-II) and a multi-objective particle swarm optimization (MOPSO) are adopted to solve the problem. Different global best selection methods (crowding distance, adaptive grids and comprehensive ranking) and perturbation methods (rapidly decreasing and elitist learning strategy) for MOPSO are analyzed. Since the parameters of the algorithms have significant effects on their performance, the Taguchi method with a novel response value is utilized to tune the parameters of the proposed algorithms. Five performance metrics are used to evaluate the results of these algorithms. Based on the results, a networked control simulation platform will be used to test and verify the schemes. Numerical results show that the proposed MOPSO with ELS operator outperforms other proposed algorithms in solving the problem. In addition, the proposed strategies could solve problems of load imbalances in systems based on HLA.

1. Introduction

With the advantages of high utilization ratio, high scalability and parallelism, distributed systems can be run in parallel to handle problems that are computationally intensive. In addition, several difficult problems are derived from the distributed structure, such as the communication problems among the nodes, the mapping and assignment problems of system resources and computing nodes.

High level architecture (HLA) is widely used for distributed simulation development. Although HLA supports large-scale distributed simulation, the load balancing mechanism in the simulation process is not standardized [1]. When the simulation system runs under the condition of non-optimized allocation, load imbalance problems will occur due to the shortage of computing resources, which will seriously affect the simulation efficiency and confidence.

Load balancing is a problem related to the resource allocation problem, which is a fundamental combinatorial optimization problem. Many fields are concerned such as portfolio management [2], sensor network management [3], node deployment [4], etc. However, load balancing problem focuses on the equilibrium of load. In the distributed simulation system based on HLA, the load is divided into the computation load and the communication load. The main purpose of load balancing is to minimize the negative effects of the two kinds of loads, that is to say, try to even out the distribution of computation load among the nodes, and minimize the communication load among them. The results of load balancing are shown in the following two aspects: firstly, shortening the average response time of tasks effectively, that is to say, trying to make the runtime of the simulation process as short as possible, so that the system could run steadily and smoothly; secondly, maximizing the use of simulation resources of the whole system, and reducing the unnecessary waste of resources. However, there are conflicts between the dis-

* Corresponding author at: School of Automation, Beijing Institute of Technology, Beijing 100081, China.
E-mail addresses: shxding@bit.edu.cn (S. Ding), xiaofan@bit.edu.cn (C. Chen), brucebin@bit.edu.cn (B. Xin), pardalos@ufl.edu (P.M. Pardalos).

https://doi.org/10.1016/j.asoc.2017.09.012
1568-4946/

tributions of these two kinds of loads, as a result, the load balancing problem is a multi-objective problem.

According to the load distribution mode, methods of load balancing are divided into static task allocation method and dynamic task migration method. Dynamic load balancing is concerned with the migration of the load, that is, the migration of the federates [5]. However, the migration operation will cause many problems, such as transforming the federates developed in traditional methods. Besides, the entire simulation process needs to be paused sometimes, and the majority operations of the migration process are complex, so that the federal implementation will be of low efficiency, which betrays the original intention of load balancing. In addition, the migration operation can't be realized unless operated by the third party software or components. The operation will increase the difficulty of its development and use [5–7]. The static load balancing method mainly studies the allocation algorithm of simulation tasks. It can be directly used without any additional software. Therefore, this paper uses the static load balancing method to optimize the simulation of the distribution simulation system based on HLA.

Static load balancing method is a kind of off-line optimization method, and its goal is to obtain an optimal allocation scheme, which can be applied to assigning tasks to different simulation nodes before running the simulation system. In the research, heuristic algorithms, such as genetic algorithm, simulated annealing algorithm, and hybrid algorithms, are commonly used to obtain the task allocation scheme. The optimum solution obtained can assign simulation tasks to different nodes.

Wu et al. [8] proposed a heuristic algorithm to schedule the simulation tasks. Martino et al. [9] used an improved genetic algorithm to realize static allocation of simulation tasks in the grid environment. According to the static scheduling problem of grid simulation, Wei et al. [10] proposed an entity scheduling strategy based on the interaction priority algorithm. This strategy can transform the multi-objective problem into a single objective optimization problem by using the weighted method, and divided the scheduling strategy into two periods. Zhao et al. [11] used an improved simulated annealing algorithm to optimize the mapping of the federates to the computing nodes. Pan et al. [12] focused on the static load balancing of the multi-level distributed real-time simulation system, and they took the real-time performance of the simulation tasks into consideration, and set up a heuristic function to realize the task map. Ban et al. [13] designed a sub-phase load balancing method based on the federal level of HLA simulation system. Yue et al. [14] studied the static load balancing and took the dynamic process into consideration at the same time. They took the total communication and computation load as well as the simulation time as two objectives. They used the Monte Carlo method and sequential optimization method to obtain the optimal allocation scheme.

According to the analysis above, the static load balancing research based on the HLA simulation system mostly focuses on the single objective optimization of the federate. However, the computation load and the communication load are two different kinds of loads, and there is no reference value to measure them. As a result, there is a lack of accurate basis for their weight, the common basis is subjective and unreasonable so that ideal task allocation schemes are extremely hard to obtain. On the basis of module design of the federates in the simulation system, this paper will consider a bi-objective model to describe the load balancing problem.

In order to solve the bi-objective load balancing model, multi-objective evolutionary algorithms (MOEAs) are usually adopted. In recent years, a non-dominated sorting genetic algorithm with elitist strategy (NSGA-II) [15], and a multi-objective particle swarm optimization (MOPSO) [16], etc. are widely used in many fields. Many of recent works on bi-objective or multi-objective optimization problems [17–22] are analyzed by comparisons of these algorithms. Salimi et al. [23] solves a task scheduling problem with load balancing by NSGA-II with fuzzy operators for computational grids. In [24], a bi-objective generalized assignment problem (GAP) with equilibrium function is solved using an integer enhanced version of NSGA-II. Dahmani et al. [25] use a discrete MOPSO to solve a bi-objective load balancing problem in aircraft cargo transportation. Total weight and the total priority of loaded cargo are maximized with load balancing constraints. In [26], a multi-objective load balancing system that avoids virtual machine (VM) migration and achieves system load balancing is developed and particle swarm optimization algorithm is applied. Alkayal et al. [27] uses a multi-objective particle swarm optimization algorithm with ranking strategy to solve the task scheduling problem in cloud computing. Cloud computing is a parallel and distributed system with virtual machines. Computational resources are allocated to multiple independent execution environments. Zhu et al. [28] use a multi-objective ant colony optimization algorithm based on load balance for virtual machine placement. The reviewed literature related to this study is summarized in Table 1.

Here are the main contributions of this paper. First, unlike many researches in which a single objective load balancing model is considered. In this paper, a novel bi-objective load balancing model with load limitation is presented. The first objective function aims to minimize the imbalances of the computation load and the second objective function, minimize the total communication load. Second, in this paper, several MOEAs based on NSGA-II [15] and MOPSO are adopted to solve the bi-objective load balancing model, and the idea of dynamic equilibrium will also be applied to these algorithm. We consider some improved strategies in MOPSO, they are global best selection methods (crowding distance [29], adaptive grids [16] and comprehensive ranking [30]) and perturbation methods (rapidly decreasing [16] and elitist learning strategy [31]). Crowding distance and adaptive grids are two popular global best selection methods. Besides, this paper also considers comprehensive ranking. Since the meta-heuristic algorithms are sensitive to the parameters, a Taguchi method is conducted to calibrate the parameters of the algorithms with a novel metric which uses a combination of convergence and diversity as the response value. Finally, several performance metrics are used to evaluate the efficiency of the proposed algorithms. The ideal allocation scheme will be verified on a networked control simulation platform.

The remainder of this paper is organized as follows: In Section 2, the problem is formulated. In Section 3, solution algorithms are presented. Simulation experiment results are provided in Section 4. In Section 5, the schemes verification is given. Conclusions and future works are presented in Section 6.

2. Problem formulation

In this paper, a bi-objective load balancing problem with computation and communication load limitation is considered. As stated before, this problem has been inspired from the load distribution problem in distributed simulation system, i.e. HLA. In the distributed simulation system based on HLA, simulation tasks can be divided into federate level and component level. The location of the federates remain unchanged during the simulation process. The load related to the simulation tasks is divided into computation load and communication load. Computation load refers to the system overhead of each simulation task (federate/component) that can be executed in parallel [12]. In this system, the computation load is mainly derived from the internal calculation of the simulation model and propulsion of the simulation process. Communication load refers to the amount of data exchange between nodes in the unit simulation period. HLA provides the distributed simulation

Table 1
Summary of reviewed literature related to load balancing.

Problem	Method	Reference
Single-objective optimization		
Assigning and scheduling parallel executable tasks to processor networks	Heuristic algorithm	[8]
Static scheduling problem of grid simulation	Improved genetic algorithm	[9]
Static tasks scheduling for simulation over the grids	Interaction priority algorithm based scheduling strategy	[10]
Mapping federates to computing nodes	Improved simulated annealing	[11]
Static load balancing of the multi-level distributed real-time simulation system	Heuristic algorithm	[12]
Static load balancing optimization of HLA distributed simulation	A two-stage mapping allocation optimization method	[13]
Load balancing problem of HLA collaborative simulation	Monte Carlo based optimization algorithm	[14]
Multi-objective optimization		
Load balancing problem in the task scheduling	NGSA-II using fuzzy operators	[23]
Generalized Assignment Problem	Integer NSGA-II	[24]
Two level load balancing problem	Discrete multi-objective particle swarm optimization	[25]
Multi-objective load balancing (MO-LB) system	Particle swarm optimization	[26]
Task scheduling in cloud computing	Multi-objective particle swarm optimization	[27]
Load balance for the VM placement problem	Multi-objective ant colony optimization algorithm	[28]

system with data distribution management mechanism (DDM), and data is transmitted in the form of objective class attribution value or the interactive class parameter value [32,33]. Therefore, the communication load is mainly derived from the update/reflection operation of objective class and the sending/receiving operation of interactive class. In the simulation system, a computer in the Local Area Network (LAN) is a simulation node. The factors related to the simulation nodes are the computing capacity of each simulation node and the communication bandwidth between two simulation nodes. The computing capacity of the simulation node is determined by many factors of the simulation nodes [34,35]: dominant frequency of CPU, memory capacity, etc. These two compacities consist of the constraints in the model. In order to utilize the simulation resources of the system, the total computation load and communication load in the simulation nodes should be minimized. However, it is assumed that the computation load of the simulation tasks stays consistent in different simulation nodes, while there is no communication load with two simulation tasks when assigned to the same simulation node. Therefore, minimizing both the variance of the computation load in the nodes and total communication load between simulation nodes are considered in this model.

2.1. Assumptions

- Each simulation task can be assigned to only one simulation node.
- Each simulation node has its own load limit, and the communication line has bandwidth limit.
- When two simulation tasks are assigned to the same simulation node, there is no communication load between these two simulation tasks.
- The computation load of the same tasks stays consistent in different simulation nodes.
- The communication load between two tasks stays consistent in different simulation nodes, as well as in both directions.

2.2. Indices

$i, j = 1, 2, \ldots, N$ Indices for simulation tasks
$k, l = 1, 2, \ldots, M$ Indices for simulation nodes

2.3. Parameters

N: Number of simulation tasks
M: Number of simulation nodes
L: Maximum computation load in a simulation node
B: Maximum communication load between two simulation nodes
pt_i: Computation load of simulation task i
mt_{ij}: Communication load between simulation task i and simulation task j

2.4. Decision variables

x_{ik} 1, if simulation task i is assigned to simulation node k; 0, otherwise

2.5. The mathematical model

$$\min \quad Z_1 = \frac{\sum_{k=1}^{M}\left(\sum_{i=1}^{N} x_{ik} pt_i - \frac{\sum_{k=1}^{M}\sum_{i=1}^{N} x_{ik} pt_i}{M}\right)^2}{(M-1)} \tag{1}$$

$$\min \quad Z_2 = \sum_{k=1}^{M}\sum_{l=1, l\neq k}^{M}\sum_{i=1}^{N}\sum_{j=1}^{N} x_{ik} x_{jl} mt_{ij} \tag{2}$$

$$s.t. \quad \sum_{k=1}^{M} x_{ik} = 1, \quad i = 1, \ldots, N \tag{3}$$

$$\sum_{i=1}^{N} x_{ik} pt_i \leq L, \quad k = 1, \ldots, M \tag{4}$$

$$\sum_{i=1}^{N}\sum_{j=1}^{N} x_{ik} x_{jl} mt_{ij} \leq B, \quad k = 1, \ldots, M, \quad l = 1, \ldots, M \tag{5}$$

$$x_{ik} \in \{0, 1\} \quad i = 1, \ldots, N, \quad k = 1, \ldots, M \tag{6}$$

Eq. (1) minimizes the variance of the computation load of the simulation nodes. Eq. (2) minimizes the total communication load between simulation nodes. Eq. (3) imposes that each simulation task is assigned exactly to one simulation node. Eq. (4) assures the computation load limitation. Eq. (5) guarantees the communication load limitation. Eq. (6) denotes the non-negativity and integrality of the variables.

Therefore, this problem is modeled as a constrained bi-objective optimization problem. It is a common problem in the real industry. In [14], an engineering experiment about collaborative simulation under HLA/RTI of high speed electricity multiple units(EMU) has been established. A good load balancing scheme shows great significance in enhancing the system.

The problem is a multi-objective optimization problem. Its solution is not a single scheme, but a set of allocation schemes which is called Pareto optimal set. With the increase of simulation tasks and simulation nodes, the number of task allocation schemes of the distributed system has shown explosive growth. It has been

proved that load balancing problem is an NPC problem [35]. Model of this problem is so complex that even the constraints have been simplified, optimal set is still difficult to find. Therefore, intelligent optimization algorithms are adopted to obtain the best solutions (set). Therefore, the aim of this paper is to apply intelligent optimization algorithms to find a set of Pareto optimal solutions for load balancing problems.

2.6. Dynamic involvement of simulation tasks

The static load balancing algorithm focuses on the load distribution strategies of federates and components before simulation. During the process of simulation, additional federates or components will be added to complete specific simulation tasks. As in shown in Fig. 1, when the simulation process is advanced to step k, a certain number of federates or components are required to be added. At this moment, the running tasks of each simulation node are different, and these tasks are fixed and irreplaceable. As a result, such load balancing problems are only related to operations of adding federates or components, rather than migration or departure operations.

The key of solving this kind of dynamic load balancing problems is to transfer dynamic states to static states, which means converting the problems into static load balancing problems. Then, according to the proposed load balancing algorithm, the optimal task allocation scheme of step k can be obtained. And the federates/components which are to be added should be arranged to their corresponding simulation nodes. When the system is running on step k, they will be added to federal executive. This problem is essentially the same with static load balancing problem, except that the remaining available computing resources in each simulation nodes are different in step k, so the constraints of load distribution are transferred from initial fixed constraints into available computing resources of simulation nodes. Therefore, this kind of problems can still be solved by MOEAs.

3. Solution algorithms

To deal with the proposed bi-objective load balancing problem, two kinds of MOEAs are reviewed and analyzed in this section. They are NSGA-II and MOPSO. They are employed to find the Pareto solutions. For a bi-objective problem to be optimized, f_1 and f_2 are the two objective functions. We say x_1 dominate x_2 if the following two conditions satisfied:

1. $f_1(x_1) \leq f_1(x_2)$ and $f_2(x_1) \leq f_2(x_2)$
2. $f_1(x_1) < f_1(x_2)$ or $f_2(x_1) < f_2(x_2)$

The solutions that cannot dominate each other are called Pareto front (PF).

3.1. NSGA-II

NSGA-II is one of the most popular multi-objective evolutionary algorithms. This algorithm has been applied in many engineering applications. The main feature of this algorithm is the elitist non-dominated sorting method. Crowding distance is used as a ranking criterion when sorting the non-dominated solutions. The pseudo-code of NSGA-II is shown in Fig. 2.

The chromosomes in the population are represents as possible solutions for the problem. The crowding distance is used to evaluate solutions with the same non-dominated rank by measuring the relative density of the individuals. As a result, we have the rank of all the individuals in the population. Elitism is guaranteed by combining parent population with offspring population and selected by the

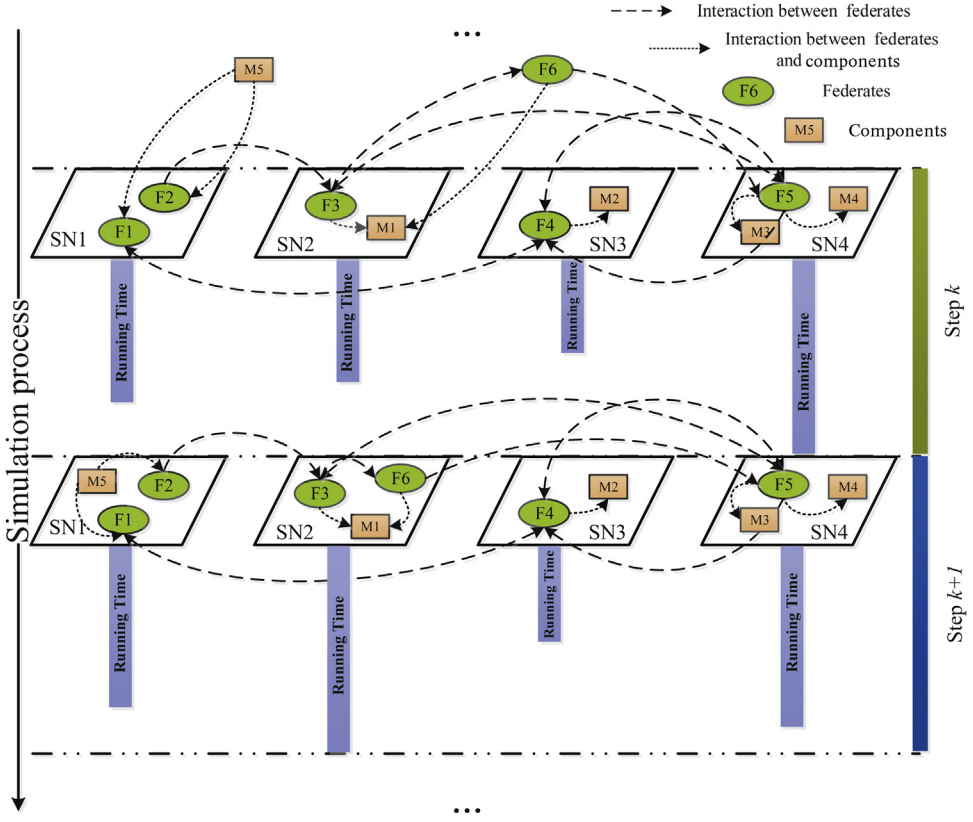


Fig. 1. Schematic diagram of dynamic involvement of federates/components.

Initialize population

Do non-dominated sorting with crowding distance to rank the individuals

$Iter = 0$

while $Iter < MaxIter$ **do**

Get offspring population from tournament selection, crossover and mutation

Merge parent population and offspring population

Do non-dominated sorting with crowding distance to rank the individuals

Select individuals by the order within the population size

$Iter = Iter + 1$

end while

Report result in the population

Fig. 2. The pseudo-code of NSGA-II.

crowding distance based non-dominated sorting. The pseudo-code of calculating crowding distance is shown in Fig. 3.

By adopting crowding distance, the individual i in the population can be easily sorted with the crowded operator [15] with two attributes: non-dominated rank (i_{rank}) and crowding distance ($i_{distance}$). The order of two individuals i and j can be as follows:

1. $i \prec j$ when ($i_{rank} < j_{rank}$) or ($i_{rank} = j_{rank}$ and $i_{distance} > j_{distance}$).

3.2. MOPSO

Here we introduce how PSO is applied to solve the multi-objective optimization problems. The PSO algorithm was first developed by Kennedy [36] in 1995. In PSO, a swarm of particles are regarded as potential solutions. Each particle has two parts: position and velocity. Different from genetic algorithm, the particles are updated through velocity vector. There are two leaders: per-

n: size of the non-dominated solutions in S

m: number of the objectives

$CD[i]$: the crowding distance of the ith individual

$f[i, j]$: the jth objective value of the ith individual Initialize population

for $i = 1$ to n **do**

$CD[i] = 0$

end for

for $j = 1$ to m **do**

Sort the solution S with objective j in ascending order

$CD[1] = CD[n] = inf$

for $i = 2$ to $n - 1$ **do**

$CD[i] = CD[i] + (f[i + 1, j] - f[i - 1, j])/(max(f[*, j]) - min(f[*, j]))$

end for

end for

Fig. 3. The pseudo-code of calculating crowding distance.

D: size of searching space

n: size of the external archive

$g[i]$: the ith solution in the external archive

$g\_mu[i]$: the mutated ith solution in the external archive

for $i = 1$ to n **do**

$d = random(1, D)$

$\sigma = \sigma_{max} - (\sigma_{max} - \sigma_{min}) \times g/G$

$g\_mu[i] = g[i] + (X_d^{ub} - X_d^{lb}) \times Gaussian(\mu, \sigma)$

Keep $g\_mu[i]$ within the range $[X_d^{lb}, X_d^{ub}]$

Evaluate the objectives with the new solution

end for

Update the external archive with the mutated solution g_mu

Fig. 4. The pseudo-code of elitist learning strategy.

sonal best (*pbest*) and global best (*gbest*). The velocity and position of particle *i* is updated in the following way:

$$v_{i,k+1} = w \times v_{i,k} + c_1 \times r_1 \times (x_{ib} - x_{i,k}) + c_2 \times r_2 \times (x_{gb} - x_{i,k}) \quad (7)$$

$$x_{i,k+1} = x_{i,k} + v_{i,k+1} \quad (8)$$

where $i = 1, 2, \ldots, N$. N is the size of the population, k is the iteration index of the particle, and w is the inertia weight, and it is usually linearly decreasing from 0.9 to 0.4 with particles updating in order to adjust the abilities for global and local search. c_1 and c_2 are two acceleration coefficients, r_1 and r_2 are two uniformed distributed random numbers from interval [0,1]. x_{ib} is the position with best value for the ith particle while x_{gb} is the best position in the entire swarm.

In order to apply PSO in solving multi-objective optimization problems, there are some issues related to be discussed [37]: (1) How to select particles as leaders in the updating process; (2) How to maintain the non-dominated solutions to obtain the optimal Pareto front; (3) How to prevent the particles from converge to local optimal and maintain the diversity of the swarm.

3.2.1. Personal best selection

When selecting personal best solution for MOPSO, the p_{best} is replaced by the new particle if it dominates the previous p_{best} or the two are non-dominated with each other. After all the particles in the swarm have updated their p_{best}, they can be used for global best selection.

3.2.2. Non-dominated solutions maintaining and global best selection

An external archive is introduced to store the non-dominated solutions found by MOPSO. At each generation of the swarm, the new particles are used to updates the external archive. Any dominated solutions are deleted from the archive. The size of the archive will increase during the updating process which adds to the computation cost. Therefore, the archive size should be fixed. We have different methods to maintain the external archive based on global best selection.

For a particle in the swarm, global best selection means selecting a non-dominated solution from external archive as its g_{best}. In this paper, we discuss three methods for global best selection: (1) crowding distance; (2) adaptive grids; (3) comprehensive ranking.

MOPSO-CD [29] uses the crowding distance (CD) to measure the relative density of the non-dominated solutions in the external archive. It is similar to NSGA-II, which use crowding distance as criteria for non-dominated sorting of the individuals. For any particle in the swarm, the g_{best} is selected randomly from a predefined top part of the archive based on non-dominate rank and crowding distance.

Adaptive grids method (GRID) proposed in [16] is another way for updating the external archive in MOPSO. The objective space of the non-dominated solutions is divided into hypercubes. Each hypercube may have different amount of solutions. Those hypercubes with fewer solutions are more likely to retain in the archive. A fitness value is assigned to each hypercube in inverse proportion to the number of non-dominated solutions in it. Roulette-wheel selection is applied to select hypercube with certain particles. Then, we randomly select a particle which is the g_{best}.

Comprehensive ranking (CR) is one of the ranking based methods in MOPSO and has been used in selecting the p_{best} and g_{best} [30]. Ranking information and Chebyshev distance are combined to get the comprehensive ranking measure. Ranking information is obtained by linear combination of average ranking [38] and global detriment [39]. This information is also called fusion ranking (FR). Besides ranking information, density of the solutions is also concerned. Chebyshev distance of a particle to the rest of swarm is used as density measure. Comprehensive ranking is shown as follows:

$$CR(X_i) = \frac{FR(X_i)}{D(X_i)} \quad (9)$$

where X_i is the ith particle in the swarm, $FR(^*)$ is the fusion ranking and $D(^*)$ is the sum of the Chebyshev distance between X_i and the rest of the particles. In this paper, however, CR is only used for global best selecting in external archive maintaining. Similar to MOPSO-CD, the g_{best} is selected randomly from predefined top part of the archive.

3.2.3. Diversity maintaining

In order to avoid early convergence of the particle to a local optimal solution and maintain the diversity of the particles in the

Initialize the particles in the swarm (velocity and position)

Evaluate each of the particle in the swarm

Initialize external archive by non-dominated solution maintaining

$Iter = 0$

while $Iter < MaxIter$ **do**

  for each particle **do**

    Update the velocity and position of the particle

    if $rand() < (1 - g/G)^{(}1/mu)$ and using *rapidly decreasing mutation* **then**

      A *rapidly decreasing mutation* strategy is applied to the particle

    end if

    Evaluate the new particle

    Update p_{best}

  end for

  Update the external archive by non-dominated solution maintaining with the new particles

  if $rand() < \sigma$ and using *elitist learning strategy* **then**

    An *elitist learning strategy* is applied to the external archive

  end if

  Control the archive size and select g_{best} for the swarm from the external archive by crowding distance, adaptive grids, or comprehensive ranking

  $Iter = Iter + 1$

end while

Report result in the external archive

Fig. 5. The pseudo-code of the MOPSO algorithms.

swarm, some mutation techniques are adopted. It is an effective way to avoid convergence to a single solution in MOPSO [40]. In this paper, we consider two methods: rapidly decreasing [16] and elitist learning strategy (ELS) [31].

Rapidly decreasing mutation method is first developed by Coello et al. This method is widely used in most of the algorithms based on MOPSO. The number of the particles being mutated is decreasing rapidly with respect to the number of the iteration. Only one dimension (randomly chosen) of the particle is mutated. The mutation operator may affect all the particles in the swarm at first with the mutation range covers the entire searching space. As the number of the iteration increases, the mutation range of the particle will decrease. The rapidly decreasing can be performed as:

$$P_d = P_d + (X_d^{ub} - X_d^{lb}) \times \left(1 - \frac{g}{G}\right)^{1/mu} \tag{10}$$

where P_d is the dth dimension of the particle, d is randomly selected among the all the dimension of the decision space, X_d^{ub} and X_d^{lb} are the upper and lower bound of the particle in dth dimension. g is the index of current iteration and G is the number of the iterations. mu is the mutation rate.

The ELS is similar with rapidly decreasing in mutation operator with different function. In rapidly decreasing, the mutation range is controlled by a nonlinear function, while the ELS adopt a Gaussian perturbation. Gaussian mutation add more diversity. Because at the iteration g, the mutation range in ELS is not stable, while the mutation range in rapid decreasing is stable. It is very effective for global optimization using PSO [41]. Besides, the ELS only deals with the g_{best} in the swarm. Global best position is an important factor to lead the particles to the optimal position. Therefore, for MOPSO, the particles in the external archives are mutated. The elitist learning is performed as:

$$gBest_d = gBest_d + (X_d^{ub} - X_d^{lb}) \times Gaussian(\mu, \sigma) \tag{11a}$$

$$\sigma = \sigma_{\max} - (\sigma_{\max} - \sigma_{\min}) \times \frac{g}{G} \tag{11b}$$

where $gBest_d$ is the dth dimension of a *gBest*, the $Gaussian(\mu, \sigma)$ is a random number of a Gaussian distribution with mean μ and

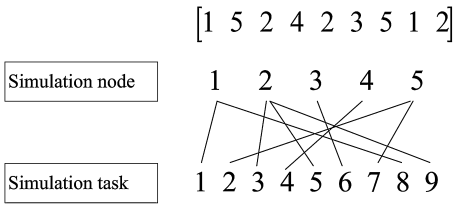


Fig. 6. An example of encoding scheme.

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \end{bmatrix}$$

Fig. 7. An example of decoding scheme.

standard deviation σ. σ is also called "elitist learning rate" with is linearly decreasing with respect to the number of the iteration. σ_{max} and σ_{min} are the upper bound and lower bound of σ. The values of the bounds are set to $\sigma_{max} = 1.0$ and $\sigma_{min} = 0.1$ respectively by empirical study.

With the new particles mutated from ELS, the external archive itself updates. The pseudo-code of ELS procedure is shown in Fig. 4.

The pseudo-code of the provided MOPSO algorithms is shown in Fig. 5.

We divide the MOPSO to six algorithms with different strategies using CD, GRID or CR (MOPSO-CD, MOPSO-GRID or MOPSO-CR) with rapidly decreasing method, and elitist learning strategy (MOPSO-CD-ELS, MOPSO-GRID-ELS or MOPSO-CR-ELS). We will run these algorithms together with NSGA-II in the next section and compare them by using some popular performance metrics.

3.3. Load balancing based on MOEAs

3.3.1. Encoding

There are some common encoded modes: binary encoding, decimal encoding, etc. Each of them shares different advantages as well as disadvantages. Besides, their scopes of application are also different. In this study, there are over 10 simulation tasks and 5 to 8 simulation nodes. Considering that the number of simulation nodes is more than 2. Therefore, the problem should be encoded in integer coding method. According to the definitions of the parameters of the simulation task model, an allocation scheme can be represented by an n-dimensional vector, n is the number of simulation tasks, and the range of each task is limited by m, which is the number of simulation nodes. Fig. 6 provides an example to explain the encoding scheme ($n = 9$, $m = 5$).

3.3.2. Decoding

The solution should be mapped to the load balancing problem, and the allocating tasks of each simulation node are represented in the form of a $n \times m$ matrix, the kth row represents all of the simulation tasks of simulation node k. Fig. 7 provides the corresponding 0-1 decision matrix. And through the matrix, it is obvious to know the whole allocated simulation tasks in each simulation node.

3.3.3. Constraint handling

The penalty function is introduced to obtain more reasonable non-dominated solutions. When a solution is infeasible, both objective values must be fined to penalty values at the same time, so that infeasible solutions will be prevented from being selected into non-dominated solutions.

4. Simulation experiment

In the experiment, the simulation system based on a networked control system is chosen as the platform to verify these algorithms. According to the model parameters, the hardware parameters and a networked control simulation system, the simulation task description diagram is shown in Fig. 8. In Fig. 8, simulation tasks $1 \sim 6$ are federates, tasks $7 \sim 12$ are the first class components of federates, and tasks $13 \sim 20$ are the second class components. Values without asterisks are communication loads between simulation tasks, and the values with asterisks represent computation loads. The computation load of each federate is divided into the first and second class components, and computation loads of the second class components are involved in components of the first class.

In order to compare the performance of different MOEAs when solving the bi-objective load balancing model, three numerical examples are generated.

4.1. Performance metrics

To analysis the performance of NSGA-II and the proposed MOPSOs, we use the following popular performance metrics.

Set coverage (*C-metric*): This measure was proposed by Zitzler and Thiele [42]. Consider two PFs P and Q. $C(P, Q)$ is defined as the percentage of the solutions in Q dominated by at least one solution in P, as in the follows:

$$C(P, Q) = \frac{|\{q \in Q | \exists p \in P : p \text{ dominates } q\}|}{|Q|} \tag{12}$$

where $|X|$ is the size of the PF X. Consider the true PF P^* and an approximation PF P, the lower the value of $C(P^*, P)$, the better the solutions in P.

Spacing (*SP*): This value measures the distribution of the non-dominated solutions along the approximation front. The definition of the metric is as follows [43]:

$$SP = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(d_i - \bar{d})^2} \tag{13}$$

where n is the number of the non-dominated solutions in the approximation front, $d_i = \min_j \sum_{k=1}^{m} |f_k^i - f_k^j|$, $i, j = 1, 2, 3, \ldots, n$, m is the number of the objectives, $\bar{d} = \sum_{i=1}^{n} d_i / n$. The solutions distributed uniformly when the value is close to zero.

Number of non-dominated solutions (*NS*): This metric represents the number of the non-dominated solutions in the set. The solution quality is better when we have large value of *NS*.

Inverted generational distance (*IGD*): This metric reflects both the convergence and diversity of the solutions. Consider a uniformly distributed along the true PF P^* and an approximation PF P, we have the *IGD* as follows [44]:

$$IDG(P, P^*) = \frac{\sum_{v \in P} d(v, P^*)}{|P^*|} \tag{14}$$

where $d(v, P^*)$ is the minimum Euclidean distance between v and all the points in P^*. The smaller value of *IGD* is, the better solution

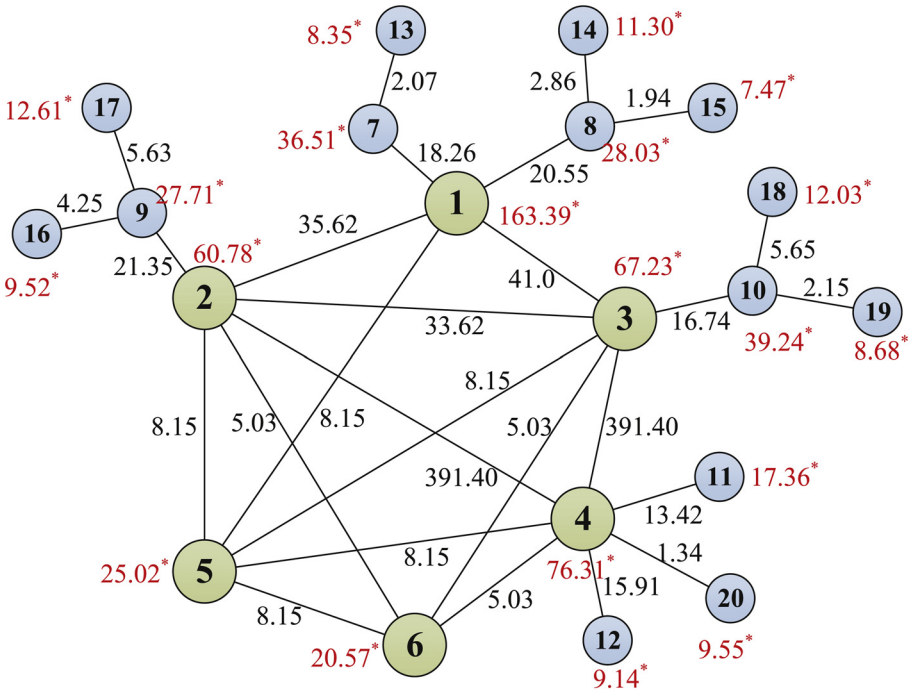


Fig. 8. Description diagram of simulation tasks of a networked control system.

Table 2
NSGA-II parameter ranges and levels.

Parameters	Range	Low(1)	Medium(2)	High(3)
PS	50–100	50	75	100
CP	0.6–0.8	0.6	0.7	0.8
MP	0.01–0.2	0.01	0.1	0.2
NOG	100–300	100	200	300

Table 3
MOPSO parameter ranges and levels.

Parameters	Range	Low(1)	Medium(2)	High(3)
PS	50–100	50	75	100
c_1	1.0–2.0	1.0	1.5	2.0
c_2	1.0–2.0	1.0	1.5	2.0
NOG	100–300	100	200	300

quality of P is. To get a lower *IGD*, the set P should be close to the true PF in any part.

Hypervolume (*HV*) [42]: This metric is obtained by calculating the hypervolume of the approximation PF with a reference point. It is a metric measures both closeness and diversity [45]. The calculation of *HV* is as follows:

$$HV(P) = \left\{ \bigcup_i a(x_i) | \forall x_i \in P \right\} \tag{15}$$

where x_i is an individual in a PF P, and $a(x_i)$ is a rectangle area bounded by a reference point and $f(x_i)$. The reference point adopted here is $[max(f_1), max(f_2)]$. The solution set P with high value of *HV* have better performance.

Since the true PF is unknown, in this paper, the P^* is obtained by merging all the approximation PFs found by all the proposed algorithms.

4.2. Parameter tuning

As mentioned previously, NSGA-II and MOPSO are proposed to find the non-dominted solutions of the bi-objective load balancing model. For NSGA-II, the parameters are population size (*PS*), number of generation (*NOG*), the crossover probability (*CP*), and the mutation probability (*MP*). The MOPSO factors are: population size (*PS*), number of generation (*NOG*), acceleration coefficient c_1, and acceleration coefficient c_2.

Table 4
Calibration process of NSGA-II.

Run order	Algorithm parameters				Response
	PS	*CP*	*MP*	*NOG*	NSGA-II
1	1	1	1	1	3796.3571
2	1	2	2	2	6638.1491
3	1	3	3	3	1343.3568
4	2	1	2	3	1380.7390
5	2	2	3	1	1258.8266
6	2	3	1	2	2481.3429
7	3	1	3	2	1211.8660
8	3	2	1	3	2600.2449
9	3	3	2	1	1526.0817

Since the meta-heuristic algorithms are very sensitive to the parameters, a Taguchi method [46] is utilized to tune the parameters of the algorithms. This method uses a set of arrays called orthogonal arrays which contains the full information of the factors that affect the performance of the algorithms. The factors are categorized into two groups: (1) controllable or signals factors and (2) noise factors. Now, based on the concept of the robustness, the method seeks to minimize the effect of noise and to determin the optimal level of signal factors. To do so, the signal to noise ratio (*S*/*N*), which calculates the amount of variation involved in the response, is used. In this research, the goal is to minimize *S*/*N*. The *S*/*N* is given as

$$\frac{S}{N} = -10 \times \log\left(\frac{S(Y^2)}{n}\right) \tag{16}$$

In Eq. (16), Y and n are the response value and the number of orthogonal arrays, respectively. $S(Y^2)$ is the summation of the response Y^2. In MOEAs, two main goals (convergence and diversity) are considered simultaneously. Among the metrics mentioned in Section 4.1, both *IGD* and *HV* are suitable metrics. However, Taguchi

Table 5
Calibration process of MOPSOs.

Run order	Algorithm parameters				Response					
	PS	*CP*	*MP*	*NOG*	MOPSO-CD	MOPSO-CD-ELS	MOPSO-GRID	MOPSO-GRID-ELS	MOPSO-CR	MOPSO-CR-ELS
1	1	1	1	1	271.0273	90.9035	538.6006	199.8051	294.1980	183.1057
2	1	2	2	2	153.7259	53.2245	316.8323	79.6946	254.3514	81.1352
3	1	3	3	3	159.8617	33.7960	241.4596	73.8463	258.6209	71.3697
4	2	1	2	3	103.0884	28.1024	241.8085	30.8273	302.0519	48.2227
5	2	2	3	1	187.6446	126.1869	486.8986	163.3043	299.8156	215.0099
6	2	3	1	2	165.9011	50.4713	263.1420	104.4232	187.1905	53.7543
7	3	1	3	2	113.8546	35.7942	245.6384	63.4018	227.2489	85.2157
8	3	2	1	3	114.3102	30.0789	249.3944	46.1153	214.9584	55.9144
9	3	3	2	1	174.0252	88.3395	495.3648	165.0683	230.1973	183.3268

Table 6
Optimal values of the parameters.

Algorithms	Parameters	Optimal value
NSGA-II	*PS*	75
	CP	0.8
	MP	0.2
	NOG	300
MOPSO-CD	*PS*	100
	c_1	1.0
	c_2	1.5
	NOG	300
MOPSO-CD-ELS	*PS*	100
	c_1	1.0
	c_2	1.5
	NOG	300
MOPSO-GRID	*PS*	100
	c_1	2.0
	c_2	2.0
	NOG	300
MOPSO-GRID-ELS	*PS*	100
	c_1	1.0
	c_2	1.5
	NOG	300
MOPSO-CR	*PS*	100
	c_1	2.0
	c_2	1.0
	NOG	200
MOPSO-CR-ELS	*PS*	75
	c_1	2.0
	c_2	1.0
	NOG	300

method only deals with one response function. Therefore, a combination of the performance measures should be defined. In this study, a new metric is introduced as

$$C.R. = \frac{IGD}{HV} \tag{17}$$

This new metric is called combinatorial ratio ($C.R.$) and acts as the response variable of the Taguchi method.

In order to utilize the Taguchi method, the levels of the factors are provided in Tables 2 and 3 for the proposed algorithms. In each of the algorithms, three levels are considered for each factor. The L9 design is the best to tune the parameters of the algorithms. This has been a common approach in some studies [47,18,22,21].The orthogonal arrays of these designs along with experimental results are shown in Tables 4 and 5.

For each algorithm, the effect plots of S/N ratio are given in Fig. 9 . Using these plots, the optimal values of the parameters for each algorithm are obtained in Table 6.

4.3. Analysis of results

In the simulation experiment, the load balancing algorithm is programmed and simulated by MATLAB R2010b on a computer including Intel(R) Core(TM) 2 Quad CPU with 2.83 GHz speed and 4 GB of RAM.

In this section, we compare the algorithms in three instances. For each instance, 20 independent runs are performed using each algorithm.

Three numerical examples about the load balancing problem are conducted in this section. Different amount of simulation tasks and nodes are considered.

Numerical example 1: According to the simulation task model in Fig. 8, a load model with 12 simulation tasks and 5 nodes is built. The model for simulation tasks is $A = \{a_i | i = 1, 2, \ldots, 12\}$, and the computation load matrix for simulation tasks is pt = [98.85, 33.07, 27.99, 49.81, 25.02, 20.57, 36.51, 28.03, 27.71, 39.24, 17.36, 9.14]. The communication load matrix is described as follows:

$$mt = \begin{bmatrix}
0 & 35.62 & 41.00 & 0 & 8.15 & 0 & 18.26 & 20.55 & 0 & 0 & 0 & 0 \\
35.62 & 0 & 33.62 & 391.40 & 8.15 & 5.03 & 0 & 0 & 21.35 & 0 & 0 & 0 \\
41.00 & 33.62 & 0 & 391.40 & 8.15 & 5.03 & 0 & 0 & 0 & 16.74 & 0 & 0 \\
0 & 391.40 & 391.40 & 0 & 8.15 & 5.03 & 0 & 0 & 0 & 0 & 13.42 & 15.91 \\
8.15 & 8.15 & 8.15 & 8.15 & 0 & 8.15 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 5.03 & 5.03 & 5.03 & 8.15 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
18.26 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
20.55 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 21.35 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 16.74 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 13.42 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 15.91 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{bmatrix}$$

The model for simulation nodes is $P = \{p_i | i = 1, 2, 3, 4, 5\}$, and the upper bound of its computation load and communication load is 200 and 500.

Numerical example 2: According to the simulation task model shown in Fig. 8, 8 simulation tasks are added based on the first example, and number of simulation nodes is 8. The model for simulation tasks of the experiment is $A = \{a_i | i = 1, 2, \ldots, 20\}$, and the computation load is $pt = [98.85, 33.07, 27.99, 40.26, 25.02, 20.57, 28.16, 9.26, 5.58, 18.53, 17.36, 9.14, 8.35, 11.30, 7.47, 9.52,$

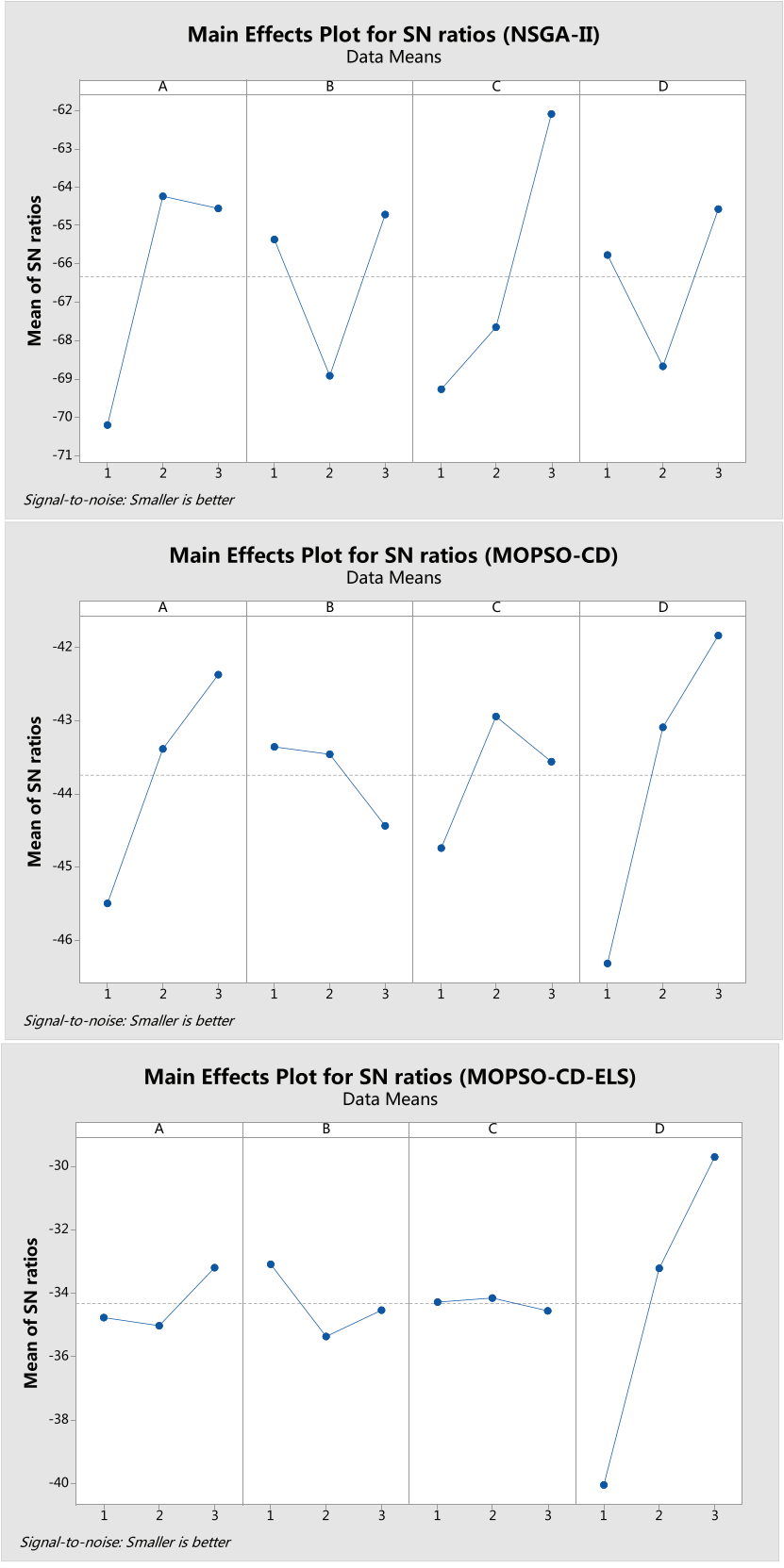
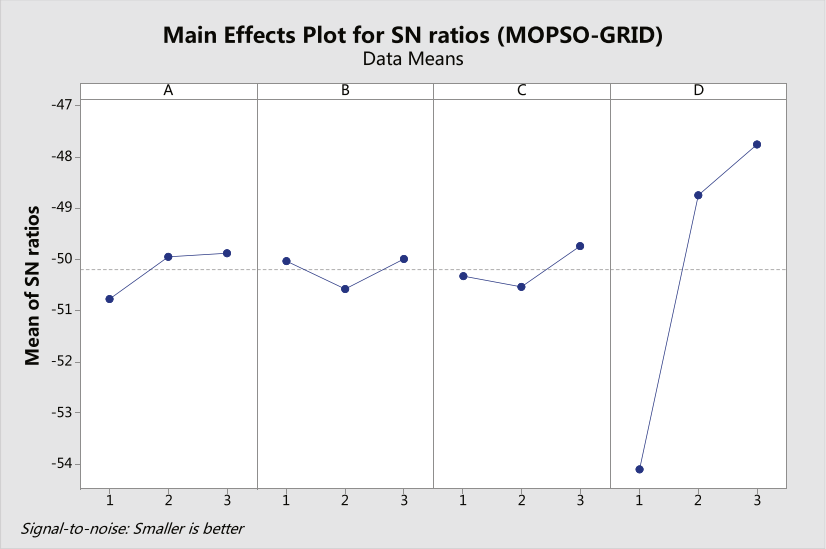
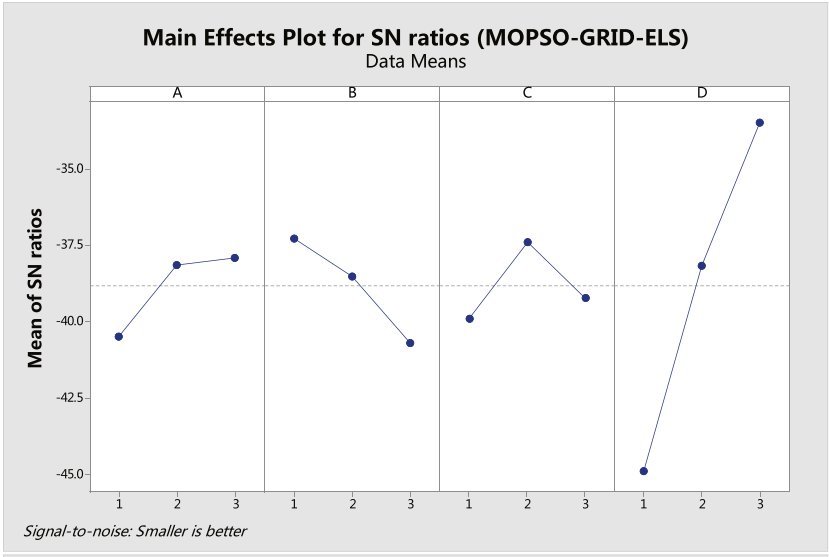


Fig. 9. Taguchi ratios for all proposed algorithms.





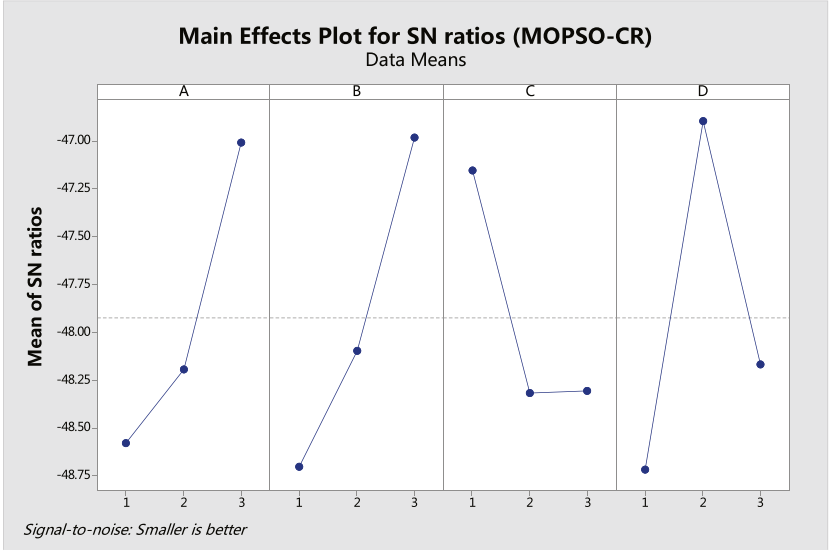


Fig. 9. *Continued*

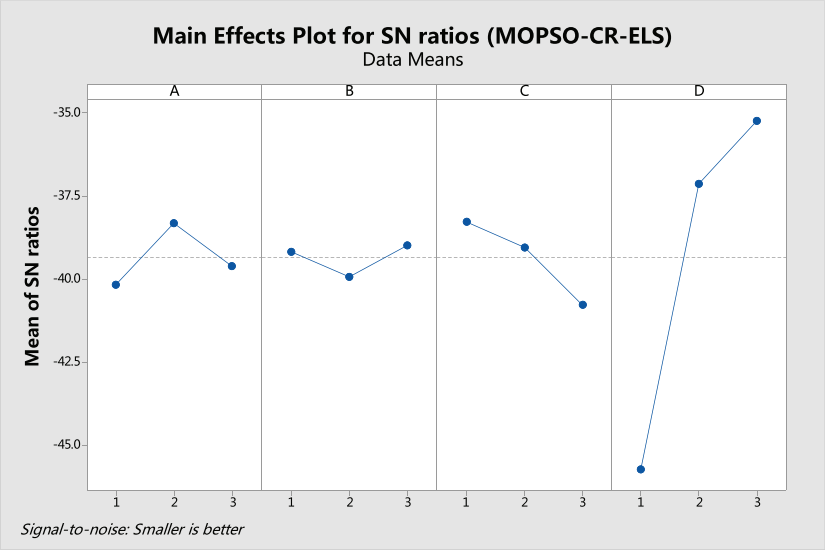


Fig. 9. *Continued*

12.61, 12.03, 8.68, 9.55]. Compared with the first example, only the added communication load will be listed in matrix: $mt_{4,20} = 1.34$, $mt_{7,13} = 2.07$, $mt_{8,14} = 2.86$, $mt_{8,15} = 1.94$, $mt_{9,16} = 4.25$, $mt_{9,17} = 5.63$, $mt_{10,18} = 5.65$, $mt_{10,19} = 2.15$. The model for simulation node, and the upper bound of computation load and communication load are the same as the first example.

Numerical example 3: According to the simulation task model shown in Fig. 8, 5 simulation tasks are added to experiment two during the simulation process to study the applicability of algorithms in dynamic load balancing problem, and number of simulation nodes is 8. In this experiment, allocation for the available 20 tasks is predefined: $A = \{7, 1, 2, 2, 4, 4, 8, 6, 1, 6, 3, 3, 5, 6, 5, 5, 3, 5, 6, 8\}$.

After calculation, the residual calculation ability of each node is [111.51, 56.46, 160.89, 154.41, 162.63, 112.75, 36.61, 153.94]. The upper bound of communication load between simulation node 1 and 2 is 100, and the upper bound between other nodes is 200.

The model of dynamically involved simulation tasks during the experiment is $A = \{a_i | i = 21, 22, 25\}$, and the computation load is $pt = [30.02, 50.13, 12, 19.43, 18.05]$, the communication loads between the involved tasks and other tasks are described as follows: $mt_{1,21} = 14.12$, $mt_{1,24} = 5$, $mt_{2,21} = 34.07$, $mt_{2,25} = 5.11$, $mt_{3,22} = 37$, $mt_{3,23} = 12.03$, $mt_{4,22} = 27$, $mt_{5,21} = 13$, $mt_{5,22} = 13$, $mt_{6,21} = 8$, $mt_{6,22} = 8$, $mt_{21,22} = 21.05$.

Figs. 10–12 show an arbitrary run of the algorithms for numerical examples 1–3. The algorithms with or without ELS are show separately in two subfigure compared to true PF. Tables 7–11 show the average values and standard derivations of the solutions with rankings on each metric for the three numerical examples. The best average result for each example is shown in **boldface**. Figs. 13 and 14 show the plots and the box plots of *C-metric*, *SP*, *NS*, *IGD*, and *HV* of the algorithms for the numerical examples 1 and 2, respectively.

We note that while in terms of *C-metric*, *SP*, and *IGD* metrics, smaller values are better, for *NS* and *HV*, bigger values are better. In examples 1 and 2. Based on the statistical results in Table 7-11 along with Figs. 10, 11, 13 and 14, MOPSO with ELS perform better than MOPSO without ELS and NSGA-II. As shown in Fig. 13 and Fig. 14, in terms of *C-metric*, MOPSO-CD-ELS has the best performance in examples 1 and 2. In terms of *SP*, the algorithms perform statistically similar in example 1, the MOPSO-CD-ELS is the best in example 2. In terms of *NS*, MOPSO-CD-ELS has the best performance in example 1, while MOPSO-GRID-ELS has the best performance in

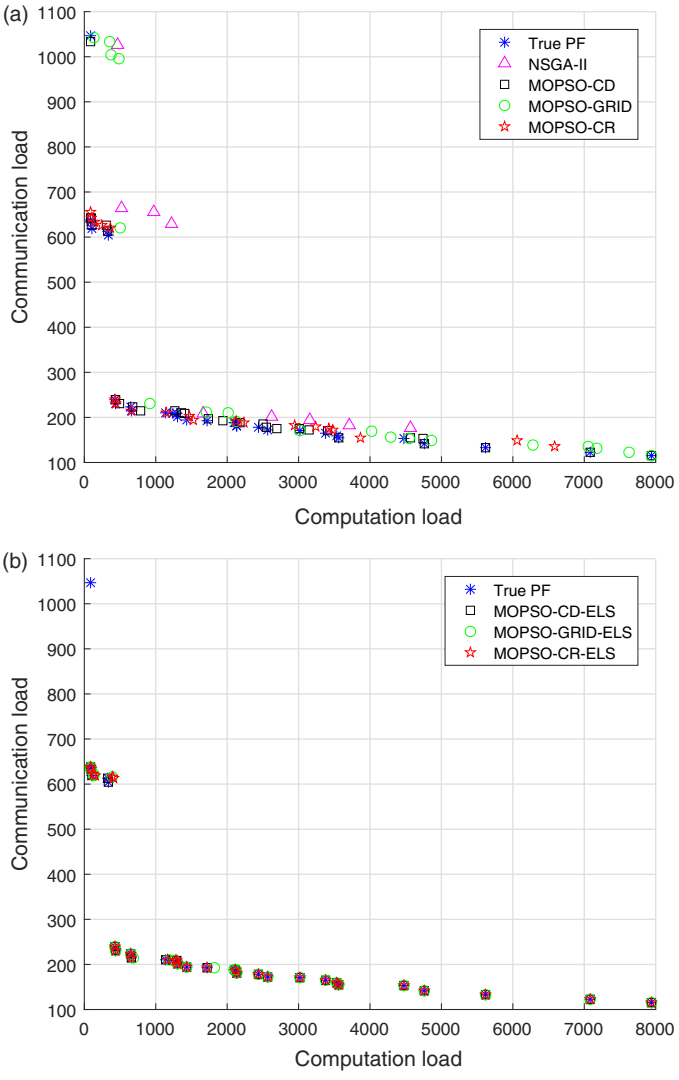


Fig. 10. Non-dominated solutions produced by the algorithms in numerical example 1.

Table 7
Comparisons of *C*-metric (C(P_{true}, P)) between NSGA-II and MOPSOs.

Examples		NSGA-II	MOPSOs					
			MOPSO-CD	MOPSO-CD-ELS	MOPSO-GRID	MOPSO-GRID-ELS	MOPSO-CR	MOPSO-CR-ELS
1	*Ave.*	1.00[7]	0.91[5]	**0.60[1]**	0.95[6]	0.80[3]	0.89[4]	0.71[2]
	St. dev.	0.00	0.08	**0.11**	0.06	0.07	0.08	0.13
2	*Ave.*	1.00[7]	1.00[7]	**0.90[1]**	1.00[7]	0.93[3]	0.98[4]	0.93[2]
	St. dev.	0.00	0.00	**0.08**	0.00	0.05	0.05	0.03
3	*Ave.*	0.40	**0.00**	**0.00**	0.40	**0.00**	0.50	**0.00**
	St. dev.	0.52	**0.00**	**0.00**	0.46	**0.00**	0.53	**0.00**

Table 8
Comparisons of SP between NSGA-II and MOPSOs.

Examples		NSGA-II	MOPSOs					
			MOPSO-CD	MOPSO-CD-ELS	MOPSO-GRID	MOPSO-GRID-ELS	MOPSO-CR	MOPSO-CR-ELS
1	*Ave.*	435.67[7]	290.53[4]	246.17[2]	311.98[5]	**245.45[1]**	339.34[6]	249.14[3]
	St. dev.	392.47	158.93	22.95	101.89	**40.83**	257.53	33.23
2	*Ave.*	93.51[4]	108.62[5]	**55.13[1]**	160.73[6]	70.21[3]	203.10[7]	59.11[2]
	St. dev.	50.66	32.11	**14.00**	111.92	20.45	123.22	15.39
3	*Ave.*	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	St. dev.	0.00	0.00	0.00	0.00	0.00	0.00	0.00

Table 9
Comparisons of NS between NSGA-II and MOPSOs.

Examples		NSGA-II	MOPSOs					
			MOPSO-CD	MOPSO-CD-ELS	MOPSO-GRID	MOPSO-GRID-ELS	MOPSO-CR	MOPSO-CR-ELS
1	*Ave.*	7.60[7]	34.00[5]	**61.20[1]**	43.10[3]	46.90[2]	29.30[6]	42.20[4]
	St. dev.	2.12	5.60	**23.55**	13.14	11.75	9.36	8.20
2	*Ave.*	22.00[7]	45.90[5]	81.50[3]	98.80[2]	**99.20[1]**	31.80[6]	74.10[4]
	St. dev.	6.99	8.76	8.89	1.93	**1.69**	6.58	9.00
3	*Ave.*	1.30	19.40	**25.00**	6.00	**25.00**	3.40	**25.00**
	St. dev.	0.67	2.88	**0.00**	5.08	**0.00**	1.96	**0.00**

Table 10
Comparisons of IGD between NSGA-II and MOPSOs.

Examples		NSGA-II	MOPSOs					
			MOPSO-CD	MOPSO-CD-ELS	MOPSO-GRID	MOPSO-GRID-ELS	MOPSO-CR	MOPSO-CR-ELS
1	*Ave.*	642.06[7]	82.65[4]	26.68[3]	164.49[6]	25.61[2]	147.60[5]	**24.05[1]**
	St. dev.	158.69	46.50	37.11	60.28	8.97	56.86	**9.78**
2	*Ave.*	412.89[7]	75.93[4]	40.39[2]	178.63[6]	**32.68[1]**	153.59[5]	49.08[3]
	St. dev.	225.94	12.41	20.78	30.94	**11.34**	43.43	32.67
3	*Ave.*	79.91	**0.00**	**0.00**	41.34	**0.00**	60.20	**0.00**
	St. dev.	5.98	**0.00**	**0.00**	42.33	**0.00**	31.81	**0.00**

Table 11
Comparisons of HV between NSGA-II and MOPSOs.

Examples		NSGA-II	MOPSOs					
			MOPSO-CD	MOPSO-CD-ELS	MOPSO-GRID	MOPSO-GRID-ELS	MOPSO-CR	MOPSO-CR-ELS
1	*Ave.*	0.46[7]	**0.75[1]**	0.73[2]	0.69[6]	0.71[4]	0.69[5]	0.71[3]
	St. dev.	0.21	**0.05**	0.04	0.07	0.01	0.01	0.00
2	*Ave.*	0.57[7]	0.74[3]	0.75[2]	0.70[6]	**0.76[1]**	0.69[5]	0.74[4]
	St. dev.	0.09	0.01	0.02	0.02	**0.01**	0.03	0.02
3	*Ave.*	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	St. dev.	0.00	0.00	0.00	0.00	0.00	0.00	0.00

example 2. In terms of *IGD*, MOPSO-CR-ELS is the best in example 1, while MOPSO-GRID-ELS is the best in example 2. In terms of *HV*, MOPSO-CD has the best performance in example 1, while MOPSO-GRID-ELS has the best performance in example 2. Fig. 13 and Fig. 14 also show the one-way ANOVA of the performance indices *C-metric*, *SP*, *NS*, *IGD* and *HV* at 95% confidence level along with the values of the corresponding p-values. In example 1, while there are significant differences among the algorithms in terms of the means of

Table 12
The overall ranking of the algorithms based on metrics in numerical examples 1 and 2.

Metric's name	Algorithms						
	NSGA-II	MOPSO-CD	MOPSO-CD-ELS	MOPSO-GRID	MOPSO-GRID-ELS	MOPSO-CR	MOPSO-CR-ELS
C-metric	7	5	1	6	3	4	2
SP	6	4	1	6	2	7	3
NS	7	5	2	3	1	6	4
IGD	7	4	3	6	1	5	2
HV	7	2	2	6	3	5	4

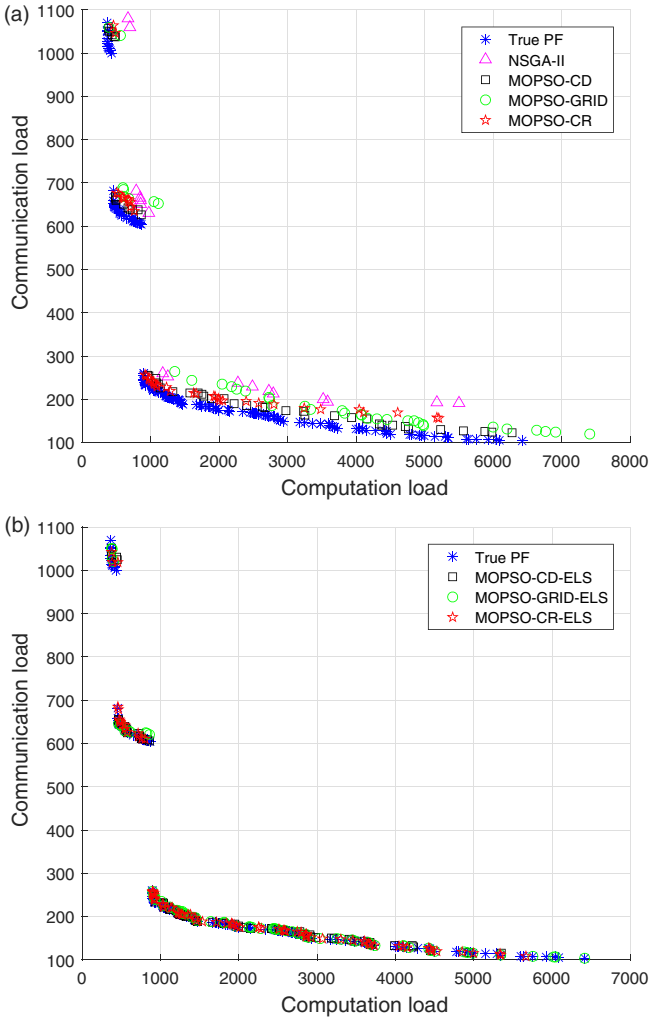


Fig. 11. Non-dominated solutions produced by the algorithms in numerical example 2.

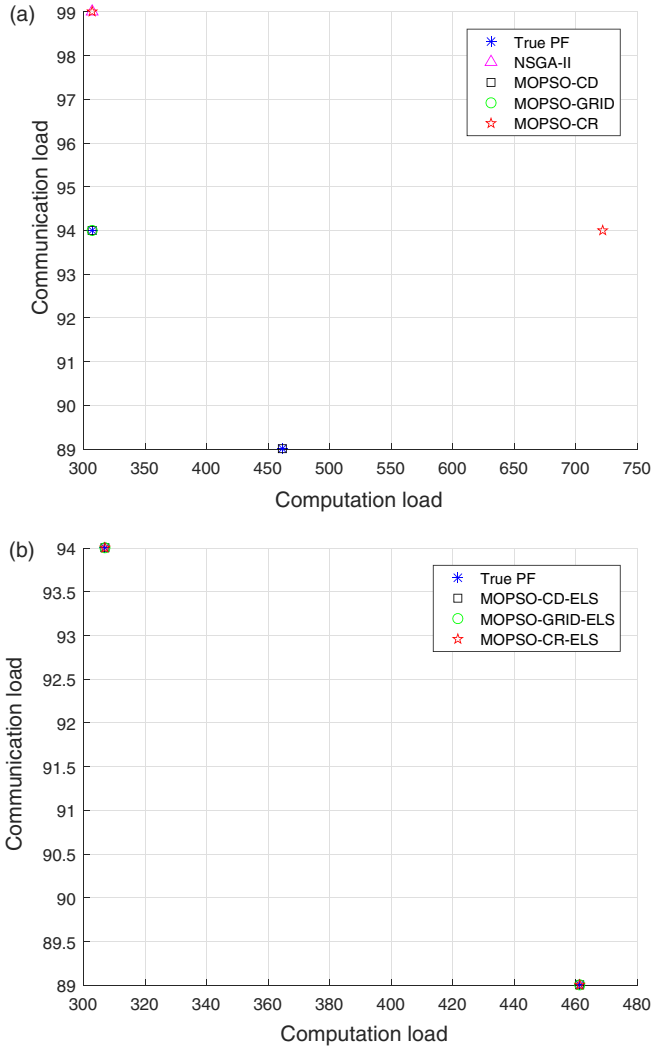


Fig. 12. Non-dominated solutions produced by the algorithms in numerical example 3.

C-metric, *NS*, *IGD* and *HV*, there are no significant differences among the algorithms in terms of the means of *SP*. In example 2, there are significant differences among the algorithms in terms of the means of all the five metrics. Table 12 presents the overall rankings of the algorithms according to the five performance metrics of numerical examples 1 and 2, respectively, as well as the overall ranking. As shown in this table, MOPSO-CD-ELS ranks first among the algorithms in terms of *C-metric* and *SP*, MOPSO-GRID-ELS are better in terms of *NS* and *IGD*, and MOPSO-CD as well as MOPSO-CD-ELS rank better in terms of *HV*.

In example 3, the dynamic involvement of new tasks is considered. Due to the limit abilities of the residual nodes, only two kinds of non-dominated solutions are found on the PF in Fig. 12. MOPSO-CD-ELS, MOPSO-GRID-ELS and MOPSO-CR-ELS find the true PF based on Table 9 with value 25. These three algorithms show better performance from Table 7-11. The value of *SP* and *HV* value is 0, since there are only two non-dominated solutions.

Finally, by observing these performance results, we can see that MOPSOs with ELS operator outperform the other MOPSOs and NSGA-II in terms of different metrics. As compared to NSGA-II, those MOPSO algorithms show better results, due to the information sharing mechanism in MOPSO. It is concluded that ELS mutation operator has significant improvements on MOPSOs. There are two reasons:

1. ELS deals with the global best particle, while rapid decreasing deals with the whole particles in the swarm. Therefore, ELS is more likely to converge to the optimal position.

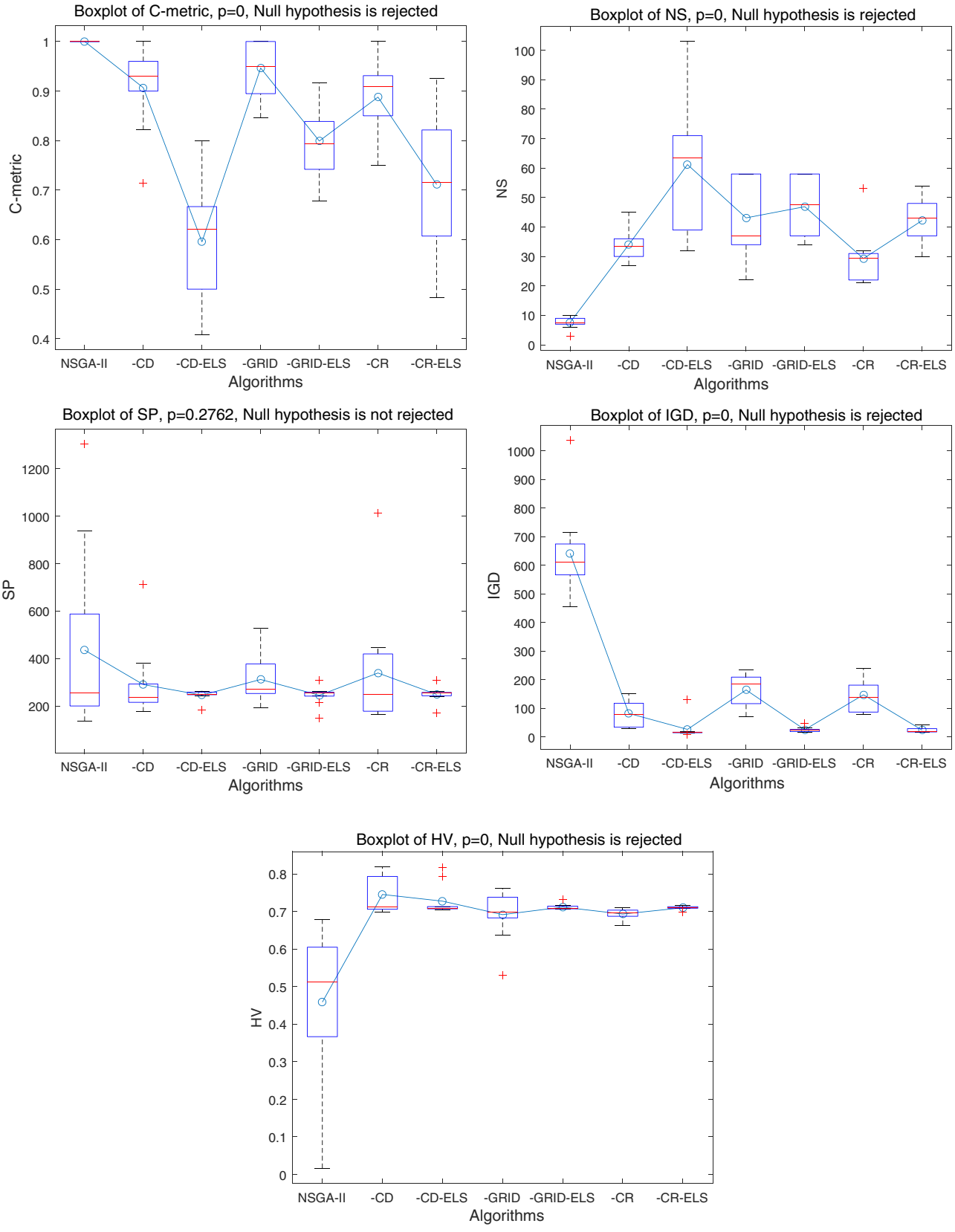


Fig. 13. The plot and the box-plot of the metrics in numerical example 1.

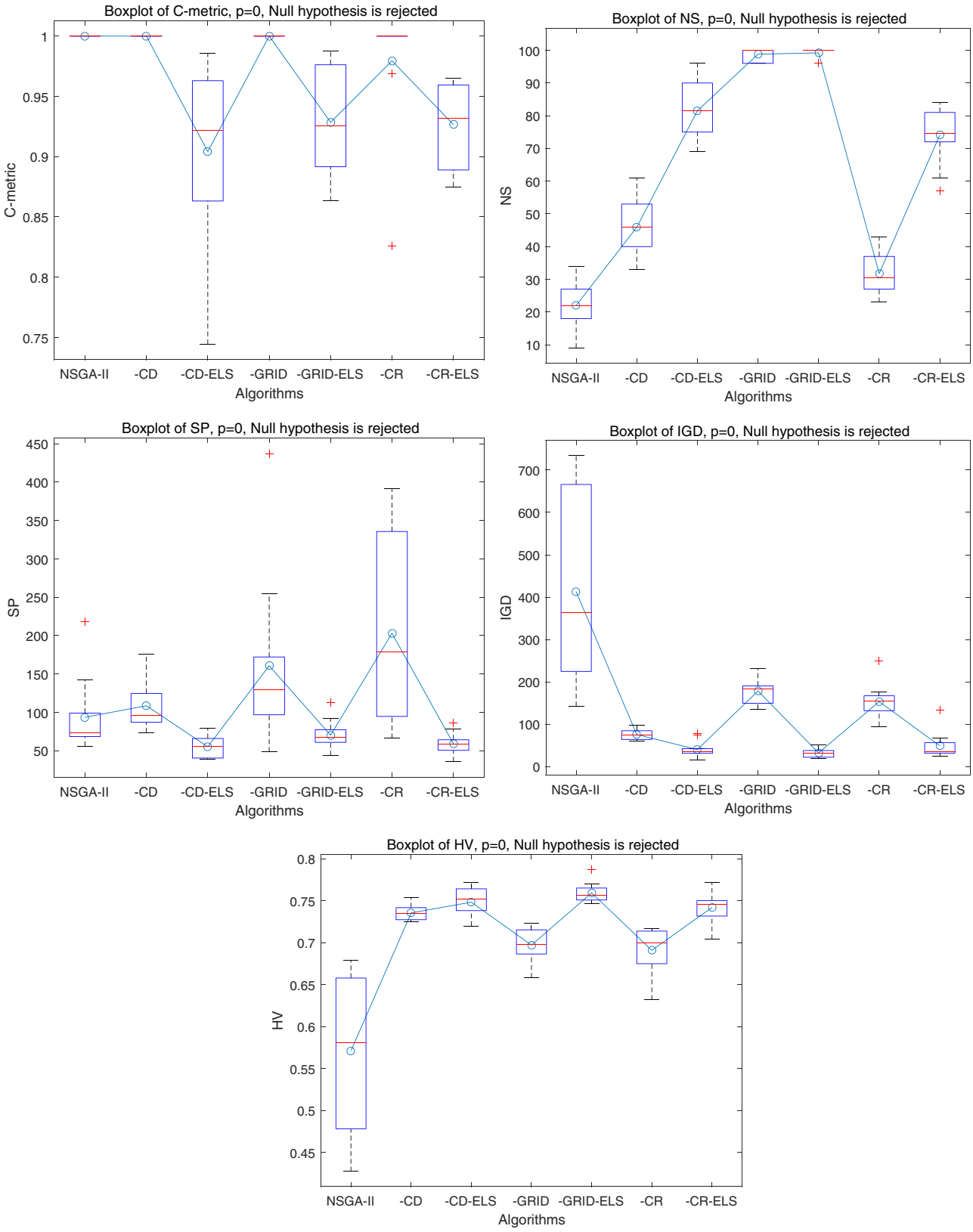


Fig. 14. The plot and the box-plot of the metrics in numerical example 2.

2. The perturbation in ELS is a stochastic function (Gaussian perturbation) rather than a stable function in rapid decreasing. Stochastic perturbation is better than stable perturbation since it provides more diversity.

5. Schemes verification

The distribution schemes of the simulation experiment are verified in the simulation system based on a networked control system.

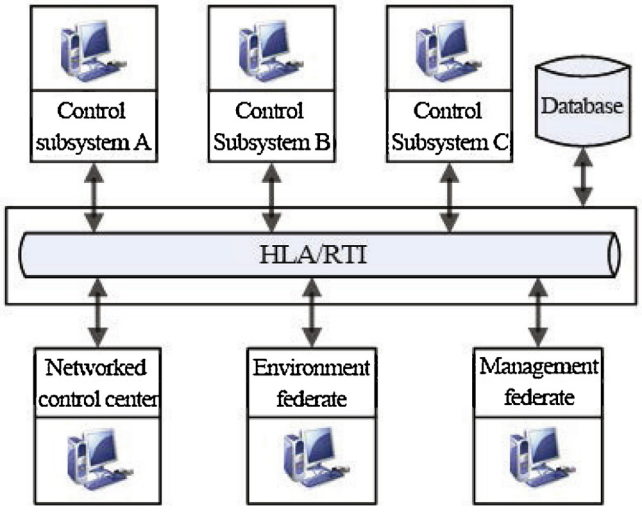


Fig. 15. Structure diagram of networked control simulation system.

Table 13
Test results of the optimization schemes of 12 simulation tasks for load mode 1.

Scheme number	1	2	3
Simulation node 1	1,8	1	1
Simulation node 2	7,11	7,8	3,4
Simulation node 3	10	9,10	2,6,9
Simulation node 4	6,9	5,6,11,12	5,7,11
Simulation node 5	2,3,4,5,12	2,3,4	8,10,12
Computation load	2435.39	436.20	85.18
Communication load	177.78	230.54	637.33
Runtime(s)	1376.5	1053.8	923.5

In this simulation system, the development tool of federates and components is Visual C++ 6.0, Oracle 9i is the database subsystem of the simulation system. Geographic information system (GIS) is applied to provide a platform for the implementation of simulation tasks. The federal executive files are developed by the object model development tool (OMDT), and the operating environment of the simulation system is pRTI 1.3. The simulation system is divided into control subsystem A/B/C, networked control center, environment and management federates. Each federate includes a number of reusable components. The composition is shown in Fig. 15.

The operating environment of the simulation system is a LAN of 100Mbps bandwidth, it includes 8 computers, and each computer is considered as a simulation node. In this section, two load models based on numerical examples 1 and 2 are considered and three representation schemes are selected by the proposed load balancing algorithm. According to the load distribution matrix, federates/components are arranged to their corresponding simulation nodes. The total runtime of the simulation system will be recorded to determine their merits and demerits. The test data are shown in Table 13 and Table 14.

Through the comparison of the three schemes under each load model, it shows the runtime results of the non-dominated solutions. Both the third scheme has the lowest runtime.

6. Conclusions and future work

In this paper, we focused on load balancing problem of the distributed simulation system based on HLA. Firstly, the load problem was formulated as a bi-objective model. The goal was to minimize the imbalanced computation load and total communication load. Then, two kinds of MOEAs were provided to solve the load distribution problem of the simulation system. With several improvement strategies of the algorithm, convergence and optimization efficiency of the algorithms were improved. Besides, considering the actual requirement of the simulation, dynamic involvement of federates and components were put forward and a new idea of dynamic load balancing problem was formed. Moreover, Taguchi method was implemented to tune the parameters of the algorithms with a novel response value considering the convergence and diversity of the solutions. Finally, three numerical examples were conducted to evaluate the performance of the provided algorithms and schemes verification was done on the networked control simulation platform. To some extent, this study could solve the load balancing problem of the distributed simulation system based on HLA, and it is of great significance for improving the simulation propulsive efficiency.

Table 14
Test results of the optimization schemes of 20 simulation tasks for load mode 2.

Scheme number	1	2	3
Simulation node 1	5,11	2,9	5,10
Simulation node 2	2,3,4,9	3,4	9,13,15,16,17
Simulation node 3	6,12	11,12,17	2,19
Simulation node 4	10,18,19	5,6	4,12
Simulation node 5	13,16,20	13,15,16,18	3,14,18
Simulation node 6	7,8,15	8,10,14,19	6,11,20
Simulation node 7	14,17	1	1
Simulation node 8	1	7,20	7,8
Computation load	1057.71	468.52	383.21
Communication load	233.49	655.09	1055.21
Runtime(s)	1830.7	1677.3	1503.2

In order to extend this paper, we have two ways. The first can be developing new strategies of MOPSO or other MOEAs: decomposition-based multi-objective evolutionary algorithm with the ε-constraint [48] to solve the bi-objective load balancing model. The ELS improves the performance a lot. Some self-adaptive mutation mechanism [49] on ELS, searching algorithm (gravitational search algorithm [50]) or other learning mechanisms (human learning optimization [51], [52,53]) can be applied in future work. The other one is consider more realistic and complex application simulated in HLA system and consider some uncertainties [54] in federates to model the distributed simulation system.

Acknowledgements

We thank Juan Li of Beijing Institute of Technology for her suggestions on improving this paper. We also thank the editors and anonymous reviewers for their helpful comments and suggestions on improving the presentation of this paper. This work was supported by the National Natural Science Foundation of China (Grant Nos. 61673058, U1609214, 61304215), the Beijing Outstanding Ph.D. Program Mentor (Grant No. 20131000704), the Foundation for Innovative Research Groups of the National Natural Science Foundation of China (Grant No. 61321002) and the Doctoral Program Foundation of Institutions of Higher Education of China (Grant No. 20131101120033), and funded by China Scholarship Council. P.M. Pardalos is funded by the Paul and Heidi Brown Preeminent Professor at Industrial and Systems Engineering, University of Florida.

References

[1] S. I. S. Committee, Others, of the IEEE Computer Society, IEEE Standard for Modeling and Simulation (M&S) High Level Architecture (HLA) – IEEE Std 1516-2000, 1516.1-2000, 1516.2-2000, Institute of Electrical and Electronics Engineers, Inc., New York, 2000.
[2] K. Lwin, R. Qu, G. Kendall, A learning-guided multi-objective evolutionary algorithm for constrained portfolio optimization, Appl. Soft Comput. 24 (2014) 757–772.
[3] V.L. Boginski, C.W. Commander, P.M. Pardalos, Y. Ye, Sensors: Theory, Algorithms, and Applications, vol. 61, Springer Science & Business Media, 2011.

[4] S. Ding, C. Chen, B. Xin, J. Chen, Status and progress in deployment optimization of firepower units, Kongzhi Lilun Yu Yingyong/Control Theory Appl. 32 (12) (2015) 1569–1581.
[5] W. Cai, Z. Yuan, M.Y.H. Low, S.J. Turner, Federate migration in HLA-based simulation, Future Gener. Comput. Syst. 21 (1) (2005) 87–95.
[6] G. Tan, A. Persson, R. Ayani, HLA federate migration, in: Proc. 38th Annu. Symp. Simul., IEEE Computer Society, 2005, pp. 243–250.
[7] M. Eklöf, M. Sparf, F. Moradi, R. Ayani, Peer-to-peer-based resource management in support of HLA-based distributed simulations, Simulation 80 (4-5) (2004) 181–190.
[8] S.S. Wu, D. Sweeting, Heuristic algorithms for task assignment and scheduling in a processor network, Parallel Comput. 20 (1) (1994) 1–14.
[9] V.D. Martino, M. Mililotti, Scheduling in a grid computing environment using genetic algorithms, Int. Parallel Distrib. Process. Symp. (2002) 235.
[10] H. Wei, C. Liu, Q. Li, W. Wang, Interaction priority algorithm based entities static scheduling strategy for simulation over the grids, J. Chin. Comput. Syst. 1 (2006) 23.
[11] X. Zhao, Research of dynamic layout technique of simulation federate and its implementation (Ph.D. thesis), National University of Defense Technology, Changsha, 2009.
[12] Y.L. Pan, X.Y. Yao, K.D. Huang, The research of static task scheduling strategy for distributed read-time simulation system(DRTSS), Comput. Simul. 27 (3) (2010) 102–105.
[13] X. Ban, B. Gan, P. Wu, A two-stage allocation optimization in static load balancing of HLA distributed simulation, in: Int. Conf. Cybersp. Technol. (CCT 2013), IET, 2013, pp. 56–61.
[14] Y. Yue, W. Fan, T. Xiao, C. Ma, Novel models and algorithms of load balancing for variable-structured collaborative simulation under HLA/RTI, Chin. J. Mech. Eng. 26 (4) (2013) 629–640.
[15] K. Deb, A. Pratap, S. Agarwal, T. Meyarivan, A fast and elitist multiobjective genetic algorithm: NSGA-II, IEEE Trans. Evol. Comput. 6 (2) (2002) 182–197.
[16] C.A.C. Coello, G.T. Pulido, M.S. Lechuga, Handling multiple objectives with particle swarm optimization, IEEE Trans. Evol. Comput. 8 (3) (2004) 256–279.
[17] W. Hu, G.G. Yen, Adaptive multiobjective particle swarm optimization based on parallel cell coordinate system, IEEE Trans. Evol. Comput. 19 (1) (2015) 1–18.
[18] P. Fattahi, V. Hajipour, A. Nobari, A bi-objective continuous review inventory control model: Pareto-based meta-heuristic algorithms, Appl. Soft Comput. 32 (2015) 211–223.
[19] J. Li, J. Chen, B. Xin, Efficiently solving multi-objective dynamic weapon-target assignment problems by NSGA-II, Proc. 34th Chinese Control Conf. (2015 CCC) (2015) 2556–2561.
[20] J. Li, J. Chen, B. Xin, L.H. Dou, Solving multi-objective multi-stage weapon target assignment problem via adaptive NSGA-II and adaptive MOEA/D: a comparison study, in: Proc. Evol. Comput. (CEC), 2015 IEEE Congr., IEEE, 2015, pp. 3132–3139.
[21] S.M. Mousavi, J. Sadeghi, S.T.A. Niaki, M. Tavana, A bi-objective inventory optimization model under inflation and discount using tuned Pareto-based algorithms: NSGA-II, NRGA, and MOPSO, Appl. Soft Comput. 43 (2016) 57–72.
[22] M. Afzalirad, J. Rezaeian, A realistic variant of bi-objective unrelated parallel machine scheduling problem: NSGA-II and MOACO approaches, Appl. Soft Comput. 50 (2017) 109–123.
[23] R. Salimi, H. Motameni, H. Omranpour, Task scheduling using NSGA II with fuzzy adaptive operators for computational grids, J. Parallel Distrib. Comput. 74 (5) (2014) 2333–2350.
[24] R.F. Subtil, E.G. Carrano, M.J.F. Souza, R.H.C. Takahashi, Using an enhanced integer NSGA-II for solving the multiobjective generalized assignment problem, in: IEEE Congr. Evol. Comput., IEEE, 2010, pp. 1–7.
[25] N. Dahmani, S. Krichen, Solving a load balancing problem with a multi-objective particle swarm optimisation approach: application to aircraft cargo transportation, Int. J. Oper. Res. 27 (1–2) (2016) 62–84.
[26] F. Ramezani, J. Lu, J. Taheri, A.Y. Zomaya, A Multi-objective Load Balancing System for Cloud Environments, 2017.
[27] E.S. Alkayal, N.R. Jennings, M.F. Abulkhair, Efficient task scheduling multi-objective particle swarm optimization in cloud computing, in: 2016 IEEE 41st Conference on Local Computer Networks Workshops (LCN Workshops), IEEE, 2016, pp. 17–24.
[28] L. Zhu, R. Tang, Y. Tao, M. Ren, L. Xue, Multi-objective ant colony optimization algorithm based on load balance, in: International Conference on Cloud Computing and Security, Springer, 2016, pp. 193–205.
[29] C.R. Raquel, P.C. Naval Jr., An effective use of crowding distance in multiobjective particle swarm optimization, in: Proc. 7th Annu. Conf. Genet. Evol. Comput., ACM, 2005, pp. 257–264.
[30] L. Li, W. Wang, W. Li, X. Xu, Y. Zhao, A novel ranking-based optimal guides selection strategy in MOPSO, Procedia Comput. Sci. 91 (2016) 1001–1010.
[31] Z.-H. Zhan, J. Zhang, Y. Li, H.S.-H. Chung, Adaptive particle swarm optimization, IEEE Trans. Syst. Man Cybern. Part B 39 (6) (2009) 1362–1381.
[32] Y. Zhou, W. Dai, HLA Simulation Program Design, Publ. House Electron. Ind., Beijing, 2002, pp. 55–157.
[33] R.E. De Grande, A. Boukerche, Distributed dynamic balancing of communication load for large-scale HLA-based simulations, in: Comput. Commun. (ISCC), 2010 IEEE Symp., IEEE, 2010, pp. 1109–1114.
[34] H. Rahmawan, Y.S. Gondokaryono, The simulation of static load balancing algorithms, in: 2009 Int. Conf. Electr. Eng. Informatics, vol. 2, IEEE, 2009, pp. 640–645.
[35] C. Banino, O. Beaumont, L. Carter, J. Ferrante, A. Legrand, Y. Robert, Scheduling strategies for master-slave tasking on heterogeneous processor platforms, IEEE Trans. Parallel Distrib. Syst. 15 (4) (2004) 319–330.
[36] J. Kennedy, R. Eberhart, Particle swarm optimization, in: Neural Networks, 1995. Proceedings, IEEE Int. Conf., vol. 4, IEEE, 1995, pp. 1942–1948.
[37] M. Reyes-Sierra, C.A.C. Coello, Multi-objective particle swarm optimizers: a survey of the state-of-the-art, Int. J. Comput. Intell. Res. 2 (3) (2006) 287–308.
[38] P.J. Bentley, J.P. Wakefield, Finding acceptable solutions in the Pareto-optimal range using multiobjective genetic algorithms, in: Soft Comput. Eng. Des. Manuf., Springer, 1998, pp. 231–240.
[39] M. Garza-Fabre, G.T. Pulido, C.A.C. Coello, Ranking methods for many-objective optimization, in: Mex. Int. Conf. Artif. Intell., Springer, 2009, pp. 633–645.
[40] H. Moslemi, M. Zandieh, Comparisons of some improving strategies on MOPSO for multi-objective (r, Q) inventory system, Expert Syst. Appl. 38 (10) (2011) 12051–12057.
[41] S. Ding, C. Chen, J. Chen, B. Xin, An improved particle swarm optimization deployment for wireless sensor networks, J. Adv. Comput. Intell. Intell. Inform. 18 (2) (2014) 107–112.
[42] E. Zitzler, L. Thiele, Multiobjective optimization using evolutionary algorithms – a comparative case study, in: Int. Conf. Parallel Probl. Solving from Nat., Springer, 1998, pp. 292–301.
[43] J.R. Schott, Fault Tolerant Design Using Single and Multicriteria Genetic Algorithm Optimization, Tech. Rep., DTIC Document, 1995.
[44] Q. Zhang, H. Li, MOEA/D: a multiobjective evolutionary algorithm based on decomposition, IEEE Trans. Evol. Comput. 11 (6) (2007) 712–731.
[45] G.G. Yen, Z. He, Performance metric ensemble for multiobjective evolutionary algorithms, IEEE Trans. Evol. Comput. 18 (1) (2014) 131–144.
[46] G. Taguchi, Introduction to Quality Engineering: Designing Quality into Products and Processes, 1986.
[47] S.H.A. Rahmati, V. Hajipour, S.T.A. Niaki, A soft-computing Pareto-based meta-heuristic algorithm for a multi-objective multi-server facility location problem, Appl. Soft Comput. 13 (4) (2013) 1728–1740.
[48] J. Chen, J. Li, B. Xin, DMOEA-εC: decomposition-based multi-objective evolutionary algorithm with the ε-constraint framework, IEEE Trans. Evol. Comput. 21 (5) (2017) 714–730.
[49] H. Wang, W. Wang, Z. Wu, Particle swarm optimization with adaptive mutation for multimodal optimization, Appl. Math. Comput. 221 (2013) 296–305.
[50] J. Pei, B. Cheng, X. Liu, P.M. Pardalos, M. Kong, Single-machine and parallel-machine serial-batching scheduling problems with position-based learning effect and linear setup time, Ann. Oper. Res. (2017) 1–25.
[51] L. Wang, R. Yang, H. Ni, W. Ye, M. Fei, P.M. Pardalos, A human learning optimization algorithm and its application to multi-dimensional knapsack problems, Appl. Soft Comput. 34 (2015) 736–743.
[52] L. Wang, L. An, J. Pi, M. Fei, P.M. Pardalos, A diverse human learning optimization algorithm, J. Glob. Optim. 67 (1-2) (2017) 283–323.
[53] L. Wang, J. Pei, M.I. Menhas, J. Pi, M. Fei, P.M. Pardalos, A hybrid-coded human learning optimization for mixed-variable optimization problems, Knowl. Based Syst. 127 (2017) 114–125.
[54] J. Li, J. Chen, B. Xin, L.H. Dou, Z.H. Peng, Solving the uncertain multi-objective multi-stage weapon target assignment problem via MOEA/D-AWA, in: Proc. Evol. Comput. (CEC), 2016 IEEE Congr., IEEE, 2016, pp. 4934–4941.