Correlation-aware resource service composition and optimal-selection in manufacturing grid

Fei Tao\textsuperscript{a,b,*}, Dongming Zhao\textsuperscript{b}, Hu Yefa\textsuperscript{a}, Zude Zhou\textsuperscript{a}

\textsuperscript{a} Hubei Digital Manufacturing Key Laboratory, School of Mechanical and Electronic Engineering, Wuhan University of Technology, Wuhan 430070, China
\textsuperscript{b} Department of Electrical and Computer Engineering, The University of Michigan, Dearborn, MI 48128-1491, USA

\section*{1. Introduction\textsuperscript{1}}

Manufacturing grid (MGrid) utilizes grid technologies, information technologies, computer and advanced management technologies, to conquer the barrier resulted from spatial distance in collaboration among different corporations to make various manufacturing resources, including design resources, manufacturing resources, human resources, and application system resources, to be fully connected. In an MGrid system, various manufacturing resources distributed in heterogeneous systems and in multiple sites can offer numerous manufacturing services to users in transparent ways by encapsulating and integrating resources into different corresponding resource service templates. Users can use all remote resources in an MGrid system as conveniently if they are local resources\textsuperscript{[1–4]}. In an MGrid system, a submitted manufacturing task (or resource service requestor) can be classified into two types: (a) single resource service request task (SRSRTask), and (b) multi-resource service request task (MRSRTask). Generally, SRSRTask is completed by invoking only one resource service while MRSRTask is completed by invoking several resource services in sequence. For a SRSRTask, the system searches all resource service qualifies for its function requirements and selects the optimal one for implementation. An MRSRTask usually can be decomposed into several subtasks that cannot be farther decomposed and can be executed by a single resource service. Hence, for an MRSRTask, in addition to searching all qualified resource services according to each subtask, the system has to select one candidate

\textsuperscript{*} Corresponding author. Address: Hubei Digital Manufacturing Key Laboratory, School of Mechanical and Electronic Engineering, Wuhan University of Technology, Wuhan 430070, China.
\textsuperscript{1} This paper is supported by Excellent Doctoral Dissertation Fund of Wuhan University of Technology (WHUT), Hubei Digital Manufacturing Key Laboratory Opening Fund Project: Research on resources service search and optimal-selection theories and experiments in manufacturing grid system (No. SZ0621) and National Nature Science Fund Project of China (No. 50620130441).

0377-2217/\$ - see front matter \textcopyright 2009 Elsevier B.V. All rights reserved.
doi:10.1016/j.ejor.2009.02.025
resource service for each subtask and generate a composite resource service (CRS), then selects the optimal resource service composite path from all possible paths to execute the task under multi-objectives, e.g., time minimization, cost minimization, and reliability maximization, along with given constrains. The above problem is defined as multi-objectives MGrid resource service composition and optimal-selection (MO-MRSCOS) problem [4].

For the problem of resource service composition, a variety of papers have recently been published. Kalasapur et al. [5] employed the service-oriented middleware platform called Pervasive Information Communities Organization (PICO) to model and represent resources as services. Their proposed service composition mechanism models services as directed attributed graphs, maintains a repository of service graphs, and dynamically combines multiple basic services into complex services. In order to allow a user to request a service in an intuitive form (e.g., using a natural language), Fujii and Suda [6] proposed a semantics-based service composition architecture. Their proposed architecture obtains the semantics of the service requested in an intuitive form, and dynamically composes the requested service based on the semantics of the service. The key Components of the proposed architecture, including Component Service Model with Semantics (CoSMoS), Component Runtime Environment (CoRE), Semantic Graph-Based Service Composition (SeGSec), are described in detail. In order to address the challenges of scalability, flexibility and quality-of-service (QoS) management for distributed multimedia service composition, Gu and Nahrstedt [7] presented a fully decentralized service composition framework, called SpiderNet, which provides statistical multi-constrained QoS assurances and load balancing for service composition and supports directed acyclic graph composition topologies and exchangeable composition orders. A context-based multi-type policy approach for web service composition is investigated in [8]. The design with object (DWO) approach to web services composition is studied in [9]. Yue et al. [10] presents an approach for automating geospatial web service composition by employing geospatial semantics in the service-oriented architecture (SOA). Rao et al. [11] introduced a method for automatic composition of semantic web services using linear logic (LL) theorem proving. Chi and Lee [12] proposed a composing platform based on a formal modeling language. An approach to facilitate dynamic and scalable web service composition called Integrated Service Planning and Execution (ISPE) based on AI planning techniques is put forward by Madhusudan and Uttamsingh [13]. Implementation of an architecture for dynamic web service composition and execution based on a domain-independent AI planning framework called Hierarchical Task Network (HTN) planning is described by them too. Sirin et al. [14] studied how HTN planning system SHOP2 can be used for web service composition. They first translate OWL-S service descriptions to a SHOP2 domain, and then investigated the composition method using SHOP2.

However, the above research works on service composition primarily concentrate on the composing method and architecture. These works fall shot of addressing the following issues: (a) the correlation among resource services are not considered, (b) the service composing process models is less understood, (c) methods to select an optimal composite path from all possible composite paths are less investigated.

To address the above issues, the correlations among resource services are taken into account during MGrid resource service composition, and a QoS description mode supporting resource service correlation is presented. The basic resource service composite modes (RSCM) for CRS are described, and the rules for translating a complicated RSCM into a simple sequence RSCM are presented for simplifying the resolving process and complexity of MO-MRSCOS problem. Then, the formulation of an MO-MRSCOS problem for minimizing execution time and cost, and maximizing the reliability is introduced. A new MGrid resource service composition and optimal-selection method, based on the principles of particle swarm optimization (PSO), is proposed for solving MO-MRSCOS problem. Unlike previous works: (a) the proposed PSO algorithms combine the non-dominated sorting technique to achieve the selection of global best position and private best position; (b) the parameters of particle updating formulation in PSO are dynamical generated in order to make a compromise between the global exploration and local exploitation abilities of PSO; (c) permutation-based and objective-based population trimming operators are applied in PSO to maintain diversity of solutions in population.

This paper is organized as follows: Section 2 describes the formulation of MO-MRSCOS problem for minimizing execution time and cost, and maximizing the reliability. Section 3 presents the proposed PSO algorithms for MO-MRSCOS problem. Section 4 reports the efficiency and effectiveness of the proposed method. Section 5 concludes this work.

2. Problem formulation and review

2.1. Motivating example

Before introducing the problem, an example is presented to illustrate the problems that are addressed in this study. The task of a product design and manufacture, denoted by $T_{prod}$, on an MGrid system is considered as the example. Apparently, $T_{prod}$ is an MRSRTask as shown in Fig. 1, and it has five subtasks such as D&A, MPA, Mfg, DA, and EP. It has to be completed by invoking five resource services such as design & analysis service (D&AS), manufacturing plan and arrangement service (MPAS), manufacture service (MfgS), delivery arrangement service (DAS), and electronic pay service (EPS). Each resource service has several candidates. In the absence of automated composition of resource services, the user invests considerable resources visiting numerous sites, determining appropriate service providers, entering his preferences repeatedly, and integrating or aligning the different type of results coming from different sites. It is preferred that the user enters information once and receives the expected results from the most appropriate services with minimal additional assistance. One possible interaction model is shown in Fig. 1.

In this scenario, the user sends a single request to a service, CPDMS shown in Fig. 1, containing the information required about $T_{prod}$, e.g., the design parameters, functional requirements, price, and delivery address. As it is unlikely to find an ad hoc service to meet the user’s requirements, the service the user interacts with tries to find several services providing as a result of one or more of the needed information. Once these services are discovered, the CPDMS composes the functionalities leading to the needed results and resolves their data type constraints. The CPDMS is defined as a composite resource service (CRS).

In order to realize a CRS such as CPDMS, the following two issues must be considered:

- The correlation among resource services in a CRS: As a candidate resource service of MfgS (manufacturing service) in Fig. 1, Manufacture_B has a discount policy, if user selects Hentong logistics as the DAS (delivery arrangement service) and Bank of China as the EPS (electronic pay service), then the service price of Manufacture_B will lower than its bid price (i.e., its default value). In addition, if the system selects Unigraphics as the design & analysis service (D&AS), its response time (i.e., manufacturing time) will be shorter than its default value. It is
**2.2. Basic models of resource service composition and their QoS evaluation**

2.2.1. QoS properties and QoS description mode supporting service correlation

We define six QoS properties of resource service in this study, they are (1) **Time** (**T**): resource service execution duration, (2) **Cost** (**C**): the cost of invoking a resource service, (3) **Reliability** (**Rel**): the possibility that the resource service can be replaced by other resource services which belong to a same resource service set, (4) **Maintainability** (**Ma**): the ability that a resource service can deal with unexpected situations, (5) **Trust-QoS** (**Trust**): the reputation of a resource service evaluated by users, (6) **Function similarity** (**FS**): the possibility that the functions of a requested resource service can be replaced by other resource services which belong to a same resource service set. The detailed definition and evaluation methods of each QoS property can be found in the authors’ previous works [15,16]. The QoS model, **QoS**, of resource service **RS** is defined as:

\[
Q_{rs} = (q_1, q_2, \ldots, q_i, \ldots, q_n),
\]

where \( K_n \) is the number of QoS properties and \( q_i \) is the ith QoS criterion. Apparently, in this paper \( K_n = 6 \) and

\[
Q_{rs} = (q_1, q_2, q_3, q_4, q_5, q_6) = (T(RS), C(RS), Rel(RS), Ma(RS), Trust(RS), FS(RS)).
\]

In this study, these QoS properties are classified into two categories: **positive QoS** and **negative QoS**. Positive QoS means that the bigger the value of the QoS property the higher the quality of the resource service, such as \( q_1, q_2, q_3, \) and \( q_4 \). While negative QoS means that the smaller the value of the QoS property the higher the quality of the resource service, such as \( q_5 \) and \( q_6 \).

It is assumed that the set of resource services that affect any QoS properties of a resource service **RS** is \( CRSS_{rs} \), and \( CRSS_{rs} \) is defined as the correlation resource service set (CRSS) of resource service **RS**. Considering the influence of correlated resource services on the value of \( q_i \) of **RS** in a CRS, this article, each QoS property, \( q_i \) of **RS** is represented as

\[
q_i = (q(i, 0) : \text{default} = q(i, 1) : \text{Correlation}(i, 1) : \ldots : \text{Correlation}(i, j) : q(i, K_i) : \text{Correlation}(i, K_i)),
\]

where \( K_i = 0, 1, 2, 3, \ldots \) is the number of correlations that affect the value of \( q_i \). \( q(i, 0) \) is the default value of \( q_i \) if it is not affected by other resource services, i.e., \( K_i = 0 \). Item \( \text{Correlation}(i, j) \) is the jth Boolean correlation statements of \( q_i \). It is used to describe the correlation regard to \( q_i \) between **RS** and other resource services, the value of \( \text{Correlation}(i, j) \) is true (i.e., 1) or false (i.e., 0). The representation of \( \text{Correlation}(i, j) \) is as follow:

**MO-MRSCOS** problem is to select one candidate resource service from each corresponding **RSS** (i.e., **RSS**), respectively, and generate all possible composite resource service executing paths (CRSEPs), then select the optimal one from \( \prod_{i=1}^{n_i} M_i \) CRSEPs to execute **Task** under multi-objective (e.g., time minimization, cost minimization, and reliability maximization) and multi-constraint (e.g., satisfy lowest maintainability, trust-QoS, and function similarity).

This paper emphasizes on addressing the above two issues.

**Fig. 1.** State charts for a composite resource service (CRS). The Complete Product Design & Manufacturing Service (CPDMS) invokes a number of other services, including D&A, AS, MPAS, MfG, DAS, and EPS.
\[
C_{\text{correlation}}() := \text{selected}_{\text{RS}} \in \text{CRS}_{\text{RS}}.
\]

Item \(\text{selected}_{\text{RS}}\) denote the selected resource service, and \(\text{CRS}_{\text{RS}}\) is the corresponding CRSS. There are three operators among \(C_{\text{correlation}}()\): `not', 'and', 'or'.

- not \(C_{\text{correlation}}()\)
- \(C_{\text{correlation}}()\) and \(C_{\text{correlation}}()\)
- \(C_{\text{correlation}}()\) or \(C_{\text{correlation}}()\)

The priority of the three operations is `not > and > or'. Based on the above model, the value of \(q^i\) is calculated as follows:

\[
q^i = \begin{cases} 
\max(\text{Bool}(i,j) + q(i,j)) & (q^i \text{ is a positive QoS}), \\
\min(\text{Bool}(i,j) + q(i,j)) & (q^i \text{ is a negative QoS and } \forall \text{ Bool}(i,j) = 1), 
\end{cases}
\]

where

- \(\text{Bool}(i,j) = \begin{cases} 
0 & (\text{if } C_{\text{correlation}}(i,j) \text{ is false}), \\
1 & (\text{if } C_{\text{correlation}}(i,j) \text{ is true}), 
\end{cases}\)

- \(C_{\text{correlation}}(i,0)\) is true if \(\sum_{j=1}^{n} \text{Bool}(i,j) = 0\).

For example, the QoS description of \(\text{Manufacture}_B\) in \(\text{Fig. 1}\) is as follows:

<table>
<thead>
<tr>
<th>QoS description of (\text{Manufacture}_B) in (\text{figure 1}).</th>
</tr>
</thead>
<tbody>
<tr>
<td>(Execution time)={700ms: default; 550ms: selected DAS (\in) {Unigraphics}; 450ms: selected DAS (\in) {Pro/Engineer})</td>
</tr>
<tr>
<td>Cost={99 USD: default, 94 USD: selected (\in) {Bank of America} or selected (\in) {Bank of China}; 87 USD: selected DAS (\in) {Henong logistics} and selected (\in) {Bank of China})</td>
</tr>
<tr>
<td>Reliability={95: default}</td>
</tr>
<tr>
<td>...</td>
</tr>
</tbody>
</table>

The above QoS model describes three QoS properties of \(\text{Manufacture}_B\): execution time\(i.e., q^1\), cost\(i.e., q^2\), and Reliability\(i.e., q^3\). The value of \(q^i\) is not influenced by other resource services. The value of \(q^1\) has three options: (a) if the selected D&AS is Unigraphics then \(q^1 = 550\) milliseconds; (b) if the selected D&AS is Pro/Engineer then \(q^1 = 450\) milliseconds; (c) otherwise, \(q^1 = 700\) milliseconds. The value of \(q^2\) has three options too: (a) if selected EPS is Bank of America or Bank of China, then \(q^2 = 94\) USD; (b) if selected DAS is Henong logistics and selected EPS is Bank of China then \(q^2 = 87\) USD; (c) otherwise, \(q^2 = 99\) USD.

2.2.2. Basic composite models of CRS and their QoS evaluation

In general, there are four basic composite modes for a CRS, they are sequence mode, parallel model, selective model and circular model, as shown in the first column in Figs. 2–5, respectively. Almost all CRS can be constructed using these four basic models. Apparently, a CRS primarily has two forms, as follows [4]:

- Sequence composite resource service (SCRS): SCRS is composed by a series of resource services, and the connecting relationship among resource services is series-wound.
- Mixed composite resource service (MCRS): In an MCRS, the connecting relationship among resource services are not only simply ordinal, but also parallel, selective, and circular.

Due to the complexity of MCRS itself, the executing paths of an MCRS are even more complex, hence it is difficult to solve MO-MRSCOS problem in MCRS comparatively. Therefore, we attempt to transfer MCRS into SCRS, and then solve the MO-MRSCOS problem in SCRS. If we can transform the above mentioned four basic composite modes into sequence mode and design relevant algorithms for calculating the corresponding aggregation QoS properties’ values, then the MO-MRSCOS problem in MCRS can be transformed into that in SCRS.

Therefore, the following four rules are constructed for transforming MCRS into SCRS and calculating the corresponding aggregation QoS properties’ values [4].

**Rule 1 (sequence mode):** a new resource service set \(\text{RSS}^\text{seq} = \{R_{1}^\text{seq}, R_{2}^\text{seq}, \ldots, R_{n}^\text{seq}\}\) is constructed to provide the functions of the sequence model (highlighted by dashed pane in Fig. 2). Addition operator is used to compute the aggregation QoS properties \(q^i(i=1,2,5,6)\) of \(R_{i}^\text{seq}\) and multiplication operator is used to compute \(q^i(i=3,4)\), because every component resource service in sequence model should be executed in sequence. It is assumed that there are \(n\) nodes, i.e., \(\{R_{1}^\text{seq}, R_{2}^\text{seq}, \ldots, R_{n}^\text{seq}\}\), in a sequence mode as shown in the first column in Fig. 2, where \(\text{RSS}^\text{seq} = \{R_{1}^\text{seq}, R_{2}^\text{seq}, \ldots, R_{n}^\text{seq}\}\). The corresponding transforming chart and calculating formulations are shown in Fig. 2.

**Rule 2 (parallel mode):** a new resource service set \(\text{RSS}^\text{comm} = \{R_{1}^\text{com}, R_{2}^\text{com}, \ldots, R_{n}^\text{com}\}\) is constructed to provide the functions of the parallel model (highlighted by dashed pane in Fig. 3). Maximum operator is used to compute \(q^i(i=1,2,6)\) of \(R_{i}^\text{com}\), addition operator is used to compute \(q^i(i=2,5,6)\), and multiplication operator is used to compute \(q^i(i=3,4)\) because every component resource service in sequence model should
### Sequence model

$$\begin{align*}
T(RS^{\text{seq}}) &= \sum_{i=1}^{n} T(RS_i) \\
C(RS^{\text{seq}}) &= \sum_{i=1}^{n} C(RS_i) \\
\text{Re}l(RS^{\text{seq}}) &= \prod_{i=1}^{n} \text{Re}l(RS_i) \\
\text{Ma}t(RS^{\text{seq}}) &= \prod_{i=1}^{n} \text{Ma}t(RS_i) \\
\text{Tr}ust(RS^{\text{seq}}) &= \sum_{i=1}^{n} \text{Tr}ust(RS_i)/n \\
FS(RS^{\text{seq}}) &= \sum_{i=1}^{n} FS(RS_i)/n
\end{align*}$$

(8)

**Fig. 2.** Transforming rule of sequence mode.

### Parallel Model

$$\begin{align*}
T(RS^{\text{com}}) &= \max \left( T(RS_1'), T(RS_2'), \ldots, T(RS_n') \right) \\
C(RS^{\text{com}}) &= \sum_{i=1}^{n} C(RS_i') \\
\text{Re}l(RS^{\text{com}}) &= \prod_{i=1}^{n} \text{Re}l(RS_i') \\
\text{Ma}t(RS^{\text{com}}) &= \prod_{i=1}^{n} \text{Ma}t(RS_i') \\
\text{Tr}ust(RS^{\text{com}}) &= \sum_{i=1}^{n} \text{Tr}ust(RS_i')/n \\
FS(RS^{\text{com}}) &= \sum_{i=1}^{n} FS(RS_i')/n
\end{align*}$$

(9)

**Fig. 3.** Transforming rule of parallel mode.

### Selective model

$$\begin{align*}
T(RS^{\text{sel}}) &= \sum_{i=1}^{n} \left( T(RS_i') \times \lambda_i \right) \\
C(RS^{\text{sel}}) &= \sum_{i=1}^{n} C(RS_i') \times \lambda_i \\
\text{Re}l(RS^{\text{sel}}) &= \sum_{i=1}^{n} \text{Re}l(RS_i') \times \lambda_i \\
\text{Ma}t(RS^{\text{sel}}) &= \sum_{i=1}^{n} \text{Ma}t(RS_i') \times \lambda_i \\
\text{Tr}ust(RS^{\text{sel}}) &= \sum_{i=1}^{n} \left( \text{Tr}ust(RS_i') \times \lambda_i \right) \\
FS(RS^{\text{sel}}) &= \sum_{i=1}^{n} \left( FS(RS_i') \times \lambda_i \right)
\end{align*}$$

Note: $\lambda_i$ is the corresponding probability that $RS_i$ been selected and $\sum_{i=1}^{n} \lambda_i = 1$

(10)

**Fig. 4.** Transforming rule of selective mode.

### Circular model

$$\begin{align*}
T(RS^{\text{cyc}}) &= k_{\text{cyc}} \times \sum_{i=1}^{n} T(RS_i) \\
C(RS^{\text{cyc}}) &= k_{\text{cyc}} \times \sum_{i=1}^{n} C(RS_i) \\
\text{Re}l(RS^{\text{cyc}}) &= k_{\text{cyc}} \times \prod_{i=1}^{n} \text{Re}l(RS_i) \\
\text{Ma}t(RS^{\text{cyc}}) &= k_{\text{cyc}} \times \prod_{i=1}^{n} \text{Ma}t(RS_i) \\
\text{Tr}ust(RS^{\text{cyc}}) &= \sum_{i=1}^{n} \left( \text{Tr}ust(RS_i') \times \lambda_i \right) \\
FS(RS^{\text{cyc}}) &= \sum_{i=1}^{n} \left( FS(RS_i') \times \lambda_i \right)
\end{align*}$$

Note: $k_{\text{cyc}} (k_{\text{cyc}} = 1,2,3,\ldots)$ is the cycle times.

(11)

**Fig. 5.** Transforming rule of circular mode.
be executed in sequence. It is assumed that there are $n$ nodes, i.e., $\{\text{RSS}^1, \text{RSS}^2, \ldots, \text{RSS}^n\}$, in a parallel mode as shown in the first column in Fig. 3, where $\text{RSS}^i = \{\text{RS}_{1}^i, \text{RS}_{2}^i, \ldots, \text{RS}_{M}^i\}$. The corresponding transforming chart and calculating formulations are shown in Fig. 3.

**Rule 3 (Selective mode):** A new resource service set $\text{RSS}_{sel} = \{\text{RS}_{1}^1, \text{RS}_{2}^2, \ldots, \text{RS}_{M}^M\}$ is constructed to provide the functions of the selective model (highlighted by dashed pane in Fig. 4). In selective mode we do not know which elements will be selected, in order to quantitatively calculate the QoS of the selective model, the corresponding probability, $\lambda_j \left( \sum_{j=1}^{M} \lambda_j = 1 \right)$, that $\text{RSS}^i$ has been selected is introduced. Addition and multiplication operators are used to compute $q^i$ of $\text{RSS}^i$. It is assumed that there are $n$ nodes, i.e., $\{\text{RSS}^1, \text{RSS}^2, \ldots, \text{RSS}^n\}$, in a selective mode as shown in the first column in Fig. 4, where $\text{RSS}^i = \{\text{RS}_{1}^i, \text{RS}_{2}^i, \ldots, \text{RS}_{M}^i\}$. The corresponding transforming chart and calculating formulations are shown in Fig. 4.

**Rule 4 (Circular mode):** A new resource service set $\text{RSS}^{QoS} = \{\text{RS}_{1}^{QoS}, \text{RS}_{2}^{QoS}, \ldots, \text{RS}_{M}^{QoS}\}$ is constructed to provide the functions of the circular model (highlighted by dashed pane in Fig. 5). The primarily difference between sequence mode and the circular model is that the elements in sequence model are only executed one time, but the elements in circular model are executed $k_{cyc}$ times. Apparently, the value of $q^1$ and $q^2$ are increasing with the executing times, while $q^i(i = 3, 4, 5, 6)$ do not change with the execution time. Therefore, addition and multiplication operators are used to compute $q^1$ and $q^2$ of $\text{RSS}^{QoS}$, multiplication operator is used to compute $q^3$ and $q^4$, and addition operator is used to compute $q^5$ and $q^6$. It is assumed that there are $n$ nodes, i.e., $\{\text{RSS}^1, \text{RSS}^2, \ldots, \text{RSS}^n\}$, in a circular mode as shown in the first column in Fig. 5, where $\text{RSS}^i = \{\text{RS}_{1}^i, \text{RS}_{2}^i, \ldots, \text{RS}_{M}^i\}$. The corresponding transforming chart and calculating formulations are shown in Fig. 5.

In above formulations of aggregation QoS properties, the value of each QoS property of each resource service is calculated by Eq. (5).

### 2.3. Formulation of MO-MRSCOS problem

Based on the transforming rules, almost all MCRS can be transformed into SCRS as shown in Fig. 6. A SCRS can be represented as $\text{CRS} = \{\text{RS}_{1}, \text{RS}_{2}, \ldots, \text{RS}_{M}\}$, where $\text{RS}_{i}$ is a composite path among $\{\text{RS}_{1}, \text{RS}_{2}, \ldots, \text{RS}_{M}\}$, i.e., $\text{RS}_{i} = \{\text{RSS}^i, \text{RSS}^j, \ldots, \text{RSS}^n\}$, where $\text{RSS}^i \in \text{RSS}^i$. The QoS model of ith ($i = 1, 2, \ldots, \prod_{i=1}^{M_n} M_i$) composite path (hereafter denoted as $p_i$) of CRS is denoted as $Q(p_i) = \{T(p_i), C(p_i), \text{Rel}(p_i), \text{Ma}(p_i), \text{Trust}(p_i), \text{FS}(p_i)\}$, and $Q(p_i)$ is calculated as follows:

$$\begin{align*}
T(p_i) &= \sum_{j=1}^{N} T(RS_j^i), \\
C(p_i) &= \sum_{j=1}^{N} C(RS_j^i), \\
\text{Rel}(p_i) &= \prod_{j=1}^{N} \text{Rel}(RS_j^i), \\
\text{Ma}(p_i) &= \prod_{j=1}^{N} \text{Ma}(RS_j^i), \\
\text{Trust}(p_i) &= \sum_{j=1}^{N} \text{Trust}(RS_j^i)/N, \\
\text{FS}(p_i) &= \sum_{j=1}^{N} \text{FS}(RS_j^i)/N.
\end{align*}$$

(Appendixa)

![Fig. 6. Sequence composite resource service (SCRS) process (a).](image-url)
where \( i = 1, 2, \ldots, |M| \); \( Ma_i \), \( Trust_i \), and \( FS_i \) are the lowest requirements of maintainability, trust-QoS, and function similarity, respectively; \( Ma(p_{ij}) \), \( Trust(p_{ij}) \), and \( FS(p_{ij}) \) are the values of maintainability, trust-QoS, and function similarity of the \( j \)th resource service (i.e., \( RS_i^j \)) invoked in the \( i \)th CRSEP (i.e., \( \{ RS_1^j, RS_2^j, \ldots, RS_K^j \} \)), respectively. Constraints (16)–(18) corresponding to the maintainability, trust-QoS and function similarity should be no less than user’ requirements.

Many studies have been devoted to the subject of multi-objective optimization (MOO), and the proposed methods to address MOO can be classified into three different types [17]:

1. A utility function or weighted function is employed to aggregate multiple objectives into a single one, so a MOO problem can be transformed into mono-objective optimization problem consists of combining the different objectives into a weighted sum. Methods in this class are based on utility functions or E-Constraint and goal programming.

2. The non-Pareto approach utilizes operators for processing the different objectives in a separated way. It refers to optimize one objective at a time while imposing constrains on the others. Because it is difficult to determine the order of all the objectives to be optimized, hence, this kind of approach is seldom used.

3. Pareto-based approach. It is directly based on the Pareto optimality concept. In this approach, a vector containing all objective values represents the solution fitness, and the concept of Pareto dominance is used to establish preference between solutions. It aims at satisfying two goals: converging towards the Pareto front while obtaining diversified solutions scattered all over the Pareto front.

In this paper, the third approach is selected to resolve MO-MRSCOS using PSO.

2.4. Pareto solution of MO-MRSCOS

**Definition 1. Pareto optimal:** Let \( p_i^* \) and \( p_j^* \) be two feasible solutions to MO-MRSCOS, and both \( p_i^* \) and \( p_j^* \) satisfy Constraints (16)–(18).

Solution \( p_i^* \) is said to dominate \( p_j^* \) if and only if the following statement is true

\[
\left( T(p_i^*) \leq T(p_j^*) \text{ and } C(p_i^*) \leq C(p_j^*) \text{ and } \text{Rel}(p_i^*) \geq \text{Rel}(p_j^*) \right) \text{ or } \left( T(p_i^*) < T(p_j^*) \text{ or } C(p_i^*) < C(p_j^*) \text{ or } \text{Rel}(p_i^*) > \text{Rel}(p_j^*) \right) .
\]

The above relationship is represented as \( p_i^* > p_j^* \), and \( p_i^* \) is defined as Pareto optimal solution (or non-dominated solution) if there is no such solution \( p_j^* \) satisfies \( p_i^* > p_j^* \). The whole of all Pareto optimal objectives vectors is called the Pareto front and the set of all the Pareto optimal decision vectors is called the Pareto optimal set.

For an MO-MRSCOS problem, the system may search out a large number of candidate solutions, (a) how to distinguish and classify the Pareto optimal solution, i.e., classification of Pareto front, and (b) how to compare different Pareto optimal solutions and select the best Pareto optimal solution, i.e., comparison of Pareto optimal solution, are the key to use Pareto-based approach in PSO.

(a) Classification of Pareto front

Let population \( P \) be a set of candidate solutions. In order to realize the classification of Pareto front, the solution \( P \) is divided into \( K \) (\( K = 1, 2, \ldots \)) subpopulations, \( SP_1, SP_2, \ldots, SP_K \), according to the dominant relationship among solutions. Any solution in set \( SP_i \) is dominated by the solutions in set \( SP_j \) if \( i < j \), where the solutions in the same set are not dominated to each other. That, all solutions in the populations are classified into \( K \) Pareto fronts, where \( SP_1 \) contains all the Pareto-optimal solutions in the current population. According to [18], the implementing algorithms of classification of Pareto front are as follows:

<table>
<thead>
<tr>
<th>NonDominateClassify(P)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong></td>
</tr>
</tbody>
</table>
| **Step 2** | For each solution \( p_j \in P \) (\( j \neq i \))
| | If \( p_i \succ p_j \), then let \( \psi_{p_i} = \psi_{p_i} \cup \{ p_j \} \)
| | Else If \( p_j \succ p_i \), then let \( N_{p_i} = N_{p_i} + 1 \) |
| **Step 3** | If \( N_{p_i} = 0 \), then let \( SP_1 = SP_1 \cup \{ p_i \} \) |
| **Step 4** | Set \( k = 1 \) |
| **Step 5** | While \( SP_k \neq \emptyset \), set \( \theta = \emptyset \) |
| **Step 6** | For each \( p_i \in SP_k \)
| | **Step 7** | For each \( p_j \in \psi_{p_i} \), \( N_{p_j} = N_{p_j} - 1 \),
| | | If \( N_{p_j} = 0 \) then \( \theta = \theta \cup \{ p_j \} \) |
| **Step 8** | \( k = k + 1 \), \( SP_k = \emptyset \) |
| **Step 9** | Output \( SP_k \) |
(b) Comparison of Pareto solutions

After the above classification, a set of candidate solutions \( P \) can be represented as \( P = \{ SP_1, SP_2, \ldots, SP_K \} \), where any solution in set \( SP_i \) is dominated by the solutions in set \( SP_j \) \((i < j \text{ and } i, j = 1, 2, \ldots, K)\). Apparently, the distribution of arbitrary two different solutions, e.g., \( p_i \) and \( p_j \), have two conditions: (a) the two solutions belong to two different Pareto fronts, and (b) the two solutions belong to a same Pareto front. In this article, if two solutions with differing non-domination ranks (i.e., belong to different Pareto fronts), the system prefers the solution with the lower rank, e.g., \( \forall p_i \in SP_i \) and \( \forall p_j \in SP_j \) if \( i < j \), the system prefers \( p_i \). Otherwise, if both solutions belong to the same Pareto front, then the system prefers solution that with bigger crowding distance according to [18]. Let \( d(p) \) be the crowding distance of solution \( p \) in \( SP_k \) \((k = 1, 2, 3, \ldots, K)\), and \( d(p_i) \) is calculated as follows:

\[
\text{CrowdDistanceEvaluate}(P)
\]

Step 1) Perform NonDominatedClassify(\( P \)) and generate \( P = \{ SP_1, SP_2, \ldots, SP_K \} \)

Step 2) Set \( k = 1 \)

Step 3) While \( SP_k \neq \emptyset \)

Step 4) For each \( p_i \in SP_k \) \(= \{ p_1, p_2, \ldots, p_{N_i} \} \), set \( d(p_i) = 0 \)

Step 5) Set \( j = 1 \)

Step 6) Sort \( \uparrow (SP_k, j) \) // Order the solutions in \( SP_k \) according to the \( j \)th objective values in ascending order.

Step 7) Set \( f_j^{max} = \max\{ f_j(p_i), \forall p_i \in SP_k, k = 2, 3, \ldots, N_i - 1 \} \) and \( f_j^{min} = \min\{ f_j(p_i), \forall p_i \in SP_k, k = 2, 3, \ldots, N_i - 1 \} \)

// \( f_j(p_i) \) be the \( j \)th objective value of \( p_i \)

Step 8) Set \( d(p_i) = d(p_{N_i}) = +\infty \)

Step 9) For \( i = 2 \) to \( (N_i - 1) \), \( d(p_i) = d(p_i) + (f_j(p_{i+1}) - f_j(p_{i-1}))/f_j^{max} - f_j^{min} \)

Step 10) If \( j = 3 \), then go to Step 11; Otherwise \( j = j + 1 \) and go to Step 6

Step 11) Output \( d(p_i) \)

Step 12) \( k = k + 1 \)

Based on above idea, a comparison operator, \( >_c \), between two Pareto optimal solutions is defined as follows: For a set of candidate solutions \( P = \{ SP_1, SP_2, \ldots, SP_K \} \), where any solution in set \( SP_i \) is dominated by the solutions in set \( SP_j \) \((i < j \text{ and } i, j = 1, 2, \ldots, K)\)

\[
(1) \quad \forall p_i \in SP_i \text{ and } \forall p_j \in SP_j \Rightarrow p_j >_c p_i,
\]

\[
(2) \quad \forall p_i \in SP_i \text{ and } \forall p_j \in SP_j, \quad \text{if } d(p_i) > d(p_j) \Rightarrow p_j >_c p_i,
\]

where \( p_i >_c p_j \) means \( p_i \) is better than \( p_j \), and the system prefers \( p_i \). Operator \( >_c \) will be used for the generation of local best solution (i.e., \( \text{gbest} \)) and global best solution (i.e., \( \text{pbest} \)) in PSO for MO-MRSCOS in the following section.

3. PSO for MO-MRSCOS

3.1. Review of standard PSO

PSO is a population-based stochastic optimization technique proposed by Kennedy et al. [19,20]. This technique has good performance, low computational cost and easy implementation. Hence, PSO has attracted significant attentions from researchers around the world since its introduction in 1995, and many successful applications of PSO, ranging from evolving weights form artificial neural networks [21], manufacture and milling [22], reactive power and voltage control [23], to image registration [24], have been reported.

The standard PSO proceeds as follows: Given an optimization function \( f(X) \) with \( N \)-dimension \((N = 1, 2, 3, \ldots)\) real-valued decision variables, the PSO searches the optimal solution by iteratively evolving a swarm of candidate solutions, called particles. The initial swarm is generated randomly. Let each particle be represented by \( P_i = (x_{i1}, x_{i2}, \ldots, x_{iN}) \), and the quality of particle \( P_i \) is evaluated by the fitness function that needs to be optimized. Each particle \( P_i \) at \( t \)th iteration has two attributes that characterize its status on the \( N \)-dimension search space: the \( j \)-dimension position \((j = 1, 2, 3, \ldots)\) and \( j \)-dimension velocity, \( v_{ij}(t) \). In each search-iteration, the current position and velocity of each particle are updated by two best values: (a) \( \text{pbest} \), individual best position (best solution or fitness) that has achieved so far, which is called \( \text{pbest} \); and (b) \( \text{gbest} \), is another “best” value that is tracked by the particle swarm optimizer, representing the best position among all particles in the population or swarm. The current position and velocity of each particle \( P_i \) are updated as follows [19]:

\[
\begin{align*}
\nu_{ij}(t) &= \omega \nu_{ij}(t-1) + c_1 r_1 (\text{pbest}_i(t-1) - x_{ij}(t-1)) + c_2 r_2 ((\text{gbest}(t-1) - x_{ij}(t-1)),
\end{align*}
\]

\[
\begin{align*}
x_{ij}(t) &= x_{ij}(t-1) + \nu_{ij}(t).
\end{align*}
\]
The velocity, \( v_i(t) \), represents the distance to be traveled by \( P_i \) from its current position. Position, \( x_i(t) \), represents \( P_i \)’s position, \( pbest \) represents its best previous position, and \( gbest \) represents the best position among all particles in the population. Terms \( r_1 \) and \( r_2 \) are random real numbers drawn from \( U(0,1) \). Parameters \( c_1 \) and \( c_2 \) are positive constants parameters called acceleration coefficients. Parameter \( \omega \) is the inertia weight, a user-specified parameter that controls, together with \( c_1 \) and \( c_2 \), the impact of previous historical values of particle velocities on its current one.

The procedure of PSO is as follows, interested readers may refer to Kennedy et al. [19,20] for details.

1. Initialize a population of particles with random positions and velocities in the search space.
2. Update the velocity and the position of each particle according to Eqs. (22) and (23).
3. Map the position of each particle into solution space and evaluate its fitness value according to the desired optimization fitness function. At the same time, update \( gbest \) and \( pbest \).
4. If the stopping criterion is met, terminate search; else go to (2).

3.2. PSO for MO-MRSCOS problem

Our creative work for designing the PSO for MO-MRSCOS can be summarized as follows:

- Based on permutation representation, PSO is applied to perform exploration in binary hyperspace.
- A non-dominated sorting technique is applied to get the personal best position (\( pbest \)) and global best (\( gbest \)) position in PSO.
- The parameters of particle updating formulation in PSO are dynamical generated in order to make a compromise between the global exploration and local exploitation abilities of PSO.
- To maintain diversity of solutions in population, permutation-based and objective-based population trimming operators are applied in PSO.

Next, we will describe PSO for MO-MRSCOS in detail, including representation, particle movement, generation of \( pbest \) and \( gbest \), and population trimming. Then the whole procedure of PSO for MO-MRSCOS problem is presented.

3.2.1. Representation

One of the key issues in using a successful PSO algorithm is the representation, i.e. finding a suitable mapping between problem solution and PSO particle. For the considered MO-MRSCOS, a search space of \( N \)-dimension is set up for a task which can be decomposed into \( N \) sub-tasks. Therefore, the \( i \)th particle can be represented as \( p_i = (x_{i1}, x_{i2}, \ldots, x_{iN}) \), where \( x_{ij} \) indicates the index of the allocated resource service for the \( j \)th sub-task, \( S_j \), of a task. Each dimension has a discrete set of possible limit values, \( \gamma = \{x_{ij}; 1 \leq x_{ij} \leq M_j \} \). \( M_j \) is the number of candidate resource services in \( RSS_j \). Fig. 7 shows a mapping between a possible composition instance and a particle position coordinates in PSO. The PSO swarm consists of \( M \) particles and the initial swarms generated randomly. Hence, the PSO population is represented as a \( M \times N \)-dimensional array consisting of \( M \) particles.

3.2.2. Particle movement (update)

As described in Section 3.1, swarm intelligence is enriched by tallying the best experiences observed by individual particles and the entire swarm during the evolution. In particular, each particle remembers the best vector with the highest fitness it has visited so far, i.e., \( pbest \), and the best vector visited by its neighbors, i.e., \( gbest \). At each evolutionary iteration \( t \), particle \( p_i \) modifies its velocity \( v_i(t) \) and position \( x_i(t) \) through each dimension \( j \) by referring to its personal best experience, \( pbest_i(t) \), and the swarm’s best experience, \( gbest(t) \), using Eqs. (24) and (25) [19,20]:

\[
\begin{align*}
v_{ij}(t) &= \omega(t)v_{ij}(t-1) + c_1(t)r_1(t)(pbest_{ij}(t-1) - x_{ij}(t-1)) + c_2(t)r_2(t)(gbest(t-1) - x_{ij}(t-1)), \\
x_{ij}(t) &= x_{ij}(t-1) + v_{ij}(t).
\end{align*}
\] (24)

In Eq. (24), \( c_1(t) \) and \( c_2(t) \) are positive constant parameters called acceleration coefficients, which control the maximum step size a particle can do. Terms \( r_1(t) \) and \( r_2(t) \) are two random functions drawn from \( (0,1) \). Inertia weight, \( \omega(t) \), is a user-specified parameters that controls, together with \( c_1(t) \) and \( c_2(t) \), the impact of previous historical values, personal best position, and global best position of particle velocities on its current one. Suitable selection of the \( \omega(t) \), \( c_1(t) \) and \( c_2(t) \) provides a balance between the global and the local searches. The setting of the parameters in Eq. (24) is discussed as follows [4].

(a) Selection of inertia weight

Generally, more exploration is needed in the initial stages when the algorithm has very little knowledge about the search space. In contrast, more exploitation is needed during the later stages so that the algorithm is able to exploit the information it has gained so far, and make better global exploration of the search space. Hence, the value of \( \omega(t) \) at \( t \)th iteration should be allowed to decrease linearly. In this paper, \( \omega(t) \) is calculated as [25,26]

\[
\omega(t) = (\omega(1) - \omega(N_{exe})) \times (N_{exe} - t)/N_{exe} + \omega(N_{exe}).
\] (26)

where \( N_{exe} \) is the maximum number of iterations, \( t \) is the current iteration number, \( \omega(1) \) is the inertia weight at \( N_{exe} \)th iteration (i.e., the minimum value) and \( \omega(1) \) is that at the first iteration (i.e., the maximum value), and \( \omega(t) \in [\omega(1),\omega(N_{exe})] \).

(b) Selection of acceleration coefficients

The values of \( c_1(t) \) and \( c_2(t) \) at \( t \)th iteration, are calculated according to Eqs. (27) and (28) [25,26]:
\[ c_1(t) = (c_1(N_{\text{exe}}) - c_1(1)) \times \frac{t}{N_{\text{exe}}} + c_1(1), \]  
\[ c_2(t) = (c_2(N_{\text{exe}}) - c_2(1)) \times \frac{t}{N_{\text{exe}}} + c_2(1), \]  
where \( c_1(N_{\text{exe}}) \) is the acceleration coefficients at \( N_{\text{exe}} \)th iteration (i.e., the minimum value of \( c_1(t) \)) and \( c_1(1) \) is that at the first iteration (i.e., the maximum of \( c_1(t) \)). They are set by user, and \( c_1(t) \in [c_1(N_{\text{exe}}), c_1(1)] \). By contrast, \( c_2(N_{\text{exe}}) \) is the acceleration coefficients at \( N_{\text{exe}} \)th iteration (i.e., the maximum value of \( c_2(t) \)) and \( c_2(1) \) is that at the first iteration (i.e., the minimum of \( c_2(t) \)), respectively, and \( c_2(t) \in [c_2(N_{\text{exe}}), c_2(1)] \).

(c) Biggest personal position value

Because the PSO particle position, \( x_{ji} \), represents a series of number that varies in \([1, M_j]\), all parameters of the \( j \)-dimensional particle positions, either initialized or updated during search, must be limited to \([1, M_j]\). Hence, to avoid infeasible particle positions that can lead to slow PSO search, each parameter of the initialized or updated position that is beyond \([1, M_j]\) can be adjusted according to Eqs. (29) and (30):

\[
\text{If } x_{ji}(t) > M_j, \quad \text{then } x_{ji}(t) = M_j. \tag{29}
\]
\[
\text{Else if } x_{ji}(t) < 1, \quad \text{then } x_{ji}(t) = 1. \tag{30}
\]

(d) Biggest velocity value

Oscillating of the particle velocity, \( v_{ji} \), may cause particles to fly outside feasible search space \([1, M_j]\) or \([x_{ij}^{\text{min}}, x_{ij}^{\text{max}}]\) when using Eq. (24). Similarly, the particle velocity based on the current position should also be limited to prevent the updated position from oscillating too heavily. Apparently, the maximum velocity, \( v_{ij}^{\text{max}} \), equals \( x_{ij}^{\text{max}} \) at most. Therefore, the velocity of an updated particle should be limited to \([-v_{ij}^{\text{max}}, v_{ij}^{\text{max}}]\) or \([-M_j, M_j]\). In this paper, each parameter of the \( N \)-dimension particle velocity that is beyond \([-v_{ij}^{\text{max}}, v_{ij}^{\text{max}}]\) is adjusted according to Eqs. (31) and (32):

\[
\text{If } v_{ij}(t) > v_{ij}^{\text{max}}, \quad \text{then } v_{ij}(t) = v_{ij}^{\text{max}}. \tag{31}
\]
\[
\text{Else if } v_{ij} < -v_{ij}^{\text{max}}, \quad \text{then } v_{ij}(t) = -v_{ij}^{\text{max}}. \tag{32}
\]

3.2.3. Generation of personal best (pbest)

The term \( \text{pbest}_i(t) \) is the best performance attained so far by particle \( p_i \) in its flight. In the proposed PSO, the current position of \( p_i(t) \) is compared with the \( \text{pbest}_i(t) \) using the comparison operator \( >_C \) defined in Section 2.4, and it replaces the latter if it dominates that solution. The generation of \( \text{pbest}_i(t) \) for particle \( p_i \) at \( t \)th iteration is as follows:
3.2.4. Generation of global best (gbest)

The term gbest represents the best solution obtained by the swarm. Often the conflicting nature of the multiple objectives involved multiple-objective optimization problems make the choice of a single optimum solution difficult. To resolve this problem, the concept of non-dominance is employed. And an external archive, \( A_t \), of non-dominated solutions (i.e., Pareto to optimal solutions) produced in the searching process of PSO is maintained, from which a solution is picked up as the gbest. In order to enhance the efficiency for selecting the best solutions from the archive, a maximum size, \( N_A(N_A = 1, 2, 3, \ldots) \), is specified for the external archive. In each iteration, the member of \( A_{t+1} \) is produced by combination of \( A_t \) and searched solutions \( P_t \). The new non-dominated solutions become a member of the archive when one of the following conditions is met [27, 28]:

1. The new solution dominates some members of the archive.
2. The current archive size is less than \( N_A \).
3. The current archive size is equal \( N_A \) and the new solution has higher density value that at least one member of the archive.

Based on above idea, the generation algorithms of \( A_{t+1} \) are as follows:

\[
\begin{align*}
\text{ArchiveUpdating}(t, A(t), N_A, P(t+1)) & \\
\text{Step 1) Let} & \quad k = 1 \\
\text{Step 2) Perform NonDominateClassify}(P(t+1)) & \quad \text{and get} \quad SP_t(t) = \{p_1, \ldots, p_{K_t}\} \\
& \quad /\text{find all non-dominated solutions in } P(t+1), \\
& \quad /\text{K_t is the number of non-dominated solutions} \\
\text{Step 3) Set} \quad A_{t+1} = A_t = \{p_1, \ldots, p_{K_t}\} & \quad /\text{N_A is the current number of members in } A_{t+1} \\
\text{Step 4) Set} & \quad i = 1 \\
\text{Step 5) For} \quad p_i \in SP_t(t), & \quad \text{set} \quad N_i = N_j = N_A = 0 \\
\text{Step 6) Set} & \quad j = 1 \\
\text{Step 7) For} \quad p_j \in A_{t+1}, & \\
& \quad \text{If} \quad p_j \succ p_i, \quad \text{then} \quad \psi = \psi \cup \{p_j\}, \quad N_i = N_i + 1; \\
& \quad \quad \text{If} \quad p_j \nless p_i, \quad \text{then} \quad N_j = N_j + 1; \\
& \quad \quad \text{Else} \quad N_i = N_i + 1; \\
\text{Step 8) If} \quad j = N_A, & \quad \text{then go to Step 9); otherwise} \quad j = j + 1 \quad \text{and go to Step 7) \\
\text{Step 9) If} \quad N_i = N_A, & \quad \text{then delete all members in } \psi \text{ from } A_{t+1} \quad \text{and then } \quad A_{t+1} = A_{t+1} \cup \{p_i\}, \\
& \quad \quad \text{N_A} = N_A - N_i + 1 \\
\text{Step 10) If} \quad N_A = 0 & \quad /\text{no members in } A_{t+1} \text{ is dominated by } p_i \\
& \quad \quad \text{If} \quad N_a < N_A, \quad \text{then} \quad A_{t+1} = A_{t+1} \cup \{p_i\} \quad \text{and} \quad N_a = N_a + 1; \\
& \quad \quad \quad /\text{N_a is the maximum size of archive} \\
& \quad \quad \text{If} \quad N_a = N_A, \quad \text{then} \quad A_{t+1} = A_{t+1} \cup \{p_i\} \quad \text{and delete a solution with the minimum crowding distance from } A_{t+1} \\
\text{Step 11) If} \quad i = K_t, & \quad \text{then go to Step 12) \quad \text{Otherwise} \quad i = i + 1 \quad \text{and go to Step 5) \\
\text{Step 12) Output} & \quad A_{t+1} \\
\end{align*}
\]
After \( A_{t+1} \) is generated, then the global best solution, \( gbest(t) \), is selected from \( A_{t+1} \) using the comparing operator in Section 2.4. The generation of \( gbest(t) \) for particle at \( t \)th iteration is as follows:

<table>
<thead>
<tr>
<th>Generation of ( gbest(t) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \text{generate} _ gbest(t, A_{t}) )</td>
</tr>
<tr>
<td>{</td>
</tr>
<tr>
<td>\text{Step 1) Perform} NonDomiinateClassify( (A_{t}) ) \quad //find all non-dominated solutions in ( A_{t} )</td>
</tr>
<tr>
<td>\text{Step 2) Perform} CrowdDistanceEvaluate( (A_{t}) ) \quad \text{and get the crowding distance of each member in} \ A_{t} \</td>
</tr>
<tr>
<td>\text{Step 3) For each solution} \ p_{i} \in A_{t}, \ \text{set} \ N_{p_{i}} = 0 \ \text{and} \ \psi_{p_{i}} = \emptyset</td>
</tr>
<tr>
<td>\text{Step 4) Set} \ i = 1</td>
</tr>
<tr>
<td>\text{Step 5) Let} \ j = i + 1</td>
</tr>
<tr>
<td>\text{Step 6) For solution} \ p_{j} \in A_{t}</td>
</tr>
<tr>
<td>\quad \begin{align*}</td>
</tr>
<tr>
<td>\quad \begin{align*} &amp;\text{If} (p_{j} \succ p_{i}) \quad \text{then let} \ \psi_{p_{i}} = \psi_{p_{i}} \cup {p_{j}} \quad \text{Else} \quad \text{If} (p_{j} \succ p_{i}) \quad \text{then let} \ N_{p_{i}} = N_{p_{i}} + 1 \end{align*}</td>
</tr>
<tr>
<td>\text{Step 7) If} \ j = N_{p_{i}}, \ \text{then go to} \ \text{Step 8); Otherwise} \ j = j + 1 \ \text{and go to} \ \text{Step 6)</td>
</tr>
<tr>
<td>\text{Step 8) If} \ i = N_{p_{i}} - 1 \ \text{, then go to} \ \text{Step 9); Otherwise} \ i = i + 1 \ \text{and go to} \ \text{Step 5)</td>
</tr>
<tr>
<td>\text{Step 9) Set} \ \psi = \emptyset</td>
</tr>
<tr>
<td>\text{Step 10) For each solution} \ p_{j} \in A_{i}, \ \text{if} \ N_{p_{j}} = 0 \ \text{, then let} \ \psi = \psi \cup {p_{j}}</td>
</tr>
<tr>
<td>\text{Step 11) Select one solution randomly from} \ \psi \ \text{as the} \ gbest(t)</td>
</tr>
<tr>
<td>\text{Step 12) Return} \ gbest(t)</td>
</tr>
<tr>
<td>}</td>
</tr>
</tbody>
</table>

3.2.5. Population trimming in PSO
For MO-MRSCOS problem, PSO algorithms may obtain a population with some same resource service permutations, and the objectives values of different population may be the same. Therefore, two trimming operators, i.e., permutation-based and objective-based trimming, are employed to remove some redundant solutions so as to enrich the diversity of the population.

(a) Permutation-based trimming operators (\( PBTrim() \)): If there are some populations with the same resource service permutations, the system deletes the redundant solutions and only one resource service permutation is reserved. The deleted permutations will be replaced by some random permutations in the population. After the trimming, all the permutations in the population are different.

(b) Objective-based trimming operators (\( OBTrim() \)): In PSO, after evaluating the quality of the solutions of the population, if there are some solutions with the same objectives values, then the system deletes the redundant solutions and only one is reserved. Those deleted solutions are replaced by some random permutations which have at least one objective value different from all the reserved in the population.

After the above two operations, the diversity of the population is enriched largely and effectively.

3.2.6. PSO algorithm for MO-MRSCOS problem
The procedures of PSO algorithm for solving MO-MRSCOS problem are presented in Fig. 8, where every key elements and functions have been explained in detail as before.

4. Performance analyse and discussion
These experiments test the effectiveness of the proposed approach to find the Pareto resolutions to MO-MRSCOS problems. In these experiments, we use the following two approaches to generate the final Pareto resolutions to a same MO-MRSCOS problem and compare their results.

(1) The first approach is that the correlation among resource services are not taken into account when calculating the aggregation QoS value of a CRS or a solution to MO-MRSCOS problem. It means that each QoS criterion value of the selected resource service in a solution is its default value. Then the proposed PSO method is used to generate the final Pareto solutions set. This approach is denoted as \( \text{NoCorrelation} \) for the sake of convenience in this section.

(2) The second approach considers the correlation among resource services during resource service composition. It means that each QoS criterion value of the selected resource service in a solution is calculated according to the Eq. (5) in Section 2.2.1. Similarly, this approach is denoted as \( \text{ProposedMethod} \) for in this section.
PSO algorithms for MO-MRSCOS

(1) Initialize \( t = 0 \)

Step 1) Randomly generate an initial population \( P(t) = \{ p_1, p_2, \cdots, p_N \} \), where \( N \) denotes the size of the population.
Step 2) Initialize the velocity of each particle \( v_i(t) \)
Step 3) Evaluate each particle in \( P(t) \), perform \( PBTrim(P(t)) \) and \( OBTrim(P(t)) \).
Step 4) Perform \( NonDominatedClassify(P(t)) \), select out the non-dominated solutions and store them in \( A \).
Step 5) Initialize the personal best position of each particle \( p_i^* \).

For \( i = 1 \) to \( N \), \( pbest(t) = p_i^* \).

(2) Repeat \( \{ t = 1 \) to \( N_{exe} \} \)

Step 6) For \( i = 1 \) to \( N \), \( pbest(t) = generate\_pbest(t, p_i^*, pbest(t-1)) \)
// generate the personal best position \( pbest(t) \) for each particle \( p_i^* \) in \( P(t) \)
Step 7) Perform \( generate\_gbest(t, A(t-1)) \) and generate the global best position \( gbest(t) \)
Step 8) Updating \( 60(t), c_1(t), c_2(t) \) according to Eq.(26), Eq.(27), and Eq.(28), respectively.
Step 9) Computing the velocity of particle \( p_i^* \) using Eq.(24) and use Eq.(31), Eq.(32), to adjust the velocity of \( p_i^* \) in order to maintain the particles with the search space in case they go beyond their boundaries.
Step 10) Generating new position of particle \( p_i^* \) using Eq.(25) and use Eq.(29), Eq.(30), to adjust the position of \( p_i^* \) in order to maintain the particles with the search space in case they go beyond their boundaries. Generate \( P(t+1) = \{ p_1^{*^t}, p_2^{*^t}, \cdots, p_N^{*^t} \} \).
Step 11) Evaluate each particle in \( P(t+1) \), perform trimming operators \( PBTrim(P(t+1)) \) and \( OBTrim(P(t+1)) \).
Step 12) Perform \( Archive\_Updating(t, A(t), P(t+1)) \) and produce \( A_{t+1} \).
(3) Output
Step 13) Output \( A_{t+1} \) /return the Pareto optimal font

Fig. 8. Proposed PSO algorithms for MO-MRSCOS problem.

The average fitness of the obtained Pareto solution set is selected as the criterion to test and compare the effectiveness of \( NoCorrelation \) and \( ProposedMethod \). The fitness of a solution \( p_i \), \( fitness(p_i) \), in this article is defined as follows [4]:

\[
fitness(p_i) = (Z(p_i) + C_{constraint}(p_i))^{-1},
\]

(33)

where

\[
Z(p_i) = \alpha T(p_i) + \beta C(p_i) + \epsilon R(p_i),
\]

(34)

\[
C_{constraint}(p_i) = \left( \lambda_1 \times \sum_{j=1}^{N} \left( \max \left( 0, \frac{M_j - Min(p_i^j)}{Max} \right) \right) + \right) \left( \lambda_2 \times \sum_{j=1}^{N} \left( \max \left( 0, \frac{Trust - Trust(p_i^j)}{Trust} \right) \right) + \right) \left( \lambda_3 \times \sum_{j=1}^{N} \left( \max \left( 0, \frac{FS - FS(p_i^j)}{FS} \right) \right) \right). 
\]

(35)

In above expression, \( j = 1, 2, 3, \ldots, N \) and \( i = 1, 2, \ldots, N_{exe} \). \( Z(p_i) \) is the original minimization objective function of the MO-MRSCOS problem. It combines the three optimization objective (time minimization, cost minimization, and reliability maximization) into a single function by three scaling factors, \( \alpha, \beta, \epsilon \). The factors \( \alpha, \beta, \epsilon \) are used as the corresponding weighted parameters to control the relative significance of each objective functions. They normalize the values of \( T(p_i), C(p_i) \) and \( R(p_i) \) to comparable ranges such that \( Z(p_i) \) will not be dominated by a single objective. \( C_{constraint}(p_i) \) is the penalty function to estimate the infeasibility of \( p_i \). It is the quantified amount of mismatch if \( p_i \) is infeasible, otherwise \( C_{constraint}(p_i) \) is set to 0. The term \( \sum_{j=1}^{N} \left( \max \left( 0, \frac{M_j - Min(p_i^j)}{Max} \right) \right) \) calculates the total insufficiency amounts of the invoked resource services in \( p_i \) against the lowest requirement of \( maintainability \) if \( p_i \) is infeasible. Otherwise it returns zero. Thus, the penalty function gives proportional amount of penalties on the infeasible solutions whose \( maintainability \) is under the user's lowest \( maintainability \) requirement. Similarly, \( \sum_{j=1}^{N} \left( \max \left( 0, \frac{Trust - Trust(p_i^j)}{Trust} \right) \right) \) and \( \sum_{j=1}^{N} \left( \max \left( 0, \frac{FS - FS(p_i^j)}{FS} \right) \right) \) calculate the total insufficiency amounts of \( p_i \)’s \( trust-QoS \) and \( function similarity \), respectively. \( \lambda_1, \lambda_2, \lambda_3 \) are the weights controlling the relative significance constraints, respectively.

The main parameters in proposed PSO algorithms are set as follows: \( N_{exe} = 30 \), \( \alpha(1) = 0.7 \) and \( \alpha(N_{exe}) = 0.4 \) in Eq. (26) as suggested in [29]. \( c_1(1) = 2.5, c_1(N_{exe}) = 0.5 \) in Eq. (27) and \( c_2(1) = 0.5, c_2(N_{exe}) = 2.5 \) in Eq. (28) as suggested in [29]. \( \alpha = 0.001, \beta = 0.01 \) and \( \epsilon = 0.25 \) in Eq. (34). \( \lambda_1 = \lambda_2 = \lambda_3 = 1 \) in Eq. (35).

During the implementations, two conditions are considered:

(1) The percent of correlation resource service is set to be 40%, and the number of candidate services of each service varies from 50 to 450 with an increment of 50.
The number of candidate services of each service is fixed at 250, and the percent of correlation resource service varies changes from 0 to 100% with an increment of 20%.

Condition (I) has nine tests and condition (II) has six tests, respectively. Each test is executed twenty times. The results of each test is the average of the twenty executions, as shown in Fig. 9. It can be concluded from Fig. 9 that, ProposedMethod has better performance than NoCorrelation in the tests both under condition (I) and condition (II).

In condition (I), as the increment of the number of candidate resource services, the QoS of CRS obtained by ProposedMethod and NoCorrelation both are increasing. The reason is that, with the increment of the number of candidate resource services, there are more candidate composition paths, which result in the improvement of QoS of CRS. But the result produced using ProposedMethod is better than that using NoCorrelation. That’s because ProposedMethod considers the correlation among resource service, and the QoS of the selected resource service in a CRS is better than it’s declared (i.e., its default value).

In condition (II), when the number of candidate services of each service is fixed at 250, and the percent of correlation resource service varies from 0% to 100% with an increment of 20%, the QoS of CRS obtained by ProposedMethod is improved. That’s because as the increments of the percent of correlation resource service, more resource services’ quality is affected by other resource services in a CRS, and the actual QoS of each selected resource service is better than its default value, and the whole QoS of the CRS is improved. However, because NoCorrelation does not consider the correlation among resource service, and the QoS of the selected resource service are the default value, which is not changed as the incensement of correlation resource services. Hence, as the increasing of percent of correlation resource services, the QoS of the CRS almost have no changing.

5. Conclusion and future works

In this paper, the correlations among resource services are taken into account during MGrid resource service composition, and a QoS description mode supporting resource service correlation is presented. Four basic composite modes for composite resource service and their aggregation QoS calculating methods are proposed. The problem of MO-MRSCOS is proposed and formulized. A particle swarm optimization (PSO) is proposed for MO-MRSCOS problem. Unlike previous works: (a) the proposed PSO algorithms combine the non-dominated sorting technique to achieve the selection of global best position and private best position; (b) the parameters of particle updating formulation in PSO are dynamical generated in order to make a compromise between the global exploration and local exploitation abilities of PSO; and (c) to maintain diversity of solutions in population, permutation-based and objective-based population trimming operators are applied in PSO. The experimental results show that the proposed methods are sound on both efficiency and effectiveness for solving MO-MRSCOS problem.
Acknowledgements

The first author wishes to acknowledge the financial support of the Excellent Doctoral Dissertation Fund of Wuhan University of Technology (WHUT). This paper is supported by Hubei Digital Manufacturing Key Laboratory Opening Fund: research on resource service search and optimal-selection theories and experiment in manufacturing grid system (No.SZ0621) and the National Nature Science Fund Project of China (No.50620130441). Thanks for all the references who gives us inspirations and helps. The authors are grateful to the editors and anonymous reviewers of EJOR of their comments that improved the quality of this paper.

References


