Contents lists available at ScienceDirect

Engineering

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

Research
Intelligent Manufacturing—Article

面向工业互联网平台的双维度制造服务协作优化

庞世宝[a,b,c], 郭顺生[a,b], Xi Vincent Wang[c,*], 王磊[a,b], Lihui Wang[c]

[a] School of Mechanical and Electrical Engineering, Wuhan University of Technology, Wuhan 430070, China
[b] Hubei Digital Manufacturing Key Laboratory, Wuhan University of Technology, Wuhan 430070, China
[c] Department of Production Engineering, KTH Royal Institute of Technology, Stockholm 10044, Sweden

ARTICLE INFO

Article history:
Received 23 February 2022
Revised 15 June 2022
Accepted 13 July 2022
Available online 17 November 2022

关键词
制造服务协作
服务优化选择
服务粒度
工业互联网平台

摘要

工业互联网平台是智能制造的关键推动者，能够允许各类物理制造资源虚拟封装后以制造服务的形式进行协作。制造服务协作优化是工业互联网平台的核心功能，其目的在于针对制造任务构建高质量的服务协作方案。在对服务协作方案进行优化时，必须同时满足制造任务的制造功能需求以及制造数量需求。然而，现有的制造服务协作优化研究主要关注面向功能需求的横向服务协作，针对面向数量需求的纵向服务协作鲜有涉及。因此，本文提出了一种同时考虑制造任务功能需求和数量需求的双维度服务协作方法。首先，提出了一种多粒度的制造服务建模方法，并在此基础上建立了双维度制造服务协作优化（DMSCO）模型。在纵向维度上，多个功能相似的制造服务组成一个服务集群以共同完成一个子任务；在横向维度上，多个功能互补的服务集群协作完成整个任务。其中，制造服务的选择和所选服务之间的制造数量分配是模型中的关键问题。针对该问题，本文设计了一种具有多种局部搜索算子的多目标模因算法，并在算法中构建竞争机制以动态调整每个局部搜索算子的选择概率。实验结果表明，与常用算法相比，本文所提算法在收敛性、质量性指标和综合指标等方面均具有优势。

1. 引言

随着工业4.0的不断深化与推进，现代制造业正在从传统操作模式向智能模式转变[1]，在此过程中各种先进制造模式和技术不断涌现[2–4]。作为智能制造的关键推动者，工业互联网平台旨在实现面向服务的按需生产[5–6]。工业互联网平台是物联网技术[7–8]和云计算技术[9–10]在工业领域的进一步延伸和实现，引起了学术界和工业界的广泛关注[11–13]。在工业互联网平台的支持下，孤立的制造资源得以互相连接，各种制造资源被封装成可交付给客户的制造服务以完成不同的制造任务，从而实现分布式制造资源的统一管理与利用[14]。一个制造任务通常需要多个制造服务的协作完成[15]，在工业互联网平台收到制造任务后，将该任务分解为多个可执行的子任务，并将每个子任务与一组候选服务进行匹配。制造服务协作方案是指每个候选服务集合中所选服务构成的服务组合。因此，工业互联网平台的主要功能之一是对制造服务进行编排以生成高质量的服务协作方案，即制造服务协作优化。

* Corresponding author.
E-mail address: wangxi@kth.se (X.V. Wang).
# These authors contributed equally to this work.

英文原文：Engineering 2023, 22(3): 34–48
引用本文：Shibao Pang, Shunsheng Guo, Xi Vincent Wang, Lei Wang, Lihui Wang. Dual-Dimensional Manufacturing Service Collaboration Optimization Toward Industrial Internet Platforms. *Engineering*, https://doi.org/10.1016/j.eng.2022.07.020

在服务协作优化过程中，必须同时满足客户提出的功能需求和制造数量需求，例如，提交给工业互联网平台的任务所需的产品数量通常不仅只有一个。为了提高协作解决方案的整体性能，同时选择多个服务并行完成一个子任务是合理且可行的。通过此种方式，制造服务能够得以充分利用，同时服务的协作方案的效率可以显著提高。此时，服务协作同时存在于两个维度：在纵向维度上，具有相似功能的服务集群并行完成一个子任务；在横向维度上，具有互补功能的多个服务集群依次完成整个任务，构成了一个复杂的协作网络。针对这种网络的服务协作优化不只包括服务的优化选择，同时也需对制造数量进行分配，即如何将一个子任务的制造数量合理分配给多个选定的制造服务。一个制造任务可能有数千种备选服务组合，同时在一个服务组合中，会有更多潜在的制造数量分配方案。因此，进行服务协作优化时，需同时考虑服务选择以及数量分配以生成高质量的服务协作方案。

为了服务在两个维度上的有效协作，合理的制造服务建模方法至关重要。在现有的服务协作优化研究中，一般假定服务调用时间和成本是固定的而不考虑要处理的具体数量[16–17]。然而当涉及数量分配时，此种假定便违反了按需原则。此外，由于子任务由多个服务协作完成，因此子任务的数量分配方案在很大程度上依赖于服务的可用性。从整个工业互联网平台的角度来看，制造服务存在于企业内部和企业间层面，导致不同的粒度[18]。例如，一个制造服务可以对应于单个资源（如机床）、一组资源（如车间中的工作站），同时也可以是集团企业涉及不同下属公司的复杂生产过程。服务的可用性受其组件状态和构成关系的影响而随时间不断变化，在服务协作优化过程中必须对此进行考虑，否则难以保证协作解决方案的可行性和效率。

为了生成高质量的协作解决方案，本文提出了一种新的双维度制造服务协作优化（DMSCO）方法。所提出的方法考虑了制造服务的多粒度特征，同时包括横向维度的功能协作和纵向维度的数量协作。本文的主要贡献如下：

（1）通过分析制造服务的组件和结构，提出了一种多粒度服务建模方法，该方法能够将复杂服务（即复合服务和服务链）引入服务协作优化中。

（2）在考虑服务粒度的情况下，建立了同时考虑服务选择和数量分配的DMSCO集成优化模型，以总体服务协作解决方案的成本、可靠性和完成时间为优化目标。

（3）提出了一种多目标模因算法从而对集成优化模型进行求解，并在算法中嵌入竞争机制，根据局部搜索算子的贡献动态调整其选择概率。

本文的后续部分组织如下：第2节对相关工作进行回顾，第3节提出一种多粒度制造服务建模方法并建立DMSCO集成优化模型，第4节设计模型求解算法，第5节进行仿真实验，第6节给出相关结论。

2. 相关工作

2.1. 制造服务建模

作为制造服务协作的先决条件，物理资源虚拟化和制造服务建模得到了广泛的研究。研究人员最初专注于服务功能性和非功能性信息描述。Shi等[19]提出了一种制造资源层次模型，其中使用可扩展标记语言将从物理资源层提取的资源信息封装在资源表示层中，并在资源接口层中描述针对资源的操作以屏蔽资源的内部信息通信细节。Vichare等[20]针对由机器资源和辅助资源组成计算机数控（CNC）加工系统建立了统一的制造资源模型，该模型是一种标准的信息表示，能够定义CNC加工系统的各种元素并支持自动化工艺规划决策。Ameri和McArthur [21]通过属性推理规则和分类规则扩展了服务描述语言，从而增强其语义并实现了高级推理。Wang和Wang [22]采用Function block将物理资源的能力转化为制造服务，从而提供了一种可行的集成方法，能够以可互操作和灵活的方式协调各种制造资源。

事实上，制造服务通常存在多种粒度[18]，随着研究的推进，多粒度制造服务的构建引起了研究人员的关注。不同粒度的服务共存使得服务协作方案更加高效和灵活。粗粒度服务可以作为一个整体被调用，进而快速满足任务的大部分需求[23–24]。同时，细粒度服务可以灵活地满足其他个性化需求。Liu等[25]提出了由资源聚合功能和资源聚类算法组成的多粒度资源虚拟化方法，将物理资源封装到云服务中，以此衔接复杂制造任务和底层资源。Yu等[26]设计了一种基于数据挖掘技术的多级聚合服务规划方法，从而确定了在不同粒度下维护的制造服务的数量。Zhang等[27]提出了一种新的应用模式，即行业联盟的云制造，为此设计了多粒度制造服务的领域驱动开发方法，包括原子服务和粗粒度服务开发两个阶段，以实现按需提供制造资源。Li等[28]提出了一种基于资源服务之间依赖关系的资源服务链组合进化算法，以构建最优的粗粒度服务，从而提高制造服务的可重用性和选择效率。

2.2. 制造服务协作优化

现有的制造服务协作优化研究主要集中在优化服务选择和配置上。到目前为止，已经提出了大量的模型。Lu

和Xu [29]研究了云制造环境中基于知识的服务组合和自适应资源规划方法，通过基于语义的系统促进了给定服务请求的快速资源分配。Ren等[30]提出了一种考虑制造业服务社会关系的服务选择模型，以最大化整体协同效应，从服务交互数据中抽取了5类的服务社会关系，并基于关系强度的加权聚合建立了服务协同效应模型。在考虑服务协作中的物流服务，Zhou等[31]分析并建立了物流和加工服务的协作优化模型，并将平均任务交付时间作为优化目标。Wu等[32]提出了一个多目标规划模型以同时优化协作解决方案的可持续性（经济、环境和社会绩效）和质量。Wang等[33]建立了一个多目标动态服务组合重构模型，其中根据真实的云制造过程引入了8个关键的实际约束，从而缩小了理论和应用之间的差距。

由于最优服务选择的是NP-hard问题[34]，智能进化算法被广泛用于该问题的求解，可以以较少的计算资源获得模型的高质量解决方案[35]。Bouzary和Chen [36]设计了一种混合遗传算法进化算子的灰狼优化算法，通过提供更多的探索强度来避免狩猎过程中的局部最优停滞。Akbaripour和Houshmand [37]对算法搜索空间进行分析，证明了局部最优值聚集在搜索空间的一小部分，并据此设计了一种将帝国主义竞争算法与局部搜索算法相结合的混合方法。Zhang等[38]提出了一种新的基于遗传的超启发式算法，其中遗传算法充当高层启发式方法，以产生直接在搜索空间执行搜索的低层启发式方法的适当组合，进而解决不确定环境中面向多任务的最优服务选择问题。Zhou和Yao [39]提出了一种多目标混合人工蜂群算法，以优化服务质量和能耗。其中引入了带Lévy飞行的布谷鸟搜索，以保持解决方案的多样性，并设计了一种全面的学习策略，以确保算法的开发和探索之间的平衡。Zhang等[40]提出了一种改进的非支配排序遗传算法II（NSGA-II），结合Tabu搜索和增强的*K*-means机制，以优化参与服务协作的提供商、消费者和算子的长期/短期效用。

迄今为止，对双维度服务协作的探索仍然有限。一些相关研究同样仅关注于其中的服务选择问题[41–42]。然而，当多个服务并行完成同一子任务时，将制造数量分配给所选服务对于制造服务协作优化是不可或缺的。Zhu等[43]针对数量分配进行研究但忽略了服务选择阶段，他们还将制造数量分配给所有候选服务。尽管缩短了完成时间，但由于涉及大量服务，协作解决方案的执行变得困难。此外，在先前的参考文献中，制造服务的可用性被假定为数值。尽管这样的解决方案可能适用于计算机科学领域，但制造业的情况更复杂。制造服务的可用性受到许多因素的影响，如服务结构、内部组件服务和已经占用的时间。因此，单个数值不能准确反映制造服务的可用性。

2.3. 概述

如上所述，制造服务建模和协作优化都得到了广泛的研究。但对服务协作优化的研究很少考虑服务粒度的特征，导致服务建模和协作优化之间的不一致。尽管在这方面做了一些努力[44–45]，但仍缺少对不同粒度服务可用性的详细描述。此外，现有的服务协作优化研究主要集中于选择满足功能需求的服务，而忽略了与数量相关的协作，需要进一步研究双维度服务协作的优化模型和具体算法。为此，本文提出了一种基于多粒度服务的服务选择和数量分配DMSCO方法。

3. 问题制定

3.1. 多粒度制造服务建模

在协作解决方案的总体性能中，总成本、可靠性和完成时间是客户最关心的问题，分别对应于服务的单位成本、可靠性以及处理速度。此外，解决方案的可行性受到服务可用性的限制。在建模制造服务时，本文主要关注这些属性。其中，服务的可用性是从时间的角度来构建的，即服务是否在特定时间段内可用。制造服务既存在于企业内部层面，也存在于企业间层面。根据其内部组件和结构，可以在三个粒度上对服务进行建模：资源服务、复合服务和服务链。表1列出了用于建模这些服务的符号，表2显示了服务表达式和属性。

*资源服务*是由企业内的单个物理资源映射的简单服务。由于资源不能一直可用，因此服务的可用性表示为一系列可用的时间段，其中元素d_k指示资源可访问的持续时间。

*复合服务*包括多个相互依赖的资源服务，通常由单个企业提供。当向外部提供功能时，同时调用内部的组件服务$V=\{S_1, S_2, \ldots, S_n, \ldots, S_N\}$，其中核心服务$S_1$为功能的主要实现者，其余的是围绕$S_1$协作的辅助服务。复合服务的可用性是其组件可用性的交集。

*服务链*是由多个相关企业（如企业集团或联盟）提供的固定服务。它由一组组件（即资源或复合）服务$V=\{S_1, S_2, \ldots, S_n, \ldots, S_N\}$组成，这些服务以特定的执行约束$E=\{<S_m, S_n>|S_m\in V, S_n\in V\}$协作。$<S_m, S_n>$表示$S_m$在$S_n$之前被调用，并且每个组件服务在被调用之前不会被占用。服务链的处理速度和可用性分别表示为组件的相应属性集。

表1 用于多粒度制造服务建模的符号

Notation	Description
S	Manufacturing service; referring to a resource service, a composite service, or a service chain
m, n	Indexes of component services in a composite service or a service chain, where m, n = 1, 2, …, N
k	Index of available durations of a resource service or a composite service, where k = 1, 2, …, K
c, r, p, a	Unit cost, reliability, processing speed, and availability of S, respectively
d_k	kth available duration of a resource service or composite service
V	Set of component services in a composite service or service chain
E	Set of execution constraints in a service chain
S_m, S_n	mth, nth component services in a composite service or service chain, respectively
c_n, r_n, p_n, a_n	Unit cost, reliability, processing speed, and availability of S_n, respectively

表2 多粒度制造服务的建模

Service granularity	Expression	Unit cost	Reliability	Processing speed	Availability
Resource service	$S=\{c, r, p, a\}$	c	r	p	$a=<d_1, d_2, \ldots, d_k, \ldots, d_K>$
Composite service	$S=\{c, r, p, a, V\}$	$c=\sum_{S_n\in V} c_n$	$r=\left[\prod_{S_n\in V} r_n\right]^{\frac{1}{N}}$	$p=p_1$	$a=\bigcap_{S_n\in V} a_n$
Service chain	$S=\{c, r, p, a, V, E\}$	$c=\sum_{S_n\in V} c_n$	$r=\left[\prod_{S_n\in V} r_n\right]^{\frac{1}{N}}$	$p=\{p_1, p_2, \ldots, p_n, \ldots, p_N\}$	$a=\{a_1, a_2, \ldots, a_n, \ldots, a_N\}$

3.2. DMSCO模型

在DMSCO模型中，制造任务T包含具有不同功能需求的I个子任务，每个子任务ST_i可以由一组候选服务$\mathrm{CS}\text{-}\mathrm{S}_i$来完成。关于纵向维度，从$\mathrm{CSS}_i$中选择功能相同的服务构成服务集群$\mathrm{CS}_i$（$\mathrm{CS}_i=\{S_{i1}, S_{i2}, \ldots, S_{ij}, \ldots, S_{iji}\}$）来完成$\mathrm{ST}_i$。除了服务选择之外，还需要确定$\mathrm{ST}_i$的制造数量在$\mathrm{CS}_i$中的服务的分配。在横向维度上，所有功能互补的$\mathrm{CS}_i$构成了$T$的最终协作解决方案CS（$\mathrm{CS}=\{\mathrm{CS}_1, \mathrm{CS}_2, \ldots, \mathrm{CS}_i, \ldots, \mathrm{CS}_I\}$）。下文从两个维度构建模型，包括不同粒度的服务的可用性约束，并以服务协作方案的总成本、可靠性和完成时间为优化目标。表3列出了DMSCO使用的符号。

3.2.1. 纵向维度的协作

考虑数量分配，首先制定了单个S_{ij}的总成本C_{ij}、可靠性R_{ij}和完成时间F_{ij}。公式因服务粒度而异。对于资源服务或复合服务：

$$C_{ij}=\mathrm{Amt}_{ij}\cdot c_{ij} \tag{1}$$

$$R_{ij}=r_{ij} \tag{2}$$

$$F_{ij}=B_{ij}+\mathrm{Amt}_{ij}/p_{ij} \tag{3}$$

$$\Phi_{ijk}=\begin{cases}1, & \text{if } \left[B_{ij}, F_{ij}\right]\subset d_{ijk}\\ 0, & \text{otherwise}\end{cases} \tag{4}$$

$$\sum\nolimits_{d_{ijk}\in a_{ij}}\Phi_{ijk}=1 \tag{5}$$

其中，公式（5）表示Amt_{ij}必须在a_{ij}的可用持续时间内处理，以确保S_{ij}的可用性。

对于服务链，C_{ij}和R_{ij}的计算方法相同。B_{ij}和F_{ij}由组件服务决定，公式如下：

$$B_{ij}=\min\nolimits_{S_{ijn}\in V_{ij}}(b_{ijn}) \tag{6}$$

$$F_{ij}=\max\nolimits_{S_{ijn}\in V_{ij}}(f_{ijn}) \tag{7}$$

$$f_{ijn}=b_{ijn}+\mathrm{Amt}_{ij}/p_{ijn} \tag{8}$$

$$\Phi_{ijnk}=\begin{cases}1, & \text{if } \left[b_{ijn}, f_{ijn}\right]\subset d_{ijnk}\\ 0, & \text{otherwise}\end{cases} \tag{9}$$

$$\prod_{S_{ijn}\in V_{ij}}\sum_{d_{ijnk}\in a_{ijn}}\Phi_{ijnk}=1 \tag{10}$$

$$f_{ijm}\le b_{ijn}\quad \forall <S_m, S_n>\in E_{ij} \tag{11}$$

其中，公式（10）和公式（11）分别确保组件服务的可用性和正确的调用顺序。

对于服务集群CS_i，总成本C_i、可靠性R_i、开始时间B_i和结束时间F_i公式如下：

$$C_i=\sum\nolimits_{S_{ij}\in \mathrm{CS}_i}C_{ij} \tag{12}$$

$$R_i=\sum\nolimits_{S_{ij}\in \mathrm{CS}_i}\frac{\mathrm{Amt}_{ij}}{\mathrm{Amt}_i}\cdot R_{ij} \tag{13}$$

$$B_i=\min\nolimits_{S_{ij}\in \mathrm{CS}_i}(B_{ij}) \tag{14}$$

$$F_i=\max\nolimits_{S_{ij}\in \mathrm{CS}_i}(F_{ij}) \tag{15}$$

$$\mathrm{Amt}=\mathrm{Amt}_i=\sum_{S_{ij}\in \mathrm{CS}_i}\mathrm{Amt}_{ij} \tag{16}$$

$$\mathrm{CS}_i\subset \mathrm{CSS}_i \tag{17}$$

其中，公式（16）保证分配给CS_i中的服务的制造数量之

表3 DMSCO模型使用的符号

Notation	Description
i	Index of subtasks, where $i = 1, 2, \cdots, I$
j	Index of manufacturing services in CS_i, where $j = 1, 2, \cdots, J_i$
T	Manufacturing task
Amt	Total required amount of T
CS	Manufacturing service collaboration solution for T
C, R, F	Total cost, reliability, and finishing time, respectively, of CS to fulfill T
ST_i	ith subtask of T
Amt_i	Total required amount of ST_i
CSS_i	Set of candidate manufacturing services for ST_i
CS_i	Cluster of manufacturing services selected from CSS_i to fulfill ST_i
C_i, R_i, B_i, F_i	Total cost, reliability, beginning time, and finishing time, respectively, of CS_i to fulfill ST_i
S_{ij}	jth manufacturing service in CS_i (S_{ij} is a specific S, which is used in the context in which T is involved)
Amt_{ij}	Processing amount distributed to S_{ij}
c_{ij}, r_{ij}, p_{ij}, a_{ij}	Unit cost, reliability, processing speed, and availability of S_{ij}, respectively
C_{ij}, R_{ij}, B_{ij}, F_{ij}	Total cost, reliability, beginning time, and finishing time, respectively, of S_{ij} to process Amt_{ij}
d_{ijk}	kth available duration of a resource service or composite service S_{ij}
Φ_{ijk}	Binary variable equal to 1 if Amt_{ij} is processed in the duration d_{ijk} of S_{ij}
V_{ij}	Set of component services in a service chain S_{ij}
E_{ij}	Set of execution constraints in a service chain S_{ij}
S_{ijn}	nth component service in a service chain S_{ij}
p_{ijn}, a_{ijn}	Processing speed and availability of S_{ijn}, respectively
b_{ijn}, f_{ijn}	Beginning time and finishing time, respectively, of S_{ijn} to process Amt_{ij}
d_{ijnk}	kth available duration of S_{ijn}
Φ_{ijnk}	Binary variable equal to 1 if Amt_{ij} is processed in the duration d_{ijnk} of S_{ijn}
C^{max}, R^{max}, F^{max}	Maximum total cost, reliability, and finishing time, respectively, of CS
C^{min}, R^{min}, F^{min}	Minimum total cost, reliability, and finishing time, respectively, of CS

和等于Amt_i，并且公式（17）保证从候选服务集CSS_i中选择CS_i。

3.2.2. 横向维度的协作

关于横向维度，服务协作解决方案CS的总成本C、可靠性R和完成时间F公式如下：

$$C = \sum_{CS_i \in CS} C_i \tag{18}$$

$$R = \left[\prod_{CS_i \in CS} R_i \right]^{\frac{1}{I}} \tag{19}$$

$$F = F_I \tag{20}$$

$$B_i \geq F_{i-1} \geq 0 \tag{21}$$

由于C、R和F的测量方法和单位不同，目标函数标准化如下，以同时优化度量：

$$\text{minimize } f_1 = \begin{cases} \dfrac{C - C^{min}}{C^{max} - C^{min}}, & \text{if } C^{max} \neq C^{min} \\ 0, & \text{otherwise} \end{cases} \tag{22}$$

$$\text{minimize } f_2 = \begin{cases} \dfrac{R^{max} - R}{R^{max} - R^{min}}, & \text{if } R^{max} \neq R^{min} \\ 0, & \text{otherwise} \end{cases} \tag{23}$$

$$\text{minimize } f_3 = \begin{cases} \dfrac{F - F^{min}}{F^{max} - F^{min}}, & \text{if } F^{max} \neq F^{min} \\ 0, & \text{otherwise} \end{cases} \tag{24}$$

4. 提出的DMSCO模因算法

服务选择已被证明是一个NP-hard问题[34]。因此，同时进行服务选择和数量分配的DMSCO同样是NP-hard问题。为了解决这个问题，开发了一种基于竞争的多目标模因算法（CMOMA）。

4.1. CMOMA框架

模因算法代表了一类元启发式算法，该算法将进化算法与局部搜索相结合[46]。所提出的CMOMA框架如图1所示，其中灰狼优化算法用于全局搜索，基于竞争的变邻

域搜索算法用于局部搜索。为了适应DMSCO，设计了一种用于全局搜索的混合交叉算子。此外，为了充分探索搜索空间，通过快速非支配排序来更新种群并从种群中随机选择每个解的父体（领导者）进行交叉。

局部搜索在模因算法中具有重要意义，针对具体问题进行定制。本文开发了两类与服务选择和数量分配相对应的局部搜索算子，每类包括4个算子。进行局部搜索时，选择其中一个算子执行。此外，还设计了一种竞争机制来更新算子的选择概率。

4.2. 编码和解码

为了适应DMSCO，提出了一种双向量编码方案，该方案涉及服务选择向量$\boldsymbol{X}$和数量分配向量$\boldsymbol{Y}$。每个向量由对应于I个子任务的I个片段组成。由于子任务ST_i由J_i个服务协作完成，因此$\boldsymbol{X}$和$\boldsymbol{Y}$中的片段$\mathbf{Seg}_i$包括J_i个元素，这些元素携带关于为ST_i选择的服务的信息。$\boldsymbol{X}$的元素x_{ij}是$[1, L_i]$中的整数，表示CSS_i中选定服务的索引（L_i是CSS_i的长度）。$\boldsymbol{Y}$用于将数量分配给所选服务。元素y_{ij}是[0, 1]中的一个实数，表示分配给$\boldsymbol{X}$的等位服务的数量的权重。图2显示了编码方案的一个示例，其中任务包含三个子任务，每个子任务有20个候选服务，最多可以选择其中的三个。基于编码方案，随机初始化CMOMA种群。

在解码$\boldsymbol{X}$之前，检查$\boldsymbol{Y}$的元素。如果y_{ij}等于0，则$\boldsymbol{X}$上的等位服务将不参与协作。同时，为了防止分配给服务的数量过小，当小于0.1时，将y_{ij}设置为0。如果属于$\mathbf{Seg}_i$的所有y_{ij}都等于0，则将其中一个设置为0.1到1.0之间的随机数。检查$\boldsymbol{Y}$后，如果等位基因y_{ij}大于0，则选择CSS_i中的第x_{ij}个候选服务。当解码$\boldsymbol{Y}$时，将每个$\mathbf{Seg}_i$中大于0的最后一个y_{ij}被标记为y_i*，然后计算分配给服务的量，如公式（25）所示。假设图2示例中的任务总量为10 000，解码结果如图3所示。

$$\mathrm{Amt}_{ij}=\begin{cases}\left\lfloor \mathrm{Amt}_i\cdot\dfrac{y_{ij}}{\sum_{y_{ij}\in \mathbf{Seg}_i} y_{ij}}\right\rfloor, & y_{ij}\neq y_i^* \\ \mathrm{Amt}_i-\sum_{\substack{y_{ij}\in \mathbf{Seg}_i \\ y_{ij}\neq y_i^*}}\mathrm{Amt}_{ij}, & y_{ij}=y_i^*\end{cases} \tag{25}$$

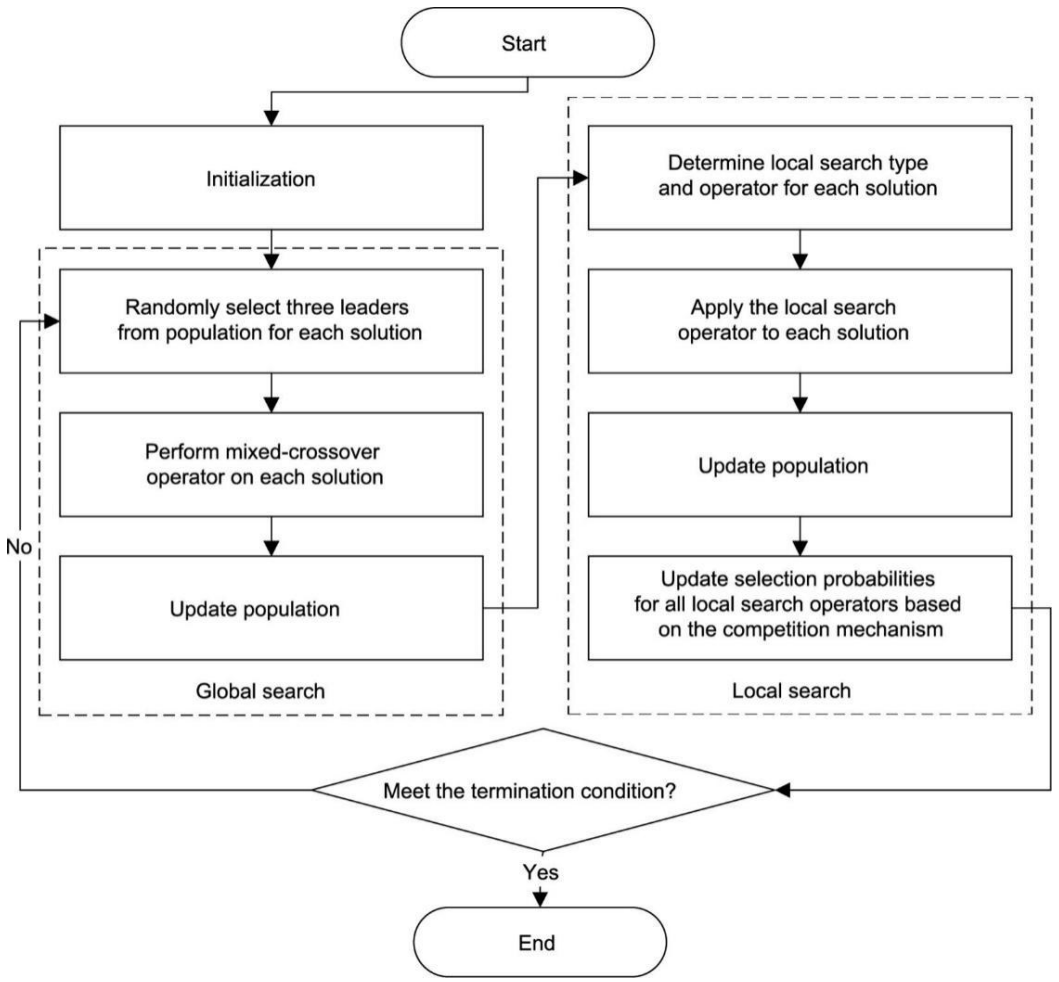

图1. CMOMA框架。

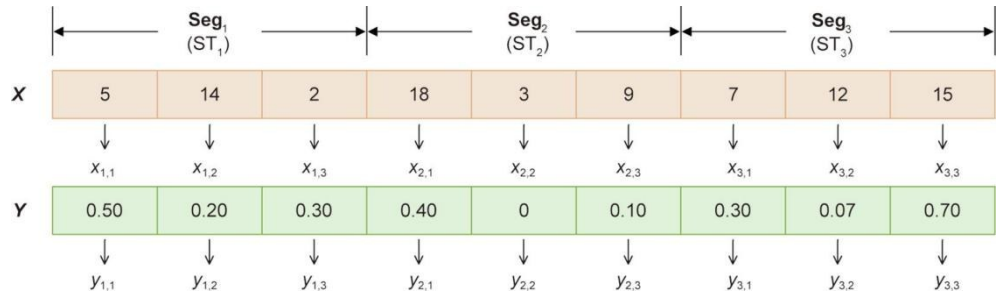

图2. 编码方案示例。

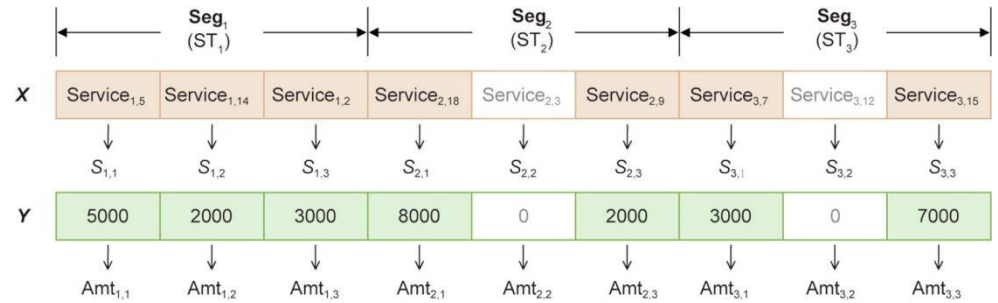

图3. 解码方案示例。

4.3. 全局搜索

全局搜索算法是从灰狼优化算法扩展而来的，灰狼优化算法是受灰狼狩猎行为启发的新兴进化算法[47]。在灰狼优化算法的每次迭代中，每个解在前三个最佳解α、β和δ的指导下更新其位置。与类似算法相比，优化器的特点是搜索速度快，易于实现。该算法最初是为解决连续优化问题而设计的。由于解中同时存在离散变量$\boldsymbol{X}$和连续变量$\boldsymbol{Y}$，因此在所提出的算法中设计了混合交叉算子。对于$\boldsymbol{X}$，交叉算子表示为公式（26）和公式（27），其中rand()是返回[0, 1]中随机数的函数，x_{ij}^{new}、x_{ij}、x_{ij}^{α}、x_{ij}^{β}和x_{ij}^{δ}是新解、原始解、α、β和δ的$\boldsymbol{X}$元素。图4显示了$\boldsymbol{X}$的交叉算子的示例。

$$r = \text{rand}() \tag{26}$$

$$x_{ij}^{\text{new}} = \begin{cases} x_{ij}^{\alpha}, & \text{if} \quad r < 0.25 \\ x_{ij}^{\beta}, & \text{else if} \quad r < 0.50 \\ x_{ij}^{\delta}, & \text{else if} \quad r < 0.75 \\ x_{ij}, & \text{otherwise} \end{cases} \tag{27}$$

对于$\boldsymbol{Y}$，采用灰狼优化算法中的原始交叉算子，如公式（28）~（31）所示，其中a在迭代期间从2线性减少到0，并且y_{ij}^{new}、y_{ij}、y_{ij}^{α}、y_{ij}^{β}和y_{ij}^{δ}是新解的$\boldsymbol{Y}$元素，即原始解α、β和δ。

$$y_{ij}^{1} = y_{ij}^{\alpha} - a \cdot (2 \cdot \text{rand}() - 1) \cdot \left| 2 \cdot \text{rand}() \cdot y_{ij}^{\alpha} - y_{ij} \right| \tag{28}$$

$$y_{ij}^{2} = y_{ij}^{\beta} - a \cdot (2 \cdot \text{rand}() - 1) \cdot \left| 2 \cdot \text{rand}() \cdot y_{ij}^{\beta} - y_{ij} \right| \tag{29}$$

$$y_{ij}^{3} = y_{ij}^{\delta} - a \cdot (2 \cdot \text{rand}() - 1) \cdot \left| 2 \cdot \text{rand}() \cdot y_{ij}^{\delta} - y_{ij} \right| \tag{30}$$

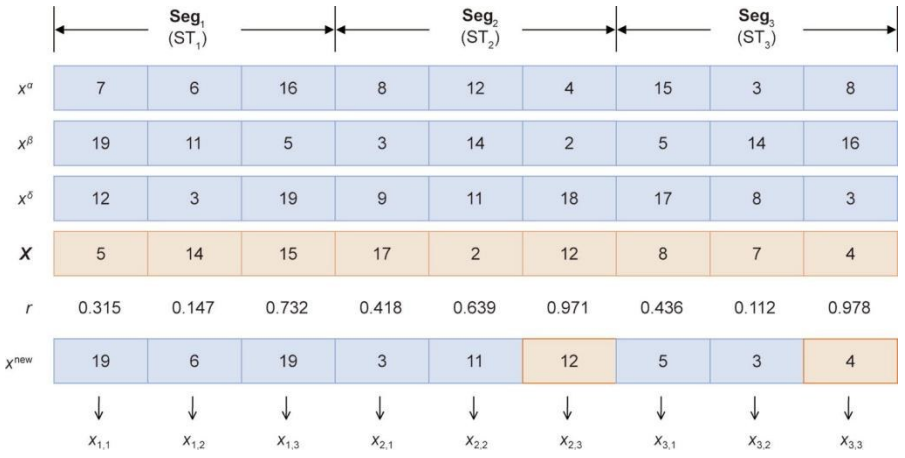

图4. X的交叉算子示例。

$$y_{ij}^{\text{new}} = \frac{1}{3}\sum_{p=1,2,3} y_{ij}^{p} \tag{31}$$

关于多目标优化，大多数研究在灰狼优化算法中引入外部档案，以保存迄今为止获得的非支配解集，并利用轮盘赌方法从外部档案中选择执行交叉的父解。然而，由于外部档案只保留了非支配解集，可能会导致算法陷入局部最优。因此，本文采用快速非支配排序来更新种群。在选择父解时，种群中的所有解具有相同的被选概率。这样，不在非支配解集中的解有机会被选择，有利于跳出局部最优，充分探索搜索空间。有关快速非支配排序的详细信息，可参见参考文献[48]。

4.4. 基于竞争的变量邻域搜索

通过基于竞争的可变邻域搜索算法来执行局部搜索。设计两种类型的局部搜索算子分别用于服务选择和数量分配，每种类型包含4个算子。当对解执行局部搜索时，首先随机确定局部搜索类型，然后根据由竞争机制更新的概率从该类型中选择某一搜索算子。

4.4.1. 局部搜索服务选择

由于高质量服务可以显著改善协作解决方案，因此局部搜索服务选择旨在用高质量服务替代低质量服务。这种类型包括三个面向目标的算子OS_1、OS_2和OS_3，以及一个混合算子OS_4。每个面向目标的算子对应于一个要优化的目标，并对解决方案中的所有片段执行相同的相应操作。混合算子对解中的每个片段随机执行针对不同目标的相应操作，致力于保持平衡。每个算子详情如下：

（1）总成本f_1的OS_1：对于X中的每个Seg_i，找到Seg_i中单位成本最高的服务，并将其替换为CSS_i中随机选择的单价较低的服务。

（2）可靠性f_2的OS_2：对于X中的每个Seg_i，找到Seg_i中可靠性最低的服务，并将其替换为CSS_i中随机选择的可靠性较高的服务。

（3）结束时间f_3的OS_3：对于X中的每个Seg_i，找到Seg_i中速度最低的服务，并将其替换为从CSS_i中随机选择的速度较高的服务。如果服务是服务链，则内部组件服务的最低速度被视为服务链的速度。

（4）OS_4用于所有目标：对于X中的每个Seg_i，随机选择一个要优化的目标，并根据上述相应的操作修改Seg_i。这样，可以在解决方案上以相同的概率执行不同的服务选择操作以优化不同的目标。

4.4.2. 局部搜索数量分配

关于总成本和可靠性，如果将所选服务中相对低质量的服务分配给较少的子任务制造数量，则协作解决方案将更好。关于完成时间，按照为子任务所选服务的速度成比例的方式分配制造数量。相应地，设计了一个用于数量分配的局部搜索，包括三个面向目标的算子OA_1、OA_2和OA_3，以及一个混合算子OA_4。算子详情如下：

（1）总成本f_1的OA_1：对于Y中的每个Seg_i，找到Seg_i中单位成本最高的服务，并根据公式（32）降低相应的y_{ij}，其中y'_{ij}是降低前的值。

$$y_{ij} = \text{rand}() \cdot y'_{ij} \tag{32}$$

（2）可靠性f_2的OA_2：对于Y中的每个Seg_i，找到Seg_i中可靠性最低的服务，并根据公式（32）降低相应的y_{ij}。

（3）结束时间f_3的OA_3：对于Y中的每个Seg_i，根据公式（33）调整属于Seg_i的所有y_{ij}，其中$p_{i,x_{ij}}$是当前服务的速度，x_{iw}是Seg_i中每个选定服务的索引。如果服务是服务链，内部组件服务的最低速度被视为服务链的速度。

$$y_{ij} = \frac{p_{i,x_{ij}}}{\sum_{x_{iw} \in \mathbf{Seg}_i} p_{i,x_{iw}}} \tag{33}$$

（4）所有目标的OA_4：对于Y中的每个Seg_i，随机选择一个要优化的目标，并根据上述相应的操作修改Seg_i。

4.4.3. 竞争机制

随着算法的迭代执行，不同目标的优化进度通常不同。因此开发一种竞争机制，将更多的计算资源分配给具有更多改进空间的目标。在该机制中，具有相同类型的算子相互竞争以获得更高的选择概率。根据算子对解的影响更新概率。如果一个算子改进了一个解决方案，它将被激励获得更高的概率；否则将被抑制。

为了量化当前迭代中算子的影响，在局部搜索之后更新种群，并识别和分类应用了局部搜索算子的解决方案。以服务选择的局部搜索为例，OS_g的影响被量化为公式（34）和公式（35），其中g（或h）是优化目标的索引。公式（34）适用于面向目标的算子，并将聚焦目标作为主要因素。其他目标被列为次要因素，以避免其过度恶化。公式（34）中的η是用于控制各因素之间权重的协调系数；η大于1/3，以突出对聚焦目标的影响。公式（35）适用于考虑所有目标的混合算子。U_{OS_g}是包含应用OS_g的解的集合，$f_g(\text{CS})\,\text{CS}'$是应用局部搜索算子之前的原始解，$f_g(\text{CS})$和$f_g(\text{CS}')$是CS和CS'的第$g$个目标，$f_h(\text{CS})$和$f_h(\text{CS}')$是CS和CS'的第$h$个目标，$\varepsilon$是一个小数字，防止分母为0。

$$e_{\mathrm{OS}_g}=\sum_{\mathrm{CS}\in U_{\mathrm{OS}_g}}\left[\eta\cdot\left(\frac{f_g(\mathrm{CS}')-f_g(\mathrm{CS})}{f_g(\mathrm{CS}')+\varepsilon}\right)+\frac{1-\eta}{2}\cdot\sum_{\substack{h=1\\h\neq g}}^{3}\left(\frac{f_h(\mathrm{CS}')-f_h(\mathrm{CS})}{f_h(\mathrm{CS}')+\varepsilon}\right)\right],\ g=1,2,3 \tag{34}$$

$$e_{\mathrm{OS}_4}=\frac{1}{3}\cdot\sum_{\mathrm{CS}\in U_{\mathrm{OS}_4}}\sum_{h=1}^{3}\left(\frac{f_h(\mathrm{CS}')-f_h(\mathrm{CS})}{f_h(\mathrm{CS}')+\varepsilon}\right) \tag{35}$$

在上述公式中，如果OS_g改善了解，则e_{OS_g}会增加；如果解恶化，则e_{OS_g}会减少，从而实现激励或抑制。为了防止e_{OS_g}为负值或0，公式（36）中给出了一个校正函数$H(e_{\mathrm{OS}_g})$，其中μ_{OS}是一个小数字。公式（37）中的μ'_{OS}是上一次迭代中μ_{OS}的值。在本文中，μ_{OS}在算法开始时设置为0.01。由于仅考虑当前迭代中的影响可能会导致选择概率的剧烈变化，从而导致算法的不稳定性，因此引入了算子的历史影响来缓解变化。当前迭代中使用的概率携带历史信息，因此使用它们来更新选择概率。公式（38）给出了更新OS_g的选择概率的公式，其中p_{OS_g}是当前迭代中使用的选择概率，$p^*_{\mathrm{OS}_g}$是下一次迭代中要使用的更新概率。不失一般性，算子的选择概率在开始时是相等的。类似地，可以更新OA_g的选择概率。

$$H(e_{\mathrm{OS}_g})=\max(e_{\mathrm{OS}_g},\mu_{\mathrm{OS}}) \tag{36}$$

$$\mu_{\mathrm{OS}}=\begin{cases}0.01\cdot\max_{h=1}^{4}(e_{\mathrm{OS}_h}), & \text{if } \max_{h=1}^{4}(e_{\mathrm{OS}_h})>0\\ \mu'_{\mathrm{OS}}, & \text{otherwise}\end{cases} \tag{37}$$

$$p^*_{\mathrm{OS}_g}=\frac{\sqrt{p_{\mathrm{OS}_g}\cdot H(e_{\mathrm{OS}_g})}}{\sum_{h=1}^{4}\sqrt{p_{\mathrm{OS}_h}\cdot H(e_{\mathrm{OS}_h})}} \tag{38}$$

5. 仿真实验

通过仿真实验对所提CMOMA的性能进行评估，每个解均同时确定服务选择方案和所选服务的数量分配方案。实验中所涉及的算法采用Java（JDK 14）实现，运行环境为Windows 10（64）、32 GB RAM、2.2 GHz Intel Core i7-8750H。同时为了提高运行效率，所有算法均以并行方式进行解码。

5.1. 性能指标和测试案例

本文采用世代距离（GD）、反向世代距离（IGD）、超体积（HV）和解集覆盖率（SC）作为算法的性能评价指标。

（1）GD为收敛性指标，其计算方式如公式（39）所示。GD越小，表示算法收敛性越好。

$$\mathrm{GD}=\frac{1}{|U|}\cdot\sqrt{\sum_{u\in U}\min_{p\in P}(d_{p,u})} \tag{39}$$

（2）IGD为综合性指标，用于同时评价解集的收敛性和多样性，其计算方式如公式（40）所示。IGD越小，表示解集的综合表现越好。

$$\mathrm{IGD}=\frac{1}{|P|}\cdot\sqrt{\sum_{p\in P}\min_{u\in U}(d_{p,u})} \tag{40}$$

（3）HV为目标空间中的解集所覆盖的范围，其计算方式如公式（41）所示。HV越大，表示解集越优。

$$\mathrm{HV}=\mathrm{volume}\left(\bigcup_{u\in U}c_u\right) \tag{41}$$

（4）SC用于测量两个非支配解集U和V之间的支配关系，可对解集的量进行比较，其计算方式如公式（42）所示。当$C(U,V)$大于$C(V,U)$时，表示解集U质量更好。

$$C(U,V)=\frac{\left|\{v\in V\mid\exists u\in U:u>v\}\right|}{|V|} \tag{42}$$

实验基于文献[40]中采用的原则生成测试实例，即具有较高可靠性或较快处理速度的制造服务的单位成本较高，反之亦然。如表4所示，生成了21个实例用于评估算法性能，其中子任务的数量有三种：15、30和45。在每种规模中，使用复杂服务（即复合服务和服务链）的子任务的比例从20%增加到80%，增长间隔为10%。资源服务的可用时间（包括复杂服务中的组件资源服务）随机生成。本文中每个子任务有50个候选服务，最多可以选择三个服务。

5.2. 协调系数η的敏感性分析

为了更新局部搜索算子的选择概率，竞争机制中的参数η是CMOMA中需要调整的唯一参数，并影响算法的性能。理论上，当更新选择概率时，较大的η值会导致算法集中关注对定向目标的影响。相比之下，较小的η值会使算法注意到对其他目标的影响。

为了研究η对CMOMA的影响并选择合适的η值，进行了一组实验。首先，测试了0.4、0.6、0.8和1.0四个值，以大致确定η的合理范围。在每个值上，CMOMA在中型实例11上独立执行20次，每次执行的运行时间为15 s。初步实验后，种群规模设置为200。图5显示了具有标准偏差的IGD的平均值。帕累托最优前沿由在所有η值下运行CMOMA获得的非支配解组成。如图5所示，$\eta<0.8$是CMOMA的理想范围，其中算法保持了较好的性能，即当更新选择概率时，考虑对定向目标的更多影响有利于算法的性能。为了细化η，在[0.8, 1.0]中进行了进一步的测试，步长为0.02。图6描述了具有95%置信区间的

表4 测试实例中不同服务的比例

Test instance	Number of subtasks	Test instance	Number of subtasks	Test instance	Number of subtasks	Proportion		
						Service chain	Composite service	Resource service
1	15	8	30	15	45	10%	10%	80%
2		9		16		15%	15%	70%
3		10		17		20%	20%	60%
4		11		18		25%	25%	50%
5		12		19		30%	30%	40%
6		13		20		35%	35%	30%
7		14		21		40%	40%	20%

HV均值。尽管差异不明显，但相比之下，[0.9, 1.0]中的η比[0.8, 0.9]中η具有优势。然而，随着η的增加，算法的性能会逐步降低。特别在$\eta = 1$点，即不包括其他目标影响的情况下，可以观察到明显的性能下降。这一现象表明，定向目标和其他目标的影响是至关重要的，需要在这两个方面之间取得平衡。在测试值中，当$\eta = 0.9$时，CMOMA达到较高的HV平均值，置信区间较窄。因此，η被设置为0.9以均衡不同目标的影响。此外，从全局角度来看，HV的轨迹显示出轻微的波动，但当$\eta > 0.8$时通常保持稳定，说明η在适当范围内是不敏感的，这也同时验证了CMOMA的鲁棒性。

5.3. 竞争机制的绩效验证

为了验证竞争机制的有效性，开发了一种CMOMA变体，即固定概率多目标模因算法（FMOMA）用于比较。该变体去除了竞争机制，并采用固定的等概率来选择局部搜索算子。两种算法在每个实例上重复20次，每次重复的运行时间相同（小、中、大型实例分别为10 s、15 s和20 s）。表5显示了GD、IGD和HV的平均值。为了计算IGD，同时采用三个先进的多目标算法的结果，即NSGA-II [48]、SPEA-2 [49]和多目标灰狼优化算法（MOGWO）[50]（在下一章中具体讨论）结果以构建帕累托最优前沿。对每个案例进行0.05显著性水平的独立双样本t检验。下面分别使用符号“+”“=”“−”表示CMOMA的结果在统计上优于、类似或低于FMOMA。很明显，CMOMA在所有实例上都获得最佳GD值。从统计角度也证实了这一优势，证明了CMOMA的较好收敛性。对于IGD，CMOMA在20个实例上获得最佳值，其中15个实例的结果显著优于FMOMA。就HV而言，尽管在统计意义上两种算法在7个实例上的结果相似，CMOMA总是比FMOMA产生更好的结果。总之，CMOMA在综合指标方面优于FMOMA。

为了评估质量，表6显示SC指标的结果。C(CMOMA, FMOMA)在所有情况下的均值明显大于C(FMOMA, CMOMA)，表明CMOMA在搜索高质量解决方案方面的优势。为了更直观地表示，图7用线性回归趋势线描述了SC结果。随着实例大小的增加，CMOMA的优势变得明显。特别是，在大型实例中，FMOMA的解集中约有一半解被由CMOMA的解集中的解支配。在相同规模的实例内部中，观察到类似的结论。随着复杂服务比例的增加，

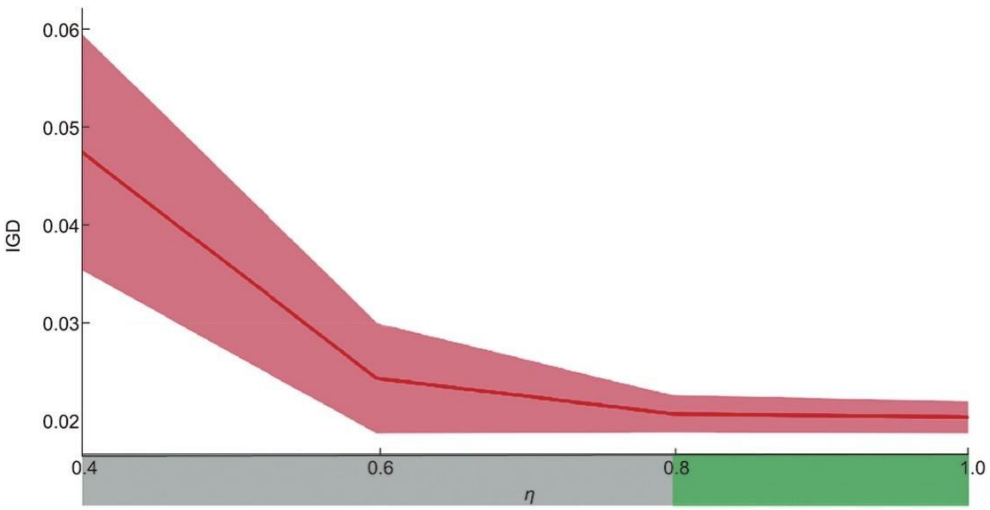

图5. 实例11上具有不同η值的标准偏差的IGD平均值。

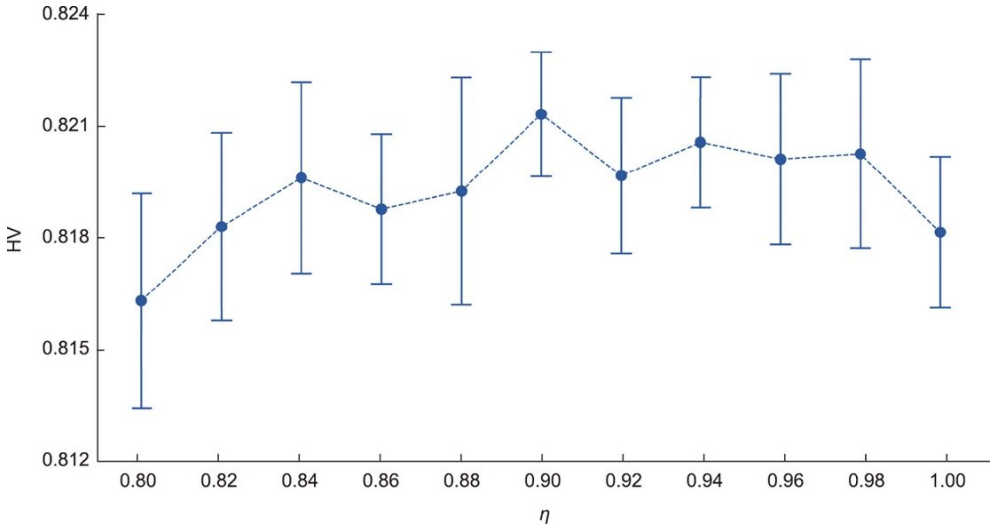

图6. 实例11上不同η值下具有95%置信区间的HV均值。

表5 所有测试实例中CMOMA和FMOMA的GD、IGD和HV的均值和两样本t检验结果

Test instance	GD		IGD		HV	
	CMOMA	FMOMA	CMOMA	FMOMA	CMOMA	FMOMA
1	1.1402×10^{-3}	$1.3180 \times 10^{-3}+$	2.3340×10^{-2}	$2.3355 \times 10^{-2}=$	7.9977×10^{-1}	$7.9802 \times 10^{-1}=$
2	8.9269×10^{-4}	$1.0554 \times 10^{-3}+$	2.1670×10^{-2}	$2.2390 \times 10^{-2}+$	8.2696×10^{-1}	$8.2328 \times 10^{-1}+$
3	9.2624×10^{-4}	$1.1366 \times 10^{-3}+$	2.0485×10^{-2}	$2.1510 \times 10^{-2}+$	8.1010×10^{-1}	$8.0836 \times 10^{-1}=$
4	1.0683×10^{-3}	$1.3717 \times 10^{-3}+$	1.9490×10^{-2}	$2.1390 \times 10^{-2}+$	8.5031×10^{-1}	$8.4925 \times 10^{-1}=$
5	9.0086×10^{-4}	$1.2245 \times 10^{-3}+$	2.0115×10^{-2}	$2.1690 \times 10^{-2}+$	8.5315×10^{-1}	$8.5117 \times 10^{-1}+$
6	8.8471×10^{-4}	$1.1432 \times 10^{-3}+$	1.7660×10^{-2}	$1.8415 \times 10^{-2}+$	8.5614×10^{-1}	$8.5769 \times 10^{-1}=$
7	1.0415×10^{-3}	$1.4650 \times 10^{-3}+$	1.7275×10^{-2}	$1.8140 \times 10^{-2}+$	8.8252×10^{-1}	$8.8138 \times 10^{-1}=$
8	7.9311×10^{-4}	$1.0661 \times 10^{-3}+$	1.9570×10^{-2}	$2.0410 \times 10^{-2}+$	7.7647×10^{-1}	$7.7718 \times 10^{-1}=$
9	8.7197×10^{-4}	$1.0704 \times 10^{-3}+$	1.9740×10^{-2}	$1.9670 \times 10^{-2}=$	8.0296×10^{-1}	$7.9883 \times 10^{-1}+$
10	7.6937×10^{-4}	$1.0079 \times 10^{-3}+$	1.9885×10^{-2}	$2.0125 \times 10^{-2}=$	8.1026×10^{-1}	$8.0704 \times 10^{-1}+$
11	7.8789×10^{-4}	$1.0368 \times 10^{-3}+$	2.0765×10^{-2}	$2.1170 \times 10^{-2}=$	8.2136×10^{-1}	$8.1340 \times 10^{-1}+$
12	7.7134×10^{-4}	$1.0219 \times 10^{-3}+$	1.7870×10^{-2}	$1.9090 \times 10^{-2}+$	7.9085×10^{-1}	$7.8832 \times 10^{-1}=$
13	8.7732×10^{-4}	$1.0757 \times 10^{-3}+$	1.8085×10^{-2}	$1.9780 \times 10^{-2}+$	8.4326×10^{-1}	$8.3691 \times 10^{-1}+$
14	7.9210×10^{-4}	$1.0462 \times 10^{-3}+$	1.8105×10^{-2}	$1.9375 \times 10^{-2}+$	8.3489×10^{-1}	$8.2903 \times 10^{-1}+$
15	7.4785×10^{-4}	$1.0317 \times 10^{-3}+$	1.9610×10^{-2}	$2.0175 \times 10^{-2}=$	7.6262×10^{-1}	$7.5633 \times 10^{-1}+$
16	7.0934×10^{-4}	$1.0484 \times 10^{-3}+$	1.8290×10^{-2}	$2.0220 \times 10^{-2}+$	7.9199×10^{-1}	$7.8373 \times 10^{-1}+$
17	6.9929×10^{-4}	$1.0574 \times 10^{-3}+$	1.9605×10^{-2}	$2.0405 \times 10^{-2}=$	7.9177×10^{-1}	$7.8538 \times 10^{-1}+$
18	7.0551×10^{-4}	$1.0221 \times 10^{-3}+$	1.7440×10^{-2}	$1.8970 \times 10^{-2}+$	8.0302×10^{-1}	$7.9714 \times 10^{-1}+$
19	8.1797×10^{-4}	$1.0681 \times 10^{-3}+$	1.8055×10^{-2}	$2.0265 \times 10^{-2}+$	8.1730×10^{-1}	$8.0498 \times 10^{-1}+$
20	8.2308×10^{-4}	$1.1589 \times 10^{-3}+$	1.8810×10^{-2}	$2.0905 \times 10^{-2}+$	8.0723×10^{-1}	$7.9788 \times 10^{-1}+$
21	7.6813×10^{-4}	$1.1156 \times 10^{-3}+$	1.8935×10^{-2}	$2.1415 \times 10^{-2}+$	8.0504×10^{-1}	$7.9633 \times 10^{-1}+$

CMOMA的优势得到了增强。这主要归因于竞争机制，使得算法能够为局部搜索算子找到合适的选择概率。综合以上所有分析，可验证CMOMA优于FMOMA，设计的竞争机制有助于提高性能。

5.4. 与其他算法的比较

为了进一步评估CMOMA的性能，将其与经典和新兴的多目标算法（即NSGA-II、SPEA-2和MOGWO）进行了比较。实验设置与第5.3节所述相同。此外，执行田口方法（Taguchi method）来调整比较算法的参数以获得最佳性能。对于NSGA-II，交叉概率为1，变异概率为0.02；对于SPEA-2，交叉概率为1，变异概率为0.03。MOGWO不需要预定义的参数。

表7至表10显示了GD、IGD、HV和SC的比较结果。

表6 CMOMA和FMOMA在所有测试实例上的SC均值比较

Test instance	*C*(CMOMA, FMOMA)	*C*(FMOMA, CMOMA)	Test instance	*C*(CMOMA, FMOMA)	*C*(FMOMA, CMOMA)
1	27.75%	17.95%	12	37.35%	11.25%
2	28.55%	14.43%	13	42.05%	11.63%
3	31.23%	13.70%	14	41.00%	12.83%
4	31.38%	16.03%	15	36.05%	9.75%
5	34.40%	13.25%	16	43.93%	8.58%
6	30.20%	14.65%	17	44.25%	6.35%
7	28.88%	16.58%	18	44.05%	8.65%
8	32.78%	10.33%	19	41.53%	10.78%
9	33.00%	11.00%	20	50.03%	8.10%
10	33.38%	11.70%	21	53.33%	8.18%
11	41.13%	9.33%			

表7 CMOMA、NSGA-II、SPEA-2和MOGWO在所有测试实例上的GD均值和两样本*t*检验结果

Test instance	CMOMA	NSGA-II	SPEA-2	MOGWO
1	1.1402×10^{-3}	$1.1716 \times 10^{-3}=$	$1.8564 \times 10^{-3}+$	$2.5177 \times 10^{-3}+$
2	8.9269×10^{-4}	$1.1832 \times 10^{-3}+$	$1.5757 \times 10^{-3}+$	$2.3667 \times 10^{-3}+$
3	9.2624×10^{-4}	$1.2792 \times 10^{-3}+$	$1.9651 \times 10^{-3}+$	$2.6771 \times 10^{-3}+$
4	1.0683×10^{-3}	$1.3604 \times 10^{-3}+$	$1.9080 \times 10^{-3}+$	$3.1077 \times 10^{-3}+$
5	9.0086×10^{-4}	$1.3917 \times 10^{-3}+$	$2.1230 \times 10^{-3}+$	$3.7516 \times 10^{-3}+$
6	8.8471×10^{-4}	$1.2978 \times 10^{-3}+$	$1.6351 \times 10^{-3}+$	$2.7415 \times 10^{-3}+$
7	1.0415×10^{-3}	$1.3033 \times 10^{-3}+$	$1.8277 \times 10^{-3}+$	$4.1431 \times 10^{-3}+$
8	7.9311×10^{-4}	$8.9139 \times 10^{-4}+$	$1.0697 \times 10^{-3}+$	$1.6005 \times 10^{-3}+$
9	8.7197×10^{-4}	$1.1159 \times 10^{-3}+$	$1.3992 \times 10^{-3}+$	$2.2571 \times 10^{-3}+$
10	7.6937×10^{-4}	$9.7697 \times 10^{-4}+$	$1.1289 \times 10^{-3}+$	$2.0441 \times 10^{-3}+$
11	7.8789×10^{-4}	$1.0573 \times 10^{-3}+$	$1.2310 \times 10^{-3}+$	$2.1723 \times 10^{-3}+$
12	7.7134×10^{-4}	$9.5573 \times 10^{-4}+$	$1.2966 \times 10^{-3}+$	$2.2548 \times 10^{-3}+$
13	8.7732×10^{-4}	$1.0020 \times 10^{-3}+$	$1.5837 \times 10^{-3}+$	$3.1690 \times 10^{-3}+$
14	7.9210×10^{-4}	$1.0903 \times 10^{-3}+$	$1.2610 \times 10^{-3}+$	$2.4987 \times 10^{-3}+$
15	7.4785×10^{-4}	$1.0657 \times 10^{-3}+$	$1.2482 \times 10^{-3}+$	$1.9489 \times 10^{-3}+$
16	7.0934×10^{-4}	$1.0747 \times 10^{-3}+$	$1.1446 \times 10^{-3}+$	$1.8948 \times 10^{-3}+$
17	6.9929×10^{-4}	$9.4979 \times 10^{-4}+$	$1.1071 \times 10^{-3}+$	$2.2707 \times 10^{-3}+$
18	7.0551×10^{-4}	$9.8959 \times 10^{-4}+$	$1.2360 \times 10^{-3}+$	$2.4601 \times 10^{-3}+$
19	8.1797×10^{-4}	$1.1376 \times 10^{-3}+$	$1.2641 \times 10^{-3}+$	$2.2323 \times 10^{-3}+$
20	8.2308×10^{-4}	$1.1959 \times 10^{-3}+$	$1.4157 \times 10^{-3}+$	$2.7575 \times 10^{-3}+$
21	7.6813×10^{-4}	$1.1232 \times 10^{-3}+$	$1.1994 \times 10^{-3}+$	$2.6607 \times 10^{-3}+$

可以看出，CMOMA在所有指标中都取得了最好的结果。关于GD，CMOMA在21个实例中的20个实例上显著优于NSGA-II，并且在所有实例上都显著优于SPEA-2和MOGWO。在IGD方面，CMOMA均显著优于比较算法。NSGA-II和SPEA-2的性能相当，超过了MOGWO。然而，它们的结果均差于CMOMA，尤其是在大型实例上。对于HV，CMOMA在所有实例上的结果都保持在0.8左右，这意味着CMOMA的非支配解始终覆盖了大部分的目标空间。相反，NSGA-II、SPEA-2和MOGWO的性能较差，在中型和大型实例上可以看到显著的性能下降。在质量比较方面，CMOMA具有巨大的优势。CMOMA的解集中解支配了SPEA2和MOGWO解集中绝大部分解。除了第一个实例之外，CMOMA的结果明显好于NSGA-II的结果，特别是在大型实例上。

由上述分析可以得出，随着实例大小的增加，CMOMA的优势变得更加明显。为了进一步探索验证此结论，图8显示了每种算法在不同规模实例上随机运行获得的非支配解。在小型实例中，NSGA-II达到了有限的帕累托前沿，并呈现出相对良好的分布，超过了SPEA-2。然而，在中型和大型实例上，NSGA-II的性能明显下降。NSGA-II、SPEA-2和MOGWO的解决方案接近且集中在一个小区域，未能充分探索目标空间。相比之下，CMOMA在所

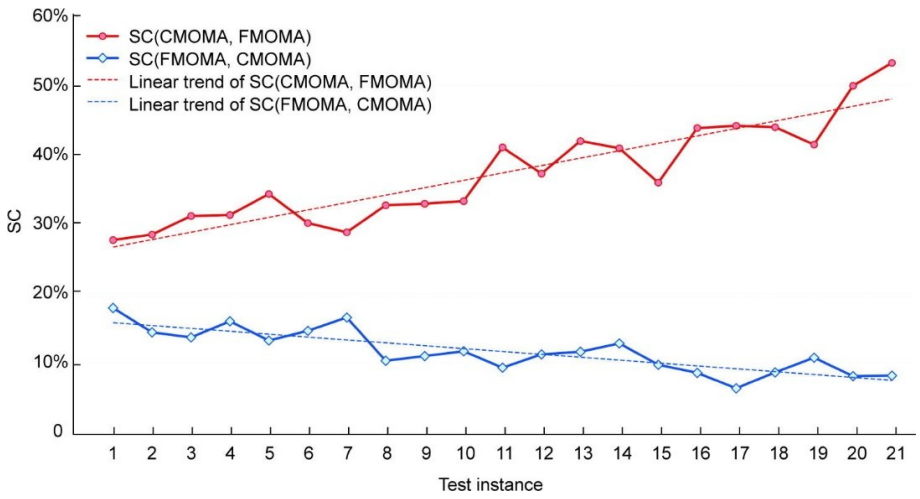

图7. CMOMA和FMOMA在所有测试实例上的SC结果。

表8 CMOMA、NSGA-II、SPEA-2和MOGWO在所有测试实例上的IGD均值和两样本t检验结果

Test instance	CMOMA	NSGA-II	SPEA-2	MOGWO
1	2.3340×10^{-2}	$7.2065\times10^{-2}+$	$1.0358\times10^{-1}+$	$1.5891\times10^{-1}+$
2	2.1670×10^{-2}	$6.1960\times10^{-2}+$	$9.6220\times10^{-2}+$	$1.4701\times10^{-1}+$
3	2.0485×10^{-2}	$5.6040\times10^{-2}+$	$8.9475\times10^{-2}+$	$1.4169\times10^{-1}+$
4	1.9490×10^{-2}	$6.0020\times10^{-2}+$	$8.8250\times10^{-2}+$	$1.3948\times10^{-1}+$
5	2.0115×10^{-2}	$7.9250\times10^{-2}+$	$1.0278\times10^{-1}+$	$1.5871\times10^{-1}+$
6	1.7660×10^{-2}	$5.7635\times10^{-2}+$	$9.4770\times10^{-2}+$	$1.5045\times10^{-1}+$
7	1.7275×10^{-2}	$7.0960\times10^{-2}+$	$8.8365\times10^{-2}+$	$1.5922\times10^{-1}+$
8	1.9570×10^{-2}	$1.0879\times10^{-1}+$	$1.2709\times10^{-1}+$	$1.8022\times10^{-1}+$
9	1.9740×10^{-2}	$1.3424\times10^{-1}+$	$1.4818\times10^{-1}+$	$2.0281\times10^{-1}+$
10	1.9885×10^{-2}	$1.4195\times10^{-1}+$	$1.5176\times10^{-1}+$	$2.0302\times10^{-1}+$
11	2.0765×10^{-2}	$1.3806\times10^{-1}+$	$1.4424\times10^{-1}+$	$1.9940\times10^{-1}+$
12	1.7870×10^{-2}	$1.5765\times10^{-1}+$	$1.5572\times10^{-1}+$	$2.1148\times10^{-1}+$
13	1.8085×10^{-2}	$1.4425\times10^{-1}+$	$1.4284\times10^{-1}+$	$1.9310\times10^{-1}+$
14	1.8105×10^{-2}	$1.5876\times10^{-1}+$	$1.5185\times10^{-1}+$	$2.0712\times10^{-1}+$
15	1.9610×10^{-2}	$1.6540\times10^{-1}+$	$1.6076\times10^{-1}+$	$2.1196\times10^{-1}+$
16	1.8290×10^{-2}	$1.8027\times10^{-1}+$	$1.7266\times10^{-1}+$	$2.2438\times10^{-1}+$
17	1.9605×10^{-2}	$1.8536\times10^{-1}+$	$1.6972\times10^{-1}+$	$2.2557\times10^{-1}+$
18	1.7440×10^{-2}	$1.7498\times10^{-1}+$	$1.7019\times10^{-1}+$	$2.1648\times10^{-1}+$
19	1.8055×10^{-2}	$1.9341\times10^{-1}+$	$1.8550\times10^{-1}+$	$2.3061\times10^{-1}+$
20	1.8810×10^{-2}	$1.7300\times10^{-1}+$	$1.6607\times10^{-1}+$	$2.0930\times10^{-1}+$
21	1.8935×10^{-2}	$1.8195\times10^{-1}+$	$1.7194\times10^{-1}+$	$2.1275\times10^{-1}+$

表9 CMOMA、NSGA-II、SPEA-2和MOGWO在所有测试实例上的HV均值和两样本t检验结果

Test instance	CMOMA	NSGA-II	SPEA-2	MOGWO
1	7.9977×10^{-1}	$7.2323\times10^{-1}+$	$6.4637\times10^{-1}+$	$5.6612\times10^{-1}+$
2	8.2696×10^{-1}	$7.3936\times10^{-1}+$	$6.5293\times10^{-1}+$	$5.8092\times10^{-1}+$
3	8.1010×10^{-1}	$7.2721\times10^{-1}+$	$6.4423\times10^{-1}+$	$5.7386\times10^{-1}+$
4	8.5031×10^{-1}	$7.5772\times10^{-1}+$	$6.8929\times10^{-1}+$	$6.0273\times10^{-1}+$
5	8.5315×10^{-1}	$7.6331\times10^{-1}+$	$7.0096\times10^{-1}+$	$6.1010\times10^{-1}+$
6	8.5614×10^{-1}	$7.6245\times10^{-1}+$	$6.7799\times10^{-1}+$	$5.8590\times10^{-1}+$
7	8.8252×10^{-1}	$7.5996\times10^{-1}+$	$7.0769\times10^{-1}+$	$5.9014\times10^{-1}+$
8	7.7647×10^{-1}	$5.8710\times10^{-1}+$	$5.4731\times10^{-1}+$	$4.7669\times10^{-1}+$
9	8.0296×10^{-1}	$6.1238\times10^{-1}+$	$5.6919\times10^{-1}+$	$5.0286\times10^{-1}+$
10	8.1026×10^{-1}	$6.0328\times10^{-1}+$	$5.6409\times10^{-1}+$	$5.0319\times10^{-1}+$
11	8.2136×10^{-1}	$6.0780\times10^{-1}+$	$5.7310\times10^{-1}+$	$5.0382\times10^{-1}+$
12	7.9085×10^{-1}	$5.7480\times10^{-1}+$	$5.4932\times10^{-1}+$	$4.8258\times10^{-1}+$
13	8.4326×10^{-1}	$5.8471\times10^{-1}+$	$5.6201\times10^{-1}+$	$4.9909\times10^{-1}+$
14	8.3489×10^{-1}	$5.9109\times10^{-1}+$	$5.8402\times10^{-1}+$	$5.0734\times10^{-1}+$
15	7.6262×10^{-1}	$5.2343\times10^{-1}+$	$5.1068\times10^{-1}+$	$4.5009\times10^{-1}+$
16	7.9199×10^{-1}	$5.2316\times10^{-1}+$	$5.1511\times10^{-1}+$	$4.6096\times10^{-1}+$
17	7.9177×10^{-1}	$5.3436\times10^{-1}+$	$5.2781\times10^{-1}+$	$4.6419\times10^{-1}+$
18	8.0302×10^{-1}	$5.2764\times10^{-1}+$	$5.1421\times10^{-1}+$	$4.6993\times10^{-1}+$
19	8.1730×10^{-1}	$5.2730\times10^{-1}+$	$5.2423\times10^{-1}+$	$4.7525\times10^{-1}+$
20	8.0723×10^{-1}	$5.1571\times10^{-1}+$	$5.1236\times10^{-1}+$	$4.5845\times10^{-1}+$
21	8.0504×10^{-1}	$5.0620\times10^{-1}+$	$5.0290\times10^{-1}+$	$4.5930\times10^{-1}+$

表10 CMOMA与NSGA-II、SPEA-2和MOGWO在所有测试实例上的SC均值比较

Test instance	C(CMOMA, NSGA-II)	C(NSGA-II, CMOMA)	C(CMOMA, SPEA2)	C(SPEA2, CMOMA)	C(CMOMA, MOGWO)	C(MOGWO, CMOMA)
1	20.93%	19.35%	55.43%	2.70%	89.64%	0.13%
2	32.73%	11.60%	54.65%	3.78%	92.09%	0.13%
3	39.73%	11.03%	75.53%	1.18%	97.25%	0.05%
4	39.53%	9.20%	63.05%	2.48%	95.83%	0.15%
5	45.25%	8.85%	66.08%	2.15%	96.74%	0.05%
6	55.08%	8.35%	65.93%	3.15%	98.28%	0.03%
7	47.53%	8.28%	66.45%	3.00%	99.51%	0
8	38.93%	6.25%	57.23%	1.35%	97.41%	0
9	56.88%	3.08%	77.70%	0.23%	99.86%	0
10	50.55%	4.78%	68.30%	1.05%	99.14%	0
11	61.75%	1.95%	84.78%	0.40%	100.00%	0
12	55.30%	3.68%	83.31%	0.65%	99.77%	0
13	49.60%	3.03%	79.80%	0.45%	100.00%	0
14	56.03%	2.70%	81.28%	0.45%	100.00%	0
15	57.63%	1.55%	77.43%	0.43%	99.37%	0
16	72.15%	0.73%	86.78%	0.18%	99.93%	0
17	70.30%	0.80%	89.95%	0.05%	100.00%	0
18	61.80%	1.08%	92.90%	0.05%	99.87%	0
19	63.74%	0.98%	83.06%	0.35%	100.00%	0
20	70.45%	0.58%	90.90%	0.05%	100.00%	0
21	82.11%	0.45%	98.20%	0.00%	100.00%	0

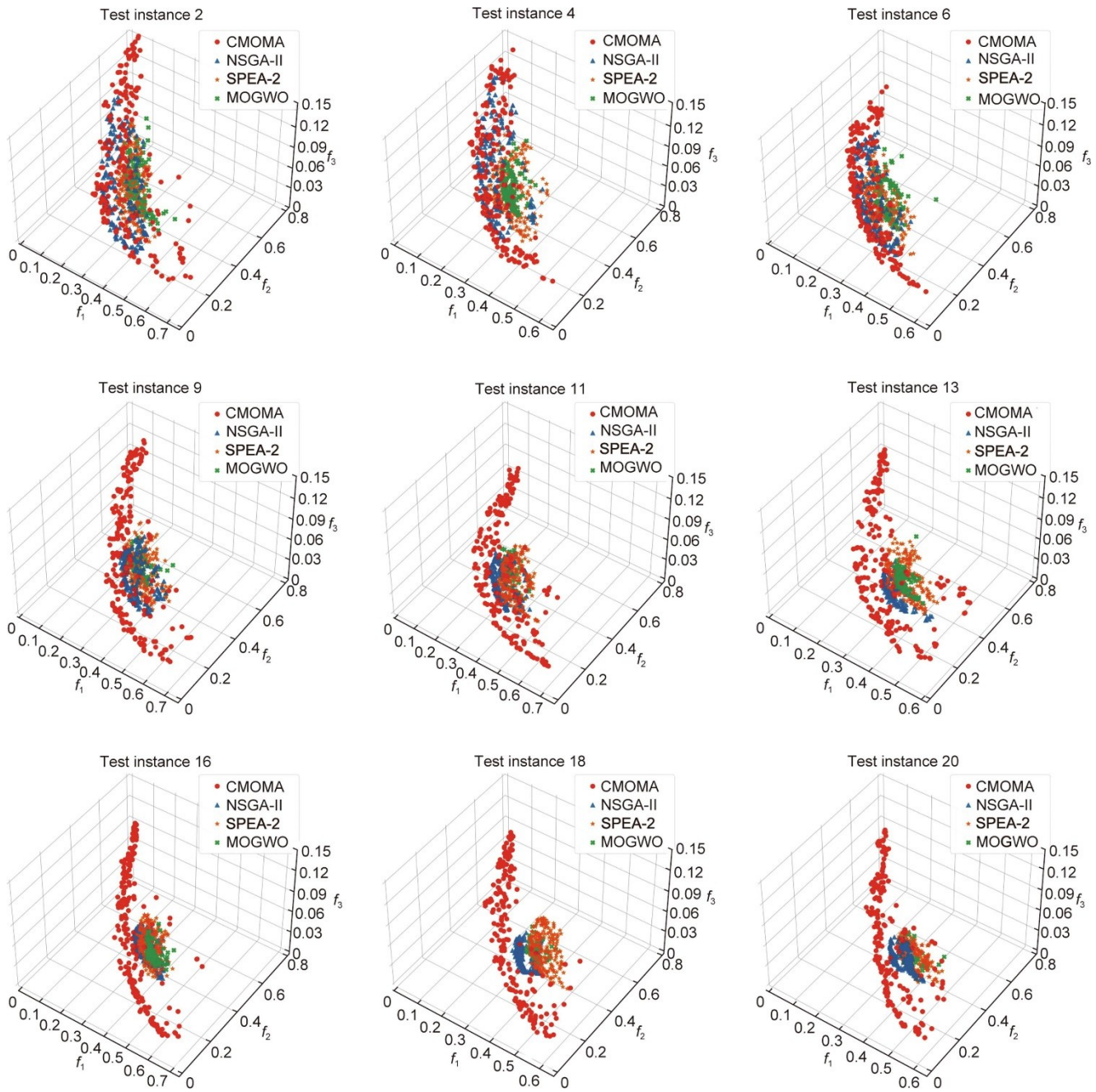

图 8. 通过在小型测试实例（2、4 和 6）、中型测试实例（9、11 和 13）和大型测试实例（16、18 和 20）上随机运行 CMOMA、NSGA-II、SPEA-2 和 MOGWO 获得的非主导解。

有实例上的解占据了很大的空间，并覆盖了比较算法的大部分解。这种优势主要源于局部搜索算子和竞争机制，这使得算法能够充分而高效地搜索。

本文通过研究任务的子任务按顺序完成的模式，说明了所提出的 DMSCO 方法的有效性。这是一种基本模式，可以简单地扩展至其他模式。例如，为了适应存在并行子任务的模式，只需进行两处修改。首先，将公式（20）$F = F_l$ 转换为 $F = \max_{CS_i \in CS}(F_i)$ 和公式（21）$B_i \geq F_{i\text{-}1} \geq 0$ 转换为 $B_i \geq F_l \geq 0$，其中 l 是子任务的索引，子任务 ST_i 应该在 ST_l 之后开始（因为在 ST_i 之前可能存在多个具有并行关系的前序 ST_l）。其次，在编码方案中，确保子任务放置位置在其所有前序子任务之后。通过这种模式，本文展示了 DMSCO 如何对服务协作方案进行双维度扩展，同时能够为其他模式提供参考。

6. 结论

随着工业互联网平台在制造业中的应用，高质量的制

造服务协作方案必不可少。本文提出了一种双维度的服务协作优化方法来优化服务选择和数量分配，以提高解决方案的质量。在本研究中，制造服务在三个粒度上进行建模，并讨论了它们的组件和结构以确保协作方案的可行性。与以前的工作不同，资源制造服务的可用性被定义为与之相关的物理资源的可访问周期序列，复杂服务的可用性受组件服务及其关系的影响。然后制定DMSCO模型，以同时优化服务选择和数量分配。设计了一种嵌入竞争机制的CMOMA对模型求解，同时进行了大量实验，结果表明所提出算法的优越性。具体而言，CMOMA在所有考虑的指标中都优于没有竞争机制的FMOMA，从而验证了该机制的有效性。与NSGA-II、SPEA-2和MOGWO相比，CMOMA也表现出优异的性能。此外，随着实例大小的增加，CMOMA优于比较算法的优势变得明显。

本文旨在为工业互联网平台引入一种新型的双维度协作模式，从而为利益相关者提供新的见解。本研究基于单独的服务优选进行扩展，所提的DMSCO易于相关从业者理解，并与现有流程兼容。同时，本研究中的实验结果揭示了所提的方法在复杂生产过程和大批量生产场景中的应用潜力，如汽车、设备和家用电器制造。生产过程中的每一个环节可以平均分配给多个制造服务，从而提高整体服务协作的性能。然而，需要注意的是，本文中的制造服务建模侧重于物理加工资源，而工业互联网平台也包含其他类型的资源，如软件和物流资源。在未来研究中，将通过真实场景和实验验证与评估所提出的方法。同时，可对如何加强制造服务建模的表达能力以及构建相应的服务协作模型展开进一步研究。此外，本文主要关注满足客户需求的服务协作方案。由于服务提供商的利益也至关重要，因此关于如何同时提高双方的满意度值得研究。

致谢

本工作由国家自然科学基金（51905396）和中国留学基金委员会资助。

Compliance with ethics guidelines

Shibao Pang, Shunsheng Guo, Xi Vicent Wang, Lei Wang, and Lihui Wang declare that they have no conflict of interest or financial conflicts to disclose.

References

[1] Zhou J, Li P, Zhou Y, Wang B, Zang J, Meng L. Toward new-generation intelligent manufacturing. Engineering 2018;4(1):11–20.
[2] Zhong RY, Xu X, Klotz E, Newman ST. Intelligent manufacturing in the context of Industry 4.0: a review. Engineering 2017;3(5):616–30.
[3] Tao F, Qi Q, Wang L, Nee AYC. Digital twins and cyber–physical systems toward smart manufacturing and Industry 4.0: correlation and comparison. Engineering 2019;5(4):653–61.
[4] Wang B. The future of manufacturing: a new perspective. Engineering 2018; 4(5):722–8.
[5] Cheng Y, Gao Y, Wang L, Tao F, Wang QG. Graph-based operational robustness analysis of Industrial Internet of Things platform for manufacturing service collaboration. Int J Prod Res 2022;2022:1–28.
[6] Wang L, Gao T, Zhou B, Tang H, Xiang F. Manufacturing service recommendation method toward Industrial Internet platform considering the cooperative relationship among enterprises. Expert Syst Appl 2022;192:116391.
[7] Song Y, Lin J, Tang M, Dong S. An Internet of Energy Things based on wireless LPWAN. Engineering 2017;3(4):460–6.
[8] Lin H, Garg S, Hu J, Wang X, Piran MJ, Hossain MS. Data fusion and transfer learning empowered granular trust evaluation for Internet of Things. Inf Fusion 2022;78:149–57.
[9] Zhou H, Yang C, Sun Y. Intelligent ironmaking optimization service on a cloud computing platform by digital twin. Engineering 2021;7(9):1274–81.
[10] Wang XV, Wang L, Mohammed A, Givehchi M. Ubiquitous manufacturing system based on cloud: a robotics application. Robot Comput Integr Manuf 2017;45:116–25.
[11] Cheng Y, Xie Y, Wang D, Tao F, Ji P. Manufacturing services scheduling with supply-demand dual dynamic uncertainties toward Industrial Internet platforms. IEEE Trans Industr Inform 2021;17(5):2997–3010.
[12] Zhang X, Ming X. A comprehensive industrial practice for Industrial Internet Platform (IIP): general model, reference architecture, and industrial verification. Comput Ind Eng 2021;158:107426.
[13] Li P, Cheng Y, Tao F. Failures detection and cascading analysis of manufacturing services collaboration toward Industrial Internet platforms. J Manuf Syst 2020;57:169–81.
[14] Wang Y, Zhang Y, Tao F, Chen T, Cheng Y, Yang S. Logistics-aware manufacturing service collaboration optimisation towards Industrial Internet platform. Int J Prod Res 2019;57(12):4007–26.
[15] Li P, Cheng Y, Song W, Tao F. Manufacturing services collaboration: connotation, framework, key technologies, and research issues. Int J Adv Manuf Technol 2020;110(9–10):2573–89.
[16] Bouzary H, Chen FF. A classification-based approach for integrated service matching and composition in cloud manufacturing. Robot Comput Integr Manuf 2020;66:101989.
[17] Yang B, Wang S, Li S, Jin T. A robust service composition and optimal selection method for cloud manufacturing. Int J Prod Res 2022;60(4):1134–52.
[18] Liu Y, Wang L, Wang XV, Xu X, Zhang L. Scheduling in cloud manufacturing: state-of-the-art and research challenges. Int J Prod Res 2019; 57(15 – 16): 4854–79.
[19] Shi SY, Mo R, Yang HC, Chang ZY, Chen ZF. An implementation of modelling resource in a manufacturing grid for resource sharing. Int J Comput Integr Manuf 2007;20(2–3):169–77.
[20] Vichare P, Nassehi A, Kumar S, Newman ST. A unified manufacturing resource model for representing CNC machining systems. Robot Comput Integr Manuf 2009;25(6):999–1007.
[21] Ameri F, McArthur C. Semantic rule modelling for intelligent supplier discovery. Int J Comput Integr Manuf 2014;27(6):570–90.
[22] Wang XV, Wang L. A cloud-based production system for information and service integration: an Internet of Things case study on waste electronics. Enterprise Inf Syst 2017;11(7):952–68.
[23] Li H, Chan KCC, Liang M, Luo X. Composition of resource-service chain for cloud manufacturing. IEEE Trans Ind Inform 2016;12(1):211–29.
[24] Wu L. Resource virtualization model in cloud manufacturing. Adv Mat Res 2010;143–144:1250–3.
[25] Liu N, Li X, Shen W. Multi-granularity resource virtualization and sharing strategies in cloud manufacturing. J Netw Comput Appl 2014;46:72–82.
[26] Yu C, Zhang W, Xu X, Ji Y, Yu S. Data mining based multi-level aggregate service planning for cloud manufacturing. J Intell Manuf 2018;29(6):1351–61.
[27] Zhang Z, Zhang Y, Lu J, Xu X, Gao F, Xiao G. CMfgIA: a cloud manufacturing application mode for industry alliance. Int J Adv Manuf Technol 2018;98(9–12):

2967–85.
[28] Li H, Weng S, Tong J, He T, Chen W, Sun M, et al. Composition of resource-service chain based on evolutionary algorithm in distributed cloud manufacturing systems. IEEE Access 2020;8:19911–20.
[29] Lu Y, Xu X. A semantic web-based framework for service composition in a cloud manufacturing environment. J Manuf Syst 2017;42:69–81.
[30] Ren M, Ren L, Jain H. Manufacturing service composition model based on synergy effect: a social network analysis approach. Appl Soft Comput 2018;70: 288–300.
[31] Zhou L, Zhang L, Horn BKP. Collaborative optimization for logistics and processing services in cloud manufacturing. Robot Comput Integr Manuf 2021; 68:102094.
[32] Wu Y, Jia G, Cheng Y. Cloud manufacturing service composition and optimal selection with sustainability considerations: a multi-objective integer bi-level multi-follower programming approach. Int J Prod Res 2020;58(19):6024–42.
[33] Wang Y, Wang S, Gao S, Guo X, Yang B. Adaptive multi-objective service composition reconfiguration approach considering dynamic practical constraints in cloud manufacturing. Knowl Base Syst 2021;234:107607.
[34] Tao F, LaiLi Y, Xu L, Zhang L. FC-PACO-RM: a parallel method for service composition optimal-selection in cloud manufacturing system. IEEE Trans Industr Inform 2013;9(4):2023–33.
[35] Wang L, Liu Z, Liu A, Tao F. Artificial intelligence in product lifecycle management. Int J Adv Manuf Technol 2021;114(3–4):771–96.
[36] Bouzary H, Chen FF. A hybrid grey wolf optimizer algorithm with evolutionary operators for optimal QoS-aware service composition and optimal selection in cloud manufacturing. Int J Adv Manuf Technol 2019;101(9–12):2771–84.
[37] Akbaripour H, Houshmand M. Service composition and optimal selection in cloud manufacturing: landscape analysis and optimization by a hybrid imperialist competitive and local search algorithm. Neural Comput Appl 2020; 32(15):10873–94.
[38] Zhang S, Xu Y, Zhang W. Multitask-oriented manufacturing service composition in an uncertain environment using a hyper-heuristic algorithm. J Manuf Syst 2021;60:138–51.
[39] Zhou J, Yao X. A hybrid approach combining modified artificial bee colony and cuckoo search algorithms for multi-objective cloud manufacturing service composition. Int J Prod Res 2017;55(16):4765–84.
[40] Zhang Y, Tao F, Liu Y, Zhang P, Cheng Y, Zuo Y. Long/short-term utility aware optimal selection of manufacturing service composition toward Industrial Internet platforms. IEEE Trans Industr Inform 2019;15(6):3712–22.
[41] Xue X, Wang S, Lu B. Manufacturing service composition method based on networked collaboration mode. J Netw Comput Appl 2016;59:28–38.
[42] Zhang S, Xu S, Huang X, Zhang W, Chen M. Networked correlation-aware manufacturing service supply chain optimization using an extended artificial bee colony algorithm. Appl Soft Comput 2019;76:121–39.
[43] Zhu LN, Li PH, Zhou XL. IHDETBO: a novel optimization method of multi-batch subtasks parallel-hybrid execution cloud service composition for cloud manufacturing. Complexity 2019;2019:7438710.
[44] Ding T, Yan G, Lei Y, Xu X. A niching behaviour-based algorithm for multi-level manufacturing service composition optimal-selection. J Ambient Intell Humaniz Comput 2020;11(3):1177–89.
[45] Zhang Y, Xi D, Li R, Sun S. Task-driven manufacturing cloud service proactive discovery and optimal configuration method. Int J Adv Manuf Technol 2016; 84(1–4):29–45.
[46] Ma L, Li J, Lin Q, Gong M, Coello Coello CA, Ming Z. Cost-aware robust control of signed networks by using a memetic algorithm. IEEE Trans Cybern 2020;50(10):4430–43.
[47] Mirjalili S, Mirjalili SM, Lewis A. Grey wolf optimizer. Adv Eng Softw 2014; 69:46–61.
[48] Deb K, Pratap A, Agarwal S, Meyarivan T. A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Trans Evol Comput 2002;6(2):182–97.
[49] Zitzler E, Laumanns M, Thiele L. SPEA2: improving the strength Pareto evolutionary algorithm. Report. Zürich: Eidgenössische Technische Hochschule (ETH) Zürich; 2001 May. Report No.: TIK-Report 103.
[50] Mirjalili S, Saremi S, Mirjalili SM, Coelho LDS. Multi-objective grey wolf optimizer: a novel algorithm for multi-criterion optimization. Expert Syst Appl 2016;47:106–19.