Contents lists available at ScienceDirect

Engineering

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

Research
Intelligent Manufacturing—Article

基于边缘计算的软件定义云制造和柔性资源调度研究

杨晨[a]，廖方茵[b,c]，兰舒琳[d,*]，王力翚[e]，沈卫明[f]，黄国全[g]

[a] *School of Cyberspace Science and Technology, Beijing Institute of Technology, Beijing 100081, China*
[b] *School of Computer Science and Technology, Beijing Institute of Technology, Beijing 100081, China*
[c] *School of Mathematics and Computer Science and Technology, Yan'an University, Yan'an 716000, China*
[d] *School of Economics and Management, University of Chinese Academy of Sciences, Beijing 100190, China*
[e] *Department of Production Engineering, KTH Royal Institute of Technology, Stockholm 10044, Sweden*
[f] *State Key Lab of Digital Manufacturing Equipment and Technology, Huazhong University of Science and Technology, Wuhan 430074, China*
[g] *Department of Industrial and Manufacturing Systems Engineering, The University of Hong Kong, Hong Kong 999077, China*

ARTICLE INFO

Article history:
Received 17 December 2020
Revised 12 August 2021
Accepted 12 August 2021
Available online 25 November 2021

关键词
云制造
边缘计算
软件定义网络
工业物联网
工业 4.0

摘要

本文研究的重点是在云制造环境中实现快速重构、实现灵活的资源调度、开发资源潜力以应对各种变化。因此，本文首先提出了一种新的基于云和软件定义网络（SDN）的制造模型——软件定义云制造（SDCM），该模型将控制逻辑从自动化硬件转移到软件上。这种转变意义重大，因为软件可以充当制造系统的"大脑"，并且可以轻松更改或更新以支持快速系统重新配置、运营和演进。随后，边缘计算被引入，以接近终端的计算和存储能力来补充云。另一个关键问题是管理由不同服务质量（QoS）要求的大量物联网（IoT）数据传输而导致的严重网络拥塞。基于SDCM的虚拟化和灵活的网络能力，本研究形式化了面向复杂制造任务集的时间敏感性数据流量控制问题，并考虑了子任务分配和数据路由路径选择。为了解决这一优化问题，提出了一种将遗传算法（GA）、Dijkstra最短路径算法和排队算法相结合的方法。实验结果表明，该方法能有效地防止网络拥塞，减少SDCM中的总通信延迟。

1. 引言

云制造（CMfg）[1]虚拟化制造资源和功能，并构建一个超大规模的共享虚拟资源池为客户提供服务。该框架利用新一代信息与通信技术（ICT）和现代制造技术来发展工业制造。云中心通常拥有强大的存储、网络和计算资源；然而，云中的集中式处理或资源管理可能会遭遇瓶颈和很大的延迟[2]。

边缘/雾计算是一种分布式计算范式，它使计算和存储资源更接近终端设备，可以加强和补充CMfg，以提供低延迟、位置感知、移动性支持和实时分析[3–5]。由于车间生产作业通常是时间敏感的，并且涉及专有信息，因此关于车间任务和对象的实时数据可以在边缘节点上进行处理，而不是发送到远程云。该框架不仅可以避免网络拥塞，还可以促进实时响应和数据保护。因此，应探索边缘/雾计算在CMfg中的应用。

此外，资源调度在CMfg中发挥着关键作用，它利用云中集成和汇集的资源来满足客户的需求。合理的资源调

* Corresponding author.
E-mail address: lanshulin@ucas.ac.cn (S. Lan).

英文原文：Engineering 2023, 22(3): 60–70
引用本文：Chen Yang, Fangyin Liao, Shulin Lan, Lihui Wang, Weiming Shen, George Q. Huang. Flexible Resource Scheduling for Software-Defined Cloud Manufacturing with Edge Computing. *Engineering*, https://doi.org/10.1016/j.eng.2021.08.022

度可以提高效率，减少资源消耗，并且增强CMfg以提供高质量的服务（QoS）。在云计算中，资源调度是指对计算资源、网络资源和存储资源的高效分配，而制造领域的资源调度主要集中于将任务分配给生产机器，以执行不同类型的生产任务。许多因素都在推动着这两个领域研究的深度融合和探索[3]。目前，在万物互联[6]和个性化生产[7]时代，要实现制造高效率和灵活性，必须克服两大限制：

（1）在制造过程中，机床、输送机和工业机器人被静态地预配置并集成在生产线中[6]，而控制软件与执行操作的机器人硬件紧密集成。因此，重新配置、部署、优化和扩展工厂自动化以执行多品种和小批量生产[8]并管理各种干扰[9]是耗时和昂贵的。由于静态的结构和配置，这些缺点还可能阻碍更有效的调度的实现。但是，可以引入结构变化（系统重新配置）来优化资源效率。在这种情况下，将制造系统中的控制软件和执行硬件分离开来，可以实现快速的系统重构和重组，以实现最佳调度。

（2）传感器的广泛部署和制造设备的普遍联网使得大量数据在制造系统网络中生成、收集和传输。现有的研究主要集中于连接物理物体，并根据物体的实时数据做出决策[5]。然而，随着传感器和联网物体数量的增加，大量来自各种来源（如大量的生产机器）的异构原始数据（未来的泽字节）可能会导致严重的网络拥塞，并损害网络服务的整体质量。必须引入边缘计算，以适当地清理和组合不同级别的数据，以减少网络中的数据流量。此外，由于制造任务分配给不同的生产机器，在网络上产生不同的数据流，数据流量模式不稳定，因此，高效的协作生产需要对网络中机器之间的数据流进行灵活的控制[10]。因此，考虑时间敏感的数据传输，应在网络流量控制中探索具有灵活网络的软件定义网络（SDN），以减少通信延迟，提高协作效率[11]。此外，在调度中必须考虑任务分配和灵活的数据流控制。

这项工作的贡献可以总结如下。为了解决上述问题，本研究提出了一种新的基于SDN的CMfg模型[12]，包括定义、架构和原理。该模型有助于消除制造资源紧密的垂直和水平耦合，实现CMfg环境下的柔性资源调度。从网络的角度来看，传统的CMfg模型已经不能满足与泛在感知、数据交互和制造业大规模协作相关的数据通信需求。本文提出了一种新的模型来解决复杂制造任务在制造系统中的网络流量控制问题。基于软件定义云制造（SDCM）的抽象和虚拟化功能，本研究将考虑子任务分配和数据路由路径选择的时间敏感数据流量控制问题形式化。为了解决这一优化问题，本研究集成使用了遗传算法（GA）、Dijkstra路径算法和排队算法。实验结果表明，该集成方法在满足时间约束和减少总通信延迟方面是有效的。

其余的论文的组织方式如下。第2节回顾了相关文献；第3节描述了新模型的概念和参考架构；第4和第5节将网络拥塞控制问题形式化表示并介绍解决问题的方法；第6和第7节描述了实验并给出了结论。

2. 相关工作

2.1. 基于云的制造

云计算在制造业中的潜力最初是在CMfg术语下探索的[12]。Ren等[13]给出了CMfg框架的关键特征，并提出了一个4-过程的多智能体协同模型，该模型首先阐明了CMfg网络物理系统的复杂运行机制。Simeone等[14]开发了一种基于制造服务推荐系统的智能决策支持工具，通过CMfg系统向客户推荐定制的制造解决方案。Mourzis等[15]提出了一种基于云的蕴含丰富知识的框架，该框架由监控系统、知识重用机制和优化系统组成，以提高加工效率。Liu等[16]提出了一种基于深度强化学习的CMfg调度框架，并证明了该框架对在线单任务调度的有效性。然而，由于云与车间事物之间的距离很远以及网络性能不可预测[3]，以云为中心的制造架构无法支持云对网络边缘车间应用程序的实时响应。Queiroz等[17]使用多智能体系统方法，而不是集中式基于云的人工智能（AI）方法，设计可以嵌入不同数据分析功能并支持智能分散化的网络物理智能体。通过这种方式，边缘/雾计算[18]可以通过快速边缘处理能力加强基于云的制造[4–5]。He等[19]提出了一种面向云边环境的演化微服务编程框架，以实现服务系统的自适应和优化演化。针对数据驱动的智能诊断服务，提出了使用AI算法处理车间数据的其他协同云边处理方法[20–21]。提出了半监督并行深度分解机（SS-PdeepFM）模型[20]和宽深度序列模型[22]等新型工业AI模型，为质量低、噪声大的异构工业数据建立深度神经网络。例如，宽深度序列模型[22]首先在工业物联网（IoT）中实现了具有隐藏耦合关系的多维异构工业数据的跨域集成学习。Ren等[23]提出了一种新颖的云边轻量级神经网络模型，以提高算法的时间效率，同时不损失预测精度。Ren等[24]提出了一种具有协同云边智能的生成编码组进化（GCGE）算法，以提高与工业物联网相关的大规模任务分配的效率和稳定性。然而，网络资源的调度对于无缝的人与机器以及机器与机器之间的协作至关重要，但尚未在基于云的制造环境中得到广泛的研究。对于复杂的制造任务，子任务分配会影响网络链路之间的数据流量模式，应该恰当管理，以促进高效的协作制造。在这种背景下，

边缘计算可以通过计算、数据缓存和数据转发功能对车间中的制造物提供快速响应，这些功能应在灵活的资源调度中进行探索。

2.2. 软件定义的网络

制造事物之间的普遍连接、高数据吞吐量和不稳定的流量模式需要细粒度的网络资源管理，而SDN代表一种有前景的解决方案，因为它可以分离网络中的控制平面和数据平面，并通过远程SDN控制器在逻辑上集中控制[25]。SDN的主要目标是增加网络的灵活性[25]。Hu [26]提出了软件定义的工业物联网的系统架构，以确保关键架构元素的软件可定义性。Salahuddin等[27]提出了路边单元云作为计算和通信基础设施的车载云。利用SDN的深度可编程性来动态地重新配置服务和相应的数据转发信息，高效服务车联网底层需求。Naeem等[28]提出了一种基于模糊归一化神经网络的基于SDN的新型无模型自适应深度强化学习框架，以解决物联网网络中的拥塞控制问题。这些研究为研究智能、高效、响应性强的CMfg框架提供了宝贵的基础。然而，从整体角度来看，与CMfg相关的现有方法不能支持细粒度的网络资源调度（用于高效的协作制造）或功能可编程性，而这对于系统的敏捷性和快速响应至关重要[3]。因此，一个新的趋势是结合SDN、边缘计算和云计算的优势，实现更有效的网络控制和管理[2]。

3. 软件定义的云制造

3.1. 定义

过去受硬件和物流限制的制造业目前正在被重塑为主要由软件定义的活动[29]。随着物联网和网络物理系统的引入，物理事物网络化为网络实体，表明向数字世界的转变。软件，而不是硬件，已经成为许多系统的主要部分[5,11]。在此背景下，SDCM被提出，可以作为未来制造业的新基础。

SDCM是融合了SDN和其他新兴ICT的一种CMfg新模型。该模型可以通过软件定义（编程）的方式来描述、模拟、整合、配置、授权、管理、执行、加速和创新制造过程以及制造活动中的其他相关要素。

如图1所示，SDCM作为一个新型模型，利用虚拟化技术打破硬件和软件的紧密垂直耦合，将制造资源的控制逻辑与底层硬件资源分离，促进制造资源控制的逻辑集中化。通过这些特征，该模型可以实现基于软件的制造资源或系统的编程。SDCM具有以下几个优点。

首先，SDCM通过其核心使能技术（资源虚拟化和功能可编程性），可以灵活地将物理制造资源组合和分离成独立的端到端逻辑片。随后，该模型为工程师创造了一个开放的编程环境。通过功能的可编程性，可以提供和利用不同层次的软件开发工具包（SDK）来操作不同层次系统的资源[1]。这种设置可以增加制造系统加速升级和运行所需的敏捷性，促进资源共享和利用。

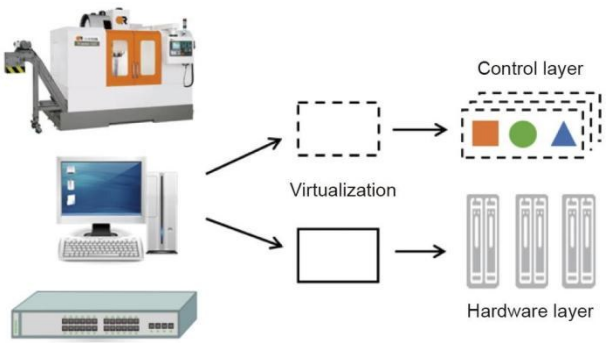

图1. 打破硬件和软件紧密的垂直耦合。

其次，在SDCM框架中，通过对软件控制器中的控制和管理逻辑进行编程，可以以用户为中心，以快速、灵活和协作的方式将集成的制造资源联网和组织起来。控制器可以对外部和内部干扰做出及时反应，并以高效和有效的方式管理制造过程。此外，消费者可以编写代码，可以自动访问、配置、协调和管理虚拟制造资源，以实现所需的功能和能力。

最后，SDCM系统更智能，因为监督自动化硬件的智能（控制逻辑）从硬件转移到软件，而软件可以通过有关制造系统和环境的信息逐步学习而变得更智能。因此，智能制造成为一个可以通过数据和智能进行自主更新、增强和改进的连续过程。

3.2. 参考架构

通过结合SDN [11]提出了SDCM的参考架构，使信息物理制造系统通过软件实现可编程性和可控性（图2）。

第一层是物理硬件的抽象层。制造机器/资源作为网络-物理接口可以在网络中进行编程，以在物理世界中提供各种功能。网络边缘的这些基本对象，如机器人和传感器被称为原子硬件。物联网和智能SDCM可以推动此类原子硬件的数字化，软件可以加强智能决策的执行，实现定制化和个性化的生产。

第二层是智能网关（GW）层。GW是可以监督和控制原子硬件的嵌入式计算机，也可以是边缘计算节点，可以将计算和数据存储设施放置在离需要它们的位置更近的地方，以减少附近数据源或车间设备的响应时间和带宽[3,5]。为了促进统一的资源管理，虚拟化技术和面向服务

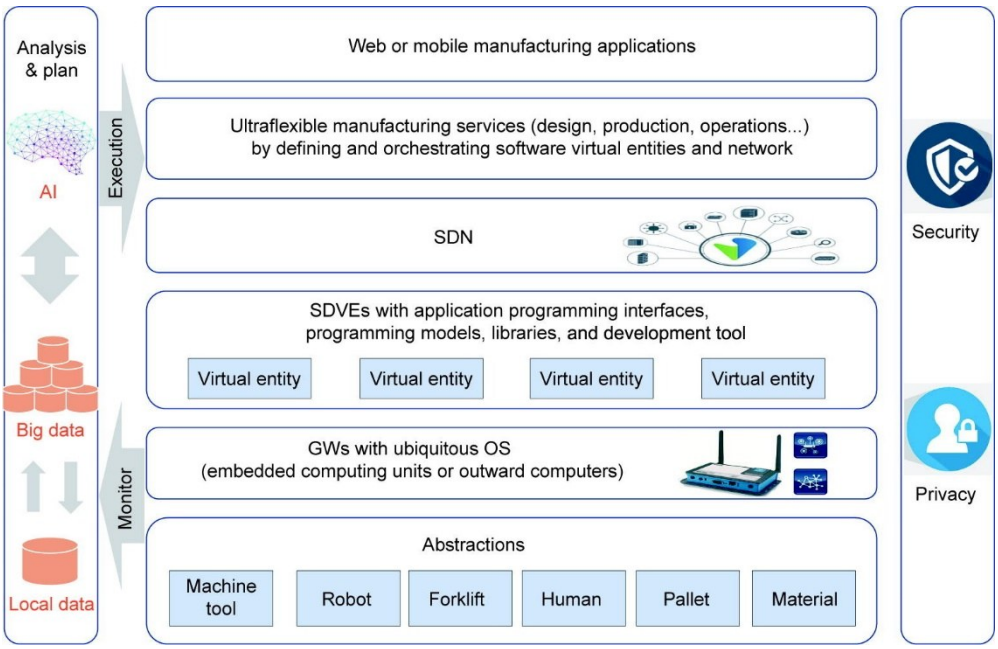

图2. SDCM参考体系架构。SDVE：软件定义的虚拟实体；GW：智能网关；OS：操作系统。

的架构可以通过抽象层和GW层将异构的原子硬件转换为网络世界中的软件定义虚拟实体（SDVE）。

第三层是SDVE层。该层由具有应用程序编程接口、编程模型、库和开发工具的SDVE组成。每个SDVE通常具有有限的硬件资源，因此应该能够很好地管理其任务列表。由于此类SDVE可以灵活定义、编程和组织，因此可以实现针对各种目标的复杂控制逻辑。例如，通过在生产机器人上构建一层机器人操作系统（OS），部署支持定制化生产的个性化模块，提高机器人的开放性和可演化性。

第四层是SDN层。SDN通过将控制平面与数据平面分离，使网络更加可编程和灵活。在该层上，SDN控制器对虚拟网络进行监管并动态调整资源分配以满足制造应用多样化的QoS要求。制造物（如SDVE）网络可以根据应用要求灵活配置，以促进高效交互和协作。例如，在（最终产品的）零件和工业机器人之间配置合适的路径。

第五层是超柔性制造服务层。为了组织SDVE并为复杂的任务形成高效的协作网络，SDN管理使网络配置更加高效，并提高了虚拟资源之间的网络性能。该框架可以满足不同QoS值的工业通信需求，促进SDVE之间的高效协作。此外，SDVE的网络可以被监控和灵活编程以满足应用需求。此外，该层还提供SDK来开发平台或应用程序。

第六层是制造应用层。在这一层，工程师和终端用户等利益相关者可以基于平台SDK或超柔性制造服务对应用软件进行编程，共同完成制造任务。

这6个横向层涉及三个重要方面。第一个方面涉及通过物联网设备或互联网从车间、工厂、供应链和物流系统收集的工业大数据。利用人工智能技术，利用大数据获取知识或信息，确保不同层次的应用都能做出明智的决策。实时数据处理在制造机器或边缘计算节点的嵌入式计算单元中实现，以实现更快的响应，而离线大数据分析在云端进行，以获得全局和全面的视图和洞察力。第二个方面，即安全和隐私，对于高度互联和开放的世界非常关键，因为普遍的感知、连接和控制可能会引起可靠性、安全性和隐私方面的重大问题。黑客可能会利用漏洞进行大规模的网络攻击[30]。由于工业大数据在网络世界中被收集和存储，这些方面可能会导致隐私和数据安全问题。因此，应该采取措施来保护工业系统和数据，考虑效率和隐私之间的平衡。最后，设备-边缘-云协作处理在SDCM中至关重要，这不仅因为需要杰出的云存储和计算能力来分析大数据并支持制造过程中的最佳决策，而且因为接近终端设备的边缘/雾计算节点（GW）[4]可以帮助增强、扩展和补充SDCM，具有低延迟、位置感知、移动支持和实时分析等优势[31]。因此，设备-边缘-云协作处理方法对于为终端设备或用户任务提供多样化的服务是必要的。

3.3. 为SDCM提供灵活的资源调度

在该架构的支持下，制造、计算和网络资源可以虚拟化并动态配置为端到端逻辑单元，以满足工业需求。该框架为打破资源之间的紧密耦合和协作奠定了基础，以确保SDVE和虚拟网络可以灵活地组织、配置和组合，形成高效的制造系统。这种设置有助于从静态预配置的制造系统中释放资源，并利用具有更大调度空间资源的潜力。制造

设备和物品的实时状态被监测并记录为工业大数据，通过设备-边缘-云计算进行处理和分析。结果可用于做出有关协作逻辑单元动态配置的明智决策。对于定制和个性化的生产订单，SDVE可以快速编程并重新用于特殊功能的实现。由SDN赋能的虚拟网络也可以根据协同SDVE的通信需求进行编程和配置。因此，可以实现高度灵活的资源调度。

总体而言，SDCM可以满足未来制造业在速度、规模、灵活性和开放性方面的需求。本文将SDCM应用于解决制造系统中的网络拥塞问题。

4. 问题称述

4.1. 问题描述

随着越来越多的制造物和机器连接成为协作网络，任何两个对象之间的数据交互都需要灵活和细粒度的网络资源调度，以实现高效的协作效率和低通信延迟。因此，采用SDCM来分离制造系统网络中的数据平面和控制平面。对于这种灵活的网络环境，有利于在控制平面上开发更有效的网络资源分配方法，以确保从全局角度确保低延迟数据传输。

4.2. 问题形式化

一个复杂的制造任务可以被称为CMT。一个CMT由多个制造过程组成，每个制造过程都可以在特定类型的制造机器上完成。每台原子制造机器都可以被抽象化、虚拟化（根据其操作逻辑和功能），并联网成为一个SDVE [如一个制造单元（MU）]。为了确保在执行任务时的有效协作，数据和信息必须通过SDN在制造资源之间传输（如MU和边缘计算节点/GW）。

4.2.1. 用于不同类型任务的MU

在一个SDCM系统中，不同的MU可以同时执行不同类型的制造工序。因此，当给定一项由多道工序组成的制造任务，记为$\mathrm{CMT}=\{a_0,\ a_1,\ldots,\ a_m\}$（$m$表示工序的数量）时，任意一道工序$a_i(i=1,\ 2,\ \ldots,\ m)$都对应于一个特定的类别$p_i\in P=\{\mathrm{type}_1,\ \mathrm{type}_2,\ldots,\ \mathrm{type}_L\}$（其中$L$表示类别的数量），而对于每一类的制造工序都有一个或多个工作单元能够完成。

记$\mathrm{SDVESet}=\{c_1,\ c_2,\ \ldots,\ c_L\}$表示执行机构（制造机器）集合，其中每个元素$c_j(j=1,\ 2,\ \ldots,\ L)$对应着能够执行$\mathrm{type}_j$类型工序的$\mathrm{MU}c_j=\{h_1^j,\ h_2^j,\ldots,\ h_{N_j}^j\}$。

因此，在选择制造任务中每道工序$a_i(i=1,\ 2,\ \ldots,\ m)$的执行单元时，需根据$p_i$（假设$p_i=\mathrm{type}_{j*}$）在相应的可选工作单元集$c_{j*}$中寻找。

4.2.2. 网络通信模式

对于一个复杂的制造任务$\mathrm{CMT}=\{a_0,\ a_1,\ldots,\ a_m\}$并指定每道工序的MU（为$a_i$指定的MU记为$h_i$），则相应的实现CMT的过程表示成一组二元数对：$\{(h_0\,h_1),\ (h_1\,h_2),\ldots,(h_{m-1}\,h_m)\}$，包括$m+1$个工序。这些工序分别在$m+1$个MU上完成。相邻两道工序间需要传递一些必要的数据和信息，因此整个过程需经过m次的信息传递。

在一个SDCM系统中，连接MU和GW的信息网络的相应的拓扑图可以表示为$\mathrm{GRAPH}\equiv(V,E)$，其中$V$为网络节点集合($V=S\cup H$)，$E$为网络节点关联关系（边）集合。如图3所示，A、B、C、D、I和J为MU，E、F、G和H为GW。假设两个互联节点之间存在用于数据传输的无线连接或有线电缆，则可以虚拟化由此产生的无线、有线或混合网络（如SDN），以提供网络切片来管理网络的各种需求集。换句话，网络资源被时隙化为网络切片，用于细粒度的网络资源调度，类似于“时分复用”的思想。此外，数据路由路径可以由SDCM网络选择和控制，以减少通信延迟。

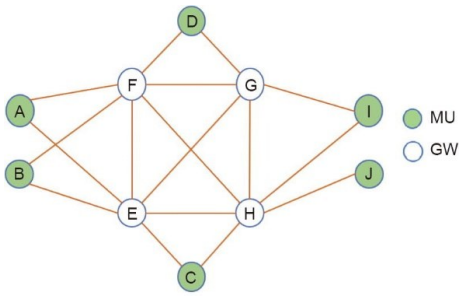

图3. 制造系统拓扑结构图。

在通道中传输之前，数据被分解为类似的结构，称为数据包。通信信道k（图GRAPH中的边k）中的数据包传输时间表示为τ_k。更宽的信道带宽对应于更高的数据传输速率和更小的τ_k。因此，信道中有l个数据包的$\mathrm{dt}(l)$总传输时间为$\Delta t=l\cdot\tau_k$。

给定制造任务集$\mathrm{CMTSet}=\{\mathrm{CMT}_1,\ \mathrm{CMT}_2,\ldots,\ \mathrm{CMT}_n\}$，记第$i$个CMT的第$j$道工序的开始时间是$t_{i,j}^s$（即工业数据信息传递至执行第$j$道工序MU的时刻），MU执行第$j$道工序的过程为$h_{i,j}$，完成该过程的时间为$t_{i,j}^e$。第$j$道工序完成所产生的有用的数据将传递至下一道（第$j+1$道）工序的MU $h_{i,j+1}$。

如果数据流（包含$l_{i,j}$个数据包）从MU $h_{i,j}$传递至

$h_{i,j+1}$的选定路径为：$\tilde{v}_0\left(h_{i,j}\right)\rightarrow\tilde{v}_1\rightarrow\ldots\rightarrow\tilde{v}_{r-1}\rightarrow\tilde{v}_r\left(h_{i,j+1}\right)$，途中经过$r-1(r\geq1)$个中间节点。该选定路径上的每个节点上数据流的到达时间及发出时刻分别为$\tilde{t}_p^{\text{in}}$和$\tilde{t}_p^{\text{out}}\left(p=0,\ 1,\ 2,\ldots,\ r\right)$。则有：

$$t_{i,j}^{\text{e}}=t_{i,j}^{\text{s}}+\Delta T_{i,j} \tag{1}$$

$$\tilde{t}_0^{\text{in}}=t_{i,j}^{\text{e}} \tag{2}$$

$$t_{i,j+1}^{\text{s}}=\tilde{t}_r^{\text{in}} \tag{3}$$

$$\tilde{t}_p^{\text{out}}=\tilde{t}_p^{\text{in}}+\Delta\tilde{t}_p^{\text{del}} \tag{4}$$

$$\tilde{t}_{p+1}^{\text{in}}=\tilde{t}_p^{\text{out}}+\Delta\tilde{t}_p^{\text{pro}}=\tilde{t}_p^{\text{out}}+l_{i,j}\Delta\tau_{k_p}\ \left(p=0,\ 1,\ \ldots,\ r-1\right) \tag{5}$$

式中，$\Delta T_{i,j}$为第i个CMT的第j道工序的执行时间；$\Delta\tilde{t}_p^{\text{del}}$表示该数据流$\text{dt}(l_{i,j})$（将要传递的）在节点$\tilde{v}_p$上排队等候的时间。特别地，当数据$\text{dt}(l_{i,j})$到达节点$\tilde{v}_p$，它们需等待该信道排在它前面的数据流传输完毕才能开始传输该数据；$\Delta\tilde{t}_p^{\text{pro}}$为数据流$\text{dt}(l_{i,j})$的总的传输时间；$\tau_{k_p}$为单个数据包从节点$\tilde{v}_p$至节点$\tilde{v}_{p+1}$的传输时间。因此，

$$\Delta\tilde{t}_p^{\text{del}}=\sum\nolimits_{q=1}^{n_{\text{seq}_p}}\tilde{l}_q^{\text{res}}\tau_{k_p}+\tilde{\Delta}_p \tag{6}$$

式中，n_{seq_p}表示节点$\tilde{v}_p$上前方排队数据流数量$\text{dt}(l_{i,j})$；$\tilde{l}_q^{\text{res}}$表示第q个排队数据流的剩余数据包数；$\tilde{\Delta}_p$表示信息流到达节点$\tilde{v}_p$时，当前传输数据包$\text{dt}(l_{i,j})$传输完毕的剩余时间，$0\leq\tilde{\Delta}_p<\tau_{k_p}$。

通过公式（1）~（6）以及$t_{i,j}^s$的值，并给出数据流$\text{dt}(l_{i,j})$传递的路径，便可计算出$t_{i,j+1}^s$的值。

$$\begin{cases}t_{i,j}^{\text{e}}=t_{i,j}^{\text{s}}+\Delta T_{i,j}\\ t_{i,j+1}^{\text{s}}=t_{i,j}^{\text{e}}+\sum\nolimits_{p=0}^{r-1}\left(l_{i,j}\cdot\tau_{k_p}+\sum\nolimits_{q=1}^{\text{n}_{\text{seq}_p}}\tilde{l}_q^{\text{res}}\tau_{k_p}+\tilde{\Delta}_p\right)\end{cases} \tag{7}$$

进一步，若假定$t_{i,0}^s=0$（i=1, 2, …, n），则可以计算每个CMT中每道工序的执行时刻$t_{i,j}^s$和$t_{i,j}^e$（$i=1,\ 2,\ \ldots,\ n;j=1,\ 2,\ \ldots,\ m_i$）。

4.2.3. 时间和性能约束

（1）**数据传输的时间约束**。对于第i个CMT的第j次处理过程，给定了数据流量$l_{i,j}$（数据流包含的数据包个数），以及时间敏感性数据传输的时间上限值$\tau_{i,j}$。因此对于选择的信息传输路径，需满足信息传递的时间上限约束条件，即：

$$t_{i,j}^s-t_{i,j-1}^e=\left.\sum\nolimits_{p=0}^{r-1}\left(l_{i,j-1}\cdot\tau_{k_p}+\sum\nolimits_{q=1}^{n_{\text{seq}_p}}\tilde{l}_q^{\text{res}}\tau_{k_p}+\tilde{\Delta}_p\right)\right|_{i,j-1}\leq \tau_{i,j}\ \left(i=1,\ 2,\ \ldots,\ n;\ j=1,\ 2,\ \ldots,\ m_i\right) \tag{8}$$

公式（8）中$(.)_{i,j-1}$表示第i个CMT中数据流由MU h_{j-1}^i传递至h_j^i的过程。

（2）**MU同时可执行任务上限约束**。每个MU在同一时间所能执行的任务是有限的。记整个SDCM系统中的MU集为$H=\left\{h^1,\ h^2,\ldots,\ h^N\right\}$，共$N$个MU。任一MU i可以同时执行的任务量上限值用$C^i\left(i=1,\ 2,\ \ldots,\ N\right)$表示。

此外，$w_{i,j}$表示第i个CMT的第j道工序的任务量，并用$\beta_{i,j}^k$表示第i个CMT的第j道工序占用第k个MU的情况，$\beta_{i,j}^k$为0~1变量。

$$\beta_{i,j}^k=\begin{cases}1\ \left(j\text{th process of }i\text{th CMT occupies the }k\text{th unit}\right)\\ 0\ \left(j\text{th process of }i\text{th CMT does not occupy the }k\text{th unit}\right)\end{cases} \tag{9}$$

那么对于所有给定任务的每道工序，都需要寻找与它有相同MU且有重叠执行时间的其他工序集合，记为$\text{Neighbor}_{i,j}\left(i=1,\ 2,\ \ldots,\ n;\ j=1,\ 2,\ \ldots,\ m_i\right)$。确定$\text{Neighbor}_{i,j}$的方法如下。

首先选定第i个CMT的第j道工序，按照前述方法可确定该工序被执行的时间区间为$\left[t_{i^*,j^*}^s,\ t_{i^*,j^*}^e\right]$。然后遍历CMTSet所有制造任务中的所有工序，若工序（第i个CMT的第j个工序）满足公式（10）所示的关系，则将它加入$\text{Neighbor}_{i^*,j^*}$（按照当前规则，第$i^*$个CMT的第$j^*$道工序自身也将包含在集合$\text{Neighbor}_{i^*,j^*}$中）。

$$\begin{cases}\beta_{i,j}^k=\beta_{i^*,j^*}^k=1\\ t_{i^*,j^*}^{\text{s}}\leq t_{i,j}^{\text{s}}<t_{i^*,j^*}^{\text{e}}\ \text{ or }\ t_{i^*,j^*}^{\text{s}}<t_{i,j}^{\text{e}}\leq t_{i^*,j^*}^{\text{e}}\end{cases} \tag{10}$$

被选中的MU需满足同时可执行任务上限的约束：

$$\sum\nolimits_{a_{i,j}\in\text{Neighbor}_{i^*,j^*}}w_{i,j}\leq\sum\nolimits_{k=1}^{N}\beta_{i^*,j^*}^k\cdot C^k\ \left(i^*=1,\ 2,\ \ldots,n;\ j^*=1,\ 2,\ \ldots,\ m_i\right) \tag{11}$$

4.2.4. 模型最优化

本模型的最终目标是合理安排给定制造任务每道工序的MU，以及同一制造任务中相邻两道工序MU间的数据传输路径，使得所有给定任务在最短时间内完成。约束是数据传输的时间约束和每一个MU可同时执行的任务上限。因此，可以建立最优模型如等式（12）所示：

$$\min\ \left(\max_{i=1,\ 2,\ \ldots,\ n}\left\{t_{i,m_i}^{\text{e}}\right\}\right)$$

Subject to

$$\begin{cases}\left.t_{i,j}^s-t_{i,j-1}^{\text{e}}=\sum\nolimits_{p=0}^{r-1}\left(l_{i,j-1}\cdot\tau_{k_p}+\sum\nolimits_{q=1}^{n_{\text{seq}_p}}\tilde{l}_q^{\text{res}}\tau_{k_p}+\tilde{\Delta}_p\right)\right|_{i,j-1}\leq\tau_{ij},\\ \left(i=1,\ 2,\ \ldots,\ n;\ j=1,\ 2,\ \ldots,\ m_i\right)\\ \sum_{a_{i,j}\in\text{Neighbor}_{i^*,j^*}}w_{i,j}\leq\sum_{k=1}^{N}\beta_{i^*,j^*}^k\cdot C^k\\ \left(i^*=1,\ 2,\ \ldots,n;\ j^*=1,\ 2,\ \ldots,\ m_i\right)\end{cases} \tag{12}$$

5. 问题求解算法

作为一种随机搜索算法，GA有两个显著优势：解决复杂问题的能力和并行性[32]。为了解决所考虑的优化模型，使用GA来优化每个制造任务的不同工序MU的选择。随后，Dijkstra最短路径算法用于寻找制造任务中两个相邻处理过程（MU）之间数据传输的最短路径。如果其他制造任务需要使用当前被任务占用的信道，可以采用考虑数据传输剩余时间的改进排队算法依次传输数据。这里，信道占用是指源节点和目的节点之间的信道（网络）当前被用于制造任务的数据传输的状态。如果其他任务需要在该信道传输数据，则需要将要发送的数据包在源节点进行排队。两个或多个任务可以共享同一个信道，每个任务在不同的时隙独占使用该信道。算法1和算法2分别显示了GA的结构和计算给定基因型（所有制造任务的各工序的MU）个体的制造任务完成时间T_e的算法框架。由于该模型主要关注点在于数据传输过程，因此在算法1中判断约束条件是否满足时，忽略了MU的最大工作能力约束，而仅考虑了传输时间条件约束。

Algorithm 1. GA for minimizing T_e.

generate and initialize Gr // Gr means population and N_e contains elements denoted by Ele
$k \leftarrow 0$ // Gr_max is the maximum number of genetic iterations
while $k <$ Gr_max do
operate cross algorithm on Ele, and obtain new element Ele_{new}
operate variation algorithm on Ele_{new}, and obtain the variant Ele_{new}
calculate T_e of variant Ele_{new}
calculate the fitness of Ele and variant Ele_{new}
choose the top N_e elements in terms of fitness as next generation among Ele and variant Ele_{new}
$k \leftarrow k + 1$
output the final Gr

（1）**GA**。对于任务集中每个CMT的每道工序，当选定其MU后，可根据下文中的方法分配信息传递路径和排队顺序，从而验证约束条件的满足情况，并计算得到所有任务完成的时长。

在本研究中，拟采用GA算法，得到最优MU制定方案。算法中，将所有CMT的每道工序都设置为一个基因座，即基因座总数为$\sum_{\text{CMT}_i \in \text{CMTSet}} \text{CMT}_i$的子任务数量。每个基因座的取值情况根据工序的类型，在对应的可选MU集中选择其对应序号（集合中MU按照制定顺序依次编号）。

为了尽可能多地得到优化问题的最优值，算法中在计算个体适应度时，根据个体基因型与种群中其他个体的差异来适当调整个体适应度。个体适应度计算公式如式（13）所示，式中d表示个体基因型与种群其他个体的平均差异，而D表示种群内所有个体的平均差异，它们的计算公式如式（14）和式（15）所示。T_e为个体基因型的“表现性状”（即设定的MU选择方案所需的制造任务执行时间，若选择的方案不能满足约束条件，则在原来的执行时间基础上增加一个较大值M）。T_{e_min}为种群中最优个体对应的执行时间。

$$\text{fitness} = \left[1 + \lg\left(d/D\right)\right] / \max\left\{\Delta,\ T_e - T_{e\_min}\right\} \tag{13}$$

$$\begin{cases} d^{i*} = \dfrac{1}{N_e}\sum_{i=1}^{N_e}\sum_{j=1}^{\text{genetic\_sum}} f\left(g_j^i - g_j^{i*}\right) \\ f(x) = \begin{cases} 0 & (x=0) \\ 1 & (\text{otherwise}) \end{cases} \end{cases} \tag{14}$$

$$D = \frac{1}{N_e}\sum_{i=1}^{N_e} d^i \tag{15}$$

在式（13）和式（15）中，Δ是很小的非零正数，N_e为种群规模（种群中的个体总数），genetic_sum表示个体基因座总数，g_j^i表示种群中第i个个体的第j个基因座的取值。

（2）**路径算法**。当算法2为所有CMT的每道工序选择数据传输路径时，可以应用Dijkstra路径算法来寻找两个给定结点（MU）之间的最短路径。

相连两节点之间的边（网络拓扑图GRAPH中的第k条边）的距离为单个数据包传输所需的时间τ_k，而不相连的两节点之间的距离设置为一个较大值$LD\left(LD \gg \max_k\ \{\tau_k\}\right)$。

Algorithm 2. Calculate T_e for given element.

obtain actuators for all CMTs’ subtasks by element’ genes
solve path planning problem by Dijkstra’s algorithm
calculate time schedule for all CMTs’ subtasks and obtain original T_e
for all CMTs’ transmission process do
if transmission time > constraint time
$T_e \leftarrow T_e + M$ // M is a huge number that is much bigger than original T_e
return T_e

（3）**排队算法**。当某条信道完成当前数据流的传输后，若有若干正在等待传输的数据流，则按照其剩余的可用延迟时间$T_{i,j}^{\text{res}}$的长短来安排排队顺序，$T_{i,j}^{\text{res}}$最小的数据流最先开始传输。$T_{i,j}^{\text{res}}$的计算方法为：

$$T_{i,j}^{\text{res}} = \tau_{i,j} - T_{i,j}^{\text{P}} - \sum_{\text{Post}} T_{i,j}^{\text{del}} \tag{16}$$

式中，$T_{i,j}^{\mathrm{P}}$为第i个CMT的第j次信息传递的传播时长，$\sum_{\mathrm{Post}} T_{i,j}^{\mathrm{del}}$为其之前的传输过程已延迟时长。

6. 实验和分析

本模型的数值实验主要考虑了模型算法在两种信息网络系统中的效果：①经典且简单的信息网络（网络中共有10个节点，其中包括6个执行单元节点和4个中间节点），如图3所示。②随机生成网络系统，记为Sys(num, prob)，其中num表示网络拓扑结构的节点数，prob表示两节点之间相连的概率。仿真实验采用Visual Studio 2019 C++进行，num = 20（从20个节点中随机选择12个执行单元节点），prob = 0.25。

简单网络系统中给定的制造任务集CMTSet与复杂网络系统中的不同。在简单网络和复杂网络中，CMTset分别包含5个和10个CMT。在每个CMTSet中，设定三类工序执行时间，执行时间设定为0（代表远小于数据传输时间的值）、约等于数据传输时间的值，以及超过数据传输时间的值（比数据传输时间大三个数量级）。

设定信息网络信道中单个数据包传输所需时间均为0.1 μs，而一个数据流传输完成时间在0.1 ~ 1.0 ms之间。因此，两个MU之间数据传输的时间限制也设定在1 ms的时间量级。相关参数及其设置列于附录A的表S1~S6中。

6.1. 简单网络结果分析

在简单网络中，按照工序执行时长的不同，分别进行了三组实验，其工序执行时长分别为：无执行时间（NET）——工序执行时间为0；较短执行时间（SET）——工序执行时间与数据传输时间同量级；较长执行时间（LET）——工序执行时间比数据传输时间大三个量级。

图4显示了三组实验中随机生成的MU选择方案是否满足时间约束条件。当工序执行时长小于或与数据传输时间相差不大时，随机安排的执行方案能够满足时间约束条件的平均概率极低，仅有1%左右，这主要是因为给定的时间约束条件较紧。然而当工序执行时长远大于数据传输时间时，由于此时数据传输较为分散，信道占用的情况较少，数据传输过程等待时长较短，因而随机方案中满足时间约束条件的平均概率大大提高，约为14%。

图5为三组实验中随机生成的“最优”情况与算法收敛解的比较情况。纵坐标为通信总时长（即为制造任务完成时间减去所有CMT中工序执行总时长的最大值，它包含了数据传输时间以及排队等待时间）。实验表明，无论执行时间如何，GA获得的收敛解决方案中的最短通信总时长都远小于随机生成的方案中的最小通信总时长。在简单网络中，算法解总是收敛到一个优解或最优解。

6.2. 复杂网络结果分析

与简单网络中的任务数相比，复杂网络增加了CMTSet中制造任务的总任务数（复杂网络中的节点数是简单网络的两倍，因此，复杂网络中的CMT也增加了一倍）。此外，适当放宽时间限制，根据制造工序时长进行了三组数值实验。

图6为随机方案满足时间约束条件的情况，图7展示了使用随机方法获得“最优”解与算法收敛解的比较情况。

图6表明，当时间约束适当放宽之后，随机方案满足时间约束的概率显著增加，并且随着工序执行时长的增大，随机方案中满足时间约束条件的平均概率逐渐增加。当工序执行时长远大于网络中数据传输时间时，满足率接近100%。

从图7可以看出，尽管随机方案中的时间约束条件满

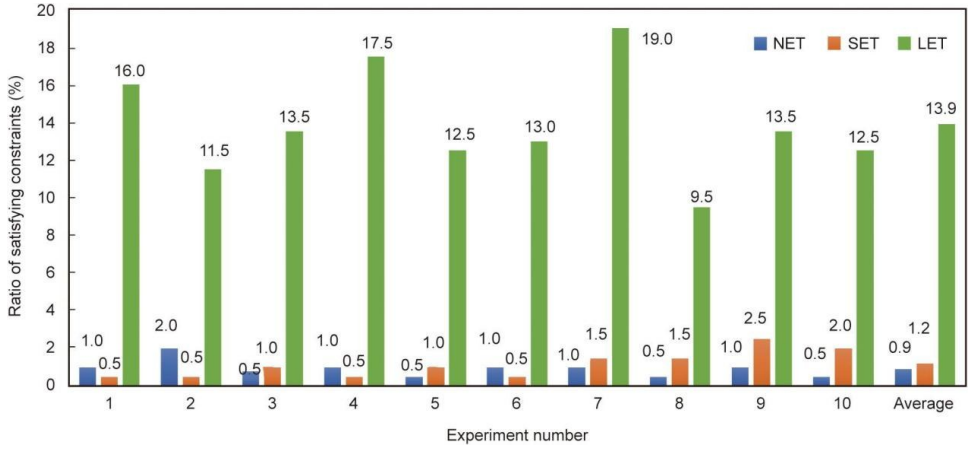

图4. 简单网络中随机方案约束条件满足率。

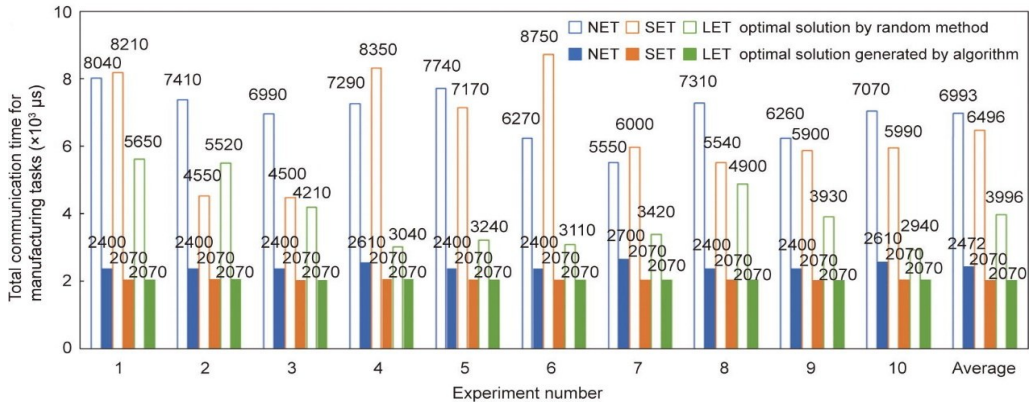

图5. 简单网络中随机方案与算法收敛解方案最短通信总时长对比。

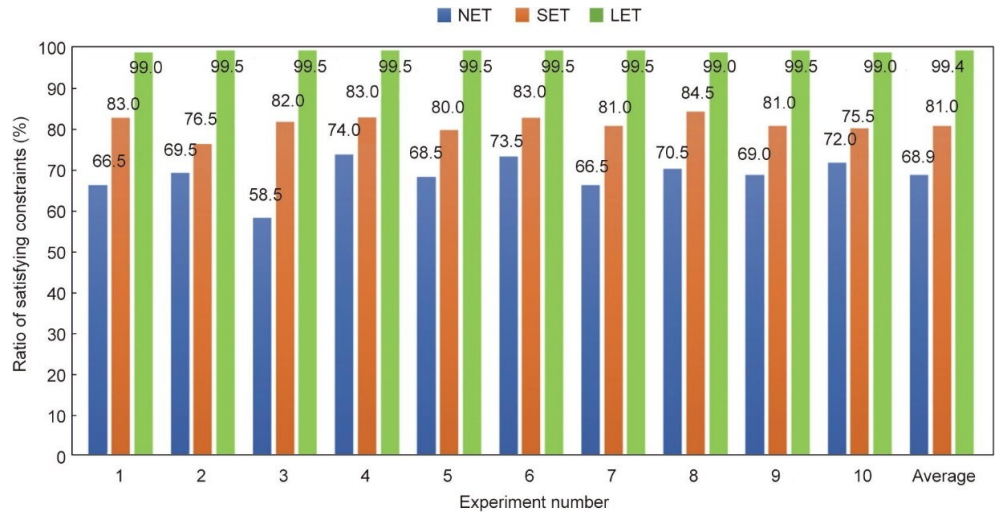

图6. 复杂网络中随机方案约束条件满足率。

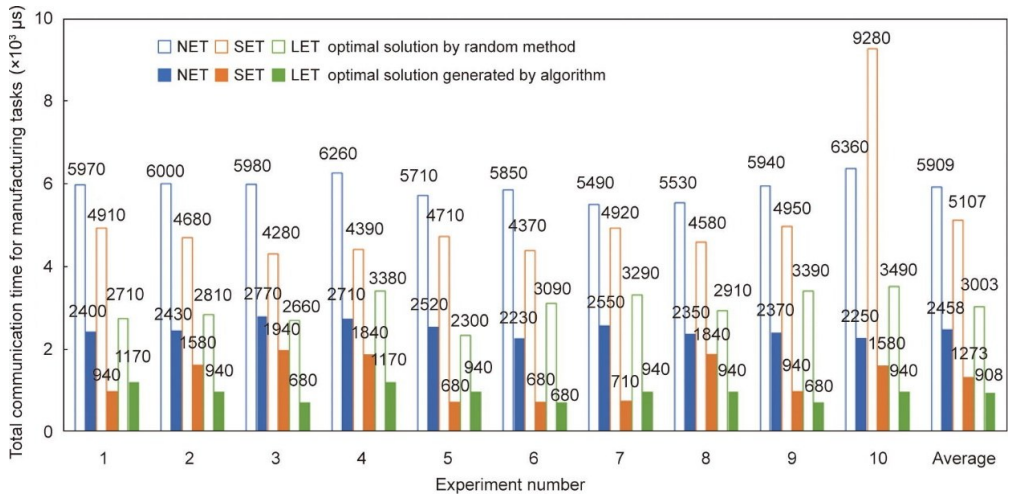

图7. 复杂网络中随机方案与算法收敛解方案最短通信总时长对比。

足率很高，但是算法收敛解中的最短通信总时长仍然远小于随机方案中“最优”方案的通信总时长。然而，与简单网络相比，在复杂网络中，所提出的算法不易收敛到理论最优解。随着工序执行时长的延长，算法收敛解的最短通信时间有逐渐趋于稳定在最小值的趋势。

6.3. 算法收敛速率分析

评估了在不同复杂程度的信息网络以及不同工序执行

时长的场景中，模型算法的收敛速度。

在GA算法中，每一代更新的收敛速率为：

$$v_{k+1}=-\lg\frac{T_{\mathrm{e\_min}\,k+1}}{T_{\mathrm{e\_min}\,k}} \tag{17}$$

式中，$T_{\mathrm{e_min}\,k}$表示第k代中种群个体中制造任务完成的最短时间。根据式（17），在6组实验数据中随机各选两组收敛速率的实验数据，得到如图8所示的收敛速率曲线。

图8表明，随着工序执行时长的增加，有效更新代数

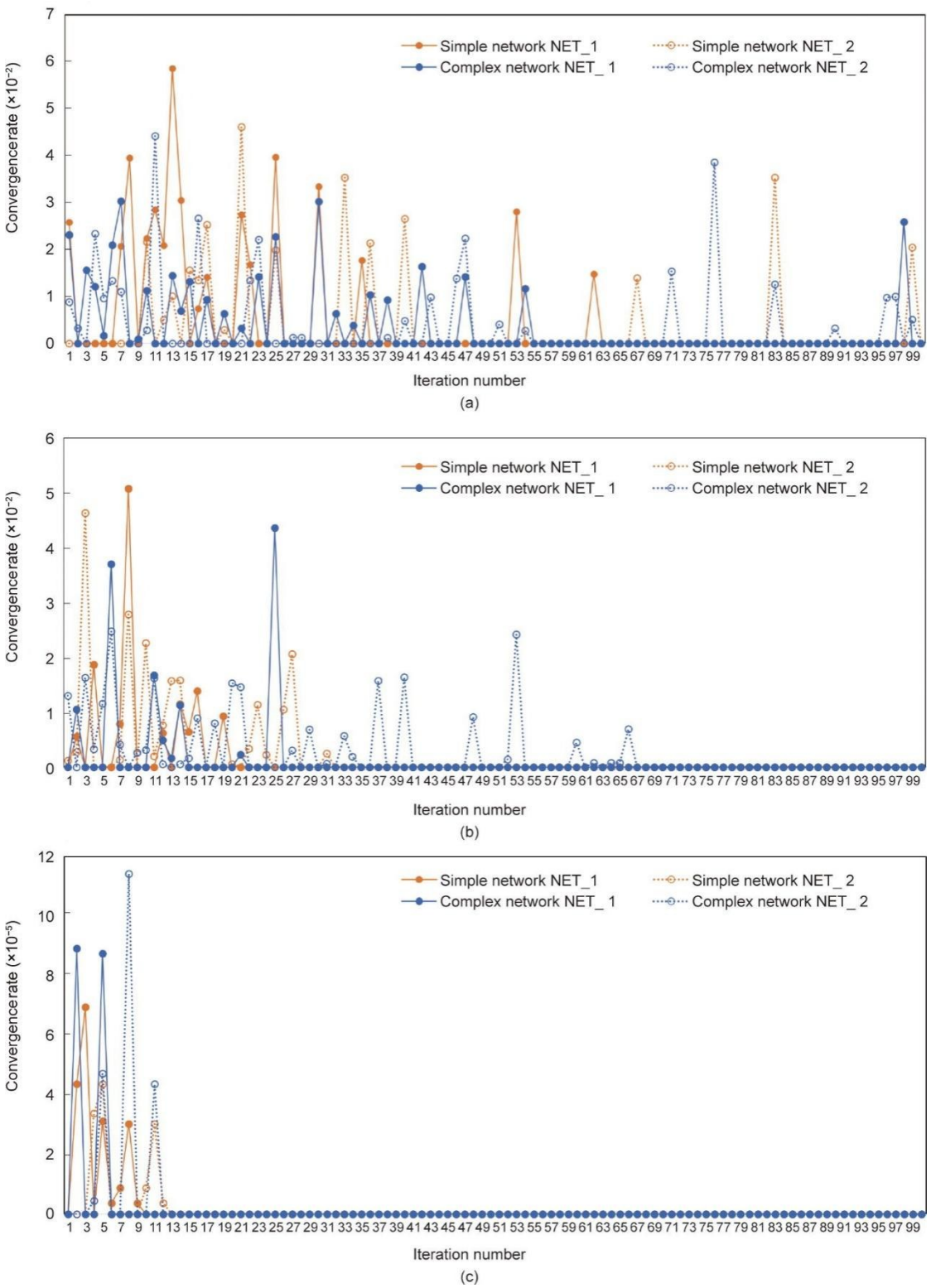

图8. 不同工序执行时长下的收敛速率曲线。（a）NET场景；（b）SET场景；（c）LET场景。

范围（此范围之后代数收敛率为0）在逐渐减小。NET、SET和LET场景下有效更新代数分别为100、70和20。因此，随着工序执行时长增加，模型算法收敛所需的代数逐渐减小，所要求的相应计算量也逐渐降低。

7. 结论

本文提出了一种新的基于SDN的CMfg模型，命名为SDCM。SDCM采用SDN和边缘计算扩展CMfg，使资源可编程。此外，该框架使得制造系统可以快速重构、运行和演进，以确保系统能够及时响应外部和内部的变化。为了减少SDCM中产生的大量数据带来的网络拥塞和数据传输延迟，本文建立了一个考虑子任务分配和数据传输路径选择的时间敏感数据流调度模型。随后，应用GA、Dijkstra算法和排队算法来求解优化问题，为制造任务分配满足时间约束的MU。实验结果表明，该模型和算法均能很好地满足约束条件，并降低总通信时间。

未来的工作可以集中在两个方面。首先，旨在针对优化问题改进所提出的路径规划算法，与其他算法进行比较，并在真实工业环境下进行实验。其次，必须考虑安全、数据隐私、任务优先级和利润分配等因素，确保企业或所有者愿意提供制造资源的控制逻辑。对于集团公司或企业的所有者，在管理层没有反对的情况下，可以建立统一安全并合理地整合和运营制造资源的技术或方法。由于涉及多个利益相关者的案例要复杂得多，因此，本研究团队将继续探索与安全、隐私和业务方面相关的问题。

致谢

该研究得到国家重点研发计划（2021YFB1715700）、国家自然科学基金（62103046）、北京理工大学青年教师学术启动计划项目、中国科学院和中国科学院大学研究项目（Y92902MED2、E1E90808 和 E0E90804）和中央高校基本科研业务费专项资金（E1E40805）的支持。

Compliance with ethics guidelines

Chen Yang, Fangyin Liao, Shulin Lan, Lihui Wang, Weiming Shen, and George Q. Huang declare that they have no conflicts of interest or financial conflicts to disclose.

Appendix A. Supplementary data

Supplementary data to this article can be found online at https://doi.org/10.1016/j.eng.2021.08.022.

References

[1] Yang C, Shen W, Wang X. The Internet of Things in manufacturing: key issues and potential applications. IEEE Syst Man Cybern Mag 2018;4(1):6–15.
[2] Hao Y, Jiang Y, Chen T, Cao D, Chen M. iTaskOffloading: intelligent task offloading for a cloud–edge collaborative system. IEEE Netw 2019;33(5):82–8.
[3] Yang C, Lan S, Shen W, Wang L, Huang GQ. Software-defined cloud manufacturing with edge computing for Industry 4.0. Proceedings of 16th International Wireless Communications and Mobile Computing; 2020 Jun 15–19; Limassol, Cyprus. New York City: IEEE; 2020.
[4] Bonomi F, Milito R, Zhu J, Addepalli S. Fog computing and its role in the Internet of Things. In: Proceedings of The First Edition of the MCC Workshop on Mobile Cloud Computing; 2012 Aug 17; Helsinki, Finland. New York City: ACM; 2012. p. 13–6.
[5] Yang C, Lan S, Wang L, Shen W, Huang GQ. Big data driven edge–cloud collaboration architecture for cloud manufacturing: a software defined perspective. IEEE Access 2020;8:45938–50.
[6] Kagermann H, Wahlster W, Helbig J. Recommendations for implementing the strategic initiative Industrie 4.0—final report of the Industrie 4.0 working group. Report. Berlin: Forschungsunion; 2013 Apr.
[7] Yang C, Lan S, Shen W, Huang GQ, Wang X, Lin T. Towards product customization and personalization in IoT-enabled cloud manufacturing. Cluster Comput 2017;20(2):1717–30.
[8] Saxena LK, Jain PK. An integrated model of dynamic cellular manufacturing and supply chain system design. Int J Adv Manuf Technol 2012;62:385–404.
[9] Yang C, Shen W, Lin T, Wang X. IoT-enabled dynamic service selection across multiple manufacturing clouds. Manuf Lett 2016;7:22–5.
[10] Meng Z, Wu Z, Gray J. Architecting ubiquitous communication and collaborative-automation-based machine network systems for flexible manufacturing. IEEE Syst J 2020;14(1):113–23.
[11] McKeown N. Software-defined networking. Infocom Keynote Talk 2009;17(2):30–2.
[12] Li B, Zhang L, Wang S, Tao F, Cao J, Jiang X, et al. Cloud manufacturing: a new service-oriented networked manufacturing model. Comput Integr Manuf Syst 2010;16(1):1–7. Chinese.
[13] Ren L, Zhang L, Wang L, Tao F, Chai X. Cloud manufacturing: key characteristics and applications. Int J Comput Integ M 2017;30(6):501–15.
[14] Simeone A, Zeng Y, Caggiano A. Intelligent decision-making support system for manufacturing solution recommendation in a cloud framework. Int J Adv Manuf Technol 2021;112:1035–50.
[15] Mourtzis D, Vlachou E, Milas N, Tapoglou N, Mehnen J. A cloud-based, knowledge-enriched framework for increasing machining efficiency based on machine tool monitoring. Proc IMechePart Eng BJ Eng Manuf 2019;233(1):278–92.
[16] Liu Y, Zhang L, Wang L, Xiao Y, Xu X, Wang MA, et al. A framework for scheduling in cloud manufacturing with deep reinforcement learning. Proceedings of 2019 IEEE 17th International Conference on Industrial Informatics; 2019 Jul 22–25; Helsinki, Finland. New York City: IEEE; 2019.
[17] Queiroz J, Leitão P, Barbosa J, Oliveira E, Garcia G. An agent-based industrial cyber – physical system deployed in an automobile multi-stage production system. In: Borangiu T, Trentesaux D, Leitão P, Giret BA, editors. Service oriented, holonic and multi-agent manufacturing systems for industry of the future. Switzerland: Springer Cham; 2019. p. 379–91.
[18] Liao H, Zhou Z, Zhao X, Zhang L, Mumtaz S, Jolfaei A, et al. Learning-based context-aware resource allocation for edge-computing-empowered Industrial IoT. IEEE Internet Things J 2020;7(5):4260–77.
[19] He X, Tu Z, Xu X, Wang Z. Programming framework and infrastructure for self-adaptation and optimized evolution method for microservice systems in cloud-edge environments. Future Gener Comp Syst 2021;118:263–81.
[20] Ren L, Meng Z, Wang X, Zhang L, Yang LT. A data-driven approach of product quality prediction for complex production systems. IEEE Trans Ind Inform 2021;17(9):6457–65.
[21] Caggiano A. Cloud – based manufacturing process monitoring for smart

diagnosis services. Int J Comput Integ Manuf 2018;31(7):612–23.
[22] Ren L, Meng Z, Wang X, Lu R, Yang LT. A wide–deep-sequence model based quality prediction method in industrial process analysis. IEEE Trans Neur Net Lear 2020;31(9):3721–31.
[23] Ren L, Liu Y, Wang X, Lu J, Deen MJ. Cloud–edge based lightweight temporal convolutional networks for remaining useful life prediction in IIoT. IEEE Internet Things J 2021;8(16):12578–87.
[24] Ren L, Laili Y, Li X, Wang X. Coding-based large-scale task assignment for industrial edge intelligence. IEEE Trans Netw Sci Eng 2020;7(4):2286–97.
[25] Kreutz D, Ramos FMV, Esteves Verissimo P, Esteve Rothenberg C, Azodolmolky S, Uhlig S. Software-defined networking: a comprehensive survey. Proc IEEE 2015;103(1):14–76.
[26] Hu PA. A system architecture for software-defined industrial Internet of Things. Proceedings of 2015 IEEE International Conference on Ubiquitous Wireless Broadband; 2015 Oct 4–7; Montreal, QC, Canada. New York City: IEEE; 2015.
[27] Salahuddin MA, Al-Fuqaha A, Guizani M. Software-defined networking for RSU clouds in support of the Internet of Vehicles. IEEE Internet Things 2015; 2(2):133–44.
[28] Naeem F, Srivastava G, Tariq M. A software defined network based fuzzy normalized neural adaptive multipath congestion control for Internet of Things. IEEE Trans Netw Sci Eng 2020;7(4):2155–64.
[29] Brody P, Pureswaran V. The new software-defined supply chain: preparing for the disruptive transformation of electronics design and manufacturing. Report. IBM Institute Business Value; 2013. Chinese.
[30] Islam K, Shen W, Wang X. Wireless sensor network reliability and security in factory automation: a survey. IEEE Trans Syst Man Cybern Part C Appl Rev 2012;42(6):1243–56.
[31] Gu B, Zhou Z, Mumtaz S, Frascolla V, Bashir AK. Context-aware task offloading for multi-access edge computing: matching with externalities. Proceedings of 2018 IEEE Global Communications Conference; 2018 Dec 9–13; Abu Dhabi, United Arab Emirates. New York City: IEEE; 2018.
[32] Yang XS. Nature-inspired optimization algorithms. Elsevier; 2014.