研究与开发

基于算力网络的异构算力请求路由算法

章刚[1,2]，黎曦[1,2]

（1. 南昌工学院信息与人工智能学院，江西 南昌 330108；
2. 南昌工学院南昌市物联网信息可视化技术重点实验室，江西 南昌 330108）

摘　要：由于算力请求具有特殊性和独特性，如何为一组异构算力请求寻找传输链路互不相交的有效路径集，使得该组请求能够到达各自目的算力节点，从而实现为该组请求分配算力资源，是当前算力网络面临的关键问题。首先，对异构算力请求的路由问题进行剖析，并通过建模将其转化为非确定性多项式（nondeterministic polynomial，NP）完全问题。针对该问题，提出一种优化型遗传算法。该算法从局部和全局两个层面进行设计：在局部层面，为保证快速收敛到目标解，采用单参数满足随机性策略初始化种群，使得种群广泛地分散在解空间中；采用多参数解（或路径）均衡选择策略进行选择操作，使得被选种群丰富多样；采用两层交叉策略进行交叉操作，目的是拓宽全域搜索广度；采用多参数随机单点变异策略进行变异操作，目的是深挖局域搜索能力。在全局层面，为保证路径不冲突，采用路径分发策略，通过构建求解矩阵，并借助评价函数、可行解随机选择、低需求优先避让原则等方法，确保最终找到一组可行解集。实验从异构算力请求的传输成功率、算法收敛延时比、算力网络负载均衡误差率等3个方面进行验证，相较于IGAGCT算法与RBDQN算法，该算法在传输成功率、算法收敛延时比和负载均衡方面分别平均优化了8.85%、15.51%、17.03%及10.41%、16.5%、16.81%。

关键词：算力网络；异构算力请求；遗传算法；算力路由；NP完全问题

中图分类号：TP393

文献标志码：A

doi: 10.11959/j.issn.1000−0801.2025016

Routing algorithm for heterogeneous computing force requests based on computing first network

ZHANG Gang[1,2], LI Xi[1,2]

1. School of Information and Artificial Intelligence, Nanchang Institute of Science and Technology, Nanchang 330108, China
2. Nanchang Key Laboratory of Internet of Things Information Visualization Technology, Nanchang Institute of Science and Technology, Nanchang 330108, China

Abstract: Due to the particularity and uniqueness of computing force requests, how to find an effective path set with

收稿日期：2024−10−12；**修回日期：**2024−11−28

通信作者：章刚，zhanggang@usst.edu.cn

基金项目：国家自然科学基金资助项目（No.61572325）；江西省教育厅科技研究项目（No.GJJ212510）；南昌工学院人才引进项目（No.NGRCZX-21-07）

Foundation Items: The National Natural Science Foundation of China (No.61572325), The Education Department Foundation of Jiangxi Province (No.GJJ212510), The Talent Introduction Project of Nanchang Institute of Technology (No.NGRCZX-21-07)

non-intersecting transmission links for a group of heterogeneous computing force requests, so that the group of requests can reach their respective destination computing force nodes, and thus allocate computing force resources for the group of requests, is a key issue facing current computing first networks. Firstly, the routing problem of heterogeneous computing force requests was analyzed and was transformed into an nondeterministic polynomial (NP)-complete problem through modeling. An optimized genetic algorithm to address this issue was proposed. This algorithm was designed from both local and global perspectives: to ensure fast convergence to the target solution locally, a single parameter satisfying the randomness strategy was used to initialize the population, making it widely dispersed in the solution space; adopting a multi-parameter solution (or path) balanced selection strategy for selection operations, making the selected population rich and diverse; adopting a two-layer crossover strategy for crossover operations, with the aim of expanding the breadth of global search; adopting a multi parameter random single point mutation strategy for mutation operations, with the aim of deepening local search capabilities. To ensure that the paths did not conflict globally, a path distribution strategy was adopted. By constructing a solution matrix and utilizing evaluation functions, random selection of feasible solutions, and the principle of low demand priority avoidance, a set of feasible solution sets was ultimately found. The experiment verifies that algorithm has been optimized by an average of 8.85%, 15.51%, and 17.03% in terms of transmission success rate, convergence delay ratio, and load balancing compared to the IGAGCT algorithm and RBDQN algorithm, and 10.41%, 16.5%, and 16.81%, respectively from three aspects: heterogeneous request success rate, algorithms convergence delay rate, and load error rate of computing first networks.

Key words: computing first network, heterogeneous computing force request, genetic algorithm, computing force routing, NP-complete problem

0 引言

异构算力请求的路由（heterogeneous computing force request for routing，HCFRR）是目前算力网络（computing first network，CFN）必须讨论的众多重要问题之一[1-4]，是CFN利用网络拓展了单一算力的边界，并整合了中央处理器（central processing unit，CPU）、图形处理器（graphics processing unit，GPU）[5]及神经处理单元（neural processing unit，NPU）[6]等现有算力资源之后，尝试为全场景下不同类型的请求分配算力资源时所形成的产物。

具体来讲，在分配算力资源的过程中，面对复杂性请求时，CFN会将所需的算力资源迁移至请求附近进行分配处理，这是由复杂性请求具有计算规模巨大、难以通过网络传输的特点所决定的。当面对无法在本地处理的简单性请求时，由于这些请求具有异构性、群组性、随机性以及计算量小等特点，如果依然采用迁移算力资源的方式进行分配，可能导致CFN内部做大量无效的计算，增加系统的负载；如果采用基于数学模型的算力资源分配策略，同样可能存在不适用性。原因在于，虽然这些分配策略都聚焦于资源的分配，但绝大多数都是在上层供给侧通过抽象的数学公式构建资源分配模型，而在底层忽视了关注算力资源到底应如何实现真正意义上的分配，即关注抽象数学模型多于关注具体实现方式，本文讨论的资源分配恰恰属于后者，这是由资源分配的实践性和应用性所决定的。HCFRR的存在为这一问题的解决指明了方向，借助HCFRR将简单性请求通过网络同时分发到不同的目的算力资源上进行处理（即传输请求至算力资源上分配处理），从而能够真正意义上实现在底层分配资源的诉求，这是由消耗网络带宽的成本远低于CFN系统负载的成本所决定的。由此可知，HCFRR是对迁移算力方式的一种补充和完善，同时也是对当前资源分配知识体系的丰富和扩展，多种技术

之间相辅相成，从而能够保证CFN应用于更加丰富的实践场景。因此，HCFRR不仅保障了有效的算力服务，而且成为构建算力网络的众多关键技术之一[1-6]。

依据以上描述，HCFRR是指给定一组异构算力请求及其所对应的目的算力节点，基于约束条件，HCFRR能够为该组请求提供到达相对应的目的算力节点的有效路径，以实现算力资源分配的目的。此处的算力请求包括CPU资源请求、GPU资源请求、NPU资源请求等。显然，相较于现场可编程门阵列（field programmable gate array，FPGA）[7]、专用集成电路（application specific integrated circuit，ASIC）[8]等集群内输入/输出（input/output，I/O）通信以及传统的Internet路由，HCFRR的出发点和技术路径完全不同：在出发点上，HCFRR是为了实现算力资源的分配而形成的一种技术手段；在技术路径上，HCFRR则是一种面向网络层面的、支持多对多连接的组路由形式。

需要注意的是，实现HCFRR的难点在于，CFN中的每个算力请求都具有独特性，并对网络上的各类度量参数具有一定要求。因此，在面对一组异构算力请求时，HCFRR在组路径选择的过程中，一定要避免组内不同的请求之间共享相同的传输链路，以此来保障每个算力请求的独特性。也就是说，如何在CFN上寻找一组相互独立的、没有传输链路交集的有效路径，是实现HCFRR的关键所在[2-4,9-10]。

现阶段，针对简单性算力请求的算力资源分配问题，可从两个层面进行考虑：一是在上层资源层面，通过构建模型进行资源分配，这一层面表现为采用数学方式构建供需双方的资源调度与匹配模型，虽然能够在一定程度上解决资源分配问题，但仅限于理论层面，尚不够全面；二是在底层实现层面，通过具体的方式和手段实现资源分配，这一层面表现为通过灵活的算力路由策略，在底层真正意义上实现资源分配的目的。后者是实现资源分配的关键，也是本文讨论的重点。

从资源分配模型的角度分析。文献[11]针对网络资源与计算资源如何权衡问题，基于图神经网络以及深度强化学习框架提出了一种资源分配模型，该模型能够高效地管理资源分配。文献[12]为了能够从服务模型、网络状态及资源联合等方面更好地分配算力资源，提出“云-边-端”资源协同优化分配模型，有效整合了各类异构算力资源，并提升了资源分配精度。文献[13]为解决算力请求导致的算力资源缺失问题，提出了一种基于容器集群部署的算力资源分配模型，旨在提高分配效率。文献[14]从系统观念出发，提出了一种“算法+知识+数据+算力”的算力资源智慧分配模型，该模型属于多层次、多功能的算力网络体系架构。尽管以上研究成果都对算力资源分配进行了探讨，但都存在一定程度的不足：（1）简单性算力请求需要在同一时刻进行群组式资源分配，而以上文献都未明确阐述是否支持这种特殊的资源分配方式；（2）以上文献都是从供给端关注资源分配的调度、管理以及优化，关注如何通过优化目标参数达到更优的资源分配效果，并未对底层实现方式进行探讨，这与本文的出发点不相符。

从算力网络路由的角度分析。文献[15]提出了一种基于集成图卷积网络（graph convolutional network，GCN）和深度Q网络（deep Q-network，DQN）的任务自适应路由算法，但该算法讨论的核心是如何将GCN集成到DQN模型中，并未涉及路径选择层面。文献[16]提出了一种基于图神经网络（graph neural network，GNN）和DQN相结合的多域算力网络路径规划模型，用于解决传输时备选路径的选择问题，但该模型主要关注的是如何利用机器学习提高传输成功率，对于组路径选择并未进行解释。文献[17]尝试将算力网络

与时间敏感型网络相结合，提出了一种基于调度与路由混合优化的智能方案，但该方案的重心在于基于深度学习对算力资源调度问题的探讨，而弱化了对算力路由的分析。文献[18]针对天地一体化算力网络提出了一种分散式算力路由算法，但该算法具有特定的领域性和特殊性，无法直接应用于本文所讨论的问题。文献[19]提出了一种信道感知路由协议（channel-aware routing protocol，CARP），该协议的特点是将计算信息作为成本引入网络，从而保证了最终用户的服务质量（quality of service，QoS），但该协议是一种类似单播的协议，在面对群组式请求时显得无能为力。由以上分析可知，目前算力网络路由的研究并未覆盖到本文所要讨论的问题。此外，物联网和工业互联网路由也值得关注。例如，文献[20]讨论了物联网环境下云侧与边缘侧的工作协同问题，并提出了一种启发式算法来解决，该算法能够有效解决云与边缘之间的多命令传输问题。文献[21]介绍了动态环境下工业互联网的群组命令传输问题，并提出通过动态遗传算法进行解决，该算法实现了网络动态环境下的群组命令传输。文献[20-21]同样不适用于本文所讨论的问题，原因是其中讨论的网络环境、传输对象以及实现手段都未满足本文问题要求。

综上，现有的研究成果不仅无法有效解决简单性请求的资源分配问题，也无法表达HCFRR的思想。因此，本文通过分析HCFRR问题的特性，提出了一种启发式算法，即优化型遗传算法（optimized genetic algorithm，OGA），旨在有效解决这一问题。

相对于已有的路由机制，本文的创新点在于：（1）从形式上看，本文所提算法是一种为了实现算力资源分配的路由策略，属于多对多的路径选择机制，即为算力网络中一组异构算力请求选择一组到达各自目的算力节点的有效路径；（2）从内容上看，本文所提算法求解的是一组没有链路相交、相互独立的路径，从而满足了算力请求的独特性。

本文主要工作包括：（1）讨论了算力网络的基础性及关键性技术问题——HCFRR问题，并对该问题进行形式化建模。（2）针对HCFRR问题，提出优化型遗传算法OGA予以解答。算法从局部和全局两个层面进行设计。在局部层面，为提升收敛度，算法设计时，分别对种群初始化、选择、交叉以及变异等操作进行优化；在全局层面，为避免路径间相交，算法构建了求解矩阵，并设计了相应的矩阵遍历方法。最后，从理论层面证明了算法的有效性。（3）从异构算力请求的传输成功率、算法收敛延时比、算力网络负载均衡误差率3个方面进行实验，验证了该算法的合理性及有效性。

1 问题描述

1.1 网络结构模型

限于篇幅，本文暂不考虑在算力网络前后动态变化下的HCFRR问题，在此条件下算力网络可定义为三元组，即$\mathrm{CFN}=\langle \mathrm{CF},\mathrm{CFNR},\mathrm{E}\rangle$。CFN拓扑结构如图1所示。

图1中，CF为算力节点集，CF_j表示CFN中第$j(j=1,2,3,\cdots)$个算力节点，每个CF_j上拥有不同类型的算力资源（包括CPU、GPU、NPU等）。CFNR为网络路由节点集，CFNR_j代表CFN中第$j(j=1,2,3,\cdots)$个网络路由节点（其与Internet的路由节点是有区别的，不仅要记录网络上的时延（Delay）、带宽（Bandwidth）、丢包率（Lost）等信息，还要记录网络上所有算力节点的算力资源信息以及下一跳路由的信息）。E为算力网络中所有链路集合，对任意一条链路$e_j\in E(j=1,2,3,\cdots)$，存在多种传统度量参数（包括Delay、Bandwidth、Lost等）。

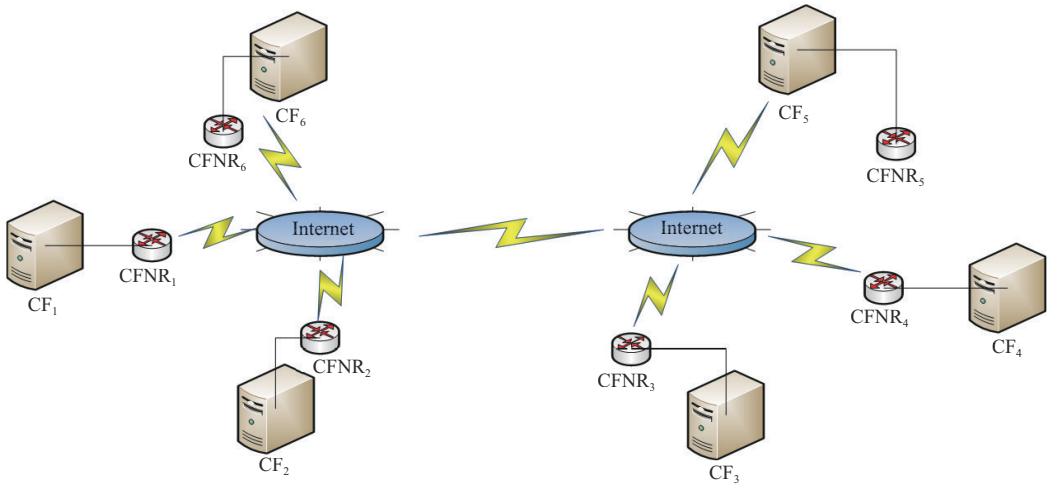

图 1　CFN 拓扑结构

1.2　数学模型描述

基于图 1 的算力网络拓扑结构，假定 CFN 上存在一组异构算力请求 Reqst，任意一个算力请求 $\text{Reqst}_i \in \text{Reqst}(i=1,2,3,\cdots)$ 上存在一组需求约束集（包括算力需求约束、时延需求约束、丢包率需求约束、带宽需求约束等），基于上述算力需求约束，通过匹配策略[11-14,22-23]确定一组目的算力节点集 $\text{CF}_{\text{sub}} \in \text{CF}$，则对单一算力请求 Reqst_i 而言，其路由问题可描述为：寻找一条满足各类约束的路径 P_{Reqst_i}，使 Reqst_i 能够达到其目的算力节点 $\text{CF}_j \in \text{CF}_{\text{sub}}$。数学模型描述为：

$$\underset{\text{CF}_j}{\text{goal}}: P_{\text{Reqst}_i} = \langle e_1, e_2, \cdots, e_k \rangle$$
$$\text{s.t.} \begin{cases} \min \text{Bandwidth}\left(P_{\text{Reqst}_i}\right) \leqslant \text{Bandwidth}_{\text{lim}} \\ \sum \text{Delay}\left(P_{\text{Reqst}_i}\right) \leqslant \text{Delay}_{\text{lim}} \\ 1 - \prod\left(1 - \text{Lost}\left(P_{\text{Reqst}_i}\right)\right) \leqslant \text{Lost}_{\text{lim}} \end{cases} \quad (1)$$

其中，$e_1, e_2, \cdots, e_k \in E$ 表示链路，$\langle \cdot \rangle$ 表示多元有序关系，$\text{Delay}_{\text{lim}}$、$\text{Bandwidth}_{\text{lim}}$、$\text{Lost}_{\text{lim}}$ 分别表示时延、带宽及丢包率的约束值。

HCFRR 问题：对于一组算力请求 Reqst 而言，寻找一组没有传输链路相交的有效路径 P_{Reqst}，使得 Reqst 能够达到各自的目的算力节点 CF_{sub}。数学模型描述为：

$$\underset{\text{CF}_{\text{sub}}}{\text{goal}}: P_{\text{Reqst}} = P_{\text{Reqst}_1} \cup P_{\text{Reqst}_2} \cup \cdots \cap P_{\text{Reqst}_n}$$
$$\text{s.t.} \begin{cases} \forall i,j, \left(P_{\text{Reqst}_i} \cap P_{\text{Reqst}_j} = \varnothing\right) \cup \left(P_{\text{Reqst}_i} \neq P_{\text{Reqst}_j}\right) \\ i,j = 1,2,\cdots,n \end{cases} \quad (2)$$

其中，n 代表算力请求的总数。综上分析可知，HCFRR 问题属于非确定性多项式（nondeterministic polynomial，NP）完全问题。该问题是一类难解问题，本文提出了一种优化型遗传算法（optimized genetic algorithm，OGA）予以解决。

2　算法描述

2.1　传统经典遗传算法

传统经典遗传算法（genetic algorithm，GA）[24]是借鉴大自然中生物界自主进化思想而延伸出的一种全局概率优化搜索算法，广泛应用于自适应控制、组合优化、工程制造等领域。

本文采用 GA 算法作为基础算法的理由：HCFRR 问题本质上是在实施路径选择，对任意一条链路而言，如果被选择则设为 1，如果未被选择则设为 0，按此思想，一条完整的有效路径其实就是一组 0 或 1 的有序组合，再结合

GA 算法的内部操作，可以较为便捷地搜索到目标的解。总之，GA 算法不但能方便地表达每条路径（或解），而且能通过内部操作较容易地实现解进化、解迭代，求解方式优于其他启发式算法，因此，本文优先选择 GA 算法作为基础算法。

然而，GA 算法在解决 HCFRR 问题时，存在以下弊端：（1）在局部层面，对于单一算力请求的路径选择，GA 算法无法快速收敛到目标解上；（2）在全局层面，对于 HCFRR 问题，GA 算法无法有效构建起路径之间链路不相交的解决方案。基于上述弊端，本文提出了 OGA，旨在优化相关问题的同时，解决 HCFRR 问题。

2.2 OGA 思想

OGA 思想从两个层面阐述：（1）在局部优化层面，OGA 要尽量避免搜索的偏向性，使算法能够快速收敛。首先，采用单参数满足随机性策略初始化种群，目的是尽可能扩散种群的分布范围；其次，采用多参数解（或路径）均衡选择策略实现选择操作，目的是确保被选种群类型更加丰富；最后，分别采用两层交叉策略与多参数随机单点变异策略执行交叉操作与变异操作，目的是从全域和局域两个维度提升算法的搜索能力。（2）在全局优化层面，为避免路径之间相互竞争与冲突，OGA 制定了路径分发策略，该策略通过构建求解矩阵，并以行（row）为单位遍历该矩阵，结合评价函数、可行解的随机选择以及低需求优先避让原则等手段，确保最终能够寻找到一组满足条件的可行解集。

2.3 OGA 局部层面设计

局部层面，OGA 需要考虑对单一算力的请求如何快速收敛到目标解，设计如下。

2.3.1 编码方式

无论是单一算力请求问题，还是 HCFRR 问题，都是路径选择问题，当链路被选择时则用二进制数“1”表示，未被选中时则用二进制数“0”表示，那么一条路径（或一个候选解）可用一组 0 或 1 的二进制数表达。依据该思想，本文首选二进制数编码为算法的编码方式。

2.3.2 种群初始化

对任意单一算力请求 Reqst_i 而言，采用单参数满足随机性策略初始化种群，以式（1）中的参数为例，具体过程如下。

步骤 1 选取 Bandwidth，以该参数为中心，初始化一组子种群，该子种群需要满足以下特性：（1）子种群中任意一个解（或路径）P_{Reqst_i} 的 $\mathrm{Bandwidth}\left(P_{\mathrm{Reqst}_i}\right)$ 取值要满足 Reqst_i 的带宽约束 $\mathrm{Bandwidth}_{\mathrm{lim}}$；（2）$P_{\mathrm{Reqst}_i}$ 的其他参数（包括 $\mathrm{Delay}\left(P_{\mathrm{Reqst}_i}\right)$、$\mathrm{Lost}\left(P_{\mathrm{Reqst}_i}\right)$）随机性取值，转入步骤 2。

步骤 2 选取 Delay，以该参数为中心，依照步骤 1 的思想初始化一组关于满足时延约束 $\mathrm{Delay}_{\mathrm{lim}}$ 的子种群，转入步骤 3。

步骤 3 选取 Lost，以该参数为中心，依照步骤 1 的思想初始化一组关于满足丢包率约束 $\mathrm{Lost}_{\mathrm{lim}}$ 的子种群，转入步骤 4。

步骤 4 判定经过 3 轮初始化后的种群规模是否满足条件，如果不满足，则转入步骤 1，否则退出初始化过程。

该策略的优势在于使得种群更加广泛地分布在解空间，为后期解向未知空间进化提供可能。

2.3.3 单一请求的适应度函数

单一请求 $[\mathrm{Reqst}]_i$ 的适应度函数用 f 表示，用于评价关于该请求的每个被选解（或路径）P_{Reqst_i} 的优劣。如果被选解（或路径）$P_{\mathrm{Re\,qst}_i}$ 全部满足约束条件，适应度函数值则为 1；反之，适应度函数值大于 1。适应度函数的数学模型如下所示：

$$f\left(P_{\mathrm{Reqst}_i}\right)=\frac{1}{\mathrm{Bandwidth}\left(P_{\mathrm{Reqst}_i}\right)\cdot\mathrm{Delay}\left(P_{\mathrm{Reqst}_i}\right)\cdot\mathrm{Lost}\left(P_{\mathrm{Reqst}_i}\right)}$$

$$\text{s.t.}\begin{cases}\mathrm{Bandwidth}\left(P_{\mathrm{Reqst}_i}\right)=\begin{cases}1,\ \mathrm{Bandwidth}\left(P_{\mathrm{Reqst}_i}\right)\leqslant\mathrm{Bandwidth}_{\mathrm{lim}}\\ R_{\mathrm{Bw}},\ \text{其他}\end{cases}\\ \mathrm{Delay}\left(P_{\mathrm{Reqst}_i}\right)=\begin{cases}1,\ \mathrm{Delay}\left(P_{\mathrm{Reqst}_i}\right)\leqslant\mathrm{Delay}_{\mathrm{lim}}\\ R_{\mathrm{Dy}},\ \text{其他}\end{cases}\\ \mathrm{Lost}\left(P_{\mathrm{Reqst}_i}\right)=\begin{cases}1,\ \mathrm{Lost}\left(P_{\mathrm{Reqst}_i}\right)\leqslant\mathrm{Lost}_{\mathrm{lim}}\\ R_{\mathrm{Lt}},\ \text{其他}\end{cases}\end{cases}\tag{3}$$

其中，R_{Bw}、R_{Dy}、R_{Lt}分别表示带宽、时延及丢包率的惩罚程度。

2.3.4　选择策略

采用多参数解（或路径）均衡选择策略，即以每一个参数为参照物，分别选择除满足该参数约束外适应度函数值较小的解（或路径）和适应度函数值较大的解（或路径）。同时，对于同一类被选择的解（或路径）要尽可能地分散，保持一定距离。该策略的目的是保证被选种群更加丰富化和多样化，从而避免偏向性搜索导致的无法收敛。具体过程如下。

（1）解（或路径）的选择

步骤 1　以Bandwidth为例，首先，对于解空间（或种群空间）中所有的解进行筛选，剔除满足该参数约束$\mathrm{Bandwidth}_{\mathrm{lim}}$的解，对剩余的所有解，根据式（3）分别计算其适应度函数值，并基于式（4）计算出在该参数下的适应度函数均值$\bar{f}_{\mathrm{Bandwidth}}$：

$$\bar{f}_{\mathrm{Bandwidth}}=\frac{\sum_j f\left(P^j_{\mathrm{Reqst}_i}\right)}{\mathrm{Num}}\tag{4}$$

其中，$P^j_{\mathrm{Reqst}_i}$表示第j个候选解，Num表示剩余所有解的总数。

根据$\bar{f}_{\mathrm{Bandwidth}}$的计算结果，分别随机选择一组适应度函数值小于$\bar{f}_{\mathrm{Bandwidth}}$的解（或路径）以及选择一组适应度函数值大于$\bar{f}_{\mathrm{Bandwidth}}$的解（或路径）。

步骤 2　同理，按照步骤1的执行过程，依次计算Delay和Lost的适应度函数均值$\bar{f}_{\mathrm{Delay}}$和$\bar{f}_{\mathrm{Lost}}$，并根据$\bar{f}_{\mathrm{Delay}}$和$\bar{f}_{\mathrm{Lost}}$的计算结果，各自随机选择出一组小于和一组大于均值的解（或路径）。

（2）同一类解（或路径）分散策略

根据算法思想，对任意被选解（或路径）P_{Reqst_i}而言，要使其尽可能远离同类的其他被选解（或路径），具体过程如下。

令$\mathrm{Dis}\left\langle P_{\mathrm{Reqst}_i},\cdot\right\rangle$表示为被选解（或路径）$P_{\mathrm{Reqst}_i}$在同一种类中的拥挤程度，则可表示为：

$$\begin{cases}\mathrm{Dis}_{\mathrm{Bandwidth}}\left\langle P_{\mathrm{Reqst}_i},P_{\mathrm{Reqst}_j}\right\rangle=1\Big/\sqrt{\left|\mathrm{Bandwidth}\left(P_{\mathrm{Reqst}_i}\right)-\mathrm{Bandwidth}\left(P_{\mathrm{Re\,qst}_j}\right)\right|^2}\\ \mathrm{Dis}_{\mathrm{Delay}}\left\langle P_{\mathrm{Reqst}_i},P_{\mathrm{Reqst}_j}\right\rangle=1\Big/\sqrt{\left|\mathrm{Delay}\left(P_{\mathrm{Reqst}_i}\right)-\mathrm{Delay}\left(P_{\mathrm{Reqst}_j}\right)\right|^2}\\ \mathrm{Dis}_{\mathrm{Lost}}\left\langle P_{\mathrm{Reqst}_i},P_{\mathrm{Reqst}_j}\right\rangle=1\Big/\sqrt{\left|\mathrm{Lost}\left(P_{\mathrm{Reqst}_i}\right)-\mathrm{Lost}\left(P_{\mathrm{Reqst}_j}\right)\right|^2}\\ \mathrm{Dis}\left\langle P_{\mathrm{Reqst}_i},P_{\mathrm{Reqst}_j}\right\rangle=\left(\begin{matrix}\mathrm{Dis}_{\mathrm{Bandwidth}}\left\langle P_{\mathrm{Reqst}_i},P_{\mathrm{Reqst}_j}\right\rangle+\mathrm{Dis}_{\mathrm{Delay}}\left\langle P_{\mathrm{Reqst}_i},P_{\mathrm{Reqst}_j}\right\rangle+\\ \mathrm{Dis}_{\mathrm{Lost}}\left\langle P_{\mathrm{Reqst}_i},P_{\mathrm{Reqst}_j}\right\rangle\end{matrix}\right)^{-1}\end{cases}\tag{5}$$

$$\mathrm{Dis}\left\langle P_{\mathrm{Reqst}_i},\cdot\right\rangle=\sum_{j\cup i\neq j}\mathrm{Dis}\left\langle P_{\mathrm{Reqst}_i},P_{\mathrm{Reqst}_i}\right\rangle \quad (6)$$

其中，$\mathrm{Dis}\left\langle P_{\mathrm{Reqst}_i},\cdot\right\rangle$中“·”表示为同一种类中任意被选解（或路径），$\mathrm{Dis}\left\langle P_{\mathrm{Reqst}_i},P_{\mathrm{Reqst}_j}\right\rangle$表示被选解（或路径）$P_{\mathrm{Reqst}_i}$与同一类的$P_{\mathrm{Reqst}_j}$之间的距离，$\mathrm{Dis}_{\mathrm{Bandwidth}}\left\langle P_{\mathrm{Reqst}_i},P_{\mathrm{Reqst}_j}\right\rangle$、$\mathrm{Dis}_{\mathrm{Delay}}\left\langle P_{\mathrm{Reqst}_i},P_{\mathrm{Reqst}_j}\right\rangle$、$\mathrm{Dis}_{\mathrm{Lost}}\left\langle P_{\mathrm{Reqst}_i},P_{\mathrm{Reqst}_j}\right\rangle$分别表示$P_{\mathrm{Reqst}_i}$与$P_{\mathrm{Reqst}_j}$在带宽、时延以及丢包率维度上的距离。$\mathrm{Dis}\left\langle P_{\mathrm{Re\,qst}_i},\cdot\right\rangle$越大，说明被选解（或路径）$P_{\mathrm{Reqst}_i}$的拥挤度越小，分散程度越高。当分散程度的结果满足阈值Q_0时，则认为该P_{Reqst_i}可以被选。其中，Q_0为拥挤程度的阈值。

2.3.5 交叉与变异策略

（1）交叉策略

交叉策略影响到全域搜索，为提升算法的全域搜索能力，本文采用两层交叉策略，具体过程如下。

一是内层交叉策略。在同一个参数下（包括Bandwidth、Delay、Lost），分别随机选择适应度函数值较小的解（或路径）和适应度函数值较大的解（或路径），让其两两组队；依次遍历这两个被选解的基因位，根据两者之间的适应度函数值比例$\overline{f_R}$确定每一基因位的交叉概率p_f，如果p_f为1，则两个被选解的基因位相互对换，如下所示：

$$\begin{cases}\overline{f_R}=\dfrac{f\left(P_{\mathrm{Reqst}_i}\right)}{f\left(P_{\mathrm{Reqst}_j}\right)}\\ p_f=\begin{cases}1,\ \overline{f_R}\geqslant Q_R\\ 0,\ \text{其他}\end{cases}\end{cases} \quad (7)$$

其中，P_{Reqst_i}、P_{Reqst_j}分别表示较小值和较大值的被选解，Q_R为随机数。

该思想的优势在于，交叉时尽可能使每一基因位平等地参与到交叉操作中，增加了全域搜索的可能。

二是外层交叉策略。对于内层交叉的结果，以同一参数为维度，分别按照适应度函数值的大小进行排序；分别从不同的参数（包括Bandwidth、Delay、Lost）中各自选择适应度函数值较小的解（或路径）和适应度函数值较大的解（或路径）两两混合组队，并采用随机两点交叉策略实现外层交叉操作。

内层和外层的交叉操作，能够有效提升算法的全域搜索能力。

（2）变异策略

变异策略影响到局域搜索，为提升算法的局域搜索能力，进一步扩散种群所处的范围，本文采用多参数随机单点变异策略，具体过程如下。

步骤1 基于交叉操作的结果，将新产生的种群按Bandwidth、Delay、Lost这3个参数随机分成3等份，然后按该参数值的大小进行排序。

步骤2 以这3个参数为维度，分别按照式（8）计算其相应的子种群的参数总量，具体如下所示：

$$\begin{cases}\mathrm{Bandwidth}_{\mathrm{Total}}=\sum\limits_i\mathrm{Bandwidth}\left(P_{\mathrm{Reqst}_i}\right)\\ \mathrm{Delay}_{\mathrm{Total}}=\sum\limits_i\mathrm{Delay}\left(P_{\mathrm{Reqst}_i}\right)\\ \mathrm{Lost}_{\mathrm{Total}}=\sum\limits_i\mathrm{Lost}\left(P_{\mathrm{Reqst}_i}\right)\end{cases} \quad (8)$$

其中，$\mathrm{Bandwidth}_{\mathrm{Total}}$、$\mathrm{Delay}_{\mathrm{Total}}$、$\mathrm{Lost}_{\mathrm{Total}}$分别为带宽、时延及丢包率的参数总量。

步骤3 按照下面的式（9）计算每个被选解（或路径）P_{Reqst_i}在该参数下的被选执行变异操作的概率：

$$\begin{cases}\mathrm{Pro}_{P_{\mathrm{Reqst}_i}}^{\mathrm{Bandwidth}}=\dfrac{\mathrm{Bandwidth}\left(P_{\mathrm{Reqst}_i}\right)}{\mathrm{Bandwidth}_{\mathrm{Total}}}\\ \mathrm{Pro}_{P_{\mathrm{Reqst}_i}}^{\mathrm{Delay}}=\dfrac{\mathrm{Delay}\left(P_{\mathrm{Reqst}_i}\right)}{\mathrm{Delay}_{\mathrm{Total}}}\\ \mathrm{Pro}_{P_{\mathrm{Reqst}_i}}^{\mathrm{Lost}}=\dfrac{\mathrm{Lost}\left(P_{\mathrm{Reqst}_i}\right)}{\mathrm{Lost}_{\mathrm{Total}}}\end{cases} \quad (9)$$

其中，$\mathrm{Pro}_{P_{\mathrm{Reqst}_i}}^{\mathrm{Bandwidth}}$、$\mathrm{Pro}_{P_{\mathrm{Reqst}_i}}^{\mathrm{Delay}}$、$\mathrm{Pro}_{P_{\mathrm{Reqst}_i}}^{\mathrm{Lost}}$分别表示$P_{\mathrm{Reqst}_i}$在各参数下的被选执行变异操作的概率。

步骤4 根据式（9）的计算结果，在各自参数维度下，将每个P_{Reqst_i}的被选执行变异操作的

概率从大到小进行排序，并按照式（10）计算每个P_{Reqst_i}的累积被选概率：

$$\begin{cases}\overline{\mathrm{Pro}_{P_{\mathrm{Reqst}_i}}^{\mathrm{Bandwidth}}}=\sum_{j=1}^{i}\mathrm{Pro}_{P_{\mathrm{Reqst}_j}}^{\mathrm{Bandwidth}}\\ \overline{\mathrm{Pro}_{P_{\mathrm{Reqst}_i}}^{\mathrm{Delay}}}=\sum_{j=1}^{i}\mathrm{Pro}_{P_{\mathrm{Reqst}_j}}^{\mathrm{Delay}}\\ \overline{\mathrm{Pro}_{P_{\mathrm{Reqst}_i}}^{\mathrm{Lost}}}=\sum_{j=1}^{i}\mathrm{Pro}_{P_{\mathrm{Reqst}_j}}^{\mathrm{Lost}}\end{cases}\tag{10}$$

其中，$\overline{\mathrm{Pro}_{P_{\mathrm{Reqst}_i}}^{\mathrm{Bandwidth}}}$、$\overline{\mathrm{Pro}_{P_{\mathrm{Reqst}_i}}^{\mathrm{Delay}}}$、$\overline{\mathrm{Pro}_{P_{\mathrm{Reqst}_i}}^{\mathrm{Lost}}}$分别表示$P_{\mathrm{Reqst}_i}$在各自参数下的累积被选概率。

步骤 5 创建一个随机数，如果该随机数大于或等于式（10）的计算结果，则取P_{Reqst_i}作为待变异个体。

步骤 6 基于随机单点变异思想，对P_{Reqst_i}进行变异操作。

变异操作的目的是进一步多样化种群，尽可能提高算法的局域深挖能力。

2.4　OGA 全局层面设计

全局层面，OGA 需要考虑对一组算力的请求如何避免路径之间链路相交，设计如下。

2.4.1　HCFRR 评价函数

HCFRR 问题的评价函数 *Gf* 用于评价求解 HCFRR 问题的每组被选解（或路径）的优劣。一组被选解（或路径）的评价值为 1，说明满足 HCFRR 问题的所有约束条件；反之，评价值则为 0。式（11）采用数学模型阐述评价函数：

$$Gf\left(P_{\mathrm{Reqst}_1},P_{\mathrm{Reqst}_2},\cdots,P_{\mathrm{Reqst}_n}\right)=\sqrt{\prod_{i,j}\mathrm{In}\left(P_{\mathrm{Reqst}_i},P_{\mathrm{Reqst}_j}\right)\cdot\mathrm{Neql}\left(P_{\mathrm{Reqst}_i},P_{\mathrm{Reqst}_j}\right)}$$

$$\text{s.t.}\begin{cases}\mathrm{In}\left(P_{\mathrm{Reqst}_i},P_{\mathrm{Reqst}_j}\right)=\begin{cases}1,(\forall i,j)P_{\mathrm{Reqst}_i}\cap P_{\mathrm{Reqst}_j}=\varnothing\\0,\text{其他}\end{cases}\\ \mathrm{Neql}\left(P_{\mathrm{Reqst}_i},P_{\mathrm{Reqst}_j}\right)=\begin{cases}1,(\forall i,j)P_{\mathrm{Reqst}_i}\neq P_{\mathrm{Reqst}_j}\\0,\text{其他}\end{cases}\end{cases}\tag{11}$$

其中，P_{Reqst_i}、P_{Reqst_j}分别表示被选路径，$\mathrm{In}(\cdot)$表示相交函数，$\mathrm{Neql}(\cdot)$表示等价函数。

2.4.2　路径分发策略

假定算力网络 CFN 中存在一组算力资源请求$\mathrm{Reqst}=\{\mathrm{Reqst}_1,\mathrm{Reqst}_2,\cdots,\mathrm{Reqst}_n\}$，对任意一个算力资源请求$\mathrm{Reqst}_i\in\mathrm{Reqst}$而言，存在一组关于$\mathrm{Reqst}_i$的可行路径集$\left\{P_{\mathrm{Reqst}_i}^1,P_{\mathrm{Reqst}_i}^2,\cdots,P_{\mathrm{Reqst}_i}^m\right\}$，$n$和$m$分别表示算力请求总数和路径总数，则可构建起HCFRR问题的$n\times m$求解矩阵 **HCFRR_Matrix**，如下所示：

$$\mathbf{HCFRR\_Matrix}=\begin{bmatrix}P_{\mathrm{Reqst}_1}^1,P_{\mathrm{Reqst}t_1}^2,\cdots,P_{\mathrm{Reqst}_1}^m\\P_{\mathrm{Reqst}_2}^1,P_{\mathrm{Reqst}_2}^2,\cdots,P_{\mathrm{Reqst}_2}^m\\\cdots\\P_{\mathrm{Reqst}_i}^1,P_{\mathrm{Reqst}_i}^2,\cdots,P_{\mathrm{Reqst}_i}^m\\\cdots\\P_{\mathrm{Reqst}_n}^1,P_{\mathrm{Reqst}_n}^2,\cdots,P_{\mathrm{Reqst}_n}^m\end{bmatrix}$$

其中，矩阵中每一行表示一个算力请求的可行路径集。根据对矩阵 **HCFRR_Matrix** 的描述，可通过依次对矩阵进行遍历从而求得问题所要的一组互不相交的路径。具体的路径分发策略过程如下。

步骤 1 设定目标路径集 *Gp*，该变量用于存储最终的求解结果。

步骤 2 以行为单位依次遍历矩阵 **HCFRR_Matrix**，并从每行中随机挑选出一个可行解$P_{\mathrm{Reqst}_i}^j$存入变量集合 *Gp* 中。其中，$i\in[1,n]$，$j\in[1,m]$。

步骤 3 将 *Gp* 中所有的元素，代入式（11）HCFRR 问题的评价函数 *Gf* 中，计算 *Gf* 的函数值。如果 *Gf*=1 且 *Gp* 中已存入 *n* 个元素，则返回集合 *Gp*，说明该集合为满足 HCFRR 问题的最终求解结果。

步骤4 如果 $Gf=1$ 但 Gp 集中未存入 n 个元素，则按照步骤2继续以行为单位遍历矩阵 **HCFRR_Matrix**，并随机选中可行解存入 Gp 中，并按照步骤3进行判断。

步骤5 如果 $Gf=0$，则对产生竞争冲突的可行解进行标记，并对竞争冲突的可行解进行比较，按照低需求先避让原则，挑选出对算力网络资源（包括Bandwidth、Delay、Lost等）需求低的算力请求，并重新遍历矩阵 **HCFRR_Matrix**，挑选出其所对应的新可行解存入 Gp 中，并按照步骤3进行判断。

2.5 OGA过程

步骤1 对HCFRR问题进行分解，分解成多个单一算力请求问题，确定每个问题所对应的约束条件并搜索求解，转入步骤2。

步骤2 依据OGA局部设计思想，对每个问题确定其编码方式和适应度函数，初始化参数 R_{Bw}、R_{Dy}、R_{Lt}、Q_0、Q_R 及种群规模，设定迭代次数与退出条件，转入步骤3。

步骤3 依据第2.3.2节，并判断是否满足条件，如果满足，则转入步骤4。

步骤4 依据第2.3.4节及式（4）～式（6）选择迭代种群，并转入步骤5。

步骤5 对被选种群按照第2.3.5节及式（7）～式（10）实现交叉与变异操作，产生下一代种群，并判断退出条件及记录满足约束条件的候选解（或路径），如果满足，则转入步骤6；否则转入步骤4。

步骤6 根据OGA全局设计思想，合并每个问题所搜索到的可行解（或路径）集，构建求解矩阵 **HCFRR_Matrix**，再根据第2.4.2节为每个问题分配一个满足约束条件的可行解，由此计算出目标路径集 Gp 并退出。

2.6 算法实例描述

为举例方便，以图1中的算力网络拓扑结构为例，假定算力节点 CF_1 上存在2个算力资源请求，算力节点 CF_2 上存在1个算力资源请求，算力请求情况见表1。

同时，路由节点 $CFNR_1$ 与 $CFNR_2$ 的路由情况，分别见表2、表3。

根据以上各路由节点中的路由情况，采用类似于开放式最短路径优先（open shortest path first，OSPF）策略，可构建到达每个目的算力节点的所有路径及其各参数信息，见表4。需要说明的是，

表1 算力请求情况

请求序号	请求类型	源点	请求时延/ms	请求带宽/（MB·s^{-1}）	丢包率	请求算力资源/GFlops
$Reqst_1$	GPU资源请求	CF_1	50	90	2%	43
$Reqst_2$	NPU资源请求	CF_1	30	50	1.1%	70
$Reqst_3$	GPU资源请求	CF_2	55	70	1.5%	30

表2 $CFNR_1$ 的路由情况

下一跳	时延/ms	带宽/（MB·s^{-1}）	丢包率	目的算力节点	CPU、GPU、NPU/GFlops
$CFNR_1$	-	-	-	CF_1	5、0、10
$CFNR_2$	20	100	0.5%	CF_2	60、0、20
$CFNR_2$	20	100	0.5%	CF_3	0、10、10
$CFNR_4$	15	50	1.1%	CF_4	0、15、85
$CFNR_6$	30	150	0.8%	CF_5	0、0、15
$CFNR_6$	30	150	0.8%	CF_6	10、90、0

表3　$CFNR_2$的路由情况

下一跳	时延/ms	带宽/（$MB·s^{-1}$）	丢包率	目的算力节点	CPU、GPU、NPU/GFlops
$CFNR_1$	20	100	0.5%	CF_1	5、0、10
$CFNR_2$	-	-	-	CF_2	60、0、20
$CFNR_3$	25	70	0.7%	CF_3	0、10、10
$CFNR_3$	25	70	0.7%	CF_4	0、15、85
$CFNR_5$	20	150	0.3%	CF_5	0、0、15
$CFNR_6$	15	100	1.0%	CF_6	10、90、0

表4　路径及其各参数信息

源点	路径	目的算力节点	时延/ms	带宽/（$MB·s^{-1}$）	丢包率
CF_1	$CFNR_1$→$CFNR_2$	CF_2	20	100	0.5%
	$CFNR_1$→$CFNR_4$	CF_4	15	50	1.1%
	$CFNR_1$→$CFNR_6$	CF_6	30	150	0.8%
	$CFNR_1$→$CFNR_2$→$CFNR_6$	CF_6	35	100	1.5%
CF_2	$CFNR_2$→$CFNR_1$	CF_1	20	100	0.5%
	$CFNR_2$→$CFNR_6$	CF_6	15	100	1.0%
	$CFNR_2$→$CFNR_1$→$CFNR_6$	CF_6	50	100	1.2%

表4只显示部分路径，此处的OSPF策略在寻找每个目的算力节点路径时，列出了所有可能达到的路径。

具体过程如下。

（1）OGA在执行前，先根据文献[11-14]及文献[22-23]的策略选择好$Reqst_1$、$Reqst_2$、$Reqst_3$所对应的算力节点CF_6、CF_4、CF_6。

（2）OGA在执行时，会将3个请求同时单独分成单一算力请求问题处理，分别寻找CF_1到CF_6、CF_1到CF_4、CF_2到CF_6的所有可能的路径。表4中CF_1到CF_6的路径有2条，为$CFNR_1$→$CFNR_6$和$CFNR_1$→$CFNR_2$→$CFNR_6$；CF_1到CF_4的路径只有1条，为$CFNR_1$→$CFNR_4$；CF_2到CF_6的路径有2条，为$CFNR_2$→$CFNR_6$和$CFNR_2$→$CFNR_1$→$CFNR_6$。

（3）依据路径分发策略，以上多条路径被构建成求解矩阵**HCFRR_Matrix**，并依次随机遍历，如$Reqst_1$取路径$CFNR_1$→$CFNR_6$，$Reqst_2$取路径$CFNR_1$→$CFNR_4$，$Reqst_3$取路径$CFNR_2$→$CFNR_1$→$CFNR_6$，依次将这3条路径代入式（11）检验，$Gf=0$意味着$Reqst_1$和$Reqst_3$存在路径冲突，冲突点为$CFNR_1$→$CFNR_6$，按照低需求先避让原则，发现$Reqst_3$需求最低，因而需要重新选择，选择后的路径为$CFNR_2$→$CFNR_6$。

（4）重新选择后的3条路径为$CFNR_1$→$CFNR_6$、$CFNR_1$→$CFNR_4$、$CFNR_2$→$CFNR_6$，代入式（11）检验，$Gf=1$表明不再有冲突，说明这3条路径分别是请求$Reqst_1$、$Reqst_2$、$Reqst_3$的最终可行解。

依照实例分析，OGA能够为一组异构算力请求寻找到满足约束条件的可行路径，从而达到资源分配的目的。

2.7　算法有效性分析

证明如下。

（1）局部层面，OGA在初始化种群时，采用单参数满足随机性策略，该策略使得每组子种群能够在不同的参数维度上扩散，从而保证整个种群遍布在搜索空间中；选择操作时，多参数解（或路径）均衡选择策略保证被选种群在不同的参数维度上更加多样化，可有效缓解无法收敛这个问题；交叉操作时，内层交叉不仅保证交叉双方身份差异化，而且保证每位基因可平等参与到

交叉操作中，从而提升了全域搜索的能力，外层不同参数维度下差异化混合交叉思想又进一步增加了这种可能；变异操作时，多参数随机单点变异策略避免了全部种群参与变异操作而形成的无效局域搜索，提升了变异操作的局域搜索效率。因此，算法在局部层面是有效的。

（2）全局层面，OGA基于每个算力请求的搜索结果，构建起求解矩阵**HCFRR_Matrix**，最终求解过程围绕对**HCFRR_Matrix**遍历展开。**HCFRR_Matrix**是规模有限的，因而OGA必定会在多项式时间内收敛。同时，OGA中的路径分发策略在**HCFRR_Matrix**规模有限下，采用随机选择和低需求优先避让原则在式（11）作用下，必然可以在多项式时间内寻找到一组满足约束的最终解集。

综上，OGA是收敛且有效的。证毕。

3 实验与分析

硬件环境：1台曙光服务器，CPU型号为AMD Opteron 4334，内存为8 GB，SATA硬盘为1 TB，操作系统为SUSE Linux Enterprise Server 12。

网络结构：基于文献[25-26]所采用的Waxman网络仿真模型，随机生成包含19个算力节点、19个算力网络路由节点、153条网络链路的算力网络结构*cfn*。其中，每个算力节点上随机设定3种算力资源，包括CPU资源、GPU资源以及NPU资源，且各类算力所拥有的资源量随机生成。每条链路上的网络参数（包括Bandwidth、Delay、Lost）随机生成。每个算力网络路由节点拥有一张路由表，记录全网算力节点资源信息、下一跳以及网络参数信息。此处的网络结构可参考图1的拓扑结构，路由情况可参考表2、表3，全局的路径信息可参考第2.6节。

场景模拟如下。

（1）模拟异构算力请求：在*cfn*上，随机设定一组异构算力请求Reqst（包括CPU资源请求、GPU资源请求、NPU资源请求），且该组请求数量$N\in[25,40]$，并随机确定该组请求的约束条件，包括算力资源约束、时延约束、带宽约束以及丢包率约束等。

（2）确定Reqst所对应的目的算力节点：由于本文所关注的重心在于构建异构算力请求的路由，而关于目的算力节点的选择并不在本文讨论范围之内，因此本文采用现有的分配策略[11-14, 22-23]选择满足算力资源约束的目的算力节点。当前，这些已有的策略在选择目的算力节点方面都具有较好的性能。

设置参数如下。

在综合衡量现有成果对参数取值的建议以及基于多次重复实验的结果下，按如下方式设定参数[1-6, 17, 20-21]：$R_{\mathrm{Bw}},R_{\mathrm{Dy}},R_{\mathrm{Lt}}\in(0.19,0.89)$分别表示带宽、时延及丢包率的惩罚程度，$Q_0$取值为间隔距离均值的倒数，$Q_R\in(0,1)$表示随机数，种群规模设定为60～70，迭代次数设定为80～110。

为体现算法的有效性，本文分别选取文献[17]基于深度强化学习的路由（reward-back deep Q-learning，RBDQN）算法和文献[20]群组命令路径选择（improved genetic algorithm for group command transfer，IGAGCT）算法与OGA对比。

测试性能指标及实验结果如下。

（1）异构算力请求的传输成功率（heterogeneous request success rate，HRSR）表示为：

HRSR＝已成功传输的算力请求数量/异构算力请求传输的总量×100% （12）

该指标用来衡量各算法在算力$N=25$资源减少下对异构算力请求的传输成功率。

实验1：在异构算力请求总量以及算力网络链路数不断减少的条件下，测试各类算法的HRSR。请求总量为25下各类算法的HRSR变化情况如图2所示。

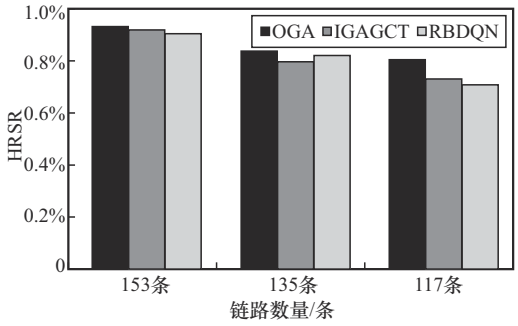

图2　请求总量为25时各类算法的HRSR变化情况

由图2可知，链路数量为153条时，OGA的HRSR接近91.5%，IGAGCT算法的HRSR接近90%，RBDQN算法的HRSR接近89.3%，OGA的HRSR分别降低了1.7%和2.5%；链路数量为135条时，OGA的HRSR接近83.6%，IGAGCT算法的HRSR接近77.9%，RBDQN算法的HRSR接近80%，OGA的HRSR分别降低了7.31%和5.6%；链路数量为117条时，OGA的HRSR接近78.9%，IGAGCT算法的HRSR接近71.3%，RBDQN算法的HRSR接近69.9%，OGA的HRSR分别降低了10.7%和12.9%。

实验2：在异构算力请求总量$N=40$以及算力网络链路数不断减少的条件下，测试各类算法的HRSR。请求总量为40时各类算法的HRSR变化情况如图3所示。

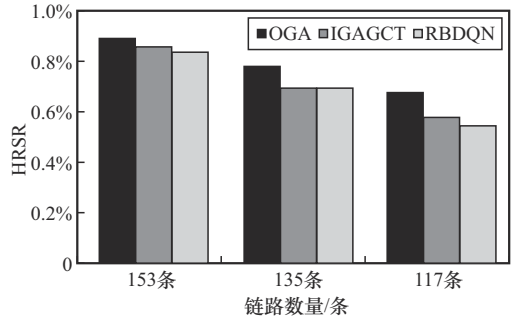

图3　请求总量为40时各类算法的HRSR变化情况

由图3可知，链路数量为153条时，OGA的HRSR接近89.1%，IGAGCT算法的HRSR接近85.2%，RBDQN算法的HRSR接近83.9%，OGA的HRSR分别降低了4.6%和6.2%；链路数量为135条时，OGA的HRSR接近77.2%，IGAGCT算法的HRSR接近68.8%，RBDQN算法的HRSR接近69.1%，OGA的HRSR分别降低了12.2%和11.7%；链路数量为117条时，OGA的HRSR接近67.5%，IGAGCT算法的HRSR接近57.9%，RBDQN算法的HRSR接近54.6%，OGA的HRSR分别降低了16.6%和23.6%。

（2）算法收敛时延比（algorithms convergence delay rate，ACDR）表示为：

ACDR＝(算法收敛时延－算法正常收敛时间)/算法正常收敛时间×100%　　（13）

该指标用来衡量各算法在算力资源减少下算法收敛的快慢程度。

实验3：在异构算力请求总量$N=25$以及算力网络链路数不断减少的条件下，测试各类算法的ACDR。请求总量为25时各类算法的ACDR变化情况如图4所示。

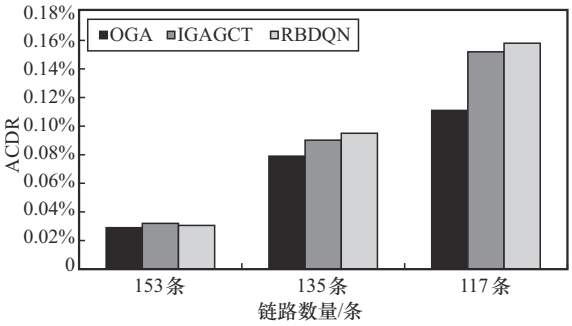

图4　请求总量为25时各类算法的ACDR变化情况

由图4可知，链路数量为153条时，OGA的ACDR接近2.9%，IGAGCT算法的ACDR接近3.2%，RBDQN算法的ACDR接近3.06%，OGA的ACDR分别降低了9.3%和5.2%；链路数量为135条时，OGA的ACDR接近7.9%，IGAGCT算法的ACDR接近9.02%，RBDQN算法的ACDR接近9.5%，OGA的ACDR分别降低了12%和16.8%；链路数量为117条时，OGA的ACDR接近11.1%，IGAGCT算法的ACDR接近15.2%，RBDQN算法的ACDR接近15.8%，OGA的

ACDR分别降低了27%和29.7%。

实验4：在异构算力请求总量$N=40$以及算力网络链路数不断减少的条件下，测试各类算法的ACDR。请求总量为40时各类算法的ACDR变化情况如图5所示。

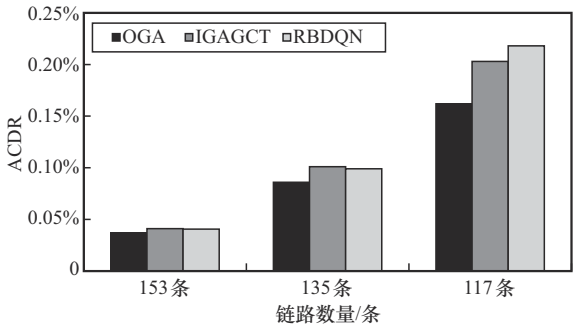

图5 请求总量为40时各类算法的ACDR变化情况

由图5可知，链路数量为153条时，OGA的ACDR接近3.7%，IGAGCT算法的ACDR接近4.1%，RBDQN算法的ACDR接近4.06%，OGA的ACDR分别降低了9.7%和8.8%；链路数量为135条时，OGA的ACDR接近8.6%，IGAGCT算法的ACDR接近10.1%，RBDQN算法的ACDR接近9.9%，OGA的ACDR分别降低了15%和13.1%；链路数量为117条时，OGA的ACDR接近16.2%，IGAGCT算法的ACDR接近20.3%，RBDQN算法的ACDR接近21.8%，OGA的ACDR分别降低了20.1%和25.6%。

（3）算力网络负载均衡误差率（CFN load error rate，CLER）表示为：

CLER＝(被测算法实际算出负载均衡程度－正常负载均衡程度)/正常负载均衡程度×100% （14）

该指标用来衡量各算法实际计算的网络负载均衡情况与正常网络负载均衡情况的误差程度。负载均衡程度间接地反映路径之间不相交的程度，即负载越均衡，则路径之间越可能不相交。

实验5：在算力网络资源不变（包含153条链路）以及异构算力请求总量不断增加的条件下，测试各类算法的CLER。各类算法的CLER变化情况如图6所示。

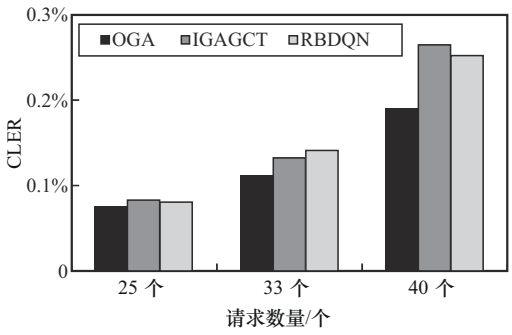

图6 各类算法的CLER变化情况

由图6可知，请求数量为25个时，OGA的CLER接近7.5%，IGAGCT算法的CLER接近8.15%，RBDQN算法的CLER接近8%，OGA的CLER分别降低了7.9%和6.25%；请求数量为33个时，OGA的CLER接近11.3%，IGAGCT算法的CLER接近13.2%，RBDQN算法的CLER接近14%，OGA的CLER分别降低了15.1%和19.2%；请求数量为40个时，OGA的CLER接近18.9%，IGAGCT算法的CLER接近26.3%，RBDQN算法的CLER接近25.2%，OGA的CLER分别降低了28.1%和25%。

通过对HRSR、ACDR和CLER的测试可知，IGAGCT算法和RBDQN算法在解决HCFRR问题时算法性能相对较劣，原因是：（1）前者是一种路径相交的多对多路径选择机制，容易产生路径抢夺与竞争，从而降低传输的成功率，增加路径选择时延以及提高网络负载；（2）后者是一种调度与路由联合的路径选择机制，其路由选择采用贪婪算法，该算法本质上依然存在路径上的竞争和冲突，从而无法保障传输成功率、收敛速度以及网络负载。OGA则优化了以上算法的不足，因此更能满足请求迁移和传输的诉求，相对更加有效。

4 结束语

算力网络是数字经济催生的变革式技术，本

文以此为背景，讨论了算力网络中基础性及关键性问题——异构算力请求的路由问题，从优化论角度对该NP完全问题分析与建模，并提出一种优化型遗传算法予以解决。最后，本文通过实验测试了该算法的合理性和有效性。未来，将重点研究支持成百上千个异构算力请求的路径选择问题，用于满足更多的请求迁移和传输的需求。

参考文献：

[1] 中国移动通信集团终端有限公司, 北京邮电大学, 中国信息通信研究院, 等. 端侧算力网络白皮书(2022年)[R]. 2022.
China Mobile Terminal Co., Ltd., Beijing University of Posts and Telecommunications, China Academy of Information and Communications Technology, et al. Edge computing force network whitepaper[R]. 2022.

[2] FU Y X, WANG J, LU L, et al. Reputation-based joint optimization of user satisfaction and resource utilization in a computing force network[J]. Frontiers of Information Technology & Electronic Engineering, 2024, 25(5): 685-700.

[3] 何涛, 杨振东, 曹畅, 等. 算力网络发展中的若干关键技术问题分析[J]. 电信科学, 2022, 38(6): 62-70.
HE T, YANG Z D, CAO C, et al. Analysis of some key technical problems in the development of computing power network[J]. Telecommunications Science, 2022, 38(6): 62-70.

[4] 贾庆民, 胡玉姣, 张华宇, 等. 确定性算力网络研究[J]. 通信学报, 2022, 43(10): 55-64.
JIA Q M, HU Y J, ZHANG H Y, et al. Research on deterministic computing power network[J]. Journal on Communications, 2022, 43(10): 55-64.

[5] XIAO Y P, HE X, YANG C, et al. Dynamic graph computing: a method of finding companion vehicles from traffic streaming data[J]. Information Sciences, 2022, 591: 128-141.

[6] LIU Z H, LENG J W, ZHANG Z H, et al. VELTAIR: towards high-performance multi-tenant deep learning services *via* adaptive compilation and scheduling[C]//Proceedings of the 27th ACM International Conference on Architectural Support for Programming Languages and Operating Systems. New York: ACM Press, 2022: 388–401.

[7] KUO P C, CHEN Y W, HSU Y C, et al. High performance post-quantum key exchange on FPGAs[J]. Journal of Information Science and Engineering, 2021(38): 1211-1229.

[8] YANG M Y, QIAN Y, PU T L, et al. Edims: an event-driven internal memory synchronized readout prototype ASIC chip developed for HFRS-TPC[J]. Nuclear Science and Techniques, 2023, 34(12): 196.

[9] WEI L, LI J Y, GAO Y F, et al. Resource matching algorithm based on multidimensional computing resource measurement in computing power network[C]//Proceedings of the 2024 27th International Conference on Computer Supported Cooperative Work in Design (CSCWD). Piscataway: IEEE Press, 2024: 2295-2300.

[10] 柴若楠, 郜帅, 兰江雨, 等. 算力网络中高效算力资源度量方法[J]. 计算机研究与发展, 2023, 60(4): 763-771.
CHAI R N, GAO S, LAN J Y, et al. Efficient computing resource metric method in computing-first network[J]. Journal of Computer Research and Development, 2023, 60(4): 763-771.

[11] HAN X Y, XIE M X, YU K, et al. Combining graph neural network with deep reinforcement learning for resource allocation in computing force networks[J]. Frontiers of Information Technology & Electronic Engineering, 2024, 25(5): 701-712.

[12] 陈星延, 张雪松, 谢志龙, 等. 面向“云-边-端”算力系统的计算和传输联合优化方法[J]. 计算机研究与发展, 2023, 60(4): 719-734.
CHEN X Y, ZHANG X S, XIE Z L, et al. A computing and transmission integrated optimization method for cloud-edge-end computing first system[J]. Journal of Computer Research and Development, 2023, 60(4): 719-734.

[13] FANG W. Optimization algorithm for computing power resource scheduling based on container cluster deployment[J]. Journal of Network Intelligence, 2024, 9(2): 835-849.

[14] 李逸博, 李小平, 王爽, 等. 面向算力网络的智慧调度综述[J]. 自动化学报, 2024, 50(6): 1086-1103.
LI Y B, LI X P, WANG S, et al. Survey on wise scheduling in computing power network[J]. Acta Automatica Sinica, 2024, 50 (6): 1086-1103.

[15] XIE M X, YU K, WU X F. Adaptive routing of task in computing force network by integrating graph convolutional network and deep Q-network[C]//Proceedings of the 2023 8th IEEE International Conference on Network Intelligence and Digital Content (IC-NIDC). Piscataway: IEEE Press, 2023: 242-247.

[16] LI X N, LU J G, ZHANG P M. A GNN-based routing and scheduling mechanism for multi-domain computing first network[C]//Proceedings of the 2023 IEEE 9th International Conference on Cloud Computing and Intelligent Systems (CCIS). Piscataway: IEEE Press, 2023: 153-163.

[17] 孙国玮, 许方敏, 朱瑾瑜, 等. 算力网络中的确定性调度与路由联合智能优化方案[J]. 北京邮电大学学报, 2023, 46(2): 9-14.

SUN G W, XU F M, ZHU J Y, et al. Deterministic scheduling and routing joint intelligent optimization scheme in computing first network[J]. Journal of Beijing University of Posts and Telecommunications, 2023, 46(2): 9-14.

[18] 李洪钧, 任保全, 巩向武, 等. 天地一体化网络分散式算力路由控制策略[J]. 指挥与控制学报, 2022, 8(4): 451-459.
LI H J, REN B Q, GONG X W, et al. Dispersed computing-dependent routing control strategies for space-ground integration networks[J]. Journal of Command and Control, 2022, 8(4): 451-459.

[19] YAO H J, DUAN X D, FU Y X. A computing-aware routing protocol for Computing Force Network[C]//Proceedings of the 2022 International Conference on Service Science (ICSS). Piscataway: IEEE Press, 2022: 137-141.

[20] 颜晓莲, 邱晓红. 支持边缘端: 云端协同工作的群组命令传输算法[J]. 计算机应用研究, 2021, 38(4): 1154-1157.
YAN X L, QIU X H. Group command transfer algorithm for edge-cloud cooperative work[J]. Application Research of Computers, 2021, 38(4): 1154-1157.

[21] 颜晓莲, 章刚, 邱晓红. 一种支撑协同制造的动态群组命令传输算法[J]. 计算机应用研究, 2020, 37(8): 2362-2365, 2394.
YAN X L, ZHANG G, QIU X H. Dynamic group command transfer algorithm for collaborative manufacturing[J]. Application Research of Computers, 2020, 37(8): 2362-2365, 2394.

[22] XU X W, DING H, WANG J Y, et al. Compute first networking scheduling strategy based on genetic algorithm in edge computing environment[M]//Communications in Computer and Information Science. Singapore: Springer Nature Singapore, 2024: 197-205.

[23] LIU Y, JIANG J B, LIU Y, et al. Network resource optimization configuration in edge computing environment[J]. International Journal of Computers and Applications, 2023, 45(1): 88-95.

[24] CREEVEY F M, HILL C D, HOLLENBERG L C L. GASP: a genetic algorithm for state preparation on quantum computers[J]. Scientific Reports, 2023, 13(1): 11956.

[25] WAXMAN B M. Performance evaluation of multipoint routing algorithms[C]//Proceedings of the IEEE INFOCOM '93 The Conference on Computer Communications, Proceedings. Piscataway: IEEE Press, 1993: 980-986.

[26] O'SULLIVAN M, ANIELLO L, SASSONE V. A methodology to select topology generators for ad hoc mesh network simulations[J]. Journal of Communications, 2020(15): 741-746.

[作者简介]

章刚（1981–），男，博士，南昌工学院副教授，主要研究方向为算法设计、优化理论。

黎曦（1982–），男，博士，南昌工学院副教授，主要研究方向为VR、AR、算法设计等。