无等待与时隙映射复用结合的时间触发流调度方法

何倩[1]，郭雅楠[1,2]，赵宝康[3]，潘琪[1,2]，王勇[2]

（1.卫星导航定位与位置服务国家地方联合工程研究中心（桂林电子科技大学），广西 桂林 541004；
2.桂林电子科技大学广西可信软件重点实验室，广西 桂林 541004；3.国防科技大学计算机学院，湖南 长沙 410073）

摘　要：工业控制系统中众多应用的正常运行依赖于确定性低时延的网络传输，这一需求推动了时间敏感网络的发展。为保证工业控制系统中流量传输的确定性，提出了无等待与时隙映射复用结合的时间触发流调度方法。首先，对工业控制系统时间敏感网络的通信要素进行建模，通过分析时间触发流的特性，简化了链路、流传输、流隔离和帧隔离等约束条件；其次，使用基础周期作为调度循环时间，实时流采取无等待调度，循环流采取时隙映射复用，从而缩短了门控列表长度；然后，提出了基于数据帧传输区间中点的冲突判别方法，降低了时隙冲突判断的时间复杂度；最后，设计了时间触发流调度优化函数，并基于改进的多目标遗传算法进行求解。实验验证了该方法的正确性与可行性，平均门控列表长度缩短了 59.7%。

关键词：工业互联网；时间敏感网络；时隙映射复用；多目标优化

中图分类号：TN393

文献标志码：A

DOI：10.11959/j.issn.1000–436x.2024134

Time-triggered stream scheduling method combining no-wait and time-slot mapping reuse

HE Qian[1], GUO Yanan[1,2], ZHAO Baokang[3], PAN Qi[1,2], WANG Yong[2]

1. Satellite Navigation Positioning and Location Service National & Local Joint Engineering Research Center, Guilin University of Electronic Technology, Guilin 541004, China
2. Guangxi Key Laboratory of Trust Software, Guilin University of Electronic Technology, Guilin 541004, China
3. College of Computer, National University of Defense Technology, Changsha 410073, China

Abstract: The proper functioning of numerous applications in industrial control systems relies on deterministic and low-latency network transmission, driving the development of time-sensitive networking. To guarantee the deterministic stream transmission in industrial control systems, a time-triggered stream scheduling method combining no-wait and time-slot mapping reuse (NW-TSMR) was proposed. Firstly, the time-sensitive network communication elements of industrial control systems were modeled, and by analyzing the characteristics of time-triggered streams, several constraints such as link, stream transmission, stream isolation, and frame isolation were simplified. The base period was used as the scheduling cycle time, by adopting no-wait scheduling for real-time streams and time-slot mapping reuse for cyclic streams, the length of gate control list was reduced. Then, a conflict detection method that utilized the midpoint of the data frame transmission interval significantly reduced the time complexity associated with slot conflict detection. Finally, several time-triggered stream scheduling optimization functions were given and solved based on an improved multi-objective genetic algorithm. The correctness and feasibility of NW-TSMR is verified through experiments, and the average gate control list length is reduced by 59.7%.

Keywords: industrial Internet, time-sensitive networking, time-slot mapping reuse, multi-objective optimization

---

收稿日期：2024–01–11；**修回日期：**2024–06–19

通信作者：何倩，heqian@guet.edu.cn

基金项目：国家自然科学基金资助项目（No.62162018, No.U22B2005）；广西自然科学基金资助项目（No.2023JJD170008）

Foundation Items: The National Natural Science Foundation of China (No. 62162018, No. U22B2005), Guangxi Natural Science Foundation (No.2023JJD170008)

0　引言

随着工业4.0和工业以太网的发展，工业控制系统中的各种工控软件、可编程控制器、传感器、执行器、机械臂和机床等部件通过工业互联网[1]连接到一起。越来越多的工业控制类应用具有确定性通信需求，这些应用对时延和丢包率的要求较低。例如，工厂中的机械臂需要在给定的周期（1~10 ms）内接收控制指令并报告其工作状态，时延过大或丢包都可能导致机械臂无法完成预定的动作[2]。为实现确定性低时延传输，时间敏感网络（TSN, time-sensitive networking）[3]扩展了标准以太网，使其更适用于具有确定性通信需求的应用场景。TSN由一系列基于IEEE 802.1Q以太网的扩展协议构成，包括时钟同步（IEEE 802.1 AS）、增强流量调度（IEEE 802.1 Qbv）、帧抢占（IEEE 802.1 Qbu、IEEE 802.3br）、路径控制（IEEE 802.1 Qca）资源预留（IEEE 802.1 Qcc）和冗余管理（IEEE 802.1 CB）等，这些机制共同保证了以太网通信的实时性和确定性[4]。

在一系列协议中，增强流量调度协议（IEEE 802.1 Qbv）[5]主要解决了以太网通信中不确定性时延的问题。该协议通过定义时间感知整形器实现微秒级逐跳逐包的细粒度调度[6]，并以门控列表（GCL, gate control list）为输入，控制网口队列的开关状态，从而实现时间触发（TT, time triggered）流的确定性传输。然而，该协议并未给出GCL的生成方法。TSN的调度是一个多目标组合优化问题，已经被证明是NP问题[7]，基于调度结果生成的调度表可以用于构建GCL。合理的GCL可以使TT流的数据帧在传输过程中尽可能缩短甚至避免在各个TSN交换机网口缓存队列中的排队时延，GCL的长度越短，所需的交换机硬件成本越低。

工业互联网联盟发布了一份白皮书[8]，基于IEEE 802.1 Q标准分析了工业控制与自动化系统中不同流量类型的特性，如表1所示，其中，n.a.表示不适用。这些特性覆盖了工业互联网中80%的应用场景，其中，实时同步流量和循环流量具有严格的时间限制，通常被称为调度流量或TT流，是调度问题关注的重点。然而，现有的调度方法没有完全考虑截止时间、时延及流量类型的区分，更多的是考虑通用的调度机制，而非针对工业互联网特定场景的需求。

本文针对工业互联网特定场景下GCL的生成问题，结合TT流的特点[9]，提出了无等待与时隙映射复用结合的时间触发流调度方法（NW-TSMR, no-wait and time-slot mapping reuse），主要贡献如下。

1) 对工业控制系统的TSN网络通信要素进行建模，调度循环时间使用基础周期（BP, base period）而非超周期，实时流采用无等待调度，循环流采用时隙映射复用（TSMR, time-slot mapping reuse），解决了周期比（PR, period ratio）过大导致GCL过长的问题，保证了TT流的传输确定性。

2) 针对TT流约束条件判断过程中时间复杂度高的问题，提出了一种基于数据帧传输区间中点的冲突判别方法，用于判断2条TT流的所有时隙是否兼容。该方法通过模运算简化运算过程，降低了时隙冲突判别的时间复杂度。

3) 设计了多个TT流调度优化目标函数，从多个维度对求解结果进行优化，并采用改进的多目标遗传算法求解调度问题，实验验证了NW-TSMR方法的正确性和有效性。

表1　流量类型

流量类型	周期性	典型周期	数据传输保证	抖动容限	数据分组丢失容限	典型数据大小/B	优先级
实时同步	√	100 μs~2 ms	截止时间	0	0	固定30~100	6
循环	√	2~20 ms	时延	≤时延	1~4帧	固定50~1 000	5
控制事件	×	n.a.	时延	≤时延	可容忍	可变100~200	4
报警和操作命令事件	×	n.a.	时延	≤时延	可容忍	可变100~1 500	3
网络控制	√	50 ms~1 s	带宽	可容忍	可容忍	可变50~500	7
配置诊断	×	n.a.	带宽	n.a.	可容忍	可变500~1 500	2
视频	√	帧率	无	n.a.	可容忍	可变1 000~1 500	1
音频/语音	√	采样率	时延	n.a.	可容忍	可变1 000~1 500	1
尽力而为	×	n.a.	时延	n.a.	可容忍	可变30~1 500	0

1 相关工作

目前，TSN 调度问题的研究主要集中在构建通用的 TT 流调度机制。文献[10]分析了以太网中影响实时通信的关键因素，提出了在多跳交换局域网中应用 TAS 算法的约束条件，包括帧约束、链路约束、流传输约束、端到端时延约束、流隔离约束和帧隔离约束，并利用可满足模理论和优化模理论求解器进行求解。文献[11]分析了工厂网络常见的层级结构，将网络划分为不同的子网，并基于文献[8]对工厂网络流量类型进行了区分，主要调度实时同步流量，每个子网应用无等待（NW, no-wait）调度算法[12]生成调度，而跨越不同层的流采取批量调度算法。虽然混合不同的机制提高了可调度性，但对于层之间边缘交换机的缓存要求较高。文献[13]将 TSN 调度问题抽象为作业调度问题，提出了一种混合遗传算法，尝试搜索流发送顺序和路由选择的最佳组合，但求解结果无法直接使用，只是生成了流的发送顺序，需要进一步转化才能得到 GCL。

针对工业互联网的特定调度场景，文献[14]基于基础循环周期（BPC, base period cycle）提出了适用于工业互联网的 TSN 调度方案，并利用基于超周期（HPC, hyper period cycle）的调度方案作为对比，验证了 BPC 方案可以缩短 GCL 长度，但调度粒度较大导致带宽浪费较多。文献[15]提出了将感知整形器与循环队列转发相结合的混合调度机制，与文献[8]中假定 TT 流周期固定不同，文献[15]中假定 TT 流的周期是可变的，并制定了周期调整策略以降低 TT 流对网络带宽的占用率，实验选取了 400~1 000 B 作为 TT 流长度的选择范围，调度结果具有较好的易用性，但每条流实际占用带宽相比于文献[8]高出一个数量级。文献[16]基于谱聚类的分组策略提出一种分组调度机制，加快了求解速度。文献[17]对基于 HPC 的启发式调度算法进行了改进，提出一种数据周期感知技术和输出缓存感知技术，以解决在 TT 流 PR 过大时 GCL 长度过长的问题，但该方案仅适用于规模较小的网络。

上述研究表明，混合调度算法在处理具有不同数据传输保证的混合流量时具有更好的适应性。当不同数据流的发送周期在数量级上有较大差异时，动态调整调度周期而不是直接使用超周期，显著缩短了调度表长度。此外，通过对数据流进行分组，可以缩小问题的求解规模，无需对所有数据流进行整体考虑，从而有效缩短求解时间。

2 网络通信模型

本文将工业控制系统的 TSN 网络抽象为有向图 $G(V,E)$，其中 V = SW ∪ ES 是节点的集合，包含交换机 SW 的集合和端系统 ES 的集合；E 表示节点之间全双工链路的集合，连接 2 个节点的逻辑链路用有向边表示，简化后的 TSN 通信模型如图 1 所示。图 1 为包含 2 个端系统和一个 TSN 交换机的直线形拓扑，TT 流（灰色实线）stream_0（stream_1）分别以 es_0 和 es_1 为源节点（目的节点）和目的节点（源节点），连接 es_0 和 sw_0 以及 es_1 和 sw_0 的物理链路（黑色实线），各自包含 2 条方向相反的链路 link_0~link_3，TT 流的数据帧最终通过每个端系统的网卡发送。

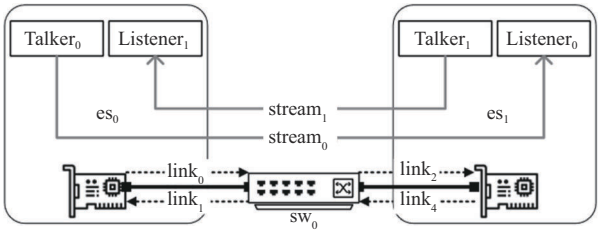


图 1 工业控制系统 TSN 网络通信示意

依据表 1 对流量进行划分，主要对实时同步流（优先级为 6）和循环流（优先级为 5）进行调度。实时同步流主要用于实时控制、时间同步等，需要截止时间（DDL, deadline）保证。DDL 通常在一个发送周期内，定义实时同步流的 DDL 与流的周期相等[8]。循环流需要时延保证，时延通常不超过周期的 10%[8]，且抖动应不大于时延。

TSN 网络中通信各个要素的模型包括端节点网口、TSN 交换机、端系统、链路、路由和 TT 流。符号含义如表 2 所示。本节主要对这些要素进行建模。

2.1 TSN 网口

TSN 交换机一般包含多个网口，假设端系统只有一个网口，GCL 的生成以网口为单位。< t,GL > 表示网口，其中 t 和 GL 分别为网口连接网线后的发送速率和网口 GCL。例如，采用 6 类网线连接的网口，发送速率为 1 bit/ns。GL 结构与 IEEE Std 802.1Q 相同，包含一个长度为 8 的布尔类型数组和

一个整数，分别表示门控状态和当前状态的持续时长。

表2　符号含义

符号	含义
S	TT流集合
R_i	流s_i的路由集合，通过计算图G中节点s到节点d的所有有向无环路径得到R_i
$R_{i,m}$	流s_i的第m条路由
$R^n_{i,m}$	流s_i的第m条路由的第n段链路
$T^{[p_a,p_b]}$	链路$[p_a,p_b]$的所有实时同步流的发送周期的最小公倍数，$[p_a,p_b]=R^n_{i,m}T^{[p_a,p_b]}$等价于$T^n_{i,m}$
m_i	流s_i源节点网口的发送速率
$d_q^{[p_a,p_b]}$	链路$[p_a,p_b]$源节点的排队时延
$d_{\mathrm{pr}}^{[p_a,p_b]}$	链路$[p_a,p_b]$源节点的处理时延
$s_{i,j}$	流s_i的第j个属性，$s_{i,0}$~$s_{i,6}$表示流s_i每帧的发送时间、周期、帧长、优先级、源节点、目的节点和路由集合
$s_{i,\alpha}^{[p_a,p_b].\phi}$	流s_i第α个数据帧在链路$[p_a,p_b]$源网口中相对于$T^{[p_a,p_b]}$的发送时间偏移
Φ	发送时间基因的数组
C	路由选择基因的数组
D	TT流分组集合
D.len	TT流分组的数量
GP	组周期的集合，每个分组的周期用GP_k表示
LS	链路和TT流的映射关系
LTS	链路与时隙的对应关系

2.2　TSN交换机与端系统

TSN交换机sw ⊂ SW用$<P,d_{\mathrm{pr}}>$表示，其中P为TSN网口的集合，d_{pr}为交换机处理时延。端系统es ⊂ ES用$<p,d_{\mathrm{pr}}>$表示，p为端系统的TSN网口，d_{pr}为端系统处理时延。由于仿真平台NeST-iNg[18]只支持对TSN交换机定义处理时延，相关内容可参考文献[19]，假设所有TSN交换机的处理时延为20 μs。实验中端系统的处理时延设置为0，但是端系统模型可以方便地扩展为处理时延不为0的情况。

2.3　链路和路由

链路$l \in E$用$<d_{\mathrm{prop}}^{[p_a,p_b]},[p_a,p_b].s>$表示，其中$d_{\mathrm{prop}}^{[p_a,p_b]}$为链路传播时延，$[p_a,p_b].s$为链路速度。

路由$R=\left[[p_s,p_{1'}],[p_1,p_{2'}],\cdots,[p_{n-1},p_{n'}],[p_n,p_d]\right]$是包含一系列链路的有序集合，依次表示源节点网口与第一跳输入网口的链路，第一跳输出网口与第二跳输入网口的链路，直至最后一跳输出网口与目的节点网口的链路。其中，$p_{i'}$和p_i表示同一交换机的2个不同的网口。

2.4　数据流

每条TT流s_i包含属性集$A=\{\phi,t,l,p,s,d\}$，其中，ϕ为第一帧相对于源节点网口基础周期$T^0_{i,m}$的发送时间（即注入时间[6]）；t为流s_i的发送周期；l为流s_i每帧的长度；p为流s_i的优先级；s和d分别为流s_i的源节点和目的节点。

3　NW-TSMR调度

NW-TSMR主要利用无等待调度和TSMR机制构建调度方法，其中无等待调度[12]满足高优先级TT流对实时性和确定性的需求，因此实时同步流采取无等待调度策略。循环流允许一定的交叉、丢包和抖动，且周期通常比实时同步流大，因此在生成实时同步流的调度方案后，将循环流的数据帧填充到无等待调度产生的间隙中。

为了缩小求解问题的规模，本文对TT流进行了分组[16]，求解过程针对每个分组而不是所有TT流。在一个分组的组周期内，GCL将被重复执行多次，每次执行的时长为一个BP。

流调度机制通常假设数据帧可能超过以太网最大传输单元（MTU, maximum transmission unit），因此一个数据帧会被拆成多个分片。为了保证每个分片的正确传输，需要帧约束、DDL约束、时延约束、无冲突约束、链路约束、流传输约束、流隔离约束以及帧隔离约束等。在此前提下，对于一条TT流，其调度结果包含每个分片在每跳的发出时间，这些发出时间是在解空间内随机生成的。为了确保所有发送时间彼此之间的大小关系，提出了链路约束和流传输约束；为了确保不同TT流的分片在交换机的流量队列内不交叉，提出了隔离约束。根据文献[8]，工业互联网中的实时同步流和循环流的数据帧长度均小于MTU，因此本文方案仅生成每个数据帧在第一跳的发出时间，后续每条链路上的发出时间通过递归方程计算得出，发送时序是正确的，无需通过流传输约束和链路约束进行保证，数据帧在队列中也不存在交叉行为，无需隔离约束。为了确保调度时间的准确性，NW-TSMR保留了通用调度机制中必要的约束条件（帧约束、DDL约束、时延约束以及无冲突约束），精简了链路约束、流传输约束、流隔离约束以及帧隔离约束。

3.1 TT流分组策略

关于TT流分组的性质，具体描述如下。在一个分组中，任意一条TT流的路由与分组内至少一条TT流的路由存在重合部分。将具有这种性质的一组TT流构成一个TT流分组。例如，图1中所有从es_0到es_1的TT流为一组，所有从es_1到es_0的TT流为另一组。这2组流的传输使用了2条逻辑上无重合部分的路由，因此它们之间不会产生冲突，而组内则存在潜在的冲突。基于TT流分组的性质，采用广度优先搜索方法对TT流进行分组，具体步骤如算法1所示。

算法1 TT流分组算法

输入 流的集合S

输出 分组D

1) while $s_i \in S$
2) while $R_{i,m} \in R$
3) 保存相同链路的TT流到LS中
4) end while
5) end while
6) 初始化所有TT流访问状态v_i为false
7) while $s_i \in S$
8) if s_i未被访问
9) 创建数组g以保存分组内的流
10) 初始化访问队列Q为∅
11) 更新s_i的访问状态v_i为true
12) 将s_i添加到g中
13) 将s_i加入访问队列Q中
14) while $Q \neq \varnothing$
15) 将队尾元素s_j出队
16) while $R^r_{j,m} \in R_{j,m}$
17) while $s_k \in \mathrm{LS}[R^r_{j,m}]$
18) if s_k未被访问
19) 将s_k保存到g中
20) 将s_k加入访问队列Q中
21) 更新v_k为true
22) end if
23) end while
24) end while
25) end while
26) if Q为空
27) 将分组g保存到D中
28) end if
29) end if
30) end while
31) return D

3.2 BP

基于HPC的调度算法，以超周期为调度循环时间会导致GCL过长。文献[14]采取了最大公因数的方式定义BP，每个BP包含若干个长度相同的时隙，且每个BP内时隙的门控状态相同，基于BP配置GCL。为了实现细粒度的控制，本文中的时隙长度与对应数据帧通过网卡的传输时长保持一致。每个网口的BP定义为通过当前网口的所有实时同步流周期的最小公倍数，如果通过该网口的只有循环流，则BP等于所有循环流周期的最小值。

3.3 帧约束

帧约束规定了最终求解的TT流s_i初始发送时间的取值范围，仅对流第一帧的发送时间进行设定，后续每帧的发送时间加上s_i周期属性的相应倍数即可，约束条件为$\forall s_i \in S: s_{i,0} \in \left[0, s_{i,1} - s_{i,2}m_i\right)$。其中，$s_{i,1} - s_{i,2}m_i$为$s_{i,0}$可取的最大值，$s_{i,2}m_i$为流$s_i$在源节点网口的发送时延。

对于实时同步流s_i，仅计算源节点的BP内s_i第一帧经过每个TSN交换机时的发送/转发时间，利用递归方程的形式描述流s_i在进入任意TSN交换机之后的发送时间。$\forall s_i \in S, [p_x, p_b] = R^r_{i,m}, \forall \alpha \in \left[0, \dfrac{T^r_{i,m}}{s_{i,1}} - 1\right]$，则定义为

$$s^{[p_x,p_b]}_{i,\alpha}.\phi = \begin{cases} d^{[p_x,p_b]}_{\mathrm{pr}} + s_{i,0} + \alpha s_{i,1}, & r = 0 \\ s^{[p_a,p_{x'}]}_{i,\alpha}.\phi + s_{i,2}p_a.t + d^{[p_a,p_{x'}]}_{\mathrm{prop}} + d^{[p_x,p_b]}_{\mathrm{pr}}, & r > 0 \text{且} \alpha = 0 \\ s^{[p_x,p_b]}_{i,0}.\phi + \alpha s_{i,1}, & r > 0 \text{且} \alpha > 0 \end{cases} \quad (1)$$

其中，当$r = 0$时，数据帧在路由源节点；当$r > 0$且$\alpha = 0$时，通过前一跳的发送时间加上发送时延、传播时延和当前节点的处理时延得到当前节点的发送时间；当$r > 0$且$\alpha > 0$时，发送时间等于当前节点第一次转发的时间与对应周期的倍数之和。

对于循环流s_i，计算GP_k在每个路由节点的输出网口所有的发送/转发时间，主要过程和同步流的计算方式类似。$\forall s_i \in S, [p_x, p_b] = R^r_{i,m}$，$\forall \alpha \in \left[0, \dfrac{\mathrm{GP}_k}{s_{i,1}} - 1\right]$，利用递归方程可定义为

$$t=\begin{cases} d_{\mathrm{pr}}^{[p_x,p_b]}+s_{i,0}+\alpha s_{i,1}+d_q^{[p_x,p_b]}, & r=0 \\ s_{i,\alpha}^{[p_a,p_{x'}]}.\phi+s_{i,2}p_a.t+d_{\mathrm{prop}}^{[p_a,p_{x'}]}+ & \\ d_{\mathrm{pr}}^{[p_x,p_b]}+d_q^{[p_x,p_b]}, & r>0 \end{cases} \tag{2}$$

其中，$s_{i,\alpha}^{[p_x,p_b]}.\phi=t$。

3.4　DDL 约束和端到端时延约束

针对任意一条 TT 流 s_i，在选定路由 $R_{i,m}$ 后，发送时延、传播时延和处理时延都可以确定。实时同步流不允许缓存，因此排队时延为 0，而循环流允许缓存，进一步得出端到端时延的计算方法，$\forall s_i\in S, s_{i,3}\in[5,6]$，则有

$$\mathrm{delay}=\sum_{[p_a,p_b]\in R_{i,m}}\left(d_{\mathrm{pr}}^{[p_a,p_b]}+d_q^{[p_a,p_b]}+s_{i,2}p_a.t+d_{\mathrm{prop}}^{[p_a,p_b]}\right) \tag{3}$$

其中，$s_{i,2}p_a.t$ 表示流 s_i 的数据帧在链路 $[p_a,p_b]$ 源网口的发送时延。

对于实时同步流 s_i，数据帧必须在 DDL 之前到达目的节点，这意味着前一个数据帧必须在当前数据帧发送前到达目的节点[8]，DDL 约束条件为

$$\forall s_{i,3}=6: s_{i,0}+\mathrm{delay}\leqslant s_{i,1} \tag{4}$$

本节仅对 s_i 的第一帧进行约束，主要原因是实时同步流的数据帧不会在任意节点缓存，并且严格按照周期间隔发送。在 BP 内满足无等待调度的前提下，如果第一帧满足 DDL 约束，BP 内的后续每帧也会满足此约束。

对于循环流 s_i，数据帧需要满足端到端时延约束，在源节点发出后，需要在时延保证时间内到达目的节点，定义时延保证时间为周期的 10%[8]，约束条件为

$$\forall s_{i,3}=5:\mathrm{delay}\leqslant 0.1s_{i,1} \tag{5}$$

3.5　无冲突约束

无冲突约束规定实时同步流在各个网口的传输时段不能重叠。文献[4,7,10]中称无冲突约束为链路约束，本质上都是描述任意 2 个数据帧在同一个网口的传输时段不能重叠。因此，对于任意 2 条共享链路 $[p_x,p_b]$ 的实时同步流，流 s_j 在前一条链路 $[p_a,p_{x'}]$ 的发送完成时间不能大于流 s_i 在 $[p_x,p_b]$ 的发送开始时间，或者流 s_j 在 $[p_x,p_b]$ 的发送开始时间不能大于流 s_i 在 $[p_x,p_b]$ 的发送完成时间，这两者是等价的。

本文采用另一种等价的描述，定义流 s_i 在链路 $[p_a,p_b]$ 上某一帧传输时段的中点，即数据帧的发送开始时间加上发送时延的一半，流 s_j 在 $[p_a,p_b]$ 上某一帧传输时段的中点为 $s_{j,\beta}^{[p_a,p_b]}.\omega$，表示为

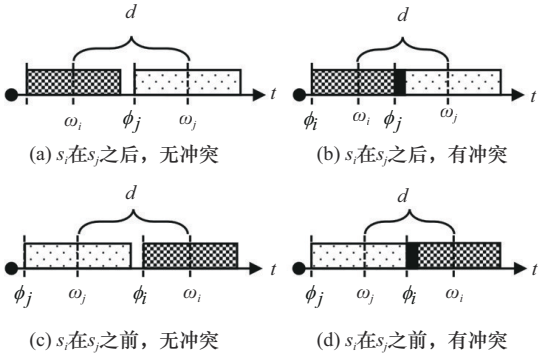
$$s_{i,\alpha}^{[p_a,p_b]}.\omega=s_{i,\alpha}^{[p_a,p_b]}.\phi+\frac{s_{i,2}p_a.t}{2} \tag{6}$$

$s_{i,\alpha}^{[p_a,p_b]}.\omega$ 与 $s_{j,\beta}^{[p_a,p_b]}.\omega$ 之间差值的绝对值为距离 d，如果满足不等式 $d\geqslant\frac{s_{i,2}p_a.t+s_{j,2}p_a.t}{2}$，即当 d 不小于 2 条流各自发送时延一半的和时，2 个数据帧的发送时段不会重叠。本文利用距离判定是否会发生冲突，假设 $s_{i,1}>s_{j,1}$，$\forall s_i,s_j\in S$，$i\neq j$，$s_{i,3}=6$，$s_{j,3}=6$，$s_{i,1}>s_{j,1}$，$\forall\alpha\in\left[0,\frac{\mathrm{lcm}(i,j)}{s_{i,1}}-1\right]$，则约束条件为

$$d=\left|s_{i,\alpha}^{[p_a,p_b]}.\omega \bmod s_{j,1}-s_{j,0}^{[p_a,p_b]}.\omega \bmod s_{j,1}\right| \tag{7}$$

其中，$d\geqslant\frac{s_{i,2}p_a.t+s_{j,2}p_a.t}{2}$，$\mathrm{lcm}(i,j)$ 表示 s_i 与 s_j 周期的最小公倍数，$s_{i,\alpha}^{[p_a,p_b]}.\omega \bmod s_{j,1}$ 将 $\mathrm{lcm}(i,j)$ 内流 s_i 在 $[p_a,p_b]$ 上第 α 帧传输时段的中点映射到流 s_j 的周期内，$s_{j,0}^{[p_a,p_b]}.\omega \bmod s_{j,1}$ 将流 s_j 在 $[p_a,p_b]$ 上的第一次发送时间的中点映射到流 s_j 的周期内，通过比较两者差值的绝对值即可检测是否会发生冲突。

假设流 s_i 和 s_j 的发送时间和传输时段的中点经过模 $s_{j,1}$ 运算后分别得到 ϕ_i 和 ω_i 以及 ϕ_j 和 ω_j，经过模 $s_{j,1}$ 运算后，流 s_i 的数据帧可能在流 s_j 的数据帧之后，如图 2(a)和图 2(b)所示；也可能在流的数据帧之前，如图 2(c)和图 2(d)所示。黑色部分表示发送时间有冲突。



(a) s_i在s_j之后，无冲突　(b) s_i在s_j之后，有冲突

(c) s_i在s_j之前，无冲突　(d) s_i在s_j之前，有冲突

图 2　冲突检测示意

$\forall\beta\in\left[0,\frac{\mathrm{lcm}(i,j)}{s_{j,1}}-1\right]$，$s_j$ 为实时流，源节点发送相邻 2 个数据帧的时间间隔等于其周期，因此有 $s_{j,0}^{[p_a,p_b]}.\omega\equiv s_{j,\beta}^{[p_a,p_b]}.\omega\,(\bmod s_{j,1})$。如果不利用模运算，

任意 2 条 TT 流最多需要比较 mn 次才能确定在 $\mathrm{lcm}(i,j)$ 内是否会发生冲突，时间复杂度为 $O(mn)$，其中 $m=\frac{\mathrm{lcm}(i,j)}{s_{i,1}}$，$n=\frac{\mathrm{lcm}(i,j)}{s_{j,1}}$。而利用模运算处理后，根据中点的定义，将周期小的流作为被比较的对象，最多需要比较 $\min\{m,n\}$ 次，即周期大的流在 $\mathrm{lcm}(i,j)$ 内的发送/转发次数，时间复杂度为 $O(\min\{m,n\})$。

3.6 时隙映射复用

基于 HPC 的调度算法在传输时隙之间存在大量未被利用的带宽，而 TT 流周期在数量级上的差异又导致了 GCL 长度的爆发式增长。在 BP 内，实时同步流利用 NW 策略进行调度，实际的控制信号通常很短小，只有几十字节[2]，有些消息甚至是仅指示开关状态的简单信号[20]，因此 BP 内存在大量可用带宽资源。将循环流的时隙利用模 BP 的运算映射到 BP 内，可抑制 GCL 长度的增长。每个网口 GCL 循环 $\frac{\mathrm{GP}_k}{T^{[p_a,p_b]}}$ 次相当于基于 HPC 的算法调度一个超周期。

为了保存调度结果，定义了如图 3 所示的 2 层嵌套型键值对存储结构 LTS，key_2 和 value_2 构成了内层的键值对，分别表示一个时隙及用到该时隙的 TT 流信息的集合，若干个内层键值对构成的集合作为外层的值（即 value_1），key_1 和 value_1 构成了外层的键值对。为了维持时隙的时序，避免频繁排序，value_1 和 value_2 均为有序集合。

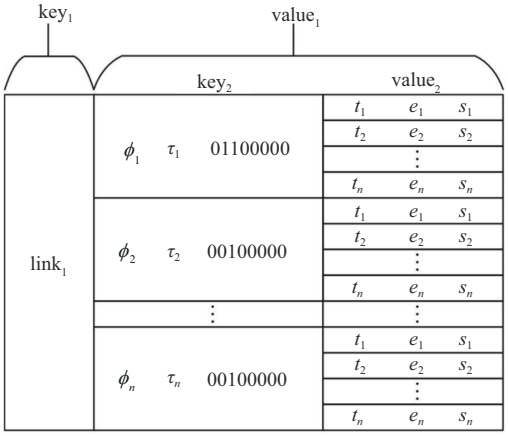


图 3 LTS 存储结构示意

内层键值对中的每个键（key_2）包含 3 个元素，以 $\{\phi_1,\tau_1,01100000\}$ 为例，这 3 个元素分别表示时隙的起始时间、终止时间和门控状态（用长度为 8 的 01 字符串表示，1 表示开，0 表示关）。值（value_2）表示在该时隙内通过的 TT 流的信息，以 $\{\phi_1,\tau_1,01100000\}$ 为例，在该时隙开启的时间段内，TT 流 $s_1,s_1,\cdots,s_n$ 通过了 link_1 源节点，传输时间段的开始和结束时间点分别为 $t_1,e_1,t_2,e_2,\cdots,t_n,e_n$。

算法 2 描述了 TSMR 过程，主要完成了对循环流的调度，最终调度结果保存在 LTS 中。

算法 2 TSMR

输入 分组 D_k 中循环流构成的集合 S'，分组 D_k 的组周期 GP_k，分组内实时流的调度方案 LTS（已经包含无等待调度的结果）

输出 true 或 false，分别表示可调度与不可调度，在可调度的情况下给出调度方案（key_2）

1) while 循环流 $s_i \in S'$
2) while s_i 路由的第 r 段链路 $[p_a,p_b]$
3) for $k \leftarrow 0$ to $\frac{\mathrm{GP}_k}{s_{i,1}}-1$
4) 利用模运算计算 t_s、t_ω 和 t_e
5) 构造时隙信息和数据帧的发送时间信息 $\mathrm{ts}_i \longleftarrow < t_s,t_\omega,\text{"00100000"} > \mathrm{ti}_i \longleftarrow < s_{i,k}^{[p_a,p_b]}.\phi,s_{i,k}^{[p_a,p_b]}.\phi + s_{i,2}\,p_a.t,s_i >$
6) if ts_i 跨越了 2 个 BP
7) return false
8) end if
9) while $\mathrm{ts}_j \in \mathrm{LTS}[R_{i.m}^r]$
10) 根据式(7)判断 ts_i 和 ts_j 冲突
11) if 有冲突
12) 将 ts_j 合并到 ts_i 中
13) while $\mathrm{ti}_j \in \mathrm{LTS}[R_{i,m}^r][\mathrm{ts}_j]$
14) 若破坏了已有调度返回 false
15) 计算排队时延
16) 更新 t_s、t_ω、t_e、ts_i、ti_i
17) 将 ts_i 和 ti_i 保存到 $\mathrm{LTS}[R_{i,m}^r]$ 中将 ti_j 保存到 $\mathrm{LTS}[R_{i,m}^r][\mathrm{ts}_i]$ 并删除 ts_j
18) end while
19) else
20) 将 ts_i 和 ti_i 保存到 $\mathrm{LTS}[R_{i,m}^r]$ 中
21) end if
22) end while
23) end for
24) end while
25) end while

4　多目标遗传算法优化

NW-TSMR 调度需要考虑网口GCL长度、循环流缓存时间以及约束等因素，是一个典型的多目标优化问题。基于NSGA_III[21]设计基因编码，并通过种群的初始化、选择、交叉和变异操作完成对NW-TSMR调度的优化。

4.1　优化目标函数

为了提高解的质量，本节设计了5个目标函数，从不同角度对个体进行优化。

1) 最小化单个网口的最长GCL：网络中的所有网口，通过比较LTS每个链路标识对应的时隙（key_2）数量，获得目标函数的值为

$$o_1 = \min\left\{\max\left\{\text{LTS}\left[\text{link}_x\right]\right\}\right\} \tag{8}$$

2) 最小化GCL的总长度o_2

$$o_2 = \min\left\{\sum_{x=0}^{n}\text{LTS}\left[\text{link}_x\right]\right\} \tag{9}$$

3) 最小化循环流缓存时间：在得出排队时延后，通过累加记录得到每个个体中循环流的总缓存时间d_q

$$o_3 = \min\left\{d_q\right\} \tag{10}$$

4) 最大化分组数量

$$o_4 = \min\left\{\frac{1}{D.\text{len}}\right\} \tag{11}$$

5) 最大化门控操作合并次数：在每次对时隙进行合并时，记录累加合并次数m

$$o_5 = \min\left\{\frac{1}{m}\right\} \tag{12}$$

4.2　个体和基因编码

个体包含2个基因，用2个数组表示，分别是每条TT流的路由选择数组C和初始发送时间选择数组Φ。这2个数组等长，对应位置的元素分别为路由方案选择以及流的初始发送时间。此外，每个个体还包含一个中间值，如图4所示，包含D、GP、LTS和O属性，分别表示当前个体的TT流分组的组周期、链路、TT流的映射关系以及优化目标值。

4.3　初始化

初始化主要实现对每个个体的2个基因编码数组的赋值。首先，初始化路由，对于循环流，从满足端到端时延的路由中随机选取一条；对于实时同步流，如果某条路由的端到端时延大于DDL，则不考虑该路由。在选取路由后，利用算法1进行分组，分组结果保存在D中。在分组后，进一步确定每个分组的组周期GP_k以及每条链路的BP。

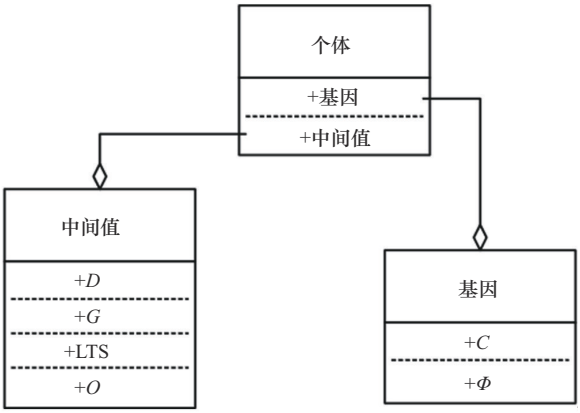


图4　个体编码示意

其次，对发送时间进行初始化。对每个分组中的TT流按照周期升序排序，对于周期最小的流，发送时间设为0。从第二条流开始，随机选择一个满足帧约束$\forall s_i \in S{:}s_{i,0} \in \left[0, s_{i,1} - s_{i,2}m_i\right)$的值为其赋值。

4.4　选择

完成初始化或遗传操作后，对每个个体进行评估。首先需要生成实时同步流的调度结果，具体做法是根据链路BP、实时同步流的初始发送时间、路由信息以及式(1)计算每条实时同步流的所有传输时隙；其次利用式(7)判断是否满足NW的条件；然后利用式(4)判断是否满足截止时间要求；最后根据算法2对循环流进行调度。

在调度过程中，进一步判断每条循环流的时隙是否可以被映射到对应的BP中，如果LTS中某个时隙的开始时间在该循环流的时隙内（算法2步骤5的ts_i）内，则视为破坏了LTS中的原有调度，无法继续利用TSMR策略调度，因此舍弃当前个体。以图2(b)和图2(d)为例，假设ts_j为LTS中的已有调度，ts_i为循环流的某个时隙，则图2(d)未破坏已有调度，图2(b)破坏了已有调度，阴影部分为排队时延，导致时隙ts_j内第一帧发生排队，第一帧可能是实时同步流的数据帧，与实时同步流不发生排队的前提相矛盾。在生成循环流调度的过程中，通过比较传输时段的方式计算排队时延，进一步利用式(3)计算端到端时延，利用式(5)判断每条循环流的传输是否满足端到端时延的要求，如果无法满足，需要舍弃当前个体。

另外，在选择的过程中，需要利用式(8)~式(12)完成对优化函数值的赋值，并保存在O中。

4.5　交叉和变异

交叉操作涉及2个亲本个体P_1、P_2以及后代个体offspring，随机生成2个介于0~1的系数δ和λ。首先生成路由选择数组，路由选择offspring.r = $\lambda P_1.r + (1-\lambda)P_2.r$。然后利用算法1重新为offspring分组，每组内周期最小的TT流的初始发送时间为0，其余TT流的初始发送时间为offspring.ϕ = $\delta P_1.\phi + (1-\delta)P_2.\phi$。

变异操作首先随机生成变异的半径，在此范围内随机生成一个值并与offspring.r相加，得到新的初始路由选择。然后进行重新分组，将周期最小的TT流的发送时间设为0，其余流则在其发送时间上累加一个随机数。

4.6　GCL输出

在遗传算法迭代结束后，每个候选解的中间值LTS中保存了详细的时隙信息，可以很方便地转化为标准的GCL，本节不再赘述。针对2个相邻门控操作之间的间隔，用一个门控全开的门控表项填充。

5　仿真实验与结果分析

为了验证NW-TSMR方法的正确性和有效性，本文基于NSGA_III函数库openGA[21]，使用C++

实现了NW-TSMR调度、优化算法和基于HPC的无等待调度方法（HP-NW）。HP-NW是一种典型的HPC算法，其严格确保TT流不发生排队，模型相对简单，扩展性更强，求解结果利用仿真平台NeSTiNg[18]进行验证。在调度能力评估中，除了NW-TSMR和HP-NW，还增加了与文献[11]中提出的两阶段调度（TSS, two-stage scheduler）方法的对比。本文实验使用的计算机系统为Ubuntu 22.04 LTS，内存大小为32 GB，CPU型号为Intel(R) Core (TM) i5-9500 CPU @ 3.00 GHz，6核6线程。

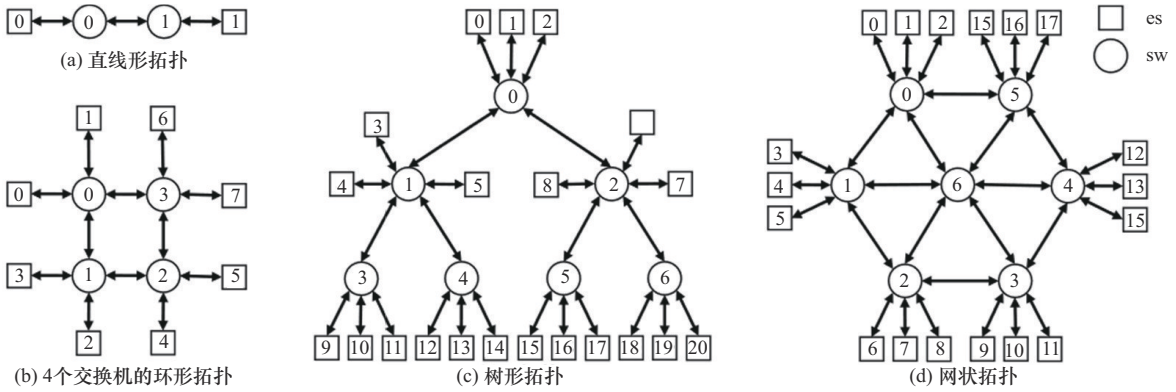
设定种群个体的数量为50，迭代100代，交叉概率为0.7，变异概率为0.4。在本文仿真实验中，所有网线长度均设为20 m，链路速率为1 Gbit/s，$d_{\text{prop}}^{[p_a,p_b]}$ = 100 ns，因此，所有网口的发送速率也均为1 Gbit/s。

5.1　门控列表长度对比实验

根据表1随机生成TT流，分别利用NW-TSMR和HP-NW求解。在每个测试用例中，实时流占75%，循环流占25%。在图5中展示的每种拓扑结构中，生成流的数量从10条增加到50条，步长为10，一般需求解10次。对于TT流数量为40条和50条的情况，由于耗时较长，只求解3次。所有结果均按照单个网口最长GCL、GCL总长度、总缓存时间、单条流最大抖动和求解时间的优先级顺序进行升序排序，取第一个解作为相对最优的解。

单个网口最长GCL和TT流数量的关系如图6所示，反映了最优解中所有GCL长度的最高水平。与HP-NW相比，NW-TSMR的单个网口最长GCL平均缩短了60.1%。平均GCL长度和TT流数量的关系如图7所示，反映了最优解中所有GCL长度的平均值。NW-TSMR与HP-NW相比，平均GCL长度缩短了约59.7%。由于遗传算法随机搜索的偶然性，少数情况下NW-TSMR解的单个网口最长GCL大于HP-NW的求解结果，如图6(c)求解30条流的情况。但总体而言，NW-TSMR求解的平均GCL长度都小于HP-NW的求解结果。

表3对比了缓存时间、抖动和求解时间，虽然在某些情况下，NW-TSMR增加了缓存时间（排队



(a) 直线形拓扑　(b) 4个交换机的环形拓扑　(c) 树形拓扑　(d) 网状拓扑

图5　测试网络拓扑

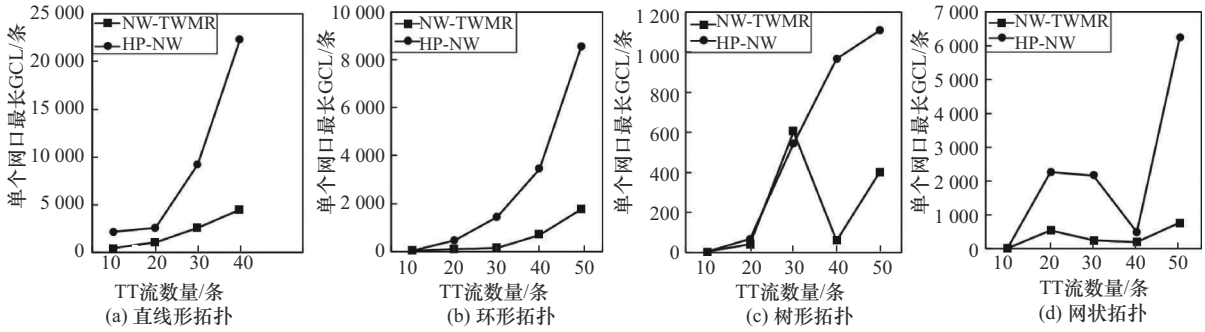


图6　单个网口最长GCL和TT流数量的关系

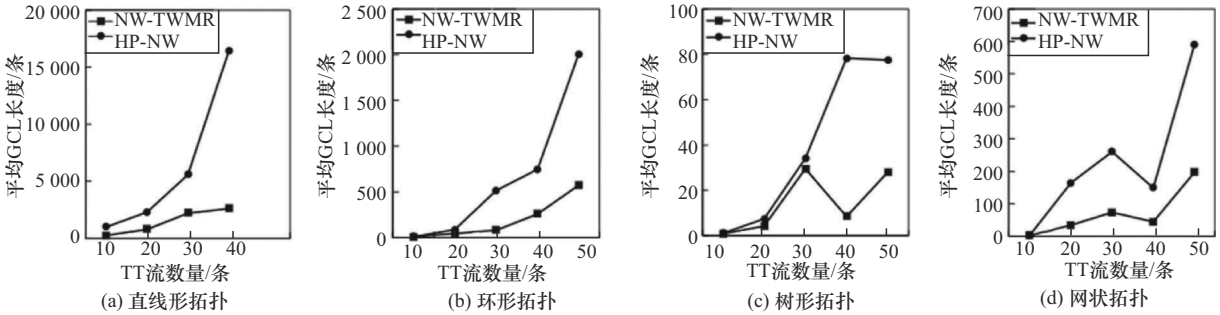


图7　平均GCL长度和TT流数量的关系

表3　　**缓存时间、抖动和求解时间对比**

拓扑	流数量/条	NW-TSMR			HP-NW		
		总缓存时间/ns	单条流最大抖动/ns	求解时间/s	总缓存时间/ns	单条流最大抖动/ns	求解时间/s
直线形	10	3 176.00	42.00	28.98	0	0	44.29
	20	272 940.00	0	133.58	0	0	118.32
	30	0	0	1 424.23	0	0	3 048.45
	40	0	0	7 523.35	0	0	37 984.60
环形	10	0	0	6.34	0	0	6.59
	20	1 784.00	0	31.52	0	0	19.31
	30	0	0	131.89	0	0	150.77
	40	0	0	370.14	0	0	233.99
	50	76 770.00	0	9 339.63	0	0	53 493.00
树形	10	0	0	3.50	0	0	3.53
	20	0	0	11.07	0	0	6.38
	30	0	0	48.30	0	0	21.53
	40	4 887.00	0	30.84	0	0	35.89
	50	0	0	160.21	0	0	54.41
网状	10	0	0	9.10	0	0	5.14
	20	96.00	0	107.07	0	0	86.40
	30	20 160.00	223.37	342.91	0	0	150.81
	40	0	0	1 017.91	0	0	416.35
	50	0	0	17 475.90	0	0	64 258.50

时延）和抖动，但都在表1定义的可接受范围内。一些情况下排队时延增加而最大抖动依然为0，如图5(a)中直线形拓扑求解20条TT流的情况，这是因为在组周期内发生排队的循环流每次传输都增加了相等的排队时延，导致每次发送的端到端时延也相等。如果排队的循环流在组周期内的排队时延不一致，最终的端到端时延也会有差异，此时抖动不为0，如图5(a)中直线形拓扑求解10条TT流的情况。

在直线形拓扑50条流时，2种方法均未给出结果，可见单条链路可容纳的TT流数量存在上限。这主要是由于约束条件要求较高，TT流在每跳的时间是通过累加严格推导出来的，适当放宽约束可以提高链路容量。在求解时间方面，当流数量较少时，两者差异不大，但在流数量较多时，NW-TSMR方法明显优于HP-NW方法。

5.2　调度能力实验

为了评估NW-TSMR在接近真实场景下的调度能力，以图5(c)中的树形拓扑为例，逐步增加TT流数量。在流数量从10条增加到50条时，步长为10；从50条增加到150条时，步长为25。其中，实时同步流占75%，循环流占25%，TT流依然采用随机生成的方式。NW-TSMR和HP-NW对所有测试用例都给出了求解结果。

图8和图9分别对比了单个网口最长GCL和平均GCL随TT流数量的变化。NW-TSMR求解的结果中，平均GCL长度和单个网口最长GCL均平稳上升；而HP-NW求解的结果中，平均GCL长度和单个网口最长GCL从125条流开始增长幅度变大。单个网口最长GCL和平均GCL长度相差的比例总体上都超过70%。图9中还对比了TSS方法，其平均GCL长度随着TT流数量的增加变化不大，主要原因是文献[11]中的同步流周期和调度循环时间一致，而本文中同步流周期的选取范围在[1, 2, 3, 4, 5, 6, 8, 10, 12, 15, 16, 18, 20]×100 μs内，循环流的周期选取范围在[2, 3, 4, 5, 6, 8, 10, 12, 15, 16, 18, 20] ms内，PR相对文献[11]较大。在PR较大的情况下，通常调度循环时间也更长[12]，周期较小的实时流在调度循环时间内发送次数更多，需要更多的门控操作来控制传输过程，导致GCL相对于PR小的TT流的调度循环时间更长。但从图8和图9中可以看出，PR相同的情况下，NW-TSMR比HP-NW生成的GCL更短，主要是因为使用BP作为调度循环时间，BP相较于HP更小，使得调度实时TT流所需的GCL更短。当TT流的数量超过100条时，NW-TSMR缩短GCL的效果尤其明显。

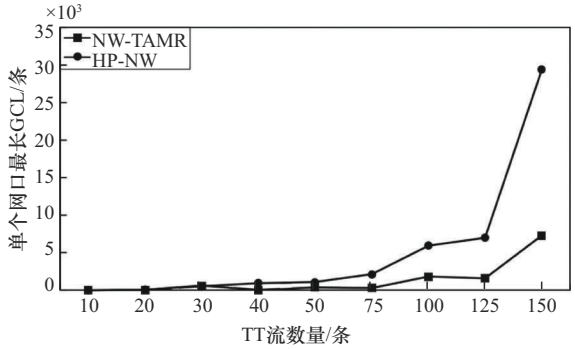


图8　单个网口最长GCL随TT流数量的变化

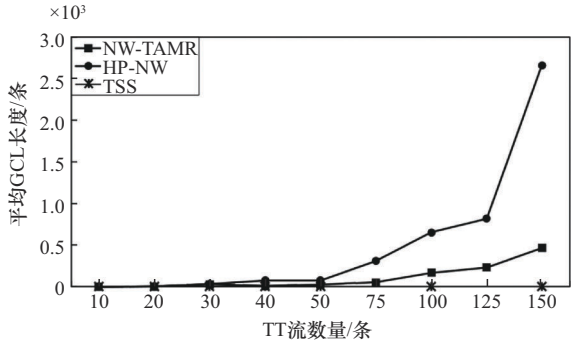


图9　平均GCL长度随TT流数量的变化

图10对比了3种策略的求解时间，可以看到，求解时间随着TT流数量的增加而增长。测试用例1到测试用例7的求解时间都在可接受范围内，而测试用例8和测试用例9的求解时间大幅增长。NW-TSMR的求解时间分别比HP-NW缩短了91.3%和69.9%，主要是因为利用BP作为调度循环时间，使得不同时隙的冲突检测局限在较小的时间范围内。而以HP为调度循环时间的HP-NW，则意味着要检测的冲突范围更大，也更耗时。TSS方法的求解时间较短，主要因为其采取批量调度的方式调度跨越不同网络层级的流，并且同一层级的TT流的PR较小，冲突检测的时间范围较小[12]。但是批量调度导致这种方式对连接不同层的边界交换机缓存压力较大[11]，而NW-TSMR方法在队列中最多缓存一帧，对交换机的缓存压力几乎可以忽略。

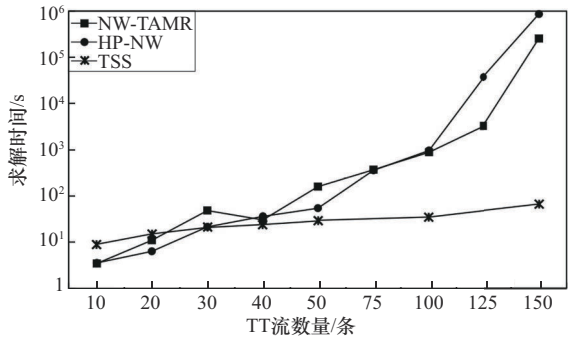


图10　3种策略的求解时间

5.3　不同优化目标的对比实验

为了验证不同优化目标对调度结果的影响，以图5(c)中的树形拓扑为例，随机生成40条TT流，利用NW-TSMR进行求解，结果如表4所示。前五组实验依次缺省优化目标函数o_1～o_5，第六组实验正常求解作为对照组。由于遗传算法的搜索存在偶然因素，因此实验结果中的每个指标取最后一代所有候选解的平均值，从而反映不同优化目标对整体的影响。表4中标记了每个优化目标下的最差情况。

在表4中，o_1对单个网口最长GCL的影响最显著，缺省o_1的情况下单个网口最长GCL平均值从787.37增加到1 201.53；o_2对GCL总长度的影响较显著，缺省o_2的情况下GCL总长平均值从5 084.66增加到5 673.11；o_3对单条流最大抖动的影响最显著，缺省o_3的情况下单条流最大抖动的平均值从12.90 ns增加到141.76 ns，对总缓存时间反而没有显著影响；o_4对分组数量的影响较显著，缺省o_4的情况下分组数量平均值从1.13个下降到1.00个；o_5对门控操作合并次数的影响较显著，门控操作合并次数平均值从7 275.81次减少到6 649.88次。同时，单个网口最长GCL、GCL总长以及总缓存时间的平均值均为最小。

分析结果表明，为了增加门控操作合并次数，进化过程中倾向于选择较长的路由。较长的路由潜在的门控操作合并次数更多，排队导致的总缓存时间也更长。因此，门控操作合并次数最大化无法使求解结果更优。

6　结束语

针对工业控制系统中TT流的调度问题，本文建立了系统模型，简化了TSN通用调度机制中的约束条件，并提出了NW-TSMR方法。与无等待调度相比，NW-TSMR在缩短GCL长度和加快求解速度方面效果显著。在设计调度方案时，本文考虑了周期对调度结果的影响，使用BP作为调度循环时间，设计了用于保存链路时隙信息的数据结构LTS，并详细分析了TSMR过程。此外，本文构造了NW-TSMR调度多目标优化遗传算法，达到了更短的GCL和更好的综合效果，具有很好的实用性。

TSN相关协议从不同角度为工业控制系统的稳定运行提供了可靠保障。如何结合这些协议进行统一调度，充分发挥各自的优势，不断提高物理链路容量，仍是一个巨大的挑战。未来的研究考虑将NW-TSMR扩展到网络规模更大的工业控制系统应用场景，结合其他TSN协议如IEEE 802.1Qch、IEEE 802.1Qci等，探索不同传输选择算法对TT流的影响，利用IEEE 802.1Qci提供的限流能力对低优先级干扰流进行限制，以保证高优先级流量的带宽需求。通过进一步优化调度机制和资源分配策略，有望提升系统的整体性能和可靠性，为实际工业应用提供更有力的支持。

参考文献：

[1] BOYES H, HALLAQ B, CUNNINGHAM J, et al. The industrial Internet of things (IIoT): an analysis framework[J]. Computers in Industry, 2018, 101: 1-12.

[2] LAKI S, GYÖRGYI C, PETŐ J, et al. In-network velocity control of industrial robot arms[C]// Proceedings of the 19th USENIX Symposium on Networked Systems Design and Implementation. Berkeley: USENIX Association, 2022: 995-1009.

[3] 黄韬, 汪硕, 黄玉栋, 等. 确定性网络研究综述[J]. 通信学报, 2019, 40(6): 160-176.

表4　优化目标对比

缺省优化函数	单个网口最长GCL	GCL总长	总缓存时间/ns	单条流最大抖动/ns	分组数量/个	门控操作合并次数/次
o_1	1 201.53	5 646.45	24 536.82	8.70	1.05	7 760.73
o_2	721.51	5 673.11	2 681.20	7.91	1.07	7 787.31
o_3	775.24	4 998.33	4 265.03	141.76	1.25	7 779.94
o_4	778.64	4 938.22	6 013.92	16.83	1.00	7 721.38
o_5	399.28	3 234.10	1 478.75	12.24	1.00	6 649.88
无	787.37	5 084.66	13 918.98	12.90	1.13	7 275.81

HUANG T, WANG S, HUANG Y D, et al. Survey of the deterministic network[J]. Journal on Communications, 2019, 40(6): 160-176.

[4] 张彤, 冯佳琦, 马延滢, 等. 时间敏感网络流量调度综述[J]. 计算机研究与发展, 2022, 59(4): 747-764.
ZHANG T, FENG J Q, MA Y Y, et al. Survey on traffic scheduling in time-sensitive networking[J]. Journal of Computer Research and Development, 2022, 59(4): 747-764.

[5] WG802.1. IEEE standard for local and metropolitan area networks-bridges and bridged networks-amendment 25: enhancements for scheduled traffic: IEEE Std 802.1Qbv-2015[S]. 2016.

[6] 聂宏蕊, 李绍胜, 刘勇. 时间敏感网络中基于IEEE 802.1Qch标准的优化调度机制[J]. 通信学报, 2022, 43(9): 12-26.
NIE H R, LI S S, LIU Y. Optimized scheduling mechanism based on IEEE 802.1Qch standard in time-sensitive networking[J]. Journal on Communications, 2022, 43(9): 12-26.

[7] STEINER W. An evaluation of SMT-based schedule synthesis for time-triggered multi-hop networks[C]//Proceedings of the 2010 31st IEEE Real-Time Systems Symposium. Piscataway: IEEE Press, 2010: 375-384.

[8] Industrial Internet Consortium. Time sensitive networks for flexible manufacturing testbed characterization and mapping of converged traffic types V1.0[R]. 2019.

[9] 蔡岳平, 李栋, 许驰, 等. 面向工业互联网的5G-U与时间敏感网络融合架构与技术[J]. 通信学报, 2021, 42(10): 43-54.
CAI Y P, LI D, XU C, et al. Integrating 5G-U with time-sensitive networking for industrial Internet: architectures and technologies[J]. Journal on Communications, 2021, 42(10): 43-54.

[10] CRACIUNAS S S, OLIVER R S, CHMELÍK M, et al. Scheduling real-time communication in IEEE 802.1Qbv time sensitive networks[C]// Proceedings of the 24th International Conference on Real-Time Networks and Systems. New York: ACM Press, 183-192.

[11] HELLMANNS D, GLAVACKIJ A, FALK J, et al. Scaling TSN scheduling for factory automation networks[C]//Proceedings of the 2020 16th IEEE International Conference on Factory Communication Systems (WFCS). Piscataway: IEEE Press, 2020: 1-8.

[12] DÜRR F, NAYAK N G. No-wait packet scheduling for IEEE time-sensitive networks (TSN) [C]// Proceedings of the 24th International Conference on Real-Time Networks and Systems. New York: ACM Press, 2016: 203-212.

[13] ARESTOVA A, HIELSCHER K S J, GERMAN R. Design of a hybrid genetic algorithm for time-sensitive networking[C]//International Conference on Measurement, Modelling and Evaluation of Computing Systems. Berlin: Springer, 2020: 99-117.

[14] LI Q, LI D, JIN X, et al. A simple and efficient time-sensitive networking traffic scheduling method for industrial scenarios[J]. Electronics, 2020, 9(12): 2131.

[15] 尹长川, 李妍珏, 朱海龙, 等. HSTC: TSN中的混合流量调度机制[J]. 通信学报, 2022, 43(6): 119-132.
YIN C C, LI Y J, ZHU H L, et al. HSTC: hybrid traffic scheduling mechanism in time-sensitive networking[J]. Journal on Communications, 2022, 43(6): 119-132.

[16] 邱雪松, 黄徐川, 李文萃, 等. 面向大规模时间敏感网络的分组调度机制[J]. 通信学报, 2020, 41(11): 124-131.
QIU X S, HUANG X C, LI W C, et al. Group-scheduling mechanism for large-scale time-sensitive network[J]. Journal on Communications, 2020, 41(11): 124-131.

[17] 李佳庆, 陈水忠, 魏刚, 等. 基于时间敏感网络的门控调度算法研究[J]. 电光与控制, 2023, 30(3): 58-62.
LI J Q, CHEN S Z, WEI G, et al. Research on gating scheduling algorithm based on time sensitive network[J]. Electronics Optics & Control, 2023, 30(3): 58-62.

[18] FALK J, HELLMANNS D, CARABELLI B, et al. NeSTiNg: simulating IEEE time-sensitive networking (TSN) in OMNeT[C]//Proceedings of the 2019 International Conference on Networked Systems (NetSys). Piscataway: IEEE Press, 2019: 1-8.

[19] FARZANEH M H, KNOLL A. Time-sensitive networking (TSN): an experimental setup[C]//Proceedings of the 2017 IEEE Vehicular Networking Conference (VNC). Piscataway: IEEE Press, 2017: 23-26.

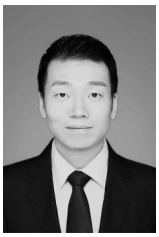
[20] YANG Y L, HANZO L. Permutation-based TCP and UDP transmissions to improve goodput and latency in the Internet of things[J]. IEEE Internet of Things Journal, 2021, 8(18): 14276-14286.

[21] MOHAMMADI A, ASADI H, MOHAMED S, et al. OpenGA, a C genetic algorithm library[C]//Proceedings of the 2017 IEEE International Conference on Systems, Man, and Cybernetics (SMC). Piscataway: IEEE Press, 2017: 2051-2056.

[作者简介]

何倩（1979– ），男，湖南安仁人，博士，桂林电子科技大学教授、博士生导师，主要研究方向为云服务、网络安全。

郭雅楠（1997– ），男，山西阳城人，桂林电子科技大学硕士生，主要研究方向为时间敏感网络流量调度。



赵宝康（1981– ），男，湖北天门人，博士，国防科技大学副教授，主要研究方向为网络体系结构与协议、卫星互联网、高性能网络、网络安全。

潘琪（1991– ），男，湖北黄冈人，桂林电子科技大学博士生，主要研究方向为网络控制系统、机器人运动规划。

王勇（1964– ），男，四川阆中人，博士，桂林电子科技大学教授、博士生导师，主要研究方向为云计算、网络流量分析、信息安全。