

基于改进萤火虫算法的 RapidIO 路由选择策略

殷从月, 张兴明, 任权, 魏帅

(国家数字交换系统工程技术研究中心, 河南 郑州 450002)

摘要: 针对 RapidIO 网络 QoS 路由选择问题, 提出一种基于改进萤火虫算法的 RapidIO 路由选择策略。首先, 利用高斯变异和存储机制对传统萤火虫算法进行优化, 高斯变异可以有效控制算法搜索空间中解的散射程度, 使算法避免陷入局部最优, 存储机制有利于评估并存储每只萤火虫的历史状态, 防止信息丢失。然后, 将改进后的萤火虫算法与实际 RapidIO 网络 QoS 问题相结合, 选择出最终的最佳路由策略。实验结果表明, 在所模拟的 RapidIO 测试网络中, 改进后的萤火虫算法时延为 42 ms, 时延抖动为 8 ms, 代价最低为 64 ms, 共需要迭代的次数为 8, 相较于其他算法曲线更加稳定, 更能快速找到最优解, 表现出的性能最优, 有效解决了 RapidIO 网络 QoS 路由选择问题。

关键词: RapidIO; 萤火虫算法; 高斯变异; 存储机制; 服务质量

中图分类号: TP393.02

文献标识码: A

doi: 10.11959/j.issn.2096-109x.2018054

RapidIO routing strategy based on improved glowworm swarm optimization algorithm

YIN Congyue, ZHANG Xingming, REN Quan, WEI Shuai

National Digital Switching System Engineering & Technological Research Center, Zhengzhou 450002, China

Abstract: Aiming at the problem of QoS routing in RapidIO network, a RapidIO routing strategy based on improved glowworm swarm optimization algorithm was proposed. Firstly, gaussian mutation and storage mechanism were used to optimize the traditional firefly algorithm. Gaussian mutation can effectively control the scattering degree of the solution in the search space of the algorithm, so that the algorithm avoids falling into a local optimum. The storage mechanism is conducive to evaluating and storing the historical state of each glowworm, preventing information loss. Then combine the improved glowworm swarm optimization algorithm with the actual RapidIO network QoS problem and select the final best routing strategy. The experimental results show that in the simulated RapidIO test network, the improved glowworm swarm optimization algorithm has a delay of 42 ms, delayed jitter of 8 ms, a minimum cost of 64 ms, and a total of 8 iterations, which is more stable than other algorithm curves. It can find the optimal solution more quickly and show the best performance, effectively solving the QoS routing problem of RapidIO network.

Key words: RapidIO, glowworm swarm optimization algorithm, gaussian mutation, storage mechanism; quality of service

收稿日期: 2018-04-14; 修回日期: 2018-05-08

通信作者: 殷从月, 503637088@qq.com

基金项目: 国家科技重大专项基金资助项目 (No.2016ZX01012101); 国家自然科学基金资助项目 (No.61572520, No.61521003)

Foundation Items: The National Science Technology Major Project (No.2016ZX01012101), The National Natural Science Foundation of China (No.61572520, No.61521003)

1 引言

随着嵌入式系统互联技术的不断发展, PCI Express、InfiniBand、RapidIO、HyperTransPort 等高性能 I/O 互联技术的出现使信号处理和数据传输的效率越来越高^[1]。其中, RapidIO 总线技术支持多种拓扑结构, 具有低时延、高带宽、高可靠性(硬件保证高达 99.999%)的特点, 在嵌入式系统应用中发挥着重要作用^[2]。此外, RapidIO 作为一项高速总线技术, 对网络分组丢失率、吞吐量、带宽、延迟等 QoS (quality of service) 参数要求较高。

RapidIO 网络 QoS 问题是一个 NPC 问题(non-deterministic polynomial complete problem), 主要目标是在多约束条件下选择满足 QoS 请求的传输路径, 目前最常用的求解方法为启发式算法^[3]。文献[4]采用蚁群优化算法解决 QoS 多约束路由问题(ACO, ant colony optimization algorithm), 但该算法收敛速度慢, 容易陷入局部最优, 导致搜索停滞, 不利于发现更好的解, 而且搜索时间长, 算法复杂度较高; 文献[5]将 A 星算法(ASO, a-star optimization algorithm)与 RapidIO 路由分配问题相结合, 但将启发值项设定为 0 会使该算法变成广度优先搜索的模式, 从而失去 A 星算法启发式的特点, 并且不同节点之间对应不同的优化次序, 因此产生的结果不同, 难以达到负载均衡以及延迟上的最优化效果, 此外, 该算法在运行过程中的节点状态需要大量的空间进行存储, 这会严重影响算法执行速度; 文献[6-7]利用遗传算法(GA, genetic algorithm)选择满足条件的路径, 但该算法为局部搜索算法, 会因陷入局部极值而找不到可行解; 文献[8]通过约束严苛度的定义来评价各个 QoS 度量参数, 再引入 K 最短路径算法来选择可行路径(CSKPA, constraint severity-K Shortest path algorithm), 但该算法是基于搜索策略提出的, 没有充分考虑道路网络的分布特性, 同时没有限制搜索区域, 算法执行效率较低; 文献[9]将广度优先搜索和贪婪算法相结合(BGA, breadth first search and greedy algorithm)来解决 QoS 路由选择问题, 该算法一定程度上更适用于较大规模的

RapidIO 网络, 但时间和空间的复杂度仍然较高。

上述研究表明, 在 RapidIO 网络 QoS 路由选择问题中, 关键是找到能够满足 QoS 请求的耗费最小的组播树^[10]。此外, RapidIO 作为一种具有高实时性需求的嵌入式应用, 无法不限时地逼近最优解, 因此反复改进、在一定时间内不断追求近似解的方式比较适合于这种 QoS 目标关联较为紧密的网络, 该方式可以避免局部元陷入自然独立的状态, 在全局范围内快速地进行搜索, 使多目标优化过程中的平衡问题得到有效解决^[11]。

针对以上问题, 本文提出一种基于改进萤火虫算法的 RapidIO 路由选择策略, 该策略主要采用基于高斯变异和存储机制的萤火虫算法(GMGSO, glowworm swarm optimization algorithm based on gaussian mutation and memory Mechanism)来解决 RapidIO 网络 QoS 路由选择问题。在常规 QoS 模型和萤火虫算法(GSO, glowworm swarm optimization algorithm)的基础上加入高斯变异和存储机制, 高斯变异使搜索空间中解的散射程度得到控制, 以此提高收敛速度并在搜索空间中寻找新的区域, 使算法有效避免陷入局部最优; 存储机制有利于记录 GSO 算法中每只萤火虫的位置, 如果在之前的搜索迭代过程中未能找到满足 QoS 请求的解, 则通过存储机制对得到的解进行评估并存储评估结果。实验结果表明, GMGSO 算法可以快速找到最优解并且最优解的可靠性较高, 保证解的质量, 有效解决了 RapidIO 网络中的 QoS 路由选择问题。

2 问题模型

2.1 RapidIO 网络结构

RapidIO 网络主要由 2 个部分组成, 分别为交换机和终端器件^[12]。图 1 是一个典型的 RapidIO 网络拓扑结构。RapidIO 网络控制和数据信息的传递过程需要 RapidIO 包, 通过交换机的内置路由表, 可以实现 RapidIO 包的路由, 即这些数据分组包含不同的目的 ID, 而内置路由表在交换机的每一个端口处也均有配置, 根据数据分组所提供的目的 ID, 交换机通过路由

表映射的方式将 RapidIO 包从输入端口传送至不同的输出端口^[13]。对 RapidIO 封装和解析的处理主要由终端器件实现，每个终端器件都匹配一个设备标识符 ID，是数据分组的目的端（源端），在 RapidIO 网络数据交换过程中起到发起或接收数据的作用，通过交换机可以实现终端设备的互联^[14]。

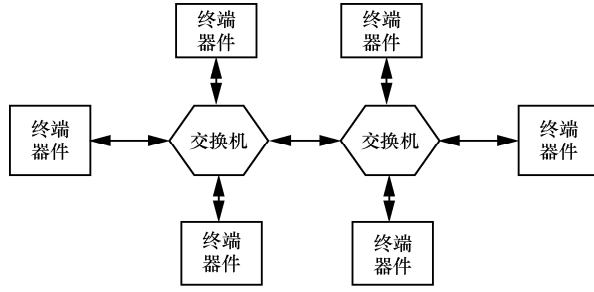


图 1 RapidIO 网络拓扑结构

2.2 QoS 模型

在 RapidIO 网络中，时延、跳数、分组丢失率和带宽是最常见的 QoS 问题^[15]，QoS 路由问题实际上就是找出所有满足这些约束条件的路径，因此将 RapidIO 网络中的 QoS 路由选择问题抽象化，可以建立其相应的数学模型。

RapidIO 总线技术采用全双工通信^[16]，可以用无向图 $G=(V,E)$ 表示 RapidIO 网络路由拓扑结构。其中， V 表示网络中的所有节点， V 由 v 组成， v 表示拓扑结构中的各个交换机（节点），或者将 V 定义成由 s 和 m 组成， s 表示负责发送数据的源节点， m 表示负责接收数据的目的节点； E 是各个节点连接线的集合， E 由 e 组成， e 表示拓扑结构中的各个通信链路。对于网络中的各个节点 v 来说，存在 4 个约束条件： $delay(v)$ 表示节点 v 的时延， $delay_jitter(v)$ 表示节点 v 的时延抖动， $packet_loss(v)$ 表示节点 v 的分组丢失率， $cost(v)$ 表示节点 v 的代价，节点 s 和 m 以此类推。对于拓扑结构中的每条通信链路 e 来说，同样存在 4 个约束条件： $delay(e)$ 表示链路 e 的时延， $delay_jitter(e)$ 表示链路 e 的时延抖动， $bandwidth(e)$ 表示链路 e 的带宽， $cost(e)$ 表示链路 e 的代价。

假定一组源节点 s 和目的节点 m 的集合 V ，其中， $\{m\} \subseteq V - \{s\}$ ，则所得到的组播树 $T(s,m)$ 将在时延、时延抖动、代价、带宽以及分组丢失率

这几个方面满足以下关系。

$$Delay(p_T(s,t)) = \sum_{e \in p_T(s,t)} delay(e) + \sum_{v \in p_T(s,t)} delay(v) \quad (1)$$

$$Delay_jitter(p_T(s,t)) = \sum_{e \in p_T(s,t)} delay_jitter(e) + \sum_{v \in p_T(s,t)} delay_jitter(v) \quad (2)$$

$$Cost(T(s,m)) = \sum_{e \in p_T(s,t)} cost(e) + \sum_{v \in p_T(s,t)} cost(v) \quad (3)$$

$$Bandwidth(P_T(s,t)) = \min\{bandwidth(e), e \in P_T(s,t)\} \quad (4)$$

$$Packet_loss(P_T(s,t)) = 1 - \prod_{v \in p_T(s,t)} (1 - packet_loss(v)) \quad (5)$$

其中， $P_T(s,t)$ 表示从源节点 s 到目的节点 t 的路由路径。基于上述假设，多约束 QoS 路由问题的数学模型可以采用如下方式描述。

$$\min = cost(T(s,m)) \quad (6)$$

$$\max\{delay(p_T(s,t)), t \in \{m\}\} \leq D \quad (7)$$

$$\min\{bandwidth(p_T(s,t)), t \in \{m\}\} \geq B \quad (8)$$

$$\max\{delay_jitter(p_T(s,t)), t \in \{m\}\} \leq DJ \quad (9)$$

$$\max\{packet_loss(p_T(s,t)), t \in \{m\}\} \leq PL \quad (10)$$

其中， D 、 B 、 DJ 、 PL 分别表示约束条件时延、带宽、时延抖动和分组丢失率。

目前，对于 QoS 多约束优化问题来说，最有效的解决方法是惩罚函数^[17]，因此该问题的适应度函数可以用式(11)描述，其中， cf_1 和 cf_2 表示惩罚因子， DC 和 DJ 分别表示时延约束和时延抖动约束。

$$F(X) = cost + cf_1(delay - DC) + cf_2(delay_jitter - DJ) \quad (11)$$

3 解决方案

3.1 GSO 算法

群体智能优化算法的目的是依据优化问题的目标函数寻求全局解，与寻求多个解相比，找到一个全局解显得更容易。由于优化问题具有相同或不同的目标函数值，所以在该算法中能够立即找到不同种类的解非常关键，因此一个

群体可以将自己划分成不同的类别很重要^[18]。

GSO 算法属于群体智能优化算法，该算法模仿了萤火虫在控制自身发光方面的行为。这些萤火虫为了不同的目的而发光，如在繁殖过程中吸引其他萤火虫^[19]。

在 GSO 算法中，群体由数量为 N 的萤火虫个体组成。因此，在函数搜索空间中，萤火虫 i 在时间 t 存在于位置 $X_i(t)$ 上，发光强度由荧光素水平 $L_i(t)$ 表示，决策域由 $rd_i(t)$ 表示。根据目标函数 J ，荧光素水平与萤火虫个体位置的目标值有关。萤火虫发出的光越强，则它含有的荧光素水平越高，这只萤火虫越接近最佳位置，其目标函数值也越高。决策域内的其他萤火虫会被比自身具有更高荧光素水平的萤火虫吸引，若局部范围内的萤火虫具有更高的荧光素水平，其邻近的萤火虫会自行向其移动，最终几乎所有的萤火虫都会聚集在搜索区域内多个亮度较高的位置上。

GSO 算法主要分为 4 个阶段：萤火虫初始化、荧光素水平更新、萤火虫移动、萤火虫决策域更新。萤火虫初始化主要表示在指定的目标函数搜索区域中随机散布 N 只萤火虫，因此用于优化的常数被初始化，同时每只萤火虫的荧光素水平用精确值 L_0 初始化，决策域 rd_i 和径向传感域 r_s 也用精确初始值 r_0 初始化。在 GSO 算法中，荧光素水平更新是最关键的一步，因为目标函数的评估是在当前萤火虫位置 X_i 进行的。对于群体内的所有萤火虫来说，荧光素水平会根据目标函数值进行修改，式(12)表示荧光素水平更新过程。

$$L_i(t) = (1 - \rho)L_i(t-1) + \gamma J(X_i(t)) \quad (12)$$

其中， $L_i(t)$ 和 $L_i(t-1)$ 分别表示萤火虫 i 更新后和更新前的荧光素水平， ρ 表示荧光素衰变常数且 $\rho \in (0,1)$ ， γ 表示光强吸收系数， $J(X_i(t))$ 表示萤火虫 i 在当前萤火虫位置 X_i 第 t 次迭代时的目标函数值。

在整个移动阶段，每只萤火虫会试图根据荧光素水平和决策域 rd_i 吸引邻组萤火虫 $N_i(t)$ ，所依据的规则如式(13)所示，其中， j 表示靠近萤火虫 i 的某个体； $N_i(t)$ 表示邻组萤火虫， d_{ij} 表示萤火虫 i 和 j 之间的欧几里得距离， r_{di} 表示萤火虫 i 的决策域， $L_j(t)$ 和 $L_i(t)$ 分别表示萤火虫 j 和 i 的荧光素水平。

$$j \in N_i(t), d_{ij} < r_{di} \text{ 且 } L_j(t) > L_i(t) \quad (13)$$

通过以下 2 个步骤识别实际选择的相邻个体，首先，找出向具有更高荧光素水平的相邻个体移动的方向并计算其概率，如式(14)所示。

$$prob_{ij} = \frac{L_j(t) - L_i(t)}{\sum_{k \in N_i(t)} L_k(t) - L_i(t)} \quad (14)$$

其中， j 表示靠近萤火虫 i 的某个体；接着萤火虫 i 使用轮盘赌的方式从相邻组挑选出一只萤火虫，概率值越大的萤火虫越有可能从邻组中被挑选出来。

在萤火虫移动阶段，根据被选择的相邻位置来更新萤火虫的位置，如式(15)所示。

$$X_i(t) = X_i(t-1) + s \frac{X_i(t) - X_i(t-1)}{\delta_{ij}} \quad (15)$$

其中， $X_i(t)$ 和 $X_i(t-1)$ 分别表示萤火虫 i 更新后和更新前的位置， s 表示步长常数， δ_{ij} 表示萤火虫 i 和 j 之间的欧几里得距离。

决策域的更新是 GSO 算法的最后一步。此处决策域 rd_i 不断被更新，这使萤火虫在后续的迭代中更加灵活地创建相邻组。

$$rd_i(t+1) = \min\{r_s, \max[0, rd_i(t-1) + \beta(n_i - |N_i(t)|)]\} \quad (16)$$

式(16)在后续的迭代中被用于 rd_i 的更新，其中， $rd_i(t)$ 和 $rd_i(t-1)$ 分别表示萤火虫 i 更新前和更新后的决策域， r_s 表示恒定的径向传感范围， β 表示模型常数， n_i 表示用于控制相邻个体计数的常参数， $|N_i(t)|$ 表示相邻个体的实际数目。

在 GSO 算法的搜索空间中，组播树被编码成个体，即个体编码。文献[20]提出了一种基于先前节点序列的编码机制，这是因为在组播树中除了源节点，对所有其他节点而言只存在一个先前节点，这个方法有利于编码和解码，可以避免在组播树中创建环路，具体的编码方法如式(17)所示

$$X = (P_1, P_2, \dots, P_n) \quad (17)$$

其中， P_n 表示节点 i 的先前节点。不在组播树内节点的先前节点可以用 0 表示，源节点的先前节点则用其本身表示。如图 2 所示，组播树可以被编码为 $X = (1_1, 1_2, 2_3, 1_4, 4_5, 5_6, 0_7)$ ，同理所编成的码也可以解码为图 2 所示的组播树。

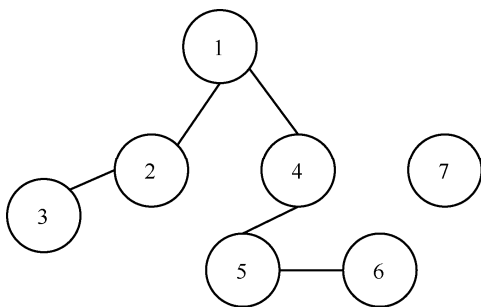


图2 组播树示意

虽然文献[20]所提的方法可以有效避免环路,但同时会产生许多不符合要求的树,因此可以对此编码方法进行改进,具体编码方法如算法1所示。

算法1 组播树编码算法

- 1) 将一棵空的组播树初始化成 $X = (0_1, 0_2, 0_3, \dots, 0_n)$ 。
- 2) 将源节点的先前节点设置成其本身,即 $X = (0_1, 0_2, \dots, S_s, \dots, 0_n)$ 。
- 3) 从该组目的节点中随机选择一个目的节点作为当前节点。
- 4) 从与当前节点相连的一组节点中删除当前节点的直接后继节点,并且生成当前节点的一组新的先前节点。
- 5) 从当前节点的一组先前节点中随机选取一个节点作为当前节点的先前节点,并且这个被选择的先前节点成为新的当前节点。
- 6) 如果当前节点的先前节点集为空,即“0”,则转到步骤3),否则转到步骤6)。
- 7) 如果所有目的节点的先前节点集为空,即“0”,则输出组播树的代码,否则转到步骤3)。

3.2 高斯变异

正态分布具有许多优良特性,许多随机因素、自然效应和概率分布可以由它来近似或体现^[21]。高斯分布在数学、物理学和工程学中是极为重要的概率分布,因此它在统计方面发挥着重要作用^[22]。高斯变异是在个体的原始状态中添加随机高斯向量的分布,其定义如式(18)所示。

$$X_i = X_i[1 + kN(0,1)] \quad (18)$$

其中,参数 X_i 表示个体 i 的状态, k 是从 1 到 0 逐渐减少的变量, $N(0,1)$ 是满足均值为 0 且方差为 1 的高斯分布条件的随机变量。如果式(18)在 X_i 的基础上增加降低高斯随机干扰因子 $X_i kN(0,1)$,则可以充分利用当前种群的给定信息进行干扰,从而增加种群状态的多样性,有利于避免陷入全

局搜索的局部极值点,同时提高算法搜索速度。

如果在算法迭代过程中萤火虫在局部极值点处聚合程度较高,则 GSO 算法的收敛速度将大大减慢甚至陷入局部极值,此时可以用种群的历史最佳状态代替最差的状态,以此获得中间状态种群,接着将高斯变异应用于处于中间状态的萤火虫。为了确定搜索结果是否随着迭代次数的增加而改善,可以在算法中设置一个调用板记录萤火虫历史最佳状态及其适应值。若在连续 3 次迭代过程中调用板上的值没有改变或变化幅度很小(小于 μ),则算法陷入局部极值范围内,此时必须开始进行变异操作。其中, μ 表示高斯变异的控制参数, μ 越大,高斯变异的概率越高,收敛速度越快。因此, μ 值不应太大或太小,数值在 10^{-4} 到 10^{-6} 之间是较为合适的选择。

3.3 存储机制

传统 GSO 算法在搜索过程中不会保留任何搜索空间信息,也不会保留每只萤火虫的历史状态信息,这导致萤火虫在不顾之前状态如何的情况下进行移动,萤火虫的历史信息到最后有可能会消失。如果在之前的搜索迭代过程中未能找到满足 QoS 请求的解,则通过存储机制对得到的解进行评估。之后具有目标值的新解被插入存储机制中,以防循环评估。此外,应该增加新的吸收参数(absorption parameter)^[23],同时参数应随时间和历史状态而变化。

3.4 QoS-GMGSO 算法

综合上述内容,将传统 GSO 算法与高斯变异和存储机制相结合成 QoS-GMGSO 算法,具体过程如算法2所示,主要分为以下步骤。

第1步,对 QoS-GMGSO 算法中的参数进行初始化,参数的设置和传统 GSO 算法中的设置相同,同时需要设置算法的终止条件。

第2步,计算当前萤火虫的适应值并启动调用板。

第3步,利用存储机制存储当前参数并时刻检查各项参数是否更新,若更新则及时存储新的参数并保留更新前的参数。

第4步,当算法到达荧光素水平更新阶段时,使用式(12)更新每只萤火虫的荧光素水平值。

第5步,当算法到达萤火虫移动阶段时,利用式(14)计算每只萤火虫的相邻个体和每只相邻个体可以被选为目标的概率。

第6步, 通过轮盘赌的方式确定搜索空间中可以替代萤火虫*i*的个体 $j(j \in N_i(t))$ 。

第7步, 利用式(15)计算搜索空间中每只萤火虫的目标位置。

第8步, 利用式(16)更新每只萤火虫的决策域, 此时萤火虫移动阶段结束。

第9步, 计算当前萤火虫的适应值, 若该值比调用板上的数据更好, 则调用板会被更新。

第10步, 高斯变异阶段, 利用式(18)将种群中最差的萤火虫用历史最佳萤火虫代替, 从而形成中间状态种群, 然后计算当前萤火虫的适应值, 若该值比调用板上的数据更好, 则调用板会被更新。

第11步, 当算法完成一次迭代时, 如果已经满足终止条件, 则退出迭代过程, 否则转到第4步进行下一次迭代。

图3为QoS-GMGSO算法的流程。图中从上到下依次按照传统GSO算法模块、存储机制模块以及高斯变异模块的顺序进行, 首先对每只萤火虫进行初始化, 接着利用存储机制对不断被更新的相关参数进行存储, 再通过高斯变异模块优化算法过程, 提高算法搜索速度和解质量, 最终输出满足QoS请求的解。

该算法总的时间复杂度为 $O(n^2)$, 其中, n 是萤火虫的数目。

算法2 QoS-GMGSO 算法

1) 设置 QoS-GMGSO 算法的维数和萤火虫数量

2) 初始化 QoS-GMGSO 算法中其他所有参数

3) 设置 $X_i(t)$ 为萤火虫*i*在时间*t*的位置

4) for $i = 1; i < m; i++$ do

5) $l_i(0) = 10$;

6) end for

7) $r_{i0}^i(0) = r_0$

8) 设置最大迭代次数 $T_{\max}$, 迭代计数值 $t = 1$;

9) while $t < t_{\max}$ do

10) for 所有的萤火虫*i* do

11) $l_i(t) = (1 - \rho)l_i(t-1) + \gamma J(x_i(t))$

12) end for

13) for 所有的萤火虫*i* do

14) for $j \in N_i(t)$ do

15) $p_{ij}(t) = l_j(t) - \frac{l_i(t)}{\sum_{k \in N_i(t)} l_k(t) - l_i(t)}$

16) $j = \text{select_glowworm}(\vec{p})$

17) $x_i(t+1) = x_i(t) + \frac{s(x_j(t) - x_i(t))}{\|x_j(t) - x_i(t)\|}$

18) $r_d^i = \min\{rs, \max\{0, r_d^+ \beta(n_i - |N_i(t)|)\}\}$

19) $\text{index} = \max(l)$

20) $\text{Brightest} = x_{\text{index}}$

21) $\text{Brightest} = \text{mutation}(\text{Brightest}, \text{try}, \text{male_num})$

22) $x_{\text{index}} = \text{Brightest}$

23) $t = t + 1$

24) end for

25) end for

26) end while

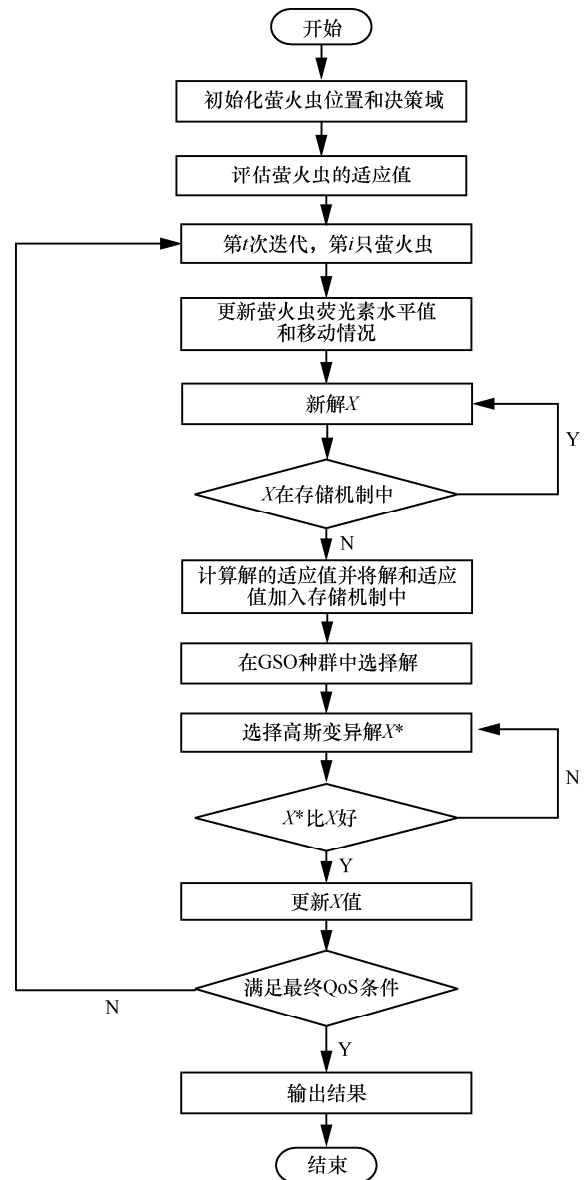


图3 QoS-GMGSO 算法流程

4 实验与结果分析

实验采用 Matlab 进行仿真,先假定一组用于实验的包含节点和链路的 RapidIO 模拟测试网络拓扑结构图和约束条件,再将 GMGSO 算法与 ACO 算法^[4]、ASO 算法^[5]、GA 算法^[6-7]、CSKPA 算法^[8]和 BGA 算法^[9]分别应用在该拓扑结构上,同时对满足 QoS 请求的输出结果进行比较。

4.1 实验设置

4.1.1 RapidIO 模拟测试网络

为了验证 QoS-GMGSO 算法的有效性,通过如图 4 所示的 RapidIO 网络拓扑结构进行模拟测试。假设图中所用交换机为 Tsi578 芯片^[24],图中包含了各个节点和链路的约束条件,节点的约束条件包括时延、时延抖动、分组丢失率和代价,链路的约束条件包括时延、时延抖动、带宽和代价,将节点 1 设置成源节点(配置主机),将节点 2、3、6、7 设置成数据必须最终到达的目的节点(终端器件)。

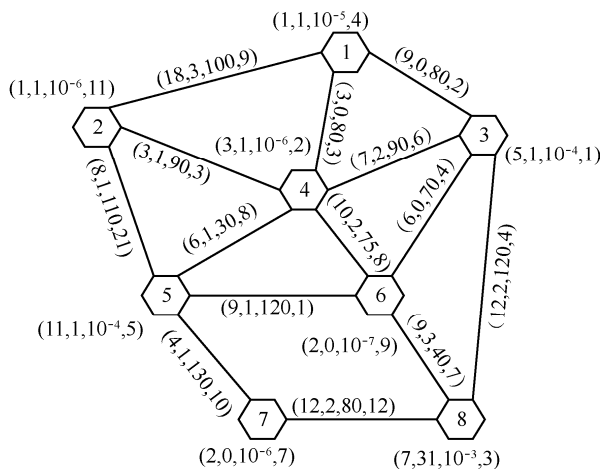


图4 RapidIO 模拟测试网络拓扑结构

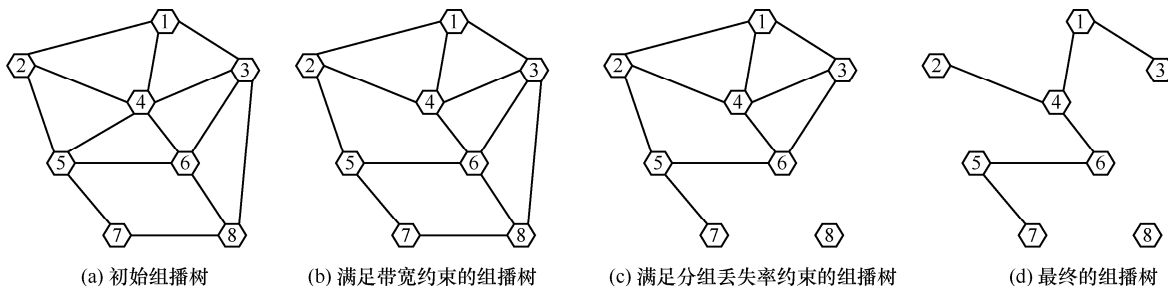


图5 RapidIO 模拟测试组播树简图

4.1.2 实验参数设置

QoS-GMGSO 算法的实验初始参数设置如表 1 所示,其中, $iter_max$ 表示最大迭代次数,其余参数在前文中均有说明。

表 1 QoS-GMGSO 算法实验参数

参数	值
ρ	0.4
r_s	5
γ	0.6
$iter_max$	200
N	20
D	46
B	70
DJ	8
PL	0.001
cf_1	10
cf_2	1

4.2 实验结果分析

4.2.1 最佳组播树

图 5(a)显示了链路的初始状态和节点的初始位置;图 5(b)显示了在考虑带宽约束条件之后的组播树,节点 4 和 5 之间的链路因不满足该约束条件而被删除;图 5(c)显示了在考虑分组丢失率约束条件之后的组播树;图 5(d)显示了在 20 次迭代之后最终的最佳组播树。

4.2.2 不同算法的性能对比

图 6 显示了不同算法多次迭代之后得到的代价、时延和时延抖动的对比情况。从图中可以看出, GMGSO 算法代价曲线的收敛速度明显快于其他算法,虽然一开始振动较大,但稳定后的代价值相较于其他算法来说更小; GMGSO 算法的时延曲线也比其他算法更加稳定,能够更快地找到最优解或者次优解;而这几种算法的时延抖动曲线大致相似;此外, ACO 算法和 GA 算法获得的只是局部最优解。

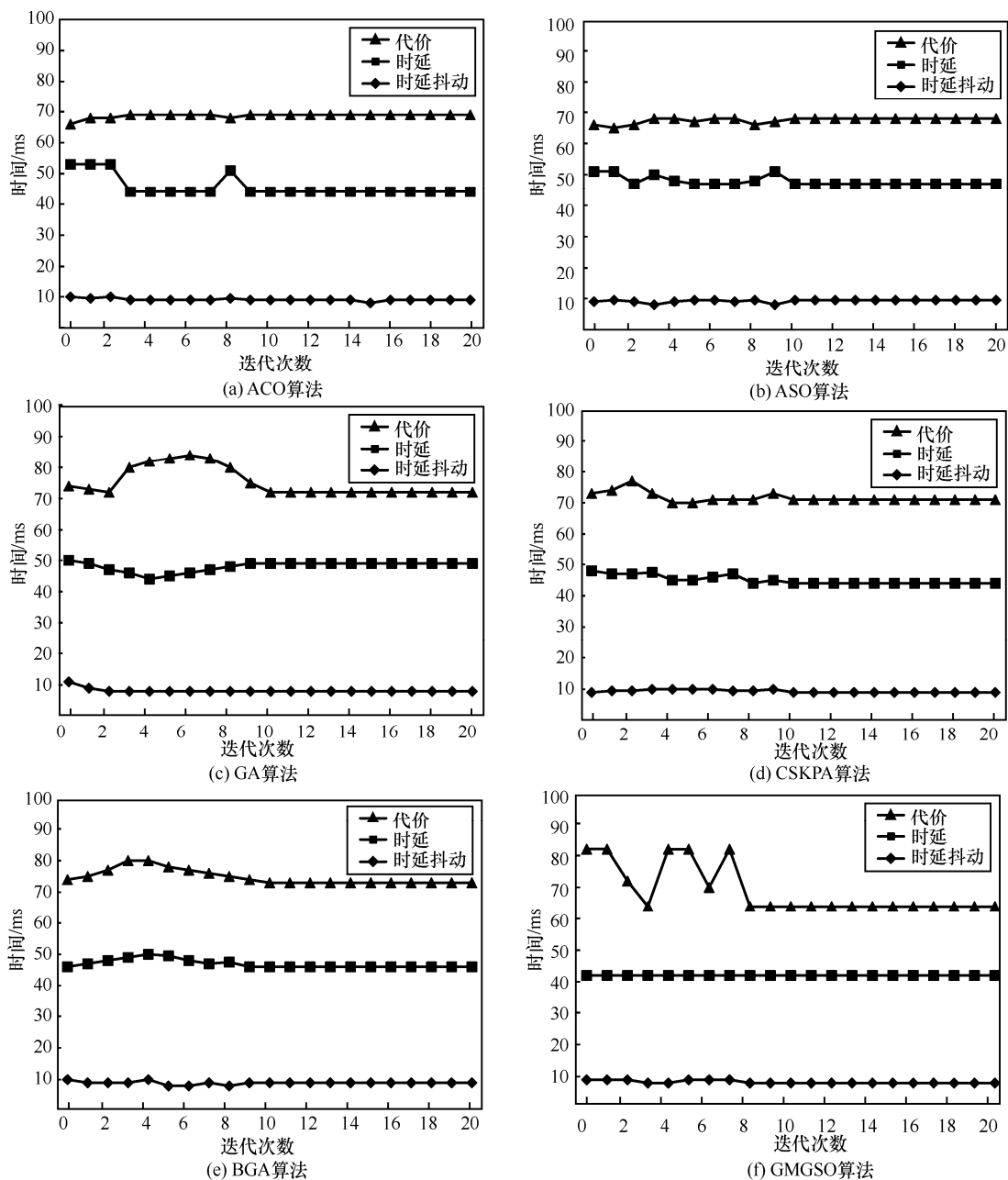


图6 不同算法性能对比

因此,在该 RapidIO 模拟测试网络拓扑结构中 QoS-GMGSO 算法的性能最优,由图 5(d)可以得到最佳组播树的编码为 $(1_1, 4_2, 1_3, 1_4, 6_5, 4_6, 5_7, 0_8)$, 时延为 42 ms, 时延抖动为 8 ms, 代价最低为 64 ms, 总共需要迭代的次数为 8。

5 结束语

本文针对 RapidIO 网络 QoS 路由选择问题,提出了一种基于改进萤火虫算法的 RapidIO 路由选择策略,并通过模拟的 RapidIO 测试网络来

验证其有效性。GMGSO 算法利用高斯变异和存储机制对传统 GSO 算法进行优化,高斯变异有效地提高了算法收敛速度,使算法避免陷入局部最优,存储机制评估和记录每只萤火虫的历史状态,有效防止信息的丢失。然后将 GMGSO 算法与 RapidIO 网络 QoS 问题相结合,通过 QoS-GMGSO 算法选择出路由最优方案。仿真结果表明,与其他算法相比,GMGSO 算法时延更加稳定,代价更低,更能快速找到最优解并且解质量较高。

参考文献:

- [1] 张娟, 苏海冰, 吴钦章. 基于多处理器的高速 RapidIO[J]. 计算机工程, 2010, 36(18): 238-239.
ZHANG J, SU H B, WU Q Z. Multi processor based on high speed RapidIO[J]. Computer Engineering, 2010, 36(18):238-239.
- [2] 黄亮, 刘福岩. 基于 RapidIO 和存储映射的高速互连网络[J]. 计算机工程, 2008, 34(14):116-117.
HUANG L, LIU F Y. High speed Internet based on RapidIO and storage mapping[J]. Computer Engineering, 2008, 34(14): 116-117.
- [3] DONG W, ZHANG W, YANG W. Node constraint routing algorithm based on reinforcement learning[C]//IEEE International Conference on Signal Processing. 2017: 1752-1756.
- [4] QIN L, CHEN Y, LUO J, et al. A new way of solving multiple constrained QoS multi-routing problems[C]//International Multi-Symposiums on Computer and Computational Sciences. 2006: 668-675.
- [5] 潘灵, 桑楠. 一种 RapidIO 网络路径分配策略[J]. 计算机应用, 2008, 28(s2): 294-295.
PAN L, SANG N. A RapidIO network path allocation strategy[J]. Computer Application, 2008, 28(s2): 294-295.
- [6] 蔡炜, 张建东, 蔡惠智. RapidIO 网络 QoS 多目标优化[J]. 计算机应用, 2010, 30(12): 3172-3175.
CAI W, ZHANG J D, CAI H Z. RapidIO network QoS multi-objective optimization[J]. Computer Application, 2010, 30(12): 3172-3175.
- [7] 蔡炜, 张建东, 蔡惠智. RapidIO 网络路由分配策略的优化和改进[J]. 计算机工程与应用, 2011, 47(14): 87-90.
CAI W, ZHANG J D, CAI H Z. RapidIO network route allocation strategy optimization and improvement[J]. Computer Engineering and Applications, 2011, 47(14): 87-90.
- [8] 李宗灿, 曹建. 基于约束分析的 RapidIO 路由选择算法[J]. 计算机工程与设计, 2014, 35(11): 3771-3775.
LI Z C, CAO J. RapidIO routing algorithm based on constraint analysis[J]. Computer Engineering and Design, 2014, 35(11): 3771-3775.
- [9] DONG W, ZHANG W, YANG W. Node constraint routing algorithm based on reinforcement learning[C]//IEEE International Conference on Signal Processing. 2017:1752-1756.
- [10] MATVEEVA N, SUVOROVA E, SHEYNIN Y. QoS mechanisms in SpaceFibre and RapidIO: SpaceWire networks and protocols, long paper[C]//SpaceWire Conference. 2016:1-8.
- [11] JAYASHEELAN P, JANE F M M. Effective route planning in road networks using multi constraint routing algorithm[C]//IEEE International Conference on Current Trends in Advanced Computing. 2016:1-6.
- [12] MCKENNY M, DINES J A B, HARLE D. Transporting multiple classes of traffic over a generic routing device - an investigation into the performance of the RapidIO™ interconnect architecture[C]//The, IEEE International Conference on Networks. 2003:39-44.
- [13] ZHANG X, GAO M, LIU G. A Scalable heterogeneous multi-processor signal processing system based on the RapidIO Interconnect[C]//International Symposium on Intelligent Information Technology Application Workshops 2008:761-764.
- [14] 石海洋. 一种 RapidIO 交换网络配置方法的设计与实现[J]. 航空计算技术, 2012, 42(2):132-134.
SHI H Y. Design and implementation of a RapidIO switching network configuration method[J]. Aviation Computing Technology, 2012, 42(2): 132-134.
- [15] 王婷, 戴小氏, 朱守园, 等. 一种静态配置复杂 RapidIO 网络的方法[J]. 信息通信, 2017(1):136-137.
WANG T, DAI X D, ZHU S Y, et al. A method for statically configuring a complex RapidIO network[J]. Information and Communication, 2017(1):136-137.
- [16] 施春辉, 柴小丽, 宋慰军, 等. 基于 SoPC 的前端 RapidIO 接口设计[J]. 计算机工程, 2011, 37(20):239-241.
SHI C H, CHAI X L, SONG W J, et al. SoPC-based front-end RapidIO interface design[J]. Computer Engineering, 2011, 37(20): 239-241.
- [17] DOLGOPOLIK M V. Smooth exact penalty functions: a general approach[J]. Optimization Letters, 2018, 10(3):635-648.
- [18] SHAHINZADEH H, MOAZZAMI M, FADAEI D, et al. Glowworm swarm optimization algorithm for solving non-smooth and non-convex economic load dispatch problems[C]//Iranian Joint Congress on Fuzzy and Intelligent Systems. 2017:103-109.
- [19] NUNES H, POMBO J, FERMEIRO J, et al. Glowworm swarm optimization for photovoltaic model identification[C]//Young Engineers Forum. 2017:59-64.
- [20] SUN J, FANG W, WU X, et al. QoS multicast routing using a quantum-behaved particle swarm optimization algorithm[J]. Engineering Applications of Artificial Intelligence, 2011, 24(1): 123-131.
- [21] CALSINA A, CUADRADO S, DESVILLETES L, et al. Asymptotic profile in selection-mutation equations: gauss versus cauchy distributions[J]. Journal of Mathematical Analysis &

Applications, 2016, 444(2):1515-1541.

- [22] WAN C, WANG J, YANG G, et al. Particle swarm optimization based on Gaussian mutation and its application to wind farm micro-siting[C]//Decision and Control. 2010:2227-2232.

- [23] LAIDANI N, BARTALI R, GOTTARDI G, et al. Optical absorption parameters of amorphous carbon films from Forouhi Bloomer and Tauc Lorentz models: a comparative study[J]. Journal of Physics Condensed Matter, 2008, 20(1):105-112.

- [24] 任雪倩, 李金城, 赵建中,等. 基于串行 RapidIO 的 Buffer 层设计[J]. 微电子学与计算机, 2016, 33(9):47-50.

REN X Q, LI J C, ZHAO J Z, et al. Design of Buffer layer based on serial RapidIO[J]. Microelectronics and Computers, 2016, 33(9): 47-50.

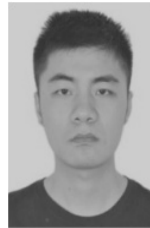
作者简介:



殷从月（1994-），女，江苏扬州人，国家数字交换系统工程技术研究中心硕士生，主要研究方向为 RapidIO 网络、异构计算系统。



张兴明（1963-），男，河南新乡人，国家数字交换系统工程技术研究中心教授、硕士生导师，主要研究方向为新型网络体系结构。



任权（1994-），男，湖南常德人，国家数字交换系统工程技术研究中心硕士生，主要研究方向为网络安全防御、顽健网络体系结构。



魏帅（1984-），男，河南南阳人，国家数字交换系统工程技术研究中心讲师，主要研究方向为嵌入式计算。