中国科学: 信息科学 2020 年 第 50 卷 第 2 期: 261–274
SCIENTIA SINICA Informationis
论文

DAG 区块链中基于确定性退火技术的融合分割遗传任务调度算法

曹怀虎*, 张艳梅, 王坚, 李海峰, 崔丽欣

中央财经大学信息学院, 北京 100081
* 通信作者. E-mail: chh@cufe.edu.cn

收稿日期: 2019–01–25; 修回日期: 2019–07–14; 接受日期: 2019–09–04; 网络出版日期: 2020–02–11

北京市社会科学基金重点项目 (批准号: 16YJA001)、国家自然科学基金项目 (批准号: 61602536, 61671030)、中央高校基本科研业务费专项资金和中央财经大学科研创新团队支持计划资助项目

摘要 传统的区块链结构, 由于其固有的响应速度慢, 不能适应大规模实时响应的应用场景, 本文针对这一问题, 提出了一种 DAG (directed acyclic graph) 区块链理论架构, 将传统区块链的链式处理过程转变为并行的处理过程, 使得快速响应成为可能. 在此基础上, 面向 DAG 区块链环境中非独立任务调度问题, 提出了基于确定性退火技术的混合分割遗传任务调度算法. 实验结果显示, 该算法能够适应 DAG 区块链节点的异质性、动态性和广域性, 其调度的性能也比传统的调度算法有所改善, 在优化任务完成时间的同时, 兼顾了负载均衡问题, 有效地提高了响应速度, 是解决 DAG 区块链环境中非独立任务调度问题的可行方法.

关键词 有向图, 遗传算法, 并行, 任务调度, 优化, 负载均衡

1 引言

区块链 (blockchain) 是通过去中心化、去信任的方式, 在一个由用户共同维护的可靠的分布式数据库基础之上, 提供可信互联网服务的技术方案 [1~3]. 从狭义上讲, 区块链是一种分布式记账方式, 它将数据区块按时间顺序线性组合而成链式数据结构, 并由密码技术保证其不被伪造和篡改. 从广义上讲, 区块链技术是一种新的分布式计算基础设施, 它利用链式数据结构对数据进行存储和验证, 利用分布式节点共识算法定义、更新数据, 利用密码技术保证数据访问和传输的安全性, 并使用自动化脚本代码组成的智能合约来操作数据并实现相应的服务 [4, 5].

目前, 比特币是比较成功的一种区块链应用, 比特币区块链的具体结构如图 1 所示. 每个区块是由区块体和区块头组成, 其中区块头包含随机数、前一个区块的区块头 Hash 值等, 区块体则是具体交易数据的集合. 比特币通过调整目标 Hash 值, 控制全网大约每 10 min 产生一个区块. 但是以比特

引用格式: 曹怀虎, 张艳梅, 王坚, 等. DAG 区块链中基于确定性退火技术的融合分割遗传任务调度算法. 中国科学: 信息科学, 2020, 50: 261–274, doi: 10.1360/N112019-00025
Cao H H, Zhang Y M, Wang J, et al. Fusion-partitioning genetic task scheduling algorithm based on deterministic annealing technology in DAG blockchains (in Chinese). Sci Sin Inform, 2020, 50: 261–274, doi: 10.1360/N112019-00025

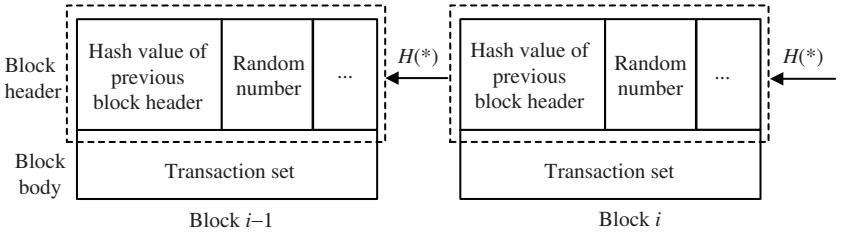

图 1　比特币区块链结构

Figure 1　Bitcoin blockchain structure

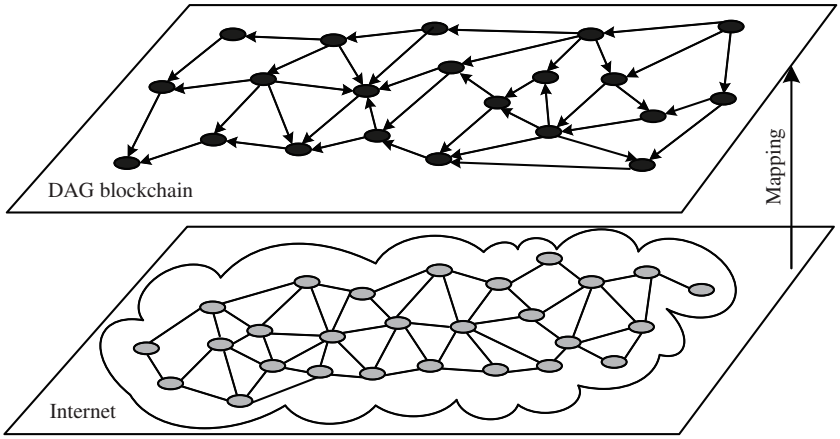

图 2　DAG 区块链理论架构

Figure 2　DAG blockchain theory architecture

币为代表的区块链技术并不适用于大规模的网络应用 [6], 例如在金融应用场景中, 高频操作是常见的服务, 银行卡、股票、外汇等金融业务的交易峰值每秒可达万笔之上, 再加上跨平台的交易需求, 传统区块链线性的链式结构已无法充分满足金融交易的需求 [7, 8]. 不论是数据存储空间, 还是网络数据同步速度, 金融应用场景对区块链的硬性要求都显著高于比特币 [9]. 为此, 业界推出了许多支持通用应用的区块链平台, 包括公有链和联盟链. 公有链中应用最广泛的通用平台是 Ethereum (以太坊), 其他的 Quorum, HydraChain, Monax, DFINITY, BCOS 等众多区块链平台都是基于 Ethereum 构建和扩展的. 联盟链中应用最广泛的通用平台是 Hyperledger Fabric, 其拥有 IBM, SWIFT, DTCC, Intel, J. P. Morgan, R3 等 130 多名成员. 但是这些平台目前也面临着一些问题, 如底层性能无法支持高并发的需求、链与链之间不能互通互联、无法适应不同的应用场景等 [10~15]. 为此, 本文提出一种 DAG (directed acyclic graph) 区块链理论架构, 该架构可以屏蔽底层网络的异构性、动态性, 并在此基础上提出了融合确定性退火技术和遗传算法的 DAG 分割任务调度算法.

2　DAG 区块链的理论架构

如前面所述, 链式的区块链结构, 由于其固有的响应速度慢, 不能适应大规模通用的应用场景, 因此, 本文提出了一种 DAG 区块链理论架构. 该架构如图 2 所示, 在物理互联网基础上, 根据具体的应用, 经过映射形成了动态虚拟的 DAG 区块链, 这是一种有向无环图, 它的响应速度将远远大于链式的结构. DAG 区块链的服务流程如图 3 所示, 首先, “拓扑映射” 模块根据某一类服务的特性, 在物理互

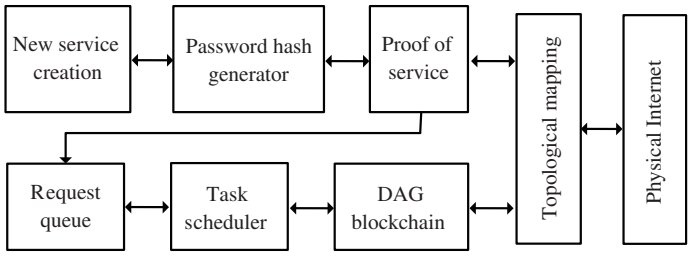

图 3 DAG 区块链服务流程

Figure 3 DAG blockchain service process

联网之上, 动态维护一个虚拟的 DAG 区块链 [16, 17], 新的服务请求创建后, 将根据区块链的 Hash 算法生成相应的密码散列, 完成服务验证, 进入服务请求队列, 同时与 "拓扑映射" 模块沟通是否需要构建新的 DAG 区块链 (可以根据服务的类型判断), 然后由 "任务调度器" 将任务分配到合适的节点上执行. 任务调度的目的是在包含大量异构资源的环境中, 同时考虑各节点之间的通信开销、计算性能等参数, 把不同的任务以最优的方案分配到合适的节点去完成. DAG 区块链的任务调度除了考虑指派任务给机器之外, 还要考虑机器的异构特性、任务之间的通信开销、任务之间的约束关系等因素, 将应用程序调度到异构的动态变化的节点上运行, 获得最优或次优的性能指标是调度算法研究的目标和方向.

3 DAG 区块链任务调度模型

3.1 DAG 区块链任务调度相关定义

一个复杂的应用是由一系列子问题组成的, 每一个任务包含一系列的子任务, 当然, 子任务之间可能不完全独立, 存在一些优先约束及通信, 通常这种任务调度问题可以用加权有向无环图 (DAG) [18~20] 来表示, 称之为任务图, 定义如下:

定义1 有向无环图 $G = (V, E, T, C)$ 由顶点和边组成, 其中 V 是任务节点集合, E 是通信边, t_i ($t_i \in T$) 是子任务 v_i ($v_i \in V$) 的处理时间, c_{ij} ($c_{ij} \in C$) 是边 e_{ij} ($e_{ij} \in E$) 上的通信开销, 表示任务 v_i 和 v_j 之间的数据传输时间. 如果 $c_{ij} = 0$, 则表示 v_i, v_j 被分配到同一个节点上, 此时, v_j 称为 v_i 的立即后继任务. 没有任何前驱的任务称为入口任务, 没有任何后继的任务称为出口任务, 没有任何前驱和后继的任务称为独立任务. 这样的有向无环图 G 称为任务图, 一个任务图的例子如图 4 所示, v_0, v_1, v_2 是入口任务, v_6, v_8, v_{10} 是出口任务, v_{11} 是独立任务.

为了叙述的统一性和严谨性, 下面对算法所用到的概念及参数进行定义:

定义2 $\mathrm{Tlevel}(v_i)$ 是从任务图的入口任务到任务 v_i 的最长路径的长度 (不包含任务 v_i 本身的执行时间), $\mathrm{pred}(v_i)$ 表示任务 v_i 的直接前驱任务集:

$$\mathrm{Tlevel}(v_i) = \max_{v_j \in \mathrm{pred}(v_i)} \{\mathrm{Tevel}(v_j) + t_j + c_{ji}\}.$$

定义3 $\mathrm{Blevel}(v_i)$ 是从任务图的任务 v_i 到出口任务的最长路径的长度 (包含任务 v_i 本身的执行时间), $\mathrm{succ}(v_i)$ 表示任务 v_i 的直接后继任务集:

$$\mathrm{Blevel}(v_i) = t_i + \max_{v_j \in \mathrm{succ}(v_i)} \{\mathrm{Blevel}(v_j) + c_{ij}\}.$$

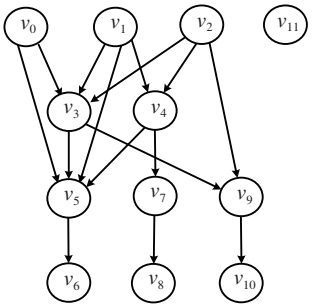

图 4　任务图示例

Figure 4　Task diagram example

定义4　任务 v_i 在主机 h_u 上的代价为 $\mathrm{Comp}(t_i,h_u)=\frac{\mathrm{Load}(v_i)}{\mathrm{CP}(h_u)}$, 这里 $\mathrm{Load}(v_i)$ 表示任务 v_i 的负载量, $\mathrm{CP}(h_u)$ 表示主机 h_u 的处理能力.

定义5　任务 v_i 和 v_j 之间的通信代价为 $\mathrm{Comm}(v_i,v_j,h_u,h_y)=\mathrm{Data}(v_i,v_j)\times\mathrm{Rate}(h_u,h_y)$, 这里 $\mathrm{Data}(v_i,v_j)$ 表示任务 v_i 和 v_j 之间的通信数据量, $\mathrm{Rate}(h_u,h_y)$ 表示 h_u 和 h_y 之间发送每单位数据的通信代价.

定义6　最早开始时间 $\mathrm{EST}(v_j,h_y)$: $\mathrm{EST}(v_j,h_y)=\max(\mathrm{Free}(h_y),\max_{\forall v_i\in\mathrm{pred}(v_j)}(\mathrm{EST}(v_i,h_u)+\mathrm{Comp}^*(v_i)+\mathrm{Comm}(v_i,t_j,h_u,h_y)))$, 这里 $\mathrm{Free}(h_y)$ 是主机 h_y 完成最后一个任务的时间, $\mathrm{Comp}^*(v_i)$ 表示任务 v_i 的预测执时间.

定义7　最早完成时间 $\mathrm{EFT}(t_j,h_y)$: $\mathrm{EFT}(t_j,h_y)=\mathrm{EST}(t_j,h_y)+\mathrm{Comp}^*(t_i)$.

3.2　DAG 区块链任务调度目标函数

调度一个 DAG 所表示的并行任务的一般目标是最小化任务执行时间 (Makespan), 但是找到最小化 Makespan 的最优算法是一个完全 NP 问题 [21], 所以许多研究采用启发式算法, 一般都不考虑节点之间的通信代价, 只考虑计算开销和 DAG 中任务优先权的关系. 目前主要的任务调度算法有 ILHA (ISO level heterogeneous allocation), CPM (critical path method), RANDOM, SCFET (smallest co-levels first with estimated times), HLFET (highestlevels first with estimated times) 等 [22, 23]. 这些任务调度算法不考虑异构环境的通信开销, 所以对于像 DAG 区块链这样基于互联网的异构系统, 这些算法将不再适用. 在这样大规模的动态环境中, 通信开销可能会很大, 所以必须要考虑通信开销对任务调度的影响. DAG 区块链任务调度问题的优化目标: 在考虑到任务之间的通信开销的前提下, 同时满足优先约束和负载平衡约束的条件下, 任务的总完成时间最小, 即

$$\min(\max(\mathrm{EFT}(t_j,h_y)))\quad \text{s.t.}\sum_{y=1}^{m}L_y\leqslant\sum_{j=1}^{n}\mathrm{Load}_j\leqslant\sum_{y=1}^{m}U_y, \tag{1}$$

其中 m 代表节点的个数, n 代表任务的个数, Load_j 代表第 j 个任务的负载, $\mathrm{EFT}(t_j,h_y)$ 代表第 j 个任务最早完成时间, U_y, L_y 代表第 y 个节点可处理负载的上下界.

4 融合确定性退火和遗传算法的 DAG 区块链任务调度算法

4.1 算法的基本思想

遗传算法的主要缺点是花费的调度时间较多, 它的调度时间取决于所调度问题的大小, 为了弥补这一不足, 本文提出了一种基于确定性退火技术的融合分割遗传算法 FPGA (fusion partitioned genetic algorithm based on deterministic annealing), 这个算法吸取了 divide-conquer (分治法) 算法的思想 [24], 在减少同一时间内所调度任务数量的同时, 将具有相关性的任务聚集在一个子任务图中, 统一调度, 减少任务之间的通信量. 这个算法依据每一个任务的 Blevel 值, 利用确定性退火技术, 将原始的任务图分割成多个子任务图. 首先计算每一个任务的 Blevel 值, 这里用 y_1, y_2, …, y_M 来表示, 针对非独立任务的分割问题, 利用确定性退火技术, 定义自由能函数 [25] 如下:

$$F(y_1,y_2,\dots,y_M,\beta)=\begin{cases}\dfrac{1}{M}\displaystyle\sum_{i=1}^{N}\sum_{j=1}^{M}(d(x_i,y_j))^2, & \beta=0,\\ -\dfrac{1}{\beta}\displaystyle\sum_{i=1}^{N}\ln\sum_{j=1}^{M}\mathrm{e}^{-\beta(d(x_i,y_j))^2}, & 0<\beta<+\infty,\\ \displaystyle\sum_{i=1}^{N}\sum_{j=1}^{M}(d(x_i,y_j))^2, & \beta=+\infty.\end{cases}\tag{2}$$

$x_i\in C_j$ 的概率为

$$p(x_i\in C_j)=\frac{\mathrm{e}^{-\beta(d(x_i,y_j))^2}}{\sum_{k=1}^{M}\mathrm{e}^{-\beta(d(x_i,y_k))^2}}.\tag{3}$$

若假定 $d(X,y_j)=\|X-y_j\|=(\sum_{k=1}^{n}(X(k)-y_j(k)^2))^{\frac{1}{2}}$, 当 $\beta=0$ 时, $F(y_1,y_2,\dots,y_M,\beta)$ 关于 $(y_1,y_2,\dots,y_M)$ 为连续可微的凸函数, 按传统的优化方法极易求出全局最优点; 当 $\beta\in[0,+\infty]$ 时, $F(y_1,y_2,\dots,y_M,\beta)$ 关于 $(y_1,y_2,\dots,y_M)$ 为连续可微, 由式 (2) 可知, $F(y_1,y_2,\dots,y_M,\beta)$ 的极小点与极小值为 β 的连续映射. 在 $F(y_1,y_2,\dots,y_M,\beta)$ 的极小点处, 满足一阶必要性条件, 即

$$\frac{\partial F(y_1,y_2,\dots,y_m,\beta)}{\partial y_j}=0\ (\forall j).$$

将式 (2) 代入上式得

$$y_i=\frac{\sum_{i=1}^{N}x_ip(x_i\in C_j)}{\sum_{j=1}^{N}p(x_i\in C_j)},\tag{4}$$

其中 $p(x_i\in C_j)$ 由式 (3) 确定. 当 $\beta=0$ 时, 有

$$p(x_i\in C_{j_i})=\frac{\mathrm{e}^{-\beta(d(x_i,y_j))^2}}{\sum_{j_i'=1}^{M}\mathrm{e}^{-\beta(d(x_i,y_{j_i'}))^2}}=\frac{1}{M}.\tag{5}$$

从而 $y_j=\frac{1}{M}\sum_{i=1}^{N}x_i$ 为 $\beta=0$ 时 $F(y_1,y_2,\dots,y_M,\beta)$ 的全局最优点 (因为对于凸规化, 局部极小点必为全局极小点), 它为 X 中所有点的加权平均. 由一阶必要性条件 (式 (4)), 可以得出求最优 y_i 的迭代公式:

$$y_j^{(k+1)}=\frac{\sum_{i=1}^{N}x_ip(x_i\in C_j^{(k)})}{\sum_{i=1}^{N}p(x_i\in C_j^{(k)})}.\tag{6}$$

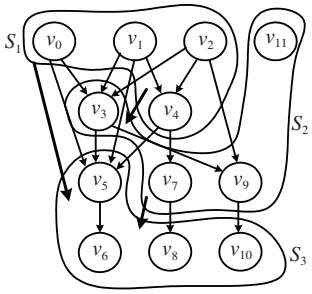

图 5　一个任务图 (图 4 所示) 的分割结果

Figure 5　Partitioning result of a task diagram (shown in Figure 4)

对应每一个 β (温度 $T \propto \frac{1}{\beta}$), 可按迭代式 (6) 来模拟分割系统的平衡态, 给出利用确定性退火技术的分割算法, 同时满足下面的约束条件:

$$\min \sum_{i=1}^{i=|G_i|} \mathrm{CP}(h_i) \leqslant \sum_{v_i \in G_i} \mathrm{Load}(v_i) \leqslant \max \sum_{i=1}^{i=|G_i|} \mathrm{CP}(h_i). \tag{7}$$

这里 G_i 表示第 i 个子任务图, $\mathrm{Load}(v_i)$ 表示任务 v_i 的负载, $\mathrm{CP}(h_i)$ 表示节点主机的处理能力. 这样经过反复的迭代, 最终会将原任务图分解为多个子任务图.

图 5 是图 4 所表示任务图的分割结果, 将原先的任务图分成了 3 个子任务群, 黑色的粗箭头表示子任务群之间的前驱约束. 注意, 经过这种分割后, 这些前驱约束并不形成任何环. 如果分割的结果不形成环, 则这个结果是可行的. 实际上, 下面的引理将证明 Blevel 分割算法总是能够生成可行解.

引理1　Blevel 分割算法生成结果的前驱约束不会形成任何环.

证明　假设 $(S_1, S_2, \ldots, S_M)$ 是 Blevel 分割算法生成的子任务图, 且 $v_i \in S_i$, $v_j \in S_j$, $i < j$. 因为 Blevel 分割算法是按顺序进行分类并分割任务图的, 显然, $\mathrm{Blevel}(v_i) \geqslant \mathrm{Blevel}(v_j)$, 根据前面 Blevel 的定义, v_i 不可能是 v_j 的后继节点, S_i 中的任务将不依赖 v_j 中的任务, 问题得证.

经过分割后, 所有的子任务群将用遗传算法来调度. 每一个子任务图的完成时间将传输给它的下一级子任务图, 这个时间作为相应的节点的准备时间, 也就是说, 除了第 1 个子任务图, 所有的节点都能在不同时间启动来处理任务. 这个传输步骤是 FPGA 算法的关键一步, 它使得最终的任务调度更紧凑.

4.2　算法的实现

下面将阐述 DAG 区块链环境中的非独立任务调度算法 FPGA 的实现.

(1) 编码. 为了保持调度的合理性, 则必须满足下面两个条件 [26]:

(i) 当一个任务可以开始执行之前, 它们的前驱任务集必须已经被处理完;

(ii) 任务图中的所有任务必须且仅仅被处理一次.

调度算法编码表示如图 6 所示. 这里序列对 (v_i, n_i) 表示任务 v_{t_i} 将被节点 N_{n_i} 处理, 显然, 在序列部分中的任务是有顺序的. 在其他的研究中, 染色体只是代表每一个节点上的所处理的任务, 并不表示任务调度的顺序, 但是本文的编码是有顺序的, 这个顺序表示任务的启动顺序, 如果 v_{t_i} 的启动时间小于 $v_{t_{i+1}}$ 的启动时间则被分配到同一个节点上, 如果 v_{t_i} 的启动时间等于 $v_{t_{i+1}}$ 的启动时间, 则被分配到不同的节点执行, 图 7 是图 6 所示任务图的一个可行的调度结果.

v_0	v_2	v_4	v_1	v_3	v_{11}	v_9	v_7	v_5	v_{10}	v_8	v_6
3	1	3	2	2	3	1	2	3	1	2	3

图 6 编码

Figure 6 Coding

Task	t_1	t_2	...	t_n	← Task sequence
Node	n_1	n_2	...	n_n	← Node sequence

图 7 一个可行的任务 (图 6 所示) 调度结果

Figure 7 Scheduling result for a viable task (shown in Figure 6)

选择编码方案需要考虑的一个重要的因素是, 可行解空间的所有可能调度都要唯一表示, 这样可以简化遗传算法操作的设计.

(2) 初始种群. 初始种群中的每一个个体是通过随机表调度产生的, 对于每一次迭代, 将被调度的任务是由下面两个原则决定的 [27~29]:

(i) 随机选择一个就绪任务, 该任务的所有前驱已经被调度;

(ii) 随机分配该任务到一个节点上.

由于节点是随机选择的, 原则 (i) 可以保证总是生成一个可行的调度, 原则 (ii) 可以保证任务在节点上是均衡分布的.

(3) 适应函数. 任务调度的目标是最小化任务的处理时间 (Makespan), 适应度函数定义如下:

$$\mathrm{fit}(i)=\frac{a+(\mathrm{maxcost}-\mathrm{cost}(j))^k}{b+\sum_{j=1}^{j=N_p}(\mathrm{maxcost}-\mathrm{cost}(j))^k}. \tag{8}$$

这里 N_p 为染色体的长度, maxcost 为群体中个体最大的调度长度, cost(j) 为第 j ($1\leqslant j\leqslant N_p$) 个染色体的调度长度, 优化的方向是使适应度值最大, a, b, k 为适应度定标因子, 用以平衡迭代过程中个体收敛和个体的多样性之间的关系.

(4) 选择. 选择是按有偏轮盘赌的原则来进行的, 如果一个个体的适应度越高, 那么它被选中的概率越高.

(5) 杂交. 杂交的输入是两个父代个体, 通过杂交, 输出是两个新的子个体, 这样子代保持了一些父代的特征. 设 s_1 和 s_2 是父代个体, s_1' 和 s_2' 是它们杂交后产生的子代个体. 这里将通过两步来实现杂交操作, 第 1 步是经典的 one-point 杂交. 如图 8 所示, s_1' 和 s_2' 是遵循下面的原则产生的:

(i) 直接保持 s_1 和 s_2 的序列部分遗传给 s_1' 和 s_2';

(ii) 随机选择杂交点分割 s_1 和 s_2 的分配部分;

(iii) 交换 s_1 和 s_2 的分配部分, 改变杂交点.

第 2 步, s_1' 和 s_2' 是遵循下面的原则产生的, 如图 9 所示:

(i) 直接保持 s_1 和 s_2 的序列部分遗传给 s_1' 和 s_2';

(ii) 随机选择任务集;

(iii) 交换 s_1 和 s_2 中所选择任务的分配部分.

注意, 这里的两个杂交机制都没有改变任务序列部分, 因此, 所产生的解总是可行的, 这样可以不必设计检查和纠正算法, 从而简化遗传算法的复杂性.

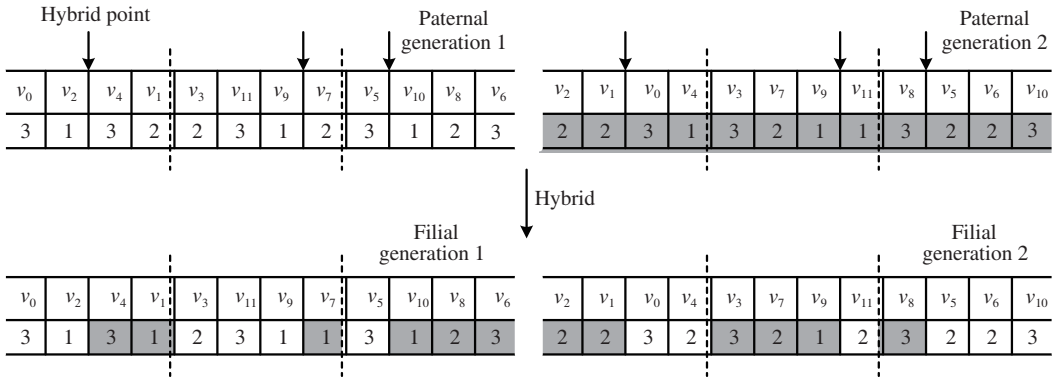

图 8　杂交 1

Figure 8　Hybrid 1

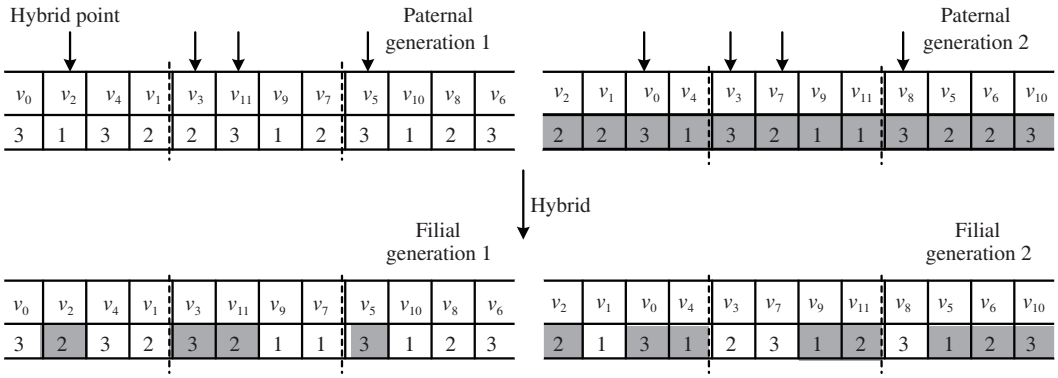

图 9　杂交 2

Figure 9　Hybrid 2

(6) 变异. 变异确保找到最优解的概率永远不为 0, 经过选择和杂交可能丢失一些好的遗传物质, 变异可以恢复这些遗传物质. FPGA 算法实现两个变异操作, 第 1 个是随机选择两个任务并且交换它们的分配部分, 第 2 个是选择一个任务随机改变它的分配部分, 这些操作生成的解也总是可行的.

(7) FPGA 算法的实现. 因为 Blevel 分割算法总是生成可行的分割结果, 子任务图能够按顺序合成全局调度. 然而, 当前所维持的 Makespan 是不精确的. 因此, 完成所有子任务图调度后, 需要一个附加的 Conquer 算法, 合成所有的局部调度, 来重新计算整体的 Makespan, 求解 DAG 区块链中的非独立任务调度问题的 FPGA 算法的伪代码如算法 1 所示.

4.3　算法的复杂性分析

该算法主要包括分割算法、遗传算法、和 Conquer 算法 3 个子算法, 设任务数为 n, 子任务图数为 m, 分割算法的复杂度为 $O(m \times \frac{n}{m})$, 同理, Conquer 算法的复杂度也为 $O(n)$, 遗传算法中, 对每一染色体解码的复杂度为 $O(\frac{n}{m}\log\frac{n}{m})$, 设 C_{select}, C_{hybrid}, C_{mutate} 分别表示 FPGA 算法中选择算子复杂度、杂交算子复杂度、变异算子复杂度, 这里 C_{select}, C_{hybrid}, C_{mutate} 的复杂度均以 $O(\frac{n}{m})$ 为界, 并且 $C_{\text{select}} = O(P)$. 这里 P 表示种群规模. 所以, 给定种群规模 P 和最大进化代数 max$X$, 该算法的计算

Algorithm 1 FPGA

Input: Task information of DAG. Node parameters and network information.

Begin
  Reading task information of DAG; //Initialization
  Call partition algorithm( );
  begin
    Computing the values of tasks' Blevel;
    Ordering all the task by Blevel value descending;
    Deciding the number of the sub-DAG;
    Partition the DAG random;
  end
  For$(i = 1, i++, i > m)$ //m is the number of subtask graphs
  do
  Call genetic algorithm( );
  begin
    Coding the tasks and processors;
    Initialing the population by random method;
    Repeat
    Select the individual by biased roulette;
    Cross the chromosome of the individuals by one-point crossover;
    Cross the chromosome of the individuals by parts;
    Mutation the chromosome;
    Computing the value of fitness function;
    Evaluation the present individuals;
    Replacement the former individuals by better offspring;
    Until
    Individual do not change or arriving at enactment times;
  end
  Call conquer algorithm( );
  Put out the outcome;
End

Output: Matching results of the tasks and nodes.

复杂度为

$$O\left(2n \times P \times \max X \times O\left(\log \frac{n}{m}\right)\right) + (P \times \max X + 2)O(n). \tag{9}$$

从上式可以看出 FPGA 算法的复杂度与一般的遗传算法的复杂度相当, 但必须说明的是, 在 FPGA 算法中, 由于采用了分割算法, 分割后的子任务图可以并行调度, 因此, FPGA 算法将比一般的遗传算法的运行时间大大降低.

4.4 算法的仿真实验结果及分析

本文针对 FPGA 算法构造了一个仿真实验系统, 这个系统包含 5 个相对独立的部分, 如图 10 所示, 第 1 部分是任务生成器, 根据所输入的任务数量, 它将随机生成任务图. 第 2 部分是基于确定性退火技术的融合分割遗传任务调度算法, 主要包括分割算法、遗传调度算法、Conquer 算法, 分割算法利用确定性退火技术, 将整个任务图分割为子任务图, 遗传算法调度所有的子任务图, 从而生成局部调度, Conquer 算法合成所有的局部调度, 并且重新计算整体的 Makespan. 第 3 部分是 DAG 区块链拓扑生成器, 模拟生成 DAG 区块链网络环境, 包括通信带宽、CPU 计算能力、存储容量等. 第 4 部

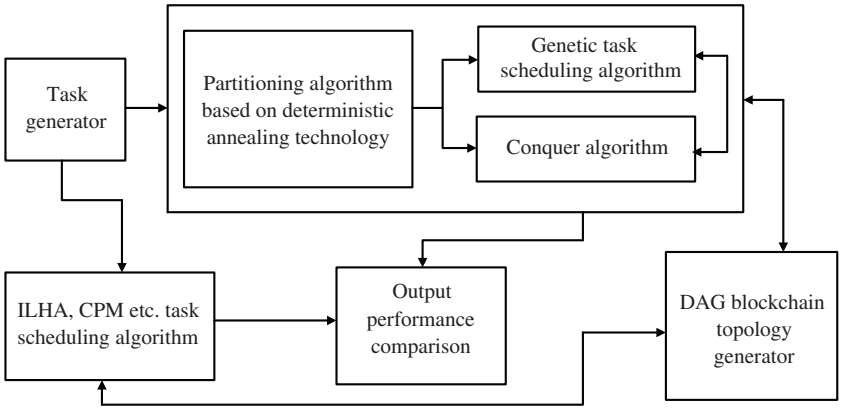

图 10　仿真实验系统

Figure 10　Simulation experimental system

分是 ILHA, CPM 等任务调度算法, 功能是实现 ILHA, CPM 等任务调度, 作为比较的对象. 第 5 部分是输出任务调度结果, 并对本文所提出的任务调度算法和其他算法进行比较. 仿真实验基于 Amazon Lightsail 云计算平台, 利用网络拓扑生成器 BRITE 的 Java 开源版本以及离散事件驱动模拟程序包 Simjava, 生成 10000 个虚拟节点, 节点计算能力和通信能力随机生成, CPU 主频范围为 [512 M∼2 GB], 内存范围为 [1 GB∼2 GB], 带宽范围为 [512 kbps∼5 Mbps], 为尽量接近现实的网络拓扑结构, 节点物理拓扑图按照 GLP (generalized linear preference) 生成算法 [30, 31] 在 BRITE 中以路由器级生成, 忽略了实际网络传输过程中的延迟、拥塞、丢包等细节问题.

第 1 个实验, 为了测试子任务图和任务数量对 FPGA 算法的影响, 这里生成了包含 20000∼100000 个任务, 它们被分成 1∼1000 个子任务图, 将被分配到模拟生成的 DAG 区块链节点上进行处理. 图 11 展示了对于不同子任务图数量 FPGA 算法的性能 (标准化的 Makespan: 任务的处理时间), 这里标准化的性能是 FPGA 算法的 Makespan 与一般遗传算法的比值, 当 PAG 算法只具有 1 个子任务图时, FPAG 算法和一般的遗传算法性能相同. 从图中可以看出, 标准化的 Makespan 始终在 0.6∼1 之间, 这表明 FPGA 算法的调度性能与一般的遗传算法相似, 但是随着子任务数量的增加, FPGA 算法的性能逐渐提高, 当子任务数量较大时, FPGA 算法的性能是明显优于一般遗传算法的, 最大可以提高约 40% 的性能. 图 12 所展示的是不同数量子任务图和任务数的 FPGA 算法的调度时间, 如图所示, FPGA 算法可以显著地降低调度时间, 调度时间和子任务图数量的关系并不是线性的, 子任务图数量越小, 调度时间降低的越明显. FPGA 的这些特性说明它非常适合于 DAG 区块链环境中大规模的任务调度问题.

第 2 个实验, 测试 FPGA 算法的调度长度, 并与 ILHA 和 CPM 算法进行比较. 这里采用相对调度长度来比较, 定义如下:

$$\mathrm{RSL}(S) = \frac{\mathrm{SL(FPGA)}}{\mathrm{SL}(S)}. \tag{10}$$

这里 RSL(S) 表示调度算法 S 的相对调度长度, SL(FPGA) 表示 FPGA 算法的调度长度, SL(S) 表示算法 S 的调度长度. 实验结果如图 13 所示, 随着任务数量的增加, FPGA 算法的调度长度始终低于 CPM 和 ILHA 算法.

第 3 个实验, 测试 FPGA 算法的负载平衡情况, 并与 ILHA 和 CPM 算法进行比较. 这里采用负

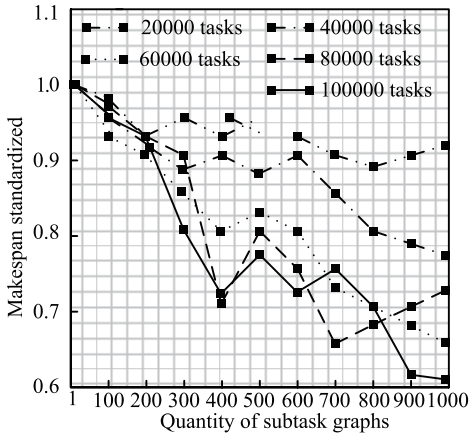

图 11 不同数量子任务图和任务数的 FPGA 算法处理时间

Figure 11 The makespan of FPGA algorithm with different number of subtask graphs and task numbers

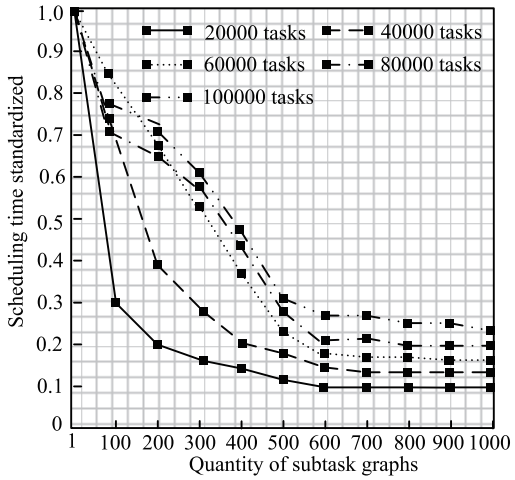

图 12 不同数量子任务图和任务数的 FPGA 算法调度时间

Figure 12 The scheduling time of FPGA algorithm with different number of subtask graph and task number

载平衡方差来比较, 定义如下:

$$\mathrm{WB} = \sum_{i=1}^{m} (\mathrm{WB}_i - \mathrm{WB}_{\mathrm{avg}})^2, \tag{11}$$

其中 WB_i 为第 i 个节点的负载率, $\mathrm{WB}_{\mathrm{avg}}$ 为系统的平均负载率. 实验结果如图 14 所示, 可以看出 FPGA 算法的负载平衡方差始终低于 ILHA 和 CPM 算法, 表明 FPGA 算法在优化任务的完成时间的同时, 兼顾了负载均衡问题, 有效地提高了资源的利用率.

5 结论

本文针对传统的链式的区块链响应速度慢的问题, 提出了一种 DAG 区块链理论架构, 将传统区块链的链式处理过程, 变为并行的处理过程, 使快速响应成为了可能. 在此基础上, 根据 DAG 区块链异构、动态、广域分布的特性, 面向非独立任务的调度问题, 提出了基于确定性退火技术的非独立任

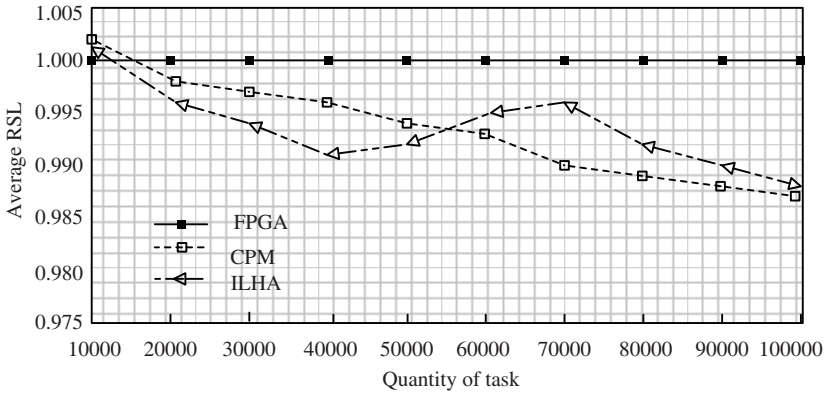

图 13　不同任务数量下 FPGA 算法的效率

Figure 13　Efficiency of FPGA algorithm under different number of tasks

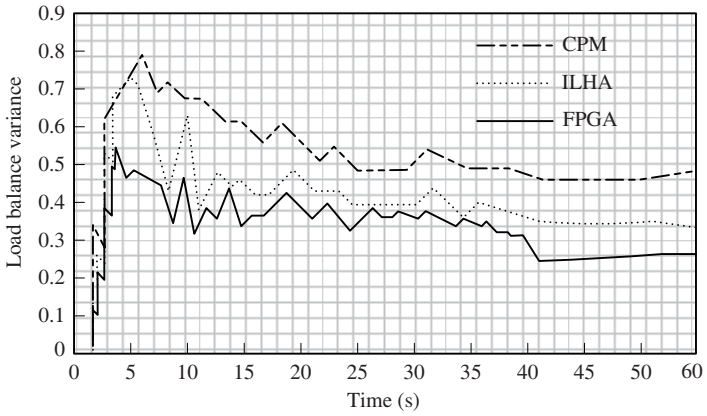

图 14　ILHA, CPM, FPGA 算法的负载平衡比较

Figure 14　The load balance comparison of ILHA, CPM, FPGA algorithm

务调度的混合分割遗传算法 FPGA. 实验结果显示, FPGA 算法能够适应 DAG 区块链节点的异质性和动态性, 其调度的性能也比传统的调度算法有所改善, 在优化任务完成时间的同时, 兼顾了负载均衡问题, 有效地提高了资源的利用率, 是解决 DAG 区块链环境中的非独立任务调度问题的可行方法. 未来需要进一步研究 DAG 区块链的安全问题, 从而在实际环境中进行部署测试.

参考文献

1　Ju C H, Zou J B, Fu X K. Design and application of big data credit reporting platform integrating blockchain technology. Comput Sci, 2018, 45: 522–526 [琚春华, 邹江波, 傅小康. 融入区块链技术的大数据征信平台的设计与应用研究. 计算机科学, 2018, 45: 522–526]

2　He P, Yu G, Zhang Y F, et al. Survey on blockchain technology and its application prospect. Comput Sci, 2017, 44: 1–7 [何蒲, 于戈, 张岩峰, 等. 区块链技术与应用前瞻综述. 计算机科学, 2017, 44: 1–7]

3　Chen W L, Zheng Z B. Blockchain data analysis: a review of status, trends and challenges. J Comput Res Dev, 2018, 55: 1853–1870 [陈伟利, 郑子彬. 区块链数据分析: 现状、趋势与挑战. 计算机研究与发展, 2018, 55: 1853–1870]

4　Yuan Y, Wang F Y. Parallel blockchain: concept, methods and issues. Acta Autom Sin, 2017, 43: 1703–1712 [袁勇, 王飞跃. 平行区块链: 概念、方法与内涵解析. 自动化学报, 2017, 43: 1703–1712]

5 Yuan Y, Wang F Y. Blockchain and cryptocurrencies: model, techniques, and applications. IEEE Trans Syst Man Cybern Syst, 2018, 48: 1421–1428

6 Novo O. Blockchain meets IoT: an architecture for scalable access management in IoT. IEEE Int Things J, 2018, 5: 1184–1195

7 Makhdoom I, Abolhasan M, Abbas H, et al. Blockchain's adoption in IoT: the challenges, and a way forward. J Netw Comput Appl, 2019, 125: 251–279

8 Andoni M, Robu V, Flynn D, et al. Blockchain technology in the energy sector: a systematic review of challenges and opportunities. Renew Sustain Energy Rev, 2019, 100: 143–174

9 Yu H, Zhang Z Y, Liu J W. Research on scaling technology of bitcoin blockchain. J Comput Res Dev, 2017, 54: 2390–2403 [喻辉, 张宗洋, 刘建伟. 比特币区块链扩容技术研究. 计算机研究与发展, 2017, 54: 2390–2403]

10 Banerjee M, Lee J, Choo K K R. A blockchain future for internet of things security: a position paper. Digit Commun Netw, 2018, 4: 149–160

11 Pan C, Liu Z Q, Liu Z, et al. Research on scalability of blockchain technology: problems and methods. J Comput Res Dev, 2018, 55: 2099–2110 [潘晨, 刘志强, 刘振, 等. 区块链可扩展性研究: 问题与方法. 计算机研究与发展, 2018, 55: 2099–2110]

12 Liu A D, Du X H, Wang N, et al. Research progress of blockchain technology and its application in information security. J Softw, 2018, 29: 2092–2115 [刘敖迪, 杜学绘, 王娜, 等. 区块链技术及其在信息安全领域的研究进展. 软件学报, 2018, 29: 2092–2115]

13 Shao Q F, Jin C Q, Zhang Z, et al. Blockchain: architecture and research progress. Chinese J Comput, 2018, 41: 969–988 [邵奇峰, 金澈清, 张召, 等. 区块链技术: 架构及进展. 计算机学报, 2018, 41: 969–988]

14 Zheng Z, Xie S, Dai H, et al. An overview of blockchain technology: architecture, consensus, and future trends. In: Proceedings of IEEE International Congress on Big Data, 2017. 557–564

15 Zhang F, Shi B X, Jiang W B. Review of key technology and its application of blockchain. Chinese J Netw Inform Secrity, 2018, 4: 22–29 [章峰, 史博轩, 蒋文保. 区块链关键技术及应用研究综述. 网络与信息安全学报, 2018, 4: 22–29]

16 Jawhar I, Mohamed N, Al-Jaroodi J, et al. Communication and networking of UAV-based systems: classification and associated architectures. J Netw Comput Appl, 2017, 84: 93–108

17 Li Z, Barenji A V, Huang G Q. Toward a blockchain cloud manufacturing system as a peer to peer distributed network platform. Robot Comput-Integr Manuf, 2018, 54: 133–144

18 Kotilevets I D, Ivanova I A, Romanov I O, et al. Implementation of directed acyclic graph in blockchain network to improve security and speed of transactions. IFAC-Papersonline, 2018, 51: 693–696

19 Munsing E, Mather J, Moura S. Blockchains for decentralized optimization of energy resources in microgrid networks. In: Proceedings of IEEE Conference on Control Technology and Applications, 2017. 108–122

20 Cao H H, Yu Z W, Wang Y Y. Research on grid task scheduling algorithm based on communication and computing cost. Comput Eng Appls, 2006, 42: 132–134 [曹怀虎, 余镇危, 王银燕. 基于通信和计算开销的网格任务调度算法的研究. 计算机工程与应用, 2006, 42: 132–134]

21 Topcuoglu H, Hariri S, Wu M Y. Performance-effective and low-complexity task scheduling for heterogeneous computing. IEEE Trans Parallel Distrib Syst, 2002, 13: 260–274

22 Zhou X G, Liang L, Huang X Z, et al. On-line scheduling and placement of real-time tasks for recongfigurable computing system. Chinese J Comput, 2007, 30: 1903–1909 [周学功, 梁楑, 黄勋章, 等. 可重构系统中的实时任务在线调度与放置算法, 计算机学报, 2007, 30: 1903–1909]

23 He X, Huang T W, Yu J Z, et al. A continuous-time algorithm for distributed optimization based on multiagent networks. IEEE Trans Syst Man Cybern Syst, 2017. doi: 10.1109/TSMC.2017.2780194

24 Guggilam S S, Dall'Anese E, Chen Y C, et al. Scalable optimization methods for distribution networks with high PV integration. IEEE Trans Smart Grid, 2016, 7: 2061–2070

25 Yang G W, Li X M, Wang Y H, et al. Deterministic annealing. Chinese J Comput, 1998, 21: 765–768 [杨广文, 李晓明, 王义和, 等. 确定性退火技术. 计算机学报, 1998, 21: 765–768]

26 Fedak G, Germain C, Neri V, et al. Xtrem web: a generic global computing system. In: Proceedings of the 1st IEEE/ACM International Symposium on Cluster Computing and the Grid, 2001. 452–460

27 Li P, Duan H B. A potential game approach to multiple UAV cooperative search and surveillance. Aerospace Sci Tech,

2017, 68: 403–415
28 Sedjelmaci H, Senouci S M, Ansari N. Intrusion detection and ejection framework against lethal attacks in UAV-aided networks: a Bayesian game-theoretic methodology. IEEE Trans Intell Transp Syst, 2017, 18: 1143–1153
29 Hussein A F, ArunKumar N, Ramirez-Gonzalez G, et al. A medical records managing and securing blockchain based system supported by a genetic algorithm and discrete wavelet transform. Cogn Syst Res, 2018, 52: 1–11
30 Bu T, Towsley D. On distinguishing between Internet power law topology generators. In: Proceedings of the 21st Annual Joint Conference of IEEE Computer and Communications Societies, 2002. 638–640
31 Cao H H, Zhu J M, Pan Y, et al. Context-aware P2P mobile social network structure and discovery algorithm. Chin J Comput, 2012, 35: 1223–1234 [曹怀虎, 朱建明, 潘耘, 等. 情景感知的 P2P 移动社交网络构造及发现算法. 计算机学报, 2012, 35: 1223–1234]

Fusion-partitioning genetic task scheduling algorithm based on deterministic annealing technology in DAG blockchains

Huaihu CAO*, Yanmei ZHANG, Jian WANG, Haifeng LI & Lixin CUI

School of Information, Central University of Finance and Economics, Beijing 100081*, China*
* Corresponding author. E-mail: chh@cufe.edu.cn

Abstract The traditional blockchain structure cannot adapt to large-scale and real-time-application scenarios because of its inherently slow response. To solve this problem, a theoretical framework of DAG (directed acyclic graph) blockchain is proposed, transforming the chain processing of a traditional blockchain into parallel processing. On this basis, the non-independent task-scheduling problem in the DAG blockchain environment is studied, and a fusion-partitioning genetic task-scheduling algorithm based on deterministic annealing technology in DAG blockchains is proposed. The experimental results show that the algorithm can adapt to the heterogeneity, dynamism, and wide area of DAG blockchain nodes, and its scheduling performance is better than that of the traditional scheduling algorithm. While optimizing the task-completion time, the algorithm takes account of the load-balancing problem and effectively improves the response speed. It is a feasible method for solving the non-independent task-scheduling problem in the DAG blockchain environment.

Keywords directed graph, genetic algorithms, parallel, task-scheduling, optimization, load balancing

Huaihu CAO was born in 1977. He received a Ph.D. in communication and information systems from China University of Mining and Technology, Beijing, in 2006. Currently, he is a professor at Central University of Finance and Economics. His research interests include network computing, social computing, and network economics.

Yanmei ZHANG was born in 1976. She received her Ph.D. in communication and information systems from China University of Mining and Technology, Beijing, in 2010. Currently, she is an associate professor at Central University of Finance and Economics. Her research interests include personalized recommendation algorithms, service computing, and data mining.

Jian WANG was born in 1975. He received his Ph.D. in signal and information processing from Beijing University of Posts and Telecommunications in 2007. Currently, he is an associate professor at Central University of Finance and Economics. His research interests include machine learning, cloud computing, and intelligent education.

Haifeng LI was born in 1979. He received his Ph.D. in computer science from Renmin University of China, Beijing, in 2009. Currently, he is an associated professor at Central University of Finance and Economics. His research interests include data mining and machine learning.