Operations Research and Fuzziology 运筹与模糊学, 2024, 14(3), 969-978
Published Online June 2024 in Hans. https://www.hanspub.org/journal/orf
https://doi.org/10.12677/orf.2024.143332

Hans汉斯

蚁群算法的大数据计算引擎均衡部署数学建模研究

肖丽丽

南京邮电大学理学院，江苏 南京

收稿日期：2024年4月23日；录用日期：2024年6月21日；发布日期：2024年6月28日

摘　要

随着大数据技术的迅猛发展，大数据计算引擎的性能优化和均衡部署问题成为研究热点。蚂蚁算法是一种模仿自然界中蚂蚁觅食的方法，它具有良好的全局寻优能力和较强的鲁棒性，非常适合求解复杂的优化问题。在此基础上，提出了一种基于蚁群算法的大规模数据挖掘方法，研究面向大数据计算引擎平衡配置问题的蚂蚁算法，通过建立数学模型，对计算资源进行优化配置。本项目的研究成果将为大数据处理引擎的平衡部署提供新的研究思路与方法。

关键词

蚁群算法，大数据，计算引擎，均衡部署模型

Research on Mathematical Modeling of Balanced Deployment of Ant Colony Algorithm Big Data Computing Engine

Lili Xiao

College of Science, Nanjing University of Posts and Telecommunications, Nanjing Jiangsu

Received: Apr. 23rd, 2024; accepted: Jun. 21st, 2024; published: Jun. 28th, 2024

Abstract

With the rapid development of big data technology, the performance optimization and balanced deployment of big data computing engine have become a research hotspot. Ant algorithm is a method that mimics ants foraging in nature. It has good global optimization ability and strong ro-

文章引用: 肖丽丽. 蚁群算法的大数据计算引擎均衡部署数学建模研究[J]. 运筹与模糊学, 2024, 14(3): 969-978.
DOI: 10.12677/orf.2024.143332

bustness, which is very suitable for solving complex optimization problems. Based on this, we propose a large-scale data mining method based on ant colony algorithm, studies the ant algorithm for the balanced configuration problem of big data computing engine, and optimizes the allocation of computing resources by establishing a mathematical model. The research results of this project will provide new research ideas and methods for the balanced deployment of the big data processing engine.

Keywords

Ant Colony Algorithm, Big Data, Computational Engine, Balanced Deployment Model

1. 引言

随着信息技术的迅猛发展，大数据已经成为当今时代的重要标志。在这个信息化浪潮中，数据成为驱动社会进步的重要引擎，而大数据计算引擎则成为处理这些海量数据的核心工具。然而，随着数据量的不断攀升，如何高效、稳定地处理这些数据，成为摆在我们面前的一大挑战。大数据计算引擎的性能优化和均衡部署，是解决这一挑战的关键所在。优化计算引擎的性能，能够提高数据处理的效率，降低成本，为企业创造更大的价值。而均衡部署则是确保计算引擎稳定运行的重要手段，通过合理分配计算资源，避免资源瓶颈和过载现象的发生，从而保障数据处理的质量和稳定性。

文献[1]基于贪心启发式算法建立计算引擎均衡部署框架，定义数据传输机制，利用限定范围的可控参数约束传输过程，结合二次分拆原理，制定分拆计划，通过贪心启发式算法实现分拆结果的均衡部署。文献[2]设计了一种适用于大数据计算的随机样本划分模型[3] [4] [5]。将大数据表示成数据块集合形式，储存在不同节点上；估计大数据的特征，建立回归分析模型，利用 Alpha 计算架构实现数据清洗，减轻计算量，再通过概率密度函数将计算引擎部署到最佳数据块上[6] [7] [8] [9]，因此，本文旨在研究蚁群算法在大数据计算引擎均衡部署中的应用，通过数学建模实现计算资源的合理分配，提高计算引擎的性能和稳定性。

2. 蚁群算法的基本概念

2.1. 蚁群算法的基本原理

蚁群算法的基本原理主要基于以下几个方面：

1) 蚂蚁为了觅食，会沿其行进路线散发一种叫费洛蒙的化学物质。这些激素会随着时间慢慢消散，但是新来的蚂蚁仍然可以在行进中感受到[10]。更浓郁的信息素，会吸引更多的蚂蚁，从而更有可能选择这条路线。

2) 蚂蚁在选择路径时并非完全随机，而是会受到信息素浓度和启发式信息的双重影响。启发式信息通常与问题的具体特征有关，例如距离、时间等[11]。蚂蚁在综合考虑这两种信息的基础上，选择下一个移动的目标点。

3) 蚁群算法还引入了正反馈机制。即蚂蚁在找到食物源或目标点后，会释放更多的信息素以加强该路径的信息素浓度[12]。这种正反馈机制使得越来越多的蚂蚁选择信息素浓度高的路径，从而形成一种

“优胜劣汰”的效应。

4) 通过多次迭代和信息素的更新，蚁群算法能够逐渐找到从起点到终点的最优路径[13]。这种路径不仅是最短的，而且可能是满足特定约束条件的最优解。

2.2. 该算法的优缺点

蚁群算法的优点主要体现在以下几个方面：

1) 全局搜索能力强：蚁群算法通过模拟蚂蚁觅食行为，能够在解空间中进行广泛的搜索，从而找到全局最优解。这种全局搜索能力使得算法在处理复杂优化问题时具有优势。

2) 鲁棒性好：蚁群算法对初始参数的设置不敏感，能够在不同的环境下保持较好的性能。这使得算法在实际应用中具有更强的适应性和稳定性。

3) 易于与其他算法结合：蚁群算法可以与其他优化算法相结合，形成混合优化算法，以进一步提高求解效果。这种灵活性使得算法在实际应用中具有更广泛的应用前景。

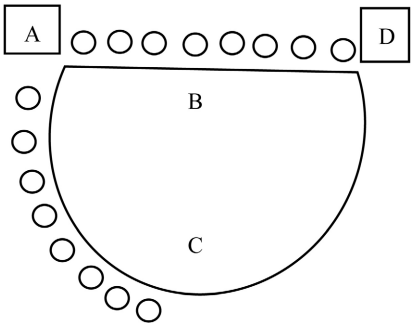
然而，蚁群算法也存在一些缺点和不足：

1) 收敛速度较慢：由于蚁群算法需要进行多次迭代和信息素更新才能找到最优解，因此其收敛速度相对较慢。这在处理大规模或实时性要求较高的优化问题时可能成为一个限制因素。

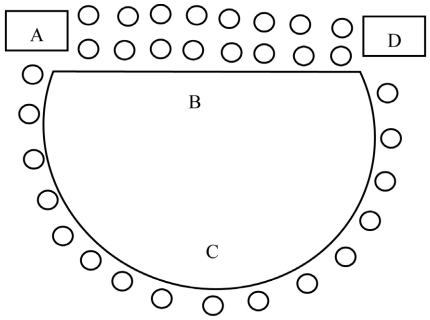
2) 计算资源需求大：蚁群算法在求解过程中需要模拟大量蚂蚁的行为，并进行复杂的信息素更新操作，这可能导致算法的计算复杂度较高，对计算资源的需求较大。

3) 参数设置困难：蚁群算法中的参数设置对算法的性能具有重要影响。然而，如何合理地设置这些参数往往是一个难题，需要根据具体问题进行调试和优化。

2.3. 蚁群算法



(a) 蚂蚁移动 8 个时间路径信息素情况



(b) 蚂蚁移动 16 个时间路径信息素情况

Figure 1. The process of ant colony searching for food
图 1. 蚁群搜索食物过程

经过许多仿生学专家长时间的观察发现，在自然界，多数蚂蚁没有视觉系统但却可以寻觅食物，说明蚂蚁内部的某种机制使他们具有了群体智能[14] [15] [16]。经研究表明蚂蚁在搜索食物的路径上会自动释放出一种特殊激素(Pheromone)的分泌物，蚁群之间能够感知这种分泌物来进行信息传递，正是这种特殊的激素分泌物实现了群体行为，该物质被称作“信息素”[17] [18]。

蚁群算法中，蚂蚁离巢觅食的过程可概括如下：开始时，蚂蚁随机选择路径，在路线上释放信息素，量取决于路线长度。蚂蚁感知自身气味和浓度，决定下一步行动，其他蚂蚁更倾向于寻找浓度高的路径[19]。蚁群行为呈正向反馈，选择较短路径的蚂蚁和信息素浓度增多，吸引更多蚂蚁选择该路径，最终形成最短路径[20]。蚂蚁能快速感知环境变化，发现障碍后重新寻找路径。单一蚂蚁具有寻优能力，但通过信息素传递和集体协作，蚁群找到最佳路径。

若蚂蚁从洞穴点 A 出发到食物点 D，已知有两条路可以走 ABD 和 ACD，蚂蚁可以随机选择一条路，详见图 1，蚂蚁的移动速度和分泌的信息素量假设相同。在开始的时候，将一只蚂蚁放在这两条道路上，让它们在单位时间内移动一小步，那么 8 个时间单元之后，情况如下：路径 ACB 的蚂蚁抵达了食料点 B，而路径 ADB 的蚂蚁一直走到了 C，但还没有抵达。又过了八个小时，情况如下：从食料点回来的蚂蚁到达了食料点 A，而沿着 ACD 路线的蚂蚁刚刚到达食料点 D。因此，可以推断出，在 16 个不同的时间单元后，从山洞到食料点来回跑了两次，而沿着 ACD 路线的蚂蚁却只跑了一次，这两条路线的信息素累积量是 2:1。

3. 大数据计算引擎工作模式与流程分析

3.1. 计算模式架构

在大数据处理的广阔领域中，计算模式架构的选择是确保数据处理效率与稳定性的关键。随着数据量的不断攀升，简单的数据处理方式已无法满足日益增长的需求，特别是在处理速度和系统响应方面。因此，计算内存和处理能力的增强成为必然要求。

目前，由图 2 所知，大数据的处理方式有两种，一种是流式处理，另一种是批处理。基于 Spark 的流式计算技术是一种新型的分布式计算技术，它是一种新型的分布式计算技术。在该模型中，多个引擎节点并行工作，从多个不同来源采集数据，各自独立地进行运算，并实时存储在数据库中。为了确保系统的高效稳定运行，引擎部署的负载均衡至关重要。另一方面，批量式计算遵循分治策略，将大数据集划分为多个子集，并分配到不同的计算节点上进行处理。这种方式减少了单个节点的计算压力，但通常需要多次迭代才能达到理想的结果。尽管处理速度可能较慢，但其计算结果的精度较高，因此在某些对时间要求不高的场景中得到了广泛应用。鉴于流式计算和批量式计算各自的优缺点，本文提出了一种双模式大数据计算架构。该架构融合了两种计算模式的优势，旨在实现更高效、更稳定的大数据处理。

3.2. 计算流程

在此基础上，提出了一种基于双模计算引擎体系结构的计算方法。主要内容有：1) 获取阶段：从多个外部数据来源获取实时数据，并对其格式进行标准化，为后续处理提供必要的素材。2) 数据预处理：采用批处理的方式，从海量的数据中抽取出关联模式，为流式计算的进一步发展奠定坚实的基础。这一阶段所要处理的数据量大，结构复杂。3) 运算部分：利用分布引擎实现对各部分的预处理过程的实时运算。在此过程中，由于数据的突发、无序等特点，使得计算引擎需要更高的性能以及更高的负载平衡能力。4) 操作阶段：依据运算结果，为使用者提供决策依据，并将数据转换为有价值的数据。在此基础上，通过以上研究，实现高效稳定的大数据计算。但是，在实际应用中，会出现诸如任务分布不均匀等问题，从而造成计算引擎负载失衡的现象。为此，本项目提出了一种基于模型的引擎配置方法，来均衡任务的

分布，从而提升算法的运行效率。这也是本项目的主要研究内容。

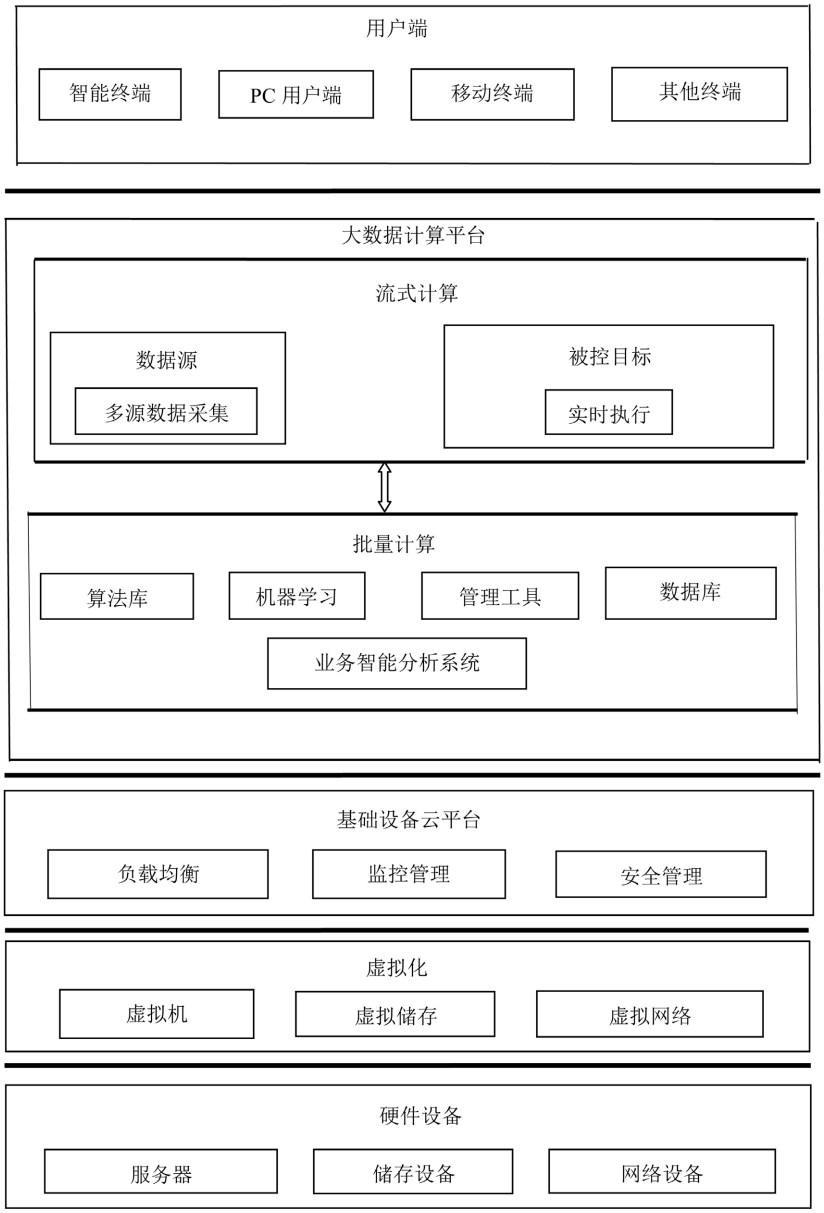


Figure 2. Architecture diagram of big data computing mode
图 2. 大数据计算模式架构图

4. 大数据计算引擎均衡部署数学建模

4.1. 均衡部署约束模型

4.1.1. 计算任务数量约束

在这样的计算环境中，由于分布引擎的硬件结构不同，造成了计算任务量的不均匀性。假定引导节点的集合可以被表达成 $N=\left\{n_1,n_2,\cdots,n_{|N|}\right\}$，其中任何一个节点中所含有的数据总数 $R=\left\{r_1,r_2,\cdots,r_{n_{|N|}}\right\}$，那么在这个拓扑结构中出现的所有计算任务的总数就被记为：

$$R=\sum_{i=1}^{N} r_{n_i}$$

$$|T|=\sum_{i=1}^{|N|} T_{n_i}$$

$$\frac{|T_{n_i}|}{r_{n_i}}=K$$

$$\sum_{i=1}^{|N|}|T_{n_i}|=\sum_{i=1}^{|N|} k\cdot r_{n_i}=k\cdot\sum_{i=1}^{|N|} r_{n_i}=k\cdot R$$

$$K=\frac{|T|}{R}$$

在此基础上，以分配的公平分配为约束，并根据引擎的实际配置状况，采用手动方法确定了该引擎所能接受的最大任务数。对于引擎 n_i，它的计算能力是这样的：

$$\left|T_{n_i}^{\max}\right|=r_{n_i}\cdot k=r_{n_i}\cdot\frac{|T|}{R}$$

结果所有的引擎都是一样的，并且不存在异构特性，那么每一个引擎所拥有的资源都是一样的。在这一点上，可以表达为：

$$\left|T_{n_i}^{\max}\right|=r_{n_i}\cdot\frac{|T|}{R}=\frac{R}{|N|}\cdot\frac{|T|}{R}=\frac{|T|}{N}$$

如果在 t 时刻已经部署了该引擎 n_i 的计算任务数 $\left|T_{i_i}(t)\right|$ 为：

$$\left|T_{n_i}^{\text{remain}}(t)\right|=\left|T_{n_i}^{\max}\right|-\left|T_i(t)\right|$$

4.1.2. 负载均衡约束

如果 n_k 部署了计算任务 t_{ij}，那么，$f\left(t_{ij}\right)=n_k$ 的另外一个形式是 $f^{-1}\left(n_k\right)=t_{ij}$：如果在一个节点上部署 n_k 引擎，那么这个时候就记录为 $\left(T_{n_k}\right)=n_k$ 或 $f^{-1}\left(n_k\right)=\mathrm{T}_{\mathrm{n_k}}$。上面的 f 是展开规则。

假定在一个具有权重的拓扑结构中的计算任务集是 T，n_x，它代表了许多引擎 n_k 中的任意一种，然后将其部署到 $f^{-1}\left(n_k\right)$ 该任务中，然后将其部署到该任务 $f^{-1}\left(n_k\right)$ 集中。

$$T=\bigcup_{x=1}^{|N|} f^{-1}\left(n_x\right)$$

$$f^{-1}\left(n_k\right)\cap f^{-1}\left(n_l\right)=\varnothing\left(l\neq k\right)$$

假设 $O_{n_x}^{C}$ 是引擎 n_x 的 CPU 负荷为：

$$O_{n_x}^{C}=\sum_{f\left(t_{ij}\right)=n_x} O_{t_{ij}}^{C}$$

使用 O^C 代表了全部引擎的负荷之和，在同构情形中，每个引擎的负荷随着任务数目的不同而变化，但是 O^C 相同：

$$O^C=\sum_{x=1}^{|N|} O_{t_{ij}}^{C}$$

如果对每个引擎都均匀地分布载荷，那么每个引擎所需负担的载荷可以表达为：

$$\bar{O}^C=O^C/|N|$$

4.1.3. 最优部署开销约束

部署代价取决于流量的大小，随着流量的增加，需要更多的部署时间和更高的能量消耗。假定两个任务和的数据流量都可以用或来表示，为了确保最佳的部署代价，需要将要计算的数据流量的总数降到最小。

$$S_{\text{best}} = \min \sum_{f(t_{ij}) \neq f(t_{kl})} v_{ij,kl}$$

$$S_{\text{best}} = \min \sum_{f(t_{ij}) \neq f(t_{kl})} k_{kl,ij}$$

4.1.4. 基于蚁群算法的均衡部署数学建模

在计算任务数、负载平衡及成本优化的条件下，采用蚂蚁算法对发动机均衡配置进行了建模。蚂蚁算法首先需要对信息素函数 $T_{ij}(t)$ 进行初始化，它表示了引擎 n_x 对特定任务的信息素 T_i 浓度，并与引擎的计算能力 MIPS_j 和通信带宽 Bandwidth_j 相结合，从而实现初始化：

$$T_{ij}(o) = \frac{\text{MIPS}_j}{D} + \text{Bandwidth}_j$$

在完成了功能的初始化之后，可以使用下面的公式来获得向任务 T_i 服务中心部署引擎 n_x 的概率：

$$\rho_{ij}^{K}(t) = \begin{cases} \dfrac{\left[T_{in}(t)\right]^{\alpha}\left[\eta_{in}(t)\right]^{\beta}}{\sum_{x_{pq} \notin tabu_k}\left[T_{in}(t)\right]^{\alpha}\left[\eta_{in}(t)\right]^{\beta}}, & x_{pq} \notin tabu_k \\ 0, & \text{其他} \end{cases}$$

$$\eta_{in}(t) = \frac{1}{D}$$

当蚁群算法 $\rho_{ij}^{K}(t)$ 决定了路径之后，这时，引擎 n_x 就会在这个节点上部署了任务 T_j，然后用下面的两个方程式来完成本地的信息素函数的更新：

$$\tau_{in}(t'+1) = (1+\delta) \times \tau_{in}(t) + \sum_{k=1}^{m} \Delta\tau_{in}^{k}(t)$$

$$\Delta\tau_{in}^{k}(t) = \begin{cases} \dfrac{Q}{D} \\ 0, \text{ 其他} \end{cases}$$

5. 仿真设计与分析

本项目拟通过一系列模拟试验，对所构建的大数据均衡配置模型进行综合评价。本项目拟以高可靠、高容错能力的分布式服务平台为研究对象，通过比较研究各种部署方案的性能差别，突出本项目研究成果的优越性。在试验中，将贪心启发式算法和随机样本划分模型两种模式进行比较，从而更加全面地评估所提出的算法。

5.1. 响应时长性能分析

大数据处理过程中，响应时间的长短直接影响着大数据的处理效率和用户满意度。三种方法的平均响应时间随并发请求数量的增多而有差异。从图 3 可以看出，虽然该算法的响应时间随并发次数的增多而增加，但是当同时出现 200 个并发请求后，该算法的响应时间就变得稳定，不再明显增长。而采用随机抽样的模式，虽然初期的反应时间和本文提出的算法相似，但是随着时间的增加，总的响应时间变长。

而贪婪型启发式算法在大量的并发请求下，其性能会显著降低。实验结果表明，该算法具有较好的实时性和实时性。

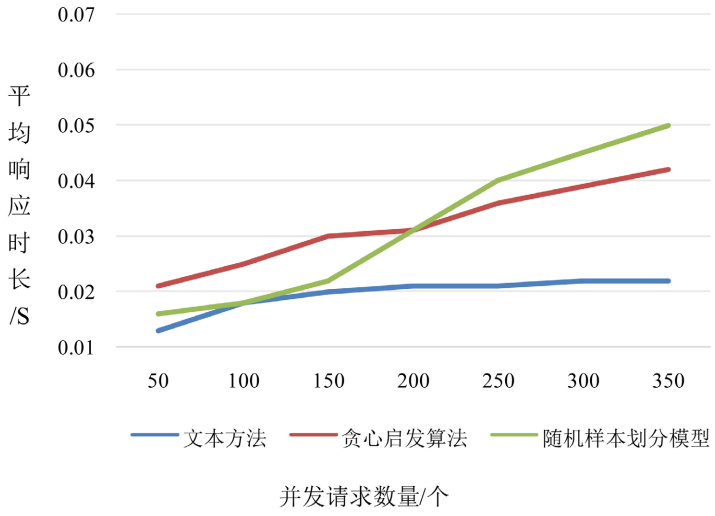


Figure 3. Comparison of deployment response time of different algorithms
图 3. 不同算法部署响应时长对比图

5.2. 引擎负载均衡评价

在计算引擎的部署中，负载平衡是一个非常关键的步骤，其性能的好坏将直接影响到整个系统的运行稳定性和运算效率。因此，本文采用具体的评估函数，量化地分析了三种算法在负载平衡方面的表现。如图 4 所示，在同等引擎数目的情况下，本项目提出的算法能够获得更高的负载平衡函数，并且没有明显的变化趋势。实验结果显示，这种部署方式的负载平衡能力较强，且与计算工作量无关。然而，当任务规模很大时，基于贪婪的启发式方法及随机抽样方法会导致系统的负载平衡效果显著降低，且不能保证系统的平衡。

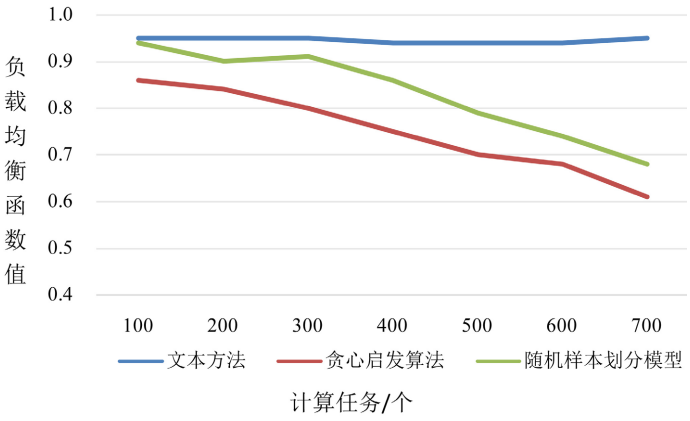


Figure 4. Comparison of load balancing using different methods
图 4. 不同方法负载均衡对比图

5.3. 部署能耗分析

计算能耗是评价发动机部署方式有效性的一个关键因素。能耗越小，就越能承担更多的运算任务，

从而提升整个运算效率。在此基础上，本文从总体能耗和单元计算能耗两个角度对三种方法进行了对比分析(图 5)。

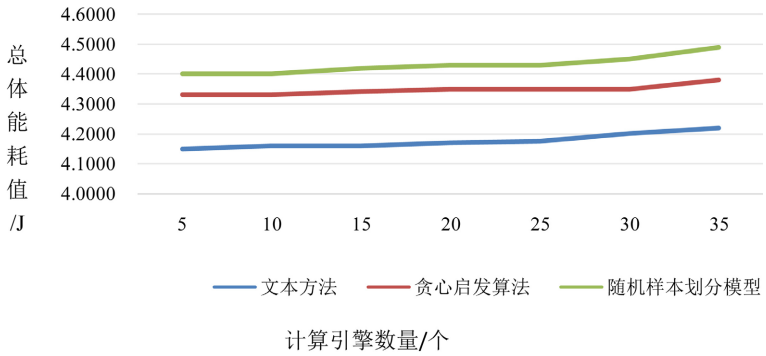


Figure 5. Overall energy consumption diagram of the system under different methods
图 5. 不同方法下系统整体能耗图

由图 5 看出，三种部署方法消耗的系统能量变化趋势基本相同，其中本文模型消耗的能量最低，可保证系统稳定运行。这是因为当计算任务量相等时，引擎数量增加会减少单位计算量，因此单个引擎的所需能耗会下降。

6. 结论

本文研究了蚁群算法在大数据计算引擎均衡部署中的应用，提出了基于蚁群算法的均衡部署数学建模方法。通过仿真实验验证了该模型的有效性，为大数据计算引擎的均衡部署提供了新的思路和方法。实验结果表明，基于蚁群算法的均衡部署方案在多个指标上均表现出较好的性能，能够有效地提高计算引擎的性能和稳定性。通过仿真设计与分析，验证了基于蚁群算法的大数据计算引擎均衡部署数学模型的性能优势。该模型在响应时长、负载均衡和部署能耗等方面均表现出良好的性能，能够有效提高大数据计算引擎的部署效率和计算效率，为实际应用提供了有力的支持。

参考文献

[1] 张令涛, 赵林, 张亮, 等. 配用电大数据分布式计算集群负载均衡框架[J]. 电网技术, 2019, 43(1): 259-265.
[2] 黄哲学, 何玉林, 魏丞吴, 等. 大数据随机样本划分模型及相关分析计算技术[J]. 数据采集与处理, 2019, 34(3): 373-385.
[3] 王龙晖. 基于并行计算的调节阀大数据智能分析及建模方法研究[D]: [博士学位论文]. 济南: 山东大学, 2020.
[4] 王超. 大规模异构传感通信大数据智能调度仿真[J]. 计算机仿真, 2019, 36(4): 445-448, 473. https://doi.org/10.3969/j.issn.1006-9348.2019.04.093
[5] 杨蕴睿. 网络通信软件共享资源均衡调度仿真研究[J]. 计算机仿真, 2019, 36(5): 333-336.
[6] 塔娜. 大数据平台网络数据库云计算技术优化与平台搭建研究[J]. 办公自动化, 2023, 28(11): 62-64.
[7] 滕飞, 刘洋, 曹芙. 基于节点实时负载的开源大数据负载均衡优化算法[J]. 吉林大学学报(信息科学版), 2023(6): 1106-1111.
[8] 邓国宝, 查晓文, 刘涛, 等. 试飞数据查询引擎设计[J]. 计算机测量与控制, 2023, 31(10): 208-213.
[9] 戚雄伟. 杭钢“平台 + 生态”打造产业数字化新引擎[J]. 企业管理, 2023(9): 110-114.
[10] 朱琳, 沈杨, 周川, 等. 基于粒子群的工业大数据雾计算多目标优化任务调度算法[J]. 南京理工大学学报, 2023, 47(1): 48-55.
[11] 杨阳, 凌东, 马杰宏, 等. 一种跨异构引擎的大数据生产任务调度框架[J]. 信息技术, 2023, 47(3): 29-34.
[12] 李海霞. 基于规则引擎的数据计算方法装置设备及存储介质[P]. 中国, CN202211559415.3. 2024-04-20.

[13] 程盛阳. 流式计算引擎中密集滑动窗口的性能优化研究[J]. 软件工程, 2023, 26(4): 42-45.

[14] 谢蒂, 顾小刚. 基于规则引擎的数据计算方法装置设备及存储存质[P]. 中国, CN202310618809.X. 2024-04-20.

[15] 张子浪, 刘海滨, 李小言, 等. 计算引擎选择模型的训练方法计算引擎选择方法及装置[P]. 中国, CN202310274395.3. 2024-04-20.

[16] 郑春辉. 计算机大数据分析与云计算网络技术研究[J]. 信息记录材料, 2023, 24(8): 146-148.

[17] 李磊. 分布式计算引擎的集群管理及负载均衡策略研究[D]: [硕士学位论文]. 成都: 电子科技大学, 2024.

[18] 乔颖 葛景晓, 王宏安, 等. 一种基于负载均衡的流式计算引擎调度方法及系统[P]. 中国, 202110790363. 2024-04-20.

[19] 迟建春, 郑伟, 王克敏. 负载均衡引擎客户端分布式计算系统以及负载均衡方法[P]. 中国, CN201710526509.3. 2024-04-20.

[20] 吴志敏, 吕慧伟, 陈明宇. 一个针对并行模拟引擎的性能评测实例[J]. 计算机科学, 2013, 40(3): 41-45. https://doi.org/10.3969/j.issn.1002-137X.2013.03.008