Software Engineering and Applications 软件工程与应用, 2023, 12(6), 883-894
Published Online December 2023 in Hans. https://www.hanspub.org/journal/sea
https://doi.org/10.12677/sea.2023.126086

Hans汉斯

基于麻雀搜索算法优化分层粒子群的虚拟机放置

钟崇楷，黄春梅

哈尔滨师范大学计算机科学与信息工程学院，黑龙江 哈尔滨

收稿日期：2023年11月1日；录用日期：2023年12月12日；发布日期：2023年12月21日

摘　要

随着用户和应用程序数量的不断增长，云数据中心对虚拟机的需求也日益增加。虚拟机放置(VMP)作为实现高效资源管理的关键问题，备受关注。本文针对VMP问题提出了一种新的优化模型，综合考虑了放置时间、功率消耗和资源浪费三个目标的最小化。为了优化VMP方案，我们采用了基于麻雀优化分层粒子群算法(SSA-HPSO)。该算法通过对粒子进行层次划分，使得粒子的搜索策略和更新规则针对不同层次和能量水平进行优化。同时，结合麻雀搜索算法，进一步提高了搜索效率和全局搜索能力。这种混合优化策略充分利用了分层粒子群算法的全局搜索和麻雀搜索算法个体之间的协同搜索能力，从而有效地解决了VMP问题。实验结果表明，所提出的基于麻雀搜索算法优化分层粒子群的虚拟机放置算法要优于传统的方法，显著提升了虚拟机放置性能。

关键词

虚拟机放置，资源管理，能量消耗，麻雀搜索算法，粒子群优化，云计算

Optimizing Virtual Machine Placement of Hierarchical Particle Swarm Based on the Sparrow Search Algorithm

Chongkai Zhong, Chunmei Huang

School of Computer Science and Information Engineering, Harbin Normal University, Harbin Heilongjiang

Received: Nov. 1st, 2023; accepted: Dec. 12th, 2023; published: Dec. 21st, 2023

Abstract

As the number of users and applications continues to grow, so does the demand for virtual ma-

文章引用: 钟崇楷, 黄春梅. 基于麻雀搜索算法优化分层粒子群的虚拟机放置[J]. 软件工程与应用, 2023, 12(6): 883-894. DOI: 10.12677/sea.2023.126086

chines in cloud data centers. Virtual machine placement (VMP), as a key issue to achieve efficient resource management, has attracted much attention. In this paper, we propose a new optimization model for the VMP problem, considering the minimization of three objectives: placement time, power consumption and resource waste. To optimize the VMP scheme, we used a sparrow-based optimization algorithm (SSA-HPSO). The algorithm optimizes the search strategy and updates rules for different levels and energy levels. At the same time, combined with the sparrow search algorithm, further improves search efficiency and global search ability. This hybrid optimization strategy fully utilizes the global search ability of the hierarchical particle swarm algorithm and individual sparrow search algorithm, thus effectively solving the VMP problem. The experimental results show that the proposed algorithm for optimizing hierarchical particle swarm based on the spar-row search algorithm is better than the traditional method and significantly improves the VVC placement performance.

Keywords

Virtual Machine Placement, Resource Management, Energy Consumption, Sparrow Search Algorithm, Particle Swarm Optimization, Cloud Computing

1. 引言

云计算系统中的虚拟化技术是一项关键技术，通过多个虚拟机将资源分配给多个用户[1]。数据中心包含大量物理机器，每台物理机器上托管着多个虚拟机。为了满足用户的请求，需要对每个请求进行建模和检查，以确定分配和执行任务所需的虚拟机资源。将虚拟机放置在物理机器上被称为虚拟机放置(VMP)，云服务提供商致力于优化 VMP 方法，以最大化每台物理机器上托管的虚拟机数量，从而增加数据中心托管虚拟机的数量[2]。VMP 是一个复杂的问题，考虑到用户请求的可扩展性、物理机器的异构性和多种资源维度。对于特定的数据中心，VMP 方法需要考虑具体要求。有效性可以通过诸如功耗、成本、资源利用率和负载均衡等指标来衡量。为了解决 VMP 问题，研究了许多元启发式算法，例如粒子群优化、遗传算法、蚁群优化算法等[3]。虽然元启发式算法比精确策略更先进，但它们计算成本较高。现有的研究主要关注降低功耗、最大化资源利用率、降低成本和负载均衡等目标，但较少关注虚拟机放置的时间消耗[4]。放置时间指的是将虚拟机放置在云服务器上所需的时间，对云服务提供商和用户都很重要。增加放置时间可能违反服务级别协议，并导致用户不满意。因此，在解决 VMP 问题时，有必要考虑放置时间，以避免由放置时间引起的问题。

本文提出了一个新模型来解决 VMP 问题，通过生成三个目标的最优解。这三个目标是：最小化将每个虚拟机放置在适当的物理机器上所需的总时间，减少数据中心物理机的能量消耗，以及最小化资源浪费。提出的多目标适应度函数包含三个参数，用于表示这三个目标。根据目标的重要性，这些参数被等权重地使用来得到合适的适应度函数。

2. 相关工作

一些研究学者已经探讨了虚拟机适当放置的重要性[5]-[11]。如何解决虚拟机布局问题一直是一个活跃的研究课题，参考文献[12]、[13]和[14]描述了虚拟机布局的背景和最新布局策略。虚拟机放置问题

(VMP)在文献中得到了广泛讨论[15]。我们根据问题的策略、目标和上下文来寻找解决方案。为了确定在物理机上虚拟机布局的最优解，现有的工作可以分为：精确算法，如线性规划、动态规划、分支定界和回溯；以及近似算法(启发式算法、元启发式算法)，如贪婪算法、局部搜索、遗传算法、禁忌搜索、模拟退火和随机演化。本文主要研究元启发式方法[16]。蚁群优化(ACO)元启发式算法被嵌入到能效启发式算法中，它减少了活跃物理机的数量，为数据中心节约能源。虽然该算法性能较好，但未考虑资源的浪费。现有研究中使用的适应度函数考虑了减少功耗、最大化资源利用、成本最小化和负载均衡等目标[17]，但未考虑放置时间，即虚拟机放置在云服务器上所需的时间，这对云服务提供商和潜在用户来说是一个重要指标。增加放置时间可能违反服务级别协议。PSO 是一种群体智能优化算法[18]，因其简单易实现和出色性能，成为许多研究者关注的研究焦点。虽然 PSO 是一种有效的算法，但在某些特定问题上可能存在早熟收敛等局限性。目前，为了提高粒子群优化算法的优化能力，存在许多不同的改进策略。这些改进策略可分为四类：参数调整策略、拓扑策略、学习策略和混合算法策略[19]。针对 PSO 算法独立实现的不足，对 PSO 进行了优化，形成了分层粒子群优化算法，同时在此基础上提出了 SSA-HPSO 算法。SSA-HPSO 算法将 PSO 算法的全局搜索能力与 SSA 算法个体之间的协同搜索能力相结合，以提高优化算法的性能。SSA-HPSO 算法是一种麻雀搜索算法优化的分层粒子群算法，这意味着它具有适应性，可以根据问题特性和优化过程的进展调整算法参数，以更好地适应不同类型的优化问题。

3. 虚拟机放置的约束优化目标模型

本节展示了 VMP 前提和应用架构。此外，还介绍了问题的具体表述。进一步地提出了拟议的适应度函数，该函数在优化算法实现中起着核心作用。该适应度函数是一个多目标函数，旨在获得云数据中心中物理机(PMs)的放置时间、能量消耗和资源浪费的最优值，如图 1 所示。

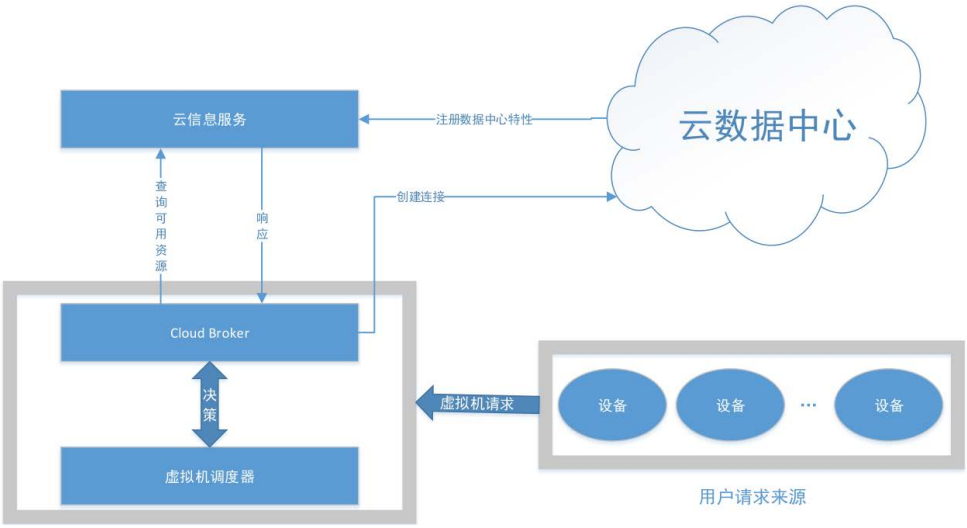

Figure 1. Architecture of the VMP model
图 1. VMP 模型的架构

3.1. 问题描述

VMP 模型的架构如图 1 所示，由四个关键组件构成：云数据中心、云信息服务、虚拟机调度器、以及云用户设备。云信息服务负责注册云数据中心内每个物理机(PM)的状态，并持续向 Cloud Broker 发送那些具备足够资源来托管所需虚拟机(VMs)的 PM 的每一次状态更新。这些状态包括每个数据中心中 PM

所提供的服务速率和队列中预期等待时间。Cloud Broker 用来接收所有用户的虚拟机请求，并将收集的关于可用 PM 和请求 VM 的信息转换成相应的向量形式，提交给 VMP 调度器。VMP 调度器利用这些向量，并依靠提出的适应度函数和优化算法，对所需的虚拟机进行放置决策。最终，根据调度器的决策，所请求的虚拟机被放置在相应的物理机上。

将 VMP 视为数学问题的模型，其中数据中心的物理机被表示为 $P=\{p_1,p_2,p_3,\ldots,p_N\}$，总数为 N 个，每个 p_i 可由 CPU 和内存进行定义。$C=\{c_1,c_2,c_3,\ldots,c_N\}$ 和 $M=\{m_1,m_2,m_3,\ldots m_M\}$ 分别表示物理机的 CPU 和内存资源。虚拟机的请求被表示为 $V=\{v_1,v_2,v_3,\ldots,v_M\}$，总数为 M 个，$C'=\{c_1^{'},c_2^{'},c_3^{'},\ldots,c_M^{'}\}$ 和 $M'=\{m_1^{'},m_2^{'},m_3^{'},\ldots m_M^{'}\}$ 分别表示虚拟机的 CPU 和内存需求。通过构建 $N\times M$ 的矩阵 $E=P^TV$ 来建立 M 个虚拟机请求和 N 个物理机之间的映射关系。每个虚拟机的位置可以用二进制变量 e_{ij} 表示，当 e_{ij} = 1 表示虚拟机 v_j 被放置在物理机 p_i 上，否则 e_{ij} 置为 0，$i=1,2,\ldots,N$，$1,2,\ldots,M$。X 表示物理机的向量，每个物理机至少托管一个虚拟机，用决策变量 x_k 表示，其中 $k=1,2,\ldots,N$，每个决策变量如公式(1)所示。

$$x_k=\begin{cases}1, & \sum_{j=1}^{M}e_{ij}\geq 1, 1\leq k\leq N\\ 0, & \text{其他}\end{cases} \tag{1}$$

元素 e_{ij} 和 x_k 被设置为一元决策变量，当满足以下两个约束条件(2)(3)时该变量等于 1，否则为 0。

$$\frac{C'}{C}\leq 1 \tag{2}$$

$$\frac{M'}{M}\leq 1 \tag{3}$$

物理机对所有托管虚拟机的标准化 CPU 利用率如公式(4)所示；物理机对所有托管虚拟机的标准化内存利用率如公式(5)所示。

$$U_i(p)=\sum_{j=1}^{M}\frac{e_{ij}c_j^{'}}{c_i} \tag{4}$$

$$U_i(m)=\sum_{j=1}^{M}\frac{e_{ij}m_j^{'}}{m_j} \tag{5}$$

3.2. 建模放置时间

虚拟机的放置时间记为 T_j，表示从虚拟机被请求的时刻开始计算，到虚拟机被放置到指定的物理机上所经过的时间。虚拟机放置时间由搜索时间 $Tsearch_j$ 和排队时间 $Tqueue_j$ 两部分组成。

搜索时间公式(6)如下所示：

$$Tsearch_j=\sum_{k=1}^{N}x_k\left(T\left(\frac{c_j^{'}}{c_k}+\frac{m_j^{'}}{m_k}\right)+\delta_k\right) \tag{6}$$

排队时间公式(7)如下所示：

$$Tqueue_j=\frac{\mu\left(\frac{\lambda}{\mu}\right)^N P_o}{(N-1)!(N\mu-\lambda)^2}+\frac{1}{\mu} \tag{7}$$

$$P_o = \left(\sum_{n=0}^{N-1} \frac{\left(\frac{\lambda}{\mu}\right)^n}{n} + \frac{\left(\frac{\lambda}{\mu}\right)^n}{N!\left(1-\frac{\lambda}{N\mu}\right)} \right)^{-1} \tag{8}$$

其中 P_o 表示空队列的概率，λ 表示到达率，$1/\mu$ 表示期望。所有的虚拟机放置的总时间如公式(9)所示。

$$T = \sum_{j=0}^{M} \sum_{i=0}^{N} e_{ij} \left(Tsearch_j + Tqueue_j \right) \tag{9}$$

3.3. 建模能量消耗

服务器的功耗可以通过功耗与 CPU 利用率之间的线性关系来准确描述。当物理机处于空闲状态时，会关闭它们以节省能源消耗。因此，它们的空闲功耗不计入总能耗。最后，我们将第 i 台服务器的功耗定义为 CPU 利用率的函数，如公式(10)所示。其中，$U_i(\tau)$ 代表第 i 台主机服务器在时间 τ 的 CPU 利用率。$P_{\max}$ 和 $P_{\min}$ 分别表示最大利用率时和空闲时的功耗。

$$E_i(\tau) = \begin{cases} (p_{\max} - p_{\min}) \times \dfrac{U_i(\tau)}{100} + P_{\min}, U_i(\tau) > 0 \\ 0, \qquad \text{其他} \end{cases} \tag{10}$$

3.4. 资源浪费

为最大化 CPU 和内存的利用率，物理机的资源浪费如下公式(11)所示。

$$W_i = \frac{\left| L_i(p) - L_i(m) \right| + \varepsilon}{U_i(p) + U_i(m)} \tag{11}$$

其中 ε 是一个非常小的正实数 0.0001。W_i 表示第 i 台服务器的资源浪费，$U_i(p)$ 和 $U_i(m)$ 表示标准化的 CPU 和内存资源使用情况。$L_i(p)$ 和 $L_i(m)$ 表示标准化的剩余 CPU 和内存资源。

云数据中心所有的物理机总资源浪费如下公式(12)所示。

$$W = \sum_{i=1}^{N} x_i \frac{\left| \left(U_i(p) - \sum_{j=1}^{M} e_{ij} c_j^{'} \right) - \left(U_i(m) - \sum_{j=1}^{M} e_{ij} m_j^{'} \right) \right| + \varepsilon}{\sum_{j=1}^{M} e_{ij} c_j^{'} + \sum_{j=1}^{M} e_{ij} m_j^{'}} \tag{12}$$

3.5. 资源浪费

可以将放置问题形式化为最小化放置时间，最小化能量消耗，最小化资源浪费，通过每个目标线性加权的形式，可以得出如下公式(13)所示。

$$F(x) = w_1 f_1(x) + w_2 f_2(x) + w_3 f_3(x) \tag{13}$$

其中 w_1, w_2, w_3 时每个目标函数的目标权重，在这种情况下，每个权重可以计算为 $w_i = 1/n$，其中 n 是目标函数的数量。因此，每个目标函数的等权值为 $w_i = 1/3$。

4. SSA-HPSO 算法

4.1. 粒子编码

在应用 SSA-HPSO 算法解决虚拟机部署问题之前，需对粒子进行编码，将问题的解空间映射到粒子

的位置上。在 PSO 算法中，合理的编码不仅能获得更优的搜索结果，还有助于减小搜索空间。虚拟机部署常见的编码方式包括二进制编码和多值编码。对于将 m 个虚拟机部署在 n 台物理机上的问题，每个粒子表示一组虚拟机请求在一组物理机上放置的候选解决方案。位置矩阵 M 定义在解空间 EQ 中，用于触发粒子的运动。这样，每个粒子代表了放置问题的 $m \times n$ 个可能解之一。举例而言，考虑两个不同的粒子，它们都是从初始种群的解空间 M 中生成的。每个粒子用二维方案表示，其中使用一对多的映射来描述粒子内每个物理机和其托管的虚拟机之间的关系。

4.2. 粒子群

4.2.1. 传统粒子群

PSO 在 D 维空间中，通过大小为 N 的粒子群模拟了极值搜索过程。可变粒子的传输动态模型如下所示：

$$v_i^{t+1} = \omega v_i^t + c_1 \times r_1 \times \left(p_i^t - x_i^t \right) + c_2 \times r_2 \times \left(p_g^t - x_i^t \right) \tag{14}$$

$$x_i^{t+1} = x_i^t + v_i^{t+1} \tag{15}$$

在上述公式中，x_i^t 表示的是第 i 个粒子在 D 维空间中时刻 t 的位置。v_i^t 表示的是第 i 个粒子在时刻 t 的速度。p_i^t 表示粒子在时间段 t 内的最佳位置。w，c_1，c_2 分别表示的惯性权重，自主学习因子和社会学习因子。r_1 和 r_2 为两个随机数。每个粒子都具有良好的能量以在空间中找到最优解，粒子的运动轨迹如图 2 所示。

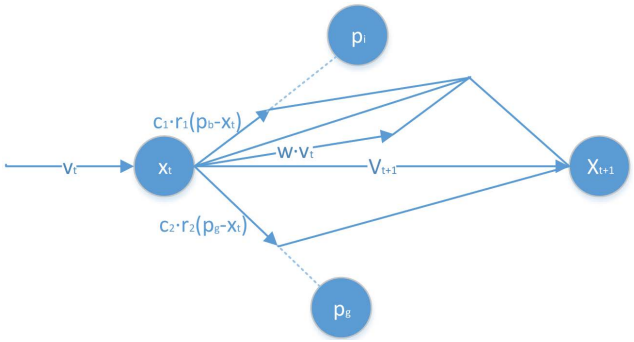

Figure 2. Particle swarm
图 2. 粒子群

4.2.2. 分层粒子群

传统粒子群优化算法存在着易陷入局部最优、搜索能力受限等缺点。而引入分层粒子群算法可以在一定程度上克服这些问题，它能够通过将粒子划分成不同的层次，实现更好的全局搜索和局部搜索的平衡，提高算法的收敛性和搜索效率[20]。这种分层设计使得粒子能够更有针对性地搜索解空间，从而更有可能找到更优的解。分层粒子群如图 3 所示。

$$E_M = \frac{1}{n} \sum_{i=1}^{n} X_{Mi} \tag{16}$$

$$E_{MH} = \frac{1}{n} \sum_{i=1}^{n} X_{MHi}, \left\{ X_{MHi} \mid f\left(X_{Mi} \right) \le fitness_M \right\} \tag{17}$$

$$\begin{cases} V_{MH_{i,d}}^{t+1} = w \times V_{MH_{i,d}}^t + c_1 \times \left(Pbest_{i,d} - X_{MH_{i,d}}^t \right) + c_2 \times \varepsilon \left(Gbest_d - X_{MH_{i,d}}^t \right) + a \times \left(E_{M-1} - X_{MH_{i,d}}^t \right) \\ X_{MH_{i,d}}^{t+1} = V_{MH_{i,d}}^t + X_{MH_{i,d}}^t \end{cases} \tag{18}$$

$$\begin{cases} V_{ML_{i,d}}^{t+1} = w \times V_{ML_{i,d}}^{t} + c_1 \times \left(Pbest_{i,d} - X_{ML_{i,d}}^{t} \right) + r \times \left(E_{MH} - X_{ML_{i,d}}^{t} \right) \\ X_{ML_{i,d}}^{t+1} = V_{ML_{i,d}}^{t} + X_{ML_{i,d}}^{t} \end{cases} \tag{19}$$

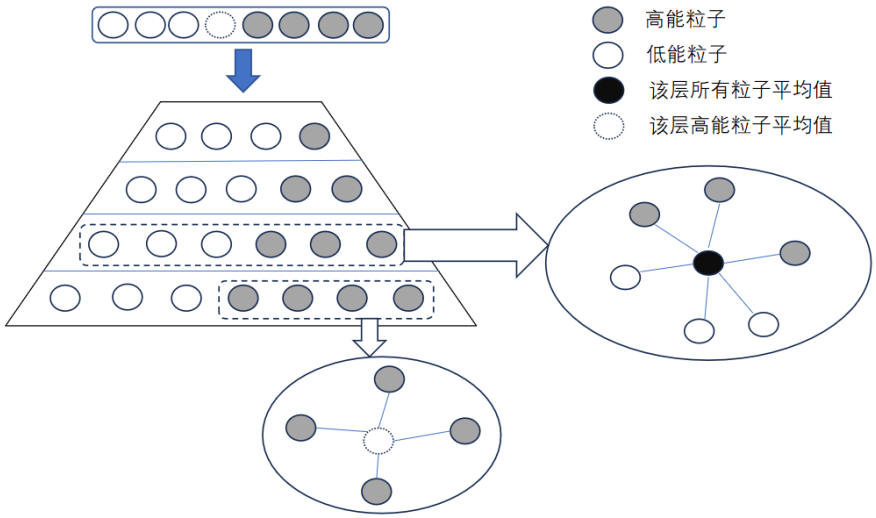

Figure 3. Hierarchical particle swarm
图 3. 分层粒子群

其中公式(16)表示该层所有粒子的平均值，公式(17)表示该层高能粒子的平均值，公式(18)该层的高能粒子表示，公式(19)表示该层的低能粒子。

4.3. 麻雀搜索算法

麻雀搜索算法是 Xue 和 Shen 于 2020 年提出[21]，主要模拟麻雀的觅食过程，包括发现者－追随者模型和检测预警机制。在一个麻雀群中，一些个体作为有能力定位食物的发现者，而另一些个体作为追随者，跟踪发现者到食物来源。同时，该算法通过选择特定比例的个体监测周围环境，结合了检测和预警机制。一旦发现危险，这些个体就会立即放弃寻找食物，把安全放在首位。通过对麻雀觅食过程的模拟，提高了搜索效率和算法的生存能力，保证了优化过程中对安全因素的考虑。

在 SSA 中，种群中最优秀的个体在搜索过程中优先获得食物。作为探索者，与追随者相比，他们有更大的搜索空间。在每次迭代中，探索者的位置更新方法如公式(20)所示。

$$X_{i,j}^{t+1} = \begin{cases} X_{i,j}^{t} \times \exp\left(\dfrac{-i}{\alpha \times iter_{\max}} \right), & IF \quad R2 < ST \\ X_{i,j}^{t} + Q \times L, & IF \quad R2 > ST \end{cases} \tag{20}$$

其中 $X_{i,j}$ 表示麻雀个体的位置。i 是当前迭代次数。$iter_{\max}$ 是最大迭代限制。α 是一个在[0, 1]范围内的随机数。R2 (R2∈[0, 1])和 ST (ST∈[0.5, 1])分别为警示值和安全值。Q 是服从正态分布的随机数。L 是一个 1 × d 矩阵，其中每个元素都被设置为 1。

当 R2 < ST，它表示附近没有捕食者，允许探索者进行全局搜索。如果 R2 ≥ ST，则表明部分麻雀探测到捕食者的存在，触发了所有麻雀之间的协调行动。在觅食过程中，一些追随者会密切关注探险者。一旦探险家发现了更好的食物，他们就会立即离开现在的位置去争夺食物。如果他们在竞争中获胜，他们可以立即获得食物；否则，必须继续按照公式(21)进行。

$$X_{i,j}^{t+1}=\begin{cases}Q\times\exp\left(\dfrac{X_{worst}^{t}-X_{i,j}^{t}}{i^{2}}\right), & IF\ i>\dfrac{n}{2}\\ X_{p}^{t+1}+\left|X_{i,j}^{t}-X_{p}^{t+1}\right|\times A^{+}\times L, & \text{其他}\end{cases} \tag{21}$$

在这个等式中，X_p 表示最佳资源管理器的位置，X_{worst} 表示当前全局最差位置，n 表示总体大小。A 是一个 1 × d 矩阵，其中每个元素取 1 或−1 的随机值。这里，A+的定义如公式(22)所示。

$$A^{+}=A^{T}\left(AA^{T}\right)^{-1} \tag{22}$$

当 i > n/2，表示适应度较低的第 i 个追随者状态较差，需要飞到别处觅食。在算法中，原始文本假设种群中有 10%到 20%的个体(在文本中设置为 20%)意识到危险，并且它们在种群中的初始位置是随机生成的，如公式(23)所示。

$$X_{i,j}^{t+1}=\begin{cases}X_{best}^{t}+\beta\times\left|X_{i,j}^{t}-X_{best}^{t}\right|, & IF\ f_i>f_g\\ X_{i,j}^{t}+K\times\left(\dfrac{\left|X_{i,j}^{t}-X_{worst}^{t}\right|}{\left(f_i-f_w\right)+\varepsilon}\right), & IF\ f_i=f_g\end{cases} \tag{23}$$

式中，X_{best} 表示当前全局最佳位置，β 为步长控制参数，其服从均值为 0，方差为 1 的随机分布；K 为[−1, 1]范围内的随机数；F 表示适应度值，f_g 和 f_w 分别表示当前最佳和最差适应度值；ε 是一个常数，用来避免被 0 整除。

总而言之，当 f_i > f_g，表示麻雀处于群体边缘，当 f_i = f_g 时，表示处于群体中间的麻雀意识到了危险，需要靠近其他麻雀以避免被捕食。其中，K 表示麻雀的运动方向，作为步长控制参数。

4.4. SSA-HPSO 算法步骤

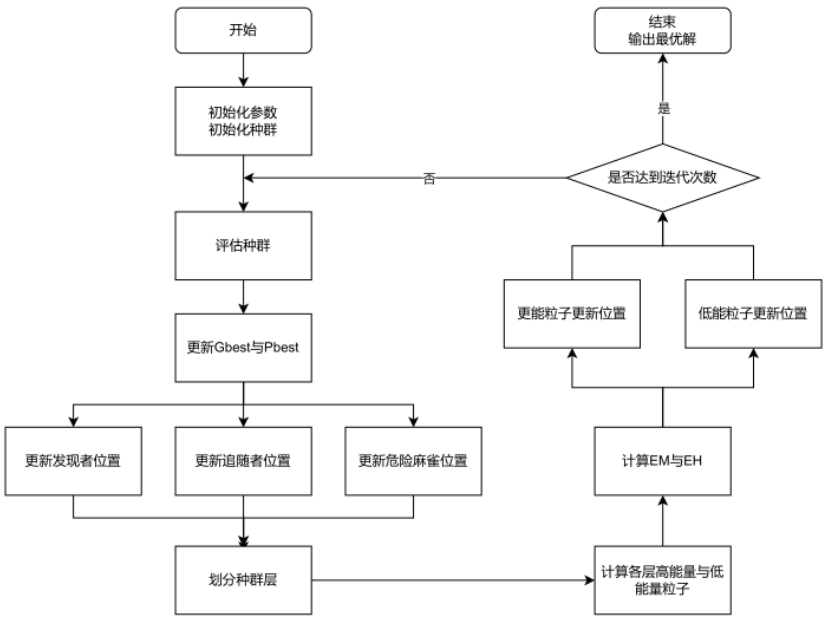

Figure 4. Flow chart
图 4. 流程图

分层粒子群算法在提升全局与局部搜索平衡方面取得了一定进展，但对于高维复杂问题，其搜索效率仍受限。将麻雀搜索算法融入分层粒子群算法中，可为优化过程注入新的活力。麻雀搜索算法的个体之间具有协同搜索的能力，这有助于多个粒子在搜索空间中相互引导和协作，从而更容易跳出局部最优解，扩大搜索范围，提高全局搜索性能。融合麻雀搜索算法于分层粒子群算法中，不仅弥补了其不足，更提升了搜索效率与解的质量。这一融合方法为解决高维复杂问题提供了一种新的、高效的优化方案。流程图如图 4 所示。

5. 实验设计与结果分析

本文进行了两种仿真实验，分别对比了 SSA-HPSO 算法和 PSO、WOA、GWO 算法在搜索虚拟机部署方案时的放置时间、能耗、资源浪费。同时，比较了 SSA-HPSO 算法、PSO 算法、WOA 算法和 GWO 算法在处理虚拟机部署时总适应函数度值。在这两种仿真实验中，SSA-HPSO 算法的参数如表 1 所示。

Table 1. Parameter setting of the proposed algorithm
表 1. 所提算法的参数设置

参数名	值
迭代次数	100
种群规模	800
w	0.6
C1	1.5
C2	1.5
a	0.4
rw	0.4
ST	0.8
PD	0.2

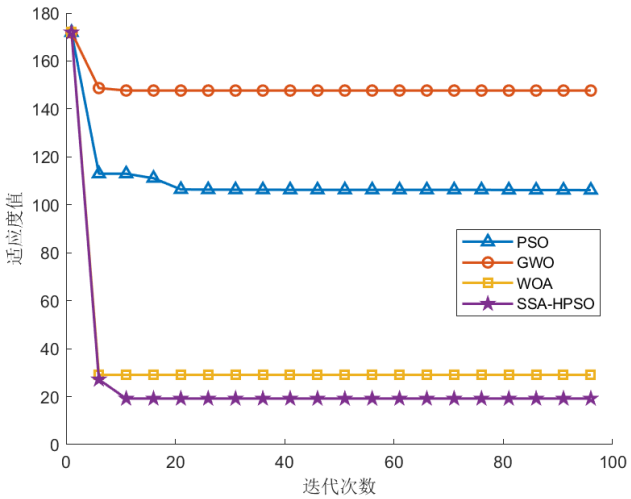

Figure 5. Fitness function comparison
图 5. 适应度函数对比

图 5 显示了三种算法在实验迭代过程中的适应度函数值。研究结果表明，SSA-HPSO 算法在适应度值下降速度方面胜过其他算法。此外，SSA-HPSO 算法在较少的迭代次数下达到了最低的适应度值。

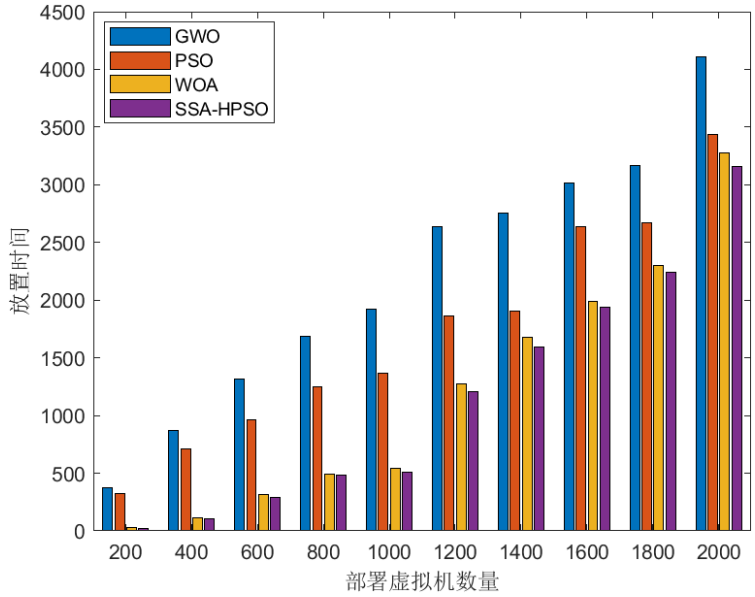

Figure 6. Placement time
图 6. 放置时间

在图 6 中，展示了三种算法的平均放置时间。随着每种算法的虚拟机请求数量的增加，平均放置时间也随之增加。然而，GWO 算法生成的平均放置时间最高，而 SSA-HPSO 算法生成的平均放置时间最低。因此，SSA-HPSO 算法有效地缩短了放置时间。

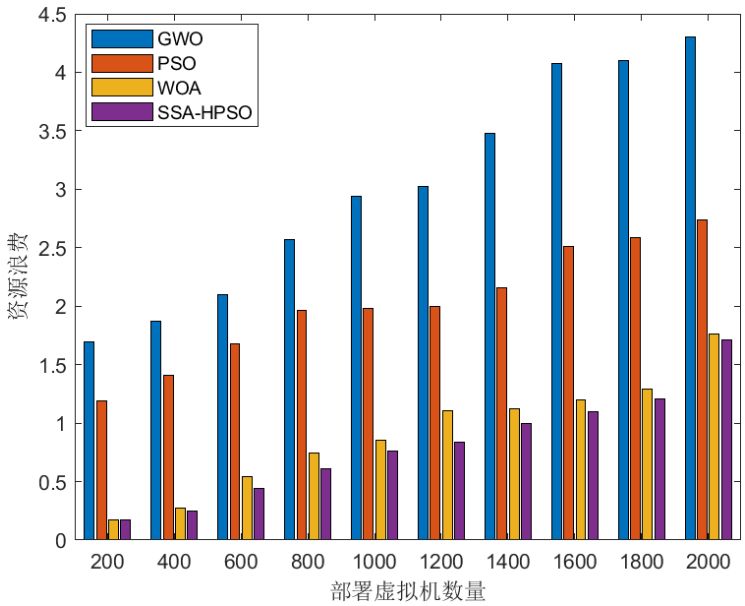

Figure 7. Wasting of resources
图 7. 资源浪费

图 7 中呈现了研究结果。可以观察到，随着虚拟机请求数量的增加，SSA-HPSO 算法相较于 GWO 算法、PSO 算法和 WOA 算法，在资源利用上浪费更少。这些结果源自 SSA-HPSO 算法的决策过程，它充分考虑了可用的剩余资源，并以平衡的方式利用这些资源，从而实现了最小的资源浪费。

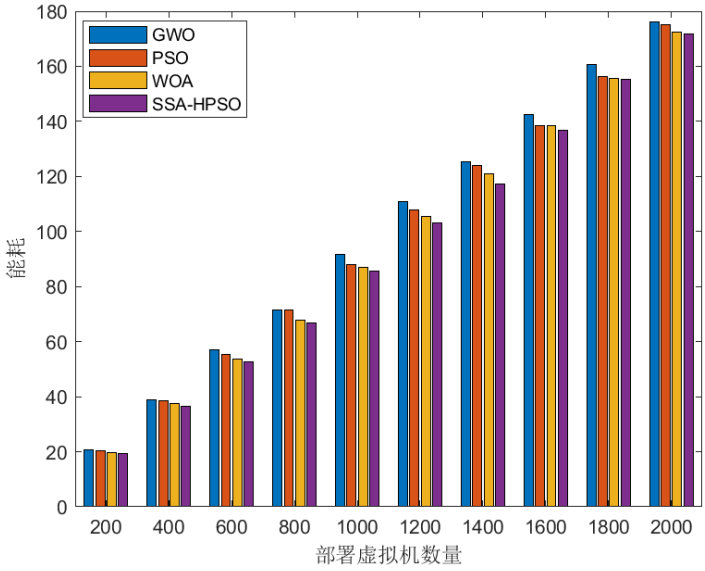

Figure 8. Energy consumption
图 8. 能耗

三种提出的算法的所有活动服务器的总耗电量显示在图 8 中。对于所有算法，随着 VM 请求数量的增加，功耗也在增加。然而，在 VM 请求数量方面，SSA-HPSO 算法的能耗始终低于其他算法。当使用 GWO 算法进行放置时，产生的功耗最高。

6. 结束语

本文提出了一种针对云计算环境中高效多目标虚拟机放置的优化策略。将麻雀搜索算法(SSA)个体之间的协同搜索能力和分层粒子群(HPSO)的全局搜索能力相结合，形成了 SSA-HPSO 算法。该算法综合利用了 SSA 的特性和 HPSO 的搜索能力，使得其在多目标虚拟机放置方面展现出更有效的性能。通过最佳的适应度函数来确定最佳的虚拟机放置解决方案，函数以三个标准值作为目标的结合，分别是虚拟机请求的总放置时间，在云数据中心的资源浪费和能耗。根据提出的适应度函数分别对 SSA-HPSO、GWO、PSO、WOA 算法进行评估。通过仿真的实验结果表明，相较于其他算法，SSA-HPSO 算法效率更高，表现更为有效。在未来的研究中，SSA-HPSO 算法可以扩展应用到其他虚拟机放置目标，例如资源利用率最大化、负载均衡、服务级别协议(SLA)、成本最小化等等。此外，所提出的算法也可结合深度学习技术用于服务云计算环境中的优化问题，进一步提升其性能和适应性。

参考文献

[1] Xiao, Z., Song, W. and Chen, Q. (2012) Dynamic Resource Allocation Using Virtual Machines for Cloud Computing Environment. *IEEE Transactions on Parallel and Distributed Systems*, **24**, 1107-1117. https://doi.org/10.1109/TPDS.2012.283

[2] Masdari, M. and Zangakani, M. (2020) Green Cloud Computing Using Proactive Virtual Machine Placement: Challenges and Issues. *Journal of Grid Computing*, **18**, 727-759. https://doi.org/10.1007/s10723-019-09489-9

[3] Gabhane, J.P., Pathak, S. and Thakare, N.M. (2021) Metaheuristics Algorithms for Virtual Machine Placement in Cloud Computing Environments—A Review. In: Pandian, A.P., Fernando, X. and Islam, S.M.S., Eds., *Computer Networks, Big Data and IoT*: *Proceedings of ICCBI* 2020, Springer, Berlin, 329-349. https://doi.org/10.1007/978-981-16-0965-7_28

[4] Donyagard Vahed, N., Ghobaei-Arani, M. and Souri, A. (2019) Multiobjective Virtual Machine Placement Mechanisms Using Nature-Inspired Metaheuristic Algorithms in Cloud Environments: A Comprehensive Review. *International Journal of Communication Systems*, **32**, e4068. https://doi.org/10.1002/dac.4068

[5] Kaur, H. and Anand, A. (2022) Review and Analysis of Secure Energy Efficient Resource Optimization Approaches for Virtual Machine Migration in Cloud Computing. *Measurement*: *Sensors*, **2022**, Article ID: 100504. https://doi.org/10.1016/j.measen.2022.100504

[6] Grit, L., Irwin, D., Yumerefendi, A., *et al*. (2006) Virtual Machine Hosting for Networked Clusters: Building the Foundations for "Autonomic" Orchestration. 1*st International Workshop on Virtualization Technology in Distributed Computing* (*VTDC* 2006), Tampa, 17 November 2006, 7. https://doi.org/10.1109/VTDC.2006.17

[7] Li, K. and Shen, H. (2004) Proxy Placement Problem for Coordinated En-Route Transcoding Proxy Caching.

[8] Li, K. and Shen, H. (2004) Optimal Placement of Web Proxies for Tree Networks. *IEEE International Conference on e-Technology, e-Commerce and e-Service*, Taipei, 28-31 March 2004, 479-486. https://doi.org/10.1109/EEE.2004.1287350

[9] Li, K. and Shen, H. (2004) Optimal Proxy Placement for Coordinated En-Route Transcoding Proxy Caching. *IEICE Transactions on Information and Systems*, **87**, 2689-2696.

[10] Li, K., Shen, H., Chin, F.Y.L., *et al*. (2005) Optimal Methods for Coordinated Enroute Web Caching for Tree Networks. *ACM Transactions on Internet Technology* (*TOIT*), **5**, 480-507. https://doi.org/10.1145/1084772.1084774

[11] Li, K., Shen, H., Cihn, F.Y.L., *et al*. (2007) Multimedia Object Placement for Transparent Data Replication. *IEEE Transactions on Parallel and Distributed Systems*, **18**, 212-224. https://doi.org/10.1109/TPDS.2007.29

[12] Talebian, H., Gani, A., Sookhak, M., *et al*. (2020) Optimizing Virtual Machine Placement in IAAS Data Centers: Taxonomy, Review and Open Issues. *Cluster Computing*, **23**, 837-878. https://doi.org/10.1007/s10586-019-02954-w

[13] Masdari, M., Nabavi, S.S. and Ahmadi, V. (2016) An Overview of Virtual Machine Placement Schemes in Cloud Computing. *Journal of Network and Computer Applications*, **66**, 106-127. https://doi.org/10.1016/j.jnca.2016.01.011

[14] Liu B, Chen R, Lin W, *et al*. (2023) Thermal-Aware Virtual Machine Placement Based on Multi-Objective Optimization. *The Journal of Supercomputing*, **79**, 1-28.

[15] Lopez-Pires, F. and Baran, B. (2015) Virtual Machine Placement Literature Review.

[16] Alharbi, F., Tian, Y.C., Tang, M., *et al*. (2019) An Ant Colony System for Energy-Efficient Dynamic Virtual Machine Placement in Data Centers. *Expert Systems with Applications*, **120**, 228-238. https://doi.org/10.1016/j.eswa.2018.11.029

[17] Adamuthe, A.C., Pandharpatte, R.M. and Thampi, G.T. (2013) Multiobjective Virtual Machine Placement in Cloud Environment. 2013 *International Conference on Cloud & Ubiquitous Computing & Emerging Technologies*, Pune, 15-16 November 2013, 8-13. https://doi.org/10.1109/CUBE.2013.12

[18] Wooldridge, M., Jörg, P.M., and Milind, T. (1995) Proceedings of the 1995 International Conference on Intelligent Agents II Agent Theories, Architectures, and Languages. Springer-Verlag, Berlin.

[19] Yu, F., Tong, L. and Xia, X. (2022) Adjustable Driving Force Based Particle Swarm Optimization Algorithm. *Information Sciences*, **609**, 60-78. https://doi.org/10.1016/j.ins.2022.07.067

[20] Wang, Y., Wang, Z. and Wang, G.G. (2023) Hierarchical Learning Particle Swarm Optimization Using Fuzzy Logic. *Expert Systems with Applications*, **2023**, Article ID: 120759. https://doi.org/10.1016/j.eswa.2023.120759

[21] Xue, J. and Shen, B. (2020) A Novel Swarm Intelligence Optimization Approach: Sparrow Search Algorithm. *Systems Science & Control Engineering*, **8**, 22-34. https://doi.org/10.1080/21642583.2019.1708830