分类号 _____ 密级 _____

U D C _____ 编号 _____

中南大學

CENTRAL SOUTH UNIVERSITY

硕士学位论文

论 文 题 目	基于改进 PSO 算法的网格任务调度算法
学 科、专 业	计算机科学与技术
研 究 生 姓 名	胡　金
导 师 姓 名 及 专业技术职务	杨 长 兴　教授

MS THESIS

Grid Task Scheduling Algorithm under the Improved PSO Algorithm

Specialty: Computer Science and Technology

Master Degree Candidate: Hu Jin

Supervisor: Prof. Yang Changxing

School of Information Science & Engineering

Central South University

Changsha Hunan P.R.C

原 创 性 声 明

本人声明，所呈交的学位论文是本人在导师指导下进行的研究工作及取得的研究成果。尽我所知，除了论文中特别加以标注和致谢的地方外，论文中不包含其他人已经发表或撰写过的研究成果，也不包含为获得中南大学或其他单位的学位或证书而使用过的材料。与我共同工作的同志对本研究所作的贡献均已在论文中作了明确的说明。

作者签名：胡金　　日期：2011年5月20日

关于学位论文使用授权说明

本人了解中南大学有关保留、使用学位论文的规定，即：学校有权保留学位论文并根据国家或湖南省有关部门规定送交学位论文，允许学位论文被查阅和借阅；学校可以公布学位论文的全部或部分内容，可以采用复印、缩印或其它手段保存学位论文。同时授权中国科学技术信息研究所将本学位论文收录到《中国学位论文全文数据库》，并通过网络向社会公众提供信息服务。

作者签名：胡金　导师签名：　日期：2011年5月20日

摘 要

网格是当今分布式计算研究领域最为活跃的部分，它以虚拟组织 VO（Virtual Organization）的形式灵活、有效地将不同管理域的异构资源组织起来，协同完成大型计算任务。任务调度是网格的重要组成部分，直接影响网格的性能。针对任务的个性化需求，如何有效地将任务合理地分配到计算资源上去，这是任务调度的主要内容。同时网格具有的动态性、异构性和分布性等特征，使得网格环境下的任务调度问题变得尤为复杂，已成为当前网格研究的热点，也被证明是个 NP 完全问题。因此，找到一个好的任务调度策略，尽可能缩短任务的总执行时间、有效均衡系统资源负载，具有非常重要的意义。

近年来，兴起将启发式算法应用于任务调度，取得了较好的研究成果。作为一种比较新颖的启发式方法，粒子群优化算法 PSO 搜索速度快、操作简单、效率高，能有效求解 NP 完全问题，被广泛应用于任务调度中。有仿真结果表明：在任务调度中，PSO 算法与其他启发式算法相比更具优越性。但是 PSO 算法固有的一些缺陷，使得算法在问题求解时易陷入局部极值、搜索精度不高。

针对 PSO 算法的不足，本文对时间 QoS 约束下独立任务的调度问题进行研究，提出了一种基于改进 PSO 算法的网格任务调度算法 MCPSO (An improved PSO of grid scheduling algorithm under the meta task)。该算法的主要思想是：首先采用混沌序列初始化大量粒子，并在产生的粒子中择优选出初始种群；然后在粒子更新时，引入混沌搜索，随机产生若干混沌序列并把最优混沌序列得到的位置和当前粒子最优位置比较。如果优于当前粒子，则更新当前粒子的最优位置，引导当前粒子跳出局部最优点，快速寻找最优解。

实验表明，该算法改善了 PSO 算法易陷入局部最优问题，同时兼顾更好的时间跨度 makespan 和负载均衡，有效提高网格的性能。

关键词 任务调度，PS0，混沌优化，最优跨度，负载均衡

ABSTRACT

Grid is currently the most active part of distributed computing research field, organizes the heterogeneous resources in different administractive domains flexibly and effectively in the form of VO(virtual organization) and coordinates the completion of large computing tasks. Task scheduling is the key issue of grid environment, and directly affects the performance of grid. Taking account to the individual needs of scheduling tasks, how to effectively allocate the tasks to the computing resources rationally, it's the main content of task scheduling in grid. In addition, as the grid is characterized by dynamic, herogeneous, distributed and so on, the issue of task scheduling in grid is becoming particularly complicated and it has become a hot issue of current grid research and also been proved to be a NP complete problem. So it is of great importance to find a better scheduling strategy which can shorten the completion time as much as possible and efficiently balance the workload of system resources.

In recent years, employing the heuristic algorithms in the field of task scheduling is on the rise, and the research results are much better. As a new one, Particle swarm optimization, PSO, is widely used for tasks scheduling which has quick search speed, simple operation, high efficiency, and can efficency resolve the NP complete problem. Some simulation results show that: compared to other heruristic algorithms, PSO algorithm has more advantages in the tasks scheduling. But it is easy to fall into local optimal solution, and the accuracy is not high when solving problems as some defects.

Aimming at the shortage of the PSO algorithm, the problem of independent task scheduling under the restraint of time QoS is studied, and a grid task scheduling algorithm based on an improved PSO algorithm is proposed(An improved PSO of grid scheduling algorithm under the meta task). The main idea of this algorithm is, it firstly initializes a large number of particles using chaotic sequences, and then selects the best particles from the generated particles as the initial population; Secondly, it introduces the chaos search while updating the

particles, randomly generates several chaos sequences and selects the best one to compare with the best position of current particle. If it is better than the current, then updates the current particle's best position with it to guide the particle to jump out of the current local optimum and find the best solution quickly.

Experiments show that this algorithm can improves the problem which the PSO algorithm is easily to be trapped into local optimum. At the same time, it also can takes better makespan and the performation of load balance into account, and improve the performation of grid system.

KEYWORDS task scheduling, PSO, Chaos optimization, optimum makespan, load balancing

目 录

第一章 绪论

1.1 研究背景

网格[1]是近年来兴起的技术研究热点，是继万维网后的又一次重大技术变革。网格的概念是借鉴日常生活中的电力网而提出的，其远大愿景是让用户使用网格中的资源如同电力网中的电能一样便捷。网格通过高速网络将不同地域的、分散的大规模资源进行有效整合，从而构筑成一个基础设施。经由这个基础设施，无需知道它上面的具体细节用户就可以自由使用自己所需资源。这个基础设施具有两大突出的优势：一是空前的计算能力；另一个就是网络中的闲散数据处理能力可以得到充分利用，而且性价比高。

网格以分布式资源和网络通信为物理支撑，消除“信息孤岛”为目标，资源管理和任务调度为核心内容。通过集成不同地域广泛分布的各类资源，网格不再受限于计算能力和存储能力的大小，可以跨地域供用户对资源的透明使用，为资源需求量大的问题，甚至是求解过程及其复杂的问题提供了一个有效的解决思路，极大地减少了用户在时间和费用方面的开销。因而，网格技术从根本上改变了传统的计算模式，必将为人类的生产生活带来非常深远的影响[2]。

资源共享是网格的核心[3]，而资源能否得到有效的利用由任务调度直接决定，因此网格中最重要且最为关键的问题之一就是任务调度。对于网格，任务调度所关心的主要问题是如何将不同用户的应用流调度到可用的计算资源上，从而使网格得到最大限度的使用，并使系统有高的吞吐率。高效的调度算法能使网格中的资源得到充分利用，提供好的服务质量。网格任务调度已被证实是一个 NP 问题[4-5]，在复杂多变的网格环境下要得到切实可行的最优调度，具有相当的难度。

目前，网格任务调度引起许多学者的关注，成为当前网格研究领域的热点问题。

1.2 研究现状

网格是近年来高性能计算领域的热门技术，研究的目标是将地域不同的、异构的存储资源、服务资源和计算资源合理的组织起来，并提供一个高可靠、高性能可透明访问的统一计算环境给用户。与传统的并行计算以及分布式计算相比，它具有性价比高、资源易于获取、地域和时间限制少等优点，因而具有很好的应用前景。

1.2.1 网格任务调度的研究现状

网格任务调度是在满足一定性能指标和任务优先约束关系的前提下，将任务按照一定的分配策略分派到不同的资源节点上执行的过程[6-7]。网格任务调度的一般目标是 makespan 最优，即达到总的完成时间最小。

当用户提交任务后，网格首先综合分析和评价所有资源，然后将任务分配到最合适的资源上执行。由于网格的用户量大，提交的任务种类和数目又千差万别，任务和资源的映射关系也变得极其复杂，使得网格的任务调度策略显得尤为重要。

由于网格中的资源具有诸多特性，如动态性、异构性、广域性等，因而任务在进行调度时需要充分考虑任务需求、机器特性，为不同任务匹配不同的资源，从而提高系统资源的利用率和任务的执行效率。

目前在任务调度方面的研究有很多[8-13]，按照调度算法的运行时间可分为静态调度、动态调度两种；根据调度算法运行的位置又可分为：集中式调度和分布式调度；而根据调度发生的时机则分为在线调度和离线调度等[14]。

任务调度算法一般按第一种方法分。静态调度算法主要针对的是任务的执行顺序及其资源在整个处理过程中不会再有变动。动态调度算法，顾名思义，要考虑网格环境是不断变化的，资源可能随时加入和退出，以及资源性能的稳定性无法保证，在调度中任务的计算时间和到达时间无法预先确定。在静态调度中，调度算法又可进一步细分为依赖任务调度、可分任务调度和独立任务调度。本文将主要研究独立任务调度。

目前关于网格环境下的独立任务调度出现了大量的研究[15-19]，它主要源于异构计算中的元任务(meta task)调度。一直以来，独立任务调度是调度领域的经典问题。对于任务间没有任何依赖，且不可再分的任务称之为独立任务。在独立任务调度方面，早期考虑的重点是同构并行处理机下的调度，并提出众多调度算法，如动态规划[20]、分支限界算法[21]、以及基于局部搜索思想的改进算法[22]、遗传算法[23]、Min-min 算法[24]等。

考虑到实际调度问题的复杂性，很多研究通过适当修改调度问题模型以简化问题，如为实现吞吐量的稳定而限定任务大小相同的调度，以最大化处理机最小负载为约束条件来追求负载均衡的调度，以存储资源大小为约束的调度等。

在网格应用中，独立任务调度的难点主要是资源的异构性和动态性。在异构性方面有比较多的研究，而对动态性问题的研究就很少。由于网格中资源的动态的加入和退出势必影响到任务调度的效率，导致调度器无法很好的工作。目前，全球网格论坛 GCF 借鉴网络中的资源预留思想，考虑使用提前预留的方法解决动态性问题[25]。

1.2.2 粒子群优化算法及其在网格任务调度中的研究现状

粒子群优化算法（Particle Swarm Optimization, PSO）是近年来兴起的比较新的启发式优化方法，它基于种群来搜索问题的解，并试图找到问题的最优解，在解决复杂的优化问题特别是NP完全问题时具有很大的优越性。

PSO算法应用的领域十分广泛。起初，PSO算法被用在优化非线性连续函数和训练神经元网络两个方面[26-27]，后来也逐渐被用来处理约束优化问题[28]。PSO算法最初处理的是连续优化类问题，之后Eberhart和Kennedy针对组合优化问题又提出了离散二进制版本的PSO算法[29]。在各类连续问题的参数调优方面，PSO算法的研究很多，如控制器设计[30]、电力系统优化[31]、机器人路径规划[32]等，并取得较好的结果。目前也有很多学者将PSO算法用于解决TSP[33]、最短路径问题[34]、车间调度[35]等。

PSO算法由于算法简单，易于编程实现，收敛速度快，可调参数少等特点，在求解连续优化问题以及组合优化问题时均有良好的表现，值得深入的研究。

PSO算法在网格领域的应用比较晚，相关的研究非常少，主要讨论的网格环境下任务调度问题。季一木等[36]将连续PSO算法来解决网格任务调度问题，取得相当不错的效果。朱海等[37]基于任务安全调度模型提出一种离散粒子群优化算法DPSO，收敛性能较好。迟玉红等[38]提出一种基于局部模型的PSO算法，有效改善算法早熟的缺陷。后两者虽然解决了连续PSO算法求解离散问题导致的收敛速度慢的问题，但是仍然没能有效的改善初始种群质量和提供更为高效的局部优化方案。因此，探究网格环境下PSO算法的改进并将其应用于任务调度问题中对网格的发展势必起到非常重要的作用。

1.2.3 研究意义

任务调度是网格的核心技术，决定网格资源是否得到有效合理的利用，是并行分布计算中最具挑战性的问题之一。随着网格的快速发展，对任务调度也做出了新的要求。网格是由大量分布在不同管理域的异构资源组成，这些资源协同工作将提供巨大的计算能力。由于网格中的资源具有分布、自治性、异构性、动态变化等特征，使得网格应用中的任务调度所面临的问题更加复杂。合理地协调和分配资源，有利于降低网格任务完成时间，有效提高网格资源的负载均衡性能，保证网格的稳定可靠性。因而，设计一种网格环境下的高效合理的任务调度算法，对促进网格的发展十分重要。

而PSO算法因固有的简单、容易实现、参数少、快速收敛等优良特性，使其在网格任务调度中有很好的适应性、实用性和有效性。本文在网格任务调度中，

对 PSO 算法进行相应的改进，使调度算法的跨度最优、系统负载更为均衡，这对提高网格的实用性具有重要的理论价值和现实意义。

1.3 研究内容

任务调度对于网格来说是至关重要，直接影响任务的执行效率乃至成败。高效的调度策略可以将系统的处理能力充分利用起来，进而提高应用程序的性能。本文针对网格中的独立任务调度问题进行了深入的研究，将混沌思想充分融入到 PSO 算法中，并将其应用于网格中的任务调度中，提出了基于改进 PSO 算法的网格任务调度算法。本文的研究工作主要如下：

1．确定网格任务调度的模型和性能指标

实际的网格环境对任务调度的影响很大，寻求精确而简单的模型往往是比较困难的。本文讨论的调度任务是相互独立、彼此间没有依赖关系的独立任务，也称元任务。目前多数调度算法也主要是针对元任务的。为了评价调度算法性能，本文将采用以下几个性能指标：

（1）求解质量

（2）时间跨度 makespan

（3）负载均衡

2．分析 PSO 算法并给出改进方案

本文通过深入的研究 PSO 算法，针对 PSO 算法的不足给出了相应的改进方案：首先采用混沌初始化粒子并选取最优的 *POPSIZE* 个作为初始种群，这样可以优化种群质量，大大加快算法的搜索速度；然后在算法的迭代中引入混沌搜索机制对每个粒子进行混沌优化，从而有效避免粒子的过早收敛得不到最优解的情况。

3．本文以 makespan 为优化目标，将改进的 PSO 算法引入到网格任务调度中，提出了 MCPSO (An improved PSO of grid scheduling algorithm under the meta task)网格任务调度算法。

该算法的主要思想是：首先采用混沌序列初始化粒子的位置，在产生的大量位置中择优选出初始种群的位置；然后在粒子更新时，引入混沌优化搜索，随机产生若干混沌序列并把最优混沌序列得到的位置和当前粒子最优位置比较。如果优于当前粒子，则更新当前粒子的最优位置，引导当前粒子跳出局部最优点，快速寻找最优解。

4. 使用 GridSim 来进行实际网格环境的仿真工作，并将本文提出的 MCPSO 算法与 PSO 算法进行对比分析。实验结果表明：MCPSO 算法改善了 PSO 算法易陷入局部最优问题，同时兼顾更好的 makespan 和负载均衡，有效提高网格的

性能。

1.4 论文结构

论文内容具体如下：

第一章 绪论。简要介绍本文的研究背景、研究意义、网格调度算法的研究现状、粒子群优化算法 PSO 及其在网格中的研究现状以及本文的研究内容和所要做的工作。

第二章 网格任务调度概述。介绍网格调度的基本概念、基本流程、调度目标、列举并分析几种常见调度算法以及调度模型和性能评价指标。

第三章 PSO 算法及混沌优化。简要介绍了最优化问题和无免费午餐定理，详细地从基本原理、流程、标准版本、主要参数、优缺点等方面对粒子群算法进行阐述，最后给出了算法的一般设计步骤和改进策略，及混沌优化的相关内容。

第四章 改进 PSO 算法的网格任务调度算法。对标准 PSO 算法进行改进，并结合混沌优化原理，提出了改进 PSO 算法 MCPSO，将其在独立任务调度中进行实验仿真，验证其有效性。

第五章 实验和性能分析。介绍几个常用的网格仿真平台，着重介绍了 GridSim 的结构和原理。然后对算法在仿真环境下进行验证，将本文提出的改进算法 MCPSO 与标准 PSO 算法在完成时间约束条件下进行对比分析，实验结果表明，本文提出的算法能缩短任务调度的完成时间，并提高网格系统的负载均衡性能。

第六章 结束语。简要地对论文的研究工作进行总结，并展望接下来所要展开的工作。

第二章 网格任务调度概述

网格环境下的任务调度机制对网格系统的性能有着直接影响，是当前网格研究领域的一个前沿方向，同时也存在许多挑战和问题。网格任务调度首先依据任务的信息寻找满足条件的网格资源，然后按照一定的调度策略从这些资源中选取一个合适的分配给该任务，任务在获得资源后就可以在该资源上执行。针对任务的多样化需求和复杂多变的网格环境，如何合理高效将任务分配到适合的资源，过程十分复杂，是个NP难解问题，已经成了一大研究热点。

本章主要介绍网格任务调度的相关内容，包括调度特点、过程和目标，几种常用网格任务相关调度算法以及网格调度模型。

2.1 网格任务调度

缩短任务集的总体执行时间 makespan，提高系统负载均衡性能是网格环境下任务调度的优化目标。任务调度的流程一般有两步，将任务分配给网格资源是第一步，这也是本文所要着重讨论的；第二步就是任务在单个资源节点上的再次划分调度，该节点对任务进行细分后，进一步将子任务分配到此资源节点的各个处理器上。因而网格任务调度通常涉及到网格系统中资源节点间的调度和资源节点上的本地调度相结合的问题。

作为网格的重要组成，任务调度的研究内容主要是异构环境下资源与任务的最佳匹配，管理及调度任务执行，不断优化网格资源的利用率。对调度造成影响的因素有很多，如应用需求、处理器性能、网络带宽和延迟等，如何对资源进行合理调度，已成为网格领域研究的重点。

网格任务调度系统具有如下特点[39]：（1）异构性。网格的异构性要求网格环境中的任务调度也是异构的；（2）规模大且非集中式的；（3）自治性。网格任务调度对网格节点内部的调度不造成任何影响；（4）可扩展性即随着资源节点不断的动态加入和退出，调度必须是可以实时扩展的；（5）容错性。调度系统必须能够自适应网格资源的动态性。

2.1.1 网格任务调度过程

一个典型的网格任务调度至少包括有三个阶段：资源发现、资源选择和作业执行[40]。网格系统调度的详细过程如图2-1所示。资源发现“负责”对已认证资源列表的识别。由于网格环境的动态性，任务调度器应始终知晓哪些资源此时可用以及它们的通信代价如何，然后综合可用资源的动态信息从而评估资源的可用

性。资源发现阶段可进一步细化为授权认证过滤、应用需求定义以及满足最小需求的资源过滤三个步骤；资源选择的工作是对前一阶段筛选后的资源进一步进行筛选，以最大程度地满足应用的需求。这一阶段的任务是动态信息采集和系统选择。例如网格信息服务 GIS 系统中的网络天气服务 NWS 可以对当前处理器可用时间片、可用内存及大小、带宽等网格详细情况实时监测；系统选择即资源调度，是通过综合相应的资源和任务信息做出匹配决策的过程，属经典的 NP 完全问题；在做出调度决策之后，系统转入作业执行阶段。这一阶段的工作是将作业提交给相关资源节点执行，包括有一系列过程：①资源预留；②作业递交；③准备任务；④过程监测；⑤作业完成；⑥清除任务。

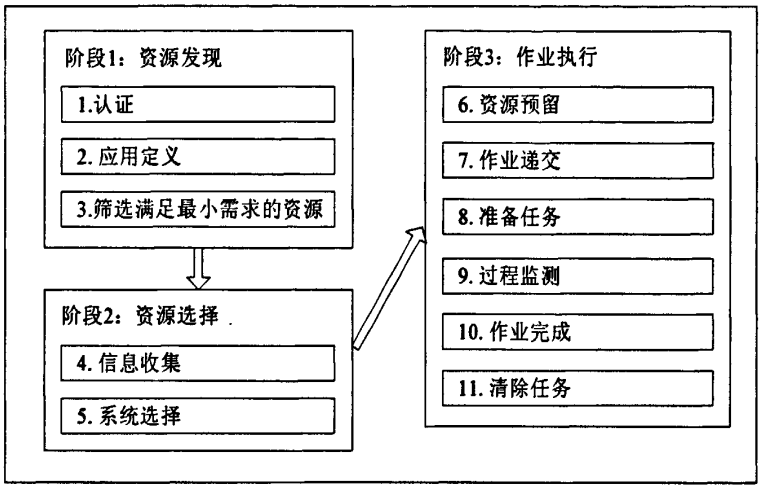

图 2-1 典型的网格任务调度过程

2.1.2 网格任务调度目标

一般来说，网格任务调度要实现的目标有如下几个方面：最优跨度（Optimal makespan）、服务质量 QoS（Quality of Service）、负载均衡（Load balancing）、经济原则（Economic principles）[39]。

（1）最优跨度

跨度指的是从开始执行第一个任务到最后一个任务执行结束所经历的时间。在网格任务调度中，跨度是最常使用的目标，跨度值越小意味着调度策略越好。目前，很多任务调度算法研究都是基于最优跨度来这一目标展开的，它不仅是网格系统的目标，也是用户所期待的。

现假定某虚拟组织 VO[3]中，用户提交有 n 个作业 $T=\{t_1,t_2,\ldots,t_n\}$，需要调度到 m 个资源 $R=\{r_1,r_2,\ldots,r_m\}$上运行，任务 t_i 在 r_k 的执行时间为 t_{ik}，执行前所需的等待时间为 t_{iw}，那么实现最优跨度就是找到一个这样的调度策略，使得跨度 makespan 值最小，makespan 的计算如式 2-1。

$$makespan = \max_{\substack{i\in\{1,2,\ldots,n\} \\ k\in\{1,2,\ldots,m\}}} \{t_{ik} + t_{iw}\} \quad (2\text{-}1)$$

（2）服务质量 QoS

服务质量 QoS 通常反映的并非用户对资源的某一性能要求而是多项性能要求，是一个综合性能指标，表征用户的满意程度。因此，任务调度时，网格应用的某些 QoS 需要同时得到保证。常见的有时间 QoS、优先级 QoS、费用 QoS、计算资源 QoS、网络资源 QoS 等。

假设某用户有 k 项 QoS 性能要求 $Q=\{q_1,q_2,\ldots,q_k\}$，q_i 表示用户对资源的第 i 项性能要求。网格任务调度系统中的各项性能效益值为：$Q_{res}=\{q_1^{res},q_2^{res},\ldots,q_k^{res}\}$。每个 QoS 参数的权重为：$W=\{w_1,w_2,\ldots,w_k\}$。各项权值的关系如式 2-2，其中，$0 \leqslant w_i \leqslant 1$。

$$\sum_{i=1}^{k} w_i = 1 \quad (2\text{-}2)$$

因而调度系统的综合效益值 *value* 的计算公式如式 2-3 所示。

$$value = \sum_{i=1}^{k} w_i \times q_i^{res} \quad (2\text{-}3)$$

（3）负载均衡

在实现分布式和并行计算应用过程中，负载均衡至关重要。网格环境下的资源多是广域分散和异构的，并且要能并行工作，因此，解决负载均衡问题显得尤为突出，并能有效的优化跨度值。

假设资源处理能力相当即同构资源，理论上，网格任务调度的最优结果是任务均分到各个处理机上，即式 2-4。

$$L_{min} = \frac{1}{m}\sum_{i=1}^{n} t_i \quad (2\text{-}4)$$

然而，L_{min} 在实际应用中一般是不可能得到的。对于一个调度算法，如果其调度长度即跨度值不存在其他能得到更短调度长度的算法，那么这个调度算法就是最优的。此时，各个处理机负载相当，也称系统负载均衡。

（4）经济原则

网格中的资源在地理上是广泛分布的，且分属的组织和机构不同。在网格中，用户对网格资源的使用需要付费，都希望得到性价比高的资源。由于资源的性能

和地域差异性较大，使用费用自然也是不同的。因此，在调度时必须从资源提供方（网格系统）和资源消费方（网格用户）双方的价格需求同时考虑，使双方能够互惠互利。

2.2 网格任务调度算法

在网格中，任务调度的实质是实现任务、资源相互间的合理匹配及任务在其匹配的资源上执行的过程。网格环境中的任务类型大体可分为两类，一类是独立任务（meta task）；而另外一类就是彼此间有依赖关系如数据依赖和控制依赖的任务，即依赖任务。本文研究的主要内容是独立任务的调度问题。独立任务相互之间既没有控制依赖也不存在数据依赖，且任务不可再分，因而其执行次序不会对调度的进行造成影响。根据调度的时机又可将独立任务的调度分为两类：在线模式（on-line）和批模式（batch-mode）。前者使用先来先服务（FCFS）的策略，优先处理先到达的任务。后者也称离线模式，是在任务数达到一定或是时间周期到或是事件触发条件下，对所到达的所有任务集中进行分配和调度。因而，批模式下收集到的资源信息更多，往往能得到更为合理的任务映射策略。而对于依赖任务，一般是通过综合任务的依赖关系先建立相应的调度列表，而后不断调度列表中的任务直至表中的任务执行完。

一般说来，网格任务调度可分为两步：

（1）空间上将任务映射到资源上，将合适的资源分配给相应的任务，然后把相关数据和任务提交给资源；

（2）在时间上和空间上对任务排序：不同任务间的通信排序与同一资源节点上的任务的排序。

网格中用于独立任务的调度算法[41-44]有：随机负载均衡算法（Opportunistic Load Balancing，OLB）、最小完成时间调度算法（Minimum Completion Time，MCT）、最小执行时间调度算法（Minimum Execution Time，MET）、Min-min 算法、Max-min 算法、Sufferage 算法、禁忌搜索算法（Tabu Search, Tabu）、模拟退火算法（Simulated Annealing，SA）、蚂蚁算法（Ant Algorithm，AA）、PSO 算法、遗传算法（Genetic Algorithm, GA）等。

2.2.1 常见启发式调度算法

网格环境下的任务调度问题是个 NP 完全问题。近年来，兴起将启发式算法应用于任务调度中，发现其能较好的解决复杂问题。下面将介绍几种比较常见的算法。

1. SA 算法

SA 算法是对金属退火过程的一种模拟，先对金属进行足够的加热，然后使其逐渐降温，而在金属加热的过程中，随着温度的不断升高金属内部的粒子趋于无秩序状态，相应的内能也会增大。而在逐渐降温的过程中，粒子会逐渐恢复秩序，在每个温度都会变得相对稳定，最后在 20℃时处于基态，此时金属的内能也最小。

在任务调度中[45]，SA 算法不断地从初始解开始，向其邻域展开搜索，最后获得一个新的解。新解可能得到的目标函数值较优，也可能恶化目标函数。对于优化解统统采纳，而对于差的解则采用 Metropolis 准则按 $exp(-\triangle E/T)$ 概率接受，作为下一次搜索的初始解。其中，$\triangle E$ 是目标函数的增量，T 是温度控制参数。

SA 算法具有较强的局部搜索能力，同时能以一定概率“突跳”出局部最优解，是一种全局搜索算法，但是搜索时间长，且不易得到最优解。

基于 SA 算法的求解优化问题一般流程如下：

（1）在可行解域内随机初始一个初始状态 x_0，根据效益函数 $f(x)$ 计算初始效益值，并初始化相应的初始控制温度 t_0；

（2）对当前状态 x_i 在可行解空间内进行一个随机扰动，得到新的状态 x_{i+1}，并计算相应的效益值 $f(x_{i+1})$；

（3）依据状态接收函数判定当前状态是否接受：若 $f(x_{i+1})<f(x_i)$，则接受新状态 x_{i+1}，否则按照 Metropolis 准则来判定对 x_{i+1} 接受，如果接受，则更新当前状态为 x_{i+1}；

（4）依据算法设定的收敛准则，判断搜索过程是否停止。是就转（5），否则转（2）；

（5）按照算法设定的降温策略降低控制温度 t；

（6）根据算法设定的收敛条件，判定退火过程是否结束，是就转（7），否则转（2）；

（7）输出当前解，即算法求得的最优解。

2. AA 算法

AA 算法是基于蚂蚁行为建立的一种启发式优化算法。在蚂蚁发现食物并往回搬时，就会在路上留下一种叫“外激素”的东西，其它蚂蚁一经闻到这种激素的“味道”，就会追踪这种气味去觅食，而且还会选择最短的路径来找到食物。

受现实生活中蚂蚁觅食过程的启发，AA 算法使用人工蚂蚁来模拟，利用信息素来不断优化当前解，使其能慢慢收敛于全局最优值。AA 算法有许多固有优点，如正反馈性、较强的鲁棒性、并行分布性等，是一种全局优化算法，非常适

合求解网格环境下的任务调度问题。

王莉等在文献[46]中将 AA 算法用于任务调度中的多目标优化问题，并取得较好的成果。

AA 算法求解最优化问题的一般流程如图 2-2 所示，以求解 TSP（旅行商问题）为例：

```
1)    初始化。随机放置蚂蚁，为每只蚂蚁建一个禁忌表 tabu_i，将初始
      节点放入禁忌表中；
2)    搜索迭代过程。
t=1;
while t<t_max do//执行迭代
   for i =1 to m do //对 m 只蚂蚁循环
      for j = 1 to n-1 do //对 n 个城市循环
         根据一定的策略，采用轮盘赌方法在窗口外选择下个城市 j；
         将 j 置入禁忌表中，蚂蚁转到 j；
      end for
   end for
   计算每只蚂蚁的路径长度；
   根据信息素更新策略对所有蚂蚁路径上的信息量进行更新；
t = t +1;
end while
3)    输出结果，算法终止。
```

图 2-2 求解 TSP 的 AA 算法

3．GA 算法

GA 算法使用一组 0-1 字串来描述问题的求解，这些 0-1 字串称为种群(Population)。每个 0-1 字串对应着生物进化理论中的染色体(Chromosome)，都可映射为目标区域内的一个潜在解。算法从初始种群开始，基于“适者生存”的规律从当前种群中“淘汰”不好的个体并保留适应自然变化的，然后不断利用交叉(Crossover)和变异(Mutation)算子来生成下一代更适应自然的种群，如此过程进行下去，直到符合一定的结束条件算法终止。

GA 算法由以下部分组成：（1）种群；（2）适应度函数，以适应度的大小来度量解的好坏；（3）遗传操作，主要包括有选择（Selection）、交叉以及变异三个基本算子。GA 算法具有相当不错的收敛性。图 2-3 显示了 GA 算法的求解问题的搜索过程。

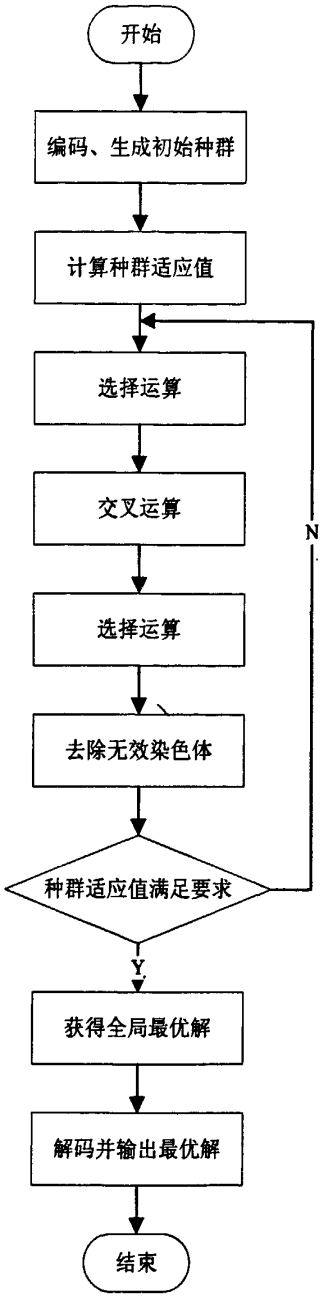

图 2-3 GA 算法求解过程

文献[47]将 GA 算法应用在网格任务调度中，提出一种改进 GA 算法的任务调度方法，对时间跨度为优化目标的任务资源映射问题有良好的效率和收敛性。

但是由于 GA 算法搜索过程是个随机性的过程，对求解质量不能确保，开销也比较大。

4. PSO 算法

PSO 算法是一种比较新的全局优化方法。算法最开始在可行解空间中随机的初始化一粒子种群，种群中的每个粒子都体现为优化问题的一个潜在、可行解。接着每个粒子在解空间中“飞行”，并有一个速度决定其“飞行”方向和距离。

通常粒子将追随当前的最优粒子“飞行”，经过迭代多次得到最优解。

PSO 算法是一种有效的解决网格任务调度问题工具，文献[48]基于独立任务提出一种局部模型的 PSO 算法网格任务调度策略，有效改善了任务调度的性能。

2.2.2 常用启发式算法性能比较

目前，对于网格任务调度问题有较多的任务调度算法。以时间跨度为优化目标的任务资源映射问题是个 NP 完全问题，解决此类问题的锐器是启发式算法。其中比较有效、常用的工具有，GA 算法、SA 算法、AA 算法和 PSO 算法。

GA 算法在处理复杂的组合优化问题方面具有先天的优越性，有很多种适用于调度问题的串并行 GA 算法，求解能力都较好。但是算法涉及的参数多，而且交叉和变异操作非常繁杂，实现起来比较困难，运行过程长。同时若所处理的组合优化问题约束也很复杂时，便难以找到与之相适应的操作算子。

SA 算法能以一定的概率接受差的解而可能“突跳”出局部极小，是一种全局最优算法。但是由于很难削弱算法对初始温度等参数的敏感依赖性，搜索时间比较长，因而很难适用到实时动态调度上；

AA 算法先天固有的正反馈机制使得其能避免早熟现象，但在实际问题求解过程中往往收敛于局部最优，而且算法复杂，搜索时间漫长；

与其他启发式算法不同，PSO 算法没有复杂的操作算子，而且搜索速度快、效率高，是求解优化问题的一种高效方法。但存在易陷入局部极值、搜索精度不高等问题。本文将充分融合 PSO 算法和混沌优化机制，实现不同优化行为互补，同时保证算法的全局收敛能力和收敛速度，提出一种新的 PSO 算法。

2.3 网格任务调度模型

网格资源的异构性、动态性以及网格应用对资源需求的差异性，同时网格环境中的资源稳定性、可用性、性能以及服务质量 QoS 各异，致使任务调度系统变得尤为复杂。如何合理调度应用程序到各种异构资源上，使任务的总执行时间和整个网格系统的吞吐性能得到有效保证，是网格任务调度的重点和难点。

影响调度的因子有很多，比如任务计算时间、通信开销以及数据分配策略等。本文主要考虑的是处理机的处理性能对任务调度长度造成的影响。

2.3.1 调度模型

调度模型是对调度问题及其对象的抽象化描述。针对不同的模型，各种调度算法的适应性会不同，效率也不一样。因而，在研究任务调度算法时要对调度问

题进行种种假定。通常，在网格中，任务调度模型由三个部分组成：网格应用、网格资源以及相应的调度策略，如图 2-4 就是一个典型的任务调度系统。现假设某虚拟组织 VO 中用户向该系统提交了一个大型应用程序，该任务调度系统首先经由任务划分模块将该应用程序划分为若干个子任务；然后调度模块从网格资源中的信息采集模块 MDS 收集一些必要信息，综合这些信息并按照一定的策略对这些任务进行分发；最后，任务在所分配到的资源节点上执行。

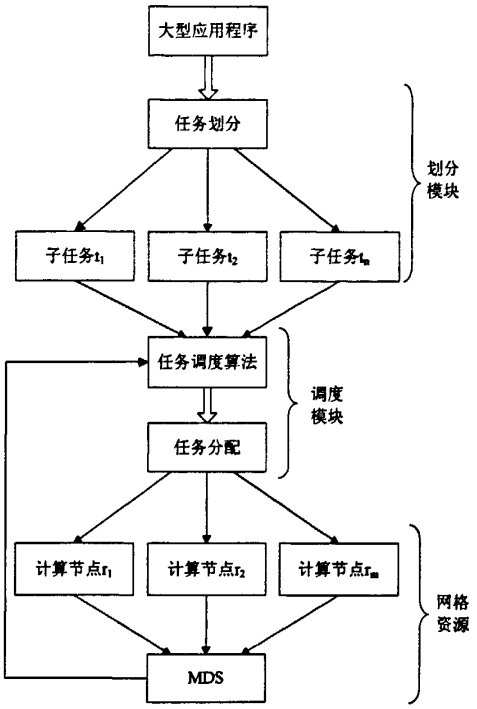

图 2-4 典型的任务调度系统

此类系统的一般目标是为了实现最优化 makespan，即使用一个调度算法，使得 makespan 最小，调度系统达到最优。本文将从网格环境下的独立任务调度问题来研究网格任务调度算法，实现最优化网格应用的完成时间。

在独立任务调度中，将一个应用划分为 N 个子任务，然后将其分配到 M 个处理机上执行，其目标是使这些子任务在 M 个处理机上的总执行时间最短。

处理机上的任务也称负载（workload）。在网格系统中，任务调度就是为了实现负载在处理机间能够均衡，任务得到合理有效地分配，使系统充分发挥高效的性能，提高整个系统资源的利用率。

任务调度一般关心两个问题：任务具体分派到哪个处理机上和处理机上所分派任务的执行顺序问题。而对于独立任务而言，执行顺序对应用的总执行时间可以忽略，自然顺序问题就不在考虑之列，故而我们只用考虑任务、资源如何进行映射即可。

2.3.2 算法度量指标

一般来说，网格任务调度考虑以下几项性能指标：

（1）优化跨度，即使 makespan 值最小，如式 2-7 所示。

$$min\ (\ makespan\ ) \tag{2-7}$$

（2）负载均衡

通常只要计算资源上的负载超出 L_{min} 就可认定该计算资源为重载资源，而当负载低于 L_{min} 时为轻载资源。任务调度的目标是最小化资源上的负载，使得各个资源负载基本相同，这个过程也称为负载均衡。负载均衡度 β 按式 2-8 计算，表征系统中计算资源整体负载偏离均衡负载 L_{min} 的程度，其数值越小，网格系统就越均衡。

$$\beta = \frac{1}{m}\sum_{1\le j\le m}(RC_j - L_{min})^2 \tag{2-8}$$

m 为资源数，RC_j 为资源 j 的运行时长。

2.4 本章小结

本章主要介绍了网格任务调度的相关内容，首先介绍了网格任务调度的特点、目标、过程，然后详细介绍了几种常用的启发式任务调度算法并分析其优缺点；最后介绍了网格调度模型，包括网格的优化目标及数学描述以及任务调度系统的几项性能评价指标，为第四章改进 PSO 算法的网格任务调度算法介绍了基础知识。

第三章 PSO算法及混沌优化

本章将从PSO算法原理、算法流程、主要参数、优缺点及改进策略等方面进行阐述，同时还将简要的介绍混沌优化相关概念，为第四章对PSO算法进行改进并应用于网格任务调度奠定基础。

3.1 最优化问题

最优化是指在一定时间内，寻找优化问题的最佳解决方案的过程。一个优化问题通常由候选解的集合来描述，可以用三元组进行表示：

（1）一个变量集合及其值域。一组变量代表着最优问题的一个可行解。

（2）一个约束集，用来约束可行解。

（3）一个目标函数，用来度量可行解的优劣。

最优化问题可分为两类：最大化和最小化问题。在本论文的网格任务调度中所讨论的是一个最小化问题，一般形式如下：

$$\begin{aligned} &\min\ f(X) \\ &s.t. g(X)=0, \\ &h(X)\geq 0, \\ &X\in S. \end{aligned} \tag{3-1}$$

其中，X为变量，S为可行解的值域，$f(X)$为目标函数，$g(X)$为等式约束，$h(X)$为不等式约束。$\min f(x)$是求目标函数的最小值。

根据变量是否连续，又可将最优化问题分为函数优化问题和组合优化问题。本文研究的网格任务调度就是个组合优化问题，即离散变量的最优化问题。因而，在用PSO算法处理这类问题时，要设计相应的离散PSO算法。

在优化理论研究领域中，最为有趣的研究成果就是“无免费午餐”，简称NFL定理[49]。该定理表明，对所有可能的函数集上，没有最优的优化算法，它们的性能是等价的。这意味着，没有任何算法比穷举搜索或纯随机搜索算法更好的算法。因而，寻找一个通用的最优问题解决方法是徒劳的。但是，在针对具体的优化问题附加一些特定的函数集，NFL则认为存在一个适用于该集合上的最佳优化方法。在求解复杂的优化问题时，可能大多数优化方法都不适用或者效果不理想，这时可以考虑使用进化算法如GA、AA、PSO等。这类算法不关心目标函数是否连续，且具有极强的适用性，因而将其用来解决复杂的优化问题有很大的现实意义。

3.2 PSO算法

3.2.1 基本原理

类似于其他进化算法，PSO算法[50]从生物进化中引入“种群”与“进化”，其搜索问题的过程是依据种群中个体的适应度而展开的。但是，在采用PSO算法中求解问题时没有用到进化算子，而是将所有个体看作是一个粒子种群，这些粒子的质量和体积均不用考虑，并以一定的速度飞行于n维搜索空间中。其中，飞行速度在粒子进化过程中会不断调整，调整的过程既有赖于个体经验，又和整个种群的经验息息相关。

设

$X_i=(x_{i1}, x_{i2},..., x_{in})$为种群中第$i$个粒子在$n$维空间里的位置；

$Vi=(v_{i1}, v_{i2},..., v_{in})$为种群中第$i$个粒子在$n$维空间的飞行速度；

$P_i=(p_{i1}, p_{i2},..., p_{in})$为种群中第$i$个粒子在$n$维空间中所经历的最好位置，也即是说粒子在此处的适应度是最好的，称为个体最佳位置。对于最小优化问题，目标函数值越小越好。

假设$f(x)$为最小化的目标函数，则可以依据式3-2来找到粒子i的当前最好位置：

$$P_i(t+1)=\begin{cases} P_i(t), if\ (f(X_i(t+1))\geq f(P_i(t))) \\ X_i(t+1), if\ (f(X_i(t+1))<f(P_i(t))) \end{cases} \quad (3\text{-}2)$$

设种群的规模大小为n，整个种群所经历过的最好位置为$P_g(t)$，也叫全局最佳位置。$P_g(t)$可描述为式3-3所示：

$$P_g(t)\in\{P_0(t),P_1(t),.....,P_n(t)\}\,|\,f(P_g(t))=\min\{f(P_0(t)),f(P_1(t)),.....,f(P_n(t))\} \quad (3\text{-}3)$$

基于以上定义，基本PSO算法的进化方程式可这样描述[45]：

$$v_{ij}(t+1)=v_{ij}(t)+c_1r_{1j}(t)(p_{ij}(t)-x_{ij}(t))+c_2r_{2j}(t)(p_{gj}(t)-x_{ij}(t)) \quad (3\text{-}4)$$

$$x_{ij}(t+1)=x_{ij}(t)+v_{ij}(t+1) \quad (3\text{-}5)$$

式3-4中$v_{ij}(t)$、$v_{ij}(t+1)$表示第i个粒子在求解空间中第j维方向上的当前速度和更新后的速度；式3-5中$x_{ij}(t)$、$x_{ij}(t+1)$表示第i个粒子在求解空间中第j维方向上的当前位置和更新后的位置；学习因子c_1、c_2通常在[0，2]间取值，是两个随机值；r_1、r_2为两个无相互影响的随机函数，值域[0，1]；$p_{ij}(t)$表示第i个粒子在求解空

间中第j维方向上经历过的最好位置矢量即个体最好位置；$p_{gj}(t)$表示群体在求解空间中第j维方向上经历过的最好位置矢量即全局最好位置。

从上述进化方程可以看出，粒子向$p_{ij}(t)$和$p_{gj}(t)$移动的步长分别由c_1、c_2来动态进行调节。为了避免粒子在进化过程中，超越搜索空间的边界，对v_{ij}通常会有限定范围，如$v_{ij}\in[-v_{max},v_{max}]$。若问题搜索的目标区域为$[-x_{max}, x_{max}]$，$v_{max}$与$x_{max}$的关系可以表示为：$v_{max}=k*x_{max}$，其中$k\in[0.1, 1.0]$。

3.2.2 算法流程

在求解问题时，算法的流程一般如下：

步骤1，设置相关参数，对种群的位置和速度进行初始化；

步骤2，对种群中每个粒子的适应度进行评估；

步骤3，将每个粒子的适应度与其最佳位置P_i的适应度相比较，若更优，则更新P_i；

步骤4，将每个粒子的适应度与种群最好位置P_g的适应度相比，若更优，则更新P_g；

步骤5，根据式3-4、3-5更新粒子的速度和位置；

步骤6，如果满足结束条件则退出，通常是当适应度较优或是进化代数达到预设阀值t_max，否则返回步骤2。

3.2.3 标准粒子群优化算法

为使基本PSO算法的收敛效率有所改进，Y. Shi与R. C. Eberhart首次在速度更新方程中用到惯性权重[51-52]，即：

$$v_{ij}(t+1)=wv_{ij}(t)+c_1r_{1j}(t)[p_{ij}(t)-x_{ij}(t)]+c_2r_{2j}(t)[p_{gj}(t)-x_{ij}(t)] \quad (3\text{-}6)$$

式3-6中w称为惯性权重，可见，基本PSO算法属特殊情况，即w等于1时。w可以用来维持粒子的运动惯性，使其能在搜索空间中进行扩展，并能充分发现新的区域。

将w引入算法，可削弱v_{max}对基本PSO算法的影响。因为无论在算法的全局搜索能力还是在局部搜索方面，w都有一个相当不错的平衡效果。这样，如果v_{max}比较大，可通过减少w来调节算法的搜索。同时w也会影响到算法的迭代次数，随着w的增大，求解问题的进化代数也会更多。

对于一个全局搜索算法来说，在算法前期和后期采取的策略是不同的，前期主要是尽量获取质量更好的粒子，而在后期要让算法能较快地收敛，因而需要不断提升开发能力。最直接有效的方法是，让w随着种群的不断进化而线性减少，

比如由0.9递减到0.4等等。Y. Shi和R. C. Eberhart的仿真实验结果也证实了w线性减少能使算法取得相当不错的实验结果[53]。

目前，许多对PSO算法的研究大多都是基于带惯性权重的PSO算法进行的。为此，标准PSO算法都是带惯性权重的，而最初始版本为基本PSO算法，我们通常所说PSO算法均指标准PSO算法。

3.2.4 算法参数

1．惯性权重

在PSO算法中，惯性权重w体现的是粒子上一代的速度对当前速度的影响程度，权衡制约着算法的全局和局部搜索能力。当w取值比较大时，会使算法的全局寻优能力在得到加强的同时降低局部搜索性能。反之，则增强算法的局部寻优能力，而全局寻优性能会有所下降。

w不宜设置太大，过大容易错过最优解，导致算法无法收敛，或者不能在最优解处收敛。而在大多数收敛的情况下，是因为种群中的所有粒子都朝着最优解的方向飞去，整个种群的速度和位置都趋向同一化，使得算法在后期收敛效果不明显，而且求解达到一定精度时，优化进程变得尤为缓慢。

显而易见，w的选取是否恰当至关重要。好的w不仅可以对算法的全局和局部搜索能力有个很好的权衡效果，而且会大大缩短算法找到最优解的进化代数。然而合适的w值并不是一直不变的，针对所求解的问题不同，会有截然不同的w值，因而要具体问题具体分析。通过大量的实验发现，w的值如果在[0.9，0.2]之间一般来说会使算法取得比较好的性能，同时w的值如果从1.4线性减少到0要比固定的值好。这是由于算法在搜索前期以一个较大的w找到的范围比较好，而在后来采用较小的w可以增强局部搜索能力[54]。此外，文献[55]中就最大速度和w对PSO算法性能的影响进行分析，发现在最大速度V_{max}比较小时，w取值近似为1是个不错的选择。而当最大速度比较大时w取值为0.8是个相当好的选择。当最大速度与搜索空间的宽度相当时，权重最适合的初始值取0.8。一般的情况下，对w的值应是从0.9线性递减到0.2。

2．学习因子

学习因子c_1、c_2用来控制粒子自身的记忆和整个种群的记忆之间的相对影响，代表着粒子朝个体极值p_{best}和全局极值g_{best}飞行的加速比。c_1、c_2取值较小时，可以允许粒子先在目标区域外“徘徊”而后再飞回，而较大的取值则会导致粒子快速地落在目标区域，同时也极为可能飞到边界之外。

（1）当$c_1=c_2=0$时，粒子将匀速运动，直至飞到问题空间的边界，此时搜索的区域非常有限，故而很难找到求解问题的最优解。

（2）当 c_1=0 并且 c_2≠0 时，粒子失去了自我“认知”能力，而只有 “社会”经验的学习。在此种情形下，粒子的行为只受到其他个体的影响而无自身的“认知”过程，这使得算法能够进入到其他搜索空间进行探索，算法收敛性能也有所保障。但当问题变得繁杂时，很容易出现“停滞”现象，获取全局最优的可能性相当小。

（3）当 c_1≠0 并且 c_2=0 时，粒子相互间没有任何的社会信息共享，亦即只有自我“认知”。此时由于个体彼此之间没有交互作用，一个规模为 N 的群体等价于 N 个粒子相互独立的简单运动，因而得到解的几率相当小。据研究表明，在算法的求解初期应该具备较强的全局搜索能力，故此 $c_2 \geq c_1$,而在求解后期应该增强算法的局部搜索能力，故 $c_2 < c_1$。

3．最大速度

一般来说，最大速度 V_{max} 宜取值为粒子的搜索空间的宽度，而不应该超过此值。如果 V_{max} 太大，虽然粒子的探索能力有所增强但容易飞过最优解的位置，反之如果较小，可以增强粒子的开发能力，却使粒子不能很好的在局部区间之外进行足够的探索，致使粒子的全局搜索能力有所下降。

在合理的范围内，最大速度对算法性能的影响非常有限。但是，在求解多峰函数时，过小的 V_{max} 会大大影响粒子的全局搜索性能，算法一旦陷入局部极值则无法跳出[55]。

4．种群规模

种群规模越大，协同进行问题搜索的粒子也就越多，PSO 算法的搜索能力将有所增强。从算法的计算复杂度方面考虑，群体规模越大，所需要的计算时间会大幅增加，当然同时会增加算法的可靠性。因而，在对种群规模大小进行选择时，应从算法的计算时间和可靠性两个方面综合考虑。通常对于一般的问题求解，这个参数取值 20 就足够了，而当问题相对比较复杂时要具体问题具体对待，种群规模会多于 50。

3.2.5 PSO 算法的优点与缺陷

PSO 算法在处理最优化问题时，找到全局最优解概率较大，且与传统随机方法相比计算效率更高。它的最大优势在于算法结构简单易行、可调参数少、收敛速度快。

尽管如此，在问题搜索过程中，PSO 算法还是存在不足，主要有两个方面：

（1）为使初始种群能够均匀分布于目标区域，可以通过随机性质的初始化来实现，但是种群质量如何则难以保证，极为可能出现的情况是：存在一部分与最

优解相距甚远的个体。如果初始种群较好的话，将对求解效率和质量的提升起到很好的促进作用。由于初始化和进化过程都具有一定的随机性，这使得 p_{best} 和 g_{best} 的更新失去明确的目标，大大制约算法的收敛性能。

（2）个体速度和位置的更新本质上依赖于自身信息、个体极值信息和种群极值这三个信息。这个过程具有正反馈性质，当前两个信息占据优势时，算法往往进行到局部最优时就停滞下来。而当某些粒子的位置及其 p_{best} 接近群体的 g_{best} 且 w 小于 1 时，速度越来越小，逐渐趋近于零，粒子的“惰性”此时尽显无疑。随着进化的不断进行，其他粒子也会很快聚集在这些惰性个体的周围，致使算法过早地收敛，难以得到最优解。

由于种群的多样性直接影响 PSO 算法的收敛性能，通过保持种群的多样性无疑可以提高算法全局搜索能力。而实际上粒子会在问题过程中逐渐丧失多样性，同时对种群盲目提高多样性会大大影响算法的收敛速度，因而设计一个合理的策略来保持粒子多样性，对于保证 PSO 算法的全局收敛性能至关重要。

本文的改进策略是通过融合混沌优化机制，降低算法对初值的强依赖性，适当提高算法的多样性，同时还保留了 PSO 算法的易实现，速度快等优点，详细改进步骤将在第四章介绍，接下来要介绍的是几种常见针对 PSO 算法的改进方法及 PSO 算法的一般设计步骤。

3.3 PSO 算法的改进策略和设计步骤

3.3.1 PSO 算法的改进策略

PSO 算法自提出之后，其最突出的优点是收敛速度快、算法简单，迅速被广泛应用于各大优化领域。但是在运用于实际时发现 PSO 算法易“早熟”，并且求解精度比较低。基于此，众多学者对 PSO 算法进行了相当深入的研究，提出了很多行之有效的改进方法和策略。以下将简要地介绍几种著名的改进策略[56-58]。

1．带有惯性权重的改进 PSO 算法

由于求解的问题不同，全局搜索能力与局部搜索能力的比例关系也不相同，如何确定比例对问题求解过程至关重要。甚至在同一个问题的进化过程中，对比例的要求也会不相同。鉴于此，Yuhui Shi 将惯性权重 w 引入粒子群优化算法以改进算法[62]。改进算法的进化方程如式 3-7 和式 3-8。

$$V_i(t+1)=wV_i(t)+c_1r_1(P_i-X_i(t))+c_2r_2(P_g-X_i(t)) \quad (3-7)$$

$$X_i(t+1)=X_i(t)+V_i(t+1) \quad (3-8)$$

当 w=1 时，式 3-7 与式 3-4 相同，显然带惯性权重的 PSO 算法是原算法的一

个特例。文献[56]所建议的 w 取值范围为[0,1.4]，但通过实验发现当 w 取值在[0.8,1.2]内时，算法取得的收敛效果更好，而当 w>1.2 时，则极有可能陷入局部极值。

w 表征的是粒子维持其原先的速率的程度。假设有某个粒子 j，其最开始的速率不为 0，当 c_1=c_2=0 且 w>0 时，则粒子将会加速到 v_{max} 为止；当 w<0，则粒子会一直减小，直至 0。当 c_1、c_2 取值都不为 0 时，这样情形变得比较繁杂，而文献[58]的实验结果发现，w 比较适宜的取值是 1。

当 w 比较大时算法会有较好的全局收敛性能，而较小的 w 则能增强算法的局部收敛能力。所以， w 应该伴随问题求解的不断进行而逐渐变少，从而可以充分权衡全局搜索和局部搜索，大大增强 PSO 算法在迭代过程的前后期的性能。在文献[56]中， w 满足

$$w(t)=0.9-\frac{t}{t\_\max}\times 0.5 \qquad (3\text{-}9)$$

式 3-9 中，t_max 为最大迭代次数， $w(t)$的值为 w 在第 t 代时的值，显然 w 将随着算法的不断进化从 0.9 线性减少到 0.4，从文献[50]仿真的实验结果来看，效果很好。

2．带收缩因子的 PSO 算法

Clerc 在研究[57]中，对收缩因子做出了定义，同时针对如何选择 w、c_1 和 c_2 的值还进行了讨论。通过正确的选择这些控制参数，$v_{i,j}$ 也就没必要严格限制在[$-v_{max}$, v_{max}]内。如果这些控制参数选取恰当，就可去除没有 $v_{i,j}$ 必须在 [$-v_{max}$, v_{max}]内的限制。下面将介绍一个带收敛因子的 PSO 算法的收敛模式特例。针对收敛模式改进的速度更新公式如下：

$$v_{i,j}(t+1)=\chi\left(v_{i,j}(t)+c_1r_{1,j}(t)(p_{i,j}(t)-x_{i,j}(t))+c_2r_{2,j}(t)(p_{g,j}(t)-x_{i,j}(t))\right) \qquad (3\text{-}10)$$

式 3-10 中，

$$\chi=\frac{2}{\left|2-\ell-\sqrt{\ell^2-4\ell}\right|} \qquad (3\text{-}11)$$

且 ℓ=c_1+c_2，ℓ>4。

设 c_1、c_2 的值均为 2.05，将 ℓ=4.1 代入式 3-11，可得收缩因子 χ 的值为 0.7298 并代入式 3-10，同时将参数 t 省略，可得如下结果：

$$v_{i,j}(t+1)=0.7298\times\left(v_{i,j}+2.05\times r_{1,j}(p_{i,j}-x_{i,j})+2.05\times r_{2,j}(p_{g,j}-x_{i,j})\right) \qquad (3\text{-}12)$$

由于20.5×0.7298=1.4962，所以，式3-12等价于PSO算法在c_1、c_2为1.4962，w为0.7298时的速率更新公式。

Eberhart和Shi对分别使用了v_{max}和χ的两种算法版本，并分析比较其性能，研究证实，后者的收敛性能一般比前者更优。但是通过对一些测试函数进行实验后发现，经过一定的迭代次数后者仍然无法得到全局最优点。Eberhart和Shi认为，那是由于粒子与目标搜索空间偏离太远导致的。他们建议在对收缩因子的使用时一开始就对算法的参数进行限定，如将速率参数v_{max}设置为x_{max}，或是将搜索空间大小首先预设好，这样可以有效降低以上的不利影响。

实验证明，通过以上对算法的限定可以使算法在对求解问题的性能有一个较大的改进，无论是算法的收敛速度还是其搜索能力。

3．基于遗传思想的改进PSO算法

在基本粒子群优化算法中，隐含着一个选择机制就是对每个粒子的最佳位置的确定，因此， Perter J.Angeline于1998年将粒子群算法中引入显示的选择机制来改进基本算法[58]，仿真结果发现改进算法在收敛速度和算法的全局搜索速度都能兼顾到。

算法使用一种称为锦标赛选择算子（Tournament Selection Method）的选择算子，然后基于每个个体的当前位置计算其适应度，与其余的个体进行比较，并纪录下最差个体。依据此记录对种群进行排序，将得分最高的放置在种群的头部。

以下是算法的简要流程：

（1）从种群中随机选出一个个体。评估其适应度，并与种群中的其余个体的适应度一一比较。在每次比较时，只要此个体更优则对该个体加上一分。依此过程遍历完所有个体；

（2）对种群按得分高低进行排序；

（3）复制种群中前一半得分较优的个体，并将其用来替代种群后半部分较差个体。在这个过程中并未改变最佳个体的适应度。

与较传统的PSO模型相比，在对解决特定的问题求解时，上述的改进型PSO模型均得到一定的改善。这也说明在特殊的优化问题求解时我们可以对传统的PSO模型作出相应的改进可以获取满意的优化结果。

3.3.2 PSO算法的设计步骤

在采用PSO算法求解特定问题时，一般遵循以下的设计步骤如图3-1所示。

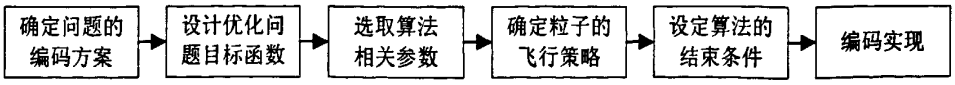

图3-1 PSO算法的一般设计步骤

3.4 混沌优化简介

在确定性系统中有这样一种现象，它表面上是没有规则的、实则内含一定规律的、且是伪随机的，这类现象被称之为混沌[59]。混沌所描述的现象是没有次序、杂乱无章的或是没有规则可言的，它说明了系统内部相当复杂，而且是随机的和没有秩序的。而混沌运动描述了系统中的一种运动，这种运动极其不稳定且受限于空间大小，因而系统固有的无序性抑或随机性可借由这种运动来体现。

3.4.1 基本原理

混沌优化方法新颖、有效，借助于混沌系统遍历这一优良特性使得其能求得全局最优，而且目标函数也不必一定具有连续性或可微性。我们可以充分利用混沌运动特有的性质来进行优化搜索。求解问题的一般过程是，首先生成一组混沌变量，变量数等于待优化目标的个数，接着将其映射到待优化目标的值域范围内，同时将待优化目标的取值范围作为混沌运动的遍历范围，然后直接在混沌变量的基础上来进行搜索。混沌运动的特性有很多，如随机性、遍历性等，这无疑使得混沌搜索技术在随机搜索方面表现良好。

基于混沌的搜索技术主要思想是：

（1）采用某种迭代方式产生混沌序列，通常使用的是 Logistic 方程，Logistic 方程是一个典型的混沌系统。其表达式如式 3-13 所示：

$$u_{n+1} = \mu u_n(1-u_n), n = 1,2,3,\cdots \qquad (3\text{-}13)$$

式中：μ 为控制参数，$0 \le \mu \le 4$；$u_n \in [0,1]$。

设 $0 \leqslant u_0 \leqslant 1$，当 μ 取值为 4 且 $u_0 \notin \{0,0.25,0.5,075\}$ 时，系统的状态就属于完全混沌[60]。

（2）通过载波方式，将混沌序列映射到优化空间中，也即是将混沌变量的值域通过式 3-14“放大”到优化变量的值域范围内。

$$x_{ni} = a_i + u_{ni}(b_i - a_i) \qquad (3\text{-}14)$$

3.4.2 主要特征

混沌产生于确定性非线性动力学系统，是一种非周期有界动态行为，其对初

始条件是极具敏感性的。混沌运动是确定性系统中产生的一种类随机过程。混沌具有的主要特征如下[61]：

（1）对初值的敏感依赖性。

亦称之为蝴蝶效应，也就是说，当系统的输出呈现为混沌状态时，即使任意靠近的各个初值状态，在经过一定时间的推移，将表现出各自独立的演化。混沌的这一特性反映了混沌序列对初值的极端敏感性；

（2）随机性。混沌表现得和随机变量一样杂乱；

（3）遍历性。用不着重复，混沌就能遍访空间内所有状态；

（4）规律性。产生混沌所用到的系统是一个确定性方程。

3.5 本章小结

标准PSO算法是一种随机的、并行的优化算法。它的优点是对被优化函数没有严格的限制，如优化函数具备可微、可导、连续等性质。它的突出优势是较快的收敛速度，简单的算法以及容易编程实现。然而，标准PSO算法的缺点则在求解如有多个局部极值点的函数时，容易收敛于局部极值点处，求解质量不是很高。此外，由于缺乏与之相配合的精密搜索方法，标准PSO算法通常是得不到比较精确的结果的。同时，虽然标准PSO算法拥有全局搜索算法的特性，但并不能保证它一定收敛于全局最优点。因而，标准PSO算法局限于求解高维的且有多个局部极值点但同时对精度要求不是很高的优化问题。所以在解决实际问题时，要针对具体问题设计出相应的改进PSO算法，以寻求较为满意的求解结果。

本章首先介绍了最优化问题的形式化定义和NFL定理；然后比较详细阐述了PSO算法的基本原理，介绍了PSO算法的标准版本，分析了其优缺点，并给出了PSO算法求解问题时的一般步骤和几种常见的改进方案；最后，简要的介绍了混沌优化的相关知识。

第四章 基于改进PSO算法的网格任务调度算法

本章将采用混沌优化思想，将混沌的遍历特性充分融入到 PSO 算法中，提出一种改进的 PSO(MCPSO)算法，从而使算法有效规避陷入局部最小的现象，降低算法对初值的敏感性，提高算法的收敛速度和求解精度。然后将其应用于网格环境下的独立任务调度中，以改进应用程序的性能和网格系统的性能。

4.1 任务调度问题定义

网格环境相比分布式环境和并行计算环境来说更为复杂，其资源具有动态性和异构性以及通信延迟的不确定性等特点。因而，在网格环境中，重要的一环是任务调度，实现任务的合理分配及资源的高效使用有赖于一个良好的调度系统。

一个任务调度系统的优劣取决于其是否具备以下条件：

（1）动态适应性。调度系统应能及时捕捉到系统中各种资源的最新信息，然后处理对资源的访问控制，这些资源主要是 CPU 资源、存储资源和网络资源；

（2）可扩展性。调度系统的可扩展性是指随着实际环境变化所使用的调度算法能动态调整；

（3）具备有效处理大量任务的能力。

由于实际网格环境中任务调度参数比较繁杂，故在此做如下假设：

（1）给定大型应用程序由若干独立任务组成，且这些任务是异构的，即各个任务的执行时间是不相同的；

（2）采用批模式；

（3）任务数远大于资源节点数；

（4）各个任务在资源节点上的运行时长是可预知的；

（5）只考虑任务的计算资源消耗，即任务在处理机上的处理时长；

（6）计算资源为同构资源，任意一个任务在所有计算资源上执行时间是相同的。

综上假设条件，网格环境下独立任务调度问题可定义如下：

（1）n 个任务的任务量集合 $T=\{T_1,T_2,\ldots,T_n\}$；

（2）m 个资源的集合 $R=\{R_1,R_2,\ldots,R_m\}$；

（3）n 个任务映射到 m 个节点的映射方案 $T\rightarrow R$ 集合即任务映射集 MAP。集合 $MAP=\{map_1,map_2,\ldots,map_n\}$，其中 map_i 表示第 i 个任务映射到的节点 map_i 上执行；

（4）资源节点的处理时长 RC_j 为第 j 个资源处理完所分配给它的所有任务

所需的时间。如式 4-1 所示。

$$RC_j = \sum_{1 \le i \le n} \begin{cases} T_i / R_j, if(j = map_i) \\ 0, if(j \neq map_i) \end{cases} \quad (4\text{-}1)$$

（5）时间跨度 makespan，是任务集 T 中第一个任务开始执行到最后一个任务完成的时间。在某调度算法下的 makespan 也称为该算法的调度长度。makespan 越小，调度策略越好。其计算公式如 4-2 所示。

$$makespan = \max_{j \in \{1,2,\cdots,m\}} \{RC_j\} \quad (4\text{-}2)$$

本文研究的是以最优跨度为优化目标，即 min（makespan），实现任务在资源上的总执行时间最短。

4.2 PSO 算法改进及关键技术

4.2.1 粒子编码和解码

针对网格环境下任务调度的特点，我们将采用（任务，资源）对应的浮点数编码方式，对各个任务所分配的资源进行编码，每个粒子位置矢量和速度矢量的维度大小都等于总任务数。各个粒子的位置矢量就是一个任务的调度方案，每个维度的编号代表任务编号，而该维度的数值则可映射为相应的资源编号。

例如，对于网格环境中 n 个任务 m 个资源的调度，粒子的位置和速度均用 n 维向量表示，维数和任务数一致。如某粒子的位置向量$(x_1,x_2,\ldots,x_n)$，就是一个任务的调度方案，其中第 i 维分量 x_i 表示任务 t_i 的权值，是$[0,m)$之间的一个随机数，对该权值取整后加 1 后就得到了该任务所分配的资源序号。当 n=10，m=5 时，各个节点对应的权值范围如表 4-1。

表 4-1 权值与节点序号对应表

权值	[0,1.0)	[1.0,2.0)	[2.0,3.0)	[3.0,4.0)	[4.0,5.0)
节点序号	1	2	3	4	5

若某粒子的位置矢量表示如下：

Particle：(2.5, 1. 8,0.6,2.4,4.8,3.2,0.7,1.9,2.5,4.0)

上述粒子即为任务的一个调度方案，由表 4-1 可得任务与节点的映射关系如表 4-2 所示，如任务 1 被分配到节点 3 上执行。

表 4-2 基于节点编码的任务分配方案

任务序号	位置分量	节点序号
1	2.5	3
2	1.8	2
3	0.6	1
4	2.4	3
5	4.8	5
6	3.2	4
7	0.7	1
8	1.9	2
9	2.5	3
10	4.0	5

由于以上的编码方式本身就包含调度中的任务和资源信息，因而无需特殊的解码操作。这里主要是因为研究任务的特殊性，任务是原子的且执行顺序对调度结果无任何影响，所以此编码解码方案比较适合。

4.2.2 适应度函数

粒子位置的优劣由粒子的适应度直接决定，引导粒子向更好的位置飞行，使 PSO 算法能够不断靠近最优解。结合本文研究的目标，最优化 makespan，我们将 makespan 作为算法的适应度函数 *Fit*。显然，粒子 *i* 的适应度 *Fit*(*i*)越小，粒子就越好。当粒子 *i* 进行位置和速度更新时，根据粒子 *i* 的当前位置矢量(x_1,x_2, ...x_n)计算出粒子的适应度 *Fit*(*i*)；然后，依据粒子的个体极值 p_{best} 、*Fit*(*i*)和种群极值 g_{best} 来对粒子的速度进行更新，使粒子飞向更好的位置。

4.2.3 种群初始化

在 PSO 算法中，种群的初始化过程具有一定的随机性，难以确保个体的质量，无法让初始种群能够均匀地分布于解空间中，使得种群中某些粒子可能远离最优解。这样大大降低了求解效率和初始解的质量，同样也致使 p_{best} 和 g_{best} 的更新具有相当大的盲目性，直接影响算法的快速收敛。

针对以上问题，这里我们充分利用混沌的遍历性特性，采用混沌序列对粒子的位置初始化。同时混沌自身还具有随机性本质和遍历性特征，使得算法初始化时没有失去随机性的特性，通过混沌可以大大提高初始种群的多样性和粒子搜索的遍历性。

种群初始化即混沌初始化具体步骤如下：

（1）初始化一个(0，1)间的随机数 u_0；

（2）以 u_0 为初值，由 Logistic 方程即式 3-13，式中 μ 取值为 4，迭代生成序列 u_i；

（3）重复（1）直至迭代次数为 2*POPSIZE，POPSIZE 为种群规模大小；

（4）通过载波的方式即式 3-14，式中 b_i 取值为 X_{max}，a_i 取值为 X_{min}，将序列 $u_1,u_2,u_3,\ldots,u_{2*popsize}$ 中的各个变量映射到优化空间中，即把混沌变量的值域“放大”到优化变量的取值范围$[X_{min},X_{max}]$内，得到 2*POPSIZE 个位置并计算相应的适应度；

（5）选出适应度最好的 POPSIZE 个位置作为初始种群的位置并随机初始化种群个体的速度，然后令粒子的个体极值 p_{best} 为当前粒子的适应度，粒子的最优位置 $p_best(i)$为当前粒子的位置；同时，计算出粒子的种群极值 g_{best} 并令具有最小适应度的粒子位置为全局最佳位置 g_Pos。

混沌初始化流程如图 4-1 所示。

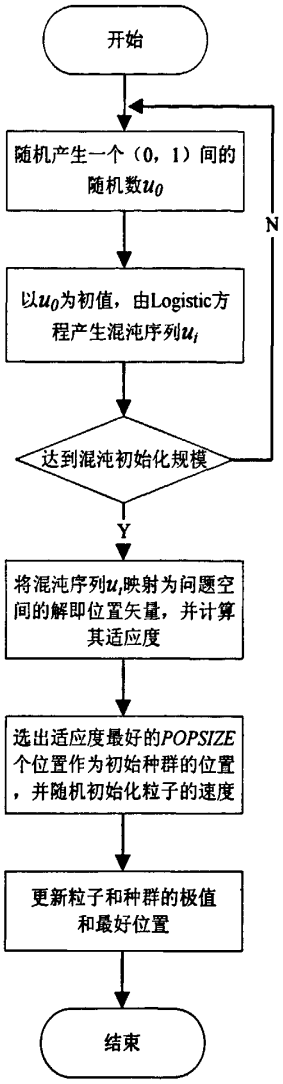

图 4-1 混沌初始化流程图

对粒子的位置使用混沌序列初始化，在产生的大量初始位置中择优选出算法的初始种群位置，这样不但可以丰富种群多样性使粒子均匀分布于解空间中，而且有效加速了算法搜索速度，优化了初始解的质量。

4.2.4 混沌优化搜索

群体中粒子的多样性直接影响 PSO 算法收敛性能，是确保算法收敛到全局最优点的一个重要因素。但是在算法的迭代过程中，群体的多样性在逐渐丧失，趋于一致。虽然这样收敛速度较快，但是容易早熟收敛，在没到达全局最优点之前就已经停滞于一点。这个点可能是局部极小值点，甚者只是其邻域的一个点。

在种群的进化过程中，不断丰富其多样性，可以有效避免种群的“早熟”，引导种群不断向全局最优搜索。这里，在算法每一次迭代更新中，对当前粒子的

历史最优位置进行“扰动”即在当前粒子的历史最优点进行混沌搜索。将所搜索到的最优点与个体历史最优点进行比较，如果更优则进行替换。这样使个体摆脱“惰性”，向最优点“飞行”。

由于混沌优化搜索和混沌初始化十分相似，下面就简要的介绍其流程：

（1）初始化一个(0，1)间的随机数 u_0；

（2）以 u_0 为初值，由 Logistic 方程迭代生成序列 u；

（3）将序列 u 通过“载波”的方式放大为优化空间的一个位置矢量 *cs_Pos* 并计算其适应度。若其适应度最好，即优于搜索到的最优位置 *cs_Pbest*，则更新最优位置 *cs_Pbest* 及其适应度 *cs_gbest*；

（4）重复（1）直至最大搜索次数 *Iter_csmax*；

混沌优化搜索的流程结构如图 4-2 所示。

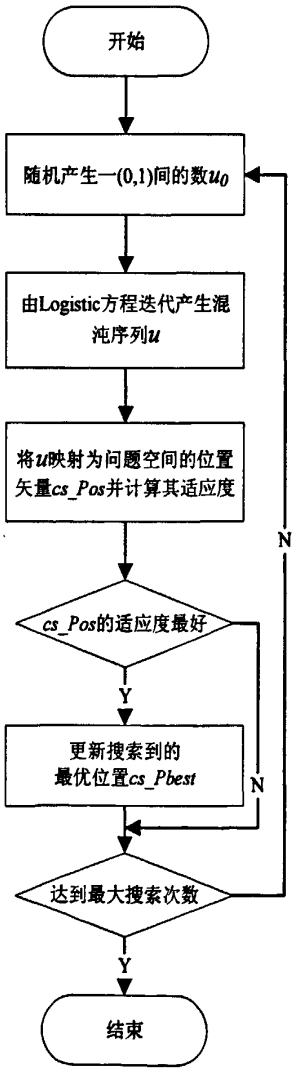

图 4-2 混沌优化搜索流程结构图

4.2.5 边界处理

在粒子迭代的进化过程中，很难避免粒子不会越界，这里我们将对粒子进行随机处理。当粒子的位置超出问题空间时，则对粒子的位置重新进行设定，随机初始一个位置矢量且每个分量的值都在问题解的值域$[X_{min},X_{max}]$内，从而保证每次搜索的有效性。

同时，由于本算法设定的位置值不小于零，因而为了粒子每次的搜索都有意义，我们可以对粒子的速度最大最小值进行设置，最小值我们设置为 0，而最大值则等于位置分量的最大值 X_{max}。这样就可以使粒子的飞行区域与解空间$[X_{min},X_{max}]$相一致，大大减少了无效搜索的次数，增强 PSO 算法的搜索效率。

4.3 基于改进 PSO 算法的网格任务调度算法

针对 PSO 算法易早熟求解质量不高的问题，实现最小化任务完成总时间(makespan)，本文提出了一种求解独立任务调度的改进 PSO 网格调度算法 MCPSO (An improved PSO of grid scheduling algorithm under the meta task)。该算法的主要思想是：首先采用混沌序列初始化粒子的位置，在产生的大量位置矢量中择优选出初始种群的位置矢量；然后在粒子“进化”时，引入混沌搜索，将搜索到的最优混沌序列转换为问题空间的位置矢量并和当前粒子最优位置比较。如果优于当前粒子，则更新当前粒子的最优位置，引导当前粒子跳出局部最优点，快速寻找最优解。

该算法流程结构如图 4-3 所示。

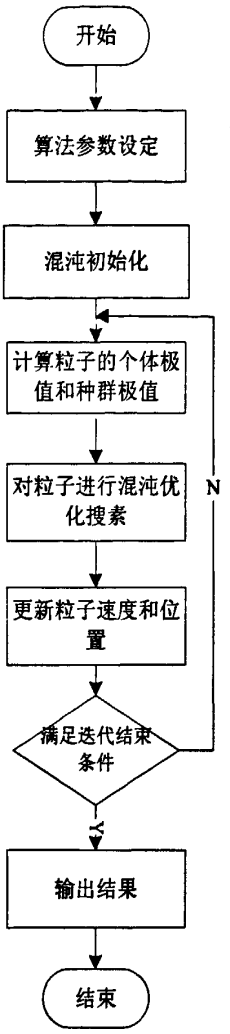

图 4-3 MCPSO 算法总流程图

MCPSO 算法求解步骤如下：

步骤 1 算法相关参数设定；

给定种群大小 *POPSIZE*，最大迭代次数 *Iter_max*，惯性权值 *w*，学习因子 c_1、c_2，混沌控制参数 μ，最大混沌搜索次数 *Iter_csmax*，粒子飞行空间 X_{max}、X_{min}，最大速度 V_{max}、最小速度 V_{min}，算法运行次数 R_{sum}。

步骤 2 种群初始化；

由于在 4.2.3 中对种群初始化已做介绍，因而不再详细说明。通过混沌初始化产生 2**POPSIZE* 个位置，依据适应度函数 *Fit* 计算公式即式 4-2 计算各个位置 *i* 的适应度 *Fit*(*i*)。由于适应度越小，位置就越好。因而这里我们选择适应度函数值较小的前 *POPSIZE* 个位置作为初始种群的位置；然后，对于每个粒子 *i* 的速度 *v*(*i*)都分别随机初始化：速度中的每个分量都是一个$[V_{min},V_{max}]$的随机值；最后，将每个粒子 *i* 的最佳位置 *p_best*(*i*)和最佳适应度 p_{best} 分别设置为粒子的当前位置 *Pos* (*i*)和当前的适应度 *Fit*(*i*)，同时如果此粒子比种群当前最佳粒子更好即拥有更

小的适应度，则种群最佳位置 g_Pos 和种群最佳适应度 g_{best} 分别更新为 $p\_best(i)$ 和 $Fit(i)$。

步骤 3 计算各个粒子的个体极值和种群极值；

（1）若粒子 i 的适应度 $Fit(i)$优于个体极值 p_{best}，则更新 p_{best} 和粒子极值点 $p_best(i)$为粒子的当前的适应度 $Fit(i)$和当前位置 $Pos\ (i)$。

（2）若粒子 i 的适应度 $Fit(i)$比种群极值 g_{best} 更优，则更新 g_{best} 和种群极值点 g_Pos 为 $p_best(i)$和 $Fit(i)$。

步骤 4 混沌优化搜索；

由于在 4.2.4 中混沌优化搜索进行了详细介绍，故此只做简要阐述。对每个粒子进行混沌优化搜索，最大搜索次数为 $iter\_csmax$。对每次搜索到的位置计算其适应度，若比粒子的最佳适应度 p_{best} 更小，则将此位置重新作为粒子极值点 $p\_best(i)$，同时记录当前混沌优化搜索所得的最佳位置 cs_Pbest 及其适应度。在混沌优化搜索结束时，若 cs_Pbest 比种群的最佳位置 g_Pos 更优，则用 cs_Pbest 及其适应度更新当前种群最佳位置 g_Pos 和种群最佳适应度 g_{best}；

步骤 5 采用式 3-4、式 3-5 分别更新粒子位置和速度；

在对粒子 i 进行位置和速度的更新过程中，若 $Pos(i)$中有某个分量不在 $[X_{min},X_{max}]$内即粒子飞出有效区域，为了粒子下次的飞行仍然有意义就必须使 $Pos\ (i)$中的每个分量都在$[X_{min},X_{max}]$区间上。对于以上情况，我们将对 $Pos\ (i)$中的每个不在$[X_{min},X_{max}]$内的分量重新随机取一个$[X_{min},X_{max}]$内的值；同时，若粒子 i 的速度 $v(i)$存在某个分量小于 V_{min} 时将其设置为 V_{min}，而当大于 V_{max} 时，粒子下次的飞行同样失去意义，因为 V_{max} 本身等于 X_{max}，$Pos(i)$中的每个分量又都在 $[X_{min},X_{max}]$，自然就很容易造成粒子飞出有效区域，粒子本次的搜索就徒劳无功了。因而，当中的某个分量大于 V_{max} 时，我们将对此分量重新取一个$[V_{min},V_{max}]$的随机值。

步骤 6 跳至步骤 3 直到算法到达最大迭代次数 $Iter\_max$ 为止；

步骤 7 输出全局最优值和全局最优粒子。

综上所述， MCPSO 算法描述如图 4-3 所示。

```
Algorithm: Grid Task Scheduling task using MCPSO
Input:
Output:
Begin
1. Setting parameters for the MCPSO
2. Chaos Initialize 2* POPSIZE Positions
```

3. For each chaos Initialized Position *i*
4. Calculate *Fit* (*i*)
5. End for
6. Select the top *POPSIZE* positions as the positions of the initial Swarm
7. For each particle *i* in the Swarm
8. Randomly initialize the velocity *v*(*i*)
9. set p_{best} equals *Fit* (*i*) and update *p_best*(*i*) using *Pos* (*i*)
10. IF *Fit* (*i*)< g_{best}
11. Update g_{best} using *Fit* (*i*)
12. Update *g_Pos* using *p_best*(*i*)
13. End IF
14. End For
15. While *iter* less than *Iter_max*
17. For each particle *i* in Swarm
18. Calculate *Fit* (*i*)
19. While *iter_chaossearch* less than *iter_csmax*
20. IF the result of chaos search better than this particle
21. Update the particle with the chaos search position
22. End IF
23. End While
24. IF *Fit* (*i*) better than *p_best*(*i*)
25. Update p_{best} with *Fit* (*i*)
26. Update *p_best*(*i*) with *Pos* (*i*)
27. End IF
28. IF p_{best} better than g_{best}
29. Update g_{best} with p_{best}
30. Update *g_Pos* with *p_best*(*i*)
31. End IF
32. Update the *velocity*(*i*) using (3-5)
33. Update the *Pos* (*i*) using (3-4)
34. End For
35. End While

End

图 4-3 MCPSO 算法描述

4.4 本章小结

本章通过改进的 PSO 算法来解决网格环境下的独立任务调度问题。首先给出任务调度的问题定义，然后介绍了改进 PSO 算法的相关技术，将混沌优化理论引入到 PSO 算法中进行算法的改进，最后描述了该改进算法的详细过程并运用于网格任务调度中。MCPSO 算法首先通过混沌初始化得到较优的初始种群，有效减少粒子飞向较优点的迭代次数；在更新粒子位置和速度时，对粒子的最优位置进行混沌搜索，试图找出更优的位置，使粒子飞向更优位置而避免陷入局部最优位置。本文的创新点在于将混沌优化思想融入到粒子群算法的种群初始化和算法的进化过程中，有效避免算法过快收敛于局部最优点，使得算法在任务调度中具有更小的完成时间和网格系统具有更高的负载均衡性能。

第五章 实验及性能分析

为了检验改进 PSO 的网格任务调度算法的性能，需要通过实验对算法的正确性和有效性进行验证，我们将采用 GridSim 网格任务调度模拟器仿真环境来进行任务调度过程的模拟。将第四章中改进的 PSO 算法与 PSO 算法应用于任务调度中，并进行分析对比。本章通览了几种常用的网格仿真工具并对 GridSim 着重进行介绍，然后介绍改进 PSO 算法和 PSO 算法的网格任务调度算法的实验方法、参数设置以及性能的分析。实验证明，与 PSO 算法相比， MCPSO 算法能有效提升任务调度问题的求解质量，缩短了 makespan，优化了网格资源的负载均衡性能。

5.1 网格仿真工具

网格仿真工具是一个很有用的网格研究工具，通过仿真这个虚拟环境可以预测、并不断优化实际网格环境。对真实网格环境下的任务调度进行模拟涉及许多方面，如网格的高层性能、底层数据处理以及网格中的资源节点等，通常是借助于一些仿真工具来完成任务调度器、资源以及网络的模拟。

5.1.1 常用仿真工具

目前，常用的网格仿真工具主要有：

1．SimGrid

SimGrid 仿真工具是由圣地亚哥网格研究和创新实验室（Grid Research And Innovation Laboratory）开发的，适用于分布并行调度，主要为其提供相应的模型和抽象，以及准确的模拟结果[61]。SimGrid 的版本有两个：SG 和 MSG。其中 SG 适合模拟基于有向无环图 DAG 的集中式调度，并提供了相应的底层 API 来建立模拟环境。而后者则是建立在前者之上，所提供的 API 是面向应用的，非常适用于独立任务调度的模拟。

2．GridSim

墨尔本大学开发的 GridSim 是基于 Java 语言的网格仿真平台，是基于计算经济模型的资源分配的有力模拟工具[62]。GridSim 吸纳资源的“买”和“卖”思想来建立“经济模型”，从而使网格资源的使用得到有效控制。GridSim 是在 SimJava 之上开发出来的，提供了丰富的函数库以及多种参数供用户进行异构分布的网格实体配置，如网格资源、大型应用程序、用户、任务调度器以及用户代理等。实体间交互是通过消息事件来实现的。

3．MicroGrid

MicroGrid 是加州大学圣地亚哥开发的，类似于 Globus，通过现有的物理资源对虚拟的网格环境进行模拟[63]。模拟引擎 MaSSF 是 MicroGrid 的重要组成部分，是在模拟引擎 DaSSF 之上建立。借鉴于对完全可控的虚拟网格环境的实现过程，我们可以完成对网格系统的设计及其性能评估。

4．GridNet

GridNet 是美国国家科学基金、IBM 奖学金项目以及 IBM 阿尔玛登研究中心一同研究的，主要针对数据网格(Data grid)中动态数据复制策略的模拟，提出了以代价估计为模型的副本决策[64]。GridNet 是采用模块化进行设计的，是基于网络模拟软件 ns 而建立的。GridNet 模拟的环境中主要有以下三种类型的节点：客户端产生访问数据的请求；服务器负责存储工作；而缓存节点则代表着中间的存储节点，实现放置在服务器上的数据的拷贝工作。此三类节点是通过 ns 虚拟的网络环境进行交互的。

5.1.2 GridSim 简介

GridSim 是基于 Java 语言开发的工具包，提供了基本的网格功能部件，并可以对各功能部件基本行为进行模拟。GridSim 为网格任务调度研究提供了一个良好的模拟环境。它能模拟世界范围内的计算、资源节点，同时还提供任务查找和虚拟处理等功能。通过分层的方法来模拟实际网格中的任务调度，且每一层负责的功能不同。GridSim 大体可以分为：运行环境层、工具层和开发应用层，如下图 5-1 所示。

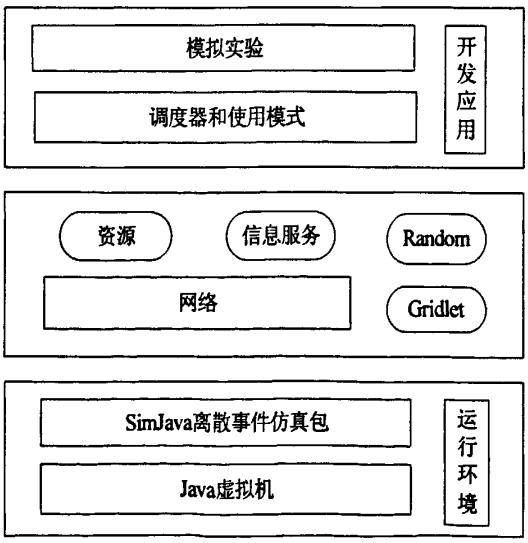

图 5-1 GridSim 体系结构

运行环境层主要给 GridSim 提供支持，有两层：上层是 SimJava 离散事件的

仿真包，包含各种基础类，并提供记录和统计功能；下层是 Java 虚拟机，为 Java 提供运行平台。

GridSim 工具中的主要实体有三个：资源实体、网络实体和信息服务实体。它们分别对应于实际网格环境中的资源、网络和信息服务，并且模拟了实际网格系统中的交互行为。资源实体是借助 GridResource 和一系列辅助类来实现，而资源的分布性和异构性则是采用一些参数来描述的。表征异构性参数主要有：处理器数、处理速度、成本、调度策略、本地负载以及机器数等。而分布性主要是通过时区参数来体现的。网络实体是采用类：Input 和 Output 来实现的，主要用于模拟数据在网络中的传输及其延迟。信息服务实体负责资源的注册和查询，是依靠类 GridInformationService 来实现。此外，GridSim 工具还包含有 Gridlet、GridSimStandardPE、GridSimShutdown 等实体。这些实体可以帮助使用者更好的进行开发、记录和控制整个模拟过程。

5.2 实验及结果分析

本文将网格任务调度模拟环境构筑在 GridSim 仿真平台之上，并在这个环境中对本文提出的 MCPSO 算法与 PSO 算法进行分析、比较。

5.2.1 实验方法

为了测试改进后的任务调度算法效率，我们实现了 MCPSO 算法和 PSO 算法并在仿真平台 GridSim 上进行大量的实验，然后从三个度量指标进行改进后与原算法的相对性能分析比较。

算法中使用到的性能评价指标有：

1．时间跨度（makespan）

第一个任务开始调度到资源节点上执行到所有任务全部执行完之间所花费的时间称为时间跨度。显然，跨度越小，算法性能越优，网格系统的性能也就越好。

2．负载均衡度

负载均衡度 β 表示计算资源负载偏离均衡负载 L_{min} 的程度，其数值越小，网格系统就越均衡。其计算方法如式 5-1 所示。

$$\beta = \frac{1}{m} \sum_{1 \le j \le m} (RC_j - \mathrm{L}_{\min})^2 \qquad (5\text{-}1)$$

3．求解质量

求解质量通过算法取得最好解、最差解及最好解次数来表征。

5.2.2 实验参数

PSO 算法涉及到的参数设置如下：

（1）本文是在文献[57]提出的带收缩因子的 PSO 算法基础上进行改进，设置学习因子 c_1=c_2=2.05，惯性因子 w 就等于 0.729。而根据实验设置的问题规模，PSO 算法的种群规模 *POPSIZE* 取值为 20，算法最大迭代次数 *Iter_max* 为 1000。由于 PSO 算法的随机性，因而在实验中两个算法的运行次数 R_{sum} 都为 50 次，然后取均值作为最终的性能比较依据。

（2）混沌初始化和混沌优化所使用的系统模型为：Logistic 方程。由于 Logistic 方程为完全混沌状态时，问题的初始解不易“聚集”在一起，能够更好地遍布于整个解空间，因而这里控制参数 μ 设为 4 时，而混沌优化搜索次数为 5。

（3）网格中的资源数 *RES_SUM* 为 5，提交的任务数 *TASK_SUM* 为 10。

（4）求解问题的解空间为[0, *RES_SUM*]，解空间的维数为 *TASK_SUM*。粒子 X_{max}= *RES_SUM*，X_{min}=0，V_{max}=X_{max}，V_{min}=X_{min}。

5.2.3 实验结果分析

本文在 GridSim 工具中编写并实现了两个模拟程序：PSO 算法和 MCPSO 算法。程序对相关参数进行设定，描述了相关资源和任务的详细情况，还包括相关调度过程。由于 PSO 算法是一种随机搜索算法，每次的运行结果都会不相同，因而为了消除随机性影响，更好体现算法的性能，算法运行 50 次，并取其平均值作为最后的评价基础。

实验中有 5 个节点，10 个任务，随机初始化值在[10,100]间各个任务的预计完成时间并保存，这里我们使用以下随机值：{10, 42, 34, 27, 56, 79, 77, 62, 81, 51}，单位为 s。

实验从调度方案的求解质量、makespan、负载均衡度三个方面比较两种算法的性能。

1．求解质量

表 5-1 实验结果比较

算法	makespan	最好解	最好解	最好解次数
PSO	116.060	111	138	10.8
MCPSO	112.980	111	118	13.32

由表 5-1 比较可知，本文提出的 MCPSO 算法比 PSO 有更好的性能，平均跨度值与最好解的偏差小，最好解的次数明显多于 PSO 算法。最差解也明显优于 PSO， MCPSO 算法的求解质量更好。

2．makespan

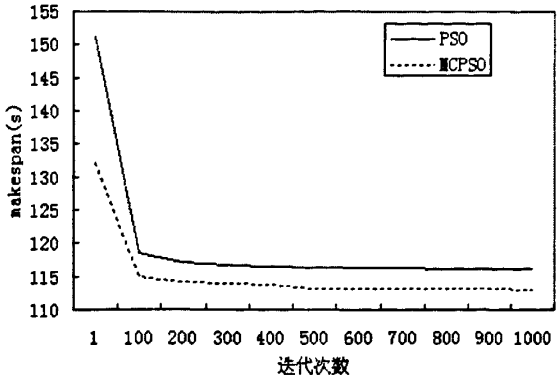

图 5-2 最优跨度收敛情况

为了观察算法的收敛特性，图5-2给出了两种算法在1000次迭代的makespan变化曲线，从图中可以看出， MCPSO算法收敛速度优于PSO算法的情况下，能得到更好的解，而且改进PSO算法的初始解明显更优。

3．负载均衡度

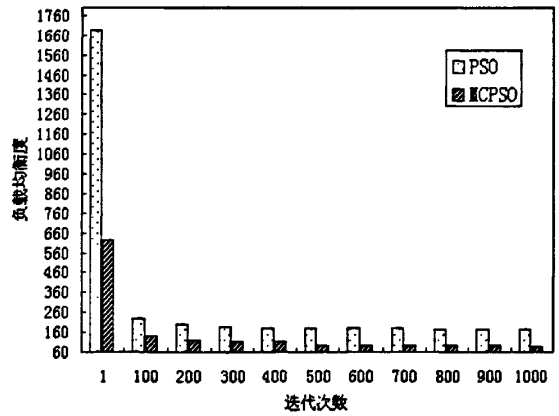

图 5-3 负载均衡情况

由图5-3可知，两种算法的负载均衡度随着迭代次数的增加而减少，即负载均衡性能都有提升。而MCPSO算法的负载均衡性能大大好于PSO算法。

通过实验可以看出，基本粒子群能快速的找到较优解，但是后期逐渐收敛于该解，而处于“停滞”状态，很难得到更优解。而引入混沌机制的MCPSO算法，拥有基本粒子群算法快速收敛的特性，同时在未得到最优解的情况下，逐渐向最优解靠近，不断得到更优解甚至最优解。

5.3 本章小结

本章首先简单的介绍几种常用的网格仿真工具，并着重介绍了GridSim。然后介绍了改进PSO算法和基本PSO算法的网格任务调度算法在GridSim仿真环

境下的参数设置，并从时间和费用两方面对两个算法进行分析比较。实验证明，与基本 PSO 算法相比，MCPSO 算法减少了整个任务集的运行时间，有效提高求解质量，大大改善了网格负载均衡性能。

第六章 结束语

本章是对论文研究工作的简要总结，并提出今后下一步工作的展望。

6.1 工作总结

网格中的任务调度问题对网格来说极为关键，直接决定网格内的资源能否得到有效合理的利用。本文围绕粒子群优化算法并结合混沌优化机制进行改进，研究网格环境中的任务调度问题。在分析粒子群优化算法和混沌优化机制的基础上，结合两者的优点使其优化行为互补，提出一种改进 PSO 算法并引入到网格任务调度中。本文的研究工作主要如下：

（1）简要介绍了网格任务调度算法的研究现状和 PSO 算法在网格中的研究现状。对网格任务调度的相关内容进行介绍，着重介绍了几种常见的启发式调度算法并分析其优缺点。

（2）介绍了 PSO 算法的概念、特点、算法流程、相关参数、相关改进策略和混沌优化理论，并针对粒子群优化算法易早熟求解质量不高的缺陷，提出了一种融合混沌初始化及混沌优化搜索的改进 PSO 算法，并详细讲述了算法的改进策略以及实现过程。

（3）根据网格任务调度的特点，对改进 PSO 算法和基本 PSO 算法基于时间 QoS 在 GridSim 网格仿真器中进行试验，给出了相应的编码方案、详细的参数设置，并对最后的结果进行分析比较。

6.2 研究展望

本文基于时间约束下的网格任务调度算法进行了相应的研究。一直以来，任务调度问题在网格研究领域是热点，也是难点问题，还存在很多问题有待解决。接下来的研究工作包括：

（1）本文的任务调度模型是基于若干假设设计的，只从数据处理时间的角度对调度系统进行评估。而实际的网格环境中应用往往是大型的，通常涉及一系列的数据通信，其对网格调度的影响不亚于计算时间，有时甚至影响程度更大。因而，以后在对任务调度问题进行建模时，要充分考虑网格应用的各类影响因素并将其纳入模型中，如通信时间。

（2）本文只是单一的考虑时间 QoS，而忽略了信任 QoS、费用 QoS 等，没能更为实际的反应网格用户的需求。因而，如何针对实际的网格环境设计一个综合 QoS 量化模型以得到更好的用户满意度急需进一步的研究。

（3）实际网格环境中，任务间并非都不存在数据依赖和通信依赖的，而是还存在关联网格任务，而且任务可能进一步还可以进行细分，如何在现有改进PSO 算法的基础上，对 PSO 算法进行相应的改进以适应新的变化，同样也是我们下一步工作的重点。

参考文献

[1] FAFNER, http://www.npac.syr.edu/factoring.html.

[2] Ian Foster, Carl Kesselman. 网格计算（第二版），金海，袁平鹏，石柯译. 北京：电子工业出版社，2004.

[3] Foster I, Kesselman C, Tuecke S. The anatomy of the Grid: Enabling scalable virtual organizations. International Journal of High Performance Computing Applications, 2001,15(3):200-222.

[4] Fangpeng Dong, Selim G. Akl. Scheduling algorithms for grid computing: state of the art and open problems[D]. Technical report. School of computing, Queen's University. Kingston, Ontario, January, 2006.

[5] Complexity results for scheduling problems. http://www.mathematik.uni-osnabrueck.de/research/OR/class.

[6] J. Gehring and T. Preiss. Scheduling a Metacomputer With Unwoperative Subschedulers. In D. G. Feitelson and L. Rudolph, editor, Proc. of 5th Workshop on Job Scheduling Strategies for Parallel Processing, volume 1659 of Lecture Notes in Computer Science, pages 179 201. Springer, 1999.

[7] Achim Streit, Self-Tuning Job Scheduling Strategies for the Resource Management of HPC Systems and Computational Grids, Ph.D Thesis, October 2003.

[8] 朱福喜，何炎祥编著. 并行分布计算中的调度算法理论与设计[M]. 武汉大学出版社 2003.

[9] H.El-Rewini, T.G. Lewis. Scheduling Parallel Program Tasks onto Arbitrary Target Machine[J]. J. Parallel and Distributed Computing, 1990, 9:138-153.

[10] Tao Yang, Apostolos Gerasoulis: DSC: Scheduling Parallel Tasks on an Unbounded Number of Processors[J]. IEEE Transactions on Parallel and Distributed Systems,1994, 5(9): 951-967.

[11] Li Guodong, Chen Daoxu, Wang Darning, Zhang Defu. Task Clustering and Scheduling to Multiprocessors with Duplication[A]. Proceedings of the International Parallel and Distributed Processing Symposium(IPDPS'03) [C]: p.6b, April 22-26, 2003, Nice, France.

[12]H.Singh and A. Youssef. Mapping and Scheduling Heterogeneous Task Graphs Using Genetic Algorithms[C]. Proc. Heterogeneous Computing Workshop, I 996:86-97.

[13]P. Shroff, D.W. Watson,N.S. Flann and r. Freund. Genetic Simulated Annealing for Scheduling Data-Dependent Tasks in Heterogeneous Environments[A]. Proc. Heterogeneous Computing Workshop[C]. 1996:98-104.

[14]尚明生. 网格计算中的任务调度算法研究[博士学位论文]. 成都：电子科技大学，2007.

[15]李小平, 徐晓飞, 战德臣. 一种独立任务的同型机调度快速算法. 软件学报，2002,13(4):812-817.

[16]康一梅, 邓应平. 同等并行处理机上独立任务的调度. 自动化学报，1997,23(1):81-84.

[17]Frangioni A, Necciari E, Scutella M Cz A multi-exchange neighborhood for minimum makespan machine scheduling problem. Journal of Combinatorial Optimization, 2004, 8:195-220.

[18]Tracy D Braun, Howard Jay Siegel, Noah Beck, etc. A comparison of eleven static heuristics for mapping a class of independent tasks onto heterogeneous distributed computing systems. Journal of Parallel and Distributed Computing, 2001, 61: 810-837.

[19]林伟伟, 齐德显, 李拥军等. 树型网格计算环境下的独立任务调度. 软件学报，2006,17(11):2352-2361.

[20]Karger D, Stein C, Wein J. Scheduling Algorithms. http://theory Ics.mit.edu/～karger/Papers/scheduling.ps.gz.

[21]Dell' Amico M, Martello S. Optimal scheduling of tasks on identical parallel peocessors. ORSA Journal on Computing, 1995, 7(2):686-689.

[22]Langston M A. Improved 0/1 interchange scheduling. BIT, 1982, 22: 282-290.

[23]Tracy D Braun, Howard Jay Siegel, Noah Beck, etc. A comparison of eleven static heuristics for mapping a class of independent tasks onto heterogeneous distributed computing systems. Journal of Parallel and Distributed Computing, 2001, 61: 810-837.

[24]He Xiaoshan, Sun Xian-He, Gregor von Laszewski. QoS Guided Min-Min Heuristic for Grid Task Scheduling, Journal of Computer Science and Technology Special Issue on Grid Computing, 2003:18(4):442-451.

[25]胡志刚, 吕祯恒. 一种基于动态资源预留的任务映射算法[J]. 计算机应用研究, 2005,(07).

[26]P. J. Angeline. Using Selection to Improve Particle Swarm Optimization. IEEE International Conference on Evolutionary Computation. Alaska, 1998: 84-89.

[27]J. R. Zhang, J. Zhang, T. M. Lok, M. R. Lyu. A Hybrid Particle Swarm Optimization-back-propagation Algorithm for Feed forward Neural Network Training. Applied Mathematics and Computation. 2007, 185(2): 1026-1037.

[28]C. J. Lin, S. J. Hong. The Design of Neuro-fuzzy Networks Using Particle Swarm Optimization and Recursive Singular Value Decomposition. Neurocomputing. 2007, 71(1-3): 297-310.

[29]Ray T, Liew K M. A swarm with an effective information sharing mechanism for unconstrained and constrained single objective optimization problems [A], Proc IEEE Int Conf on Evolutionary Computation [C]. Seoul. 2001. 75-80.

[30]Eberhart R, Kennedy J. Discrete binary version of the particle swarm algorithm [A], Proc IEEE Int Conf on Systems, Man and Cybernetics [C], Orlando, 1997, 4104-4108.

[31]C. Karakuzu. Fuzzy Controller Training Using Particle Swarm Optimization for Nonlinear System Control. ISA Transactions. 2008, 47(2): 229-239.

[32]A. A. Abou El-Ela, T. Fetouh, M. A. Bishr, R. A. F. Saleh. Power Systems Operation Using Particle Swarm Optimization Technique. Electric Power Systems Research. 2008, doi:10.1016/j.epsr.2008.03.021.

[33]Q. Zhao, S. Z. Yan. Collision-free Path Planning for Mobile Robot Using Chaos PSO Algorithm. Lecture Notes in Computer Science. Advances in Natural Computation. 2005, (3612): 632-635.

[34]X. H. Shi, YC. Liang, H. P. Lee, C. Lu, Q. X. Wang. Particle Swarm Optimization-based Algorithms for TSP and Generalized TSP. Information Processing Letters. 2007, (103): 169-176.

[35]A. W. Mohemmed, N. C. Sahoo, T. K. Geok. Solving Shortest Path Problem Using Particle Swarm Optimization. Applied Soft Computing. 2008, doi: 10.1016/j.asoc.2008.01.002.

[36]季一木，王汝传. 基于粒子群的网格任务调度算法研究[J] . 通信学报，2007，28(10)：60-66.

[37]朱海，王宇平，权义宁，王晓丽. 采用离散粒子群算法的网格任务安全级调度[J]. 西安交通大学学报，2010，44(6)：21-26.

[38]迟玉红，白鹏，于俊发，喻春明. 基于 PSO 算法的网格任务调度策略[J] . 辽宁工程技术大学学报(自然科学版)，2010，29(2)：274-277.

[39]罗红，慕德俊，邓智群，等. 网格计算中任务调度研究综述[J]. 计算机应用研究，2005: 16-19.

[40]Wang. Y, Vassileva. J. Bayesian Network Trust Model in Peer-to-Peer Networks[C]. In Proceedings of Second International Workshop Peers and Peer-to-Peer Computing, July 14, Melbourne, Australia, 2003.

[41]Siegel H J, Ali S. Techniques for mapping tasks to machines in heterogeneous computing systems. Journal of Systems Architecture, 2000, 46:627-639.

[42]Tracy D Braun, Howard Jay Siegel, Noah Beck, etc. A comparison of eleven static heuristics for mapping a class of independent tasks onto heterogeneous distributed computing systems. Journal of Parallel and Distributed Computing, 2001, 61, 810-837.

[43]Muthucumaru Maheswaran, Shoukat Ali, Howard Jay Siegel, etc. Dynamic mapping of a class of independent tasks onto heterogeneous computing systems. Journal of Parallel and Distributed Computing, 1999, 59:107-131.

[44]Anil Kumar V S, Marathe M V Approximation algorithms for scheduling on multiple machines. Proc. of the 46th IEEE Symposium on Foundations of Computer Science (FOCS 2005), 2005, pp: 254-263.

[45]Ahraham A, Buyya R. Nature's Heuristics for Scheduling Jobs on Computational Grids[C].Un: The 8th Int Conf. on Advanced Computing end Communications, Cochin, India, 2000.

[46]李宗勇, 彭霞, 王智学, 刘影. 基于蚁群算法的参数相关网格任务调度算法研究[J]. 系统仿真学报, 2007,(14) .

[47]钟求喜, 谢涛, 陈火旺. 基于遗传算法的任务分配与调度[J]. 计算机研究与发展, 2000,(10).

[48]Lei Zhang, Yuehui Chen, Bo Yang. Task Scheduling Based on PSO Algorithm in Computational Grid[M]. Intelligent System Design and Applications，2006：696-704.

[49]Wolpert D H and Macteady W G. No free lunch theorems for optimization. IEEE Transaction on Evolutionary Computation, x1(1), 2005, 67-82.

[50]Kennedy J, Eberhart R. Particle Swarm Optimization[A]. In proc. IEEE Int Conf on Neural Networks[C]. Perth, 1995.

[51]Kennedy J, Eberhart R. Particle Swarm Optimization[C]. In: Proceeding of IEEE International Conference on Neural Networks, Piscataway, NJ: IEEE CS, 1995:1942-1948.

[52]Eberhart R, Kennedy J. A new optimizer using particle swarm theory[C]. In: Proceeding of the 6th International Symposium on Micro Machine and Human Science, NJ: IEEE CS, 1995:39-43.

[53]Shi Y, Eberhart R. A Modified Particle Swarm Optimizer[C]. In: IEEE World Congress on Computational Intelligence, Anchorage, AK, USA, 1998:69-73.

[54]P. N. Suganthan. Particle Swarm Optimizer with Neighborhood Operator. Proceedings of Congress on Evolutionary Computation. 1999: 1958-1962.

[55]高芳. 智能粒子群优化算法研究[博士学位论文]. 哈尔滨：哈尔滨工业大学，2008.

[56]Shi Yuhui, Eberhart R. A modified particle swarm optimizer. Proc IEEE Int Conf on Evolutionary Computation. Anchorage, USA, 1998: 69-73.

[57]Clerc, M. The swarm and queen: towards a deterministic and adaptive particle optimization. Proceedings of the IEEE Congress on Evolutionary Computation, 1999:1951-1957.

[58]李兵，蒋慰孙. 混沌优化方法及其应用[J]. 控制理论与应用.1997(4): 613-615.

[59]丁文霞. 基于混沌理论的多媒体信息安全算法研究. 长沙：国防科学技术大学，2008.

[60]张化光，工智良，黄伟编著. 混沌系统的控制理论[M]. 沈阳:东北大学出版社，2003.

[61]H. Casanova. Simgrid: A Toolkit for the Simulation of Application Scheduling, Proceedings of the First IEEE/ACM International Symposium on Cluster Computing and the Grid, Brisbane, Australia, May 15, 2001.

[62]I. Foster, C. Kesselman, J. Nick, S. Tuecke. The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration[C]. Open Grid Services Infrastructure WG, Global Grid Forum, June 22, 2002.

[63]H. Song, X. Liu, D. Jakobsen, et al. The MicroGrid: -A Scientific Tool for Modeling Computational Grids, Proceedings of IEEE Supercomputing (SC 2000) Conference, Dallas, USA, Nov. 4-10, 2000.

[64]Czajkowski K, Fitzgerald S, Foster I, et al. Grid Information Services for Distributed Resource Sharing[C]// In Proceedings of the 10}" IEEE International Symposium on High Performance Distributed Computing (HPDC-10). [s.l.]: IEEE Press, 2001:181-195.

致 谢

至此硕士论文完成之际，谨向所有给予我指导、关心、帮助和鼓励的老师、同学和亲人朋友表达我由衷的谢意。

首先，要衷心感谢的是我的恩师杨长兴教授。在三年的研究生学习生涯里，杨老师在生活上给予我极大的帮助，关怀备至。在学习上，杨老师对我严格要求，对我本文的研究工作给予悉心指导，并且提供了一个宽松自由的发展环境。杨老师渊博的知识，严谨的治学作风，务实的工作态度使我受益匪浅。杨老师无论在课题的研究上还是人生发展都给予我无私的指导和教诲，使自己在完成学位论文的同时自身的综合素质有了很大提高。谨此，对杨老师多年的无私关爱和辛勤培养致以我崇高的敬意和由衷的感谢。

感谢和我一起走过研究生生涯的同学、朋友。一路走来，他们在生活和学习给予了我许多启发和无私帮助，我们共同探讨学术和人生发展，共同学习进步，互相勉励，一起度过这难忘的求学时光。感谢信息院的领导和老师们对我的悉心指导和支持，是他们为我提供了良好的学习环境和机会。

感谢我的父母和亲人，他们的无私关爱和支持让我有了克服困难的信心和勇气，让我一路平坦，指引我继续向前。我前进道路上的每一个脚印，都倾注着他们无私的爱。

最后，感谢曾经教导和帮助过我的所有老师。衷心感谢为评审本论文而付出辛勤劳动的论文评审委员会的老师们。

攻读学位期间主要的研究成果

发表论文

[1] 杨长兴，胡金.一种改进的 PSO 网格调度算法.微型机与应用.（已录用）

基于改进PSO算法的网格任务调度算法

作者：胡金
学位授予单位：中南大学