软件学报 ISSN 1000-9825, CODEN RUXUEW
*Journal of Software*,2012,23(11):3000−3008 [doi: 10.3724/SP.J.1001.2012.04305]

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel/Fax: +86-10-62562563

# 多 Agent 主从粒子群分布式计算框架*

郑宇军[1+], 陈胜勇[1], 凌海风[2], 徐新黎[1]

[1](浙江工业大学 计算机科学与技术学院,浙江 杭州 310023)

[2](解放军理工大学 机械工程系,江苏 南京 210007)

## Multi-Agent Based Distributed Computing Framework for Master-Slave Particle Swarms

ZHENG Yu-Jun[1+], CHEN Sheng-Yong[1], LING Hai-Feng[2], XU Xin-Li[1]

[1](College of Computer Science and Technology, Zhejiang University of Technology, Hangzhou 310023, China)

[2](Department of Mechanical Engineering, PLA University of Science and Technology, Nanjing 210007, China)

+ Corresponding author: E-mail: yujun.zheng@computer.org, http://www.zjut.edu.cn

**Zheng YJ, Chen SY, Ling HF, Xu XL. Multi-Agent based distributed computing framework for master-slave particle swarms. *Journal of Software*, 2012,23(11):3000−3008 (in Chinese).** http://www.jos.org.cn/1000-9825/4305.htm

**Abstract**: To effectively solve large-scale optimization problems, the paper proposes a distributed agent computing framework based on the parallel particle swarm optimization (PSO). The framework uses a master swarm for evolving complete solutions of the problem, and uses a set of slave swarms for evolving sub-solutions of the subproblems concurrently. The master swarm and slave swarms alternatively implement the PSO procedure to improve the problem-solving efficiency. Using the asynchronous team based agent architecture, a master/slave swarm consists of different kinds of agents, which share a population of solutions and cooperate to evolve the population, such as initializing solutions, moving particles, handling constraints, and decomposing/synthesizing sub-solutions. The framework can be used to solve complicated constained and multiobjective optimization problems efficiently. Experimental results demonstrate that this approach has significant performance advantage over two other state-of-the-art algorithms on a typical transportation problem.

**Key words**: agent; particle swarm optimization (PSO); master-slave model; cooperative evolution; distributed computing

**摘 要**: 面向大规模复杂优化问题,提出了一个基于并行粒子群优化的分布式 Agent 计算框架.框架中使用一个主群(master swarm)来演化问题的完整解,并使用一组从群(slave swarm)来并行优化一组子问题的解,主群和从群通过交替执行来提高问题的求解效率.采用异步组结构,主群/从群中的各类 Agent 共享一个解群,并通过相互协作,对解群进行构造、改进、修补、分解和合并等演化操作.该框架可用于求解复杂的约束多目标优化问题.通过一类典型运输问题上的实验,其结果表明,所提出的方法明显优于另外两种先进的演化算法.

**关键词**: agent;粒子群优化;主从模型;协同进化;分布式计算

**中图法分类号**: TP18 **文献标识码**: A

* 基金项目: 国家自然科学基金(61105073, 61173096, 61103140, 61020106009, 61070043); 浙江省自然科学基金(R1110679)
收稿时间: 2012-06-09; 定稿时间: 2012-08-21

群智能(swarm intelligence,简称 SI)源于对自然界中昆虫、鸟类等生物社会行为的观察和模拟,主要研究“简单智能的主体通过合作表现出复杂智能行为的特性”,是目前自然计算研究中最为活跃、应用也最为广泛的一个分支[1,2].

Eberhart 和 Kennedy 提出的粒子群优化(particle swarm optimization,简称 PSO)算法[3]是群智能的一种典型模式.它模拟鸟类的聚集飞行行为,将鸟群栖息地类比于问题解空间中可能解的位置,将鸟抽象为无质量和体积的粒子,通过相互协作和信息共享来不断调整粒子的飞行方向和速度,从而促使整个群体向最优解收敛.

PSO 具有实现简单、收敛速度快等特点;但和其他许多随机优化方法一样,PSO 算法的性能也会随着搜索空间维度的增加而快速下降.为此,van den Bergh 等人提出了协同进化的 PSO 算法[4].该方法将高维解向量分解为多个子向量,并使用多个子群来对不同的子向量进行并行优化,最后通过合并子向量来得到完整的解向量.这种方法能够显著提高算法性能,但分解过程中需要谨慎处理子问题之间可能存在的变量相关性.Goh 等人将协同 PSO 扩展用于多目标优化[5],并通过引入竞争机制来进一步改善算法效率.

从智能特性来看,PSO 中的粒子与软件 Agent 存在很大的相似性,包括具有自主性(具有自身独立的知识和知识处理方法)、反应性(对环境的变化作出反应)、社会性(与环境中的其他个体相互协作)、进化性(随着环境的变化不断扩充自身的知识和能力)[6].Ahmad 等人发现,通过赋予粒子更强的自治和学习能力,能够有效提高 PSO 算法的寻优能力[7].Lorion 等人提出了一种基于 Agent 的并行 PSO 算法[8],其中各个 Agent 采用并发方式对原问题的各个子问题进行优化.Kumar 等人也提出了一种基于多 Agent 的混合 PSO 算法[9],但是其中每个 Agent 是对应一个粒子.这些基于 Agent 的 PSO 算法在能源分配[10]、电机设计[11]、教育管理[12]、生产调度[13]等领域得到了有效的应用.此外,一些学者也对 Agent 技术与其他智能优化算法的结合进行了研究,如多 Agent 遗传算法[14]、人工免疫算法[15]等.

随着技术的不断发展,在实际应用中,优化问题的规模和复杂度也不断增长,使用传统的数学优化方法和一些简单的启发式优化方法往往难以有效求解,很多问题求解所需的数据和软硬件等资源还可能分散在不同的网络节点上.

针对此类复杂优化问题(特别是多目标约束优化问题),本文提出了一个基于 PSO 的分布式 Agent 计算框架.它通过一个主群(master swarm)来优化原问题,同时又将问题分解为一组子问题、使用一组从群(slave swarm)来并行优化这些子问题.该框架充分利用分布式 Agent 的灵活性来对解群进行协同演化,从而降低问题求解的难度、提高算法求解的效率.

## 1 粒子群优化算法

### 1.1 标准粒子群优化算法

在求解 $n$ 维空间优化问题时,PSO 算法首先初始化一个粒子群,群中的每个粒子 $i$ 都具有一个位置向量 $\boldsymbol{x}_i=(x_{i1},x_{i2},\ldots,x_{in})$和一个速度向量 $\boldsymbol{v}_i=(v_{i1},v_{i2},\ldots,v_{in})$,并在算法的每次迭代中对其位置和速度进行更新.经过长期改进,目前标准的 PSO 算法采用如下的运动方程来移动粒子[16]:

$$v_{id}^{(t+1)}=\chi(v_{id}^{(t)}+c_1r_2(p_{id}^{(t)}-x_{id}^{(t)})+c_1r_2(g_{id}^{(t)}-x_{id}^{(t)})) \tag{1}$$

$$x_{id}^{(t+1)}=x_{id}^{(t)}+v_{id}^{(t+1)} \tag{2}$$

其中,$d$=1,2,…,$n$,$r_1$ 和 $r_2$ 为(0,1)区间上的随机数,$c_1$ 和 $c_2$ 分别表示自适应学习因子和社会学习因子,$p_i$ 表示粒子 $i$ 搜索过的最佳位置,$g_i$ 表示 $i$ 的某个邻域中的所有粒子搜索过的最佳位置,$\chi$为收缩因子.一般建议取 $c_1=c_2=2.0$;令 $\varphi=c_1+c_2$,则收缩因子

$$\chi=\frac{2}{|2-\varphi-\sqrt{\varphi^2-4\varphi}|} \tag{3}$$

另一类常用的 PSO 算法是使用一个惯性权重 $w$ 来替换收缩因子$\chi$,即使用方程(4)取代方程(1)来确定粒子的移动速度

$$v_{id}^{(t+1)} = wv_{id}^{(t)} + c_1 r_2 (p_{id}^{(t)} - x_{id}^{(t)}) + c_1 r_2 (g_{id}^{(t)} - x_{id}^{(t)}) \quad (4)$$

$w$ 和$\chi$的作用都是为了避免粒子收敛速度过快而使整个群陷入早熟,但实际应用表明,大多数情况下,使用收缩因子的 PSO 算法性能更佳.此外,胡旺和李志蜀还提出速度和位置变化方程可以简化为一个方程,从而避免由粒子速度项引起的粒子发散问题[17].

### 1.2 多目标粒子群优化算法

Parsopoulos 等人最早将 PSO 应用于多目标优化问题[18],但他们只是通过加权方式将多个目标函数转换为单目标函数.

近年来,多目标 PSO(MOPSO)算法吸引了众多研究者的兴趣,并在许多重要问题上表现出了优异的性能.Li 在其提出的算法[19]中引入 Pareto 排序方法,即通过比较粒子间的支配关系来选取运动方程中的 $p_i$ 和 $g_i$,从而真正在解空间中搜索整个非支配解集.Ho 等人[20]使用一个通用的 Pareto 评价函数来对粒子进行评分,并采用分治法来帮助确定粒子的移动位置.Tripathi 等人[21]在 MOPSO 算法中根据粒子群的状态来不断改变惯性权重和学习因子等参数的值,从而提高算法的收敛速度和解集质量.Cooren 等人提出的 MOPSO 算法[22]也是动态改变粒子群的结构以及粒子替换策略,以降低算法对适应函数的敏感度.

## 2 分布式 Agent 计算框架

### 2.1 总体框架

本文提出的计算框架顶层由一个 Master-Swarm 和多个 Slave-Swarm 组成,它们均采用异步组结构[23,24]:每个异步组包含一个 Population 以及一组在 Population 上进行工作的 Agent.

这些 Agent 由以下 4 个组件构成:

- Requester,用于获取事件通知,并判断是否参与相关工作.
- Selector,用于从 Population 中选择要处理的解.
- Operator,核心组件,用于对解执行相关运算.
- Distributor,用于将新解输出到 Population 中.

Population 是异步组中 Agent 的共享工作环境.但与一般群智能算法所使用的解群相比,Population 除了维护当前问题的一组解之外,还存放了问题的输入输出结构、目标函数和约束函数等元信息[25];对于多目标优化问题,Population 中还划分出一块区域专门用于维护当前已找到的非支配解集(即 Pareto 最优解集).

如图 1 所示,Master-Swarm 和 Slave-Swarm 都包含以下 5 类 Agent:

- Scheduler,负责获取用户提出的问题.
- Constructor,负责构造初始(可行)解.
- Improver,负责按照 PSO 运动方程对解进行不断改进.
- Repairer,负责对违反约束的解进行修正处理.
- Destroyer,负责清除 Population 中的无用解.

此外,Master-Swarm 中还包含 2 类专用 Agent:用于对主问题及其解进行分解的 Decomposer 以及用于合并子问题解的 Synthesizer.Decomposer 分解出的内容由 Slave-Swarm 中的 Scheduler 放入其 Population,Slave-Swarm 演化后的子解则通过其 Coordinator 传递给 Synthesizer.它们共同组成了 Master-Swarm 和 Slave-Swarm 之间的联系纽带.

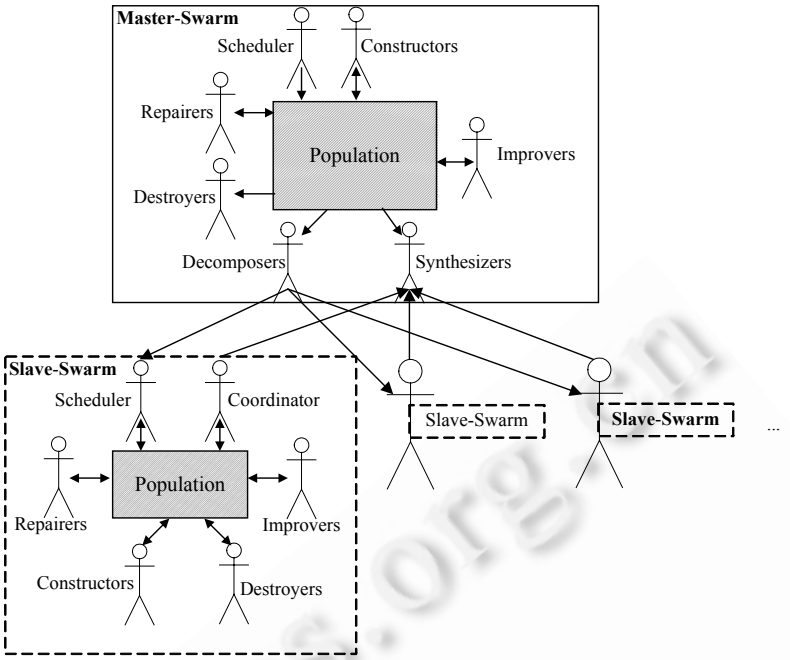



Fig.1 Illustration of the distributed agent computing framework

图 1 分布式 Agent 计算框架示意图

### 2.2 构造PSO Agent

针对一个复杂优化问题,用户(决策者)首先在所在节点上构造一个 Master-Swarm,由 Scheduler 将问题放入 Population 中进行求解.当 Population 中出现了一个新问题以后,Decomposer 对问题进行分解,针对分解出的每个子问题构造一个 Slave-Swarm,并将子问题元信息发送给由其 Scheduler 进行进一步求解.

Decomposer 主要采用以下 4 种问题分解策略:

(1) 按网络计算节点进行分解.即拟使用多少个计算节点,就分解出多少个子问题,并构造相应数量的 Slave-Swarm.分解方式可以是平衡分解,也可以是根据各节点的计算能力进行非平衡分解.这主要适用于高维连续优化问题.

(2) 按问题结构进行分解.即根据问题的元信息进行形式化的分解,分解方式包括笛卡尔积分划、单点分划、半群分划等[26].这主要适用于复杂的组合优化问题.

(3) 按问题自然属性及其求解资源进行分解.比如,对于一个综合规划问题,可为每个参与规划的子部门分解出一个子问题.

(4) 按问题目标函数进行分解.这主要适用于多目标优化问题,通常是针对每个目标函数分解出一个子问题.

上述分解策略也可结合使用,比如在按问题结构或自然属性进行分解时,也可兼顾考虑网络计算节点的数量和分布情况.某些问题的分解过程可能还需要用户的参与.

### 2.3 主从群交替演化

Master-Swarm 的 Population 中放入待求解的问题后,由 Constructor 为其构造一组初始解,而后由 Improver, Repairer 和 Destroyer 共同对这组解进行演化.一个阶段的演化结束后,Decomposer 对 Population 中的解进行分解,并发送给各个相应的 Slave-Swarm,后者在自己的 Population 中对子解进行演化.各个 Slave-Swarm 均完成了一个阶段的演化以后需要进行同步,由 Master-Swarm 的 Synthesizer 对子解进行合并,重新开始新一轮的主群-

从群演化,直至满足算法终止条件.

主从群交替演化的基本算法框架见表 1,其中假定 Slave-Swarm 个数为 $m$,$k^{\max}$ 为 Master-Swarm 每阶段最大迭代次数,$k_i^{\max}$ 为第 $i$ 个 Slave-Swarm 的每阶段最大迭代次数,1.$x$ 和 2.$x$ 分别表示 Master-Swarm 和 Slave-Swarm 所执行的算法步骤.

**Table 1** Alternative master-slave particle swarm optimization algorithm

**表 1** 主从粒子群交替优化算法

| Step | Instruction | Executive agent |
|---|---|---|
| 1.1 | Construct a set of feasible solutions for the population; | Constructor |
| 1.2 | let $k$=0; | |
| 1.3 | while ($k$<$k^{\max}$) do | |
| 1.4 | foreach solution $x$ in population do | |
| 1.5 | move $x$ according to PSO motion equations; | Improver |
| 1.6 | if ($x$ is infeasible) then | |
| 1.7 | if *TryRepair*($x$)=*false* then | Repairer |
| 1.8 | mark $x$ as unrepairable; | |
| 1.9 | if ($x$ is redundant or low-quality or unrepairable) then | |
| 1.10 | remove $x$ from the population; | Destroyer |
| 1.11 | construct a new feasible solution to replace $x$; | Constructor |
| 1.12 | if (termination condition is satisfied) then return; | |
| 1.13 | foreach solution $x$ in population do | |
| 1.14 | decompose $x$ into $m$ parts and send it to the corresponding slave-swarm; | Decomposer |
| 2.1 | for $i$=1 to $m$ do | |
| 2.2 | let $k_i$=0; | |
| 2.3 | while ($k < k_i^{\max}$) do | |
| 2.4 | foreach solution $y$ in slave-population do | |
| 2.5 | if ($y$ is infeasible) then | |
| 2.6 | if *TryRepair*($y$)=*false* then | Repairer |
| 2.7 | mark $y$ as unrepairable; | |
| 2.8 | if ($y$ is redundant or low-quality or unrepairable) then | |
| 2.9 | remove $y$ from the population; | Destroyer |
| 2.10 | construct a new feasible solution to replace $y$; | Constructor |
| 2.11 | move $y$ according to PSO motion equations; | Improver |
| 2.12 | send sub-solutions back to the master swarm; | Coordinator |
| 2.13 | synthesize sub-solutions to complete solutions and goto Step 1.2; | Synthesizer |

在该算法中,步骤 2.1~步骤 2.12 对应的各个 Slave-Swarm 演化过程可在不同节点上并行执行,而步骤 1.4~步骤 1.11 所对应的 Master-Swarm 演化步骤以及步骤 2.4~步骤 2.11 所对应的 Slave-Swarm 演化步骤均可并发执行.这种主从群交替演化方式简单易行,且实验结果表明其对于典型的函数优化问题具有较好的效率.在实际应用时,也可根据问题的特性来调整主从群的执行比例,如适当增加 Slave-Swarm 的执行次数.

### 2.4 多目标优化扩展

对于多目标优化问题,分解出的子问题可以是多目标的,也可以是单目标的.在执行多目标 PSO(multiobjective PSO,简称 MOPSO)演化时,方程(1)中 $p_i$ 和 $g_i$ 的选取需要利用到各个解之间的 Pareto 排序信息[19],且演化过程中都需要维护一个非支配解集 NA.无论是 Master-Swarm 还是 Slave-Swarm,其求解多目标问题时的 Population 结构都如图 2 所示,其中的 PManager 用于对 NA 进行维护.

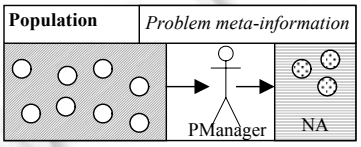



Fig.2 Population structure for multiobjective optimization problems

图 2 多目标优化问题的 Population 结构

每当一个新解 $x$ 被加入到 Population 中以后,PManager 检查其是否被某个现有的解所支配(即在所有目标

函数上都不优于该现有解),否则将 $x$ 也添加到 NA 中,并删除 NA 中那些被 $x$ 所支配的解.

在 MOPSO 执行过程中,非支配解的数量可能在某些时间增长很快,这往往会严重影响算法的效率,因此有必要对 NA 的规模进行限制.当 NA 中非支配解的数量已经达到上限时,PManager 计算并保存 NA 中非支配解集的平均距离;如果又找到一个新的非支配解 $x$,那么只有当解集的平均距离能够增加时才将其替换 NA 中的某个现有解[27].

## 3 应用案例与计算实验

本节以一个综合运输问题来展示上述分布式 Agent 计算框架的应用.考虑 $m$ 个供应点和 $n$ 个需求点,其中第 $i$ 个供应点的库存量为 $a_i$,第 $j$ 个需求点的需求量为 $b_j$.现在要确定从各供应点到需求点运输方案,使得运输总费用和总时间最小.该问题的数学模型可表述为

$$\begin{cases} \min f_1(\boldsymbol{x}) = \sum_{i=1}^{m}\sum_{j=1}^{n} c(x_{ij}) \\ \min f_2(\boldsymbol{x}) = \sum_{i=1}^{m}\sum_{j=1}^{n} t(x_{ij}) \\ \text{s.t. } \sum_{j=1}^{n} x_{ij} \leqslant a_i,\ i = 1,2,...,m \\ \quad \sum_{i=1}^{m} x_{ij} \geqslant b_j, j = 1,2,...,n \\ \quad x_{ij} \geqslant 0, \quad i = 1,2,...,m; j = 1,2,...,n \end{cases} \tag{5}$$

其中,$\boldsymbol{x}=[x_{ij}]_{m\times n}$,$c(x_{ij})$表示将 $x_{ij}$ 数量的货物从供应点 $i$ 运输到需求点 $j$ 所需的费用,$t(x_{ij})$表示相应的运输时间.但这里考虑 $c$ 和 $t$ 的都不是 $x_{ij}$ 的线性函数.实际上,在确定了所有的 $x_{ij}$ 值之后,还需要为每个供应点制定一个车辆调度方案,在该方案的基础上才能计算出相关的运输费用和时间.

可知,该问题是要确定一个复杂的约束多目标优化问题,问题的维数为 $m\times n$.应用本文提出的 Agent 计算框架时,可采用基于自然属性的问题分解方式,即将原问题分解为 $m$ 个子问题,每个供应点 $i$ 对应一个子问题,其数学模型可表述为如下形式:

$$\begin{cases} \min f_1(\boldsymbol{x}_i) = \sum_{j=1}^{n} c(x_{ij}) \\ \min f_2(\boldsymbol{x}_i) = \sum_{j=1}^{n} t(x_{ij}) \\ \text{s.t. } \sum_{j=1}^{n} x_{ij} \leqslant a_i \\ \quad x_{ij} \geqslant 0, j = 1,2,...,n \end{cases} \tag{6}$$

其中,$\boldsymbol{x}_i=[x_{i1},x_{i2},\ldots,x_{in}]$.这样,子问题的维数就下降为 $n$,在 $n$ 维解空间进行搜索的效率会大大高于 $m\times n$ 维解空间.子问题(6)解的演化除了要同时优化两个目标函数以外,还应当考虑子解对整个问题的解的贡献程度.由于子问题未考虑原问题中的第 2 项约束条件,为此,定义如下的收益函数来评价子解 $\boldsymbol{x}_i$ 所完成的运输任务占整体任务的比重:

$$g(\boldsymbol{x}_i) = \sum_{j=1}^{n} \frac{w_j x_{ij}}{b_j} \tag{7}$$

其中,$w_j$ 表示需求点 $j$ 的权重系数,$\sum_{j=1}^{n} w_j = 1$.如无相关权重信息,可令 $w_1 = w_2 = ... = w_n = \frac{1}{n}$.

在每次更新粒子位置信息以后,Improver 按照如下方程来评价每个解 $\boldsymbol{x}_i$ 的适应度,并据此来选取个体和群

体最优解(适应度值越小,解的评价越优):

$$fit(\boldsymbol{x}_i)=\frac{o(\boldsymbol{x}_i)}{g(\boldsymbol{x}_i)} \tag{8}$$

其中,$o(\boldsymbol{x}_i)$为在 Population 中的 Pareto 排序.

Synthesizer 将一组子解合并为一个完整解 $\boldsymbol{x}$ 以后,需要由 Repairer 检查其可行性.如果 $\boldsymbol{x}$ 不满足某项约束 $\sum_{i=1}^{m}x_{ij}\geqslant b_j$, 则计算 $\Delta_j=b_j-\sum_{i=1}^{m}x_{ij}$, 并将未完成的任务部分按如下方式分配给各供应点:

$$x_{ij}=x_{ij}+\Delta_j\frac{a_i-\sum_{j=1}^{n}x_{ij}}{\sum_{i=1}^{m}\left(a_i-\sum_{j=1}^{n}x_{ij}\right)} \tag{9}$$

我们在一组多目标综合运输问题实例上对基于主从粒子群模型的分布式 Agent 计算方法(记为 PSO-Agent)进行了测试.计算环境为一台标配 4×Intel Xeon 3430(4Cores)处理器的服务器.实验时,Master Swarm 和每个 Slave Swarm 分别运行在一个处理器核上,因此可最多支持 15 个 Slave Swarm 的并行执行.为了进行比较,我们还分别选取了较为先进的一种改进粒子群算法[28](记为 CPSO)和一种混合遗传算法[29](记为 HGA).它们在每个问题实例上均使用与 PSO-Agent 相同的处理器核数进行并行运算.在每个问题实例上,所有算法均运行 20 次,并设置相同的终止时间;对各算法求得的最终解集,我们计算其相互之间的覆盖值 $C$.设两种算法求得的解集分别为 $A$ 和 $B$,$C(A,B)$定义为 $B$ 中的解被 $A$ 所支配的比例[30]:

$$C(A,B)=\frac{|\{b\in B\mid \exists a\in A: a\prec b\}|}{|B|} \tag{10}$$

算法运行的平均结果见表 2.从中可以看出:对于较小规模的问题实例,各算法得到的结果差异不大;而随着问题规模的增长,CPSO 和 HGA 算法所求得的结果越来越多地被 PSO-Agent 的结果所支配;在问题规模达到 10×14 以后,CPSO 和 HGA 的整个结果解集都被 PSO-Agent 的结果所支配.相反地,在所有问题实例上,PSO-Agent 求出的解从未被另外两种算法的结果所支配.由于所有算法使用相同的计算时间,PSO-Agent 求出的解集质量明显优于其他两种算法,这充分说明了基于主从粒子群模型的计算方法的优越性.

**Table 2** Comparative experimental results of the three algorithms on the test problem instances

**表 2** 3 种算法在测试问题实例上的比较实验结果

| Problem size ($m\times n$) | $C$ (CPSO, PSO-Agent) (%) | $C$ (HGA, PSO-Agent) (%) | $C$ (PSO-Agent, CPSO) (%) | $C$ (PSO-Agent, HGA) (%) |
|---|---|---|---|---|
| 3×4 | 0.00 | 0.00 | 0.00 | 0.00 |
| 3×8 | 0.00 | 0.00 | 2.25 | 0.00 |
| 5×5 | 0.00 | 0.00 | 2.67 | 0.00 |
| 5×10 | 0.00 | 0.00 | 9.84 | 3.75 |
| 8×6 | 0.00 | 0.00 | 44.60 | 16.35 |
| 8×12 | 0.00 | 0.00 | 75.15 | 52.19 |
| 10×7 | 0.00 | 0.00 | 89.75 | 70.53 |
| 10×14 | 0.00 | 0.00 | 100 | 100 |
| 12×8 | 0.00 | 0.00 | 100 | 100 |
| 12×16 | 0.00 | 0.00 | 100 | 100 |
| 15×10 | 0.00 | 0.00 | 100 | 100 |
| 15×20 | 0.00 | 0.00 | 100 | 100 |

## 4 结束语

本文提出了一个基于主从粒子群模型的分布式 Agent 计算框架.其主要思想是,综合运用主从并行计算模型和多 Agent 系统来实现粒子群优化算法,特别是使用异步组 Agent 来构造粒子群,群中的 Agent 共享一个 Population 并对其中的解进行协同演化,从而有效地降低了问题求解的难度、提高了求解效率.在一种典型约束多目标优化问题上的算法实验结果显示,本文提出的方法具有明显的性能优势.

对于一些超大规模的优化问题,基于单重 Master-Slave 模型的方法求解起来仍然较为吃力.下一步研究拟

将此模型扩展到多重 Master-Slave 结构,从而更充分地利用分布式计算能力来提升复杂问题的求解效率.

**References**:


[1] Bonabeau E, Dorigo M, Theraulaz G. Swarm Intelligence: From Natural to Artificial Systems. New York: Oxford University Press, 1999.

[2] Wu QD, Kang Q, Wang L, Lu JS. An Introduction to Nature-Inspired Computation. Shanghai: Shanghai Science and Technology Press, 2011. 1−3 (in Chinese).

[3] Kennedy J, Eberhart RC. Particle swarm optimization. In: Proc. of the IEEE Conf. on Neural Networks, Vol.4. Perth: IEEE Press, 1995. 1942−1948. [doi: 10.1109/ICNN.1995.488968]

[4] van den Bergh F, Engelbrecht AP. A cooperative approach to particle swarm optimization. IEEE Trans. on Evolutionary Computing, 2004,8(1):225−239. [doi: 10.1109/TEVC.2004.826069]

[5] Goh CK, Tan KC, Liu DS, Chiam SC. A competitive and cooperative co-evolutionary approach to multi-objective particle swarm optimization algorithm design. European Journal on Operations Research, 2010,202(1):42−54. [doi: 10.1016/j.ejor.2009.05.005]

[6] Nwana HS, Ndumu DT. An introduction to agent technology. Lecture Notes in Computer Science, 1997,1198:3−26. [doi: 10.1007/3-540-62560-7_35]

[7] Ahmad A, Lee YC, Rahimi S, Gupta B. A multi-agent based approach for particle swarm optimization. In: Proc. of the Integration of Knowledge Intensive Multi-Agent Systems. 2007. [doi: 10.1109/KIMAS.2007.369820]

[8] Lorion Y, Bogon T, Timm IJ, Drobnik O. An agent based parallel particle swarm optimization—APPSO. In: Proc. of the Swarm Intelligence Symp. IEEE Press, 2009. 52−59. [doi: 10.1109/SIS.2009.4937844]

[9] Kumar R, Sharma D, Kumar A. A new hybrid multi-agent-based particle swarm optimisation technique. Int'l Journal of Bio-Inspired Computation, 2009,1(4):259−269. [doi: 10.1504/IJBIC.2009.024724]

[10] Zhao B, Guo CX, Cao YJ. A multiagent-based particle swarm optimization approach for optimal reactive power dispatch. IEEE Trans. on Power System, 2005,20(2):1070−1078. [doi: 10.1109/TPWRS.2005.846064]

[11] Kargar HK, Aghmasheh R, Safari A, Govar GRZ. Multi-Agent-Based particle swarm optimization approach for PSS designing in multi-machine power systems. In: Proc. of the IEEE Conf. on Power and Energy. Johor Baharu, 2008. 73−78. [doi: 10.1109/PECON.2008.4762448]

[12] Chen P, Meng A, Zhao C. Particle swarm optimization in multi-agent system for the intelligent generation of test papers. In: Proc. of the IEEE Congress on Evolutionary Computation. Hong Kong, 2008. 2158−2162. [doi: 10.1109/CEC.2008.4631085]

[13] Li F, Zheng Q. Collaborative production scheduling and optimization of hybrid particle swarm optimization and multi-agent systems. Microcomputer Information, 2011,27(8):25−27 (in Chinese with English abstract).

[14] Zhong W, Liu J, Xue M, Jiao L. A multiagent genetic algorithm for global numerical optimization. IEEE Trans. on Systems, Man, and Cybernetics B, 2004,34(2):1128−1141. [doi: 10.1109/TSMCB.2003.821456]

[15] Xu XL, Ying SY, Wang WL A fuzzy flexible job-shop scheduling method based on multi-agent immune algorithm. Control and Decision, 2010,25(2):171−178 (in Chinese with English abstract).

[16] Bratton D, Kennedy J. Defining a standard for particle swarm optimization. In: Proc. of the IEEE Swarm Intelligence Symp. 2007. 120−127. [doi: 10.1109/SIS.2007.368035]

[17] Hu W, Li ZS. A simpler and more effective particle swarm optimization algorithm. Journal of Software, 2007,18(4):861−868 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/18/861.htm [doi: 10.1360/jos180861]

[18] Parsopoulos KE, Vrahatis MN. Particle swarm optimization method in multiobjective problems. In: Proc. of the ACM Symp. on Applied Computing. 2002. 603−607. [doi: 10.1145/508791.508907]

[19] Li X. A non-dominated sorting particle swarm optimizer for multiobjective optimization. Lecture Notes in Computer Science, 2003, 2723:37−48. [doi: 10.1007/3-540-45105-6_4]

[20] Ho SJ, Ku WY, Jou JW, Hung MH, Ho SY. Intelligent particle swarm optimization in multi-objective problems. Lecture Notes in Artificial Intelligence, 2006,3918:790−800.

[21] Tripathi PK, Bandyopadhyay S, Pal SK. Multi-Objective particle swarm optimization with time variant inertia and acceleration coefficients. Information Science, 2007,177(22):5033−5049. [doi: 10.1016/j.ins.2007.06.018]

[22] Cooren Y, Clerc M, Siarry P. MO-TRIBES, an adaptive multiobjective particle swarm optimization algorithm. Computational Optimization & Application, 2011,49(2):379−400. [doi: 10.1007/s10589-009-9284-z]

[23] Talukdar SN, Baerentzen L, Gove A, Souza PD. Asynchronous teams: Cooperation schemes for autonomous agents. Journal of Heuristics, 1998,4(4):295−321. [doi: 10.1023/A:1009669824615]

[24] Zheng YJ, Wang LL, Xue JY. An A-Team based architecture for constraint programming. Lecture Notes in Computer Science, 2006, 4088:552−557. [doi: 10.1007/11802372_58]

[25] Zheng YJ, Wang JQ, Xue JY. A-Team based supply chain management agent architecture. Int'l Journal of Artificial Intelligence Tools, 2009,18(6):801−823. [doi: 10.1142/S021821300900041X]

[26] Zheng YJ, Xue JY, Ling HF. Combinatorial optimization problem reduction and algorithm derivation. Journal of Software, 2011, 22(9):1985−1993 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/3948.htm [doi: 10.3724/SP.J.1001.2011.03948]

[27] Hajek J, Szollos A, Sistek J. A new mechanism for maintaining diversity of Pareto archive in multi-objective optimization. Advances in Engineering Software, 2010,41(7-8):1031−1057. [doi: 10.1016/j.advengsoft.2010.03.003]

[28] Gao F, Zhao Q, Liu H, Cui G. Cultural particle swarm algorithms for constrained multi-objective optimization. Lecture Notes in Computer Science, 2007,4490:1021−1028. [doi: 10.1007/978-3-540-72590-9_155]

[29] Mousa A. Efficient evolutionary algorithm for solving multiobjective transportation problem. Journal of Natural Sciences and Mathematics, 2010,4(1):77−102.

[30] Zitzler E, Deb K, Thiele L. Comparison of multiobjective evolutionary algorithms: Empirical results. Evolutionary Computation, 2000,8(2):173−195. [doi: 10.1162/106365600568202]

**附中文参考文献**:

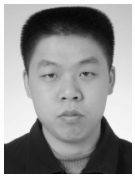
[2] 吴启迪,康琦,汪镭,陆金山.自然计算导论.上海:上海科学技术出版社,2011.

[13] 李芳,郑晴.混合粒子群算法和多 Agent 系统的协同生产调度及其组合优化.微计算机信息,2011,27(8):25−27.

[15] 徐新黎,应时彦,王万良.求解模糊柔性 Job-shop 调度问题的多智能体免疫算法.控制与决策,2010,25(2):171−178.

[17] 胡旺,李志蜀.一种更简化而高效的粒子群优化算法.软件学报,2007,18(4):861−868. http://www.jos.org.cn/1000-9825/18/861.htm [doi: 10.1360/jos180861]

[26] 郑宇军,薛锦云,凌海风.组合优化问题简约与算法推演.软件学报,2011,22(9):1985−1993. http://www.jos.org.cn/1000-9825/3948.htm [doi: 10.3724/SP.J.1001.2011.03948]



**郑宇军**(1979－),男,福建莆田人,博士,工程师,CCF 会员,主要研究领域为人工智能优化,基于 Agent 的软件工程.

**凌海风**(1972－),女,博士,副教授,主要研究领域为人工智能优化与应用.

**陈胜勇**(1973－),男,博士,教授,博士生导师,CCF 高级会员,主要研究领域为智能信息处理,机器学习.

**徐新黎**(1977－),女,博士,副教授,主要研究领域为人工智能优化.