上海交通大學

SHANGHAI JIAO TONG UNIVERSITY

学士学位论文

BACHELOR'S THESIS

论文题目：基于 CFD 和近似模型的船型优化技术开发

学生姓名：王木

学生学号：5120109148

专　　业：船舶与海洋工程

指导教师：万德成

学院(系)：船舶海洋与建筑工程学院

基于 CFD 和近似模型的船型优化技术开发

摘要

基于先进的 CFD 技术，并结合近似模型及最优化技术，船舶设计领域打开了崭新的局面。本文考察了国外在这一方面的研究成果，并且详细阐述了基于 CFD 和近似模型的船型优化技术的整个流程及其关键技术。本文在单目标及多目标全局最优化算法及并行计算方面做出了贡献。

首先，作者对比了几种随机搜索算法并论述了各自的优劣。随后，重点研究了单目标 PSO 算法，并改进了该算法。在非多峰函数和多峰函数的测试下，均取得了很好的结果。在此基础上，引入 Pareto 非劣解的概念，重点研究了多目标粒子群优化算法。在阅读了大量文献的基础上，比较了各种多目标粒子群算法的变种，做了详细的评述。最终完成的 IMOPSO(Improved Multiple Objective Particle Swarm Optimization)算法具有收敛速度快、靠近 Pareto 前沿且稳定的特点。然后，针对粒子群算法的可并行性特点，开发了并行优化算法，大大加快了计算速度，进一步缩短了设计周期。

关键词：船型优化设计，CFD，近似模型，最优化算法，粒子群优化算法，并行计算

DEVELOPMENT OF SHIP TYPE OPTIMIZATION TECHNOLOGY BASED ON CFD AND APPROXIMATE MODEL

ABSTRACT

Nowadays, computer technology has changed the world. With its rapid development and the continuous improvement of optimization theory, the attention from the field of ship type design has been concentrated on its big future. The concept and connotation of ship type optimization technology based on CFD and approximate model are analyzed in detail. The key technologies, such as global optimization technology and parallel computing, have been focused on.

First of all, the author compares several random search algorithms. Then, the single objective PSO algorithm is studied, and the algorithm is improved. Very good results have been obtained under the test of non-multimodal function and multi peak function. Based on it, with the concept of Pareto non dominated solution, the multi objective particle swarm optimization algorithm is mainly studied. On the basis of reading a large number of documents, a variety of multi objective particle swarm optimization algorithm is compared. The final completion of the IMOPSO algorithm has the features of fast convergence, close to the Pareto front and strong stability. Then, thanks to the characteristics of the particle swarm optimization algorithm, the parallel optimization algorithm is developed, which greatly accelerates the calculation speed and further shortens the design cycle.

Key words: Ship type optimization design, CFD, approximate model, optimization algorithm, particle swarm optimization algorithm, parallel computing

目 录

第一章 绪论

1.1 研究背景和意义

船型优化设计需要使用水动力性能工具来进行准确的数值预报，从而使得可靠性能够得到保证。以往的船型设计需要按照约定俗成的规则，根据系列模型试验数据和母类船舶型线，由人工经验对目标船舶的外形进行修改；随后，一方面，利用近年来发展期来的可以逐渐取代势流理论的 CFD 软件来进行辅助分析；另一方面，则是模型制作进行试验。传统的设计模式严重制约了船舶型线的创新设计能力，因为它强烈依赖于母船型数据和设计人员的经验；模型试验的资源消耗巨大并具有长周期的严重缺点；而且想要得到最优秀的船型设计方案是异常困难的，通常得到的方案仅能满足设计规格，虽然可行，但性能不佳。

上述问题有其历史背景。在计算机技术尚未发展的时期，传统的“正向”思维可以在一定程度上解决这个问题。然而，随着计算机技术和理论的不断发展，出现了 SBD（Simulation Based Design）技术。它通过计算流体力学数值评价技术（或者利用已很完善的势流理论进行评估）和现代优化技术，在给定的设计空间内来优化船舶的一项（或某几项）性能。最终得到的船型具有给定约束下的最优性能。

1.2 船型优化技术简述及国外船型优化技术研究进展

因为利用 CFD 软件来进行船型优化设计会消耗大量的时间和计算资源，如果在整个阶段完全依靠 CFD 来进行计算，难度非常大。计算资源和时间及准确性，需要一个平衡，对此，学者提出了两种方法：一种缩短计算时间的方法是借助高性能计算或者并行处理技术来提升效率；另一种，是在优化过程中建立一个近似模型，该近似模型能够模拟实际工程问题中复杂的映射关系。目前常用的代替实际模型的近似模型有以下这些[1]：VFM 模型、RBF 模型、RSM 模型、Kriging 模型等。通过使用这些近似模型，可以极大地提高设计效率。

(1) 简约策略[1]

为了促进面向实际工程应用的 SBD 技术，Campana 和 Peri 等（2008~2009）将解决高精度 CFD 数值计算带来的响应长度和计算成本作为目标，认真归纳评述了近似技术中各种不同的简约策略。他们的研究表明，不同的近似方法由于其具有各自的特点，他们适合处理的问题是有差异的。

(2) 最优化技术[1]

鸟类的群体性捕食行为与寻优过程极为相似。Eberhart 和 Kennedy 这一行为经过长期的观察及总结分析后，首先提出了基本的 PSO（Particle Swarm Optimization）搜索算法。随后，Coello 和 Lechuga 于 2002 年提出了一种多目标粒子群优化算法——CMOPSO（Cloud Multiple Objective Particle Swarm Optimization）。Liuzzi 和 Campana（2009）对粒子群优化算法也进行了研究，提出了一种新的用于解决复杂优化问题的多目标全局优化算法——DDFPSO（Deterministic Derivative-Free Particle Swarm Optimization），并且使用标准测试函数得到的结果表明该算法的收敛性很好，不容易陷入局部寻优或早熟的情况，而且效率较高。

(3) 船体几何重构[1]

Yang 和 Kim（2010）在 $Fn=0.33, 0.305, 0.22$ 三个速度下的总阻力作为目标函数，分别采用了基于径向基差值函数的船体局部几何重构方法、基于 Lackenby 变换的船舶整体几何

重构方法以及将二者结合起来进行重构的方法，对系列 60 线型进行了优化设计。总阻力由兴波阻力（基于 NM 理论的高效评估器计算）和摩擦阻力（ITTC 公式计算）相加获得，优化算法采用多目标遗传算法。优化结果表明：基于 Lackenby 的整体几何重构方法，总阻力的收益对应的三个航速分别为：-0.34%、2.12%、2.32%；基于局部几何重构方法，总阻力的收益相对较大，对应的三个航速分别为：2.45%、9.04%、5.71%（波幅云图比较如图 1.9 所示）；基于整体与局部相结合的几何重构方法，总阻力的收益最大，对应的三个航速分别为：1.45%、11.65%、7.97%。此外，Kim 采用上述方法还对 Wigley 船型和 KCS 船模进行了优化设计研究。

综上所述，在多项关键技术上，国外研究人员已经取得了较大的进展，并且在船舶设计方面已经开始尝试应用这些技术。

1.3 研究思路及理论

船型优化设计思路流程图如下所示：

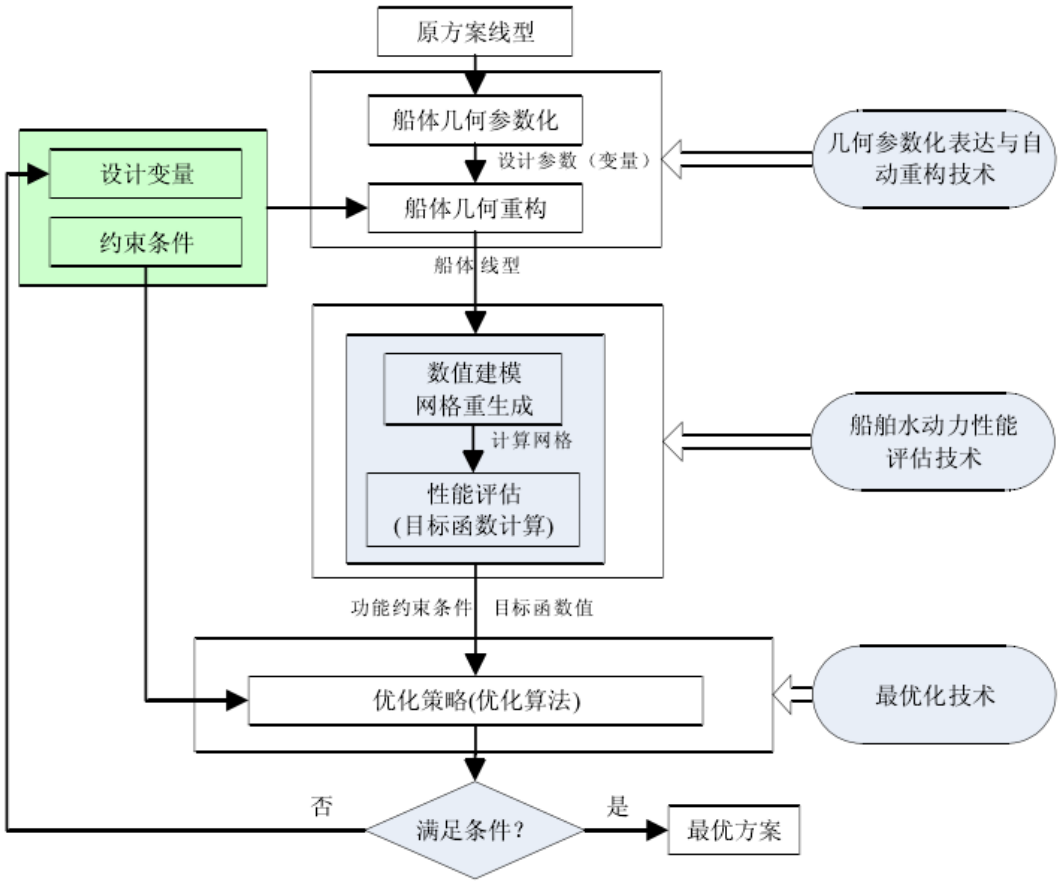

Fig.1 船型优化设计思路流程图[1]

在研究方法阶段已大致讨论过各项技术理论，现在选择其中几项进行较详细的介绍。

(1) 船体几何重构技术

最优化算法和船体阻力性能、快速性能等的分析评估并不能直接结合，它们之间需要船体几何重构技术来进行黏合，这是整个设计流程中相当重要的一个环节。从船型方面入手对船舶的性能进行优化时，需要参数化外型。并且，在进行参数化表达时，还有以下条件必须满足：1、参数应该尽量少，因为参数过多难建模型，在优化搜索时参数过多效果也不好；2、抓准参数间的联系。算法寻优后，所得到的结果再反向复原为船体的实际外形，经过这一调整就得到了该阶段最终的结果。然而，整个船型优化设计中的一个难点也在于此：不同

的船型可能适合不同的设计变量取法，但是为了保证统一与简便性，需要让某一套取法方案能够尽可能多地覆盖不同的几何构型。

当面对一个船型优化设计问题时，首先需要确定的就是采用何种方法来重构船体的几何外形。这种方法一旦得以确定，那么整个问题的设计变量空间就确定下来了。在后面的寻优过程中，变量的搜寻范围被死死地限定在这个空间内。因此，几何重构方法直接决定了船型优化问题的设计空间“大小”[3]。

按照船型参数化方式的不同，可以将这些重构方法大致分为两类：1、基于几何构型技术的方法（如自由变形法、Bezier Patch 法等）。该方法首先在欲改变外形的面上布置若干控制点，而后通过调整这些点的空间位置来达到变形的目的；2、基于船型参数的方法（如 Lackenby 变换法、参数化模型法等）。该方法通过调整这些参数来改变船体的外形。当然，不管采用何种方法，都必须满足下列条件：

a) 设计变量的整个取值范围反向映射到几何构型时，能够取得丰富的船型结果。
b) 在满足前一个条件的情况下，尽力减少设计变量的数目从而削减计算成本。
c) 在满足前两个条件的情况下，尽可能使得生成的船体外型曲面足够光顺。

但是，若要将上述三个条件完全施加于复杂船型的重构上，是比较困难的。由此可见这一技术在船体几何外形的优化上是相当重要而关键的。

于 1950 年提出的经典 Lackenby 方法[4]大大推进了船体几何重构技术的发展。这一方法的关键就在于形变函数$\Delta X = C(1-X)(X+D)$的引入。通过利用该函数，研究人员能够方便的由几个基本的船型参数（如 C_p, L_p, X_{cb} 等）直接计算出各个站位的横剖面在纵向上的改变量ΔX。由于该形变函数有着很好的实用性，它常被应用于由母型船变化得到新船型的设计方法中，而且效率相当高。但设计人员在使用该方法时，需要注意选择合适的参数，这需要大量的经验才能得到保证；否则，在变换过程中可能会出现的错误不会马上显现，这需要到整个变换工作结束后才能看出成败。产生上述问题的原因在于，基本的 Lackenby 变换法在解方程时忽略了一个高阶小量。

在经典 Lackenby 方法被提出之前，船体的外形变换通常都是使用线性变换函数。该方法最突出的贡献就是提出了适用性更广的多项式变换。此后，商用软件 FRIENDSHIP 优化了经典的 Lackenby 变换方法，它引入了功能性曲面优化方法，使该方法能够更好地应用于实际的工程问题。计算机辅助设计（CAD）通常使用下述三种方法进行建模：1、传统建模；2、半参数化建模；3、全参数化建模。这三种方法在 FRIENDSHIP 中均可实现。

(2) 最优化算法

最优化算法在众多领域都有着广泛的应用。除开传统学科，新兴的运筹学、管理学等也依靠最优化技术取得了巨大的发展。根据其基本的数学原理，最优化算法可以被分为以下两类：1、基于梯度理论的优化算法；2、随机搜索算法。

基于梯度的方法主要包含这些：1、变梯度法（CGM）；2、序列二次规划（SQP）；3、最速下降法（SDM）；4、序列线性规划（SLP）等。这些传统的优化算法虽然有着较高的计算效率，却有着几个严重的缺陷：1、从实际工程问题抽象出来的模型需要足够精巧才能应用这些方法；2、需要认真选取初始搜索点，算法稳定性差。虽然传统优化算法通常只需要一个初始搜索点便能开始优化迭代，其复杂性低于随机搜索算法并且易于实现。然而只有单个搜索点，我们能得到的搜索信息很有限并且效率低下。基于梯度理论的传统优化算法的另一个严重问题表现在这类算法非常容易陷入局部寻优的情况而导致早熟收敛，在面对多峰函数时，基本不可能得到整个设计空间的最优解。

与基于梯度理论的传统优化算法相比，随机搜索算法在各方面的总体表现均要优秀得多，而且这类算法的适用范围也更加广阔。在使用这一类搜索算法时，并没有必须从实际工程问题抽象出一个精巧的数学模型这一要求，它仅需一系列实际的目标函数值就能通过某种简便

的方法转换为适应度，然后就能自我调整搜索的方向或者其他参数。并且随机搜索算法可以直接使用目标函数或者个体的适应度，从而也能用于解决很难求导甚至无法求导或导数不存在的函数作为目标函数的问题。此外，随机搜索算法还能在搜索过程中，自适应地调整整个搜索空间的大小，从而避免无意义的计算资源耗费，提高了搜索效率。这类算法的特点是，由一个初始群体（由很多个随机初始化的个体组成）开始进行迭代最优搜索，每一次迭代，都将适用于当前群体的进化算子作用于这个群体上来产生出下一代群体。此外，每一次迭代都会产生一些有用的信息，例如粒子群优化算法可以利用这些信息来自适应地调整惯性权重或者学习因子的大小从而提高优化效率。

随机搜索算法对不同种类的问题都有很强的健壮性，通常都能获得全局最优解，这是因为它能够不依赖于具体的设计问题的具体领域，而能提供一种通用的框架。目前来说，应用得较多的较为经典的随机搜索算法主要有模拟退火算法（SA）、非劣解排序遗传算法（NSGA）、粒子群优化算法（PSO）等。

1.4 船型优化技术的关键技术介绍

现阶段所使用来优化船型的数值模拟分析技术主要是 CFD 技术，另外还需要构建近似模型来辅助计算以缩短计算时间。在具体操作上，还包括设计变量选择、样本空间构成、船体几何重构、生成计算模型（前处理）、优化算法以及收敛判断、循环算法设计等问题。

在研究时，通常会使用不同的策略进行组合来研究上述各个问题。在本课题的初始研究阶段，针对每个问题，理论上应该多考虑几种方法。但同时由于效率的原因，针对某一问题，由于已经有研究确定的相对更高效的方法，则优先选择。船体几何重构技术分为整体重构和局部重构，分别采用 Lackenby 和 FFD 或 Bezier Patch 方法。样本空间构成针对不同的性能评估有不同的组合。

查找优化算法相关资料，发现目前有三种最优化算法具有较大的可行性：遗传算法（GA）、粒子群算法（PSO）、差分进化算法（DE）。下面对这三种算法的编写可行性、实用性、性能等方面进行简要归纳。

(1) 编码标准

遗传算法采用二进制编码，粒子群优化算法和差分进化都采用实数编码[2]。近年来许多学者通过整数编码将 GA、PSO 应用于求解离散型问题，特别是 0-1 Non-Linear Optimization 问题、整数筹算问题等，而离散的差分进化算法则研究的比较少，而采用 Hybrid Encoding 技术的进化差分算法则研究更少。

(2) 参数设置问题

进化差分算法较之另外两个优化算法来说，它更容易使用，因为它主要需要调整的参数只有两个，而且参数的调整对结果的影响并不大。另一方面，遗传进化算法和粒子群优化算法有较多的系数需要仔细设置，例如前者的交叉、变异概率，后者的惯性权重和学习因子，并且算法的寻优性能对这些控制量的改变相当敏感，因此降低了使用的简便性。

(3) 高维问题

在使用这些方法处理维度较高的问题时，需要投入很大的注意，这是由于实际生活中常遇到的就是有不止一个变量的问题。大量经验表明，在面对高维问题时，粒子群优化算法和进化差分算法的收敛性很好并且效率较高，然而遗传进化算法则表现欠佳，常会遇到早熟收敛的问题。

(4) 收敛性能

在收敛速率上，对有差异的优化设计目标，PSO 算法那相对于进化差分算法和遗传进化算法来说，耗费时间少很多。然而，基本粒子群优化算法在面对多峰函数时有着容易在局部寻优进而早熟的缺陷，此外，它的稳定性也较差。

(5)应用广泛性

遗传进化算法已经被世界上的研究人员应用了几十年；粒子群优化算法从上个世纪末真正实现以来，已经在世界范围内掀起新的研究热潮，在各领域的实际应用也逐渐增多；这个世纪初才逐渐投入应用得进化差分算法则由于其性能较好，渐渐吸引了研究人员的注意。

这三种算法具体的介绍及比较将在第二章第二节进行，并会给出选择粒子群算法的原因。

第二章 全局最优化方法开发及并行算法的开发

2.1 引言

最优化技术在改进船体几何外形的问题上，有着举足轻重的地位。这是因为船体外形的变化对于其整体性能的影响是难以用简单函数表示的。最优化技术自从作为一门独立的学科以来，已经在机械、船舶、航空、航天等诸多工程设计领域大放异彩，得到了广泛的应用，这门技术依靠的理论也发展得较为完善。但是，由于各种优化算法有着各自独有的指导思想，它们最适合处理的问题是不一样的。另外，由于船舶航行性能优化问题通常还需要进行多目标方面的优化，故有开发多目标寻优算法的必要性。解决这一问题的传统做法是给各目标赋予一定的权重进行加权来转化为单目标问题，然后再选择一个适合该问题的单目标搜索算法来进行求解。然而这种方法存在着严重的缺陷，因为我们常常不清楚各个目标到底该分配一个多大的权重。另外，在面对复杂非凸问题时，传统做法往往不能得到全局的最优解，容易陷入局部最优的情况。因此，本章在第 5 节将重点阐述 Pareto 非劣解的概念，并基于此开发改进的多目标粒子群优化算法。

综上，针对船型优化设计的目标函数的特点，本章重点探讨了不同优化方法的效果并且对最优化方法进行了改进，开发了改进的全局单目标和多目标粒子群优化算法。同时，对给出的算法，做了详细的的测试，并提供数据和图形进行参考。

2.2 最优化方法的选取

本研究小组之前使用的处理单目标和多目标问题的最优化方法均是遗传算法以及它的变种。在查阅了大量资料了解了船型优化的特点后，本文选择了较新的粒子群算法作为最优化方法。

(1) 粒子群算法（PSO）

1) 基本原理

Eberhart 和 Kennedy 于上个世纪在仔细观察鸟类群体捕食行为后，提出了基本的单目标粒子群优化算法，其算法主要思想可以这样描述：

目标搜索空间的维度数用 D 表示，随机初始化一个种群，其含有的粒子总数为 m。对于第 i 个粒子来说，它的位置使用$x_i = [x_{i_1}, x_{i_2}, \cdots, x_{i_D}]$这个矢量来描述，它的飞行速度用$v_i = [v_{i_1}, v_{i_2}, \cdots, v_{i_D}]$来描述。另外，它曾达到过的最优解的变量组合使用一个矢量表示，保存为$p_i = [p_{i_1}, p_{i_2}, \cdots, p_{i_D}]$，整个群体曾经到达过的最优解的位置用$p_g = [p_{g_1}, p_{g_2}, \cdots, p_{g_D}]$这一个矢量来保存描述。同时，每经过一次迭代，群体中的每个粒子都需要更新其速度和坐标向量。见下式：

$$v_i(\mathrm{n}+1) = v_i(\mathrm{n}) + c_1 r_1 (p_i - x_i(n)) + c_2 r_2 \left(p_g - x_i(n)\right) \quad (1)$$

$$x_i(\mathrm{n}+1) = x_i(\mathrm{n}) + v_i(\mathrm{n}) \quad (2)$$

式中，i=1,2,⋯,m，表示不同的粒子。c_1、c_2 在通常情况下均取作 2，它们被称为学习因子（或者叫加速系数），分别用来调整某个粒子朝向自身曾经得到的历史最优点的坐标飞行的速度和朝向这个群体曾经共同得到的历史最优点的坐标飞行的速度；随机数 r_1、r_2 的取值范围为(0,1)；n 则代表目前已经迭代的步数。

另外，需要将每个粒子的速度限制在一个区间内（记做$[-v_{max}, v_{max}]$）。通过施加这样的限制，我们能够防止粒子飞行得过于快而错过最优解（v_{max} 的大小一般对不同的问题设置不同的值）。最后，一般来说，在碰到以下这两种情况时均可结束迭代，输出最优解：1、达到之前设定的最大步数；2、粒子的飞行速度足够小。

粒子群优化算法的背后意义可以从社会学方面来探讨[5]。第(1)个公式的第一部分存储的是粒子之前的速度矢量，这项被称为“记忆”项，它的内在含义是上次迭代的某粒子的速度会影响下一次迭代的该粒子的速度；这个公式的第二个部分存储的是某粒子现在的位置到曾经到达的最优解的位置的一个矢量，这项被称为“自身认知”项，它的内在含义是粒子通过自己的“思考”计算出当前迭代步数的行动受到之前经验的影响；这个公式的第三个部分代表某粒子当前迭代步数的坐标到整个群体曾经到达的最优点的坐标的一个矢量，这项被称为“群体认知”项，它的内在含义是粒子通过“思考”计算出当前迭代步数受到整个群体经验的影响。通过这样类似鸟类捕食行为的个体及群体性思考，整个群体的每一个粒子都能自动地调整自身的飞行方向和速度。

2) 基本 PSO 算法流程

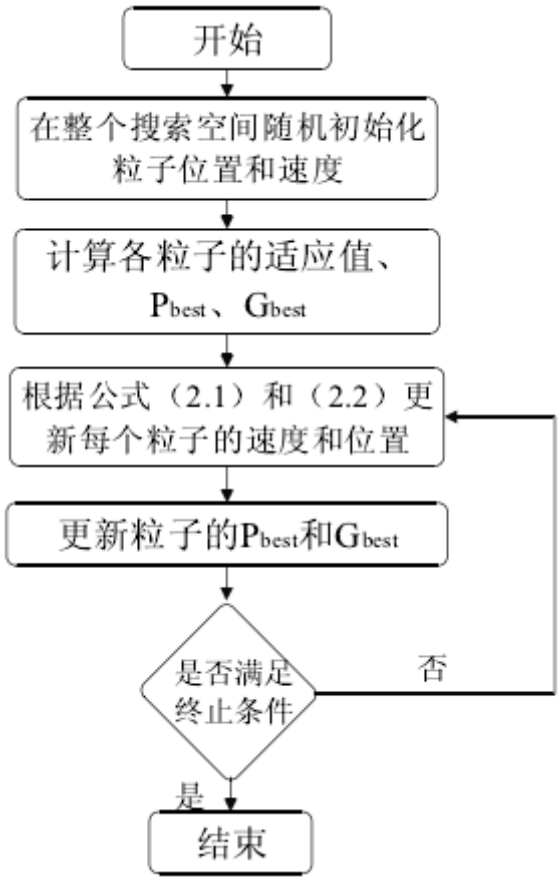

Fig.2 基本粒子群优化算法流程图[1]

(2) 差分进化算法（DE）

进化算法（EA）的一个重要分支——差分进化算法，由于在上个世纪末的几大比赛中有着优秀的表现，从而俘获了许多研究人员的注意，并且随后获得的成果相当丰富。

该算法对个体的优劣评价通过对已定义好的适应度函数作一简单映射来实现，这与被解决的问题有关。基本的差分进化算法实现过程如下[2]：

第 1 步：探讨确定进化差分算法的种群数量、交叉算子、变异算子、最大迭代步数、终止条件等控制参数和所采用的具体策略。

第 2 步：产生一个随机的初试种群，由一定多个随机的个体组成，此时的迭代步数（进化代数）为 t=1。

第 3 步：对整个群体中各个个体进行评价（计算它们的适应度大小）。

第 4 步：对是否需要终止迭代（进化）进行判断。在出现下述情形时，终止迭代：1、达到终止条件；2、进行到最大迭代数。另外，在终止迭代后，需要输出最优解。

第 5 步：对群体中的每个个体都按一定概率作用上变异算子和交叉算子（同时要用边界条件进行修正），得到新的临时性群体。

第 6 步：对得到的临时性群体中的每一个个体都进行评价（计算它们的适应度大小）。

第 7 步：对得到的群体中的个体进行选择从而得到新的群体。

第 8 步：迭代步数（进化代数）t=t+1，转至第 4 步。

在选择进化差分算法的几个控制变量时，有一些经验规则可以参考（因为 DE 的控制参数对全局优化来说影响相当大）：

a) 种群数量：

大量的经验表明，合理的种群数量与预求解问题的维数有较大的关系，通常在[5D,10D]这个区间内选择。其中 D 为维度数。而且为了保证 DE 具有足够的个体进行变异，NP 至少应为 4。

b) 交叉算子：

交叉算子 CR 是用来控制一个矢量来自另一个随机选择的矢量而不是自己本身这个矢量的概率的参数。它是一个在 0 到 1 之间的实数。通常情况下，我们会选择 0.1 作为交叉算子的值。然而在大多数情形下，CR 值越大越能帮助我们加快速度收敛。因此我们通常选择 0.9 或者 1.0 最为 CR 值来看是否能获得一个粗略的解。

c) 变异算子：

变异算子 F 是用于调节偏差向量的放大比例的一个参数。它是一个在 0 到 2 之间的实数。到目前为止，大量研究表明，F 若取为 0.5 通常能得到一个较好的结果，当它大于 1 或者小于 0.4 时则常常失效。当你发现群体收敛得过早时，你应该增大编译算子的值或者群体的个体数量。

d) 最大迭代步数：

最大迭代步数是用来控制算法终止的。当迭代到最大步数时，程序停止搜索并打印出目前为止找到的最优点的坐标和适应度。它的取值通常在 100 到 200 这个区间内。当具体问题需要较高的求解精度时，可以突破这个范围继续增大最大迭代步数，当然这样会耗费更多的计算资源。

e) 终止条件：

一般来说，会有另外的判定条件来作为 DE 的终止条件。例如，一般当适应度值的变化（率）小于之前设定的精度值时程序终止。

(3) 遗传算法（GA）

在处理单目标问题上，遗传算法的思路是模拟达尔文的《进化论》提出的自然选择机制和遗传定理构建的计算模型，它通过模拟自然进化过程来搜索到最优解[6]。它的基本运算过程如下：

第 1 步：初始化。设置迭代器 iter=0，设置最大迭代步数 iterMax。初始化一个随机种群。

第 2 步：个体评价。对整个群体中各个个体进行评价（计算它们的适应度大小）。

第 3 步：选择。对群体中的各个个体都按概率 1 作用上选择算子（同时要用边界条件进行修正），得到新的临时性群体。

第 4 步：交叉和变异。对群体中的每个个体都按一定概率先后作用上交叉算子和变异算子（同时要用边界条件进行修正），得到新的临时性群体。

第 5 步：淘汰。本次迭代中表现最差的个体被淘汰，将其替换为本次迭代中表现最优的个体（此步为改进基本遗传算法的新添加步骤，可以有效加快收敛）。

第 6 步：终止判断。

在将基本遗传算法拓展应用至处理多目标问题上，最经典的是上世纪末提出的非劣解排序遗传算法（NSGA）。该算法的核心在于引入了 Pareto 非劣解的概念。除此之外，它在基本遗传算法的基础上仅仅引入支配关系的分层这一技术。由于采用了这一技术，NSGA 算法可以增大优良个体的基因遗传到下一代的几率。2000 年，Deb 又针对 NSGA 的几个缺陷，开发了 NSGA-II 算法。该算法对 NSGA 的改进主要表现在：1、改进了标准 Pareto 算法，缩短了计算时间；2、提出了拥挤度的概念。

然而，正如之前所提到的，由于各种优化算法有着各自独有的指导思想，它们最适合处理的问题是不一样的。同时，带精英策略的非劣解排序算法（NSGA-II）还有着这些缺陷：1、算法过程过于复杂，难以实现；2、调节参数数目过多，难以调节，需要大量经验才能熟练使用；3、面对非线性、多峰等复杂函数时，优化效果差强人意等问题。在查阅相关资料后，发现粒子群优化算法能够很好地克服上述问题。该方法在科学研究和工程实践中的广泛应用便可以说明其优越性。鉴于以上所述，本文将主要针对求解船型优化设计问题，对粒子群优化算法进行研究、改进并开发出实用的版本。

2.3 单目标粒子群算法开发

PSO 算法是一种模拟群体智能的算法。Eberhart 和 Kennedy 于上个世纪在仔细观察鸟类群体捕食行为后，提出了基本的单目标粒子群优化算法。虽然 PSO 算法也采用了“群体”和“进化”的概念，以及仅计算个体的适应值大小，它仍有较大的不同：

PSO 算法并不会在每一次迭代过程中，对各个粒子作用上进化算子，而是将其视为 D 维设计变量空间中的一个粒子（仅作为一个点，无其他物理属性），它在这个空间中以某一速度飞行，并按照个体和整个群体的历史信息来自动地进行导向并调整速率大小，从而更新粒子的状态实现进化。

该算法的基本版本的流程已在前一小节有过详细叙述，故此处不再赘述。然而由于该算法生成的群体中的各粒子很依赖于自己本身和整个群体的历史信息，这些粒子很容易快速聚集起来。前述的粒子群算法的这一快速趋同效应，使得该算法天生就带有几个重要缺陷：1、容易陷入局部极值；2、早熟收敛；3、停滞现象。另外，算法中的几个参数也会对其性能有很大的影响。因此，世界上的研究者不断提出了各种改进措施来克服前述的不足。本节将从四个方面分别进行简单的介绍：1、种群初始化优化；2、参数选择优化；3、邻域拓扑；4、混合策略。

(1) 种群初始化优化

薛明志[7]于 2005 年提出了采用正交设计方法对种群进行初始化的方法，刘衍民[8]于 2011 年也进行了相关的研究；Campana[9]等人于 2006 年使用线性动态系统改写了标准 PSO 迭代公式，并且研究了种群初试位置这一问题，他们提出的解决办法能够使粒子的运动轨迹相互正交；Clerc[10]于 2008 年发现使用均匀分布的随机数对初始种群进行初始化虽然很容易实现，但是对高维问题很不友好、效果差，并且比较了另外三种进行初始化的方法。

(2) 参数选择优化

该算法的参数主要包括以下几个：1、各维最大速率 v_{max}；2、惯性权重（收缩因子）w；3、学习因子 c_1、c_2 等。下面详细阐述优化这几个参数的方法。

1) 各维最大速率的选择：

式(1)所示的粒子在各维上的速率大小及方向均是随机的，并受自身及群体的历史信息影响。由于这种随机性，粒子的运动是毫无规律可言的。此外，粒子还很

容易在搜索空间内围绕某一个点往复运动，这种无意义的运动会耗费许多计算资源。因此，我们往往会给速度一个限制区间[-v_{max}, v_{max}]，来抑制粒子的这种无规律的往复循环跳动。较大的 v_{max} 对于全局寻优来说更为有利；同时，较小的 v_{max} 则对局部寻优来说更为有利。但是若最大速度过大，粒子的运动轨迹则会变得杂乱无章，整个搜索过程难以收敛；同时，过小的最大速度又会使搜索较易陷入局部寻优，因为粒子的运动步长过于短。以往，最大速度一般规定为搜索空间的百分之 10 到 20 之间，这是根据经验给定的。

2) 惯性权重（收缩因子）的选择：

Shi 和 Eberhart 首次在速度迭代公式中引入了收缩因子 w，即$v_i(\mathrm{n}+1) = v_i(\mathrm{n}) + c_1 r_1(p_i - x_i(n)) + c_2 r_2\big(p_g - x_i(n)\big)$，来改善基本 PSO 算法的收敛性能。他们于 1998 年在 IEEE 国际进化计算学术会议上发表的题为"A Modified Particle Swarm Optimizer"的论文阐述了这一想法[11]。

粒子的搜索能力对收缩因子的大小改变非常敏感。这一参数能够直接控制粒子之前的速度对现在的影响程度。以往使用的基本 PSO 算法使用的是固定的收缩因子，这在很多情况下并不能兼顾搜索前期和后期的搜索质量。这是因为使用固定的 w 浪费了搜索过程中获得的大量信息。后来经过大量的研究发现，在搜索过程中根据当前获得的信息不断地调整惯性权重的值能够获得更好的优化结果。理论分析认为，这是由于当 w 值较大时，收敛速率会加快；反之，当 w 值较小时，能够让粒子在当前找到的最优解附近进行细致的搜索。因此，通过动态调节 w 的值，我们能够把握收敛的快慢和在局部进行细致寻优间的平衡。目前采用较多的是 Shi 建议的线性递减权值（Linearly Decreasing Weight, LDW）策略：$\mathrm{w} = w_{max} - n\frac{w_{max}-w_{min}}{n_{max}}$。w 的取值范围通常为[0.1, 0.9]，一般是随着迭代步数的增大而逐渐变小的。这一线性变换使得 PSO 算法在搜索初期能够很快地收敛到一个较优的解，而在后期又能在局部进行细致的寻优。w 的引入极大的提高了基本 PSO 算法的性能，而且能够通过调节 w 来针对不同的搜索问题，从而达到令人满意的全局和局部搜索效果。引入了该改进策略的 PSO 算法被称为标准粒子群优化算法（Standard PSO, SPSO）。

3) 学习因子的选择：

学习因子 c_1 用来影响粒子朝向自己历史最优点飞行的速度；而学习因子 c_2 则用来影响粒子朝向整个群体曾找到的最优点飞行的速度。此外，经验表明，在通常情况下，学习因子的值应有以下的限制：$c_1+c_2 \leq 4.0$（c_1、c_2 均为正数），并且通常取 $c_1 = c_2 = 2$。

(3) 邻域拓扑

PSO 算法的一种分类依据是粒子的领域情况。若其为整个群体，则为全局模型，否则为局部模型。全局模型的特点是：群体中各个粒子都会与除自身外的粒子交换所获得的经验，并且各粒子都会趋向于朝着全体粒子曾经找到的最优点移动。然而，全局模型虽然能够较快的收敛，但是却有着容易陷入局部极值的缺陷，这一点 Kennedy 已于 1999 年在论文中指出。有很多研究人员用来克服这一缺陷的方法是，定义一个领域的概念，并限制它的大小，让粒子仅在该范围内交换各自获得的经验。他们由此提出的局部模型主要有[12]：空间领域（Spatial Neighborhood）、性能空间（Performance Space）领域和社会关系领域（Sociometric Neighborhood）。现对这三个模型作一简要介绍：1、空间领域模型对领域的划分所按照的依据是各个粒子之间相距的远近。该模型常用的方法是引入一个根据时间自动调整的领域算子：在迭代搜索的第一步之前，将领域限定为各粒子

本身所在的位置；随着搜索的深入，领域范围会自动地调整变大，直至最后扩大到整个群体。2、性能空间模型对领域的划分则是按照一系列性能指标（如目标函数值、适应度等）来进行的。此外，还要按某种方法（如适应度距离比值，fitness-distance-ratio）来为某粒子选择其相邻粒子。3、社会关系领域模型对领域的划分采用了更加与众不同的方法，它事先将粒子存储的位置表达为一个拓扑阵列并加以编号，而后按照这个索引编号进行领域的划分。该领域模型用来编号所采用的拓扑手段主要有：轮形拓扑（wheel topology）或星型拓扑（star topology）、环形拓扑（ring or circle topology）、塔形拓扑（pyramid topology）以及随机拓扑（random topology）等。上面介绍的各种拓扑手段都有着自己独有的特点，因此对于不同的优化问题，它们的表现会有较大差异。但是经验表明，随机拓扑手段能够在满足一定的性能要求下，处理大多数问题，故使用该拓扑手段是相当方便的。

(4) 混合策略

由于各种搜索算法有着自己独有的优势，许多研究人员想到了将各种优化算法的长处混合起来的方法。通过这一混合策略，可以达到与改进 PSO 各控制参数相近的某方面性能（例如收敛速度、精度、局部寻优能力等），甚至超越它。通常有两种途径来进行混合：1、混合其他优化方法，使得算法能够在迭代过程中自动调整惯性权重（收缩因子）、学习因子等控制参数；2、将其他随机优化算法中使用的操作个体的算子与 PSO 算法进行结合。闫元元等于 2011 年在其论文中将 GA 算法中的交叉算子和编译算子引入了 PSO 算法中来避免过早收敛；兰登等于 2005 年在其论文中提出了将模拟退火算法的核心思想引入 PSO 算法的策略；Clerc[10]则采用了将禁忌技术与 PSO 算法结合的策略；Santos 于 2007 年采用了将模糊理论与 PSO 算法结合的策略。

从前面所介绍的改进基本 PSO 算法的策略可以看出，若采用领域拓扑或者混合策略等方法来进行改进过于复杂，而改进群体的初始化方法或者算法参数则比较简单并且容易实现，效果也不错。因此本文要介绍的主要要改进的方面有：群体初始化方法优化、算法参数优化。以下将详细阐述：

(1) 群体初始化方法优化——试验设计法

SPSO（Standard PSO）算法采用的方法是利用计算机自动生成的随机数来初始化粒子群，但是这种方法存在一个问题：有可能会使群体的粒子不能均匀的散布到整个设计空间。本文所采用的解决这个问题的方法是引入“试验设计”的思想到 PSO 算法中来，也就是说，采用试验设计方法来初始化粒子的速度和在空间的坐标。如此进行初始化能够稍微减少种群的规模而又不会对优化结果有过多的影响（因为效率是优化时必须考虑的一个因素）。试验设计方法有正交设计方法、拉丁方方法等，本文将采用拉丁方方法来进行优化改进，具体技术将在下面介绍：

拉丁方是一种平衡试验顺序的技术，采用拉丁方方法设计能够减少试验顺序对试验的影响。在试验设计时，通常会采用一种叫做拉丁方格的辅助手段。拉丁方格是一个正方形矩阵，它内部元素是由待排序的几个设计变量定下的。举一个简单的粒子，现在有行和列两个变量，处理数是 6，则其中一个可行的拉丁方如下：

通常情况下，使用拉丁方技术是为了减少各试验组先后安排的不同对其结果的影响，它是一种增强试验平衡性的技术。在试验设计时，通常会使用利用该方法生成的方格来辅助设计。拉丁方格是一个正方形矩阵，它内部元素是由待排序的几个设计变量定下的。举一个简单的粒子，现在有行和列两个变量，处理数是 6（为基数时方格设计有所不同），则其中一个可行的拉丁方如下：

A	B	F	C	E	D
B	C	A	D	F	E
C	D	B	E	A	F
D	E	C	F	B	A
E	F	D	A	C	B
F	A	E	B	D	C

在将该技术应用于 PSO 算法时，本文所采用的具体操作分为以下几步：

第 1 步：构造位置选择空间。将搜索空间的各个维度的区间进行等距划分，每个维度被划分为均匀的 n（n 为粒子数）段（子区间）。

第 2 步：构造速度选择空间。将之前设定的各维上的速度区间进行等距划分，每个维度同样被划分为均匀的 n 段。

第 3 步：初始化粒子位置。从位置选择空间的第 j（j = 1, 2, …, m，m 为搜索空间维度）维中选择一个子区间，并在这个子区间内随机初始化第 i（i = 1, 2, …, n）个粒子的第 j 维的位置。随后，从位置选择空间中删去这个子空间。

第 4 步：初始化粒子速度。从速度选择空间的第 j（j = 1, 2, …, m，m 为搜索空间维度）维中选择一个子区间，并在这个子区间内随机初始化第 i（i = 1, 2, …, n）个粒子的第 j 维的速度。随后，从速度选择空间中删去这个子空间。

(2) 惯性权重优化——时间与适应度结合的自适应惯性权重

标准粒子群优化算法的惯性权重自适应策略采用的是简单的随时间而线性减少的方法。采取这种策略的算法在搜索后期会失去探索新区域的能力，这是它的惯性权重（收缩因子）在后期会变得过于小的缘故。本文所采用的方法是，将时间和粒子适应度相结合来调节惯性权重，它的基本表达式如下：

$$w=\begin{cases} w_{min}-\dfrac{(w_{max}-w_{min})(f-f_{min})}{(\overline{f}-f_{min})}+n^2\dfrac{(w_{max}-w_{min})}{n_{max}^2}, & f\le\overline{f} \\ w_{max}, & f>\overline{f} \end{cases} \tag{3}$$

其中，w为单个个体的自适应权重，f为单个个体的适应度，$\bar{f}$为群体的平均适应度，n为迭代步数，n_{max}为最大迭代数。当群体中各个体的适应度差异较大时，惯性权重（收缩因子）会自行调整为一个较小的值，从而对个体朝最优解的位置处靠近有利；反之，当群体中各个体的适应度趋向于同一值时，w 则会变得比较大，从而避免陷入局部搜索寻优的情况。这种结合了时间上的二次方递减和粒子适应度来进行自适应调节的方法有两个方面的好处：1、有利于加快收敛的速度；2、有利于避免早熟收敛。

采用了上述两个策略对 SPSO 进行改进后，我们得到了改进的 PSO 算法（Improved Particle Swarm Optimization, IPSO）。其基本流程如下所示：

第 1 步：设置群体的规模大小，并采用拉丁方设计的试验设计方法来对粒子的速度和位置进行初始化。

第 2 步：计算群体中各粒子的适应度。设置两个临时变量（组），其中一个为变量组 P_{best}，它存放每个粒子历史上达到的最优解的适应度及其位置；另一个为变量 G_{best}，它存放整个群体全部粒子曾经达到的最优解的适应度及其位置。

第 3 步：按照自适应惯性权重公式(3)计算惯性权重（收缩因子）。

第 4 步：按照方程(1)和(2)更新每个粒子的速度和位置。

第 5 步：重新计算群体中各个粒子的适应度，并且挨个比较各个粒子现在的适应度和其之前达到的最优解 $P_{best,i}$ 中存放的适应度。如果现在的解更加优秀，则更新该粒子的 $P_{best,i}$ 的适应度和位置。

第 6 步：将各个粒子现在的适应度和整个群体全部粒子历史上达到的最优解 G_{best} 中存放的适应度值作比较。假如某粒子的解更加优秀，则更新 G_{best} 的适应度和位置。

第 7 步：按照自适应惯性权重公式(3)计算惯性权重（收缩因子）。

第 8 步：判断当前得到的结果是否满足收敛停止条件或者当前寻优步数是否已达最大设定步数，若满足了其中一个条件，则停止整个寻优过程，输出最优解的适应度及位置；否则，返回第 4 步。

本文介绍的 IPSO 算法使用 Matlab 和 C++语言进行开发。

2.4IPSO 优化算法效果验证

为了验证当前开发的 IPSO 算法效果，本文选用了两个测试函数分别使用 IPSO 算法和 GA 算法来测试并进行比对。两个测试函数的公式如下，其中 D 代表维度数：

$$f_1(x) = \sum_{i=1}^{D-1}\left(100\left(x_{i+1} - x_i^2\right)^2 + (x_i - 1)^2\right) \qquad (4)$$

$$f_2(x) = \sum_{i=1}^{D}\left(x_i^2 - 10\cos(2\pi x_i) + 10\right) \qquad (5)$$

测试函数 f_1 为 Rosenbrock 函数，它是一个单峰函数，但是此函数的各个变量之间的耦合性非常强。它的全局极小值为 $f(x^*)=0$，在 $x^*=(1,1,\cdots,1)$处。测试函数 f_2 为 Rastrigrin 函数，它是一个多峰函数。它的全局极小值为 $f(x^*)=0$，在 $x^*=(0,0,\cdots,0)$处。当 D=2 时，这两个函数的图像如下图所示：

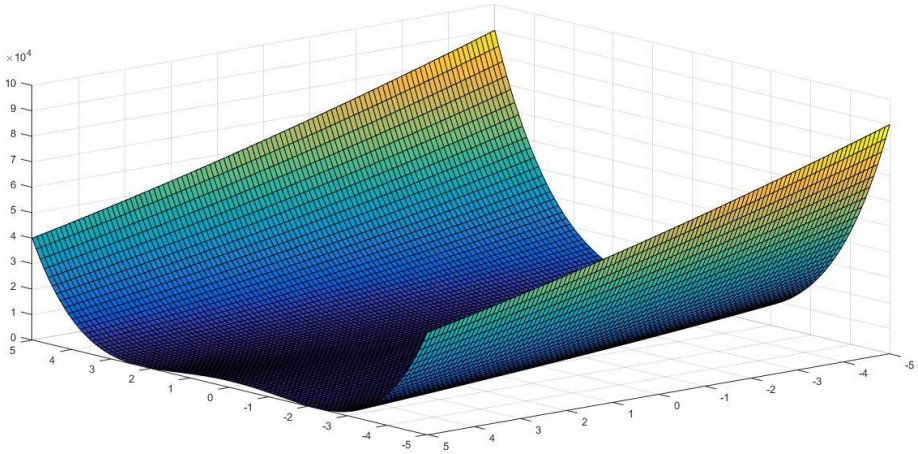

Fig.3 Rosenbrock 函数

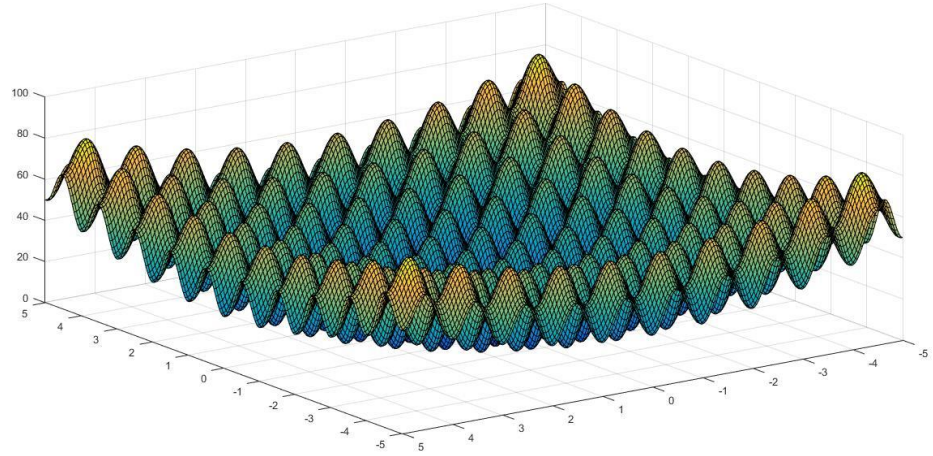

Fig.4 Rastrigrin **函数**

测试过程中，IPSO 算法和 GA 算法所选取的参数见表 1。其中，他们的群体粒子数均为 20，粒子的维数均为 10，每个函数运行 10 次，优化结果取其平均值。

测试结果见表 2。从表 2 可以看出，对于两个测试函数来说，IPSO 算法中对粒子在各维上设定的最大速度越小，算法收敛越慢，但是得到的结果越好；反之，若最大速度越大，则收敛更快，但得到的结果则会较差。同时，在与 GA 算法的比较中，可以发现，GA 算法虽然收敛快，但是有着严重的早熟收敛问题，尤其是优化第一个测试函数得到的结果。

表 1 算法参数

IPSO 算法参数			
学习因子 c_1	2.0	学习因子 c_2	2.0
惯性权重 w_{max}	0.9	惯性权重 w_{min}	0.3
群体粒子数 M	20	粒子维度 D	10
最大速度 $v_{max,1}$		$\frac{b-a}{2(M+1)}$	
最大速度 $v_{max,2}$		$\frac{b-a}{(M+1)^{1.5}}$	
最大速度 $v_{max,3}$		$\frac{b-a}{(M+1)^2}$	
注：其中 a 代表每个维度的搜索下界，b 代表每个维度的搜索上界			
GA 算法参数			
交叉概率	0.8	变异概率	0.2
群体粒子数 M	20	粒子维度 D	10

表 2 测试函数的平均适应值（最优值）

测试函数	下限 a	上限 b	IPSO			GA
			$V_{max,1}$	$V_{max,2}$	$V_{max,3}$	
Rosenbrock	-5	5	10.7027	9.8820	4.0418	6163.96
Rastrigrin	-5	5	46.3288	27.5575	11.9946	34.797

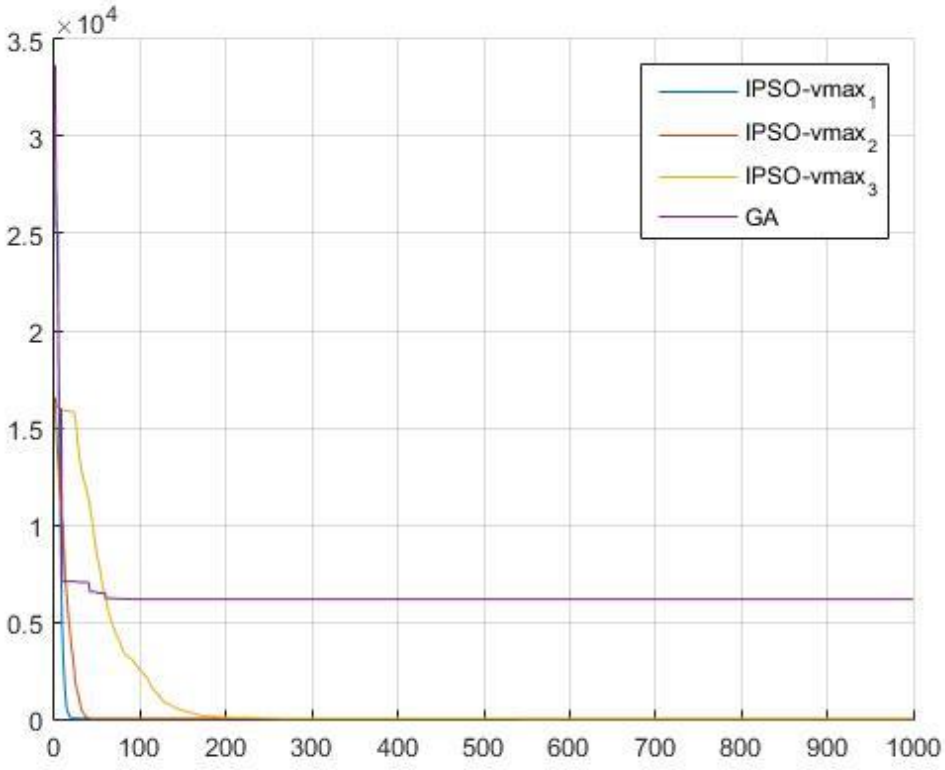

Fig.5 Rosenbrock 函数寻优曲线

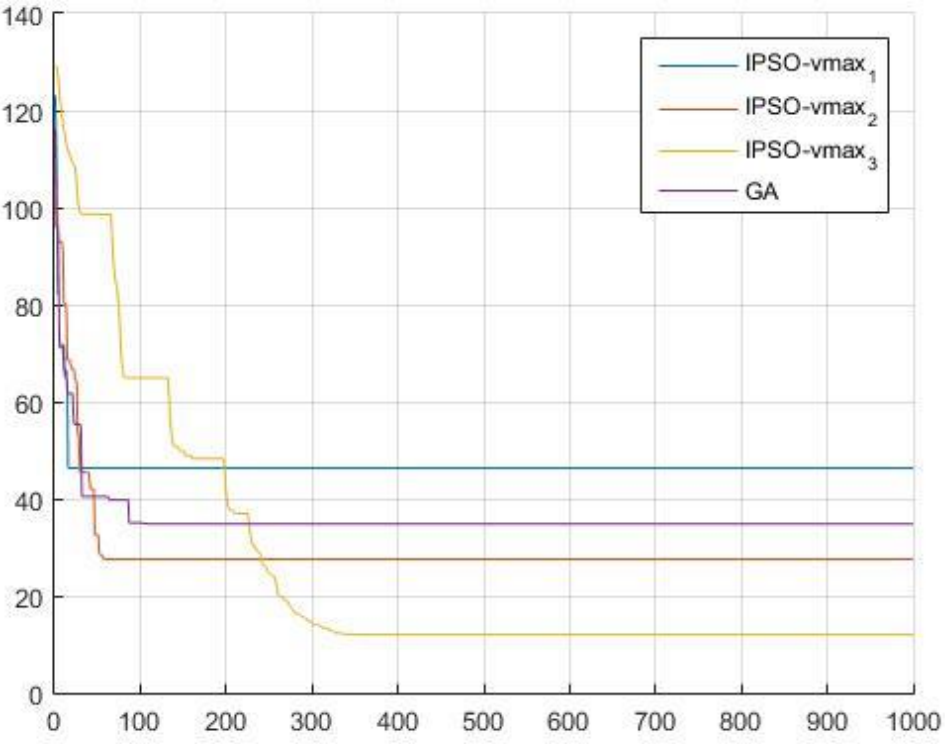

Fig.6 Rastrigrin 函数寻优曲线

另外，为了考察算法在粒子数较少的情况下的优化性能，设置种群规模 M=10，并且调小了最大迭代步数到 50，其他参数不变的情况下，重复进行试验。

优化结果见表 3。从表 3 中可见，当 IPSO 算法的最大速度过小时，它的收敛速度会很慢，在给定的最大迭代步数内，得到的结果甚至比不上更大的最大速度。另外，GA 算法发挥很不理想的一个原因也可以看出来，它很依赖种群规模。

表 3 测试函数的平均适应值（最优值）

测试函数	下限 a	上限 b	IPSO			GA
			$V_{max,1}$	$V_{max,2}$	$V_{max,3}$	
Rosenbrock	-5	5	28.7844	13.3824	82.2453	22103.4
Rastrigrin	-5	5	46.2381	41.7310	60.0073	76.2079

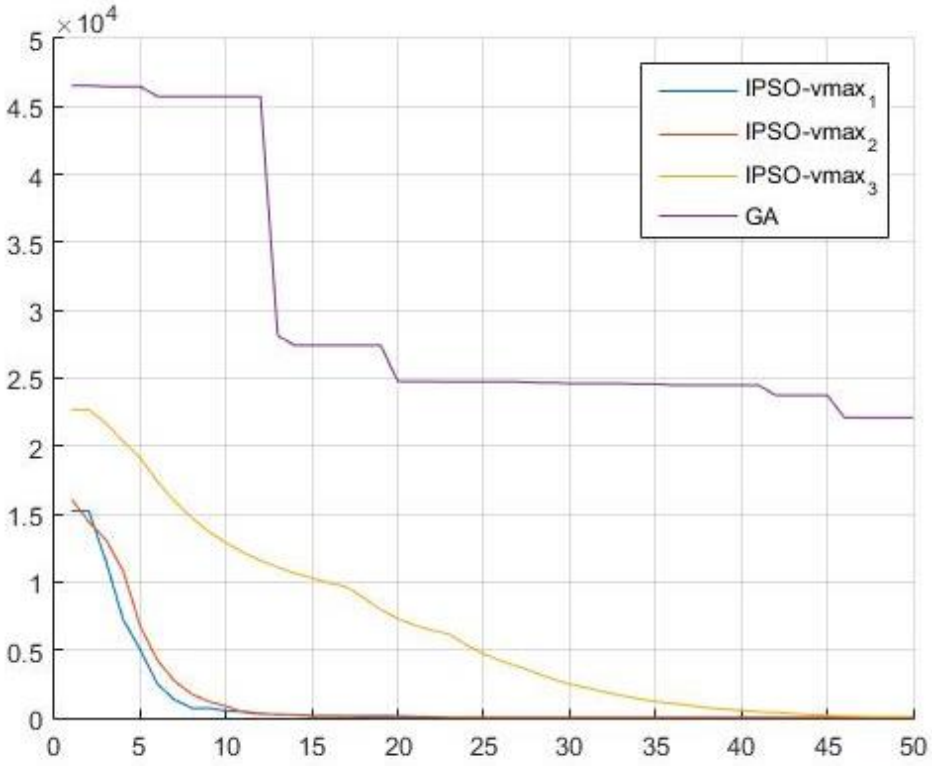

Fig.7 Rosenbrock 函数寻优曲线

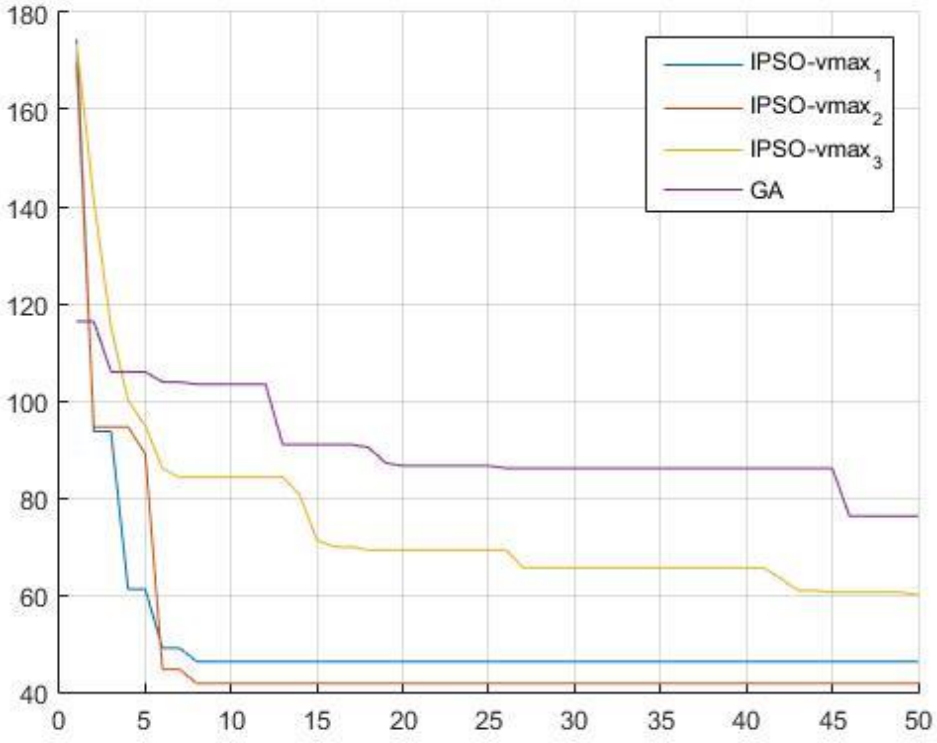

Fig.8 Rastrigrin **函数寻优曲线**

2.5 多目标粒子群算法开发

前文详细叙述了单目标优化问题的一种解决方法——粒子群优化。但是，现实生活中，尤其是实际的工程问题中，显现出来的往往都是多准则或者多目标下的优化问题，并且这些准则或目标之间很多是相悖的。世界上已有很多研究人员在从事解决这类多目标优化问题的研究，目的都是为了找出一组最优解，以满足不同情况下的需求。

（1）多目标优化问题的数学表述

现在考察一个通常的多目标优化问题（此处为最小化），它包含 M 个目标。它被组织成为数学方式的表述，见下：

$$\begin{cases} Minimize: & f(x) = [f_i(x), i = 1, \dots, M] \\ Subject\ to\ the\ constraints: & g_j(x) \geq 0, j = 1,2, \dots, J, \\ & h_k(x) = 0, k = 1,2, \dots, K, \end{cases} \quad (6)$$

此处的$x = [x_1, x_2, \dots, x_n]$，它表示此问题的一个候选解。n 表示的是决策空间的维度。$f_i(x)$是第 i 个目标函数，$g_j(x)$表示第 j 个不等式约束，而 $h_k(x)$则表示第 k 个等式约束。因此，上面的框架所描述的多目标优化问题可以被缩减为：寻找一个 x，从而使得 f(x)得到优化。

由于多目标优化问题的概念并不等同于单目标优化问题，因此我们需要一个不同于之前简单的适应度全序排列的解的评价方法。

(2) Pareto 非劣解[13]

由 Vilfredo Pareto 提出的 Pareto 优化概念，提供了一种被称为 Pareto 支配的方法来对解进行评价。Pareto 支配关系的定义可以被表示为：

考虑最小化优化问题，即 $\min = f_i(X), i = 1, 2, \dots, n$，现任意给出两个设计变量 X_u, X_v：

当且仅当对于任意$i \in \{1,\dots,n\}$，都有 $f_i(X_u) < f_i(X_v)$，则 X_u 支配 X_v；

当且仅当对于任意$i \in \{1,\dots,n\}$，都有 $f_i(X_u) \le f_i(X_v)$，且至少存在一个$j \in \{1,\dots,n\}$，使得 $f_i(X_u) < f_i(X_v)$，则 X_u 弱支配 X_v；

当且仅当存在$i \in \{1,\dots,n\}$，使得 $f_i(X_u) < f_i(X_v)$，同时，存在$j \in \{1,\dots,n\}$，使得 $f_i(X_u) > f_i(X_v)$，则 X_u 与 X_v 互不支配。

上述最小化优化问题可以扩展到最大化优化问题，甚至是最小化和最大化交织起来的优化问题。改写上述规则的关键是抓准“优”、“劣”的定义。

根据上述概念，我们可以得到一系列偏序关系。然而，我们很难找到一个能够支配其他所有解的最优解。我们得到的通常是这样的一些解：它对某一个或某几个目标函数来说是最优解，但是对其他的目标函数来说并不是最优的，有时甚至是最差的解。因此，在解决多目标优化问题时，我们得到的结果通常是一个解集，这个解集里的解之间的关系是互不支配的，即无法比较优或者劣的。它的特点是法同时达到这两个目标：1、提升任意一个目标函数的适应度；2、在满足第一个目标的情况下，不削弱至少一个其他目标函数的适应度。满足上述条件的解被称为非支配解（Non-dominated solutions）或 Pareto 最优解（Pareto optimal solutions）。对于以最小化为目标的多目标优化问题，假如 X_u 为 Pareto 非劣解，则它需要满足以下条件：

当且仅当，不存在这样的 $X_v \in U$，$f_i(X_v)$支配 $f_i(X_u)$，即不存在这样的 $X_v \in U$ 使得该式成立：任意 $i \in \{1,\dots,n\}, f_i(X_v) \le f_i(X_u)$ 且 存在 $i \in \{1,\dots,n\} \mid f_i(X_v) < f_i(X_u)$

Pareto 非劣解的定义直接表明：在得到的可行解集中，Pareto 非劣解是最优秀的解。所有 Pareto 非劣解对应的目标函数值所构成的一条曲线（当目标空间维度大于 2 时，为一个高维面）——它被称为 Pareto 前沿。

Pareto 解集形态通常为凸集或者凹集，当待搜索函数（数据）较为复杂时，其形态还可能呈现半凸或者半凹甚至是不连续的情况。

(3) 多目标粒子群算法

基本的 PSO 算法是一种单目标算法，为了用它来解决多目标优化问题，必须将其拓展为多目标 PSO 算法。

Sierra 等[14]于 2006 年对这个问题进行了全面总结。他们提出了在将单目标粒子群算法拓展成为多目标算法时需要解决的三个问题：1、如何构建非支配解；2、进化过程中如何选取最优粒子；3、如何保持 Pareto 前沿上解的均匀性和多样性。

Coello 和 Lechuga[15]于 2002 年提出了一种多目标粒子群优化算法。该算法的核心不同点是将适应度空间分割成若干个超立方体（Super Cube）。在搜索过程中，根据各个超立方体中所包含的非支配解的个数来更新外部归档。由于该算法具有需要划分超立方体这一特点，它被称为 CMOPSO（Cloud Multi-Objective Particle Swarm Optimization）。该算法利用外部档案和自适应网格这两大手段很好地避免其他多目标 PSO 算法容易陷入局部寻优的问题。

CMOPSO 算法运行过程中，得到的非支配解将会保存到外部归档中，并按下列规则进行更新：依次评估每一个新得到的非支配解，假如它受到外部档案中的任一粒子支配，则舍弃这个新解；假如它反过来支配档案中的某个或某几个粒子，则将这些被支配的粒子移除外部档案，并将这个新解加入外部归档中；假如它和档案中的每个粒子都互不支配，则直接将其加入外部归档中；假如外部档案中的粒子数大于之前设定的某个值 N，那么需要根据一定的策略选择性地保留其中 N 个粒子继续存放在归档中（通常是使用随机方法进行选择）。CMOPSO 的自适应网格技术是用来为每个粒子选取全局最优解所引入的辅助技术。在优化迭代过程中，会按某种策略将各个粒子划分入不同的超立方体中，并为处于不同超立方体中的粒子分配伪全局最优解（因为这个解对于不同的超立方体是不同的，通常是在属于某个超立方体的所有非劣解中随机选出）。CMOPSO 的 P_{best} 则和其他多目标粒子群算法相似，都是

采用了为每个粒子分配一个独立变量，在迭代过程中根据支配关系进行更新，一般规则如下：对于粒子 i，假如新得到的解支配 $P_{i,best}$，那么更新 $P_{i,best}$ 为新解；假如 $P_{i,best}$ 反过来支配新得到的解，那么不作任何变化；除开上述两种情况，即它们互不支配或者弱支配，则按一定概率更新 $P_{i,best}$ 为新解。

然而 CMOPSO 搜索算法使用的是固定的惯性权重，且 w 通常设置在 0.4 附近，搜索时很可能会陷入局部寻优。另一方面，该算法的两个学习因子 c_1、c_2 反而又不是按通常的 2.0 的固定设置，而是在 0.1~0.9 这个范围内，且需要根据不同的待搜索目标函数组合来选取不同的学习因子，因此很难使用。最后，该算法编写复杂，难以实现，容易出错。

为了解决 CMOPSO 的局部搜索问题，高志强、刘丽霞等[16]于 2015 年在其论文中提出嘞一种基于 Pareto 云隶属度的 MOPSO 算法。该算法利用 Logistic 映射产生混沌序列初始化种群，并借鉴布谷鸟搜索（cuckoo search, CS）的优势，增强种群全局多样性[16]。

夏立荣、李润学等[17]也于 2015 年对其他多目标搜索算法的研究作了简单评价。他们批评其他研究者提出的方法对于全局最优粒子的选择都具有随机性，缺少客观的认识。他们为了强调客观性在搜索过程中选取全局最优粒子的重要性，提出了一种基于动态层次分析的自适应多目标 PSO 算法。其中的关键点在于“动态层次分析”与“自适应”。

“动态层次分析”是这样一种方法——它基于模糊一致判断矩阵来评估粒子的优劣。这种方法的关键在于建立实用的一致判断矩阵。他们经过研究，抛弃了难以使用的 9 标度方法，转而根据另一文献使用方便简单的 3 标度方法，从而较好地保证一致判断矩阵的可综合性和一致性。在“自适应”方面，他们采用的方法是将惯性权重和两个学习因子和粒子间的距离进行关联，从而达到全局与局部寻优的平衡。其基本计算步骤如下：

1) 计算群体中第 i(i = 1, 2, …, N)个粒子与其他粒子的平均欧氏距离，即：

$$d_i = \frac{1}{N-1}\sum_{j=1,j\neq i}^{N}\sqrt{\sum_{k=1}^{D}\left(x_i^k - x_j^k\right)^2} \qquad (7)$$

2) 按照上式计算出的全局最优粒子的距离定义为 d_g，d_{max} 为其中最大的距离，d_{min} 则为其中最小的距离。此后，计算进化因子 f：

$$f = \frac{d_g - d_{min}}{d_{max} - d_{min}} \qquad (8)$$

3) 计算惯性权重系数以及两个学习因子：

$$w = \frac{1}{1+1.5e^{-2.6f}} \qquad (9)$$

$$C_1 = \frac{1}{0.2+0.2e^{2.2f}} \qquad (10)$$

$$C_2 = \frac{1}{0.1+0.2e^{-1.85f}} \qquad (11)$$

当 C_1 与 C_2 的和比 4 还大时，必须标准化他们的值，即：

$$C_i = 4C_i/\sum_{i=1}^{2} C_i \qquad (12)$$

然而经过实际编写程序并进行测试，上述两个方法都并不理想。此外，也有研究人员通过仅改变惯性权重的自适应方式来改进算法的性能。许艺、吴利平等 2015 年提出了将线性递减的自适应方法改变为正切函数递减的策略。他们认为，在搜索前期，应使用较大的惯性权重来保证避免陷入局部寻优，并维持这一较大的值较长时间，在中期又类似阶跃性地突然减小为一个较小的值，而后维持这一较小的值并让它缓慢减小以加强后期局部寻优能力。这本质上仍然是一种基于时间的自适应策略，并不考虑具体的粒子寻优及分布情况。他们提出的自适应惯性权重表达式为：

$$w = (w_{max} + w_{min})/2 - [(w_{max} - w_{min})/2arctan(A)]arctan(2A_t/T_{max} - A) \qquad (13)$$

然而经过测试，这样的“改进”得到的效果并不理想，其性能甚至不如线性迭代。

在查阅相关资料并经过仔细试验后，确定了下面的方案：

1) P_{best} 选取。为每个粒子保存一个个体最优解，其更新方案依据 Pareto 支配关系，与 CMOPSO 算法所采用的方案相同。
2) G_{best} 选取。根据拥挤度排序来选择全局最优粒子，拥挤度的计算步骤及全局最优粒子的选取如下：
   a) 设置一临时变量组，它保存当前迭代步数的外部归档中各个粒子的拥挤度。将所有拥挤度都初始化为 0。解集的大小为 n，目标数为 m，设置当前操作的最外层循环步数 i=1（最大迭代 n 次）。
   b) 对第 j（j=1,3,…,n）个目标进行排列，并重排粒子的位置。
   c) 按照下式计算第 k（k=2,3,…,n-1）个粒子的拥挤度距离，其中f^j为重排后的第 j 个目标函数值：

$$dist(k) = dist(k) + \frac{f_{k+1}^j - f_{k-1}^j}{max(f^j) - min(f^j)} \qquad (14)$$

   d) 判断 k 是否大于 n-1，若否，则跳回第 c 步；若是，则跳到第 e 步。
   e) 判断 j 是否大于 n，若否，则跳回第 b 步；若是，则跳到第 f 步。
   f) 判断 i 是否大于 n，若否，则跳回第 b 步；若是，则设置第 1 个和第 n 个粒子的拥挤度距离为无穷大并停止整个循环。
   g) 查找第 2 个到第 n-1 个粒子中拥挤度距离最大的粒子，将其作为全局最优粒子。
3) 速度及位置的更新。在基本速度和位置更新公式的基础上，舍去第一维的更新，即只更新剩下的第 2 到第 D 维的速度和位置，并判断速度和位置是否越界，公式如下：

$$v_{i,2:D} = w * v_{i,2:D} + c_1 * rand * \left(P_{best,i,2:D} - x_{i,2:D}\right) + c_2 * rand * \left(G_{best} - x_{i,2:D}\right) \qquad (15)$$

$$x_{i,2:D} = x_{i,2:D} + v_{i,2:D} \qquad (16)$$

   通过这样的操作，能够保留一定的随机性防止早熟现象的发生。经过测试，取得了很好的效果。
4) 惯性权重及学习因子的设定。由前述可知，过于复杂的自适应策略反而起不到良好的优化效果，结果是要么对参数过于敏感，要么就是性能更差。因此，此处采用了基于时间的线性递减自适应调整惯性权重的策略，基本公式为：

$$w = w_{max} - (w_{max} - w_{min}) \times t / t_{max} \qquad (17)$$

   其中 t 代表迭代步数。另外，两个学习因子都设置为固定值 2，这样可以增大算法使用的简便性，并且性能仍能保持一个好的水平。

多目标粒子群优化算法的整个流程步骤如下：

第 1 步：设定种群规模，设置惯性权重的最大值为 0.9，最小值为 0.4，设置两个学习因子为固定值 2，设置每个维度的最大速度为该维可行范围的 25%。

第 2 步：采用拉丁方设计方法初始化种群的位置，速度初始化为 0。

第 3 步：初始化当前非支配解集，并使外部归档保存这个解集，得到全局最优粒子 G_{best}。

第 4 步：自适应地调整惯性权重，更新每个粒子的速度和位置，并按照前述策略更新个体最优解 P_{best}。

第 5 步：得到当前迭代步数整个群体的非支配解集，将其与外部归档合并为新的解集。然后，筛选出新解集中的非支配解集，假如这个解集的大小大于之前的设定值，则随机挑选最大值个非支配解存放入外部归档中，将其余解全部舍弃。

第 6 步：计算得到全局最优粒子 G_{best}。

第 7 步：判断是否达到最大步数，若是，则终止迭代输出结果；否则，返回第 4 步。

本文提出的改进的多目标粒子群优化算法（IMOPSO）主要改进了速度及位置更新公式。在更新速度及位置时，放弃了第一维的更新。这种增加随机性的策略规避了粒子群优化算法收敛过快的问题。同时引入了自适应惯性权重，算法在前期和后期的搜索质量都能得以保证。本文提出的 IMOPSO 算法采用 Matlab 语言编程开发。下面对其效果进行验证。

2.6 IMOPSO 优化算法效果验证

为了验证 IMOPSO 算法的效果，并与小组之前所使用的 Matlab 自带的多目标遗传算法（使用 gamultiobj 调用）性能作比较，首先介绍两个衡量性能的指标。

(1) 世代距离（generational distance, GD）[16]

衡量搜索算法得到的近似 Pareto 前沿与真实的 Pareto 前沿之间的欧氏距离，按照下式计算：

$$GD = \left(\frac{1}{n}\sum_{i=0}^{n} d_i^2\right)^{0.5} \qquad (18)$$

式中：n 表示近似 Pareto 前沿中粒子的个数；d_i 表示近似 Pareto 前沿中第 i 个粒子离真实 Pareto 前沿的最短欧氏距离。算法的收敛性与 GD 的大小成负相关。

(2) 均匀性指标（Spacing, SP）[16]

SP 的计算公式如下：

$$SP = \sqrt{\frac{\sum_{i=1}^{n}\left(d_i - \frac{\sum_{i=1}^{n-1} d_i}{n-1}\right)^2}{n-1}} \qquad (19)$$

式中各符号含义与计算 GD 的公式相同。算法得到的近似 Pareto 前沿的分布性与 SP 的大小成负相关，即，SP 越小，表示解的分布越均匀。

本文选择了 ZDT1、ZDT2、ZDT3、ZDT4 函数组来进行测试。试验的相关参数见下表所示。

表 4 算法参数

IPSO 算法参数			
学习因子 c_1	2.0	学习因子 c_2	2.0
惯性权重 w_{max}	0.9	惯性权重 w_{min}	0.4
群体粒子数 M	100	粒子维度 D	30
最大迭代数 N	100	最大速度 v_{max}	$0.25 \times (b-a)$
注：其中 a 代表每个维度的搜索下界，b 代表每个维度的搜索上界			
GA 算法参数			
交叉概率	0.8	变异概率	0.2
ParetoFraction	0.3	粒子维度 D	30
群体粒子数 M	100	最大迭代数 N	100
ZDT1、ZDT2、ZDT3、ZDT4			
各维下限 a	0	各维上限 b	1

ZDT1 和 ZDT2 的函数形式相近，他们的 Pareto 前沿形态也很相似，只不过一个为凸集、一个为凹集。

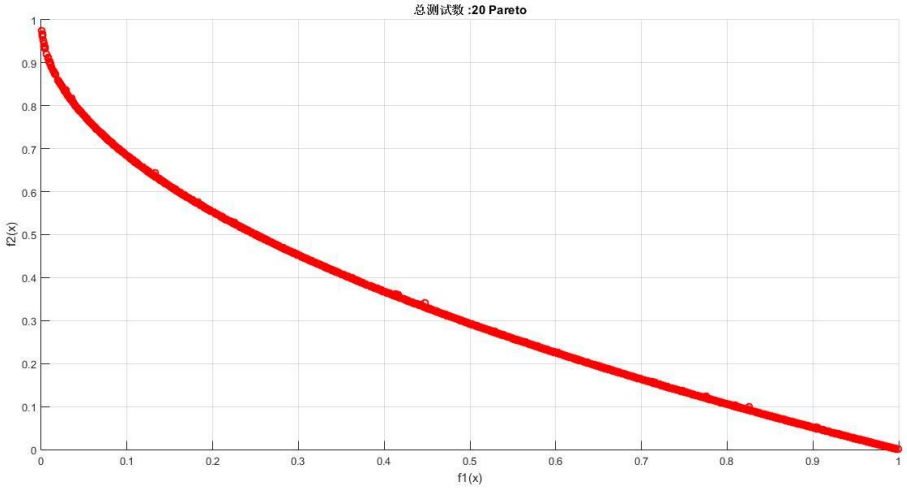

Fig.9 IMOPSO-ZDT1, GD = 2.456420278338573e-05, SP = 7.194518374624272e-04

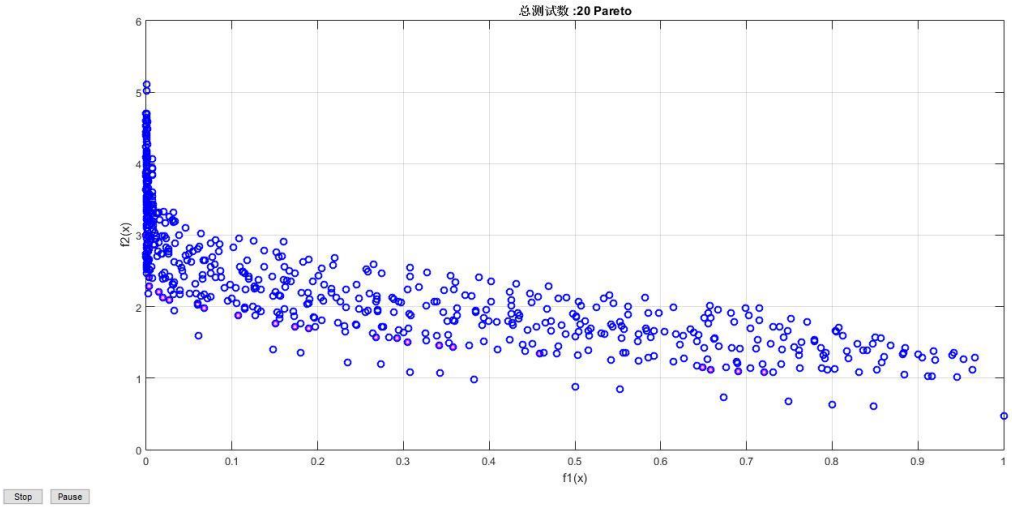

Fig.10 GAMO-ZDT1, GD = 0.069395372361884, SP = 0.021679000416276

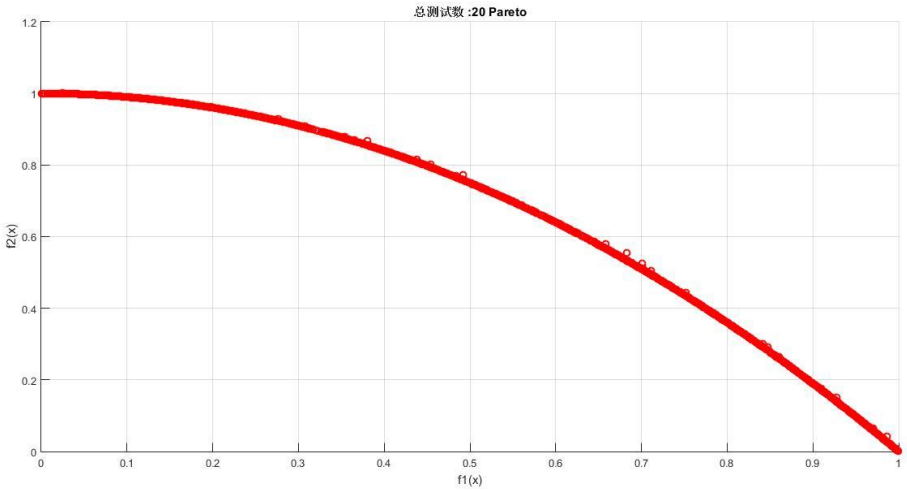

Fig.11 IMOPSO-ZDT2, GD = 0.005753442892032, SP = 9.385682874031506e-04

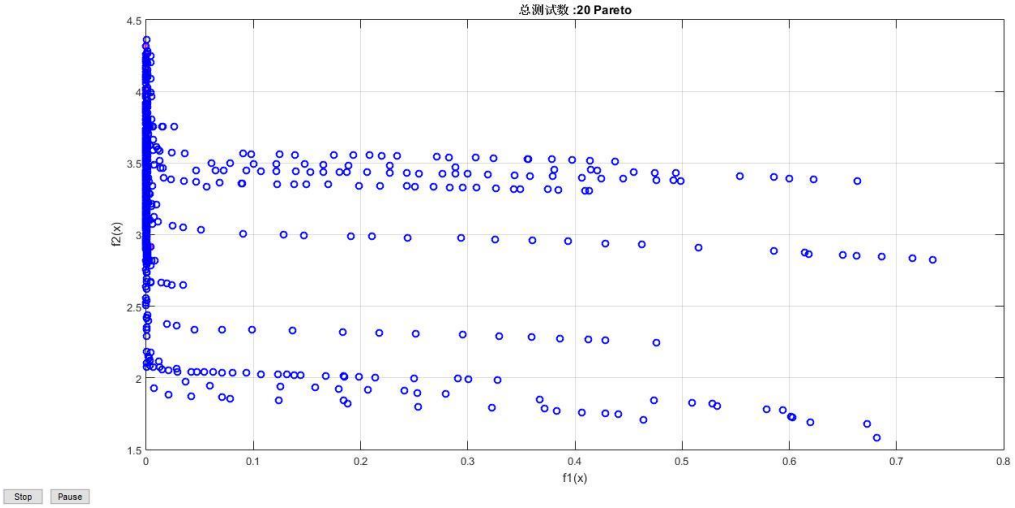

Fig.12 GAMO-ZDT2, GD = 0.092938645795985, SP = 0.013549539110927

从上面两组对比可以看出，IMOPSO 算法能够非常好地逼近真实的 Pareto 前沿，而 GAMO 则显得力不从心。Matlab 自带的多目标遗传算法在面对非多峰函数时，收敛性很差；从图中可看出，GAMO 在 f1=0 附近的 Pareto 解差异非常大，经分析，这是它容易陷入早熟收敛的缘故。而经过改进的 MOPSO 算法则没有这个问题。

ZDT3 的 Pareto 前沿是不连续的，这是它一个很显著的特点。

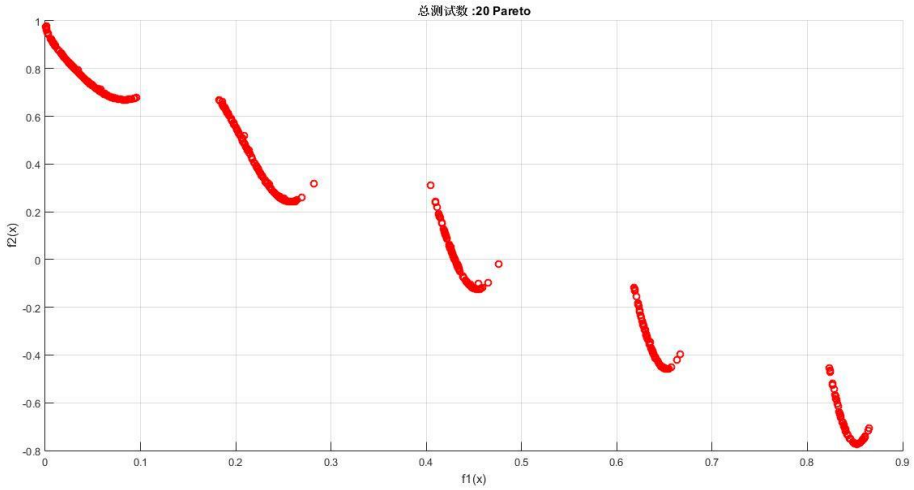

Fig.13 IMOPSO-ZDT3

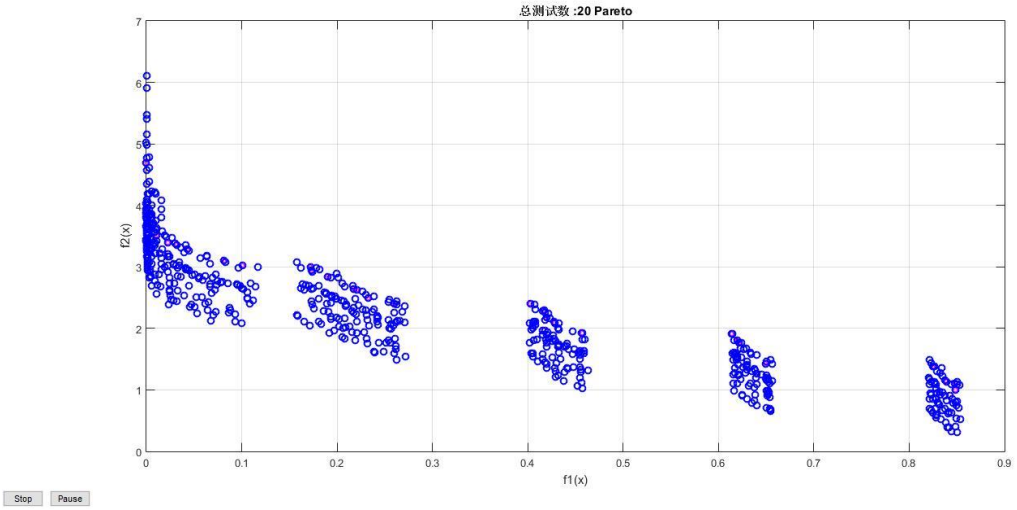

Fig.14 GAMO-ZDT3

从上面两幅图的对比可以看出，GAMO 和 IMOPSO 在面对非连续的 Pareto 前沿时，都能较好地还原其解集的形态，即非连续的形态。然而，GAMO 仍然面临着之前遇到的容易陷入早熟收敛的问题，其收敛性较 IMOPSO 差很多。

ZDT4 函数的形式比前三者都要复杂。

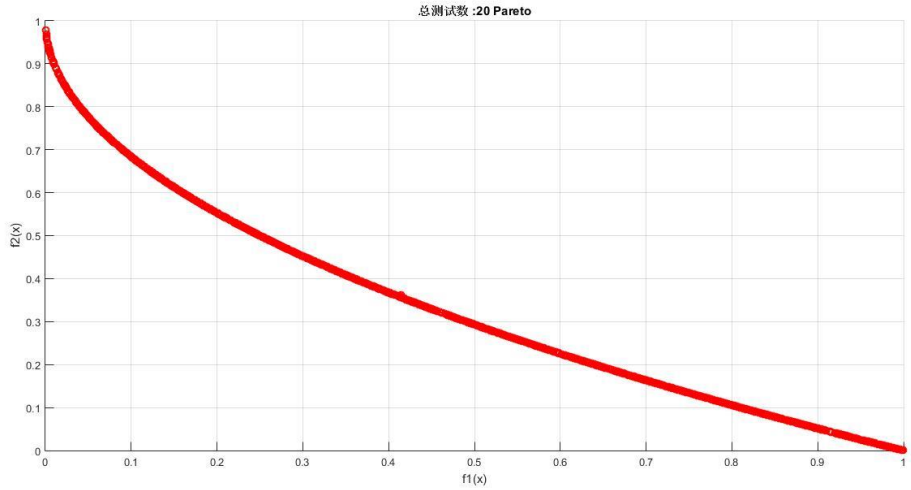

Fig.15 IMOPSO-ZDT4, GD = 2.607786416952833e-05, SP = 6.292697934065191e-04

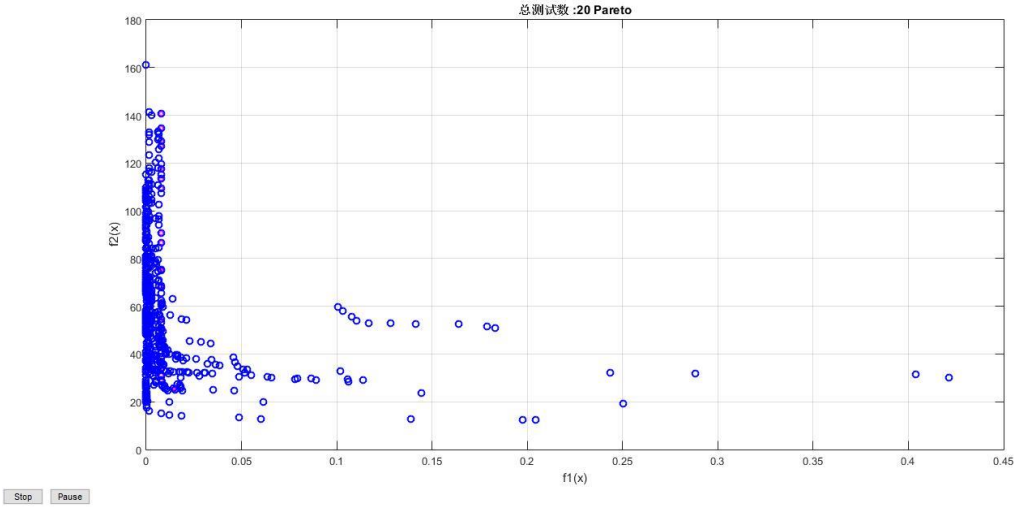

Fig.16 GAMO-ZDT4, GD = 2.757737163883514, SP = 0.851193724341380

在面对 ZDT4 时，GAMO 容易早熟收敛的缺陷暴露得更加明显；反观 IMOPSO，收敛性及分布性都非常好，而且平均性能好。

另外，本文另设了两组测试函数，第一组由之前在单目标测试中使用的 Rosenbrock 及 Rastrigrin 函数组成。设计这个测试的目的是考察 IMOPSO 在面对多峰函数时的表现。测试函数的设计变量维度为 10，每个维度的下限都为-2，上限都为 2。

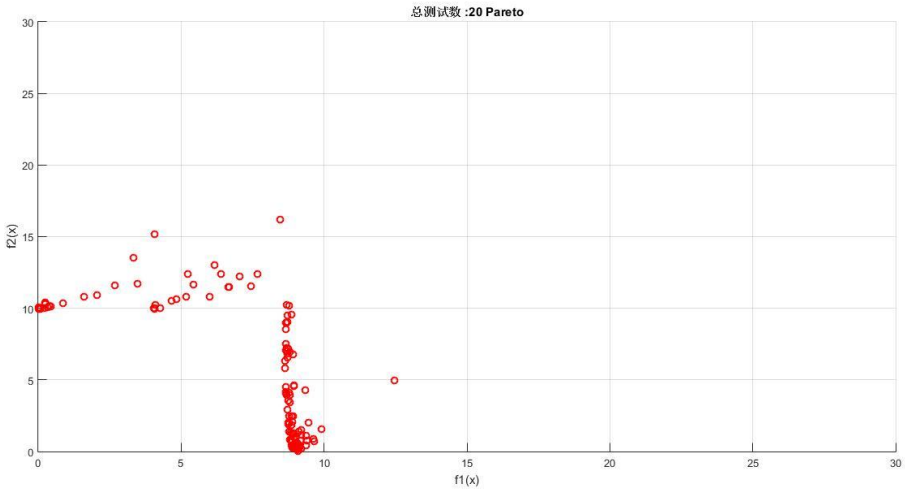

Fig.17 IMOPSO-Rosenbrock & Rastrigrin

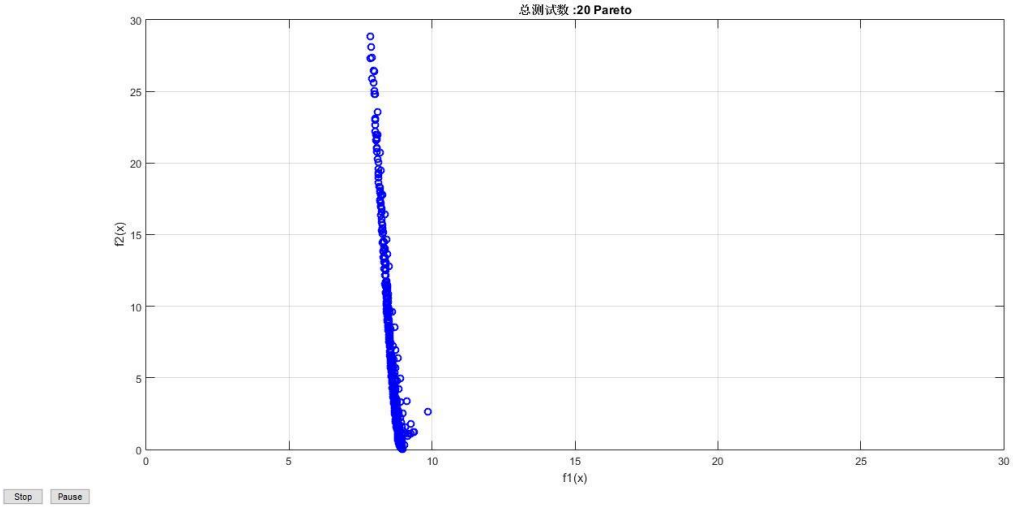

Fig.18 GAMO-Rosenbrock & Rastrigrin

可以看出，之前表现不佳的多目标遗传算法在面对多峰函数时，能够较接近且稳定地找到 Pareto 前沿。但是，相比于改进的多目标粒子群优化算法，GAMO 仍不能较好地靠近真实的 Pareto 前沿，这可从图像上直观地得出：GAMO 只得到了 f1=10 附近的 Pareto 前沿，却丢失了另一半。为了更突显二者的不同，现将第一个测试函数 Rosenbrock 的每个维度都向右平移 0.1 个单位，得到下面的结果。

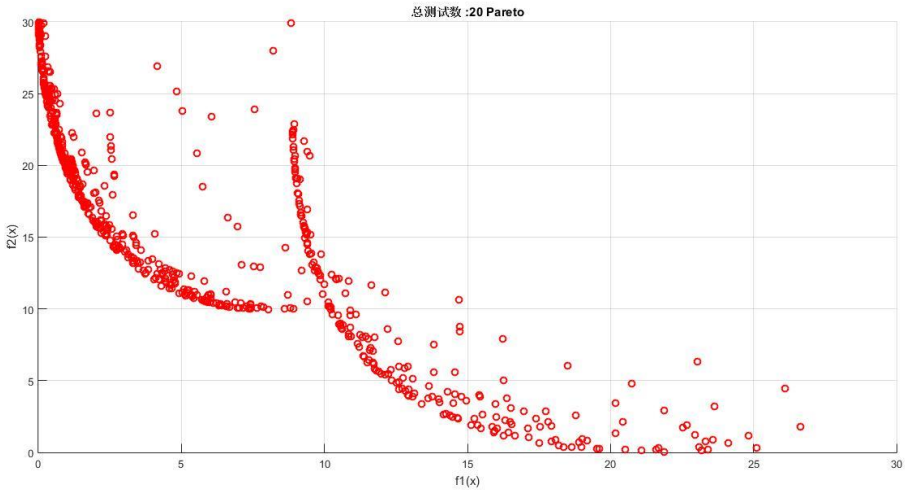

Fig.19 IMOPSO-Rosenbrock & Rastrigrin

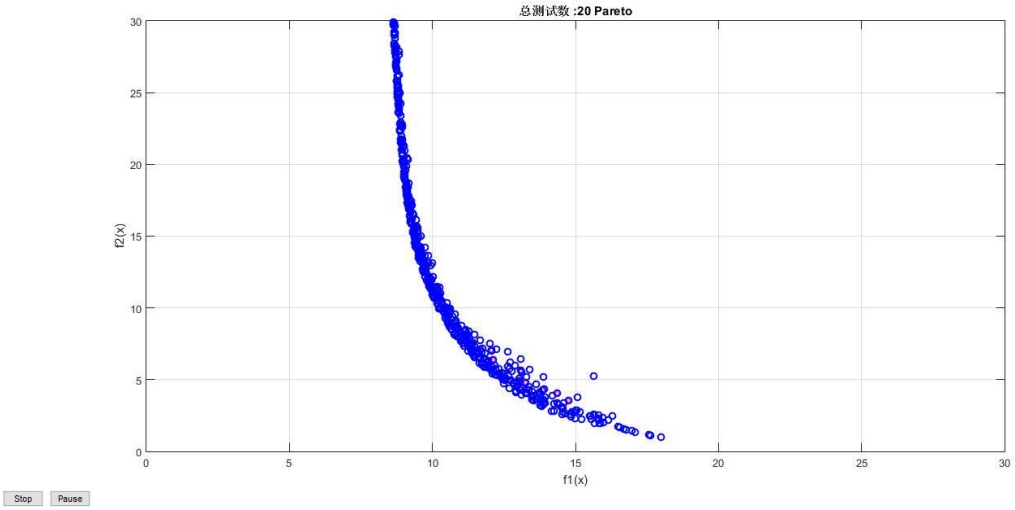

Fig.20 GAMO-Rosenbrock & Rastrigrin

第二组由 Rosenbrock 和 Griewank 函数组成。其中，Griewank 仍为一个多峰函数，并将第一个测试函数 Rosenbrock 的每个维度都向右平移 0.1 个单位。得到下面的结果。

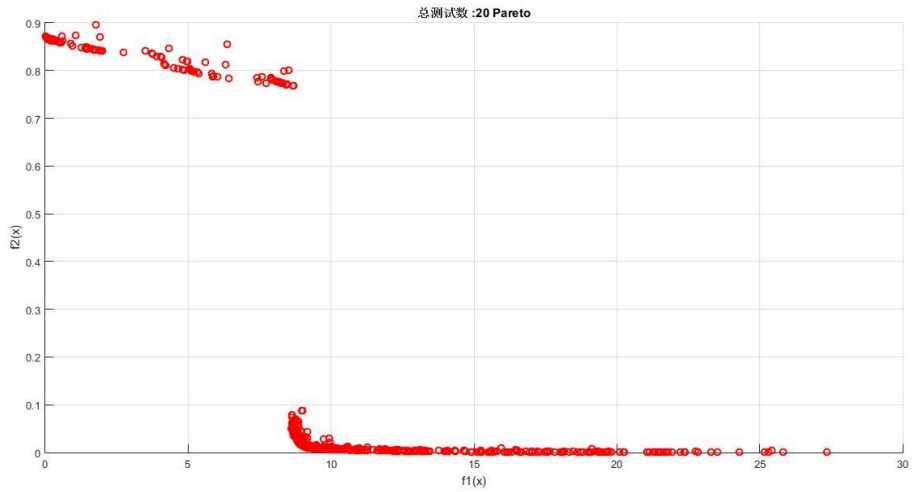

Fig.21 IMOPSO-Rosenbrock & Griewank

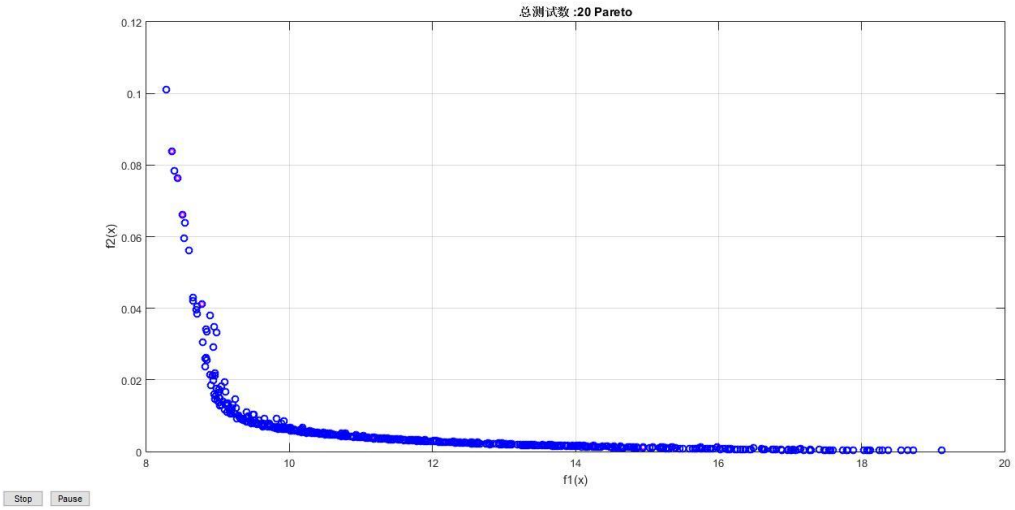

Fig.22 GAMO-Rosenbrock & Griewank

从平移后的结果可以更明显地看出，虽然 GAMO 在面对多峰函数时有较好地结果，且相当稳定，但是却丢失了一部分 Pareto 前沿；另一方面，IMOPSO 虽然在面对多峰函数时，稳定性稍稍弱于多目标遗传算法（可从图中直接看出），然而，它得到了 GAMO 丢失的另一半 Pareto 前沿。

结合前面几个有限的测例，可知：在面对非多峰函数时，GAMO 的收敛性较差，不能很好地逼近真实的 Pareto 前沿，而 IMOPSO 则能很好地逼近真实的前沿，并且分布性较好，算法的平均性能佳；在面对多峰函数时，GAMO 能够较好地逼近真实的 Pareto 前沿，并且整个群体距离这一前沿非常近，少有飞离这一前沿较远距离的点，IMOPSO 算法虽然找到了这一前沿，但飞离这一前沿的点较 GAMO 的更多。可知，GAMO 在面对多峰函数时，收敛性好于 IMOPSO，但是同样可以看出，GAMO 并没有近似地靠近整个真实的 Pareto 前沿，IMOPSO 则很好地搜寻到了这一近似前沿。

2.7 并行算法的开发

由于适应度的计算需要耗费大量的计算资源，时间成本很高，因此采用了建立近似模型的方法来减少资源的耗费并缩短时间。然而，经过测试研究发现，不同的近似模型在计算适应度时耗费的时间差异是巨大的。简单的模型只需要毫秒级的时间便能完成适应度的计算，然而由于船型优化近似模型的复杂性，这一时间耗费仍然相当高，每一次的计算甚至高达数百秒。因此，本文使用 C++语言开发了近似度计算的并行版本，加快了计算速度。

首先简要介绍与线程相关的基本概念。

线程（Thread）与进程（Process）是一对联系紧密的概念。线程的另一个称呼是 miniprocess，即轻量级进程。一个通常的 C++程序，被系统调用后，系统会维护一个进程控制块（Process Control Block, PCB），该进程有一个名为 main 的线程。一个进程通常含有这几部分：1、地址空间；2、全局变量；3、打开的文件；4、子进程；5、信号量和信号处理器等。与之相比，线程自己则不拥有系统资源。它是进程中的一个实体（如 main 线程），是系统独立调度和分派的基本单位。虽然线程只拥有很少的资源，但是它能够和父进程中的其他线程共享该进程的所有资源。线程的这一特性使得在线程间的切换开销很小，并且线程间的通信也比进程间通信（Inter-Process Communication）更为方便快速。

线程的实现方式有两类：1、用户级线程；2、内核级线程。本文所实现的线程为前一种。同时由于 PSO 算法内在的并行性，在开发并行算法时，并不需要处理复杂的同步、互斥及死锁等问题。但是由于几条线程操作的是同一个地址空间，因此必须注意内存方面的操作。另外建议在传递不会被改变的变量时，使用常量引用进行传递。

下面给出一个多线程代码的实例。

```
#include <thread>
void parallelFunc(…){
    vector<std::thread> threads;
    for(int i = 0; i < SIZE / MaxThreads; i++)
    {
        for(int j = 0; j < MaxThreads; j++)
            threads.push_back(std::thread(func, parameters));
        for(int j = 0; j < MaxThreads; j++)
            threads[j].join();
        for(int j = 0; j < MaxThreads; j++)
            threads.pop_back();
    }
```

```
        int t = SIZE % MaxThreads;
        for(int k = SIZE-t; k < SIZE; k++)
            threads.push_back(std::thread(func, parameters));
        for(int k = 0; k < t; k++)
            threads[k].join();
        for(int k = 0; k < t; k++)
            threads.pop_back();
        return;
    }
```

其中，SIZE 代表 PSO 算法中的粒子数，而 MaxThreads 则为之前定义的最大线程数。std::thread(func, parameters)表示生成一个线程，该线程使用名为 func 的函数来操作参数表 parameters 所代表的参数。.join()方法则让父进程在某一（几）条线程结束后才能继续运行。所有生成的线程均放在一个 vector 容器中保存。

实现适应度并行计算的另一方法是改写原先的（即 func 函数中）大型矩阵运算。该方法更偏向底层。理论分析，采用此种方法所占用的内存量会较少，但是本文所采取的方法更具有普遍性。

在使用本文所提供的方法改写其他适应度计算代码时，尤其需要注意文件的操作，否则有可能出现几条线程同时操作同一个文件的现象，而这一问题又难以被发现。

同时，考虑到可移植性，本文使用了最新的 C++11 标准，使用的编译器为 gcc5.3.0(版本号至少应为 4.8.0，否则编译无法通过)。下面是测试数据。

表 5 并行算法测试数据

操作系统		Linux openSUSE Leap
中央处理器		Intel Core i5-2450M @ 2.50GHz * 4
线程数	计算一次适应度平均耗时	时间减少百分比
1	2min58s	0
2	2min01s	-32.02%
3	1min29s	-50.00%

例如，当粒子数设置为 10，最大迭代 50 步时，使用单线程版本耗时约 25 小时，而使用三线程版本则能缩短一半时间到约 12.5 小时，大大提升了效率。

第三章 结论

本文首先详细叙述了基于 CFD 和近似模型的船型优化技术被提出的背景原因，并比较了该技术和传统船型设计的异同，阐述了该技术的优越性。而后，调查研究了该技术在国外的发展情况。本文介绍了该技术的整个流程，并就其中某些关键技术作了详细介绍。

本文详细介绍了最优化方法的相关理论，其中主要介绍了现今广泛使用的几种随机搜索算法，分别为：粒子群优化算法、差分进化算法和遗传算法。本文详细阐述了这三种方法的背后思想及各种改进策略。在细致地分析他们的优劣后，本文最终选择了粒子群算法作为船型优化过程中的最优化方法。这是因为，较之本研究小组之前使用的遗传算法及其多目标版本，在单目标寻优时，粒子群算法能够较快地收敛到一个可行解；另外，在多目标寻优时，该算法面对多峰和非多峰函数都能较快地接近真实的 Pareto 前沿，而本小组之前使用的多目标遗传算法在面对非多峰函数时较难靠近这一前沿。虽然它在面对多峰函数时能够收敛到一个近似 Pareto 前沿，但比较发现，该前沿与 IMOPSO 算法得到的前沿相比，它缺失了一半。因此，本文给出的 IMOPSO 算法针对各种不同的优化问题，具有更强地适应性。这一关键技术的改进，是基于 CFD 和近似模型的船型优化技术得以成功应用的重要保障。

最后，本文介绍了并行计算在优化过程中的应用。作者开发的并行计算程序能够有效地减少计算时间，这具有重要的意义。

参考文献

[1] 李胜忠. 基于 SBD 技术的船舶水动力构型优化设计研究[D]. 中国舰船研究院, 2012.
[2] 赵艳丽. 差分进化算法在图像处理中的应用研究[D]. 中国石油大学(华东), 2010.
[3] 沈通, 冯佰威, 刘祖源,等. 基于径向基函数插值的船体曲面修改方法研究[J]. 中国造船, 2013(4):45-54.
[4] Lackenby H. On the Systematic Geometrical Variation of Ship Forms[J]. NA-Transaction, 1950,92:289-315
[5] Ratnaweera A, Halgamuge S K, Watson H C. Self-organizing hierarchical particle swarm optimizer with time-varying acceleration coefficients[J]. IEEE Transactions on Evolutionary Computation, 2004, 8(3):240-255.
[6] 席裕庚, 柴天佑. 遗传算法综述[J]. 控制理论与应用, 1996(6):697-708.
[7] 薛明志, 左秀会, 钟伟才,等. 正交微粒群算法[J]. 系统仿真学报, 2005, 17(12):2908-2911.
[8] 刘衍民, 赵庆祯, 牛奔. 基于正交设计的多目标粒子群算法[J]. 计算机应用研究, 2011, 28(1):72-74.
[9] Campana E F, Fasano G, Pinto A. Dynamic system analysis and initial particles position in Particle Swarm Optimization[J]. Swarm Intelligence, 2006.
[10] Clerc M. Initialisations for particle swarm optimization. http://clerc.maurice.free.fr/pso/, 2008.
[11] Shi B Y, Eberhart R. A modified particle swarm optimizer, The 1998[C]// Evolutionary Computation Proceedings, 1998. IEEE World Congress on Computational Intelligence. The 1998 IEEE International Conference on. IEEE, 2010:69-73.
[12] 戴朝华. 搜寻者优化算法及其应用研究[D]. 西南交通大学, 2009.
[13] 苏为华. 多指标综合评价理论与方法问题研究[D]. 厦门大学, 2000.
[14] Reyes-Sierra M, Coello Coello C A. Multi-Objective Particle Swarm Optimizers: A Survey of the State-of-the-Art[J]. International Journal of Computational Intelligence Research, 2006, 2(3):287-308.
[15] Coello Coello C A, Lechuga M S. MOPSO: a proposal for multiple objective particle swarm optimization[C]// Evolutionary Computation, 2002. CEC '02. Proceedings of the 2002 Congress on. 2002:1051-1056.
[16] 高志强, 刘丽霞, 刘悦,等. 一种基于 Pareto 云隶属度的 MOPSO 算法[J]. 中国科技论文, 2015(14):1610-1613.
[17] 夏立荣, 李润学, 刘启玉,等. 基于动态层次分析的自适应多目标粒子群优化算法及其应用[J]. 控制与决策, 2015(2):215-221.

谢辞

毕业前最后的繁忙还差一个完满的句号。回首在上海交大生活学习的这四年时光——大一的青涩与自卑让我茫然不知所措，大二的低沉与放纵掩埋了曾经的抱负，爆发于大三而又渐归平静，终于在最后的时间收获了一个真实的自己。

认识自我是一个深刻的话题。

不远万里求学于此的我们有着不同的成长经历，也就有着各自的缺陷与美好。在这里的学习生活和思考，我对自己的生理、心理有了更深入的了解。接受了真实的自我，而不是否定压抑。不断改进自己，为的是解放自己。正如前不久去世的杨绛先生所说，“我们曾如此渴望命运的波澜，到最后才发现，人生最曼妙的风景，竟是内心的淡定与从容。”

首先要认识世界，才能认识自己。

大学并不是一般的职业技术学院。感谢学校提供了丰富的通识教育资源和自由的环境，曾经懵懂的我们对自我及这个世界有了更加清醒的认识。在此，我要感谢汪国琴老师，感谢您引导迷茫中的我走向光明；我要感谢李威仪老师，感谢您重新激活了我对未知的好奇；我要感谢国际经济与贸易的各门专业课的老师，感谢你们让我对这个世界的经济运作有了初步的认识……当然，我还要感谢学校自由的网络环境，感谢它让我能方便地接受外部世界的信息，了解时代的发展，并改进自己。

最后，感谢在我完成毕业论文期间帮助过我的吴建威师兄和缪爱情师姐。没有你们的提点，我无法对课题有深入的认识；还要感谢刘晓义师兄对本论文给出的详细、诚恳的建议；最后还要感谢万教授不时的关心，这才保证了课题完成的质量。

感谢这四年来知名与不知名的同学、朋友对我的帮助与关心。曲终人将散，曾经被命运召唤于此的我们终归还是要面对别离的这一刻，还愿大家能唱出了无遗憾的终章。

DEVELOPMENT OF SHIP TYPE OPTIMIZATION TECHNOLOGY BASED ON CFD AND APPROXIMATE MODEL

The optimal design of ship type requires the use of hydrodynamic performance tools to carry accurately forecast, so that the reliability can be guaranteed. In the past, when designers wanted to get a ship type that has excellent hydrodynamic performance, they had to firstly choose a parent ship type and modify it with their experience; then analyses the hydrodynamic performance of this new type with model tests or some tools such as CFD (Computational Fluid Dynamics) soft-wares. Nevertheless, that approach has some severe flaws, that is, a) designers' experience and parent ship types are strongly required. Without them, a good design is almost impossible; b) a lot of resources are consumed to carry out a model test; c) the design cycle is too long; d) It is extremely difficult to get the best ship type, and usually the obtained type can just meet the design specifications. Although it is feasible, it's hydrodynamic performance is not so good. But nowadays, computer technology has greatly changed the world. With its rapid development and the continuous improvement of optimization theory, the attention from the field of ship type design has been concentrated on its big future. The concept and connotation of ship type optimization technology based on CFD and approximate model are analyzed in detail. The key technologies, such as single-objective and multi-objective global optimization technology and parallel computing, have been focused on.

1. Content

The writer first describes the traditional approach to design a new ship type. Its ineluctable problems are then discussed. After that, the writer introduces a new technology, that is, SBD (Simulation Based Design). That technology combines traditional Numerical Evaluation Technique of Computational Fluid Dynamics and Approximate Model Technique. Next, the writer introduces the core techs of that technology and its development in foreign countries. The whole cycle of ship type optimization technology based on CFD and approximate model is also presented. Among all the core techs, optimization technology is selected to be studied. Then, different optimization approaches are introduced and discussed.

Mainly, three kinds of techs are talked about. They are Differential Evolution, Particle Swarm Optimization and Genetic Algorithm. First, they are briefly introduced and distinguished from these aspects: coding standard, parameter setting problem, multidimensional problem, convergence property and wide application. Then detailed information about the three different kinds of techs are introduced in the second chapter.

Optimization technique plays an important role in improving hydrodynamic performance of ship hull by changing its geometric shape. This is because that the influence of the change of the hull shape on the whole performance is difficult to be expressed by a simple function. Optimization

technology since as a separate discipline, has been widely used in the field of many engineering, machinery, shipbuilding, aviation, aerospace and other design shine. The theory that it relies on has developed a lot. However, due to various optimization algorithms have their own unique guiding ideology, they are most suitable for processing different problems. In addition, because of the optimization of ship hydrodynamic performance, it is usually necessary to optimize the multi-objective optimization, so it is necessary to develop the multi objective optimization algorithm. To solve this problem, the traditional approach is to give a certain weight to the target to be weighted to the single target problem, and then for the problem select a suitable single target search algorithm to solve the problem. However, this method has a serious flaw that we often do not know how to determine the weight. In addition, in the face of complex and non-convex problems, the traditional approach often cannot get the global optimal solution, and is easy to fall into the local optimal situation. Therefore, the second chapter will focus on the concept of Pareto non dominated solutions in the fifth section. And based on that, improved multi-objective particle swarm optimization algorithm is developed.

In the end of the second chapter, that is, the seventh section, the parallel version of improved particle swarm optimization is talked about. Because of the huge cost of computational resources, the time spent is very long. So the approximate model is used to reduce the cost and shorten the time. However, after the test, it is found that the time spent is tremendously different between the different approximate models. Simple model only need millisecond time to complete the adaptation degree calculation, however due to the complexity of the approximate model of ship optimization, its time cost is still relatively high. Every time the calculation even costs as high as several hundreds of seconds. Therefore, this thesis uses C++ language to develop a parallel version of the approximate calculation, in order to speed up the calculation speed.

First, a brief introduction of the basic concepts related to the thread is presented. Thread and the process is a pair of closely related concepts. Another name of thread is mini-process (or lightweight process). When a common C++ program is called by the system, the system will maintain a Process Control Block (PCB), and the process has a thread called main. A process usually contains several parts: 1, address space; 2, the global variable; 3, open the file; 4, the child process; 5, the amount of signal and signal processor, etc. In contrast, the thread itself does not own the system resources. It is an entity in a process (such as the main thread). Although threads only have very little resources, they can share all the resources of the process with the other threads in the parent process. This feature of the thread makes the switching overhead of the online process is very small, and the communication between the threads is more convenient and faster than the inter process communication.

2. Conclusions

This thesis describes the background of the ship type optimization technology based on CFD and approximate model, compares the similarities and differences between the technology and the traditional ship design, and expounds the advantages of the technology. Then, the development of the technology in foreign countries was investigated and studied. This thesis introduces the whole process of the technology, and some key technologies are introduced in detail. This thesis introduces the related theory of the optimization method in detail, and introduces several kinds of random search algorithms which are widely used in the present thesis, including: particle swarm

optimization algorithm, differential evolution algorithm and genetic algorithm. In this thesis, the ideas behind the three methods and the improvement strategies are described in detail. After careful analysis of their advantages and disadvantages, this thesis finally chooses the particle swarm optimization algorithm as the optimization method of ship form optimization. This is because, compared with the team prior to the use of the genetic algorithm and the multi-objective version, in the single objective optimization, particle swarm optimization algorithm can quickly converge to a feasible solution; in addition, in the multi-objective optimization, when faced with multi peak and multi peak function, it can quickly close to the true Pareto front. Although the origin algorithm can converge to an approximate Pareto front in the face of multi peak functions, it is found that compared with the leading edge of the IMOPSO algorithm, it misses half of the edge. Therefore, the IMOPSO algorithm presented in this thesis is more adaptable to different optimization problems. The improvement of this key technology is an important guarantee for the successful application of ship hull optimization technology based on CFD and approximate models. At last, the thesis introduces the application of parallel computing in the optimization process. It is of great significance for the authors to develop parallel programs to reduce the computation time.