基于改进粒子群优化算法的网络化仿真任务共同体服务选择

孙黎阳[1,2], 林剑柠[2], 毛少杰[2], 刘中[1]
(1. 南京理工大学 电子工程与光电技术学院, 江苏 南京 210094;
2. 中国电子科技集团公司第 28 研究所 信息系统工程重点实验室, 江苏 南京 210007)

摘要: 作为网络化仿真中新的应用需求,如何动态地把散布在网络上各种服务整合起来以形成新的、满足不同用户需求的仿真任务共同体(STC)成为了当前研究热点。提出了一种基于粒子群优化(PSO)算法的仿真服务选择方法,针对传统 PSO 易陷入局部最优和收敛速度慢等不足,设计了一种惯性权重动态变化策略和一种可选的变异操作方法。该算法不仅能提高服务选择收敛速度,还能避免算法陷入局部最优。通过实验,采用典型函数进行了测试,并详细介绍了算法在 STC 服务选择上的实际运用,说明了算法的可行性和有效性。

关键词: 计算机应用; 网络化仿真; 任务共同体; 服务选择; 粒子群优化算法
中图分类号: N945.15 **文献标志码**: A **文章编号**: 1000-1093(2012)11-1393-11

Service Selection of Network Simulation Task Community Based on Improved Particle Swarm Optimization Algorithm

SUN Li-yang[1,2], LIN Jian-ning[2], MAO Shao-jie[2], LIU Zhong[1]
(1. School of Electronic Engineering and Optoelectronic Technology, Nanjing University of Science and Technology, Nanjing 210094, Jiangsu, China;
2. Information System Engineering Laboratory, China Electronic Science and Technology Group, Nanjing 210007, Jiangsu, China)

Abstract: As one of new application requirements in network simulation, to dynamically integrate the distributed various services in network to form a new simulation task community (STC) which meets the needs of different users has become current research focus. This paper presents a simulation service selection method based on the particle swarm optimization. The traditional particle swarm algorithm has some shortcomings that may easily fall into local optima and have slow convergence rate. We design a dynamic inertia weight strategy and a selectable method of mutation. The algorithm can improve the convergence speed not only, but also avoid falling into local optimum. Finally, some typical functions are chosen to test the algorithm. And the results show that the algorithm can select services feasibly and effectively for STC.

Key words: computer application; network simulation; task community; service selection; particle swarm optimization algorithm

收稿日期: 2012-02-17

作者简介: 孙黎阳(1985—), 男, 博士研究生。E-mail: coffee1219@hotmail.com;
刘中(1963—), 男, 教授, 博士生导师。E-mail: eezliu@mail.njust.edu.cn

0 引言

网络化仿真是军事仿真在网络中心战形势下发展的新阶段,它采用面向服务技术,以动态创建和运行仿真任务共同体的方式完成各种仿真应用任务,是一种新的仿真应用构建和运行方法。网络化仿真与传统的仿真方法相比,其主要特点之一是支持面向任务的仿真任务共同体动态构建和运行,实现网络上的仿真资源动态集成和运行[1-2]。在实际应用中,单个服务一般提供一些比较单一的功能,通常无法满足复杂应用的需求。因此,如何查找散布在网络中的仿真资源,集成单一服务所提供的各种功能以形成新的、功能更强大的仿真任务共同体来满足军事领域的复杂应用需求已成为一个新的研究热点。

仿真任务共同体的构建实际上是指对现有服务资源按照一定的业务逻辑进行集成,构建服务执行流程,从而更好地满足网络化仿真需求,是服务组合在特定领域的一种体现。服务组合广义上可以分为静态组合、半自动组合和自动组合[3]。在网络化仿真环境中,服务是经常变化的,大部分情况下,静态组合的方式并不能满足实际的应用需求,同时,由于完全智能化的自动组合方式是一个非常复杂的过程,因此,很多关于服务组合的应用和研究工作都侧重于半自动方式。半自动组合实现,首先要由业务人员根据特定的行业背景,建立适合具体应用需求的服务组合流程模型。服务组合流程模型由多个服务节点组成,各服务节点包含功能需求描述,但不指定具体的服务调用实例。在网络化仿真环境中,满足相同功能需求而具有不同服务质量(QoS)参数的服务存在多个,用户在对组合服务提出功能要求之外,也会对 QoS 提出全局约束要求。如何从中选择满足各服务节点功能 QoS 约束的具体服务,形成一个可执行的任务共同体来完成用户的需求就成为仿真任务共同体组合中的一个关键问题,本文称为网络化仿真任务共同体服务选择。

QoS 全局最优 Web 服务选择问题是被证明了的 NP 完全问题[4],受到学者的普遍关注。文献[5-6]根据每个任务中候选服务 QoS 的属性值进行加权和排序,选择加权和大的候选服务作为组合服务中的一部分,属于局部最优化的选择方法,无法选择出具有 QoS 全局最优的组合服务。文献[7-8]通过引入证书规划来求解,将所有的候选方案转换为线性 QoS 属性和约束条件进行计算,当组合规模大时,算法的计算量巨大,而整数规划要求目标函数和约束条件是线性的,因此限制了此类方法的实用性。文献[9-10]采用了遗传算法,将服务组合的方案编码为染色体,设计了适应度函数,通过群体适应度值寻优的方法进行服务选择。该算法可以在一定时间内近似最优解,但该算法并未针对早熟问题进行研究。文献[11-12]基于文献[10-11],通过采取多样性保持策略,在一定程度上解决了遗传算法早熟问题,最优解质量及执行时间均有所提高,但是其采用的遗传算法染色体信息共享、整个种群比较平均向最优区域移动,决定了收敛速度较慢。

为此,本文提出了一种基于粒子群优化(PSO)算法的仿真任务共同体 QoS 全局最优服务动态选择(DGOSS)实现算法,通过把仿真任务共同体服务组合中的服务动态选择 QoS 全局最优问题转化为一个带约束条件的多目标服务组合优化问题,利用 PSO 的智能优化原理,通过同时优化多个目标参数,即在不同的目标之间取均衡,最终产生一组满足约束条件的最优非劣服务组合解集。用户可以根据仿真需求在解集中选取最满意解,同时,未被选中的非劣解可以作为备选方案,以便所选聚合路径发生意外时启用。

1 仿真任务共同体服务组合模型

1.1 问题描述

假设在已建立的仿真服务语义描述规范之上,对仿真服务的能力、属性、开发意图、限制条件等进行了描述,为需求与服务之间的精确匹配提供前提和基础,避免语义等方面的冲突。另外为更好地研究服务组合问题,建立了一种仿真服务语义元数据的存储、索引和匹配机制。

在上述基础之上,本文着重研究如何从候选仿真服务中选择 QoS 最优的服务进行组合,来构建仿真任务共同体。假设一任务共同体包括 m 个共同体成员,每个共同体成员节点 N_i 对应的仿真服务群 SG_i 包含 n_i 个服务选项(WS_1,…,WS_{ni}). 即 SG_i = (WS_1,…,WS_{ni}),则仿真任务共同体构建流程图如图 1 所示。

构建流程如下:

1) 统一仿真资源描述规范,在任务共同体构建

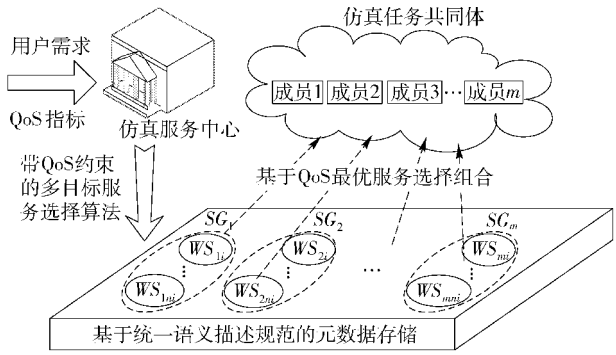


图 1　基于服务选择的仿真任务共同体构建流程

Fig. 1　Construction flow of simulation task community based on service selection

之前对仿真资源进行描述。

2）仿真应用服务构建前在基础服务层对仿真资源及核心服务进行发布，建立发布目录。

3）对仿真需求进行描述，生成配置文件，确定仿真所需仿真资源、服务。

4）通过仿真服务中心统一配置仿真应用成员之间的信息订购关系及内容，对描述的仿真需求配置文件进行资源动态查找、定位，从候选仿真服务中选择 QoS 最优的服务进行组合，确定仿真应用服务与模型间映射关系，构建仿真任务共同体。

1.2　服务流程模型及其 QoS 计算方法

仿真任务共同体可以分成多个子任务共同体，而仿真任务共同体实际上是一个基于服务的工作流。根据各个任务之间的逻辑关系，可以把仿真任务共同体服务组合流程分解成 4 种基本结构[13]：顺序结构；循环结构；并行结构；分支结构，如图 2 所示。

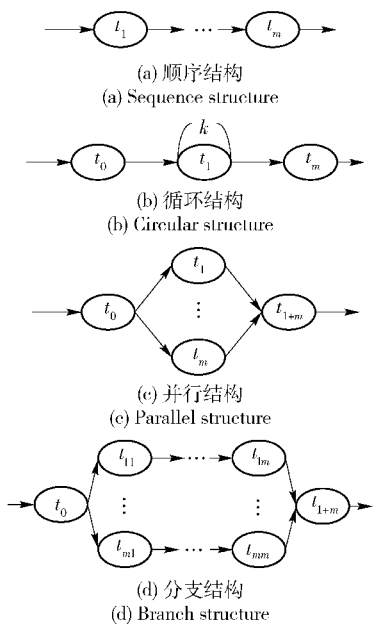


图 2　服务组合 4 种基本结构

Fig. 2　Four basic construction of service combination

组合服务是以上 4 种基本结构的组合和嵌套，因此组合服务的 QoS 可通过基本结构的 QoS 计算方法获得。本文选取仿真中 5 种常用非功能属性作为 QoS：服务价格（Price，P）、服务执行时间（Response Time，T）、服务延迟（Latency，L）服务可靠性（Reliability，R）和服务信誉（Reputation，Rep），如表 1 所示。

表 1　基本结构的 QoS 计算方法

Tab. 1　QoS calculation method for basic construction

结构	Price，P	Reliability，R	Reputation，Rep	Latency L	Response Time，T
顺序	$\sum_{i=1}^{m} P_{ws_i}$	$\prod_{i=1}^{m} R_{ws_i}$	$\prod_{i=1}^{m} Rep_{ws_i}$	$\sum_{i=1}^{m} L_{ws_i}$	$\sum_{i=1}^{m} T_{ws_i}$
循环	kP_{ws}	R_{ws}^{k}	Rep_{ws}^{k}	kL_{ws}	kT_{ws}
并行	$\sum_{i=1}^{m} P_{ws_i}$	$\prod_{i=1}^{m} R_{ws_i}$	$\prod_{i=1}^{m} Rep_{ws_i}$	$\max\{L_{ws_i}\}$	$\max\{T_{ws_i}\}$
分支	P_{WS_I}	R_{ws_i}	Rep_{ws_i}	L_{ws_i}	T_{ws_i}

1.3　任务共同体 QoS 全局最优服务选择模型

为方便阐述，本文将执行费用、执行时间和延迟作为 3 个目标准则，希望任务共同体服务选择执行费用极少、时间极短、网络中延迟最少。信誉等级和可靠性作为两个约束条件，Rep_0 和 R_0 分别为任务共同体所要求的最低信誉等级和最低可靠性，则一个带约束条件的任务共同体全局最优服务选择模型可以形式化地描述如下：

$$\min F(x) = (T(x), P(x), L(x)), \tag{1}$$

式中函数 $F(x)$ 受限于两个约束条件，即

$$\begin{cases} R(x) \geqslant R_0, \\ Rep(x) \geqslant Rep_0. \end{cases} \tag{2}$$

(1) 式表示取向量极小化,使得 $F(x)$ 中的目标函数值均同时极小化,实际上,服务 QoS 参数的种类很多,主要思想是将它们看作优化模型的目标函数和约束条件。所以,本文模型可以推广到任意多个目标函数和约束条件。需要说明的是,QoS 各属性量纲的不同、对整体效果的贡献不同,须首先对其进行归一化处理,由于非本文工作重点,在此不再赘述。

PSO 算法作为一种智能优化算法,具有并行计算、群体寻优的特点。同时 PSO 算法不需要与背景知识相关的启发式信息,只需确定其优化目标函数和约束条件即可,因此它已广泛应用于相关问题的求解。为此本文设计了一种仿真任务共同体 QoS 全局最优服务动态选择 DGOSS 实现算法,采用改进的 PSO 算法搜寻服务产生若干组解,供用户参考。

2 DGOSS 算法描述

2.1 PSO 算法简介及存在问题

PSO 算法是由 Kennedy 等[14]提出的一种全局优化进化算法。算法首先生成一群粒子,粒子本身具有两个参数:位置和速度。粒子的位置表示优化问题的一个解,粒子的速度表示位置变化的方向,指导粒子在迭代过程中向最优解的运动。粒子同时具有记忆功能,每次迭代时,粒子根据本身达到过的最优位置 x_p 和整个粒子群到达过的最优位置 x_g 来调整当前的位置。其速度和位置更新公式如下:

$$v_{i+1} = wv_i + c_1 r_1 (x_p - x_i) + c_2 r_2 (x_g - x_i), \tag{3}$$

$$x_{i+1} = x_i + v_{i+1}, \tag{4}$$

式中:w 为粒子保持惯性的权重;c_1、c_2 是加速度系数,分别为粒子向本身最优位置和全局最优位置移动的趋势;r_1、r_2 是(0,1)之间的随机数;x_i 为第 i 个粒子当前的位置;v_i 为其当前速度。

PSO 算法在解决带约束条件的数值优化问题时取得了很好的效果,但是如若将 PSO 算法运用于高维复杂多峰函数时,会容易陷入局部极值而导致早熟收敛[15]。PSO 算法会产生此问题主要有两方面原因:

首先,惯性权重大有利于增强算法的全局探索能力,而较小的惯性权重则更利于局部的精细搜索。因此,如何正确选取惯性权重对算法的求解效果会有很大影响。基本 PSO 算法在整个迭代过程中,惯性权重 w 保持不变,这很容易使算法要么陷入局部极值陷阱,要么算法的求精度不高,因此,惯性权重的设置存在难点。

其次,在基本 PSO 算法的进化过程中,有可能造成粒子快速地受局部最优解吸引而集中到其附近,使得群体多样性迅速降低。事实上,对于拥有大量局部极值的多峰高位函数,x_g 很有可能仅仅是一个局部最优解。

因此,采用 PSO 处理多目标优化工作要考虑以下两方面问题:1) 如何有效动态调整惯性权重,以适应不同阶段的搜索需求;2) 如何保证种群的多样性,以避免陷入局部最优。本文 DGOSS 算法从这两方面入手,设计了一个惯性权重动态调整策略并提出了基于自适应变异算法,分别从上述两个方面解决早熟问题。

2.2 惯性权重动态调整策略

通过前面的分析已经知道,惯性权重是平衡算法全局探索和局部开发能力的重要参数,对惯性权重的研究和应用已经有很多成果[16-20]。算法早期,需要尽可能大的全局探索,而算法后期,则更需要在可能的最优位置进行精细化开发。因此,对惯性权重的需要是根据算法运行不同阶段而定的,而决定算法所处阶段的基本方法是根据当前算法的迭代次数,但这会带来一种风险,即算法在中后期全局搜索能力大大降低,从而易陷入局部极值。

因此,在算法迭代次数上,综合考虑用当前粒子群的多样性来动态调节惯性权重,从而在增强局部开发能力的同时,降低所有粒子全部集中到某个局部点的风险,从而保证粒子群仍然存在一定的全局搜索能力。群体多样性采用文献[21]提出的测量方法:

$$\text{diversity}(t) = \frac{1}{m \times |L|} \sum_{i=1}^{m} \sqrt{\sum_{d=1}^{D} (x_{id} - \bar{x}_d)^2}, \tag{5}$$

式中:m 为粒子群规模;x_{id} 为第 i 个粒子 t 时刻的第 d 维分量;$\bar{x}_d$ 为 t 时刻所有粒子第 d 维分量的平均值;D 为问题的维数;$|L|$ 为搜索域的最长对角线长度。

惯性权重按 (6) 式进行动态调节:

$$w(t) = w_f + e^{-\text{diversity}(t)} \left(1 - \frac{T + T_{\min}}{T_{\max} + T_{\min}}\right)(w_i - w_f), \tag{6}$$

式中：T_{max} 和 T_{min} 分别为最大迭代次数和最小迭代次数；T 为当前迭代次数；w_i 为初始权值；w_f 为最终权值。

从(6) 式可以得出，当 T 较小，w 值较大时，粒子群全局搜索能力较强，随着 T 增加，w 逐渐减小，粒子群更加侧重于局部搜索。该调整策略使 PSO 寻优速度能自适应调整，提高了算法的收敛速度，使其不易陷入局部最优，避免早熟现象。

2.3　可选择的随机扰动算法

文献[22]根据粒子群的适应度方差作为全局最优化变异条件，提出自适应变异的粒子群优化算法。文献[23]采用进化停滞步数 t_r 作为触发条件，对个体极值 x_p 和全局极值 x_g 同时进行随机扰动。该方法属于调整个体极值和全局极值，使所有粒子飞向新的位置，这使所有粒子均进行迁移并重新聚集，经历新的搜索路径和领域，因此发现更优解的概率较大。

本文采用了一种可选的随机扰动方式，以针对粒子群同时发生两种情况：

1) 粒子群的适应值方差 σ^2 趋于 0，表示群体有可能陷入一个或多个最优点。适应值方差定义为

$$\sigma^2 = \sum_{i=1}^{n} \left(\frac{f_i - \bar{f}}{f}\right)^2, \qquad (7)$$

式中：f_i 为粒子 i 的适应值；$\bar{f}$ 为粒子群的群体平均适应值；$f = \begin{cases} \max|f_i - \bar{f}|, & \text{若 } \max|f_i - \bar{f}| > 1; \\ 1, & \text{其他。} \end{cases}$

2) 粒子群的全局最优值停滞步数超过了触发条件 t_r 表示群体有可能陷入局部最优。极值扰动的策略如下：

$$x_p = P_1 x_p,\ x_g = P_2 x_g, \qquad (8)$$

式中：P_1 为个体极值变异概率；P_2 为全局极值变异概率：$P_1 = \begin{cases} 1, & t_0 \leqslant T_0; \\ U(0,1), & t_0 > T_0. \end{cases}$ 和 $P_2 = \begin{cases} 1, & t_g \leqslant T_g; \\ U(0,1), & t_g > T_g. \end{cases}$ 其中：t_0 为个体极值进化停滞步数；t_g 为全局极值进化停滞步数；T_0、T_g 分别为个体极值和全局极值需要扰动的停滞步数阈值；$U(0,1)$ 表示(0,1)间随机数。

2.4　DGOSS 算法流程

仿真任务共同体 QoS 全局最优服务动态选择实现算法流程如图 3 所示。

1) 随机初始化粒子群中粒子的位置与速度。

2) 计算每个粒子的适应值。

3) 将每个粒子的当前适应值与该粒子的过去已有的最优位置 x_p 的适应值相比较。如果当前适应值优于该粒子的最优适应值，则用 x_p 记录下当前粒子的位置坐标。

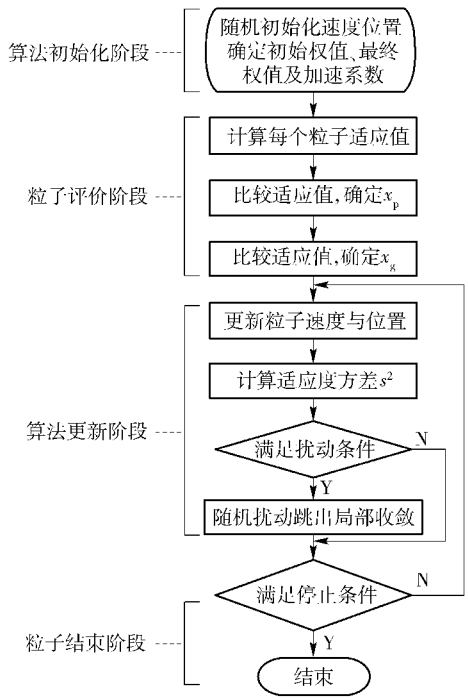


图 3　DGOSS 算法流程图

Fig. 3　Flowchart of DGOSS algorithm

4) 将每个粒子的当前适应值与群体最优位置 x_g 的适应值相比较。如果当前适应值优于粒子群的最优适应值，则用 x_g 记录下当前粒子的位置坐标。

5) 按照(1)式、(2)式更新粒子位置与速度。

6) 按照(7) 式计算群体的适应值方差 σ^2. 如果 σ^2 趋于 0 或者全局最优值停滞代数超过 t，则按照(8) 式进行极值扰动。

7) 判断算法收敛准则是否满足，如果满足，执行步骤 8)，否则转向步骤 4)。

8) 输出结果，算法结束。

3　DGOSS 算法验证与分析

3.1　DGOSS 算法性能验证与分析

为测试 DGOSS 算法在有约束条件下的性能，本文选用了 5 个常用测试函数，求它们的最小值，具体见表 2. 表 2 中的测试函数分为带约束优化函数(f_1, f_2, f_3)和非约束优化函数(f_4, f_5). 本文将所提算法与标准 PSO 算法、惯性参数衰减的粒子群优化(DWPSO)算法[24]、简化的粒子群优化(SPSO)算法[20]进行比较。表 3 列出了各个函数在相同进化

表 2 DGOSS 性能测试函数表

Tab. 2 DGOSS performance test functions

测试函数	维数	约束条件
$f_1: f(x) = (x_1 - 10)^3 + (x_2 - 20)^3$	2	$-(x_1-5)^2-(x_2-5)^2+100\leqslant 0$ $(x_1-6)^2+(x_2-5)^2-82.81\leqslant 0$ $13\leqslant x_1\leqslant 100$ $0\leqslant x_2\leqslant 100$
$f_2: f(x) = (x_1-10)^2+5(x_2-12)^2+x_3^4+3(x_4-4x_6x_7-10x_6-8x-11)^2+10x_5^6+7x_6^2+x_7^4$	7	$-127+2x_1^2+3x_2^4+x_3+4x_4^2+5x_5\leqslant 0$ $-282+7x_1+3x_2+10x_3^2+x_4-x_5\leqslant 0$ $-196+23x_1+x_2^2+6x_6^2-8x_7\leqslant 0$ $4x_1^2+x_2^2-3x_1x_2+2x_3^2+5x_5-11x_7\leqslant 0$ $-10\leqslant x_i\leqslant 100(i=1,2,\cdots,7)$
$f_3: f(x) = x_1 + x_2 + x_3$	8	$-1+0.000\,25(x_4+x_6)\leqslant 0$ $-1+0.002\,5(x_5+x_7-x_4)\leqslant 0$ $-1+0.01(x_8-x_5)\leqslant 0$ $-x_1x_6+833.332\,52x_4+100x_1-83\,333.333\leqslant 0$ $-x_3x_8+1\,250\,000+x_3x_5-2\,500x_5\leqslant 0$ $-x_2x_7+1\,250x_5+x_2x_4-1\,250x_4\leqslant 0$ $100\leqslant x_1\leqslant 10\,000$ $1\,000\leqslant x_i\leqslant 10\,000(i=2,3)$ $10\leqslant x_i\leqslant 1\,000(i=4,\cdots,8)$
$f_4: f(x) = \sum_{i=1}^{n} x_i^2$	30	$-100\leqslant x_i\leqslant 100(i=1,2,\cdots,30)$
$f_5: f(x,y) = 0.5 + \frac{(\sin\sqrt{x^2+y^2})^2-0.5}{(1.0+0.001(x^2+y^2))^2}$	2	$-100\leqslant x,y\leqslant 100$

表 3 算法的最优值、平均值、最差值比较

Tab. 3 Algorithm comparison in optimal, average, worst values

项目	算法	函数 f_1	函数 f_2	函数 f_3	函数 f_4	函数 f_5
进化代数		1 500	1 500	1 500	150	150
种群大小		30	30	100	15	15
最优值	PSO	-6 874.787 36	752.885 18	9 245.802 70	48 305.062 48	0.009 85
	DWPSO	-6 961.647 67	680.654 99	7 757.842 54	4 374.514 16	0.009 72
	SPSO	-6 824.492 06	701.590 61	8 830.105 50	0	0
	DGOSS	-6 972.882 61	680.532 94	7 035.670 72	0	0
平均值	PSO		855.054 97	10 785.176 88	61 444.260 52	0.016 89
	DWPSO		681.204 46	10 153.825 64	12 424.446 70	0.009 71
	SPSO		730.633 94	9 571.900 35	0	0.006 80
	DGOSS	-6 948.352 16	675.346 12	7 132.538 49	0	0
最差值	PSO		945.167 94	17 980.839 82	73 102.396 34	0.033 27
	DWPSO		685.055 12	13 683.468 84	19 445.485 70	0.009 74
	SPSO		767.518 56	10 188.274 45	0	0.009 72
	DGOSS	-6 924.834 56	681.468 34	7 381.348 76	0	0.000 00
适应值标准方差	PSO		65.010 28	2 583.887 14	8 753.861 10	0.007 95
	DWPSO		1.356 82	2 440.472 23	5 011.312 24	0.004 70
	SPSO		19.434 76	351.646 35	0	0.000 12
	DGOSS	0	0.156 63	96.346 49	0	0

代数下 30 次独立运行的最优值、平均值、最差值。

为便于比较,在仿真实验中各算法解决同一函数的演化代数设置为相同值。表 3 中符号“—”表示该算法在某函数优化过程中存在未求出合法解的情况,因此平均值和标准方差无法计算。

图 4(a) ~图 4(e)分别表示了函数 f_1、f_2、f_3、f_4 和 f_5 的适应值进化情况,图中横坐标表示迭代次数,纵坐标表示函数的适应值。图 4 中 DGOSS 在 30 次随机实验中最优值、平均值、最差值几乎均优于 PSO、DWPSO、SPSO 算法得出的结果。

在求解函数 f_1、f_4 和 f_5 的最小值时,DGOSS 在 30 次随机实验中最优值、平均值、最差值几乎均优于 PSO、DWPSO、SPSO 算法得出的结果。

在求解函数 f_1、f_4 和 f_5 的最小值时,DGOSS 算法适应值标准方差为 0,可知实验中该算法每次都能收敛到最优值。DGOSS 在求解函数 f_2 和 f_3 的最小值时,适应值标准方差远小于其他算法,也可说明该算法的稳定性好。从图 4 中可看出对于带约束的函数 f_1、f_2、和 f_3,迭代到 200 次时,DGOSS 基本达到了最优值,其中针对函数 f_2 采用 DGOSS 算法,在算法迭代初期约 50 次,即已收敛。对于非约束函数 f_4 和 f_5,迭代到 20 次时(图 4(d)和图 4(e)),DGOSS 基本达到了最优值。

从表 3 和图 4 中可分析出,DGOSS 算法在收敛精度和收敛速度上比 PSO、DWPSO、SPSO 有显著的提高。

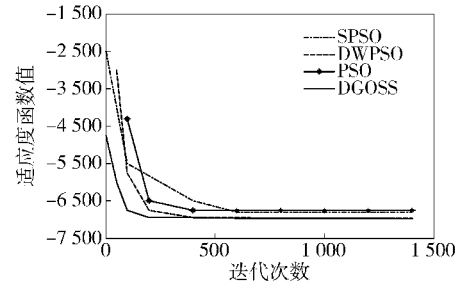
3.2　DGOSS 算法应用验证与分析

3.2.1　仿真任务共同体应用设计

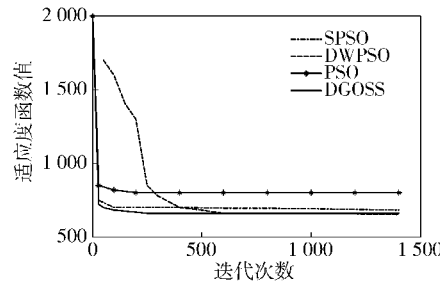
本节以构建一个作战仿真任务共同体来验证算法 DGOSS 在实际应用中的效率和性能。

仿真任务共同体是指为了完成共同的仿真任务,由分布在网络上的相关仿真资源和设施组成的“虚拟组织”。当任务执行完就解散。仿真任务共同体由若干成员组成,成员主要是仿真应用类资源,也可包括仿真支撑类中的仿真控制、数据采集等仿真管理设施资源。

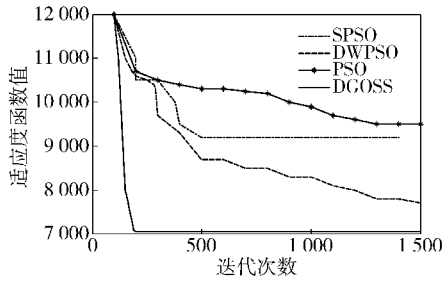
军事仿真任务共同体的构建是针对军事领域的一种服务选择过程,通过选择地理上分布的各种空间服务资源,为指挥控制部门的决策提供支持,共同体应用流程如图 5 所示。其中,S 为剧情软件服务;R 为雷达系统服务;C 为指控系统服务;W 为武器平台资源服务。每个共同体成员节点均有一个相应的仿真服务群,每个服务群又包含多个服务选项。



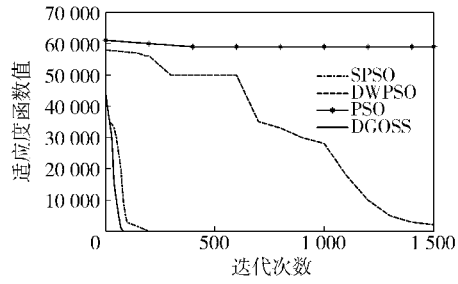
(a) 函数 f_1 适应值进化曲线
(a) Function f_1 fitness value evolution curve



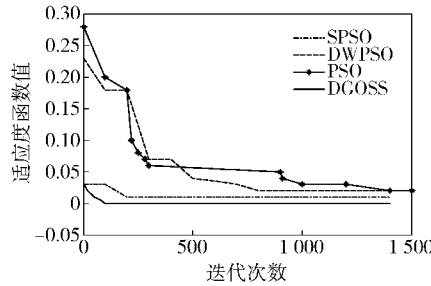
(b) 函数 f_2 适应值进化曲线
(b) Function f_2 fitness value evolution curve



(c) 函数 f_3 适应值进化曲线
(c) Function f_3 fitness value evolution curve



(d) 函数 f_4 适应值进化曲线
(d) Function f_4 fitness value evolution curve



(e) 函数 f_5 适应值进化曲线
(e) Function f_5 fitness value evolution curve

图 4　各算法适应度进化曲线

Fig. 4　Algorithm evolution curve

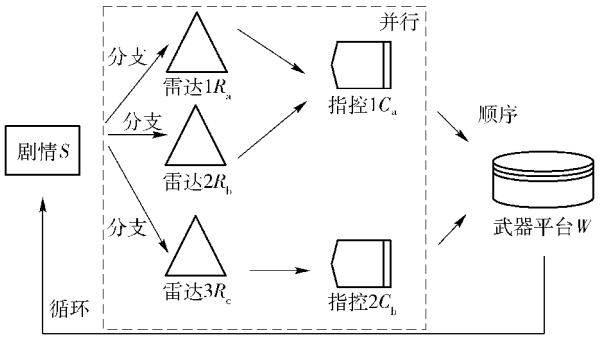


图 5 某仿真任务共同体应用流程

Fig. 5 Application process of a simulation task community

该仿真任务共同体囊括了图 2 中的顺序、循环、并行、分支 4 种逻辑关系。首先,从剧情服务集合中选择剧情服务软件来进行任务共同体任务设定;接受任务设定,通过服务搜索来查找雷达系统;雷达系统接收到剧情信息之后,同样通过服务查找获取指控中心,对雷达采集到的信息情报进行处理;处理完的信息传至武器平台,同样通过 DGOSS 来选择最优武器进行任务操作;将操作的内容返还给剧情软件,进行下一次的共同体运转。

在该流程中,根据第 1 节中描述,所有任务共同体成员均拥有 5 个非功能属性。本节实验目标是,将 DGOSS 运用于共同体服务选择过程中,来证明采用该算法可以优化服务选择结果。

流程中的每一个任务共同体成员对应一个服务群。服务群的建立和维护由仿真运行支撑平台来完成,本文不作描述。实验环境为 100 M 局域网,算法运行微机配置为 Core2 处理器,1 GB 内存,操作系统为 Windows XP,算法用 C + + 实现。各个服务群中服务的 QoS 参数采用随机方法在一定范围内生成,服务及其 QoS 参数信息在集中式 UDDI 注册中心进行注册,把仿真任务共同体的响应时间、费用和延迟作为 3 个目标准则,信誉等级以及可靠性作为两个约束条件。希望利用 DGOSS 从各个服务结点对应的服务群中选择具体服务形成可执行的任务共同体,使得共同体在满足两个约束条件的情况下,执行时间极短、费用极少、网络延迟小。按照(1)式作为目标准则,(2)式作为约束条件来进行性能判定。系统适应函数及约束条件在下节介绍。

3.2.2 仿真任务共同体目标函数设计

由第 1.2 节的描述可知,算法目标函数和约束函数分别对应聚合流程的 QoS 参数计算式。根据上文描述的服务组合流程基本模型及 QoS 评价方法,可以得到流程的 QoS 参数计算公式,其中,图 5 中 $R_aR_bR_c$ 构成一个分支模型;由并行模型 QoS 计算方法可以得到的 QoS 参数;从剧情 S 到武器平台 W 共同组成串联模型,而从武器平台 W 回到剧情则是一循环过程。综上所述,可由串联模型 QoS 计算方法可以得到仿真任务共同体服务组合的 QoS 参数计算式,从而得到算法的目标函数和约束函数。

设循环上限为 k

目标函数 $\min F = \{T,P,L\}$,其中:

$$\begin{cases} P:\left[S+\sum(R_a+R_b+R_c)+C_a+C_b+W\right]k; \\ L:[S+\max(R_a,R_b,R_c)+\max(C_a,C_b)+W]k; \\ T:[S+\max(R_a,R_b,R_c)+\max(C_a,C_b)+W]k. \end{cases} \tag{9}$$

约束条件 $\begin{cases} R\geqslant R_0, \\ Rep\geqslant Rep_0, \end{cases}$ 其中:$R=\mathrm{e}^{-(T+L)}$; $Rep = kR/(P+1)$;

$$\begin{aligned} R&:\left\{\prod\left[S\prod R\prod CW\right]\right\}^k; \\ Rep&:\left\{\prod\left[S\prod R\prod CW\right]\right\}^k. \end{aligned} \tag{10}$$

通过(9)式和(10)式得到五项属性值之后,将其进行标准化,将其统一至标准范围内。QoS 属性标准化是算法运行前必须完成的工作[25]。本文采用标准化后各 QoS 属性用于最终的适应度函数里,该工作非本文工作重点,在此不再赘述,具体的标准化方法见文献[26 - 28]. 最终参数范围标准化为 $0<T\leqslant 5, 0<P\leqslant 5, 0<Rep\leqslant 5, 0<R\leqslant 1, 0<L\leqslant 5$,最小信誉等级为 2,最小可靠性为 0.1. 在上述基础之上设计了任务共同体服务选择适应度函数

$$\text{Fitness}=\frac{\mathrm{e}^{T+L}}{1+\mathrm{e}^{\rho 2}}. \tag{11}$$

3.2.3 服务选择速度分析

将本文所设计的 PSO 算法运用于仿真任务共同体,从实际应用角度进行算法性能验证,目的是用于验证算法 DGOSS 找到 QoS 全局最优共同体成员的可行性。

由于多目标优化问题的解通常不是一个,而是一组,这些解的特点是,无法在改进任何目标函数的同时不削弱至少一个其他目标函数,即综合考虑所有的目标,不存在比它们更优的解,这些解被称为 Pareto 最优解,有时也笼统地称为非劣解、有效解

等[29]。用 DGOSS 算法对(9)式和(10)式进行求解,可以得到 Pareto 最优解。此时得到的 Pareto 解集并非是服务集合中的服务实例。决策者需要根据某种决策原则在这些 Pareto 最优解之间进行权衡,从而找出最终的满意解。

采用基于 Euclidian 距离的方法来获取实际最优解[30]。该参数反映了获取到的 Pareto 最优解与真实解集接近程度的关系。Euclidian 距离 d_i 表示了算法计算得到的解集与最接近的真实解集之间的距离。

$$d_i = \min_{k=1}^{|p^*|} \sqrt{\sum (f_m^{(i)} - f_m^{*(k)})^2}, \qquad (12)$$

式中:P^* 为所有可能得到真实最优解的解集;$f_m^{*(k)}$ 为P^* 中第 k 个成员的第 m 个目标函数值。

在上述基础上,将 DGOSS 算法用于仿真任务共同体的资源查找与构建,并将查找所需花费与穷举法作比较,其中穷举法即逐个地穷举计算出满足约束条件的成员的各个目标函数值最优解。每次试验均在各参数有效范围内随机取值;选择 3 种迭代次数:10 次、100 次、1 000 次;让粒子种群数分别从 100 递增至 1 000、200 递增至 2 000、400 递增至 4 000. 图 7 给出了 DGOSS 算法与穷举法在每种情况下对任务共同体服务选择所花费时间,所得的试验数据均是求 100 次平均后的结果。

从图 6 可以看出,在迭代次数较少且种群数较小情况下,穷举法花费时间要低于 DGOSS 算法。采用 DGOSS 算法,虽然由于种群数较小时花费时间较多,但随着迭代次数的增加以及种群数目的扩大,DGOSS 算法优势显露无疑。在迭代次数仅为 10 次时,当种群数目增加到 600 后,DGOSS 算法花费时间就要优于穷举法;而当迭代次数上升到 1 000 后,当种群数为 4 000 时候,DGOSS 平均花费时间要比穷举法节约八万多毫秒,即八十多秒。这在仿真任务共同体的运用构建中,很大程度上节约了服务组合的时间,是非常有效的。因此,图 6 验证了 DGOSS 算法运用于大规模仿真资源参与的任务共同体构建可以有效减少系统时间花费,提高了搜索速度。

3.2.4　服务选择性能分析

上节中证明了将 DGOSS 算法运用于任务共同体的构建,能提高搜索速度,特别是资源数目较多,种群规模较大的时候,采用 DGOSS 可以节约较多资源组合时间。在本节,对 DGOSS 算法与传统 PSO 算法进行搜索结果性能比较。

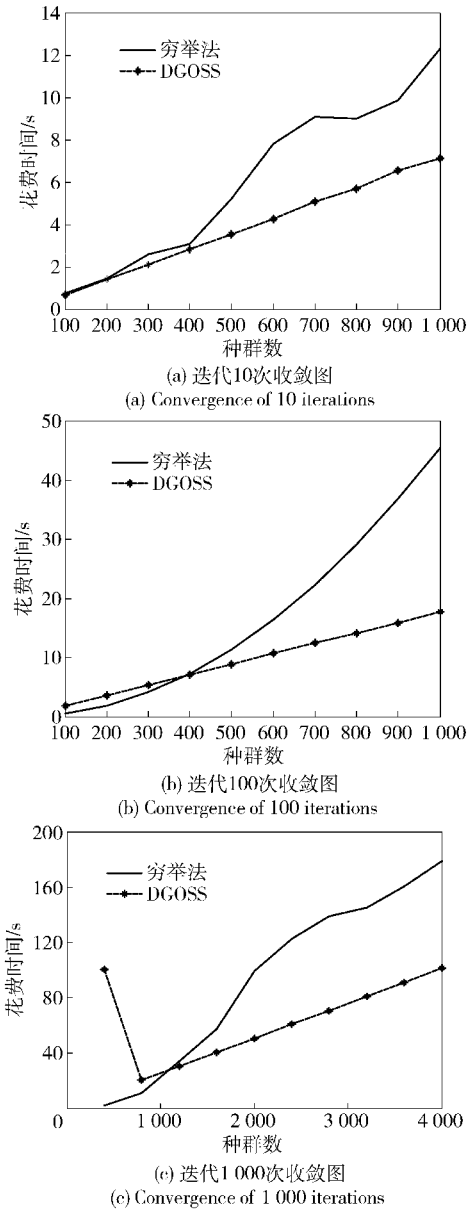


图 6　服务选择花费时间对比图

Fig. 6　Comparison of service selection time cost

设定种群数目为 100,500,1 000;迭代次数为 1 000,将 PSO 与 DGOSS 分别运用于同一任务共同体构建过程,按照前文所描述方法得到 Pareto 最优解集,将解集中的资源与 PSO 计算所得出的最佳资源进行对比,从而得到两种算法计算出最佳资源的概率,以此来比较两者的服务选择性能。图 7 显示了算法求得解集中的资源与 PSO 算法所求得最优资源相同的比率,以此来判断算法性能的优劣性。

通过图 7 对比,可以看出,不论种群大小为多大

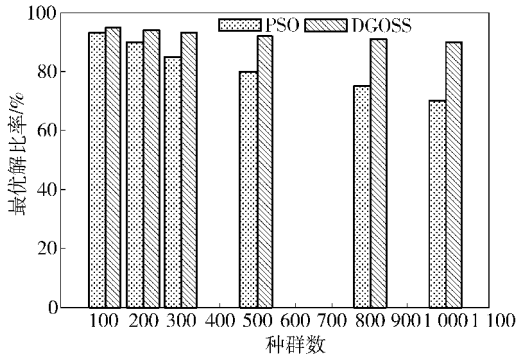


图 7 算法最优解比例对比图

Fig. 7 Optimal solution ratio

情况下,PSO 算法得到最优解的概率均要小于 DGOSS 算法。DGOSS 算法在有限进化代数(1 000 代)条件下找到各优化指标最优值的概率都在 90% 以上,因此, DGOSS 解决仿真任务共同体服务组合的性能是较优的,解决带约束多目标优化(QoS 全局优化)问题是可行且有效的。

4 结论

本文针对网络化仿真任务共同体服务选择问题,通过把仿真任务共同体服务组合中的服务动态选择 QoS 全局最优问题转化为一个带约束条件的多目标服务组合优化问题运用了改进的 PSO 算法,提出了一种改进的 PSO 算法的仿真任务共同体 QoS 全局最优服务动态选择实现算法 DGOSS. 采用了权值动态自适应调整策略和可选择的随机扰动算法,通过这两方面的改进,分别来加强粒子搜索能力及种群多样性,从而避免了在搜索过程中早熟现象,加快了收敛速度并提高了搜索结果为最优解的概率。首先采用典型函数分别从无约束情形和有约束情形进行了测试,结果验证了 DGOSS 算法性能;然后结合实际应用,将算法运用于仿真任务共同体构建过程,并对横向纵向两个方面对算法进行了对比,证明了 DGOSS 算法相比其他 PSO 算法,可以做到“又快又好”。综上所述,DGOSS 算法的性能符合理论依据,具有可行性和有效性。

参考文献(References)

[1] 毛少杰,李玉萍,林剑柠,等. 网络中心化仿真概念与方法研究[J]. 系统仿真学报,2010,22(7):1660 – 1663.
MAO Shao-jie, LI Yu-ping, LIN Jian-ning, et al. Research on concepts and technique of network-centric simulation[J]. Chinese Journal of System Simulation, 2010, 22(7): 1660 – 1663. (in Chinese)

[2] Sun L Y, Mao S J, Liu Z, et al. Research on the runtime support platform for the net-centric simulation[C] // IEEE International Conference on Advanced Computer Theory and Engineering Conference. New York: ASME,2010: 253 – 257.

[3] Shalil M, Walker D W, Gray W A. A framework for automated service composition in service-oriented architectures[C] // Proceedings of the ESWS 2004. Berlin: Springer-Verlag, 2004: 269 – 283.

[4] Yu T K, Lin J. Service selection algorithms for composing complex services with multiple QoS constraints[C] // Proceedings of the 3rd International Conference on Service Oriented Computing. Amsterdam: Springer, 2005: 130 – 143.

[5] Benatallah B, Dumas M, Sheng Q Z, et al. Declarative composition and peer-to-peer provisioning of dynamic Web services[C] // Proceedings of the 18th International Conference on Data Engineering. San Jose: IEEE, 2002: 297 – 308.

[6] Liu Y, Ngu A H, Zeng L Z. QoS computation and policing in dynamic Web service selection[C] // Proceedings of the 13th International Conference on World Wide Web. New York: ACM, 2004: 66 – 73.

[7] Zeng L Z, Benatallah B, Ngu A, et al. QoS-aware middleware for Web services composition[J]. IEEE Transactions on Software Engineering, 2004,30(5): 311 – 327.

[8] Ardagna D, Pernici B. Adaptive service composition in flexible processes[J]. IEEE Transactions on Software Engineering, 2007, 33(6): 369 – 384.

[9] Zhang L J, Li B, Chao T, et al. On demand Web services based business process composition[C] // International Conference on System, Man, and Cybernetics. Washington: IEEE, 2003: 4057 – 4064.

[10] Canfora G, Penta M D, Esposito R, et al. A lightweight approach for QoS-aware service composition[C] // Proceedings of the 2nd International Conference on Service oriented Computing. New York: ACM, 2004: 36 – 47.

[11] Ma Y, Zhang C W. Quick convergence of genetic algorithm for QoS-driven Web service selection[J]. Computer Networks, 2008, 52(5): 1093 – 1104.

[12] 刘书雷,刘云翔,张帆,等. 一种服务聚合中 QoS 全局最优服务动态选择算法[J]. 软件学报, 2007, 18(3): 646 – 656.
LIU Shu-lei, LIU Yun-xiang, ZHANG Fan, et al. A dynamic Web services selection algorithm with QoS global optimal in web services composition[J]. Chinese Journal of Software, 2007,18(3): 646 – 656. (in Chinese)

[13] WFMC. Workflow management coalition terminology & glossary, WFMC-TC-1011[R]. United Kingdom: Workflow Management Coalition, 1999.

[14] Kennedy J, Eberhart R. Particle swarm optimization[C] // International Conference on Neural Networks. Washington: IEEE,

1995: 1942 – 1948.

[15] Zhao X C, Gao X S, Hu Z C. Evolutionary programming based on non-uniform mutation[J]. Applied Mathematics and Computation, 2007,192(1): 1 – 11.

[16] 王文彬,孙其博,赵新超,等. 基于非均衡变异离散粒子群优化算法的 QoS 全局最优 Web 服务选择方法[J]. 电子学报, 2010,38(12): 2774 – 2779.
WANG Wen-bin, SUN Qi-bo, ZHAO Xin-chao, et al. Web services selection approach with QoS global optimal based on discrete particle swarm optimization with non- uniform mutation algorithm[J]. Chinese Acta Electronica Sinica, 2010, 38 (12): 2774 – 2779. (in Chinese)

[17] 龙文,梁昔明,董淑华,等. 动态调整惯性权重的粒子群优化算法[J]. 计算机应用, 2009, 29(8): 2240 – 2242.
LONG Wen, LIANG Xi-ming, DONG Shu-hua, et al. Particle swarm optimization with dynamic change of inertia weights[J]. Chinese Journal of Computer Applications, 2009, 29(8): 2240 – 2242. (in Chinese)

[18] 张长胜,孙吉贵,欧阳丹彤. 一种自适应离散粒子群优化算法及其应用研究[J]. 电子学报,2009,37(2):299 – 304.
ZHANG Chang-sheng, SUN Ji-gui, OUYANG Dan-tong. A self-adaptive discrete particle swarm optimization algorithm[J]. Chinese Acta Electronica Sinica, 2009,37 (2): 299 – 304. (in Chinese)

[19] Yang Q Y, Sun J G, Zhang J Y, et al. A hybrid discrete particle swarm algorithm for open2shop problems[C]//Proceedings of the 6th International Conference on Simulated Evolution and Learning. Hefei, China: LNCS, 2006:158 – 165.

[20] Lian Z G, Gu X S, Jiao B. A similar particle swarm optimization algorithm for permutation flowshop scheduling to minimize makespan[J]. Applied Mathematics and Computation, 2006,175(1): 773 – 785.

[21] Riget J, Vesterstorem J S. A diversity-guided particle swarm optimizer-the ARPSO[R]//Technical Report 2002-02. Denmark: Department of Computer Science, University of Aarhus, 2002: 345 – 350.

[22] 吕振肃,侯志荣. 自适应变异的粒子群优化算法[J]. 电子学报, 2004,32(3):416 – 420.
LV Zhen-su, HOU Zhi-rong. Paticle swarm optimization with adaptive mutation[J]. Acta Electronica Sinica, 2004, 32(3): 416 – 420. (in Chinese)

[23] 胡旺, 李志蜀. 一种更简化而高效的粒子群优化算法[J]. 软件学报, 2007, 18(4): 861 – 868.
HU Wang, LI Zhi-shu. A simpler and more effective paticle swarm optimization algorithm[J]. Chinese Journal of Software, 2007, 18(4): 861 – 868. (in Chinese)

[24] Shi Y, Eberhart R C. Particle swarm optimization: development, applications and resources[C]//Proceedings of the IEEE International Conference of Evolutionary Computation. Piscataway, USA: IEEE Press,1998: 69 – 73.

[25] Atanassov K T. Intuitionistic fuzzy sets[J]. Fuzzy Sets and Systems, 1986, 20(1): 87 – 96.

[29] Ulungu E L, Teghem J. Multi-objective combinatorial optimization problems: a survey[J]. Journal of Multi-Criteria Decision Analysis, 1994, 3(2): 83 – 104.

[30] Veldhuizen V. Multiobjective evolutionary algorithms: classifications, analyses, and new innovations[D]. Ohio: Air Force Institute of Technology, 1999.