

## 军事信息系统服务资源分配并行优化方法

于晓浩<sup>1,2</sup>, 胡 丹<sup>3</sup>, 罗雪山<sup>2</sup>, 刘俊先<sup>2</sup>

(1. 北京系统工程研究所, 北京 100101; 2. 国防科技大学 信息系统工程重点实验室, 长沙 410073;  
3. 空军工程大学 电讯工程学院, 西安 710077)

**摘 要** 针对如何提高面向服务军事信息系统中任务 workflow 执行的时效性和成功概率, 提出了服务资源分配的并行优化方法. 首先给出了服务资源分配的系统框架, 在分析服务并行执行数目、任务成功率、任务完成时间及服务执行代价之间关系的基础上, 建立了服务并行优化的目标规划数学模型, 并提出了一种求解该模型的改进粒子群算法 (DPSO). 该算法通过引入粒子细微扰动、优化粒子飞行边界及粒子优胜劣汰等扩大搜索范围, 提高获得最优解的概率. 实验结果表明服务分配的并行优化及其 DPSO 求解算法是提高任务 workflow 执行成功率和时效性的有效方法.

**关键词** 服务并行优化; 军事信息服务; 粒子群算法

## Parallel optimization method of service resource allocation in military information system

YU Xiao-hao<sup>1,2</sup>, HU Dan<sup>3</sup>, LUO Xue-shan<sup>2</sup>, LIU Jun-xian<sup>2</sup>

(1. Beijing Institute of System Engineering, Beijing 100101, China; 2. Science and Technology on Information Systems Engineering Laboratory, National University of Defense Technology, Changsha 410073, China; 3. Telecommunication Engineering Institute, Airforce Engineering University, Xi'an 710077, China)

**Abstract** Towards how to improve the efficiency and successful probability of task-workflows in service oriented military information system, a parallel optimization method of service resource allocation was proposed. Firstly, a service resource allocation framework is offered. By analyzing the relationship of service executing number, task executing time, task successful probability and service executing cost, a target programming mathematical model for parallel optimization of service resources was established. Then, an ameliorated particle swarm optimization (called DPSO) algorithm was proposed to resolve the mathematical model. By introducing random disturbance, searching boundary optimization and survival of the fittest for the particles, DPSO extended searching scope to obtain the optimal solution with a higher probability. Experimental results show that parallel optimization method of service resource allocation and the DPSO algorithm are effective methods to improve the efficiency and successful probability of task-workflows.

**Keywords** parallel optimization of services; military information service; PSO arithmetic

### 1 引言

军事信息系统综合集成是国内外学者长期研究的一个焦点问题. 现阶段, SOA/Web 服务技术以其松散耦合、支持应用系统高效整合和业务流程随需应变等特点, 有效地解决了系统功能复用和异构系统间信息交互的问题, 正在成为推动系统综合集成的强大力量. 美军的全球信息栅格 (global information grid, GIG) 已开始研究采用 SOA 体系架构, 通过开发网络中心企业服务 (NCES)<sup>[1]</sup>, 实现对上层各种军事业务应用的支持, 并最终实现网络中心化跨系统的信息共享及军事应用的快速建立和综合集成. 与民用领域相比, 军用网络处于复杂的战斗环境中, 军事信息服务随时会受到基础设施失效、通信模式变化、网络失效、服务拒绝攻击等问题的影响而导致服务调用超时或失效, 进而造成战机的贻误, 甚至影响整个作战任务的顺利完成, 导致不

收稿日期: 2010-06-03

资助项目: 国家自然科学基金 (70601036); “十一五” 装备预先研究项目

作者简介: 于晓浩 (1981-), 男, 汉, 山东莱阳人, 博士研究生, 研究方向: 体系结构, 信息系统集成.

可挽回的后果. 因此, 如何保证军事信息系统中任务工作流执行的时效性和成功率, 是面向服务的军事信息系统集成要解决的一个关键问题.

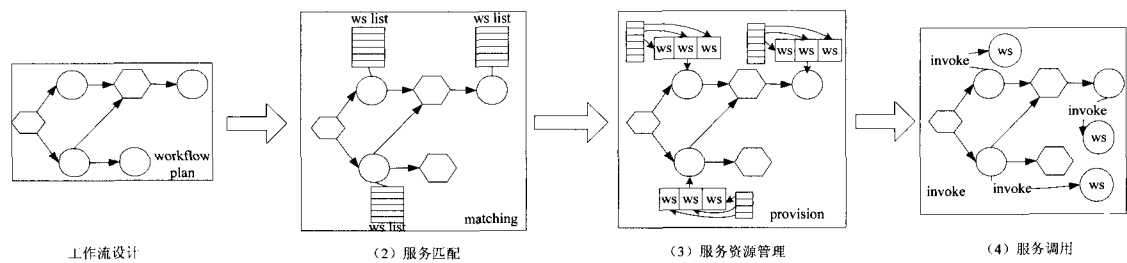


图 1 服务组合过程

从图 1 所示的基于工作流的服务组合过程看, 在服务组合的每个阶段都存在多种方法提高任务执行的时效性和成功率. 如文献 [2] 从工作流设计的层面研究了提高任务完成成功率的方法, 对业务流程建模语言 WS-BPEL 进行了改进, 通过在工作流中指定服务失效的替代方案, 使其支持服务失效的异常处理, 保证任务的完成. 但是该方法很难适应于大规模的网络, 尤其在事先无法知道服务状态的动态环境下, 其应用受到限制. 文献 [3-4] 从服务匹配的角度, 针对服务 QoS 的可靠性和时效性指标, 提出了具有服务质量保证的服务选择算法. 文献 [5] 从服务调用的角度, 对服务之间交互的消息格式进行改进, 以提高服务调用的时效性. 本文的研究集中于服务资源管理方面, 在这方面文献 [6-7] 提出了一种服务失效替换算法, 对服务匹配阶段发现的大量功能相似的服务按用户的 QoS 需求进行排队, 当有服务调用失效时, 在服务队列中选择合适的服务进行替换以提高任务完成的成功率, 但该方法在提高时效性方面没有得到改进, 而且由于服务的重新调用还带来了额外的时间开销. 文献 [8-9] 采用目标规划的方法, 通过建立服务组合完成的时间和成功率的目标函数, 求解一条满足需求的服务调用最优路径, 该类方法虽然综合考虑了任务工作流完成的时效性和成功率, 但当服务执行路径中存在服务执行瓶颈时, 即使求得的路径最优, 依然无法满足用户需求.

针对上述问题, 本文并提出了一种服务资源分配的并行优化方法, 以同时提高任务工作流完成的时效性和成功率. 首先介绍了服务资源分配的系统框架, 并对服务组合的并行优化问题进行了描述; 其次分析了单任务情况下服务并行执行数目、任务完成时间以及成功概率之间的关系, 在此基础上建立了服务组合情况下服务资源并行优化的数学模型, 并基于改进的粒子群算法对问题进行求解.

2 服务资源分配系统框架

在服务资源管理的系统实现框架方面, Aron<sup>[10]</sup> 等人提出了资源容器的概念对网络中的服务资源进行集中式管理, 文献 [11] 基于 Aron 等人的研究成果提出了具有 QoS 保证的服务资源管理系统 (QG-SRMS), 该系统对服务资源进行分类, 由服务虚拟资源容器 VRC 进行服务的管理, 本文借鉴了 QG-SRMS 的有关研究成果, 提出了如图 2 所示的服务资源分配系统框架, 其对服务请求的处理流程如下:

- 1) 各虚拟的服务资源管理容器 (service resource manage container, 简称 SRMC) 根据服务匹配阶段发现的大量功能相似的服务, 动态维护一个服务资源列表.
- 2) 当有服务请求到达时, SRMC 根据服务请求中要求的并行执行的服务数目, 动态绑定多个服务实例, 通过服务的并行执行, 保证任务完成的时间和成功率要求.
- 3) 当 SRMC 对应的任务完成后, SRMC 释放相关资源.

虚拟服务资源管理容器的存在可以实现服务资源的运行时绑定, 提高服务并行执行的成功率. 同时由于 SRMC 的虚拟性, 当任务完成后 SRMC 的相关资源会被释放从而降低了系统的额外开销.

在上述服务请求处理流程的第 2 步描述中, 如何确定服务并行执行的数目, 以保证任务完成的时间和成功率是本文要研究的主要问题.

为了便于问题的分析, 本文对图 2 中的任务工作流给出如下定义:

**定义 1** 任务工作流:  $W = (T, E, Q)$  表示工作流设计阶段建立的抽象工作流. 其中,  $T = \{t_1, t_2, \dots, t_n\}$  表示工作流中任务的集合, 代表了对服务的功能需求.  $E = T \times T$  是任务之间关系的集合, 如果  $\langle t_i, t_q \rangle \in E$ , 则表示任务  $t_i$  是任务  $t_q$  的前提, 即只有在任务  $t_i$  完成的情况下才可以执行任务  $t_q$ .  $Q = \{Q_1, Q_2, \dots, Q_k\}$

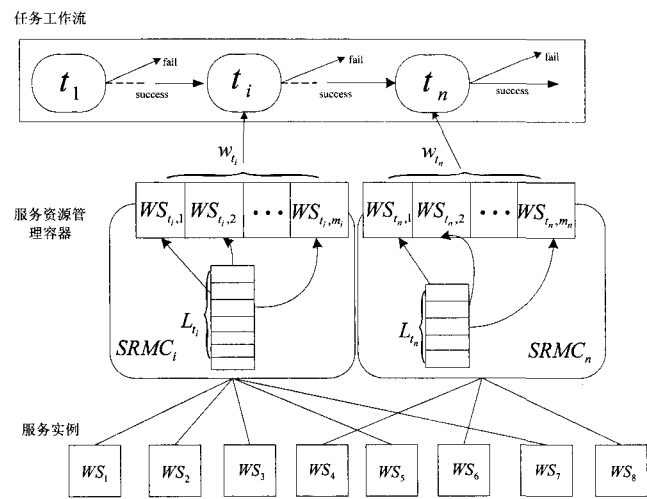


图 2 服务资源分配系统框架

表示用户对工作流中有关任务或任务组合提出的 QoS 约束, 主要包括时间约束和成功率约束.

3 服务并行执行对单任务的影响

以图 2 中第  $i$  个任务的执行为例, 令  $SRMC_i$  为第  $i$  个任务  $t_i$  对应的虚拟服务资源管理容器,  $t_i$  所对应的服务资源总数用  $L_i$  表示,  $W_{t_i}$  为  $t_i$  的等待时间,  $WS_{t_i,j}$  为  $t_i$  对应的第  $j$  个并行执行的服务 ( $j \in [1, m_i]$ ,  $m_i$  为  $t_i$  对应的并行执行的服务数目). 假设  $t_i$  所对应服务源列表中的服务实例是相互独立的, 令  $WS_{t_i,j}$  执行的成功概率为  $f_{t_i,j}$ , 执行时间服从均值为  $\mu_{t_i,j}$ 、方差为  $\sigma_{t_i,j}$  的正态分布. 则当等待时间为无穷大时,  $t_i$  完成的成功率可如下表示:

$$P_{t_i}(m_i) = 1 - \prod_{j=1}^{m_i} (1 - f_{t_i,j}) \tag{1}$$

$t_i$  完成时间的分布函数可以表示为  $N_{t_i} = \min(ts_{t_i,1}, ts_{t_i,2}, \cdots, ts_{t_i,m_i})$  ( $ts_{t_i,j}$  为第  $j$  个服务执行时间的分布) 的分布函数. 由于  $ts_{t_i,j}$  服从正态分布, 所以  $N_{t_i}$  的分布函数可表示为:

$$F_{t_i}(t, m_i) = 1 - \prod_{j=1}^{m_i} (1 - F_{t_i,j}(t)) \tag{2}$$

其中  $F_{t_i,j}(t) = \frac{1}{\sqrt{2\pi}\sigma_{t_i,j}} \int_0^t e^{-\frac{(t-\mu_{t_i,j})^2}{2\sigma_{t_i,j}^2}} dt$  为  $t_i$  对应的第  $j$  个服务的时间分布函数.

在时间  $W_{t_i}$  范围内  $t_i$  执行的成功概率可表示为:

$$PW_{t_i}(w_{t_i}, m_i) = P_{t_i}(m_i) \cdot F_{t_i}(w_{t_i}, m_i) \tag{3}$$

为了便于说明, 本文对同一组服务集合进行分析, 服务集合产生规则如表 1 所示. 在分析中令  $W_{t_i}$  分别为 8s、9s、10s、11s、12s, 则基于 (3) 式求得的服务集合中任务执行成功概率与并行执行的服务 (从数据集中随机选择) 数目之间的关系如图 3 所示.

表 1 服务集合产生规则

对象	数据集产生规则
服务执行时间均值	(8s-10s) 均匀分布
服务运行时间方差	(3s-5s) 均匀分布
服务成功率	(0.88-0.98) 均匀分布
服务数目	100

从图 3 的结果中可以看出, 并行执行的服务数目 ( $m_i$ ) 越多, 任务完成的成功率越高. 当服务资源有限时, 通过增加服务执行等待时间可以提高任务完成的成功概率. 图 3 还表明, 当并行执行服务数目  $m_i > 11$  时, 不同的等待时间 ( $w_{t_i}$ ) 下任务执行的成功概率基本相同 (接近于 1), 并行执行服务数目的增加或任务等待时间的增加对任务成功概率影响不大, 因此并不是  $m_i$  或  $w_{t_i}$  的值越大越好. 另外, 虽然服务并行执行数目

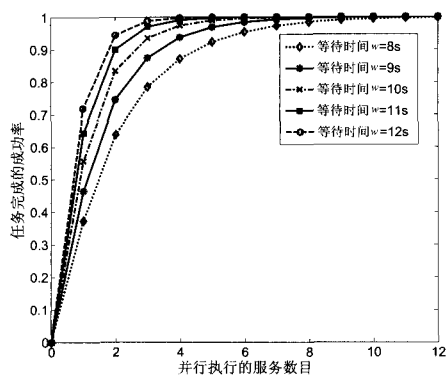


图 3 并行执行服务数目与任务成功率之间关系

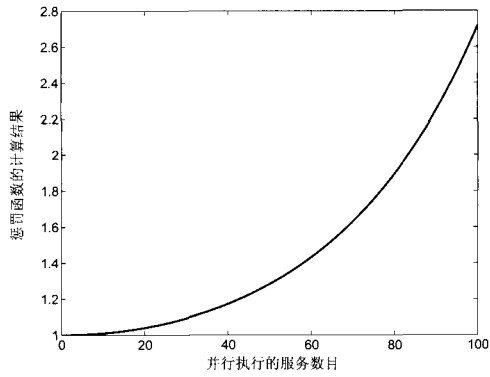


图 4 惩罚函数分布曲线

的增加可以提高任务完成的成功概率、减少任务等待时间,但是服务执行数目的增加会给网络带来额外的开销,如果并行执行的服务数目太多,当同时有其它工作流中的任务对该类服务提出请求时,将可能导致无法满足新任务的需求.因此,为单个任务提供并行执行的服务时,应在保证任务成功率和完成时间的前提下尽量优化并行执行服务的数目.本文假设由于服务的并行执行而带来的额外开销用定义 2 给出的惩罚函数表示.

**定义 2** 惩罚函数:  $C_i(m_i) = e^{(\frac{m_i}{L_i})^2}$  表示第  $i$  个任务中  $m_i$  个服务并行执行对系统带来的额外开销 (本文称为服务执行代价,用  $C_i$  表示).图 4 为当  $L_i = 100$  时  $C_i$  的分布情况,该函数服从指数分布,随着并行执行服务数目的增加,惩罚函数的增长速度逐渐加快,这在一定程度上反映了  $SRMC_i$  对服务资源管理的效能,即针对单个任务并行执行的服务数目越多  $C_i$  的值越大,  $SRMC_i$  的资源管理效能越弱,而且减弱的速度越来越快.惩罚函数的计算方式可以根据具体情况进行进一步的优化.

4 服务组合条件下的服务资源分配及求解

4.1 数学模型

第 2 节针对单个任务分析了任务完成的成功概率与并行执行的服务数目、任务完成等待时间之间的关系,下面讨论服务组合的情况.假设工作流中有  $n$  个任务需要完成,令  $\vec{m} = \langle m_1, m_2, \dots, m_n \rangle$ ,  $m_i$  为  $t_i$  所对应的并行执行的服务数目.则通过选择合理的  $\vec{m}$  既可以满足用户对任务完成的时间和成功率需求,又可以使服务执行的代价最小.

在军事应用中,用户对服务组合的 QoS 需求往往具有多阶段特征.即在一个完整的任务工作流中,针对任务执行的不同阶段分别对任务的完成时间、成功概率等提出要求.本文以针对某次袭击制定隐蔽方案为例进行说明,其任务工作流程如图 5 (图中的“**And**”表示与关系)所示,分为 4 个阶段:

- 1) 获取威胁目标的属性及运动轨迹 (用 T1 表示);
- 2) 计算威胁目标打击范围 (用 T2 表示);
- 3) 获取敌方目标打击范围内我方需要隐蔽的资源 (用 T3 表示) 以及该范围内的地理环境信息 (用 T4 表示);
- 4) 制定我方资源隐蔽方案 (用 T5 表示).

为保证作战任务的顺利完成,任务工作流中每个阶段都可能对任务完成的时间或成功概率提出要求,假设本例中具体的时间和成功率要求由图 5 中  $W_j$  和  $P_j$  给出.对于上述多阶段的目标优化问题而言,不同阶段约束条件之间存在相互协调关系,以任务执行的前两个阶段为例,任务 T1 的完成时间小于 10 秒、成功概率大于 98%,任务 T1 和任务 T2 组合的完成时间小于 15 秒,成功概率大于 96%,对于时间约束而言,如果任务 T1 的时间缩短了则留给任务 T2 的时间将更加充裕,由第 3 节的分析可知,时间的更改会影响并行执行的服务数目及执行代价,因此需要建立合理的数学模型对问题进行求解.

假设任务工作流中服务执行的总代价为各任务对应的服务执行代价之和,用  $C(\vec{m})$  表述,则:

$$C(\vec{m}) = \sum_{i=1}^n C_i(m_i) = \sum_{i=1}^n e^{(\frac{m_i}{L_i})^2}$$

(4)

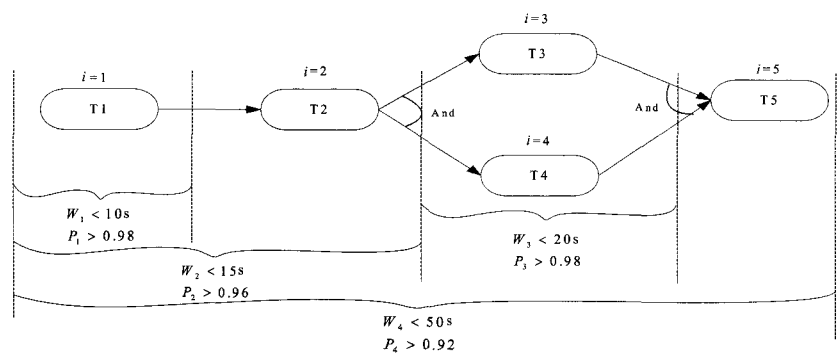


图 5 任务流程建模实例

在假设  $F_{t_i}(t)$  相互独立的情况下, 虽然可以通过计算特征函数及其反变换, 求得  $Z = F_{t_1}(t) + F_{t_2}(t) + \cdots + F_{t_n}(t)$  的分布函数, 进而求得各阶段规划的可行解, 但很难保证解的最优性, 甚至局部最优性. 因此, 需要建立合理的数学模型反映  $\overline{m}$ 、 $W_i$ 、 $P_i$  以及目标函数之间的关系, 本文将其转化为目标规划问题进行求解, 其数学模型如下:

$$\min C(\overline{m}) = \min \sum_{i=1}^n e^{\left(\frac{m_i}{L_i}\right)^2}$$

(5)

s.t.

$$\begin{cases} (PW_1(w_1, m_1), PW_2(w_2, m_2), \cdots, PW_1(w_k, m_k)) > (P_1, P_2, \cdots, P_k) \\ < w_1, w_2, \cdots, w_k > < W_1, W_2, \cdots, W_k > \end{cases}$$

(6)

以图 5 给出的任务流程为例, 假设各任务对应的服务资源列表中有关服务的信息如表 2 所示, 则以实现服务执行总代价最小为目标 (目标函数为式 (7)) 建立的约束条件如式 (8) 所示.

表 2 各任务对应服务资源列表的有关服务信息

任务名称	服务执行时间均值	服务运行时间方差	服务成功概率	服务数目
T1	9s	4s	(0.97-1) 均匀分布	50
T2	6s	3s	(0.96-0.99) 均匀分布	100
T3	18s	8s	(0.96-1) 均匀分布	50
T4	12s	6s	(0.97-1) 均匀分布	10
T4	16s	4s	(0.97-1) 均匀分布	20

$$\min C(\overline{m}) = \min \sum_{i=1}^5 e^{\left(\frac{m_i}{L_i}\right)^2}$$

(7)

s.t.

$$\begin{cases} PW_1(w_1, m_1) > 0.98 \\ PW_1(w_1, m_1) \cdot PW_2(w_2, m_2) > 0.96 \\ \min(PW_3(w_3, m_3), PW_4(w_4, m_4)) > 0.98 \\ PW_1(w_1, m_1) \cdot PW_2(w_2, m_2) \cdot \min(PW_3(w_3, m_3), PW_4(w_4, m_4)) \cdot PW_5(w_5, m_5) > 0.92 \\ w_1 < 10, w_1 + w_2 < 15, \max(w_3, w_4) < 20, w_1 + w_2 + \max(w_3, w_4) + w_5 < 50 \\ 0 < m_1 \leq 50, 0 < m_2 \leq 100, 0 < m_3 \leq 50, 0 < m_4 \leq 10, 0 < m_5 \leq 20 \\ w_i > 0, i = 1, 2, 3, 4, 5 \end{cases}$$

(8)

4.2 基于粒子群算法的问题求解

粒子群优化 (particle swarm optimization, PSO) 算法<sup>[12]</sup> 是 Kennedy 和 Eberhart 受人工生命研究结果的启发, 通过模拟鸟群觅食过程中的迁徙和群聚行为而提出的一种基于群体智能的全局随机搜索算法. PSO 不像遗传算法那样对个体进行交叉、变异、选择等操作, 而是将群体中的个体看作是在  $D$  维搜索空间中的一个粒子, 每个粒子以一定的速度在解空间运动, 并向自身历史最佳位置 pbest 和邻域历史最佳位置 lbest 聚集, 实现对候选解的进化. PSO 算法不能直接应用于离散优化问题 (服务并行执行数目为整数值), 本文针对

所研究的问题, 在已有基本粒子群算法基础上, 提出了一种改进的粒子群算法 (DPSO), 该算法对粒子进行离散编码, 在粒子运动过程中引入粒子细微扰动、优化粒子飞行边界及粒子优胜劣汰等方法扩大搜索范围, 提高获得全局最优解的概率。

粒子群算法的本质是利用本身信息、局部较优信息和全局较优信息来指导粒子下一步迭代的位置。PSO 算法数学表示如下: 设搜索空间为  $D$  维, 总粒子数为  $n$ 。第  $i$  个粒子位置表示为向量  $X_i = (x_{i1}, x_{i2}, \dots, x_{iD})$ ; 第  $i$  个粒子“飞行”历史中的过去最优位置 (即该位置对应解最优) 为  $P_i = (p_{i1}, p_{i2}, \dots, p_{iD})$ , 如果第  $g$  个粒子为所有粒子的过去最优位置, 则  $P_g$  为所有  $P_i (i = 1, 2, \dots, n)$  中的最优; 第  $i$  个粒子的位置变化率 (速度) 为向量  $V_i = (v_{i1}, v_{i2}, \dots, v_{iD})$ 。每个粒子的位置按如下公式进行变化 (“飞行”):

$$v_{id}(t+1) = w \times v_{id}(t) + c_1 \times r_1 \times [p_{id}(t) - x_{id}(t)] + c_2 \times r_2 \times [p_{gd}(t) - x_{id}(t)]$$

(9)

$$x_{id}(t+1) = x_{id}(t) + v_{id}(t+1), 1 \leq d \leq D$$

(10)

其中  $c_1$  和  $c_2$  为学习因子, 称加速常数 (acceleration constant),  $r_1$  和  $r_2$  为  $[0, 1]$  范围内的均匀随机数。式 (9) 右边由三部分组成, 第一部分为“惯性 (inertia)”部分, 反映了粒子的运动“习惯 (habit)”, 代表粒子有维持自己先前速度的趋势; 第二部分为“认知 (cognition)”部分, 反映了粒子对自身历史经验的记忆 (memory), 代表粒子有向自身历史最佳位置逼近的趋势; 第三部分为“社会 (social)”部分, 反映了粒子间协同合作与知识共享的群体历史经验, 代表粒子有向群体或邻域历史最佳位置逼近的趋势<sup>[13]</sup>。 $w$  为调节因子,  $w$  较大适用于对解空间进行大范围探查,  $w$  较小适于进行小范围开挖。迭代中若位置和速度超过边界范围则取边界值。

基本粒子群优化算法容易陷入局部极值、早熟收敛或停滞现象<sup>[14]</sup>。本文对基本粒子群算法的改进如下:

1) 粒子离散编码: 粒子的维度与任务工作流中的任务相对应, 粒子在某一维度上的位置代表其对应任务中参与并行执行的服务数目。如本文的例子中编码格式如表 3 所示, 该粒子有 5 维, 每一维对应分配给一个任务的并行服务的数目, 表 3 所给出的某粒子的编码为 (50, 100, 50, 10, 20), 表示各任务中所有的可用服务都参与并行执行的情况。另外, 由于本文中粒子的位置只能取整数值, 因此对公式 (6) 计算的位置信息我们采用去尾取整的方式更新种群中相应粒子的位置。

表 3 粒子编码

任务节点	t1	t2	t3	t4	t5
任务节点点	1	2	3	4	5
服务并行执行数目	50	100	50	10	20
粒子编码	50	100	50	10	20

2) 粒子细微扰动: 公式 (9) 中  $P_{id}(t) - x_{id}(t)$  与  $P_{gd}(t) - x_{gd}(t)$  是反映粒子运动速度和方向的主要因素, 可以看出粒子在位置变化较快的维度上运动速度较快, 这虽然提高了种群收敛的速度却也容易造成局部最优, 因此, 本文借鉴文献 [15] 中通过调节加速常数正负号增加种群多样性的方法, 在  $P_{id}(t) - x_{id}(t) = 0$  或者  $P_{gd}(t) - x_{gd}(t) = 0$  的维度上对粒子的速度作细微扰动 (细微扰动既可防止因破坏个体极值和全局极值的优良信息, 而导致解的质量恶化, 又能保证种群运动的多样性) 以改善算法过早收敛问题, 针对本文中的例子, 由于问题求解规模较少, 细微扰动的具体做法是令  $d_d$  为  $(-1, 0, 1)$  中的一个随机值, 当  $P_{id}(t) = x_{id}$  时令  $P_{id}(t) - x_{id}(t) = d_d$ , 当  $P_{gd}(t) = x_{id}$  时令  $P_{gd}(t) - x_{id}(t) = d_d$ 。

3) 粒子飞行边界优化: 由于多阶段规划问题约束条件较多, 粒子的位置和速度容易超过边界范围, 但由于边界无规则, 超出边界的粒子可能重新飞入边界范围之内。本文对超越边界的粒子保留一定的运行次数 ( $R$ ), 以观察其是否能重新飞回边界范围之内。

4) 粒子优胜劣汰: 如果在一定的越界次数范围内粒子始终在边界之外则记录该粒子在边界之内的最优解, 并对该粒子进行淘汰, 同时为了保证种群范围不至于减少, 从总群中选取两个最好的粒子, 运用遗传算法中的交叉操作, 生成一个新的粒子, 加入到优化种群中。

虽然上述过程中后三步的操作会在一定程度上降低算法收敛的速度, 但却增加了粒子的搜索空间, 提高了获得最优解的概率。

DPSO 算法的实现过程如下:

**Step 1** 设置粒子最大越界运行次数 ( $R$ )。初始粒子总群  $x = (x_1, x_2, \dots, x_m)$  ( $m$  为种群规模), 计算粒子  $x_k$  的惩罚函数  $\min C(x_k)$ , 将该值设置为个体极值  $P_k$ , 各粒子的个体极值组成  $Pbest \leftarrow P_k$ , 全局极值

$Gbest \leftarrow \max Pbest;$

**Step 2** 基于式 (9)-(10) 和本文的细微扰动策略计算粒子的速度和位置;

**Step 3** 判断粒子越界次数是否超过  $R$ , 对小于  $R$  的粒子, 求得其位于边界范围内的最优解, 更新  $Pbest$  和  $Gbest$ . 对于大于  $R$  的粒子进行淘汰, 并从总群中选取两个最好的粒子, 运用交叉操作, 生成一个新的粒子;

**Step 4** 运行一定次数 ( $N$ ), 未发生改变转 Step 5, 否则转 Step 2;

**Step 5** 输出最优解, 算法结束.

## 5 实例计算与评估

### 5.1 DPSO 与 PSO 求解结果比较

粒子群初始化会对算法性能产生一定影响. 为使初始种群尽可能均匀覆盖整个搜索空间, 本文在表 3 给出的粒子基础上分别建立仅在某一个维度上进行适度调节的 15 个粒子作为初始种群. 基于第 4 节给出的求解方法, 令公式 (9) 中学习因子  $c_1 = c_2 = 1.5$ 、调节因子  $w = 0.6$ , 令算法中最大越界次数  $R = 3$ , 运行次数  $N = 50$ , 鉴于本例中问题规模较小令粒子的速度范围为  $[-2, +2]$ , 任务完成时间保留小数点后一位, 采用本文所述方法求得的最优解如图 6 中的 (a) 所示, 图 6 中的 (b) 为基于基本粒子群算法求得的最优解. 根据公式 (4) 可知, 基于 DPSO 算法求得的目标函数值为 6.988, 基于基本 PSO 算法求得的目标函数值为 7.104, 显然 DPSO 算法求得的结果优于基本 PSO 算法.

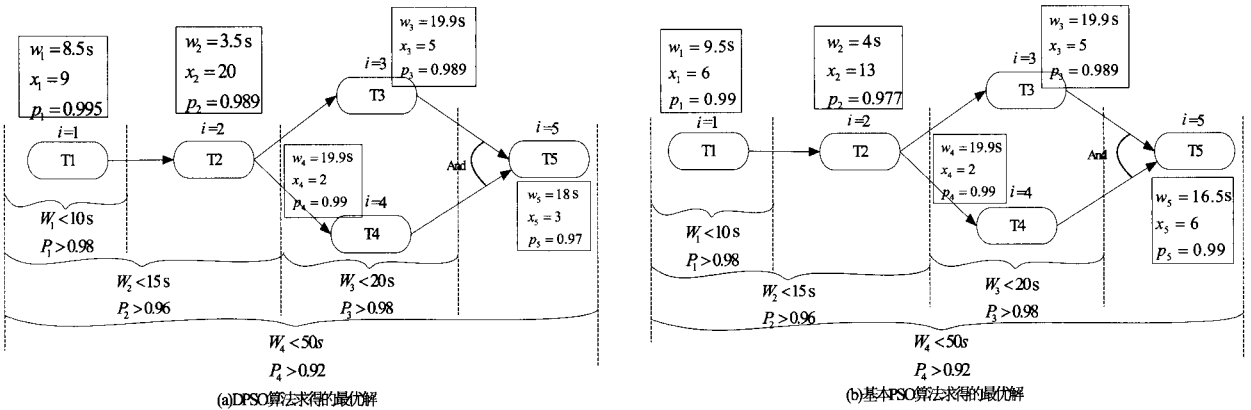


图 6 案例求解结果

本文对图 5 中任务 workflows 的时间约束进行修改, 对各阶段任务的可用时间分别以 10% 递增, 分别采用 DPSO 算法和基本 PSO 算法求得的最优解如图 7 所示, 可以看出 DPSO 算法的解始终优于基本 PSO 算法, 图 7 还表明尽管目标函数中没有时间变量, 但是通过改变任务完成时间仍然会对目标函数的求解结果产生影响, 这是因为在保证一定的任务成功率前提下, 任务完成时间的改变会影响服务并行执行的数目.

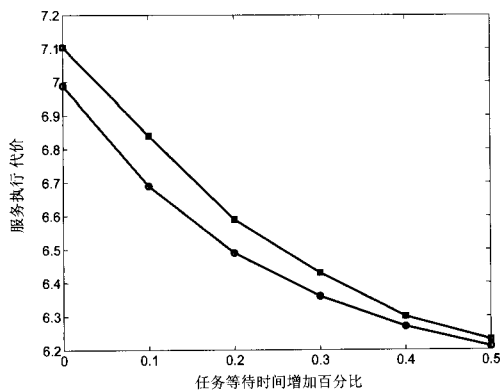


图 7 增加任务完成时间对目标函数最优解的影响

为了进一步分析在更加复杂情况下 DPSO 算法的求解能力, 实验中对图 5 的任务 workflow 以串并联的方式增加任务的数目, 并对任务对应的并行执行的服务数目、任务完成时间及成功概率等进行设置, 仿真计算中发现随着任务的增加 (workflow 复杂性增大), DPSO 算法的解质量改进非常好, 虽然由于种群中粒子的细微扰动和进化, 会花费稍长的计算时间, 但 DPSO 算法基本上可以得到全局最优解。

5.2 不同服务分配策略的比较

为简化问题, 本文以表 1 给出的数据集为例, 只针对单任务情况分别采用服务并行执行、选择最优服务、服务失效替换三种方法, 对比分析随着时间的增加, 任务完成成功概率的情况, 其中服务失效替换方法采取的策略是在服务资源列表中随机选择一个服务, 当等待时间超过服务执行时间分布的均值加 3 倍  $\delta$  时 ( $3\delta$  法则) 认为该服务调用失败, 从服务资源列表中选择新的服务进行替换; 服务并行执行方法中服务并行执行的数目设置为 10, 得到的结果如图 8 所示, 图 9 为任务完成的代价。

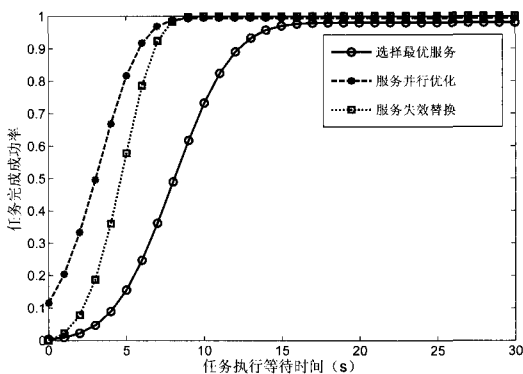


图 8 并行执行服务数目与任务成功率之间关系

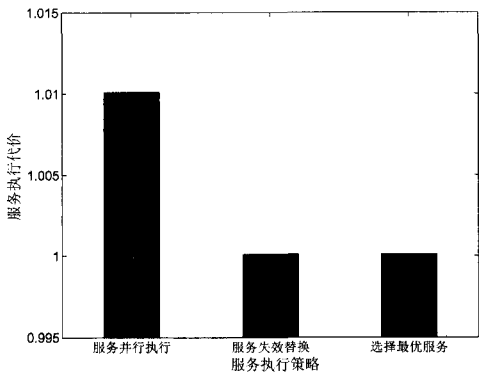


图 9 任务执行代价

从图 8 中可以看出本文提出的服务并行执行方法在保证任务完成的成功率和缩短任务完成时间方面都要优于其它两种方法, 图 9 表明服务并行执行方法也会带来更多的系统资源开销. 因此, 在实际应用中需要结合具体的应用需求采用合适的方法。

6 结论

提高服务组合的时效性和成功率是服务组合优化要解决的关键问题, 对于军事应用而言具有重要的现实意义. 本文从服务资源并行分配的角度研究了服务组合的优化方法, 建立了问题求解的数学模型, 并针对具体问题通过改进已有的基本粒子群算法对问题进行求解. 下一步, 将结合本文给出的服务资源分配系统框架研究系统的实现方法, 并对多任务情况下的服务资源并行优化方法展开研究。

参考文献

[1] Shook R. Net-enabled command capability and joint combat capability enveloper overview brief[R]. US Joint Forces Command, 29 Aug 2007.

[2] Konig D, Lonhmsnn N, Moser S, et al. Extending the compatibility notion for abstract WS-BPEL processes[C]// Proceedings of the International World Wide Web Conference, Beijing, China: WWW'08, 2008: 785-794.

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

[4] 冯名正. Web 服务组合关键技术研究 [D]. 南京: 东南大学, 2006.

Feng M Z. Research on the key technologies of Web services composition[D]. Nanjing: Southeast University, 2006.

[5] Werner C, Buschmann C. Compressing SOAP messages by using differential encoding [C]// IEEE International Conference on Web Services, California: ICWS'04, 2004: 540-547.

[6] Nicolas S, Charles F J. Fault tolerance collectors for unreliable Web Services[C]// 37th Annual IEEE / IFIP International Conference on Dependable Systems and Networks, Edinburgh: DSN'07, 2007: 51-60.



- [7] Hamcy J, Doshi P. Speeding up adaption of web service composition using expiration times[C]// International World Wide Web Conference, Canada: WWW'07, 2007: 1023–1032.
- [8] Zheng H Y, Zhao W L, Yang J. QoS analysis for web service composition[C]// 2009 IEEE International Conference on Services Computing, Bangalore: SCC'09, 2009: 235–242.
- [9] Zeng L Z, Benatallah B, Ngu A H H, et al. QoS-aware middle ware for web services composition[J]. IEEE Trans Softw Eng, 2004, 30(5): 311–327.
- [10] Aron M, Drushnel P, Zwaenepoel W. Cluster reserves: A mechanism for resource management in cluster-based network servers[C]// Proceeding of the Int'l Conf on Measurement and Modelling of Computer Systems (SIG-METRICS 2000), Santa Clara: ACM, 2000: 90–101.
- [11] 伍之昂, 罗军舟, 宋爱波, 等. 具有 QoS 保证的服务资源联合分配与管理 [J]. 软件学报, 2009, 20(12): 3150–3162.  
Wu Z A, Luo J Z, Song A B, et al. QoS guaranteed service resource co-allocation and management[J]. Journal of Software, 2009, 20(12): 3150–3162.
- [12] Kennedy J, Eberhart R. Particle swarm optimization[C]// Proceedings of the 4th IEEE International Conference on Neural Networks, Piscataway: IEEE Service Center, 1995: 1942–1948.
- [13] Ling S H, Iu H H C, Leung F H F. Improved hybrid particle swarm optimized wavelet neural network for modeling the development of fluid dispensing for electronic packaging[J]. IEEE Trans Ind Electron, 2008, 55(9): 3447–3460.
- [14] 戴朝华. 粒子群优化算法综述 [EB/OL]. <http://www.sciencenet.cn/upload/blog/file/2010/2/20102792556213369.pdf>.
- [15] Langdon W B, Riccardo P. Evolving problems to learn about particle swarm and other optimizers[C]// Proc CEC-2005, 2005, 1: 81–88.