文章编号:1671-4598(2024)02-0036-06 DOI:10.16526/j.cnki.11-4762/tp.2024.02.006 中图分类号:TP15 文献标识码:A

基于混合人工蜂群算法的并行测试任务优化研究

毛志宾[1]，任慧敏[2]，鲁承金[3]，沈海阔[1]
(1. 北京交通大学 机械与电子控制工程学院，北京 100044；
2. 北京航天自动控制研究所，北京 100854；
3. 北京航天万源科技有限公司，北京 100176)

摘要：并行测试技术可以同时进行多个任务的测试，提高资源利用率，节约测试成本；并行测试调度问题是一种复杂的组合优化问题，是并行测试技术的核心要素；并行测试系统作为并行测试技术的载体，自身的性能和求解效率尤其重要；文章对并行测试完成时间极限定理进行了研究，建立了并行测试任务调度的数学模型，分析了传统元启发式算法求解并行测试问题的不足，提出了基于动态规划的递归搜索技术和人工蜂群算法相结合的混合人工蜂群算法，并采用整数规划精确算法和遗传算法对混合人工蜂群算法进行验证；得出结论采用混合人工蜂群算法进行并行测试任务的调度节约了接近50%的时间，降低了约20%的硬件资源占用，提高了测试效率，可以满足工程实际的应用。

关键词：并行测试；任务调度；人工蜂群算法；时序递归搜索；测试效率

Research on Parallel Test Task Optimization Based on Hybrid Artificial Bee Colony Algorithm

MAO Zhibin[1], REN Huimin[2], LU Chengjin[3], SHEN Haikuo[1]
(1. School of Mechanical and Electronic Control Engineering, Beijing Jiaotong University, Beijing 100044, China;
2. Beijing Aerospace Automatic Control Institute, Beijing 100854, China;
3. Beijing Aerospace Wanyuan Technology Co., Ltd., Beijing 100176, China)

Abstract: Parallel testing technology can perform testing on multiple tasks simultaneously, improve resource utilization, and save testing cost; Parallel test scheduling problem is a complex combinatorial optimization problem, and it is the core element of parallel test technology; As the carrier of parallel testing technology, it is particularly important for the performance and solving efficiency of parallel testing systems; The limit theorem of parallel test completion time is studied, the mathematical model of parallel test task scheduling is established, and the shortcomings of traditional meta heuristic algorithms are analyzed to solve parallel test problems. A hybrid artificial bee colony algorithm based on the combination of the recursive search technology of dynamic programming and artificial bee colony algorithm is proposed, and the precise algorithm of integer programming and genetic algorithm verify the hybrid artificial bee colony algorithm; The conclusion is that the scheduling parallel testing tasks saves nearly 50% of time by using hybrid artificial bee colony algorithm, it reduces about 20% of hardware resource usage, and improves testing efficiency, the algorithm can meet practical engineering applications.

Keywords: parallel testing; task scheduling; artificial bee colony; temporal recursive search; test efficiency

0 引言

在自动测试领域，传统的测试方法是对被测单元(UUT，unit under test)进行串行测试，比如在主线程中直接调用测试函数执行测试任务，测试系统往往是单线程的，不能同时进行多项任务的测试。虽然这种方式对于管理测试任务比较方便，但是测试的效率比较低，不能满足当下测试任务的需求[1]。

串行测试任务中往往高达80%的时间处在空闲时间[2]，而并行测试技术则可以很好地解决这些问题。它可以通过将测试任务分解为多个被测单元子任务，利用多个处理器、多核心或者多台计算机同时执行这些子任务，来提高测试效率和测试覆盖率。并行测试技术的关键在于对并行测试调度任务的建模，找到最优的调度方案[3]。

针对并行测试任务，需要根据实际问题采用特定的方法。进行并行测试的目标可以是单一的，也可以是多个的。梁旭[4]等对并行测试任务进行工序建模，以完工时间作为

收稿日期:2023-03-29； **修回日期**:2023-05-03。
作者简介:毛志宾(2000-)，男，硕士研究生。
沈海阔(1979-)，男，博士，教授。
引用格式:毛志宾，任慧敏，鲁承金，等.基于混合人工蜂群算法的并行测试任务优化研究[J].计算机测量与控制，2024，32(2)：36-41，49.

目标函数设计遗传算法实现了对测试任务和资源的并行调度。

缩短测试完成时间很多情况可以从并行测试时间极限定理出发。王正元[5]等基于并行测试时间极限定理设计算法实现对并行测试任务的调度，确定测试完成最短时间。姚静波[6]等以测试任务的资源约束为出发点，同时深入研究了并行测试时间极限定理，对并行测试模型进行改进，设计出一种基于测试资源数量约束的并行任务调度改进算法，但是没有考虑到测试任务之间的时序约束，虽然缩短了测试时间，但是却增加了测试的成本。李恒璐[7]和宋春霞[8]则不以时间最短为目的求解任务调度序列。

在进行并行测试任务调度时，不仅要考虑到任务之间的时序约束，还要考虑到资源约束。夏锐[9]等将并行测试任务抽象为数学模型，提出了并行率的概念，设计了一种满足资源约束和任务时序约束的混合遗传退火算法。Yang[10]等针对传统测试任务调度问题（TTSP，test task scheduling problem）的不足，提出了一种求解任务调度问题的混合整数线性规划模型（MILP），通过隐含序列寻找过程（ISF）和基于序列的迭代优化（SBIO）过程来减少计算时间。

在进行并行测试任务调度时经常用到元启发式算法，但是这些算法容易陷入局部最优，导致不能得到期望的结果。为此 Jain[11]等使用混合整数线性规划模型来解决调度问题，将资源约束和测试任务的排序结合起来获得全局最优解；Lu[12]等设计集成编码方案，以遗传算法为基础加入混沌映射算子，避免了陷入局部最优，获得高质量的解。Guan[13]等提出了一种基于信息熵的蚁群优化算法，提高了蚁群算法求解约束满足问题（CSP）的解质量；Shi[14]设计了适合于 TTSP 问题的方案选择规则，并将其与遗传算法相结合，用于求解最优测试序列。刘正雷[15]将 petri 网和蚁群算法相结合，避免算法陷入局部最优解。陆晓飞[16]对粒子群算法进行改进，在粒子群迭代前期或者没有取得期望的最优解时，粒子进行全局开拓；在粒子群后期或者已经取得期望的最优解时，寻找邻域内适应值最小的点，从而以更快的速度跳出局部最优解并提高求解的精度。秦勇[17]等重新设计编码方式，将贪婪算法与遗传算法相结合，提高了种群质量，但是该方法只能用来求解任务间无前后约束关系的并行测试调度问题。卢茜[18]等在遗传算法中引入模拟退火算法和禁忌搜索算法的思想，避免了遗传算法过早收敛的问题，但是仅考虑了多个被测设备之间的并行测试任务调度，而不能实现一个被测设备的各个测试任务之间的并行执行。

并行测试任务的调度方法往往与所研究的实际问题密切相关。苏萍贞[19]使用任务调度引擎，基于异步调用的方式实现对测试任务的多线程异步执行，但是比较依赖.NET 框架，适用范围比较单一。付新华[20]等对蚁群算法进行改进，使算法的使用范围由一维静态优化问题扩展到多维动态组合优化问题，解决了并行测试任务调度复杂、难以优化的问题。谈恩民[21]等使用遗传算法对多 IP 核进行并行测试，相较于传统 Soc 测试，在功耗约束的情况下缩短了测试时间。

对于本文所要优化的问题而言，各个测试任务均有各自的资源依赖关系。所以，需要设计有效的算法既要保证各测试任务的时序关系，还要保证其在资源约束上没有冲突。国内外学者对此类问题研究较少，即使有前人对类似问题提出过可行性方案，针对本文所研究的问题也不能直接使用。为此，本文基于所研究的调度任务，设计每个任务的编码方式以及各测试任务之间的时序约束和资源依赖关系，通过将基于动态规划的递归搜索方法与人工蜂群算法结合，提出了混合人工蜂群算法，以实现在保证各测试任务在满足其对应的资源依赖关系和时序约束关系的基础上最终测试完工时间最小化。

1　并行测试任务调度问题描述

1.1　相关概念介绍

1）并行度：并行度是指在并行计算中同时执行的任务或线程的数量。在并行计算中，可以将一个任务分解成多个子任务并在多个处理器或计算节点上并行执行。并行度是用于衡量并行计算系统中的任务并行性能的一个指标。如果并行度越高，则意味着系统能够同时处理更多的任务或线程，从而提高计算性能。并行度通常可以通过以下公式计算：

$$\eta = \frac{T_1}{T_1 + T_2} \tag{1}$$

式中，η 为并行度，T_1 为执行时间，T_2 为等待时间。执行时间是指计算任务实际运行的时间，等待时间是指任务等待资源或其他线程完成的时间。在测试任务确定后，并行度越高，测试完成时间越少。

2）时序约束：设 ra，rb 是两个测试任务，并且它们的测试顺序在测试任务中不能改变，那么就认为它们存在时序约束。

3）串行任务链：在并行测试系统中，由于不同测试任务之间存在的时序相关性或资源相关性，以及在单个 UUT 上进行多个测试任务时，UUT 存在的唯一性等原因，导致这些测试任务只能串行测试，将这些任务组成的集合称为测试资源串行任务链。假设存在测试任务 A，测试任务 B 和测试任务 C，其中 B 和 C 都依赖于 A 的输出。在这种情况下，我们需要按照 $A \rightarrow B \rightarrow C$ 的顺序依次执行这些任务，以确保 B 和 C 可以使用 A 的输出进行测试。如果我们将这些任务并行执行，可能会导致测试结果不准确或出现错误。因此，必须按照正确的顺序执行串行任务链，以确保测试结果的准确性和可靠性。

4）并行测试完成时间极限定理：在并行测试中，测试任务的完成时间受限于最慢的测试任务，即使其他任务已经完成，整个测试任务也必须等待最慢的任务完成。这是由于并行测试的结果取决于所有测试任务的完成情况，而最终结果必须等待所有任务完成后才能确定。

1.2 组合优化问题描述

并行测试任务调度通常被归类为 TTSP。因此，一个测试项目的并行测试任务优化需求可以被抽象描述为，m 个测试任务需要被分配到 n 个资源上执行，并且部分测试任务之间由于既定工程约束需要满足一定的顺序关系。此外，每个测试任务的执行需依赖一个或多个资源才能完成，且每个资源上不允许出现测试任务间的抢占冲突。图 1 为一个并行测试任务调度结果的可视化展示。

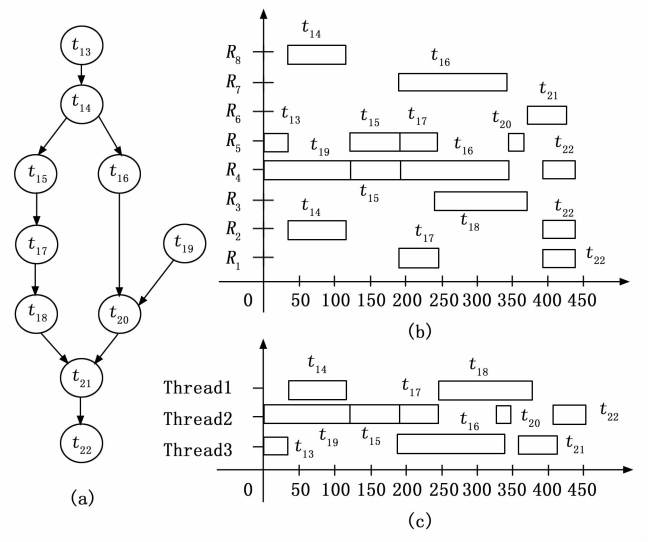

图 1 并行测试任务调度典型实例

图 1 为 10 个测试任务在 8 个资源上的测试实例，其中允许最大并发任务数为 3。图 1（a）展示了测试任务顺序拓扑关系，其中从测试任务 t_{13} 到测试任务 t_{23} 均有既定测试的时序关系。图 1（b）为各测试任务与依赖资源的调度顺序甘特图，显然任意两个任务在所需相关资源上无抢占冲突。图 1（c）为在限制最大任务并发数量为 3 的约束下，所有测试任务在多个线程上分布式执行的顺序安排。此外，该分布式执行顺序与所依赖资源的调度安排的时刻相对应。

由图 1 的实例可知，既要考虑各个测试任务在依赖资源上的抢占冲突，又要满足既定资源之间的执行顺序与最大并发任务数，显然并行测试任务的优化场景是极其复杂的。该问题特性使得相应优化技术需采用启发或元启发算法框架，传统的精确求解技术无法在可接受时间内得到最优解。

1.3 并行测试任务数学模型描述改进

对于并行测试任务，首先定义测试任务集 $\boldsymbol{T}=\{t_1, t_2, t_3, \ldots, t_m\}$ 和测试资源集 $\boldsymbol{R}=\{r_1, r_2, r_3, \ldots, r_n\}$，集合 $\tau=\{\tau_1, \tau_2, \tau_3, \ldots, \tau_m\}$ 对应于所有测试任务的时间消耗，$\boldsymbol{TR}$ 是一个 $m\times n$ 维的矩阵，表示测试任务和资源之间的依赖关系。矩阵 $\boldsymbol{TS}$ 表示预定的技术测试顺序矩阵，与实际问题需要满足的约束有关。为了实现并行测试，可以通过预定的线程数量来同时处理不同的测试任务，所有满足 $\boldsymbol{TS}$ 限制的时间序列都可以被认为是可行的解决方案。

为满足并行测试任务的需求，并结合本文所研究实际问题特点的抽象，并行测试任务调度模型进行如下改进：

1）任何资源最多可以同时由一个测试任务占用；

2）占用资源的时间消耗等于相关测试任务的时间消耗；

3）所有测试任务的排序必须满足预定的技术测试顺序（$\boldsymbol{TS}$）；

4）所有资源上同时的测试任务数量必须小于预定的线程数量。这个问题的目标是尽量减少上次测试任务的完成时间。

资源任务集由 $\boldsymbol{TR}$ 的列向量中对应元素为 1 的任务组成，这些任务之间彼此互斥；并行任务序列用来约束任务间的时序关系。

1.4 优化问题数学建模

为建立一个标准的整数规划模型，以下内容首先定义了相关的决策变量。令二值变量 $z_{ij}\in\{0, 1\}$ 表示任务 t_i 与任务 t_j 之间的排布关系，其中 $z_{ij}=1$ 表示任务 t_j 排布在任务 t_i 之后（可以不连续）；反之表示任务 t_j 排布在任务 t_i 之前。令非负整型变量 s_i 表示任务 t_i 的最早开始测试时间，因此任务 t_i 完成测试的时间为 $s_i+\tau_i$。非负整型变量 $T_{\max}$ 表示整个并行测试系统的完工时间。此外，M 表示为一个极大的正整数，在数学模型中对非关键约束起到虚化的作用。

该问题的混合整数规划线性模型建立如下：

$$\text{Minimize } T_{\max} \tag{2}$$

Subject to

$$s_i+\tau_i\leqslant s_j+\tau_j \tag{3}$$

$$s_j-s_i\geqslant\tau_i-M(1-z_{ij}) \tag{4}$$

$$T_{\max}\geqslant s_i+\tau_i \quad i=1,\cdots,m \tag{5}$$

$$s_i\geqslant 0 \quad i=1,\cdots,m \tag{6}$$

$$z_{ij}\in\{0,1\} \quad i=1,\cdots,m, j=1,\cdots,m, i\neq j \tag{7}$$

$$T_{\max}\geqslant 0 \tag{8}$$

公式（2）为模型的目标函数，即最小化整个系统的总测试时间。剩余公式定义了本模型的约束和决策变量。约束（3）为各测试任务完工时间的时序约束。当 $\boldsymbol{TS}(i, j)=1$ 时，约束 1 转化为 $s_i+\tau_i\leqslant s_j+\tau_j$，即要求任务 t_j 的结束时间大于任务 t_i；反之，模型中没有任务 t_j 与任务 t_i 的结束约束。约束（4）是保证多任务占用资源的非冲突约束。约束（6）～（8）为相应的决策变量定义。

2 混合人工蜂群算法设计

为解决优化场景中存在既定的测试任务顺序约束，本文主要基于动态规划的递归搜索以及人工蜂群算法框架开发出了混合人工蜂群算法。通过递归搜索技术找到一系列任务隐含序列，然后使用找到的隐含序列纠正人工蜂群算法的单个编码。

2.1 算法总体设计

混合人工蜂群算法首先通过时序递归搜索找到一系列任务隐含序列，然后依次执行“全局个体优化”“部分个体强化”和“少量个体淘汰”3 个主搜索流程，纠正找到的隐

含序列混合人工蜂群算法的单个编码。在执行“全局个体优化”和“少量个体强化”流程时，均会依次执行二进制选择、邻域搜索模块、隐集编码修正，如果随机搜索的概率满足局部搜索条件，还会执行局部搜索模块以及再一次的隐集编码修正。

不同的是，在“全局个体优化”搜索流程中，每次进入邻域搜索模块的是一个顺序选择个体和一个经过二进制选择的个体；而进入“部分个体强化”搜索流程中邻域搜索模块两个个体均为经过二进制选择的个体。当执行完“全局个体优化”和“部分个体强化”搜索流程后，则会在“少量个体淘汰”搜索流程阶段对超过限定迭代次数而未优化的个体进行淘汰。最后，经过一次迭代之后更新全局最优个体记录。混合人工蜂群算法的基本流程如图 2 所示。

当在既定的优化场景中通过混合人工蜂群算法的数据流输入接口传入原始的数据之后，就会将其存储在混合人工蜂群算法的变量中，通过调用混合人工蜂群算法内部的函数对相应的原始数据进行一定的操作，最后通过混合人工蜂群算法的数据流输出接口将处理后的数据返还给外部。

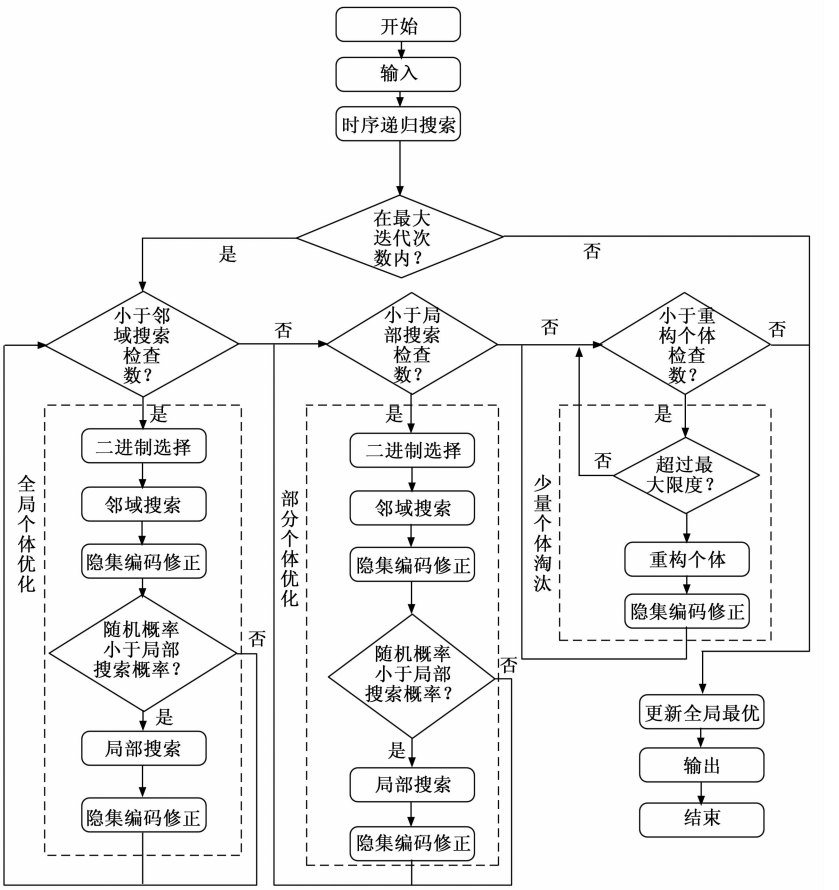

图 2　算法流程图

2.2　时序递归搜索技术

递归搜索技术是为了找到一系列任务隐含序列，这些序列不违反属于每个 UUT 的任务的预定测试顺序，然后使用找到的隐含序列纠正 ABC 算法的单个编码。让序列 U_0^w 表示包含受限制任务 w 的 UUT 的序列，$\boldsymbol{Z}^w$ 是包含输出子序列 $\boldsymbol{Z}_p^w \subset \boldsymbol{Z}^w$ 的矩阵。此外，$\boldsymbol{Q}^w = \boldsymbol{U}_0^w$ 是一个标准序列。标签设置（LS）算法是一个典型的递归过程，$l_i =（i，Z_i^w，U_i^w）$ 表示 LS 中的标签，其中 i 是任务编码，Z_i^w 是在任务 i 结束的分区计划，U_i^w 是任务集，可在节点 i 扩展。假设 $l_j =（j，Z_j^w w，U_j^w）$ 是 l_j 的后继标签，因此可以根据扩展方程对 l_j 进行扩展。

$$j = u_0 \in U_i^w, U_j^w = U_i^w \setminus \{j\},$$
$$Z_j^w = Z_i^w \cup \{j\} \tag{9}$$

图 3 显示了示例的递归路径。递归树依据违反限制的节点路径被剪枝，有效地降低算法的搜索空间，提高算法的搜索效率。图 2 最终的任务隐含序列为（1，2，3，4），（1，3，2，4）以及（1，3，4，2）。然后人工蜂群编码序列通过对与其隐含及序列相关的局部任务排序调整，即可实现该编码序列满足既定时序约束，结果如图 4 所示。

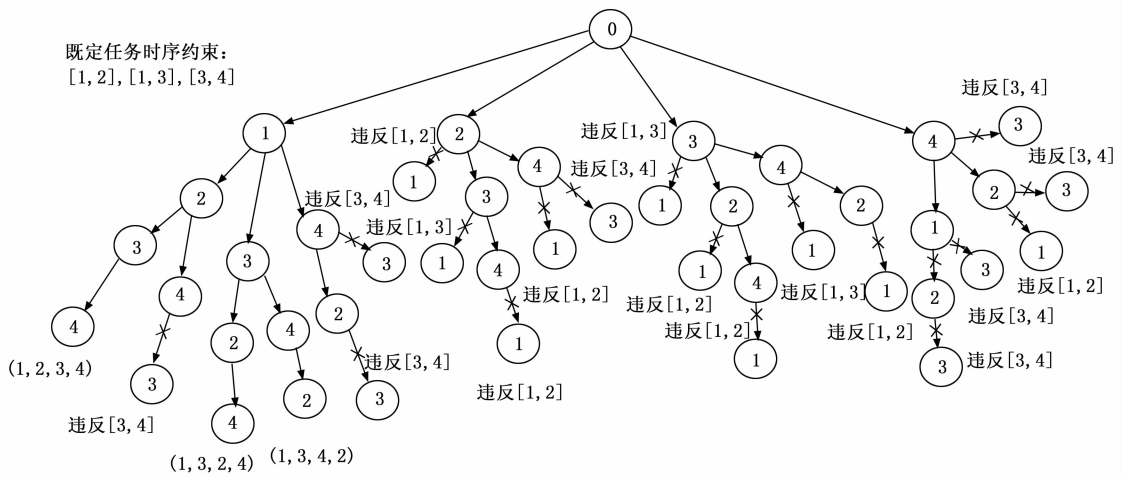

图 3　时序约束的递归路径

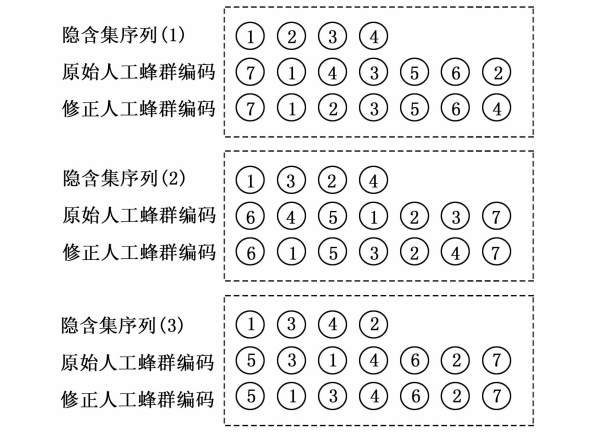

图 4 时序约束 $TS=\{[1,2],[1,3],[3,4]\}$ 隐含集序列对应的修正人工蜂群编码

2.3 邻域搜索技术

邻域搜索技术（Neighborhood Search）结合了多点插入算子与多点交换算子，该技术旨在对一个体编码及其邻域编码实现高质量的子代编码搜索。假设 X_t 是一个按照顺序选择的个体，X_f 为通过二进制选择的一个个体，即为 X_t 的邻域个体。邻域搜索流程主要依据概率 *nbrPro* 执行，如果随机概率大于 *nbrPro*，则只对 X_t 执行多点交换算子。如果随机概率小于 *nbrPro*，则首先判断两个体的适应度值；如果适应度值相同，仍然只对 X_t 执行多点交换算子；否则，对两个体执行多点交换算子获得子代编码。

多点插入算子（Multi－insertion）：基于既定的交换点数量与两个个体，通过给定的规则生成子个体。假设 X_t 是一个按照顺序选择的个体，X_f 为通过二进制选择的一个个体，即为 Xt 的邻域个体。*nbrNum* 为既定交换的个体中的节点数量。例如，X_t 和 X_f 可以分别被表示为：

$X_t=$（1，5，3，2，9，8，10，7，4，6）和 $X_f=$（2，6，5，9，3，1，7，8，4，10）。在 X_t 中随机选择 *nbrNum*（通常 *nbrNum*）个保留点位，则包含保留点位的 X_t 可以被表示为：

$$X_t=(x,5,x,2,9,x,x,x,x,x) \tag{10}$$

其中：x 表示 X_t 的空白位置。随后，从 X_f 中抽取非 X_t 中保留点位的其他数值并依次填充入 X_t。最终的 X_t 可以被重新生成为 $X_t=$（6，5，3，2，9，1，7，8，4，10）。上述过程是多点插入算子的标准流程，多点插入算子擅长于在调度问题中更大几率的发现质量更高的解。

多点交换算子（Multi－swap）：针对既定的一个编码，通过多个点的位置交换，从而产生一个新个体编码。多点交换算子依据一个随机产生的交换点数量 *nrb*，然后随机交换 *nrb* 个节点的位置，从而产生新编码。该算子旨在通过多点交换方式，从而避免算法陷入局部最优的情况。

2.4 局部搜索技术

局部搜索技术（Local Search）结合了贪婪搜索过程，通过局部枚举的方式尽可能地实现高质量解编码的生成。局部搜索算子通过一个弱枚举的循环，最大限度地提升个体解的质量。对于个体编码 $X_t=(x_1, x_2, \cdots, x_i, \cdots, x_m)$，将第一个任务编号 x_1 与相邻位置进行交换；如果 x_1 交换后无法提高 X_t 的质量（适应度值），则重复上述过程；如果 x_1 交换后提高了解 X_t 的质量（适应度值），则终止循环返回当前生成的新个体。局部搜索算子的执行效率较低，但可以配合弥补其他算子收敛性不强的不足。

3 实验验证分析

本文采用整数规划精确算法和遗传算法作为混合人工蜂群算法的对比算法，分别通过小规模用例和大规模用例进行算法的性能测试对比。小规模用例用来测试算法处理一般测试任务调度的能力，大规模用例用来探究算法的极限性能。以测试任务完成时间和 cpu 占用率作为评价算法的指标。

3.1 小规模用例测试

小规模测试用例 1 如表 1 所示，规模：测试任务数为 8，资源数为 7。

表 1 小规模测试用例 1

$TR(i/j)$	r_1	r_2	r_3	r_4	r_5	r_6	r_7	τ_i
t_1	1	1	0	0	0	0	0	5
t_2	1	0	1	0	0	0	0	12
t_3	0	0	0	1	0	0	0	3
t_4	1	0	0	1	0	0	0	2
t_5	0	0	1	0	1	0	0	20
t_6	1	0	0	0	0	1	0	4
t_7	0	0	0	0	0	0	1	40
t_8	0	0	1	0	0	0	0	15

表 1 的算例分别调用了整数规划精确算法、遗传算法以及混合人工蜂群算法求解。3 个算法都可以求得该算例的最优解，并且它们的求解效率差别不大。3 个算法的求解结果如图 5 所示。

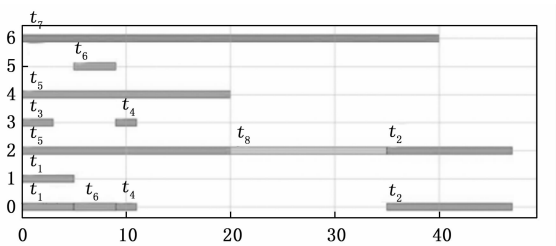

图 5 小规模测试用例 1 结果甘特图

3 个算法的求解时间和 CPU 占用率如表 2 所示。

表 2 小规模测试用例 1 结果比较

算法名称	求解时间/s	CPU 占用率/%
遗传算法	1.5	50
整数规划精确算法	0.8	60
混合人工蜂群算法	0.7	45

从表 2 可以看出，3 个算法的求解时间接近，混合人工

蜂群算法的 CPU 占用率更低。

小规模测试用例 2 如表 3 所示，规模：测试任务数为 15，资源数为 5。

表 3　小规模测试用例 2

$TR(i/j)$	r_1	r_2	r_3	r_4	r_5	τ_i
t_1	1	1	0	1	1	31
t_2	1	1	1	1	0	45
t_3	0	1	1	1	1	46
t_4	0	1	1	0	1	37
t_5	1	1	1	0	0	60
t_6	0	0	1	1	1	97
t_7	1	1	0	1	0	15
t_8	0	0	1	1	1	17
t_9	0	0	1	0	1	12
t_{10}	1	0	0	1	0	42
t_{11}	0	0	1	1	1	37
t_{12}	0	0	1	0	0	34
t_{13}	0	0	0	1	1	54
t_{14}	1	1	0	0	0	75
t_{15}	1	1	1	1	0	22

表 3 的测试用例同样分别调用了整数规划精确算法、遗传算法以及混合人工蜂群算法求解。3 个算法都求解到了该算例的最优解，即最优完工时间为 $T_{\max}=412$。其最优解的甘特图如图 6 所示。

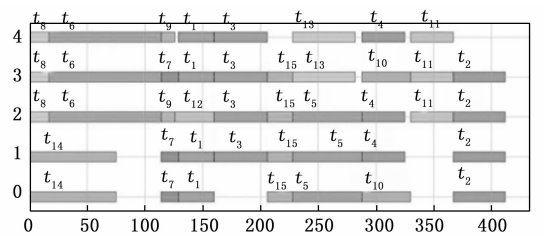

图 6　小规模测试用例 2 结果甘特图（有时序约束）

在小规模用例 2 中 3 个算法的求解时间和 CPU 占用率如表 4 所示。

表 4　小规模测试用例 2 结果比较(有时序约束)

算法名称	求解时间/s	CPU 占用率/%
遗传算法	13	75
整数规划精确算法	9	67
混合人工蜂群算法	5	52

图 7 表示在无时序约束的情况下，3 个算法的最优解甘特图。

表 5 为无时序约束下 3 个算法结果对比。

表 5　小规模测试用例 2 结果比较(无时序约束)

算法名称	求解时间/s	CPU 占用率/%
遗传算法	230	83
整数规划精确算法	123	71
混合人工蜂群算法	9	59

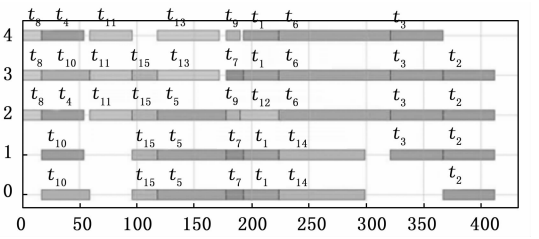

图 7　小规模测试用例 2 结果甘特图（无时序约束）

混合人工蜂群算法可以在小于 10 s 内完成，而整数规划精确算法的求解时间约为 123 s，遗传算法的求解时间为 230 s。这表明在没有时序约束的情况下，混合人工蜂群算法的性能要远远优于传统并行测试算法。

3.2　大规模用例测试

大规模算例：测试任务数为 100，资源数为 10。测试结果如表 6 所示。

表 6　小规模测试用例 2 结果比较(无时序约束)

算法名称	求解时间/s	CPU 占用率/%
遗传算法	——	91
整数规划精确算法	——	79
混合人工蜂群算法	368	65

当测试规模扩大到一定程度，整数规划精确算法以及遗传算法已经不能完成任务，无法求得最优解。而混合人工蜂群算法依然可以在 400 多秒内搜索到最小完成时间。这表明，相比传统的启发式算法，混合人工蜂群算法在大规模测试任务下有很大的优势。

4　结束语

本文基于所研究的并行测试任务，在确保各任务间的时序约束关系和资源依赖关系的基础上，将动态规划的递归搜索方法与人工蜂群算法相结合，提出混合人工蜂群算法。分别用小规模用例和大规模用例进行测试，并与整数规划精确算法和遗传算法进行对比。结果表明，本文所提出的方法相较于传统求解器节省时间接近 50%，硬件资源的占用率降低了接近 20%，提高了求解该类问题的效率。

参考文献：

[1] ANDERSON JR J L. High performance missile testing [C] // AUTOTESTCON 2003. IEEE Systems Readiness Technology Conference, IEEE, 2003: 19 - 27.

[2] 肖明清，朱小平，夏　锐. 并行测试技术综述 [J]. 空军工程大学学报（自然科学版），2005，6 (3)：22 - 25.

[3] 肖明清，付新华. 并行测试技术及应用 [M]. 北京：国防工业出版社，2010.

[4] 梁　旭，李行善，于劲松. 基于遗传算法的并行测试调度算法研究 [J]. 电子测量与仪器学报，2009，23 (2)：19 - 24.

[5] 王正元，刘卫东，景慧丽，等. 一种并行测试任务调度优化方法 [J]. 兵工学报，2018，39 (2)：399 - 404.

（下转第 49 页）