文章编号:1001-506X(2012)05-0947-08

基于 DLS 和 GA 的作战任务-平台资源匹配方法

张杰勇,姚佩阳,周翔翔,王 欣
(空军工程大学电讯工程学院,陕西 西安 710077)

摘 要:作战任务和平台资源的合理匹配是战役作战准备阶段的主要内容。考虑平台资源能力在作战过程中的损耗,在问题建模的过程中引入了资源能力的损耗系数,使得所建模型更加符合实际作战。提出了基于动态列表调度(dynamic list scheduling, DLS)和遗传算法(genetic algorithm, GA)的模型求解方法,使用 DLS 选择处理的任务,使用 GA 为选定任务分配平台资源,给出了该方法具体的设计思路和流程。最后结合联合作战的战役算例,验证了所提方法的优越性和适用性。

关键词:运筹学;任务-平台匹配;损耗系数;动态列表调度;遗传算法
中图分类号:TP 391.9 **文献标志码**:A **DOI**:10.3969/j.issn.1001-506X.2012.05.17

Approach to operation task and platform resource matching based on DLS and GA

ZHANG Jie-yong, YAO Pei-yang, ZHOU Xiang-xiang, WANG Xin
(*Telecommunication Engineering Institute, Air Force Engineering University, Xi'an 710077, China*)

Abstract: The match between operation tasks and platform resources is the main content in the preparation phase of battle. In order to consider the loss of the platform capacity in the process of combat, the loss coefficient is introduced in the process of problem modeling, and the problem model can be more conformable with actual combat. An approach to the problem model based on the dynamic list scheduling (DLS) and genetic algorithm (GA) is proposed. DLS is used to select the task and the GA is used to select the platform for the selected task, then the design flow of this approach is presented. Finally, the superiority and applicability of this approach are illuminated by the cases of joint campaign.

Keywords: operations research; match between task and platform; loss coefficient; dynamic list scheduling (DLS); genetic algorithm (GA)

0 引 言

信息化战争面临竞争激烈的战场环境,使得作战任务类型多样,关系复杂;同时高科技的发展带来武器装备的进步,使得可调用的平台资源数量庞大,功能各异,能力多样。如何将可使用的平台资源在使命的任务流程上进行科学配置和部署,做到作战任务-平台资源的合理匹配,从而实现在合适的时间和地点运用最为适合的平台执行合适的任务。该问题实质上是平台资源的调度问题,平台资源调度是战前使命规划的中心内容。

关于平台资源调度问题,常见的解决方法有[1-2]:分支定界法(branch and bound)、动态规划(dynamic programming, DP)、动态列表调度(dynamic list scheduling, DLS)等。其中基于 DLS 的启发式方法是解决该问题较为有效的方法,常见的基于 DLS 的启发式方法主要有多维动态列表调度(multi-dimensional DLS, MDLS)[3-4]算法和多优先级列表动态调度(multi-PRI list dynamic scheduling, MPLDS)[5-6]算法。在平台资源的调度问题中,为单个任务分配最佳的平台或平台组是获得优化的平台资源调度方案的关键[7],而 MDLS 和 MPLDS 都是基于贪心的准则来为选定的任务分配平台或平台组。贪心算法是一种近似最优的启发式算法,只能得到一个优化的解,而该优化的解未必是最优解。

因此,本文在完善任务-平台匹配问题模型的基础上,针对 MDLS 算法和 MPLDS 算法存在的不足,提出了基于 DLS 和遗传算法(genetic algorithm, GA)的任务-平台的匹配算法,使用 GA 这种全局搜索方法来为选定的任务分配最佳的平台或平台组。

收稿日期:2011-05-19; **修回日期**:2011-11-09。
基金项目:空军工程大学电讯工程学院博士创新基金(200907);空军工程大学创新专项(Dx201006)资助课题
作者简介:张杰勇(1983-),男,博士研究生,主要研究方向为指控组织建模。E-mail:dumu3110728@126.com

1 任务-平台匹配问题建模

1.1 任务-平台匹配问题的基本概念

为了实现任务和平台的匹配，首先对任务和平台进行描述，给出任务和平台以匹配为目的的属性。

(1) 任务(Tasks)：由使命分解可以得到完成该使命所需要执行的任务序列图。对任务的描述就是建立任务序列图的数据属性，任务序列图的数据属性包括任务集 T、任务的自身属性和任务间的顺序关系 G_T。

任务集：$T=\{T_i\}(i=1,2,\cdots,N)$，$N$ 表示任务序列图中任务的数量。

任务自身属性：作战任务 T_i 的处理时间 $t_i(i=1,2,\cdots,N)$；执行 T_i 的地理位置坐标 $La_i=(x_i,y_i)$；成功执行 T_i 所需要的资源能力矢量 $\boldsymbol{RE}_i=(R_{i1},R_{i2},\cdots,R_{iL})$，$R_{il}$ 是指成功处理 T_i 所需第 l 种类型的资源能力的数量($l=1,2,\cdots,L$，L 为资源能力类型的数量)。

任务间的顺序关系：G_T 描述任务之间的依赖关系。

(2) 平台(Platform)：文中平台指平台资源，平台是功能的载体。对平台的描述就是建立平台的数据属性，包括平台集 P 和平台的自身属性。

平台集：$P=\{P_m\}(m=1,2,\cdots,K)$，$K$ 表示可使用平台的数量。

平台自身属性：P_m 初始所具备的资源能力矢量 $\boldsymbol{PO}_m=(r_{m1},r_{m2},\cdots,r_{mL})$，$r_{il}$ 指 P_m 初始所具备的第 l 种类型的资源能力的数量 P_m 的最大移动速度 v_m。

1.2 任务-平台匹配问题的数学描述

任务-平台的匹配问题就是把平台分配到使命的任务序列图上，该问题的求解结果就是任务-平台匹配方案。通常某一任务的处理需要具备处理这一任务的所有条件，这些条件包括：

(1) 这一任务的所有前导任务都已处理完毕；

(2) 分配给这一任务的所有平台都已部署到达到该任务的执行地点；

(3) 执行这一任务的所有平台的资源能力之和不小于执行该任务所需的资源能力。

本文设定任务-平台匹配问题的目标就是确定最佳的任务-平台匹配方案，从而极小化使命完成的总时间。

对任务-平台匹配问题的建模可以通过以下步骤来描述。

1.2.1 匹配过程的变量定义

(1) 平台-任务的分配变量 ω_{im}：

$$\omega_{im}=\begin{cases}1, & P_m \text{ 分配给 } T_i\\ 0, & \text{否则}\end{cases}$$

(2) 平台在任务间的转移变量 x_{ijm}：

$$x_{ijm}=\begin{cases}1, & P_m \text{ 在执行完 } T_i \text{ 后被分配给 } T_j\\ 0, & \text{否则}\end{cases}$$

当 $i=j$ 时，本身不存在平台在任务间的转移变量，但为了下文数学描述的方便，设定当 $i=j$ 时，平台在任务间的转移变量为 0，即 $x_{iim}=x_{jjm}=0(i,j=1,2,\cdots,N;m=1,2,\cdots,K)$。

假设存在虚拟任务 T_0，表示所有任务的起始或终止，当平台在任务间的转移变量为 $x_{0jm}(j=1,2,\cdots,N)$ 时，T_0 表示所有任务的起始，而当转移变量为 $x_{i0m}(i=1,2,\cdots,N)$ 时，T_0 表示所有任务的终止。在使命执行开始和使命执行完毕，所有平台都在虚拟任务 T_0 上，并令转移变量 $x_{00m}=0$ $(m=1,2,\cdots,K)$。

(3) 任务的顺序变量 a_{ij}：

$$a_{ij}=\begin{cases}0, & T_i \text{ 必须在 } T_j \text{ 开始执行前完成}\\ 1, & \text{否则}\end{cases}$$

(4) 时间变量 s_i：s_i 是执行 T_i 的开始时间。

(5) 使命的完成时间 Y：即完成使命中最后一个任务的时间。任务-平台匹配问题的目标函数就是极小化 Y。

(6) P_m 的资源能力矢量 $\boldsymbol{PO}'_m$：由第 1.1 节可知，P_m 初始所具备的资源能力矢量为 $\boldsymbol{PO}_m=(r_{m1},r_{m2},\cdots,r_{mL})$。本文假设平台在执行任务的过程中，由于操作人员疲劳以及平台损耗等原因，其资源能力随执行任务量的增加而逐渐递减。

因此，在问题模型中引入平台在执行完某一任务后其资源能力更新公式：

$$r_{ml}^{\text{renewed}}=r_{ml}\left(1-W_l\frac{\tilde{r}_{ml}}{r_{ml}}\right),\ l=1,2,\cdots,L \tag{1}$$

式中，W_l 表示第 l 种资源能力的损耗系数，本文假设 W_l 只与第 l 种资源能力的特点有关，而与发生能力损耗的平台和执行的任务无关；r_{ml} 为 P_m 在执行某一任务前所拥有的第 l 种资源能力的数量；r_{ml}^{renewed} 为 P_m 在执行完该任务后所拥有的第 l 种资源能力的数量；$\tilde{r}_{ml}$ 为 P_m 在执行任务过程中实际使用的第 l 种资源能力的数量。

由式(1)可知，平台在执行完某一任务后，其资源能力的损耗与该资源能力的损耗系数以及执行该任务过程中实际使用的该资源能力数量有关。

更新完后，平台 P_m 的资源能力数量为 $\boldsymbol{PO}_m^{\text{renewed}}=(r_{m1}^{\text{renewed}},r_{m2}^{\text{renewed}},\cdots,r_{mL}^{\text{renewed}})$。设使命执行过程中，$P_m$ 变化的资源能力数量为 $\boldsymbol{PO}'_m=(r'_{m1},r'_{m2},\cdots,r'_{mL})$。

1.2.2 匹配过程的约束分析及匹配问题的数学描述

匹配过程的约束分析：

(1) 平台的分配约束

假如 P_m 和 T_i 存在分配关系，即 $\omega_{im}=1$，则 P_m 肯定是执行完某一个任务(包括所有任务的起点虚拟任务 T_0)之后被分配执行 T_i，即

$$\sum_{j=0}^{N}x_{jim}-\omega_{im}=0 \tag{2}$$

而 P_m 执行完 T_i 之后，又会被分配执行下一个任务(包括所有任务的终止虚拟任务 T_0)，即

$$\sum_{j=0}^{N}x_{ijm}-\omega_{im}=0 \tag{3}$$

所有平台都是从起始虚拟任务开始执行使命，也都是在终止虚拟任务完成执行使命，即

$$\sum_{j=0}^{N}x_{0jm}=\sum_{i=0}^{N}x_{i0m}=1,\ m=1,2,\cdots,K \tag{4}$$

(2) 任务开始时间约束

对于 P_m 在 T_i 和 T_j 之间存在两种关系,即 $x_{ijm}=0$ 或 $x_{ijm}=1$。

当 T_i 必须在 T_j 开始执行前完成,即 $a_{ij}=0$ 时,任务的开始时间必须满足

$$s_j \geqslant s_i + t_i \tag{5}$$

考虑 P_m 在 T_i 和 T_j 之间的执行关系,式(5)可以转化为

$$s_j \geqslant s_i + t_i + x_{ijm}\frac{d_{ij}}{v_m} \tag{6}$$

式中,d_{ij} 表示 T_i 与 T_j 之间的空间距离,表达式为

$$d_{ij} = \sqrt{(x_j - x_i)^2 + (y_j - y_i)^2} \tag{7}$$

当不满足 T_i 必须在 T_j 开始执行前完成,即 $a_{ij}=1$ 时,任务的开始时间必须满足

$$s_j \geqslant s_i + t_i - Y' \tag{8}$$

式中,Y'为所有任务完成时间的上界(一般设置为较大的值),满足 $Y' \geqslant Y$。

综合式(6)和式(8),得出当 $a_{ij}=0$ 或 $a_{ij}=1$ 时,任务开始时间约束满足

$$s_j \geqslant s_i + t_i + x_{ijm}\left(\frac{d_{ij}}{v_m} + a_{ij}Y'\right) - a_{ij}Y' \tag{9}$$

(3) 任务的资源需求约束

成功执行 T_i 的资源能力需求条件为:执行这一任务的所有平台的资源能力向量之和中的每一种类型的资源能力数量都不小于该任务对应类型所需的资源能力数量,即

$$\sum_{m=1}^{K} r'_{ml}\omega_{im} \geqslant R_{il} \tag{10}$$

(4) 使命完成时间约束

Y 不小于任意任务的完成时间,即

$$Y \geqslant s_i + t_i \tag{11}$$

综合以上匹配过程的约束条件,得出任务-平台匹配问题的数学描述如下:

$$\min Y$$

$$\text{s.t.}\begin{cases} \sum_{j=0}^{N} x_{jim} - \omega_{im} = 0 \\ \sum_{j=0}^{N} x_{ijm} - \omega_{im} = 0 \\ \sum_{j=0}^{N} x_{0jm} = \sum_{i=0}^{N} x_{i0m} = 1 \\ s_j \geqslant s_i + t_i + x_{ijm}\left(\frac{d_{ij}}{v_m} + a_{ij}Y'\right) - a_{ij}Y' \\ d_{ij} = \sqrt{(x_j - x_i)^2 + (y_j - y_i)^2} \\ Y \geqslant s_i + t_i \\ \sum_{m=1}^{K} r'_{ml}\omega_{im} \geqslant R_{il} \\ 0 \leqslant Y \leqslant Y' \\ s_i \geqslant 0 \\ x_{jim}, \omega_{im} \in \{0,1\} \end{cases} \tag{12}$$

2　问题求解

由式(12)关于任务-平台匹配问题的数学描述可知,任务-平台匹配问题是混二元线性规划问题,该问题被证明是一个非确定性多项式(non-deterministic polynomial, NP)难问题[3]。本文提出了基于 DLS 和 GA 的问题求解算法,算法的流程如图 1 所示。该算法主要包含两个关键步骤:使用 DLS 选择当前需要处理的任务;使用 GA 为选定任务分配最佳的平台或平台组。

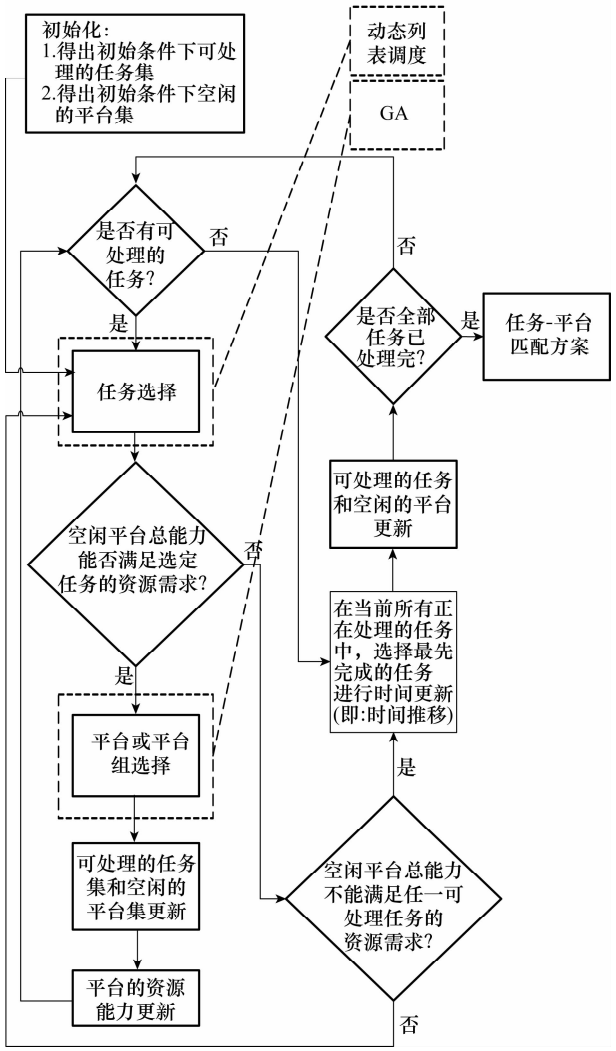


图 1　基于 DLS 和 GA 的匹配算法流程图

本节以下部分就是这两个关键步骤的具体描述。

2.1　任务选择

任务选择取决于任务序列图的连接关系,在当前可选的任务集 T_{ready}(T_{ready}是指所有前导任务已被处理完的任务集合,是一个任务列表)中根据任务的优先权系数选择当前要处理的任务。对于任务优先权系数的计算,文献[8]给出了 3 种算法:关键路径(critical path, CP)算法、层次分配(level assignment, LA)算法和加权长度(weighted length, WL)算法。

其中，在 WL 算法中，任务的优先权系数取决于该任务的处理时间和该任务直接后续任务的优先权系数，用 WL 算法计算任务优先权系数的公式如下：

$$WL(i)=t_i+\max_{j\in OUT(i)}WL(j)+\sum_{j\in OUT(i)}WL(j)/\max_{j\in OUT(i)}WL(j) \tag{13}$$

式中，$OUT(i)$为任务序列图中 T_i 的直接后续任务集，而 $WL(i)$为用 WL 算法求得的 T_i 的优先权系数。

WL 算法的另一种变换的算法为加权关键路径(weighted CP, WCP)算法，其计算公式为

$$WCP(i)=CL(i)+\max_{j\in OUT(i)}CL(j)+\sum_{j\in OUT(i)}CL(j)/\max_{j\in OUT(i)}CL(j) \tag{14}$$

式中，$CL(i)$为任务序列图中 T_i 到使命结束的关键路径长度，而 $WCP(i)$为用 WCP 算法求得的 T_i 的优先权系数。

有关 CP、LA、WL 和 WCP 4 种算法的详细描述，可参考文献[3,8]。

2.2 基于 GA 的平台资源的选择

2.2.1 染色体的编码和解码方式

设选定的任务为 $T_M(M\in\{1,2,\cdots,N\})$，从当前空闲的平台集 P_{free}(P_{free}为当前没有分配任务的平台资源集合)中选择最佳的平台或平台组分配给该任务，该问题可建模为多维的 0-1 背包问题(multi-dimensional 0-1 knapsack problem, MKP)。设 P_{free}中平台的数量为 N_{free}，对 P_{free}中的平台进行编号，平台标号为 $1,2,\cdots,N_{free}$。对于不同的任务，P_{free}中平台的数量和类型都是不同的，因此 N_{free}也是动态变化的。

使用 GA 求解 MKP，如果采用单纯 0-1 二进制编码方式，有时就会产生很多不可行的个体(不满足任务的资源能力需求)，从而降低 GA 的效率。为了克服这个缺点，本文采用双串(double strings)结构[9]对染色体进行编码，如图 2 所示。

平台标号	$S_{(1)}$	$S_{(2)}$	……	$S(N_{free})$
0-1值	$G_{S(1)}$	$G_{S(2)}$	……	$G_{S(N_{free})}$

图 2 染色体的双串结构

图 2 中，$S(i)(i=1,2,\cdots,N_{free})$表示平台标号，对应 P_{free}中的平台，序列$[S(1),S(2),\cdots,S(N_{free})]$为 $1\sim N_{free}$的一个随机序列，$G_{S(i)}$为平台标号为 $S(i)$的平台所对应的状态取值，值为 0 或 1。

这种结构通过特定的解码方式产生的原象总为可行解，从而可以提高 GA 的效率，以下为这种双串结构的解码方式：

步骤 1 设染色体解码后产生的原象为$(x_1,\cdots,x_i,\cdots,x_{N_{free}})(i=1,2,\cdots,N_{free})$，$x_i$ 表示 P_{free}中平台标号为 i 的平台的真实状态(1 表示为选定任务分配了相应的平台，0 表示没有为选定任务分配相应的平台)。设平台标号为 i 的平台所拥有的资源能力向量为 $\boldsymbol{PO}_i=(r_{i1},r_{i2},\cdots,r_{iL})$，可计算得 P_{free}中所有平台资源能力向量之和为 $\boldsymbol{PS}=(rs_1,rs_2,\cdots,rs_L)$，定义一个中间变量 $\boldsymbol{Tmp}=\boldsymbol{PS}$，设选定任务 T_M 的资源能力需求向量为 $\boldsymbol{RE}_M$。

步骤 2 令 $j=1$。

步骤 3 假如 $G_{S(j)}=1$，则令 $x_{S(j)}=1$，转步骤 6；否则，转步骤 4。

步骤 4 假如满足约束条件 $\boldsymbol{PS}-\boldsymbol{PO}_{S(j)}\geqslant\boldsymbol{RE}_M$(是指向量中每一种类型的资源能力都大于)，则令 $x_{S(j)}=0$，$\boldsymbol{Tmp}=\boldsymbol{PS}-\boldsymbol{PO}_{S(j)}$；否则，令 $x_{S(j)}=1$。

步骤 5 $\boldsymbol{PS}=\boldsymbol{Tmp}$。

步骤 6 $j=j+1$，假如 $j>N_{free}$，结束循环；否则，转步骤 3。

2.2.2 适应度函数

一个染色体代表着 P_{free}中平台的一种组合方式，求染色体的适应值，就是求平台组合的适应值。记平台组合为 CP，表示染色体原象中为“1”的位置所对应平台的集合，记 NCP 为 CP 中平台的数量。设 GA 的群体规模为 ND，则一个种群中共有 ND 个 CP，记为 $CP_n(n=1,2,\cdots,ND)$，相应的平台数量记为 NCP_n。

平台组合 CP 时间优先权系数的计算方法如下。

记平台组合 CP_n 开始执行任务的时间$ST(CP_n)$，$ST(CP_n)$为 CP_n 中最迟到达任务处理地点的平台的到达时间：

$$ST(CP_n)=\max_{m\in CP_n}\left(s_{l(m)}+t_{l(m)}+\frac{d_{l(m),M}}{v_m}\right) \tag{15}$$

式中，$l(m)$为 P_m 最后处理的任务。如果 P_m 没有执行过任务(即还在起始虚拟任务 T_0 上)，则 $ST(P_m)=0$。

记一个种群中最优的个体所对应的任务开始时间(即开始时间最小)为 $ST_{\min}$，则 $ST_{\min}$为

$$ST_{\min}=\min_{n=1,2,\cdots,ND}(ST(CP_n)) \tag{16}$$

以 $ST_{\min}$为基准，对这个种群中的所有 CP_n 的$ST(CP_n)$进行归一化处理，处理的结果就是 CP_n 的时间优先权系数：

$$PT(CP_n)=\frac{ST(CP_n)-ST_{\min}}{\sum_{i=1}^{ND}(ST(CP_i)-ST_{\min})} \tag{17}$$

平台组合 CP 资源能力矢量优先权系数的计算方法如下。

记 CP_n 分配给 T_M 的资源能力冗余程度为 $DR(CP_n)$，则 $DR(CP_n)$为

$$DR(CP_n)=\sum_{l=1}^{L}\left(\sum_{m\in CP_n}r'_{ml}-R_{Ml}\right) \tag{18}$$

记 CP_n 中的各个平台相对于 T_{ready} 中其他任务的平均资源满足程度 $QR(CP_n)$，则 $QR(CP_n)$为

$$QR(CP_n)=\frac{1}{NCP_n}\sum_{m\in CP_n}\sum_{j\in T_{ready}\setminus\{M\}}\sum_{l=1}^{L}\min(r'_{ml},R_{jl}) \tag{19}$$

综合式(18)和式(19)，得出 CP_n 的资源能力矢量的综合值 $SR(CP_n)$为

$$SR(CP_n) = DR(CP_n) + QR(CP_n) \tag{20}$$

记一个种群中最优个体所对应的资源能力矢量的综合值 $SR_{\min}$(资源能力矢量的综合值最小),则 $SR_{\min}$ 为

$$SR_{\min} = \min_{n=1,2,\cdots,ND} SR(CP_n) \tag{21}$$

以 $SR_{\min}$ 为基准,对这个种群中的所有 CP_n 的 $SR(CP_n)$ 进行归一化处理,处理的结果就是 CP_n 的资源能力矢量的优先权系数:

$$PR(CP_n) = \frac{SR(CP_n) - SR_{\min}}{\sum_{i=1}^{ND}(SR(CP_i) - SR_{\min})} \tag{22}$$

记 CP_n 的优先权系数为 $Pr(CP_n)$,则综合式(17)和式(22),$Pr(CP_n)$ 为

$$Pr(CP_n) = \alpha PT(CP_n) + (1-\alpha)PR(CP_n) \tag{23}$$

式中,$\alpha \in (0,1)$,α 和 $1-\alpha$ 表示规划者在进行任务和平台匹配时的主观倾向性。

优先权系数越小,平台组合优先等级越高,该平台组合越适合与选定的任务相匹配。因此,设计 GA 的适应度函数为 CP_n 优先权系数的倒数,为

$$\text{Fitness} = \frac{1}{1 + Pr(CP_n)} \tag{24}$$

2.2.3　遗传算子

(1) 交叉算子

与传统 GA 不同,对于这种双串结构的染色体编码方式,本文采用部分匹配交叉(partially matched crossover, PMX)[10]的方法进行交叉操作。PMX 能够使这种双串的父代染色体结构经过交叉后得到需要的子代染色体,而不改变这种双串的结构。

初始种群通过交叉算子可以得到规模为 $2 \cdot ND$ 的新种群,称这个新种群为交叉种群。

(2) 变异算子

采用均匀变异(uniform mutation)[10]的方法进行变异操作。

初始种群通过变异算子可以得到规模为 ND 的新种群,称这个新种群为变异种群。

(3) 选择算子

由文中第 2.2.2 节可知,个体的适应值会随种群的改变而变化,只有在一个种群内的个体才能进行适应值的比较。因此,本文设计的选择算子是:将原始种群、交叉种群和变异种群 3 个种群合并为一个种群,这个种群的规模为 $4 \cdot ND$,称这个种群为合并种群。采用第 2.2.2 中计算适应值的方法,计算每个个体在合并种群中的适应值,使用精英保留策略(elitism)和轮盘赌选择(roulette wheel selection)相结合的方法进行选择操作。

精英保留策略:将合并种群中适应值最大的个体保留到新一代种群中;

轮盘赌选择:新一代种群中的另外 $ND-1$ 个个体通过对合并种群进行轮盘赌选择的方式产生,轮盘赌的次数为 $ND-1$ 次。

2.2.4　算法步骤

GA 的具体步骤如下:

步骤 1　根据 N_{free},设置合适的种群规模 ND,采用双串结构对染色体进行编码,随机产生初始种群 NI;

步骤 2　对 NI 进行概率为 P_c 的交叉操作和概率为 P_m 的变异操作,产生交叉种群 NC 和变异种群 NM;

步骤 3　将种群 NI,NC 和 NM 合并为一个种群,进行选择操作,产生新一代的种群,并替代 NI;

步骤 4　重复步骤 2～步骤 3 直至达到某种停止准则,本文的停止准则为最大迭代次数。

3　算例分析

3.1　特殊算例分析

以文献[11]中联合作战的战役想定为算例,在 Inter (R) Pentium(R) 4 CPU 3.00 GHz 的计算机上进行仿真实验。

算例中使命的任务序列图如图 3 所示,所需要执行的作战任务数量为 $N=18$,能够使用的平台数量为 $K=20$,任务的属性和平台资源的属性如表 1 和表 2 所示(表中只列举了部分数据,完整数据可参考文献[11])。

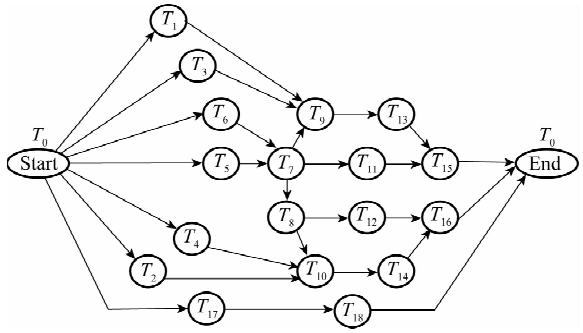


图 3　使命的任务序列图

表 1　作战任务的属性

任务	资源需求向量(***RE***)								时间	位置	
	R_1	R_2	R_3	R_4	R_5	R_6	R_7	R_8		x	y
T_1	5	3	10	0	0	8	0	6	30	70	15
T_2	5	3	10	0	0	8	0	6	30	64	75
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
T_{17}	0	0	0	0	0	8	0	4	10	5	60
T_{18}	0	0	0	8	6	0	4	10	20	5	60

表 2　平台资源的属性

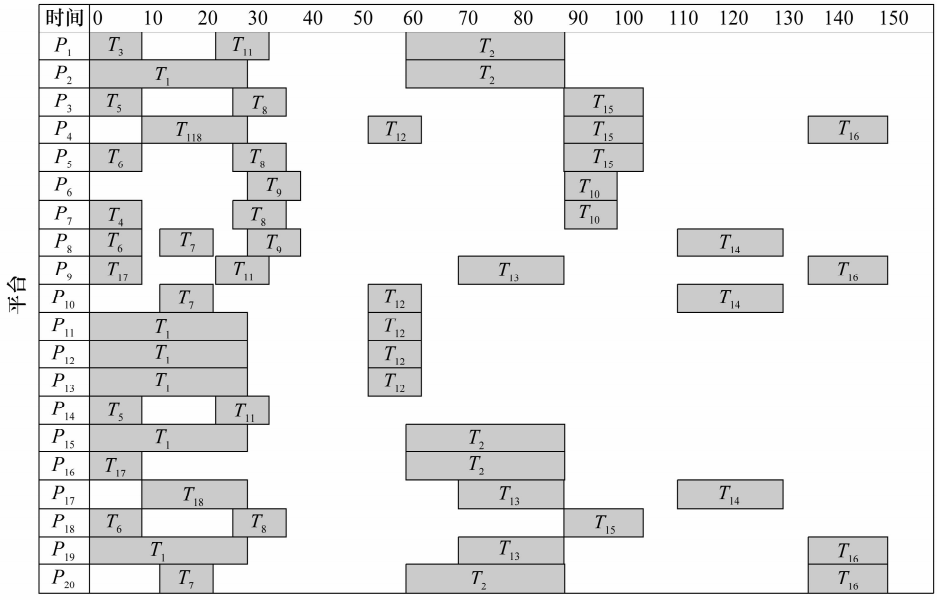
平台	平台拥有的资源能力向量(***PO***)								速度
	r_1	r_2	r_3	r_4	r_5	r_6	r_7	r_8	
P_1	10	10	1	0	9	5	0	0	2
P_2	1	4	10	0	4	3	0	0	2
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
P_{19}	1	0	0	10	2	2	1	0	1.35
P_{20}	1	0	0	10	2	2	1	0	1.35

假设终止虚拟任务 T_0 的任务优先权系数为 0,利用式(13)计算得出任务的优先权系数如表 3 所示,任务优先权系数越大,表示该任务优先级越高。

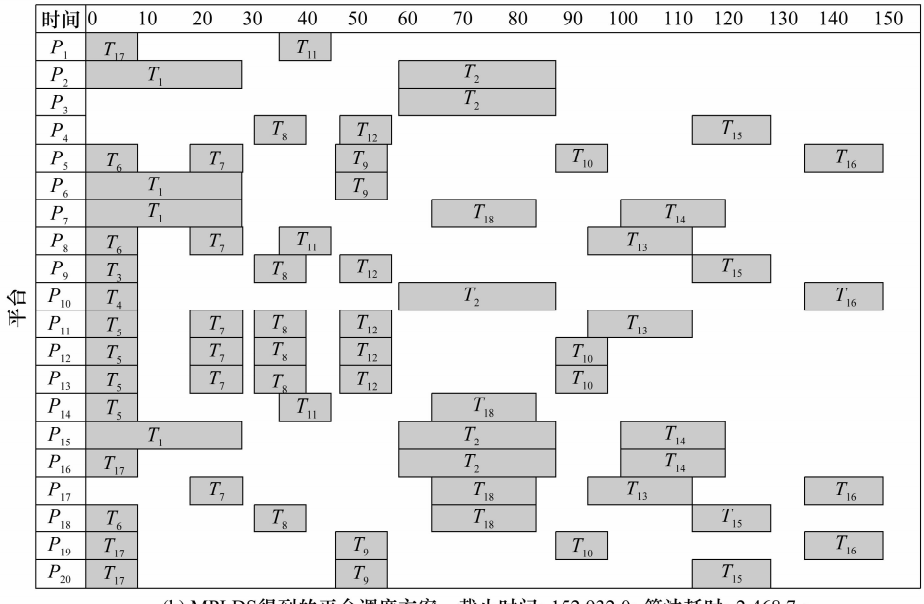
表 3 任务的优先权系数

T	WL	T	WL	T	WL	T	WL	T	WL	T	WL	T	WL	T	WL	T	WL
T_1	78	T_2	78	T_3	58	T_4	58	T_5	81.8	T_6	81.8	T_7	70.8	T_8	58.5	T_9	47
T_{10}	47	T_{11}	26	T_{12}	26	T_{13}	36	T_{14}	36	T_{15}	15	T_{16}	15	T_{17}	31	T_{18}	20

假设各种资源能力的损耗系数都为 $W_l=0.1(l=1,2,\cdots,8)$，分别使用 MDLS，MPLDS 以及本文设计的算法对以上算例进行求解，求解结果如图 4 所示。实验中，MDLS 算法选用的平台选择参数为 $R^1(m)$[4]，MPLDS 选用的权参数为 $\lambda=1$[5]（对于极小化使命完成时间这个目标函数，MDLS 算法和 MPLDS 算法在该选择的参数下得到调度方案是该算法所能得到的最优的调度方案，有关 MDLS 算法和 MPLDS 算法详细描述可参考文献[4-5]），本文设计的算法所设置的 GA 参数为：$ND=10$，迭代次数 50，$P_c=0.8$，$P_m=0.05$，$\alpha=0.5$。

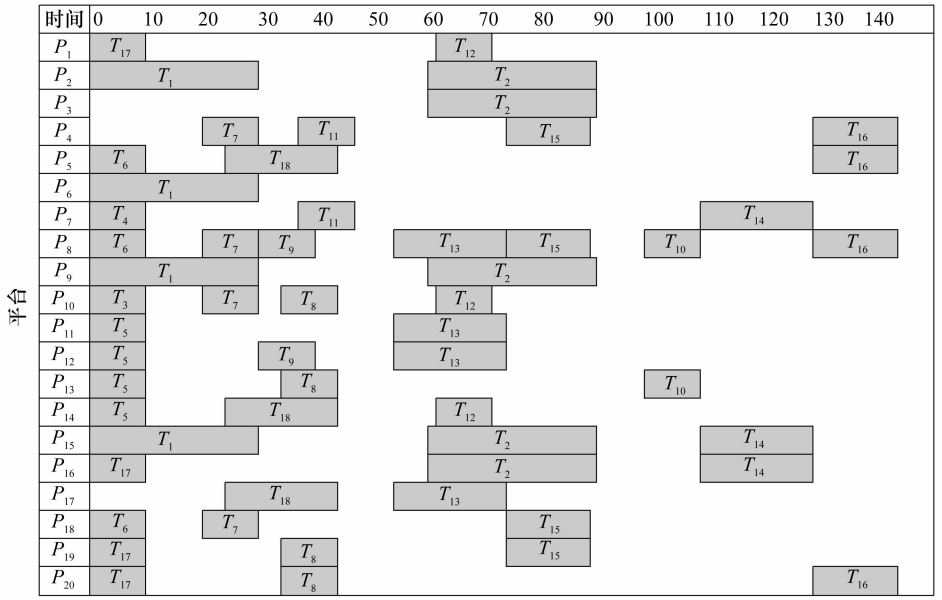


(a) MDLS得到的平台调度方案，截止时间: 151.296 0; 算法耗时: 2.265 6 s



(b) MPLDS得到的平台调度方案，截止时间: 152.932 0; 算法耗时: 2.468 7 s

图 4 不同方法得到的任务-平台匹配方案



(c) DLS和GA得到的平台调度方案，截止时间: 143.603 6: 算法耗时: 17.093 7 s

续图 4　不同方法得到的任务-平台匹配方案

3.1.1　*方法的优越性分析*

由图 4 中仿真的结果可得,MDLS 算法得到的任务-平台匹配方案的使命完成时间为 151.296 0,MPLDS 算法得到的任务-平台匹配方案的使命完成时间为 152.932 0,而本文设计的方法得到的任务-平台匹配方案的使命完成时间为 143.603 6,表明了本文设计的算法相比于 MDLS 算法和 MPLDS 算法,节省了全部任务处理的完成时间,提高了使命完成的效率,同时也提高了战场平台资源的利用率,体现了本文设计方法的优越性。该方法也为解决任务-平台匹配问题提供了一条新的求解思路。

3.1.2　*方法的适用性分析*

同时,由图 4 中仿真的结果可得,本文设计的算法运算耗时稍长于 MDLS 算法和 MPLDS 算法的运算耗时,但是本文研究的任务-平台的匹配问题是战前使命规划的中心内容,属于一种离线(off-line)的规划,所以对算法的实时性要求不是很高,因此,虽然该算法耗时稍长,但是仍然可以满足战前离线使命规划的需求。

3.2　一般算例分析

为了进一步验证本文所提算法的优越性,对一个有 18 个任务和 20 个平台的作战使命,采用蒙特卡罗的方法,随机产生任务之间的序列关系、任务的自身属性和平台资源的自身属性,分别使用 MDLS,MPLDS 以及本文设计的 DLS+GA 3 种算法计算使命的完成时间(实验中 3 种算法所采用的参数与文中第 3.1 节 3 种算法所采用的参数相同),并以 MDLS 算法得到的使命完成时间为基准,计算 3 种算法得到的使命完成时间相对于这个基准的优化率(本文设计的优化率是指 3 种算法得到的使命的完成时间与 MDLS 算法得到的使命完成时间的比值),三种算法 2 000 次蒙特卡罗仿真实验得到的优化率统计的盒须图分布如图 5 所示。

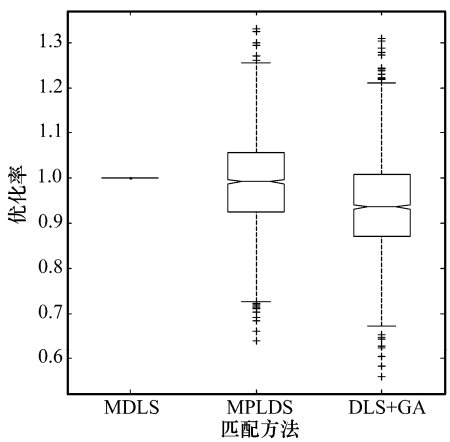


图 5　3 种算法得到优化率的盒须图分布

由图 5 中 3 种算法得到的优化率分布可知,本文提出了 DLS+GA 的任务-平台匹配算法在统计意义上得到优化率要小于 MDLS 算法和 MPLDS 算法得到优化率,表明了本文所提匹配算法在统计意义上能够得到更优的任务-平台的匹配方案,同时,也充分体现了本文算法在战前离线使命规划中的优越性。

4　结　论

本文结合 DLS 和 GA,提出了一种作战任务和平台资源的匹配算法,使用 DLS 选择当前需要处理的作战任务,而 GA 用来为选定任务分配最佳的平台或平台组。本文的

主要工作有:① 通过引入资源能力的损耗系数,完善了作战任务-平台资源匹配问题的数学模型;② 提出了使用 GA 为选定任务分配平台或平台组,针对所研究问题,设计了 GA 的双串编码和解码方法,GA 的交叉、变异和选择算子,以及适应度函数;③ 以联合作战的战役想定为背景,分别从特殊算例和一般算例两个方面对本文所提算法的优越性和适用性进行了分析和验证。

但是,本文提出的算法和模型也存在不足之处:①由于本文提出的匹配算法耗时较长,因此本算法只适合于战前离线规划中任务-平台的匹配,而不适合于作战过程中作战计划的动态调整以及一些应急情况的处置;②在大多数情况下,使用本文算法得到的调度方案是稳定的,但在极少数情况下,由于 GA 的随机性和多次使用 GA,可能会产生多个不同调度方案;③由于引入了资源能力的损耗系数,如果平台执行过多作战任务,会带来平台资源能力的大量损耗,这可能会导致使命后期的任务无法完成(所有平台的资源能力总和小于任务所需的资源能力),从而导致模型无解,要解决这个问题,可以通过设置任务完成精度,来得到使命完整的平台资源调度方案。这些将是本课题下一步的工作。

参考文献:

[1] Bui H, Han X, Mandal S, et al. Optimization-based decision support algorithms for a team-in-the-loop planning experiment [C]// *Proc. of the IEEE International Conference on Systems, Man, and Cybernetics*, 2009.

[2] Mandal S, Han X, Pattipati K R, et al. Agent-based distributed framework for collaborative planning[C]// *Proc. of the IEEE International Conference on Aerospace Conference*, 2010.

[3] Levchuk G M, Levchuk Y N, Luo J, et al. A library of optimization algorithms for organizational design[C]// *Proc. of the Command and Control Research and Technology Symposium*, 2000.

[4] Levchuk G M, Levchuk Y N, Luo J, et al. Normative design of organizations—Part I: mission planning[J]. *IEEE Trans. on Systems, Man, and Cybernetics-Part A: Systems and Humans*, 2002, 32(3):346 - 359.

[5] 阳东升, 张维明, 刘忠, 等. 战役任务计划的数学描述与求解算法研究[J]. 系统工程理论与实践, 2006, 26(1): 26 - 34. (Yang D S, Zhang W M, Liu Z, et al. Research on mathematical description and solving algorithms of tasks scheduling for campaign [J]. *Systems Engineering — Theory & Practice*, 2006, 26(1): 26 - 34.)

[6] 陈洪辉, 赵亮, 芮红, 等. 作战任务和资源间的匹配模型及求解算法研究[J]. 系统工程与电子技术, 2008, 30(9): 1712 - 1716. (Chen H H, Zhao L, Rui H, et al. Research on the match model and solving method between operational tasks and resources[J]. *Systems Engineering and Electronics*, 2008, 30(9): 1712 - 1716).

[7] 阳东升, 张维明, 刘忠, 等. 指控组织设计方法[M]. 北京: 国防工业出版社, 2010: 141 - 142. (Yang D S, Zhang W M, Liu Z, et al. *Designing of command and control organization* [M]. Beijing: National Defense Industry Press, 2010: 141 - 142.)

[8] Shirazi B, Wang M F. Analysis and evaluation of heuristic methods for static task scheduling[J]. *Journal of Parallel and Distributed Computing*, 1990, 10(3): 222 - 232.

[9] Sakawa M, Kato K. Genetic algorithms with double strings for 0-1 programming problems[J]. *European Journal of Operational Research*, 2003, 144(3): 581 - 597.

[10] 王小平, 曹立明. 遗传算法——理论、应用与软件实现[M]. 西安: 西安交通大学出版社, 2002: 28 - 38. (Wang X P, Cao L M. *Genetic algorithms — theory, application and software realized* [M]. Xi'an: Xi'an Jiaotong University Press, 2002: 28 - 38.)

[11] Yu F, Tu F, Pattipati K R. Integration of a holonic organizational control architecture and multiobjective evolutionary algorithm for flexible distributed scheduling[J]. *IEEE Trans. on Systems, Man, and Cybernetics-Part A: Systems and Humans*, 2008, 38(5): 1001 - 1017.