文章编号:1000-6788(2006)01-0026-09

战役任务计划的数学描述与求解算法研究

阳东升,张维明,刘　忠,鲁音隆
(国防科技大学 信息系统与管理学院,湖南 长沙 410073)

摘要:　以多兵种联合作战战役计划为例,在定义任务和作战平台模型的基础上,对战役计划问题进行了数学描述.在定义平台能力矢量与任务需求矢量距离的基础上,基于新的搜索策略提出了多优先级列表动态规划算法(MPLDS),并针对战役案例分析比较了 MPLDS 与 MDSL 算法的求解结果与计算性能.

关键词:　平台;任务;任务计划;多优先级动态列表规划(MPLDS)
中图分类号:　C931.1　　**文献标识码**:　A

Research on Mathematical Description and Solving Algorithms of Tasks Scheduling for Campaign

YANG Dong-sheng, ZHANG Wei-ming, LIU Zhong, LU Yin-long
(College of Information System and Management, National University of Defense Technology, Changsha 410073, China)

Abstract: Tasks and platforms are defined in the case of joint campaign, on which mathematical description about campaign scheduling is given. For solving the problem of campaign scheduling, a new approach—multiPRI list dynamic scheduling (MPLDS) algorithm is advanced, with which vector distance between vectors of platform capability and task resource requirements is defined and a new search strategy is adopted. Results and compute complexities from MPLDS algorithms and traditional MDLS algorithms to solve our case are analyzed and compared.

Key words: platform; task; task scheduling; multiPRI list dynamic scheduling (MPLDS)

1　引言

典型的计划问题可这样的描述:给定一个任务图和可获取的资源集合.任务图确定了需要处理的所有任务,任务之间的执行顺序(包括任务的串行、并行以及交叉关系)、信息和数据流向,同时明确了任务处理的时间需求,资源需求等任务的基本特征;资源具备处理任务的功能,资源有自身的基本属性(如运行速度、具备的功能能力、信息获取范围、数据或信息处理能力等),资源和任务之间通过任务的能力需求和资源的功能能力关联,以此进行资源 - 任务分配.资源到任务的分配通常以整个任务流程完成的时间最短或者以资源的充分利用为目标.分配过程的约束问题包括同一资源能同进处理的任务数量、任务需求的满足程度以及整个任务流程的时限等等.

有效的计划是任务处理策略关键,而任务的有效计划需要考虑资源的分配和任务的并行处理,这一问题经常出现在兵力规划、车辆运输调度、指派问题、多处理器并行处理机制和多处理器系统的动态调度[1,2]上,对这一问题的解决有许多调度算法的研究.

这类问题通常都是 NP 完全问题,快速启发式方法被认为是解决这一问题的途径[3].解决这一问题的大多方法可以归为列表规划(list-scheduling)方法,这些方法包括 PCT (Partial Completion Time static priority)[4], BIL (Best Imaginary Level)[5], HEFT (Heterogeneous Earliest Finish Time)[6], DLS (Dynamic Level Scheduling)[7],MDLS (Multidimensional dynamic list scheduling)[8],等等.所有这些启发式方法的重点都可以划分为两个部分:一是确定任务的优先权函数,二是确定目标函数.在这两个部分划分的基础上这些算法

收稿日期:2004-11-18
资助项目:国家自然科学基金(70271004;70401003)
作者简介:阳东升(1976 -),男,博士,湖北公安人.主要研究方向为任务计划、资源调度以及组织设计;张维明(1962 -),男,教授,博士生导师.主要研究方向:信息系统工程,指挥自动化,体系与体系对抗.

都分两步进行搜索求解,第一步是对每一个任务搜索最优分配方案,即分配执行这一任务的成员组(如多处理器,平台,主体等)最小化任务的执行时间;第二步是对已经分配的任务确定任务执行的进度表,即确定任务执行的时间顺序,在这一过程中最小化整个任务过程的完成时间,最小化整个任务流程的完成时间的典型方法是关键路径最小化方法.

在资源与任务的匹配需要多维变量测度,任务需要不同资源或异构平台的协同处理,资源个体能同时处理多个任务的情况下,这一调度问题的复杂性大大增加. 对这一类复杂问题的求解 Levchuk 提出了多维动态列表规划算法(MDLS),而 MDLS 在资源分配过程中又存在局部搜索和优先权函数的合理性问题,导致这一算法在解决这一类复杂调度问题上存在缺陷. 本文以多兵种联合战役为例,对战役计划问题进行了数学描述,在描述的基础上对其计划问题的求解提出了新的算法,并通过算法对本文案例的求解分析比较了两种算法的计算性能与求解结果.

2　案例分析 ——多兵种联合战役

有效的计划是战役胜负的关键. 以一次联合作战的登陆战役为例,战役环境、战场作战平台资源和战役的任务区域划分如图 1 所示,其使命是登陆抢占机场和港口,为后续部队向纵深推进扫清障碍,为有效计划我们需要对战场作战资源进行分类,建立任务和作战平台模型(案例数据来源于 A2C2 实验七[10]).

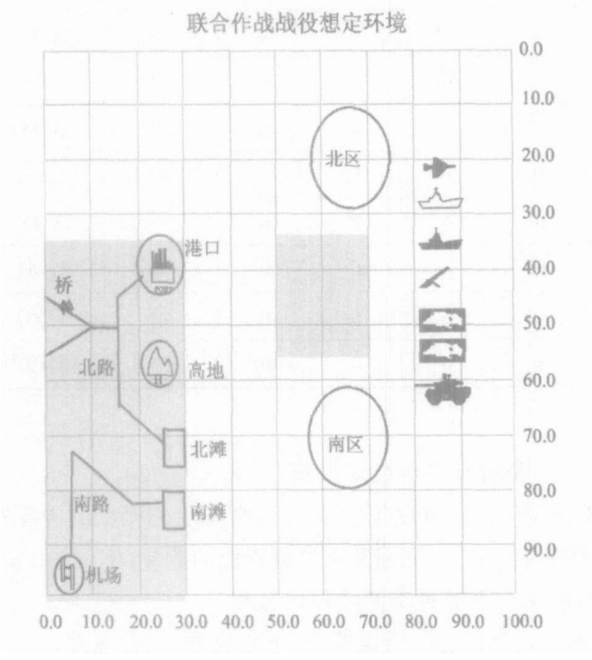

图 1　多兵种联合作战战役想定环境

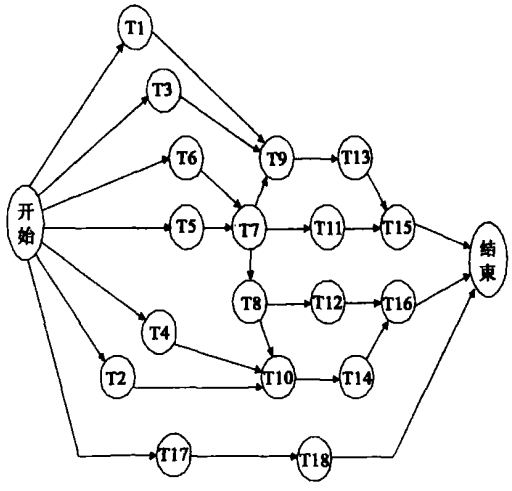

图 2　任务流程图

战役拥有兵力资源可分为防空类 l_1、反舰类 l_2、反潜类 l_3、地面攻击类 l_4、炮兵类 l_5、装甲类 l_6、扫雷类 l_7 和识别探测类 l_8. 定义平台资源的能力矢量 $C=[c_1,c_2,\cdots,c_8]$, c_i 为 l_i 类资源(在案例中 $1\leq i\leq 8$)的能力度量,则作战平台具备的资源以及战役任务对资源的需求都可以以矢量 C 进行描述.

2.1　任务、平台建模

1) 任务模型

任务模型包括的数据信息有任务集 T_s、任务自身属性 T_A 和任务间的顺序关系 G_T. T_s 是战役使命分解而得到的子任务集合, $T_s=\{t_1,t_2,\cdots,t_j\}$, j 是使命分解得到的子任务数量. T_A 指任务处理时间 t_i、处理的资源需求 R_i 和地理位置 location(x,y) 等. 任务 T_i 资源需求可表示为矢量 $R_i=C=[r_{i1},r_{i2},\cdots,r_{in}]$,其中 r_{in} 是任务 t_i 处理时所需要的第 n 类型资源(案例中 $1\leq n\leq 8$). G_T 定性描述任务之间的依赖关系,如任务的优先顺序、数据流程以及任务间的输入输出关系等. 一般采用关系图来描述任务关系.

由战役使命分解得到任务集 $T_s=\{t_1,t_2,\cdots,t_{10}\}$,如图 1 中任务表所示,任务属性和任务流程图如下.

表 1 任务参数

任务 \ 参数		资源需求								处理时间 (h)	地理位置 (x, y)
		r_1	r_2	r_3	r_4	r_5	r_6	r_7	r_8		
T_1	北区防御	5	3	10	0	0	8	0	6	30	(70 ,15)
T_2	南区防御	5	3	10	0	0	8	0	6	30	(64 ,75)
T_3	补给(北区)	0	3	0	0	0	0	0	0	10	(15 ,40)
T_4	补给(南区)	0	3	0	0	0	0	0	0	10	(30 ,95)
T_5	清除海区障碍	0	3	0	0	0	0	10	0	10	(28 ,73)
T_6	压制高地	0	0	0	10	14	12	0	0	10	(24 ,60)
T_7	抢占北滩	0	0	0	10	14	12	0	0	10	(28 ,73)
T_8	抢占南滩	0	0	0	10	14	12	0	0	10	(28 ,83)
T_9	北滩防御	5	0	0	0	0	5	0	0	10	(28 ,73)
T_{10}	南滩防御	5	0	0	0	0	5	0	0	10	(28 ,83)
T_{11}	北路行进	0	0	0	0	0	10	5	0	10	(25 ,45)
T_{12}	南路行进	0	0	0	0	0	10	5	0	10	(5 ,95)
T_{13}	清除北路威胁	0	0	0	0	0	8	0	6	20	(25 ,45)
T_{14}	清除南路威胁	0	0	0	0	0	8	0	6	20	(5 ,95)
T_{15}	占领港口	0	0	0	20	10	4	0	0	15	(25 ,45)
T_{16}	占领机场	0	0	0	20	10	4	0	0	15	(5 ,95)
T_{17}	桥头遭遇	0	0	0	0	0	8	0	4	10	(5 ,60)
T_{18}	炸桥阻援	0	0	0	8	6	0	4	10	20	(5 ,60)

2) 作战平台模型

平台实体模型包括平台集 P_s 和平台自身属性 P_A. P_s 是所有平台组成的集合. $P_s = \{ p_1 , p_2 , \cdots, p_h \}$, h 是中平台的数量. P_A 定义平台的类型、资源能力、信息获取范围、地理位置以及其他属性. 平台 m 具备的资源能力矢量为 $S_m = [s_{m1} , s_{m2} , \cdots, s_{ml}]$,其中 s_{ml} 中表示在平台 P_m 上可获取 l 类型资源的数量(案例中 $1 \leq l \leq 8$). 如一架飞机平台上带有各种类型的导弹,一艘军舰平台上各种类型武器的数量.

战役中所有平台如表 2,表 2 描述了各参战平台属性,包括参数:资源能力、平台类型、运行速度.

2.2 战役任务计划的数学描述

通常,某一作战任务的处理需要具备处理这一任务的所有条件,这些条件包括足够的平台资源、平台资源在任务区域集结完毕以及该任务准备就绪. 由此,假设某一任务被平台资源进行处理需要具备如下条件和约束:

1) 在这一任务之前的所有任务都已处理完毕;

2) 分配到这任务的所有作战平台已到达指定地点;

3) 聚集的平台资源不小于任务的资源需求;

4) 一个作战平台每次只能处理一个任务.

战役计划目标是缩短完成总任务的时间,提高完成战役使命的有效性,其有效性需要分配合适的平台或平台组到正确的区域去执行合理的任务,具体地说,就是在满足任务的资源需求的情况下提高平台资源的利用率、缩短完成任务过程的时间,同时,减少作战平台在任务执行上不必要的协作. 因此,对这一计划过程可以进行如下步骤的描述:

1) 定义分配过程变量;

平台-任务分配变量 w_{im} :平台 p_m 分配给任务 t_i 时 $w_{im}=1$,否则 $w_{im}=0$;

平台在任务间的转移变量 x_{ijm} :平台 p_m 处理任务 t_i 后分配给任务 t_j 则 $x_{ijm}=1$,不分配给任务则 $x_{ijm}=0$;

任务顺序变量 a_{ji} :如果任务 t_j 的处理必须在任务 t_i 处理完后才能开始则为 0 ,否则 $a_{ji}=1$;

时间变量 s_i : s_i 任务 t_i 的处理开始时间.

表 2　平台参数

参数 / 平台	资源能力 s_1	s_2	s_3	s_4	s_5	s_6	s_7	s_8	速度 v
1	10	10	1	0	9	5	0	0	2
2	1	4	10	0	4	3	0	0	2
3	10	10	1	0	9	5	0	0	2
4	0	0	0	2	0	0	5	0	4
5	1	0	0	10	2	2	1	0	1.35
6	5	0	0	0	0	0	0	0	4
7	3	4	0	0	6	10	1	0	4
8	1	3	0	0	10	8	1	0	4
9	1	3	0	0	10	8	1	0	4
10	1	3	0	0	10	8	1	0	4
11	6	1	0	0	1	1	0	0	4.5
12	6	1	0	0	5	1	0	0	4.5
13	6	1	0	0	1	1	0	0	4.5
14	0	0	0	0	0	0	10	0	2
15	0	0	0	0	0	0	0	6	5
16	0	0	0	0	0	0	0	6	7
17	0	0	0	6	6	0	1	10	2.5
18	1	0	0	10	2	2	1	0	1.35
19	1	0	0	10	2	2	1	0	1.35
20	1	0	0	10	2	2	1	0	1.35

2) 分配过程的约束分析.

对任意平台 p_m $(1\le m\le K)$ 和任务 t_i $(1\le i\le N)$ 之间如果存在分配关系 $(w_{im}=1)$,则平台 p_m 有且仅有两种情况被分配去执行任务 t_i :一种情况是平台 p_m 在处理完任务 t_j 后被分配处理任务 t_i ,即转移变量 $x_{jim}=1$;另一种是平台 p_m 被首次使用,直接被分配处理任务 t_i ,在这种情况下不存在转移变量,即转移变量 $x_{jim}=1$. 由此,我们假设存在虚拟任务 t_0 , t_0 是所有任务的起点,在分配初始,所有平台资源都在虚拟任务 t_0 上,记 $x_{iim}=x_{jjm}=0$,则平台-任务分配变量 w_{im} 和平台在任务间的转移变量 x_{ijm} 存在如下约束关系:

$$\sum_{j=0}^{|T|}x_{jim}-w_{im}=0\quad i=1,2,\cdots,|T|;m=1,2,\cdots,|P|. \tag{1}$$

式中 T , P 分别为任务集和平台集.

同时,被分配处理任务 t_i 的平台 p_m ,由于每次只能处理一个任务,在处理完任务 t_i 后只能被分配到某一个任务,而不能同时被分配处理多个任务,即

$$\sum_{j=0}^{|T|}x_{ijm}\le w_{im}\quad i=1,2,\cdots,|T|;m=1,2,\cdots,|P|. \tag{2}$$

由于任务间的顺序关系,任务 t_i 的处理开始必须在其所有前面任务处理完毕之后,因此,存在顺序关系的任务的处理时间存在如下约束:

$$\begin{cases} s_j-s_i\ge D(t_i) \\ a_{ij}=1 \\ i,j=1,2,\cdots,|T| \end{cases}. \tag{3}$$

(3)式中 $D(t_i)$ 为任务 t_i 的处理时间.

平台 p_m $(1\le m\le K)$ 在处理完任务 t_i 后被分配处理任务 t_j 时,由于任务的处理需要处理这一任务的所有平台都达到任务区域,集结完毕,显然执行这一任务的所有平台不可能同时到达,先到达的平台需要等待,因此平台 p_m 开始处理任务 t_j 的开始时间 s_j 不小于平台 p_m 到任务 t_j 区域的时间,即

$$s_j\ge s_i+x_{ijm}\cdot\frac{d_{ij}}{v_m}+D(t_i)\quad i,j=1,2,\cdots,|T|;m=1,2,\cdots,|P|, \tag{4}$$

式中 $D(t_i)$ 为任务 t_i 的处理时间, d_{ij} 为任务 t_i 与任务 t_j 间的空间距离:

$$d_{ij} = \sqrt{(x_j - x_i)^2 + (y_j - y_i)^2} \quad i,j = 1,2,\cdots,|T|,$$

(x_i, y_i)、(x_j, y_j)分别为任务 t_i 与任务 t_j 的地理位置.

综合(3)和(4)式,记 Y 为所有任务完成时间的上界(一般设置为较大的值),则作战平台在任务的分配以及任务间的顺序关系约束可如下式描述:

$$s_j \geq s_i + D(t_i) + x_{ijm} \cdot \left(\frac{d_{ij}}{v_m} + a_{ij} \cdot Y\right)' - a_{ij} \cdot Y', i,j = 1,2,\cdots,|T|; m = 1,2,\cdots,|P|. \quad (5)$$

当任务 t_i 与任务 t_j 之间存在顺序关系 $a_{ij} = 1$,且 $x_{ijm} = 1$ 时,(5)式就描述了平台在分配过程中的等待行为,即(4)式;当 $a_{ij} = 1$,且 $x_{ijm} = 0$ 时,(5)式可进一步表示如下式:

$$s_j \geq s_i + D(t_i) - a_{ij} \cdot Y', \quad i,j = 1,2,\cdots,|T|; m = 1,2,\cdots,|P|.$$

如果设置任务完成的初始值 Y 较大,上式显然是成立的.

任务 t_i 的成功处理的条件是被分配处理这一任务的所有平台资源的能力不小于任务 t_i 的需求 R_i,即

$$\sum_{m=1}^{|P|} s_{ml} \cdot w_{im} \geq r_{il} \quad i = 1,2,\cdots,|T|; l = 1,2,\cdots,|C|, \quad (6)$$

式中 n 为资源类型(案例中 $n = 8$). 令所有任务全部完成的时间为 Y,则对任意任务的处理的时间约束下式总能成立:

$$Y \geq s_i + D(t_i). \quad (7)$$

综上所述,战役计划过程可以描述如下式:

$$\begin{cases} \min Y \\ \sum_{j=0}^{|T|} x_{jim} - w_{im} = 0; \sum_{j=0}^{|T|} x_{ijm} \leq w_{im} \\ s_j \geq s_i + D(t_i) + x_{ijm} \cdot \left(\frac{d_{ij}}{v_m} + a_{ij} \cdot Y\right)' - a_{ij} \cdot Y' \\ d_{ij} = \sqrt{(x_j - x_i)^2 + (y_j - y_i)^2} \\ \sum_{m=1}^{|P|} s_{ml} \cdot w_{im} \geq r_{il} \\ Y \geq s_i + D(t_i) \\ 0 \leq Y \leq Y'; s_i \geq 0 \\ x_{ijk}, w_{ik} \in \{0,1\} \\ i,j = 1,2,\cdots,|T|; m,k = 1,2,\cdots,|P|; 1 \leq l \leq |C| \end{cases} \quad (8)$$

(8)式的数学描述是混元线性规划问题,式中包含了多个连续的二元变量. 这一问题的求解已被证明为 NP 难问题,涉及到一个众所周知的问题集:当可获取的平台资源仅有一个时,(8)式的求解是一个"货郎担"问题(TSP)和"货郎担"问题的扩展(如时序依赖 TSP,优先关系 TSP,等等);当所有的平台能处理所有的任务时,其求解问题又简化为一个伴随优先关系的多 TSP 问题;如果任务的处理可在不同平台之间适时分解,这一问题就是车辆路径规划问题和这一问题的扩展;如果平台在不同任务地点之间的行进时间远小于对该任务的处理时间(可以忽略不计的话),此问题就是一个伴随优先次序约束的多处理器调度问题.

3 计划问题的求解

对(8)式的求解可以转化为平台 - 任务分配的状态空间,通过算法进行搜索求解.

定义状态空间

$$\Phi = (M, LT_1, LT_2, \cdots, LT_K, f_1, f_2, \cdots, f_K)$$

其中 M 为所选任务集,$M \subset \{t_1, t_2, \cdots, t_N\}$,$Lt_j$ 是平台 p_j 最后处理的任务,$Lt_j \in \{0\} \cup M$,f_j 是平台 p_j 处理完最后的任务到达的时间. 每个状态描述了三个问题:一是任务集 M;二是任务集 M 中被各平台最后处理的任务集 $Lt_1, Lt_2, \cdots, Lt_K$;三是与 $Lt_1, Lt_2, \cdots, Lt_K$ 对应的平台处理最后任务所到达的时间 $f_1, f_2, \cdots, f_K$.

则一次完整的任务计划的状态空间 Φ 可表示为(N 为任务数量):

$\boldsymbol{\Phi} = \boldsymbol{\Phi}_1 \cup \boldsymbol{\Phi}_2 \cup \cdots \cup \boldsymbol{\Phi}_N$, $\boldsymbol{\Phi}_m = \{(M, LT_1, LT_2, \cdots, LT_K, f_1, f_2, \cdots, f_K) \in \boldsymbol{\Phi}, | M | = m\}$.

由此(1)式可简化为

$$Y = \min_{(M, LT_1, LT_2, \cdots, LT_K, f_1, f_2, \cdots, f_K)} \max\{f_1, f_2, \cdots, f_K\}. \tag{9}$$

状态 $\boldsymbol{\Phi}_m$ 到 $\boldsymbol{\Phi}_{m+1} = \{(M', Lt_1', Lt_2', \cdots, Lt_K', f_1', f_2', \cdots, f_K') \in \boldsymbol{\Phi}, | M' | = m+1\}$的转换通过在任务集中选择 M 中没有的任务分配到 M',该任务的所有前导任务都必须已在 M 中,否则不能被选入分配. 如果选入分配的任务为 T_j,且被分配平台组 $G_P(t_j)$ 来处理,记 t_j 的前导任务集为 $Pr(t_j)$,则新的状态可用(10)式表示:

$$\begin{aligned} &M' = M \cup \{t_j\}, \\ &Lt_i' = \begin{cases} Lt_i, & \text{if } p_i \notin G_P(t_j) \\ t_j, & \text{if } p_i \in G_P(t_j) \end{cases}, \\ &f_i' = \begin{cases} f_i, & \text{if } p_i \notin G_P(t_j) \\ d(t_j) + \max\left(\max\limits_{t \in Pr(t_j)} a(t), \max\limits_{p_z \in G_P(t_j)} (f_i + d_{Lt_i j}/v_z)\right) & \text{if } p_i \in G_P(t_j) \end{cases}. \end{aligned} \tag{10}$$

(10)式中 v_z 为处理任务 T_j 的平台 p_z 从任务 Lt_i 转到任务 t_j 的速度,$a(t)$ 表示任务 t 处理完到达的时间,$d(t)$ 为任务 t 的处理持续时间.

$\max\limits_{p_z \in G_P(T_j)} (f_i + d_{Lt_i j}/v_z)$ 表示任务 t_j 的处理必须等到所有处理该任务的平台到达任务 t_j 的处理地点时才能开始进行任务的处理.

求解(8)式的问题就转换为对(10)式的状态空间进行搜索,缩小状态空间,寻找最优解.

4　多优先级列表动态规划(MPLDS)算法

以上内容对计划问题进行了数学描述,并把计划问题的求解转化为状态空间的搜索. 在这一节我们对状态空间的搜索求解提出多优先级列表动态规划(MPLDS)算法,算法中定义任务选择平台、平台选择任务以及在任务图中选择任务的优先权,以及平台能力矢量与任务需求的矢量间距离,采用了任务选择平台、平台选择任务以及两者间冲突消除的搜索策略,并建立资源能力提供与完成时间的优先级列表.

4.1　三种优先权定义

在平台-任务分配过程需要选择当前用于的任务、任务对平台的选择以确立执行该任务的平台组、平台对任务的自主选择. 由此定义平台-任务分配的三种优先权:任务优先权系数 pr、任务选择平台的优先权 tp 和平台自主选择任务的优先权 pt. 三种优先权关系如图 3 所示.

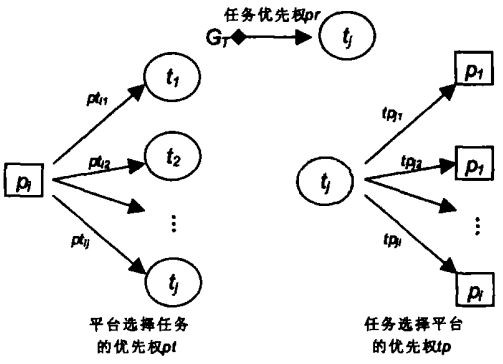

图 3　平台-任务分配的三种权系数

1) 任务优先权系数 pr

当某一任务的所有前导任务(即在该任务处理前必须完成的所有任务)都已处理完时,该任务便进入可分配的任务集 Ready 中,在 Ready 中选择任务优先权系数高的任务首先进行平台分配,任务的优先权系数是对任务图 G_T 中的任务序列依据算法来确定,这些算法包括关键路径算法 CP,层次分配算法 LA 和加权长度算法 WL[3]. 记 N 为任务图 G_T 边的数量,则三种算法的复杂性都为 $O(N)$. 本文结果采用了加权长度算法 WL 来计算任务的优先权系数 pr:

$$pr(i) = D(t_i) + \max_{j \in OUT(i)} pr(j) + \sum_{j \in OUT(i)} pr(j) \Big/ \max_{j \in OUT(i)} pr(j), \tag{11}$$

式中 $out(i)$ 表示任务 t_i 的后续任务集,$D(t_i)$ 为任务 t_i 的处理时间.

2) 任务选择平台的优先权 tp

任务选择平台一方面要最小化任务的完成时间,另一方面聚集的平台资源充分利用. 最小化任务的完成时间需要选择能在较短时间内到达任务区域进行任务处理的平台,而聚集平台资源的充分利用需要最小化聚集资源的冗余,即聚集的平台资源正好满足任务的资源需求是最佳的平台聚集.

由此,定义任务选择平台的时间优先系数 tp_1 和平台资源矢量距离优先系数 tp_2.

$$tp_1(m) = (s_{l(m)} + D(t_{l(m)}) + d_{l(m),i}/v_m),$$
$$tp2(m) = \| S_m - R_i \| = \sqrt{\sum_{l}^{Cl}(s_{ml} - r_{il})^2}. \tag{12}$$

(12)式中 $l(m)$ 为平台 m 最后处理的任务.

时间优先系数是平台在处理完最后的任务后运行到当前需要处理的任务区域的到达时间,平台资源矢量距离优先系数是平台资源能力与任务需求的满足程度.

在平台的分配过程中存在两种平台:一种是已经被分配处理过任务的平台;另一种是还没有参与分配的平台.

记第一类平台为 P_a,第二类平台为 P_b,已经执行的任务为 T_a,未执行的任务为 T_b,则

$$\text{if } p_i \in P_b \text{ then } p_i \notin Gp(t_k)(t_k \in T_a).$$

在分配过程中的第一类平台需要在不同的任务区域转换,而第二类平台是可以随时调用的平台,不需要从一个区域运行到另一个任务区域,因此,对 P_b 中的平台我们认为时间优先系数 $tp_1 = 0$,优先权只需要计算其平台资源矢量距离优先系数 tp_2,即

$$pt_1(m) = 0, \text{if } p_m \in P_b.$$

时间优先系数 tp_1 和平台资源矢量距离优先系数 tp_2 属于两个不同的概念,不能简单地将它们进行加权运算,由此,按照 tp_1 和 tp_2 升序建立任务选择平台的优先级表[12],即平台到达任务区域时间越早越靠前,平台资源能力矢量与任务资源需求矢量距离越近越靠前.

令 i,j 分别为平台时间优先系数 tp_1 和平台资源矢量距离优先系数 tp_2 在各自序列中的位置,则任务选择平台的优先权 tp 可以按下式计算:

$$tp = (i + j - 1) * (i + j - 2) + i,$$

tp 越小,则平台的优先级越高.

3) 平台自主选择任务的优先权 pt

在平台-任务分配过程定义任务的三种状态:已经处理的任务、正在处理的任务和还没有处理的任务,令处于三种状态的任务集合分别为 $ST1$,$ST2$,$ST3$,则平台自主选择任务的优先权 pt 是对 $ST3$ 任务集中的任务进行优先级的排序.

同样,定义平台选择任务的时间优先系数 pt_1 和任务资源需求矢量距离优先系数 pt_2.

对某一个具体的平台 p_m,其对任务选择的时间优先系数只需要考虑平台当前处理任务的区域到达等待处理的各任务间的距离,由此,两种优先系数可定义如下式:

$$pt_1(m) = d_{l(m),i}/v_m, \quad t_i \in ST3,$$
$$pt_2(m) = \| S_m - R_i \| = \sqrt{\sum_{l}^{Cl}(s_{ml} - r_{il})^2}, \quad t_i \in ST3. \tag{13}$$

令 h, k 分别为优先系数 pt_1 和 pt_2 在各自序列中的位置,则平台选择任务的优先权 pt 可以按下式计算:

$$pt = (h + k - 1) * (h + k - 2) + i.$$

4.2 消除优先选择冲突

对所选任务 T_i 上一节确定了对平台选择的优先权系数,由优先权定义可知其时间优先系数 pt_1 在任务确定后的排序是不变的;由于任务的资源需求在平台的选择过程在不断变化,直到选择出最佳平台组使得任务需求矢量为 0,因此,矢量距离优先系数 tp_1 是随着任务对平台的选择而变化的.也就是说任务对平台选择的优先级列表是动态的.

对于平台 p_m 对任务选择的优先级列表是静态的,在确定了平台和等待处理的任务集 $ST3$ 后就确定了平台 p_m 对 $ST3$ 中任务选择的优先级列表.

由于任务对平台的选择是短视的、局部的,只考虑任务自身的需求,而平台对任务的选择是全局的,是

对所有未处理任务的优选,因此,任务对平台的选择与平台对任务的选择经常是冲突的,也就是说 tp 与 pt 是不一致的,解决二者之间的冲突是平台-任务最佳分配的关键问题. 二者之间存在以下几种情况:

1) 任务-平台的最佳选择:仅当平台 p_m 被任务 T_i 选择的优先权系数 tp_{im} 与任务 T_i 被平台 p_m 选择的优先权系数 pt_{mi} 同时最小时.

2) 任务-平台的互补选择: tp_{im} 有较高的优先等级(从 1 到| P_a|),而 pt_{mi} 的优先级较低(从 1 到| $ST3$|);或者反之.

3) 任务-平台的最坏选择: tp_{im} 和 pt_{mi} 的优先级较低.

确定第二种情况下的任务选择平台的优先级就解决了两种选择的冲突问题. 在任务-平台的选择过程中我们期望选择 tp_{im} 优先级高而 pt_{mi} 优先级也尽可能高. 记任务对平台选择的动态优先级列表为 DL,平台对任务选择的静态优先级列表为 SL, Stp 和 Spt 分别为 tp_{im} 和 pt_{mi} 在 DL、SL 中的排序,由此,我们采用加权方法解决任务-平台优先选择的冲突. 记 $PR(tp, pt)$ 两种选择优先权的协调,则加权方法可定义如下:

$$PR(tp, pt) = (\lambda \cdot (Stp - 1 - \beta) + 2 \cdot Spt - 2) \cdot (Stp + \beta)/2 + Stp$$

其中 λ 为权系数, $\beta = \lceil (Spt - 2)/\lambda \rceil$.

4.3 MPLDS 算法流程

算法的主要工作包括以下三部分:

1) 分配可行性检查;

2) 从任务图 G_T 中根据优先权选择要处理的任务;

3) 选择处理任务的最佳平台组,最佳平台的选择.

分配可行性检查是对从当前的平台-任务处理状态中选择的任务的可分配性检查,其中第三步最佳平台组的选择包括:任务对平台的选择、平台对未处理任务的选择以及两种选择冲突的消除. 算法流程如图 4 所示(图中 $G(i)$ 为对任务 i 处理聚集的平台资源).

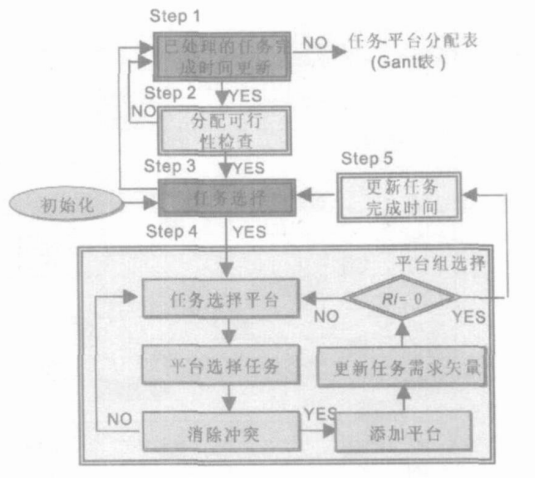

图 4　MPLDS 算法流程

5　结论与讨论

对本文案例的求解结果如图 5 所示,图 5a 是文献[8,9]所采用传统 MDLS 方法的求解结果,图 5b～5c 是本文 MPLDS 方法通过不同的加权消除冲突方法的求解结果,各分配方案比较了全部任务完成的截止时间和平台执行任务过程总的协作量. 全部任务完成的截止时间即最后一个任务完成到达的时间;平台在任务上的协作指在同一任务的执行上需要多个作战平台的协同,在任务执行上的协作总量指所有任务的执行所导致的协同作战平台的数量. 平台间的交互协作过多时降低了完成任务的效能,不必要的交互甚至会导致任务处理出错,在平台-任务的分配过程中期望减少作战平台在任务的执行上不必要的协作,即某一平台的能力能满足某一任务的能力需求时就不用选择多个平台来协同完成这一任务.

由两种方法的结果比较可知,通过平台的自主选择与任务对平台选择两者的结合使得任务计划过程更为紧凑,节省了全部任务处理的完成时间,同时也充分地利用了战场平台资源,提高了作战平台资源利用率;另一方面,不同分配方案的协作总量不同,MPLDS 算法的求解结果减少了平台在任务的执行过程中不必要的协作.

在任务-平台分配过程中,任务选择平台与平台自主选择任务的不同加权处理方法消除二者冲突也导致了调度方案的不一样,如图 5b,5c 所示.

在 MPLDS 和 MDLS 算法的复杂性上不同的任务总数有不同的计算复杂性,计算时间随任务总数变化的曲线图如图 6 所示. 由图 6 可以看出在任务总数较小的情况下 MPLDS 的性能优于 MDLS,而当任务总数较多时 MPLDS 性能劣于 MDLS. 由此,尽管本文提出的 MPLDS 方法对任务计划问题的解决有更为理想的结果,但在任务总数较多时却牺牲了算法的复杂性.

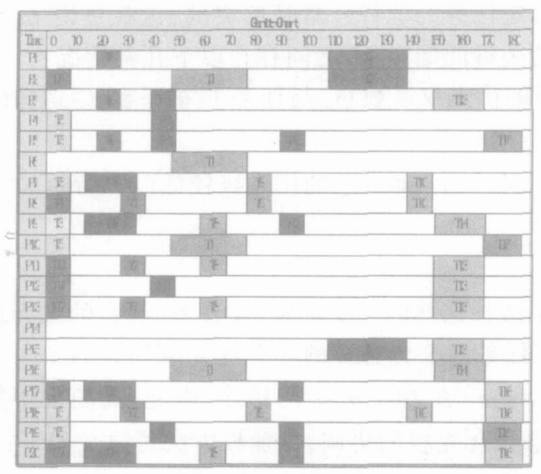

图 5a 案例的分配方案
(MDLS 方法;截止时间:184.5;协作总量:65)

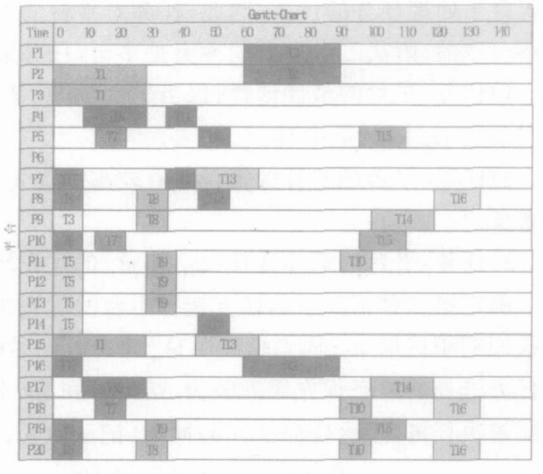

图 5b 案例的分配方案
(MPLDS 方法,λ = 1;截止时间:133.8;协作总量:47)

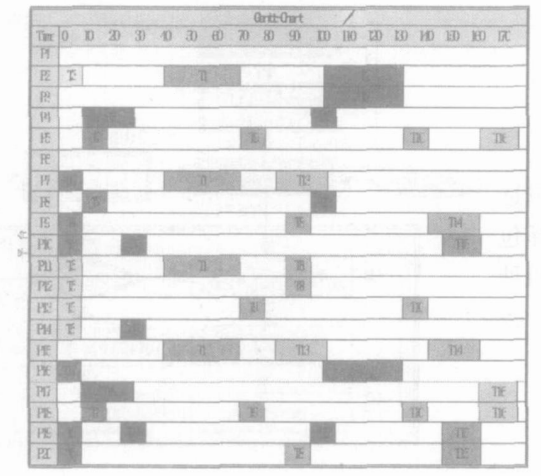

图 5c 案例的分配方案
(MPLDS 方法,λ = 5;截止时间:176.5;协作总量:65)

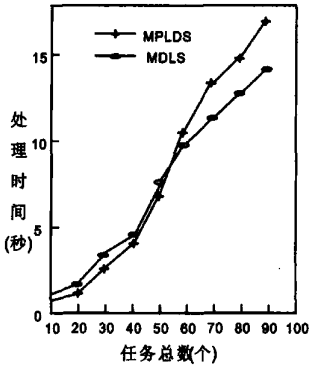

图 6 MPLDS 与 MDLS 计算性能比较

参考文献:

[1] Ramamritham K J, Stankovic A, Shiah P - F. Efficient scheduling algorithms for real-time multiprocessor systems[J]. IEEE Transactions on Parallel and Distributed Systems, 1990,1(2):184 - 194.

[2] Manimaran G, Murthy C S R. An efficient dynamic scheduling algorithm for multiprocessor real-time systems[J]. IEEE Transactions on Parallel and Distributed Systems, 1998,9(3):312 - 319.

[3] Shirazi B, et al. Analysis and evaluation of Heuristic methods for static task scheduling [J]. J. of Parallel and Distributed Computing, 1990,10:222 - 232.

[4] Muthucumaru Maheswaran and Howard Jay Siegel. A Dynamic Matching and Scheduling Algorithm for Heterogeneous Computing Systems[M]. HCW '98, pages 57 - 69, Orlando, USA, March 1998. IEEE Computer Society Press.

[5] Hyunok Oh and Soonhoi Ha. A Static Scheduling Heuristic for Heterogeneous Processors. Proceedings of Europar '96[M]. Volume 1124 of Lecture Notes in Computer Science, pages 573 - 577, Lyon, France, August 1996. Springer-Verlag.

[6] Volker Strassen. Gaussian elimination is not optimal[J]. Numerische Mathematik, 1969,14(3):354 - 356.

[7] Gilbert Sih and Edward Lee. A compile-time scheduling heuristic for interconnection constrained heterogeneous processor architectures[J]. IEEE Transactions on Parallel and Distributed Systems, 1993,4(2):175 - 187.

[8] Levchuk Georgiy M, et al. Normative design of organizations-part I: Mission planning [J]. IEEE Transactions on Systems, man, and Cybernetics-part A:Systems and Humans, 2002,32(3):346 - 359.

[9] Levchuk G M, Levchuk Y N, Luo J, et al. Pattipati, A library of optimization algorithms for organizational design[A]. Proceedings of the 2000 Command and Control Research and Technology Symposium[C]. Monterey, CA: NPS, June 2000.

[10] Kleinman D L, Young P, Higgins G S. The DDD-Ⅲ: A tool for empirical research in adaptive organizations[A]. Proc. Command and Control Research and Technology Symp [C], Monterey, CA, June 1996.