doi:10.3969/j.issn.1671-637X.2014.10.007

联合作战资源的再调度算法仿真

谢 斌, 林 华
(海军工程大学电子工程学院,武汉 430033)

摘 要: 资源调度是联合作战中的一个重要环节,各种作战资源合理的编成将为作战任务的顺利完成提供保障。但是,在作战过程中,无法保证资源任何时刻都处于良好的状态,一旦资源无法使用将影响整个作战任务的完成。文章改进了多维动态列表(MDLS)算法并进行了仿真,使某些资源在无法使用时能够进行再调度以完成任务。

关键词: 联合作战; 作战资源; 再调度; 算法; 仿真
中图分类号: V271.4 **文献标志码**: A **文章编号**: 1671-637X(2014)10-0028-05

Simulation for Joint Operation Resource Rescheduling Algorithm

XIE Bin, LIN Hua
(College of Electronic Engineering, Naval University of Engineering, Wuhan 430033, China)

Abstract: Resource scheduling is an important part for joint operation, and reasonable combination of all kinds of operation resources will supply a guarantee to the operation. But during the operation, it is difficult to keep all the operation resources in good condition all the time. Once the resources can not be used, it will influence the accomplishment of the whole mission. In this paper, the Multi-Dimensional List Scheduling (MDLS) algorithm is improved and simulation is made, thus to reach the goal of resource rescheduling and guarantee the accomplishment of mission when certain resources are unavailable.

Key words: joint operation; operation resources; rescheduling; algorithm; simulation

1 问题描述

调度问题是如何在限定的时间内利用有限的资源执行一组给定的动作[1]。联合作战的资源调度是根据作战任务和现有的可调用资源情况,对作战资源进行调度。调度方案包括了任务开始时间、结束时间以及所分配到的资源等信息[2]。联合作战中资源的调度是对各作战资源进行合理的编成编组,使各作战资源发挥出最大的作战效能。可以说,联合战场资源的调度是保证联合作战任务完成的一大关键,它的一般流程如图1所示[3]。

在作战资源完成初始分配之后,由于在资源使用过程中可能会出现突发的情况,如战斗损伤等,这就需要对作战资源进行再调度以保证完成任务。

作战资源在作战过程中随时会面临战斗损伤、故障以及其他情况,这些突发情况会导致资源暂时性或永久性无法使用。一旦发生这种情况,原先资源所保障的作战任务就无法顺利完成,因此需要对资源进行再调度。

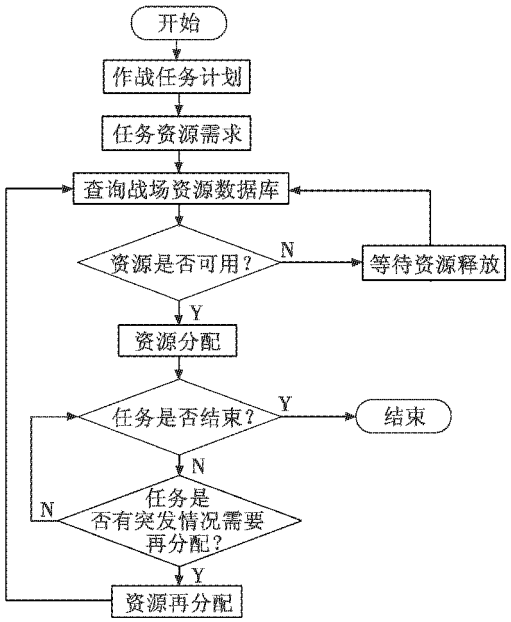

图1 作战资源的调度流程

Fig. 1 The flow chart of joint operation scheduling

收稿日期:2013-11-13 **修回日期**:2013-11-22
基金项目:总装"十二五"预研项目(51306030302)
作者简介:谢 斌(1989—),男,广东兴宁人,硕士生,研究方向为舰艇作战系统工程。

资源状态是反映资源在当前时刻下是否能正常工作的一个指标，也是作战资源再调度问题的一个基础，通过资源状态这个指标可以判断资源是否可以满足任务需求，是否需要进行再调度。在作战资源分配后，应继续监视资源的状态，若资源一直运转良好，资源可以完成现在的任务后进行释放，等待执行下一个任务；若资源在执行任务过程中有突发的情况发生，任务进入资源再分配流程，对资源进行重新调度。

为了便于监视资源的状态，本文将资源的状态分为“可用”、“忙”、“不可用”3 种情况，分别用数字 1、0、-1 与之对应。其中，“可用”表示在当前时刻，资源处于空闲状态，可以分配给任务；“忙”表示在当前时刻，资源已经分配，无法分配给其他任务；“不可用”表示在当前时刻，资源可能遇到突发情况，如故障、战斗损伤等导致无法使用。当某个正在执行任务的资源中状态标识出现了 $s=-1$，表示资源出现了不可用的状况，该任务应该进行资源再分配，才能够保证任务顺利完成。

2　改进的多维动态列表算法

作战资源调度问题已经被证明是一类 NP-hard 问题[3]，解决这类问题一般使用启发式算法进行求解，多维动态列表（MDLS）算法是一种求解作战资源调度的算法，由 Levchuk[4] 等人提出，文献[2,5-6]对算法进行过改进，但改进的出发点大多集中于优化任务完成时间和改进平台优先权。本文主要从资源状态改变进行考虑，改进算法流程使其在资源出现突发情况时能对资源进行再调度。

2.1　原始的 MDLS 算法

原始的 MDLS 算法主要包括 2 个步骤：1）任务的选择；2）平台资源的选择[2]。任务的选择是根据各种优先权计算公式，计算出各个任务的优先权，然后按照优先权的大小进行任务的安排和资源的分配，优先权高的任务先分配资源并优先执行，优先权低的任务则后执行。选定任务之后，算法进入平台资源选择阶段。在这一阶段，平台根据优先权计算公式计算平台资源的优先权，优先权较高的平台先分配给当前选定的任务，选好平台后再进行平台裁剪，直到每一个平台对于当前任务来说都是不可或缺的。

2.2　改进的 MDLS 算法

原始 MDLS 算法的一大特点是假设在整个作战任务的任何时刻，所有的平台作战资源都是完好无损可以调用的，每一次任务结束之后，所有该任务调用的资源都释放返回可调用资源的集合，等待执行下一次任务。

原始 MDLS 算法的弊端显而易见，作战资源在实际作战过程中不可能时刻都保持在完好状态，总会受到故障、战斗损伤的影响，一旦资源出现这些突发情况，算法无法进行及时的调度处理，将会对任务的执行产生很大影响；另一方面，作战过程中故障的资源可能通过抢修后可以重新投入使用，因此应该及时检查故障资源是否能够重新调用，以便使资源利用率最大化。

在平台资源模型中加入状态信息，在原始的 MDLS 算法中加入故障资源判断以及已经开始执行的任务中是否有任务需要资源再分配这两个步骤，每次时间更新后，进行故障资源的判断：即在无法调用的资源集合中查找是否有故障排除能够重新投入使用的资源，若有则将其重新放入可调用的资源集合；再查看已经开始执行的任务中是否有因为资源损毁或故障需要重新进行资源调度的任务，若有则进行资源重新调度。

2.3　改进的 MDLS 算法流程

改进后的 MDLS 算法流程如图 2 所示。

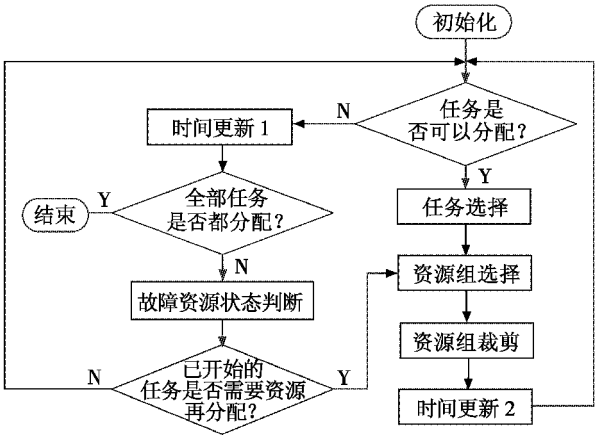

图 2　改进后的 MDLS 算法流程

Fig. 2　The flow chart of the improved MDLS algorithm

算法步骤中使用的各个符号注释如下：READY 为就绪任务集合，即没有前序任务或前序任务都已经完成的任务集合；TEMP 为当前选定的任务集合；TEMP1 为已经开始执行的任务集合；FREE 为可以调用的资源集合；BROKEN 为无法调用资源的集合，包括战斗损毁资源和故障资源。

初始化阶段假设所有资源的状态都是良好的，都可以调用，将状态标识置 1。FREE 集合中包含了所用资源，没有前序任务的任务进入 READY 集合。

1）完成时间更新 1。若已经分配的任务数等于整个任务数，则结束程序；否则从 TEMP1 中选择最接近完成的任务进行更新操作，将其调用的平台都释放，重新进入 FREE，再观察该任务后续任务中是否有前序任务已经全部完成的任务，若有，则将其添加进 READY，然后转入步骤 2）。

2）故障资源状态判断。在时间更新后，在 BROKEN 中逐个寻找是否有故障排除，即状态标识重新置 1 的资源。若有，将这些资源添加入 FREE 中，用以之后任务的调用。完成步骤 2）后进入步骤 3）。

3）已开始的任务是否需要资源再分配。在 TEMP1 中查看已经开始的任务中执行任务的资源是否有出现损毁或故障情况，即查看资源状态标识是否出现了被置为 -1 的情况。若有，则该任务重新进入 TEMP 集合，并在 TEMP1 集合中将该任务删除并进入步骤 6）；若没有则进入步骤 4）。

4）分配可行性检测。检测 FREE 中的资源是否能够满足至少一个 READY 中任务的执行。若不满足返回步骤 1）；满足则进入步骤 5）。

5）任务选择。若 READY 为空集合，则返回步骤 1）；若 READY 不为空，则根据任务的优先权函数计算出 READY 中各个任务的优先权，然后选择优先权最大的任务进入 TEMP，并将其从 READY 中删除，然后进入步骤 6）。

6）资源选择。根据步骤 5）选定的任务，即 TEMP 中的任务，在 FREE 中根据各资源的优先权函数计算资源的优先权，并添加到处理任务的资源组，直到 TEMP 中任务所需资源都得到满足，停止添加。资源选择完成后进入步骤 7）进行资源裁剪。

7）资源裁剪。在步骤 6）中确定的资源组集合中进行资源的裁剪，裁剪的方法是，若去掉某一个资源后剩下的资源都能满足任务的资源需求，则把该资源裁剪掉，否则保留该资源。裁剪后的资源组必须保证每一个资源对于 TEMP 中的任务来说都是不可或缺的。完成后进入步骤 8）。

8）完成时间更新 2。将被调用的资源从 FREE 中删除，更新资源组中的每个资源最后执行的任务，将资源组中每个资源的状态标识置 0，并更新任务的开始时间，输出资源分配方案包括任务代号、任务开始时间、任务结束时间以及任务分配到的资源代号，然后返回步骤 3）再次循环操作。

3 仿真及结果

3.1 仿真设定

本文采用美军 A2C2 实验 7——联合登岛作战实验的数据进行仿真，该实验包含了 18 个任务，20 个平台资源[4]。

图 3 为作战任务图，图中显示了各作战任务时序关系。任务属性表和资源属性表则分别包含了任务资源需求和处理时间，资源名称、速度以及资源对应任务各项需求的能力等信息，如表 1、表 2 所示。

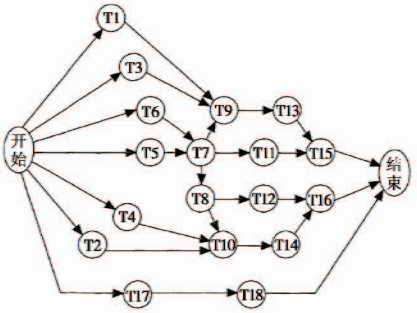

图 3 作战任务图

Fig. 3 Mission graph

表 1 任务属性表

Table 1 Task attribute

任务	资源需求								处理时间/h
	r_1	r_2	r_3	r_4	r_5	r_6	r_7	r_8	
T1	5	3	10	0	0	0	0	6	30
T2	0	3	10	0	0	0	0	6	30
T3	0	3	0	0	0	0	0	0	10
T4	0	3	0	0	0	0	10	0	10
T5	0	3	0	0	0	0	0	0	10
T6	0	0	0	10	14	12	0	0	10
T7	0	0	0	10	14	12	0	0	10
T8	0	0	0	10	14	12	0	0	10
T9	5	0	0	0	0	5	0	0	10
T10	5	0	0	0	0	5	0	0	10
T11	0	0	0	0	0	10	5	0	10
T12	0	0	0	0	0	10	5	0	10
T13	0	0	0	0	0	8	0	6	20
T14	0	0	0	0	0	8	0	6	20
T15	0	0	0	20	10	4	0	0	15
T16	0	0	0	20	10	4	0	0	15
T17	0	0	0	0	0	8	0	4	10
T18	0	0	0	8	6	0	4	10	20

表 2 资源属性表

Table 2 Resource parameters

ID	Platform Name	AAW	ASUW	ASW	GASLT	FIRE	ARM	MINE	DES	Velocity
1	DDG	10	10	1	0	9	5	0	0	2
2	FFG	1	4	10	0	4	3	0	0	2
3	CG	10	10	1	0	9	5	0	0	2
4	ENG	0	0	0	2	0	0	5	0	4
5	INFA	1	0	0	10	2	2	1	0	1.35
6	SD	5	0	0	0	0	0	0	0	4
7	AHI	3	4	0	0	6	10	1	0	4
8	CAS1	1	3	0	0	10	8	1	0	4
9	CAS2	1	3	0	0	10	8	1	0	4
10	CAS3	1	3	0	0	10	8	1	0	4
11	VF1	6	1	0	0	1	1	0	0	4.5
12	VF2	6	1	0	0	1	1	0	0	4.5
13	VF3	6	1	0	0	1	1	0	0	4.5
14	SMC	0	0	0	0	0	0	10	0	2
15	TARP	0	0	0	0	0	0	0	6	5
16	SAT	0	0	0	0	0	0	0	6	7
17	SOF	0	0	0	6	6	0	1	10	2.5
18	INF (AAAV-1)	1	0	0	10	2	2	1	0	1.35
19	INF (AAAV-2)	1	0	0	10	2	2	1	0	1.35
20	INF (MV22-1)	1	0	0	10	2	2	1	0	1.35

实验采用 VC ++ 6.0 平台进行仿真，仿真计算机 CPU 主频 1.6 GHz。仿真步长的设定值从理论上说应尽可能小，因为对资源的状态监视总是期望是实时的，一旦资源状态出现变化，可以尽可能早地做出反应。在这里设定仿真步长为 0.001 h。实验中平台优先权算式采用文献[4]中的 R^4。

$$R^4 = \frac{B_R(m) - B(m,i)}{B(m,i)} \tag{1}$$

式中：$B(m,i)$ 为平台 m 所提供的能满足任务 i 的各类资源的总和；$B_R(m)$ 为平台 m 对 READY 集合中任务的资源满足程度。

$$\begin{cases} B(m,i) = \sum_{i=1}^{L} \min(s_{ml}, r_{il}) \\ B_R(m) = \sum_{i \in \text{READY}} B(m,i) \end{cases} \tag{2}$$

式中，s_{ml} 为对某类资源 i、平台 m 能提供给任务 l 资源 i 的数量，任务 l 所需资源 i 的数量为 r_{il}。

采用 R^4 是因为其计算较为简便，不涉及平台移动等情况，可以较快输出结果，便于观察再调度情况。

实验中任务优先权的计算采用加权长度计算方法，任务优先权算式为

$$P(i) = C_P(i) + \max_{j \in O_{\text{OUT}}(i)} C_P(j) + \frac{\sum_{j \in O_{\text{OUT}}(i)} C_P(j)}{\max_{j \in O_{\text{OUT}}(i)} C_P(j)} \tag{3}$$

式中：$O_{\text{OUT}}(i)$ 为任务 T_i 的后续任务集；$C_P(i)$ 为任务 T_i 的关键路径，等于任务 T_i 到任务流程结束时所需要的最短时间[2]；$C_P(j)$ 为任务 T_i 的后续任务集合 $O_{\text{OUT}}(i)$ 中的任务 T_j 的关键路径。

3.2　仿真结果

在假设资源全部正常的情况下，输出仿真结果，如表 3 所示。

表 3　资源正常情况下的调度方案

Table 3　Scheduling result when all the resources are available

任务代号	开始时间/h	结束时间/h	分配平台
T1	0	30	P1 P2 P15
T3	0	10	P11 P12 P13
T4	0	10	P3
T5	0	10	P7 P14
T6	0	10	P8 P18 P5
T17	0	10	P9 P16
T2	60.150	90.150	P3 P16 P2
T7	30	40	P20 P8 P10
T18	30	50	P17 P4
T8	40	50	P9 P18 P19
T9	65.805	75.805	P1
T11	40	50	P7 P14
T13	89.885	109.885	P1 P11 P12 P13 P15
T12	75.806	85.806	P10 P8 P4
T10	90.150	100.150	P9 P6
T14	100.150	120.150	P7 P17
T15	109.886	124.886	P20 P5 P10
T16	120.150	135.150	P18 P19 P9

为了测试算法的再调度模块，假设在 t = 35 h 时刻资源 P4 发生了故障而不可用，程序输出再调度方案，如表 4 所示。

表 4　资源 P4 在 t = 35 h 时刻发生故障生成的再调度方案

Table 4　Rescheduling result when resource P4 is unavailable at t = 35 h

任务代号	开始时间/h	结束时间/h	分配平台
T1	0	30	P1 P2 P15
T3	0	10	P11 P12 P13
T4	0	10	P3
T5	0	10	P7 P14
T6	0	10	P8 P18 P5
T17	0	10	P9 P16
T2	60.150	90.150	P3 P16 P2
T7	30	40	P20 P8 P10
T18	30	50	P17 P4
资源 P4 发生故障进行再调度			
T18	41.605	61.605	P7 P9 P17 P19
T8	42.500	52.500	P8 P5 P18
T9	65.805	75.805	P1
T11	60.859	70.859	P20 P10 P14
T13	89.885	109.885	P1 P13 P12 P11 P15
T12	97.785	107.785	P5 P8 P14
T10	90.150	100.150	P7 P6
T14	100.150	120.150	P9 P17
T15	109.886	124.886	P10 P20 P19
T16	121.298	136.298	P5 P18 P3

从仿真结果来看，程序在资源发生故障之后可以重新输出资源的再调度方案，再调度方案仍然以满足任务的需求为目标，调用在当前 FREE 中的资源重新开始执行被中断的任务。任务 T18 需求 r_4 = 8、r_5 = 6、r_7 = 4 和 r_8 = 10 在 P4 不可用后，调用 P9、P17、P19 3 个资源仍然可以满足任务执行的要求。从仿真结果来看，某一正在执行的任务的资源若发生故障或其他不可用的情况，在资源数量有限的情况下，将会推迟整个作战任务的完成时间，也会影响到后续其他任务的资源调用。

在实际作战过程中，作战资源分布在各个区域，可建立战场资源属性数据库[7]，数据库包含了作战资源的各种信息，对分布式战场资源进行管理。在资源发生不可用情况时，资源属性数据库的状态信息发生变化，通过数据链，可以实现不同军兵种资源状态信息的传输[8]，这些信息汇总到一个指挥点后，再通过相应的算法即可输出再调度的方案。

4　结束语

本文通过改进 MDLS 算法流程，在作战资源出现不可用情况下可生成再调度方案。但是改进的 MDLS 算法无法估量故障资源执行的任务完成情况，现在的再调度方案都是让任务重新开始执行，这势必会造成作战资源的浪费，延长整个作战任务的完成时间。实际作战过程中，战场资源都是分布在各个区域的，现在

的仿真只是在单一平台上进行,下一步应努力实现分布式的作战资源再调度。此外,现在的仿真都是建立在任务既定的基础之上,所有设定任务的属性都是固定不变的,但是在实际作战过程中,战斗情况瞬息万变,各任务随着时间的推移对资源的需求也在变化中,而在一些特殊情况下,紧急任务的插入也会造成资源需要重新调度,因此,如何在动态变化的情况下实现资源的再调度是下一步重点研究的方向。

参考文献

[1] GHALLAB M, NAU D, TRAVERSO P. Automated planning: Theory and practice[M]. San Francisco: Morgan Kaufmann, 2004.

[2] 鲁音隆. 多兵种联合作战战役任务计划方法研究[D]. 长沙:国防科学技术大学,2004.
LU Y L. Research on algorithm of task planning in joint campaign[D]. Changsha: National University of Defense Technology, 2004.

[3] 谢斌,林华. 联合战场资源调度问题综述[J]. 舰船电子工程,2013,33(10):23-26.
XIE B, LIN H. Survey on joint battlefield resources scheduling problem[J]. Ship Electronic Engineering, 2013, 33(10):23-26.

[4] LEVCHUK G M, LEVCHUK Y N, LUO J, 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):346-357.

[5] 包卫东,王江峰,张茂军. 一种改进的基于MDLS与GA的作战资源分配算法[J]. 火力与指挥控制,2008,33(9):18-21.
BAO W D, WANG J F, ZHANG M J. A novel algorithm of task resource distribution based on MDLS and GA[J]. Fire Control and Command Control, 2008, 33(9):18-21.

[6] 张杰勇,姚佩阳,周翔翔,等. 基于DLS和GA的作战任务——平台资源匹配方法[J]. 系统工程与电子技术,2012,34(5):947-954.
ZHANG J Y, YAO P Y, ZHOU X X, et al. Approach to operation task and platform resource matching based on DLS and GA[J]. Systems Engineering and Electronic, 2012, 34(5):947-954.

[7] 林华,周翔,董银文. 海战场资源属性数据库的建立[J]. 舰船电子工程,2013,33(4):101-102.
LIN H, ZHOU X, DONG Y W. Establishment of naval field resources attribute database[J]. Ship Electronic Engineering, 2013, 33(4):101-102.

[8] 梅文华,蔡善法. JTIDS/Link16数据链[M]. 北京:国防工业出版社,2007.
MEI W H, CAI S F. JTIDS/Lin16 data link[M]. Beijing: National Defense Industry Press, 2007.

(上接第23页)

[10] 钱杏芳,林瑞雄,赵亚男. 导弹飞行力学[M]. 北京:北京理工大学出版社,2008:49-57.
QIAN X F, LIN R X, ZHAO Y N. Missile flight dynamics[M]. Beijing: Beijing Institute of Technology Press, 2008: 49-57.

[11] 陈小庆,侯中喜,刘建霞. 基于直接配点法的滑翔轨迹快速优化设计[J]. 航空计算技术,2010,40(1):37-41.
CHEN X Q, HOU Z X, LIU J X. Reentry trajectory optimization for hypersonic glide vehicle[J]. Aeronautical Computing Technique, 2010, 40(1):37-41.

(上接第27页)

[4] MOON G Y, KIM Y D, CHO S B. Variable structure control with optimized sliding surface for aircraft control system[Z]. AIAA 2004-5420.

[5] 罗生. 最优滑模制导律设计与仿真[J]. 航空兵器,2012(1):34-37.
LUO S. The optimal sliding mode guidance law design and simulation[J]. Aero Weaponry, 2012(1):34-37.

[6] 黄汉桥,黄长强,赵辉,等. 考虑前馈作用的BTT导弹自动驾驶仪设计方法研究[J]. 西北工业大学学报,2012,30(3):307-313.
HUANG H Q, HUANG C Q, ZHAO H, et al. An effective design method of BTT missile autopilot considering feedforward effect[J]. Journal of Northwestern Polytechnical University, 2012, 30(3):307-313.

[7] 周凤岐,周军,郭建国. 现代控制理论基础[M]. 西安:西北工业大学出版社,2011:200-218.
ZHOU F Q, ZHOU J, GUO J G. Modern control theory[M]. Xi'an: Northwestern Polytechnical University Press, 2011:200-218.

[8] ZHAO S Y, ZHOU R, WEI C. Design and feasibility analysis of a closed-form guidance law with both impact angle and time constraints[J]. Journal of Astronautics, 2009, 30(3):1065-1072.