一种基于规划合并的作战方案优化方法

李 皓 常国岑 杨 凡
(空军工程大学电讯工程学院 西安 710077)

摘 要 高效的作战方案是对敌实施有效打击的保证。运用 Agent 技术,提出一种规划合并机制,并将其运用于作战方案的优化。通过资源共享,将其它规划产生的可共享资源取代本规划中某些动作的输出,从而删除本规划中的冗余动作,提高规划执行效率,降低执行成本。最后给出规划合并的算法,采用倒序处理动作的方式,使得在实现目标的前提下,最大程度地删除规划中的冗余动作。

关键词 作战方案,规划合并,资源共享,智能体

Method for Operational Plan Optimization Based on Plan Merging

LI Hao CHANG Guo-cen YANG Fan
(Telecommunication Engineering Institute, AFEU, Xi'an 710077, China)

Abstract The efficient operational plan is the guarantee of the effective attack towards enemy. Based on agent, a plan merging mechanism was put forward and applied in optimizing the operational plan. By resource sharing, a plan could use sharable resource made by other plans to replace its some actions' outputs, and then delete its redundant actions for the purpose of increasing execution efficiency and reducing its cost. In the end, the algorithm of plan merging was given. In which actions were dealt with in reverse order. As a result, the redundant actions were deleted to the fullest extent.

Keywords Operational plan, Plan merging, Resource sharing, Agent

1 引言

制定作战方案是作战准备与实施阶段一项十分重要的工作,是实现对敌有效打击的关键。20 世纪 80 年代以来的局部战争表明,现代战争对作战方案的要求越来越高[1],对作战方案的优化显得格外重要。随着智能体(Agent)技术的迅速发展,它在军事领域的应用也愈加广泛。本文运用 Agent 规划技术来实现对作战方案的优化。

在 Agent 领域,规划是一种通过模拟人类求解复杂问题的过程而形成的一种方法[2]。它的本质是自动程序设计,Agent 根据意图、信念和可采取的动作通过规划算法产生一系列的动作——规划[3]。规划合并是一种分布计划的分布规划方法,其基本思想是假定全局目标已分配给多个主体,各主体根据子任务单独进行规划,然后通过共享和交换局部计划,加入约束条件以保证各自计划执行时相互不冲突[2]。当 Agent 执行规划的时候,还可以通过规划合并来消除冗余,提高执行效率[4]。在规划合并研究领域,已有许多学者做出了很多成果:John F. Horty 提出了一种 Agent 在动态环境下理性选择规划的理论。Agent 通过把即将执行的规划与本来执行的规划合并,得出即将执行的规划在合并条件下的范围费用,以此同执行此规划的效用进行比较,最后判断是否执行此规划[5]。Jeffrey S. Cox 提出了一种多 Agent 规划合并算法,通过挖掘耦合信息,使得算法的复杂度降低到了多项式级[4]。Mathijs de Weerdt 提出了一种多 Agent 规划合并的资源逻辑,Agent 之间通过交换资源,删除规划中的动作,以提高执行效率[6]。刘志忠对多 Agent 规划进行了深入研究。他提出了一种体现时间约束和资源可用性约束的多主体计划融合算法并将其运用于空军战役规划[7]。

本文提出一种基于资源共享的规划合并机制,通过利用其它规划产生的资源来替代本规划中部分动作产生的效果,从而达到减少执行成本的目的。

2 相关概念预定义

规划:规划 P 由一个元组构成$\langle S,O,L,R,G\rangle$,其中 S 为 P 中所有动作 s_i 的集合,其中每一个 s_i 对应一个时间点 t_i;O 是一个时序约束集,形式为 $t_i=t_j$,$t_i<t_j$,$t_i=m_k$ 或 $t_i<m_k$;L 为一个因果约束集,形式为$\langle s_i,A,s_j\rangle$,A 为 s_i 产生并能被 s_j 利用的资源;R 为规划最小输入资源;G 为规划产生的资源。时序约束的一致性:如果有$\langle s_i,A,s_j\rangle$,则有 $t_i<t_j$。

资源:资源是 Agent 操作的对象,也是 Agent 产生的对象。资源可被分为两类:消耗性资源和非消耗性资源。消耗性资源是通过 Agent 的操作会造成其量的减少的资源,这种资源可用量的多少来描述,一般有形的资源都属于此类,比如燃料、弹药等;而非消耗性资源是不能用数量来描述,或 Agent 的操作不会导致其量的减少的资源,一般是无形的资源属于此类。比如空气、地点、时间等。

可共享资源:规划 P 的各个动作的输出资源中除去必要部分中的消耗性资源的剩余部分的总和称为规划 P 的可共享资源。实质上,规划 P 的可共享资源是各动作产生的必要部分中的非消耗性资源与副作用的总和,也就是规划 P 的各个动作产生资源中除去自身所需资源的剩余部分。

动作:动作是 Agent 最基本的运行过程,是规划的组成元素。动作 s 实际上是输入资源向输出资源的转化:$In(s)->Out(s)$,其中输入资源包括 s 的前提条件,输出资源包括对环境以及对 Agent 自身状态的改变。规划中每个动作的输出资源可分为 3 部分:直接用于实现目标;用于为后续动作提供前

提条件;对自身规划的执行没有任何贡献的副作用。前两部分定义为必要部分。必要部分不能为空,否则该动作在规划中就没有任何意义。

可调度规划:对于一个规划 $P=\langle S,O,L,R,G\rangle$ 来说,一个调度就是一个时序约束集 O',对于 S 中每一个动作 s_i,都存在一个形式为 $t_i=m_k$ 的约束,并且 $O\cup O'$ 满足时序约束的一致性。若对于一个规划来说,存在一个调度,则称该规划可调度。

若规划的时序约束不一致,则肯定不能被调度,只有时序约束一致的规划才可能被调度。

完全规划:设在规划 P 的因果约束集 L 中,若存在一个因果关系 $\langle s_i,A,s_j\rangle$,则称动作 s_j 前件是确定的。若还存在一个动作 s_k 产生 $\neg A$,并且有一个调度 O' 使得 $O\cup O'\mid -t_i<t_k<t_j$,则称此因果联系被威胁。若某个规划中所有的动作的前件都是确定的并且没有因果联系被威胁,则称这是个完全规划。

规划联合:设规划 $P=\langle S,O,L,R,G\rangle$,规划 $P=\langle S',O',L',R',G'\rangle$,则 $P\cup P'=\langle S\cup S',O\cup O',L\cup L',R\cup R',G\cup G'\rangle$ 为规划 P 与 P' 的联合。

两个规划的联合可能会产生很多问题。两个完全规划的联合可能是不完全的。其中一个规划中的动作可能会威胁到另一个规划的因果关系。两个可调度规划的联合也可能是不可调度的。它们的合并的时序约束集可能不满足一致性。

兼容:设规划 P 和规划 P' 为完全规划,若 $P\cup P'$ 为完全且可调度,则称 P 和 P' 兼容;若通过添加一个时序约束 O' 使得 $\langle S\cup S'',O\cup O'\cup O',L\cup L',R\cup R',G\cup G'\rangle$ 完全且可调度,则称这两规划强兼容。

费用:设规划 $P=\langle S,O,L,R,G\rangle$,规划 P 的费用是所消耗资源的度量,是规划中所包含动作的费用之和,由费用求解函数得到,记为:$\kappa(P)=\sum_{s_i\in S}Cost(s_i)$。

设 $\beta(p)$ 为执行规划 p 所获得的收益,$\kappa(p\mid p_m)$ 为 p 在合并规划 p_m 中时的费用。定义 $\kappa(p\mid p_m)=\kappa(p_m)-\kappa(p_m/p)$,该式的涵义是 p 在合并规划 p_m 中时,其费用为合并规划 p_m 的费用与 p_m 中除去 p 的剩余部分的费用之差。在规划合并的条件下,规划的费用要受到其背景规划的影响。若 p_m 包含 p,则 $\kappa(p\mid p_m)=0$;若 p_m 为空,则 $\kappa(p\mid p_m)=\kappa(p)$。

在规划单独执行时,执行规划 p 的必要条件是 $\beta(p)>\kappa(p)$,而在规划合并的情况下,执行规划 p 的必要条件变为 $\beta(p)>\kappa(p\mid p_m)$。若不满足这个条件,则不能执行 p。

3 规划合并机制

本规划的思想是利用其它规划产生的无用资源来实现自己的目标或为某些动作提供输入资源,从而删除本规划的冗余动作。因此,能否合并规划的关键是看是否存在可共享资源。如果其它规划中的可共享资源中包含有本规划中某个动作产生结果的必要部分,则可将此动作删除。

规划合并后可能会产生很多问题,假设规划 P 中动作 s_1 输出资源的副作用包含了规划 P' 的动作 s_2 输出的必要部分,这时 s_2 就可以删除。而在规划 P' 中 s_2 后的动作 s_3 需要 s_2 为其提供前提条件,这时就要求 s_1 应发生在 s_3 之前。但是 s_1 发生的时间点在 s_3 之后,这就产生了时序约束的不一致。规划合并还可能导致死锁,假设规划 P 中动作 s_1 输出资源的副作用包含了规划 P' 的动作 s_2 输出的必要部分而删除 s_2,P' 中动作 s_4 输出资源的副作用包含了规划 P 的动作 s_3 输出的必要部分而删除 s_3,而 s_3 发生在 s_1 之前,s_2 发生在 s_4 之前,这样就形成了相互等待的情形。所以要对规划合并可能产生的问题进行处理,保证合并后的规划完全且可调度,不能产生时序约束和因果约束的矛盾和死锁。

规划合并算法:

1 $P_m=null$
2 **loop**
3 **while** some P left **do**
4 **if** Scompatible $(P_m,P)=true$ **then**
5 $R_{P_m}=\sum_{s_i\in P_m}R_{s_i}$, n=Count (P)
6 **loop**
7 **while** some $s_n\in P$ left **do**
8 **if** $(R_{s_n}\subseteq R_{P_m})\cap(Consistency=true)$ **then**
9 delete s_n from P
10 $R_{P_m}=R_{P_m}\cup in(s_n)-(R_{s_n}\cap R_{cost})$
11 **end if**
12 $n--$
13 **end loop**
14 **if** $\kappa(P_m\cup P)-\kappa(P_m)>\beta(P)$ **then**
15 $P_m=P_m\cup P$
16 **end if**
17 **end if**
18 **end loop**
19 **return** P_m

图 1 规划合并算法

规划合并算法是将各 Agent 将要执行的规划依次进行合并,即先合并前两个规划,然后将第三个规划与前两个合并的结果进行合并,依此类推。在实际情况中,并不是只有一个动作产生的可共享资源代替另一个动作输出资源的必要部分这种简单的情况,资源的供给关系不仅有一对一,还有一对多、多对一、多对多的情况。本算法同样也适合这些复杂的情况。规划合并算法如图 1 所示。第一步定义合并规划,初始为一个空规划。第二步和第十八步之间为一个循环体,对于每个 Agent 将要执行的规划进行合并。第三步为一个判断,还未处理的规划才能进入下一步操作。第四步到第十七步为一个条件语句,对与合并规划强兼容的规划进行下一步操作,否则不进行合并。由于规划合并会产生很多潜在的冲突,而又要保证合并后的规划可行,即完全且可调度,因此要进行此操作。在此 Scompatible (P_m,P) 为判断 P_m 和 P 是否强兼容的 *bool* 函数。用这个函数去筛选规划实际上是一种非常简化的方法。对于不与合并规划强兼容的规划,Agent 可以对其进行修改,进行动作的增减以及因果联系的增减以满足与合并规划的兼容。但是这些操作都会消耗大量的运算资源。出于对可行性与运算资源两方面考虑,最终选择了这种处理方式。第五步为计算合并规划的可共享资源以及规划 P 中动作的数目。第六步到第十三步之间为一个循环体,对每个属于 P 的动作进行合并操作。第七步为一个判断,使 P 中的动作从最后一个开始进入下一步操作。第八步到第十一步为条件语句,判断 P 中某个动作能否被删除。P 中动作 α_n 能够被删除首先应该满足 P_m 的可共享资源包含了 α_n 输出资源的必要部分。另外,资源共享必然会产生新的因果联系和时序约束,因此要保证合并后时序约束的一致性,还要保证不能产生死锁。若满足这些条件,则进入第九步,将 α_n 从 P 中删除。第十步为更新 P_m 的可共享资源。从 P_m 的可共享资源中减掉 α_n 输出资源的必要部分中的消耗性资源,并把 α_n 的输入资源并入

R_{P_m}。在对规划 P 中每一个动作的处理中，首先是以倒序进行的，而且在删除某动作后，将这个动作的输入资源并入了 R_{P_m}，这样是因为在一个规划中，动作的输出资源有可能是用于为后续动作提供输入资源的，若某动作被删除，那么为它提供输入资源的动作也就没有存在的必要。以倒序进行处理并且在删除某动作后将这个动作的输入资源并入 R_{P_m}，使得为此被删除的动作提供输入资源的动作也有可能被删除。这样处理使得在保证实现目标的前提下，规划 P 中的动作最大程度地被删减，从而最大程度地减少执行规划的花销。极端的情况是前面动作的输出都是为最后一个动作提供输入时，若最后一个动作可以被删除，那么整个规划就被删除。这个规划所要实现的目标由其它规划所提供的可共享资源来实现。第十二步为 n 自减一，使得以倒序对下一动作进行处理。第十四步到第十六步为条件语句，当规划 P 满足在合并条件下执行规划的必要条件时，对 P_m 和 P 进行合并。若不满足，则跳过。第十九步为当所有规划都被处理完毕后，返回 P_m。算法流程如图 2 所示。

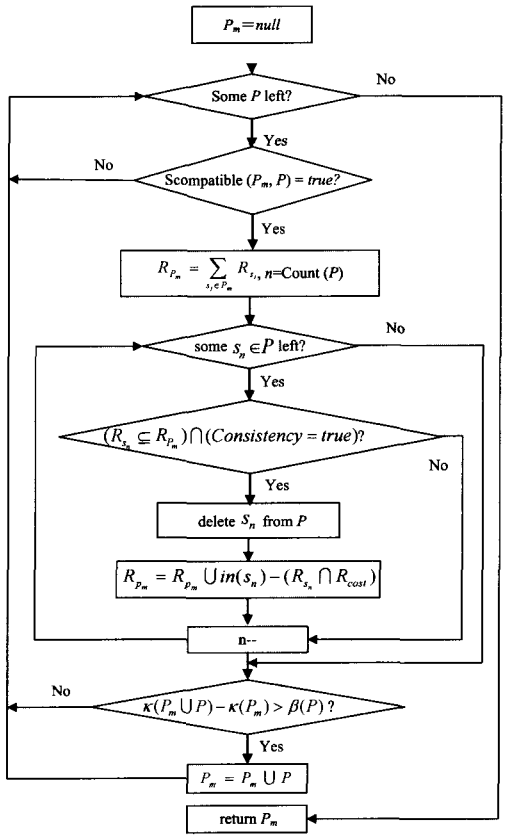

图 2　算法流程图

4　作战方案的优化

每个作战方案都可看成是指挥控制智能体(C2-Agent)将要执行的规划。运用上述的规划合并算法，各作战方案通过共享战场资源，能够删减冗余战术动作，减少执行费用，使得作战方案更为精练。

结束语　本文提出了一种基于资源共享的规划合并机制并给出了相关算法，运用这种算法可以在保证作战方案实现战术目标的前提下更大程度地删减冗余战术动作，提高作战方案的执行效率。这种算法主要解决的是如何提高 Agent 执行规划的效率，对于合并规划后产生的冲突的处理不够细致，这也是今后研究的方向。

参 考 文 献

[1] 王进，徐洮. 分布式作战计划辅助生成系统研究. 计算机工程与设计，2001，22(2)：13-14

[2] 姚莉，张维明. 智能协作信息技术. 北京：电子工业出版社，2002

[3] Wooldridge M. 多 Agent 系统引论[M]. 第一版. 石纯一，等译. 北京：电子工业出版社，2003

[4] Cox J S，Durfee E H. Efficient Mechanisms for Multiagent Plan Merging. New York，USA：AAMAS'04，2004

[5] Horty J F，Pollack M E. Evaluating new options in the context of existing plans. Artificial Intelligence，2001，127：199-220

[6] de Weerdt M，Bos A，Tonino H，et al. A resource logic for multi-agent plan merging. Annals of Mathematics and Artificial Intelligence，2003，37：93-130

[7] 刘志忠. 多主体规划技术及其在空军战役规划中的应用. 学位论文. 长沙：国防科学技术大学，2002

(上接第 264 页)

制 Agent 的运行。信息窗口显示了 Agent 运行时候的各种输出信息，信息窗口包括 5 种，分别是运行环境信息、Agent 信息、Agent 输出信息、Agent 信念值和 Agent 消息。

结束语　本文介绍的 Agent 平台具有以下特点：1)可以比较直观地将业务过程映射到 Agent 模型，其中，业务对象信息直接作为 Agent 的信念；业务规则以约束的形式作为 Agent 信念；其负责的流程片段则以 ECA 规则的形式保存在规划中；2)采用 ECA 规则表达的规划可以定义复杂的 Agent 行为；3)通过扩展动作可以支持多种组件，具有很好的集成能力和扩展能力；4)采用消息中间件作为底层通信架构，能够保证可靠的通信；5)BPM-A 属于构建式，不需要进行编程，就可以创建新的 Agent 系统，并且同时采用图示化界面，所以使用方便直观。下一步我们的工作重点将放在如何增强 Agent 的学习能力和自组织能力上：通过 Agent 的学习能力能够不断扩展并精华其规划库，研究如何通过一组预定义的规划能够实现自组织的协作，以进一步提升业务过程执行的智能性。

参 考 文 献

[1] van der Aalst W M P，et al. Business Process Management：A Survey// BPM 2003，LNCS 2678. Berlin，Heidelberg：Springer-Verlag，2003：1-2

[2] Plesums C. An Introduction to Workflow，extracted from the Workflow Handbook 2002. http://www.wfmc.org/information/introduction_to_workflow02.pdf

[3] Jennings N R，Faratin P，Normam T J，et al. Implementing a Business Process Management System Using ADEPT：A Real-World Case Study. Intl. Journal of Applied AI，2000，14(5)：421-465

[4] AgentBuilder，AgentBuilder Reference Manual. http://www.agentbuilder.com/Documentation/ReferenceManual-v1.4.pdf

[5] Tilab，Introduction to JADE. http://jade.tilab.com/doc/html/intro.htm

[6] 曹健，张申生. 基于 ECA 规则的适应性工作流技术研究[J]. 计算机集成制造系统，2002，8(9)：737-741