班级 1704031

学号 17040310084

本科毕业设计（论文）

外文资料翻译

毕业设计题目 《基于蚁群算法的 Petri 网路径规划》

外文资料题目 《Scheduling Flexible Manufactory Systems Using Petri Nets and Heuristic Search》

学　　院 机电工程学院

专　　业 自动化

学 生 姓 名 李旭东

指导教师姓名 陈玉峰

《基于 Petri 网和启发式搜索的柔性制造系统调度》

摘要

结合启发式搜索的 Petri 网建模为柔性制造系统提供了一种新的调度方法，该方法用 Petri 网模型来描述调度问题，然后根据 Petri 网模型的变迁的触发序列生成并搜索部分可达性图，以找到最优或接近最优的可行调度方案。该方法可以处理诸如具有路径灵活、共享资源、批量和并发性等特征，通过遵循生成的调度可以避免 Petri 网模型和系统中的潜在死锁，因此可以消除保证模型和系统活性的分析成本，探索出一些有效搜索的启发式函数并给出实验结果。

1 介绍

柔性制造系统可以利用机器人、多用途机器等各种资源生产多种类型的产品，虽然柔性制造系统增加的灵活性提供了更多的资源和路线选择，并允许更高的生产率，但同时也带来了一个具有挑战性的问题，即给定的资源分配给生产每种产品所需的不同过程，以及调度活动顺序以实现最佳效率，此文章提出了一种新的调度柔性制造系统的方法，该方法采用了 Petri 网模型和启发式搜索。

即使对于简单的公式[4]，[5]，[9]，[19]，[32]，调度问题也是复杂的，并且在许多情况下[11],[24]是 NP 难题，各种调度方法已得以发展来解决制造系统日益增加的复杂性和灵活性[12]、[33]、[40]、[41]、[42]、[49]

传统上，完成一系列所需流程的零件工艺路线被视为计划，根据确定的工艺路线资源分配被视为调度，计划和调度通常是分开进行的，很少交互，但在柔性制造系统中，计划和调度应该共同进行这样就可以充分利用系统的灵活性。由于柔性制造系统中的一台机器可以用于处理多项任务，并且可以选择要使用的资源，因此在计划零件的工艺路线时必须考虑资源的分配或调度。

由于生成式调度方法的复杂性，诸如模拟与启发式调度规则相结合的评估方法近年来受到重视，然而有效处理具有灵活性和自动化式的制造环境的综合仿真模型既昂贵又难以开发，仿真模型通常用于特定的应用，因此很难将模型和仿真结果一般化处理，此外目前的启发式调度规

往往依赖于经验，而不能够系统地处理柔性制造系统的约束条件，这再次阻碍了模拟方法的推广。

一个完整和通用的调度方法必须能够明确和简明地表述调度问题，并提供一种有效和通用的技术来解决所表述的问题，目前文献中在公式和/或求解技术上有调度方法的困难，诸如整合程序[14]、[15]、[16]、[31]、[37]，搜索[10]、[20]、[29]、[30]、[48]、[52]和网络模型(CPM/PERT、排队网络、图表)[1]、[6]、[8]、[21]、[22]、[43]等方法可以提供有效的解决方案技术，但在共享资源、并发性、路径灵活性、批量等方面存在规划困难问题，诸如代数模型[2]，[42]和控制理论[12]，[18]，[41]的方法在提供有效的解决技术方面有困难。

Petri 网非常适合于对柔性制造系统的动力学建模，Petri 网可以用一个连贯的公式简明扼要地表示系统的活动、资源和约束，由于 Petri 网是一种图形工具，设计者因此也可以更好地理解和规划调度问题。拉森和奥东尼[23]讨论了图形模型的优点，综上 Petri 网现已广泛用于给定调度的性能分析[3]、[7]、[13]、[17]、[39]、[46]、[47]和[50]。

尽管 Petri 网作为规划和解决柔性制造系统调度问题的有效工具，这给人以希望，但是调度的实际生成在 Petri 网领域还没有得到足够的重视。最近已经有一些人开始使用 Petri 网来生成有关柔性制造系统的调度，Shih 和 Sekiguchi[45]提出了一个 FMS 调度系统，该系统模拟了由 Petri 网建模的 FMS 的演化，每当有冲突时调度系统就调用波束搜索路径，随后在波束深度构建部分时间表，并对其进行评估以选择最佳路径，重复循环直到完成一个完整的时间表停止循环。这种基于部分计划的方法不能保证全局优化，奥娜佳等人[36]提出了一种线性规划方法，用于由 Petri 建模系统的周期性调度，沈等人[44]提出了一种方案，该方案从任意调度表开始并应用分支和边界搜索来找到最佳调度，张[51]提出了一种方法，该方法将基于规则的规划系统的规则转换为延时谓词/变迁网模型，并应用 A *算法来找到一个最佳规划。

本文提出的调度方法使用 Petri 网模型来规划一个调度问题，并采用全局搜索并通过使用启发函数来限制搜索空间[25]、[26]，该方法根据 Petri 网模型变迁的触发序列产生最优或接近最优的可行调度表，与

时间驱动相反，该方法是事件驱动的，即根据活动发生的顺序才提供出调度表，当前大多数的调度方法都可以被认为是时间驱动的，即调度表是某些活动将要发生的时刻的列表[35]。 这种方法可能并非总是适用于自然、离散事件驱动的柔性制造系统的调度[40]，事件驱动的调度聚力于活动的优先约束并且对干扰具有鲁棒性。

在制造系统中有许多优化目标，例如使最大完工时间和/或拖延时间最小化是经常需要达到的目标之一，关键机器的最大利用率也经常被考虑在内，生成具有最小或接近最小的最大完工时间的调度是本文的重点。

在时延 Petri 网中，时间可以与地点(时延库所 Petri 网)或转移(时延变迁Petri网)相关联，通常一个时延变迁会从其输入库所移除托肯，并需要花费一些时间才能将托肯移入输出库所。因此在触发开始和结束之间系统的标识(系统的状态)是不确定的，根据是否使用时延变迁或时延库所，活动分别与变迁或库所相关联。处在时延变迁网的情况下必须允许多次变迁初始化或触发来表示活动的并发性。因此必须跟踪每个与已启动变迁相联系的时间，以便正确更新标识或状态，由于在 Petri 网建模的应用中可能无法跟踪已启动的变迁，因此这就需要一种额外的跟踪方法。

但是当使用时延库所时多个被标记过的库所自然表示与该库所相关联活动的并发性，我们只需要跟踪被标记库所的时间即可，然后在任何给定的时刻，系统标识都没有歧义。在本文中，时间仅与库所相关联，并且所有变迁都是瞬发的，变迁表示所涉及活动的开始或终止，因此库所代表活动或资源[18]，[45]，库所中的每个托肯都可用来表示资源的可用性、一个已为下个过程做好准备的零件或一个正在加工的零件。

二 调度的 Petri 网模型

对于本文的讨论，进入柔性制造系统（FMS）并在 FMS 内部移动的每个零件或部分已完工的成品称为部件，离开 FMS 的已完工的零件称为产品，每个产品都是根据其技术要求进行一系列处理的结果。流程中不用考虑资源需求，一连串的流程定义了一个产品类型或作业类型。FMS 可以生产出多个相同产品类的产品，即很多相同的作业类。一个过程可以以多种方式进行，例如考虑以下情况：两个机器 M1 和 M2、一个机器人 R 以及一个零件，可以用机器人 R 在机器 M1 或 M2 中对该零件进行处理，最终会获得相同的产品，在 M1 或 M2 处处理零件称之

为一项操作，因此在操作中会考虑资源需求。一个操作$O_{i,j,k}$表示在第 k 台机器上执行的第 i 个作业类的第 j 个流程，我们假定所有操作都是非抢占式的。因此尽管一般的流程序列保持不变，但可以通过执行不同的操作序列来完成该作业，这些不同的操作顺序列对于 FMS 的灵活性至关重要，即具有生产多种类型的产品和共享资源的能力，以下示例中对可选操作和必要资源的描述给出了作业要求。

例 1： 一个 FMS 具有三个机器 M1，M2，M3 和一个机器人 R，将执行四种作业类型 J1，J2，J3，J4，表 I 列出了执行每个作业的要求。

	Job			
Process	J_1	J_2	J_3	J_4
1	M_1R/M_2R	M_1	M_3	M_2R
2	M_3	M_2R/M_3	M_1/M_2	M_3
3	M_1R/M_2	M_3	N/A	M_1R/M_2R

表 1 示例 I 的作业执行要求

可以在带有机器人 R 的机器 M1 或带有机器人 R 的机器 M2 中执行 J1 的第一个流程，仅可以在机器 M3 上执行 J1 的第二个流程而此流程无需使用 R。在这个表 1 中也给出了在所有流程中的技术优先约束，例如仅在 J1 的第一个过程完成后才能执行 J1 的第二个过程，我们假定在不同的工作类型之间没有技术优先约束，需要注意的是 J3 仅需要完成两个过程。图 1 描述的是具有一个初始标识的作业 J1 的 Petri 网模型。

变迁是瞬发的，$t^b_{i,j,k}$和$t^e_{i,j,k}$分别代表操作$O_{i,j,k}$的开始和结束，$p^i_{j,k}$是作业类型J_j的第 k 个初始库所，代表作业类型J_j的开始也就是说J_j的一个零件已准备就绪，初始库所中的托肯数量代表相应作业批量大小，$p^f_{j,k}$是J_j的第 k 个结束库所它代表作业J_j的结束，即产品J_j已完工。$p_{i,j,k}$是一个操作库所代表$O_{i,j,k}$，操作库所中的托肯表示正在执行的特定操作。

$p^n_{i,j,k}$是第 j 个流程之后作业J_j的第 k 个中间库所，中间库所的托肯表示零件已准备好进行下一个流程，也可以说中间库所表示缓冲区，p^r_i和p^m_i分别是第 i 个机器人和第 i 个机器的资源库所，资源库所中的托肯表示相应资源的可用性，p^r_1被绘制了两次就是为了清晰显示此库所但是它实际上是一个单独的库所。

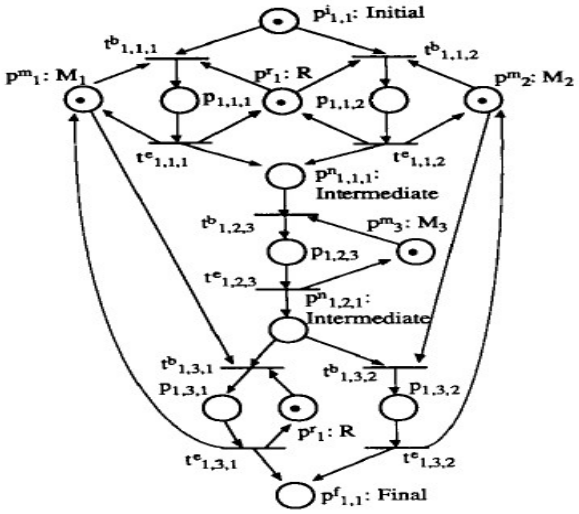


图 1 作业 J1 的 Petri 网模型

正如图中所指示的那样，一个作业描述可以具有多个初始库所和/或结束库所，例如假设由$p_{1,2,3,}$表示的操作需要额外的部分已完工零件，因为该操作是集合类的，然后可以修改作业J_1相应的 Petri 网模型如图 2 所示，$p^i_{1,2}$是一个附加的初始库所。

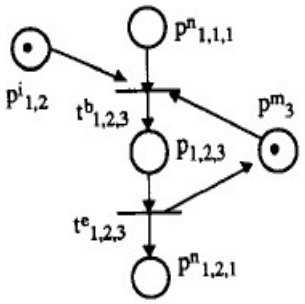


图 2 具有一个额外初始库所的模型修改后的部分

此外，如果新的作业类型J_5从开始操作直到由$p_{1,2,3}$表示的操作与J_1共享相同的通用流程序列，并且不需要进一步流程操作，则可以如图 3 所示修改 Petri 网模型的相应部分。$p_{5,1}^{f}$是J_5的附加的结束库所，这种情况可以在化学过程中得以发觉，其中一个流程会产生副产品。

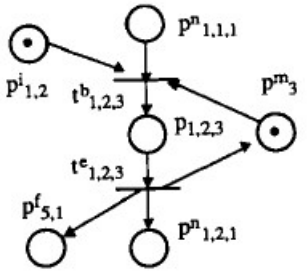


图 3 具有一个额外结束库所的模型修改后的部分

时延与库所有关，中间库所或资源库所的托肯很容易获得，在对应于操作持续时间的延时之后，操作库所的托肯将变得可用。操作的持续时间如表 II 所示。

Operation	Operation Time
$O_{1,1,1}$	10
$O_{1,1,2}$	12
$O_{1,2,3}$	7
$O_{1,3,1}$	5
$O_{1,3,2}$	8

表 II 示例 1 中作业J_1的操作时间

通过使用中间库所可以隐含地避免潜在死锁，当操作$O_{i,j,k}$在机器M_k和$O_{i,j+1,\gamma}$被完成时，潜在的死锁发生，这个过程需要机器M_7，而操作$O_{\alpha,\beta,\gamma}$是在机器M_γ上完成的并且操作$O_{\alpha,\beta+1,k}$需要机器M_k，如图 4 所示。

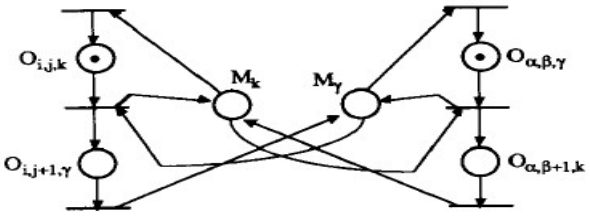


图 4 一个潜在的死锁

通过简单地交换资源，即M_k和M_γ互换可以启动操作$O_{i,j+1,\gamma}$和$O_{\alpha,\beta+1,k}$，但是图 4 的模型不能实现这一点，放置中间库所可以解决这个问题，如图 5 所示。图 5 的模型对应于在机器上有一个缓冲器，在操作过后通过(由机器人或操作员)将零件从机器卸载到缓冲器，此时机器将变得可用且不会出现死锁。在每台机器中，操作之前(之后)将零件装载(卸载)到机器上可被视为操作的一部分。

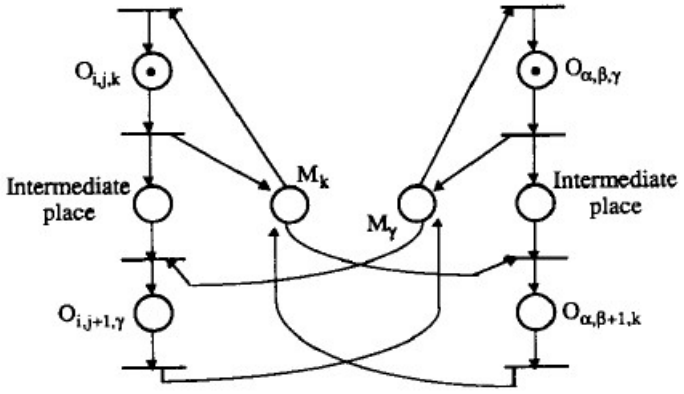


图 5 对潜在死锁的解决方法

这消除了特定类型的潜在死锁，在第三章节中介绍的调度算法在搜索过程中可能会遇到死锁，该算法不断寻找其他方法来绕过死锁并选择那些路径，如果完全没有办法绕过死锁，那么从最初的标识就无法到达期望的最终标识。在这种情况下算法会停下来并报告出错。

在图 5 中，中间库所即缓冲区具有无限大小，当缓冲区大小有限时可以修改模型，如图 6 所示，缓冲区的大小由托肯的数量来表示，例如图 6 所示模型的缓冲区大小是三。

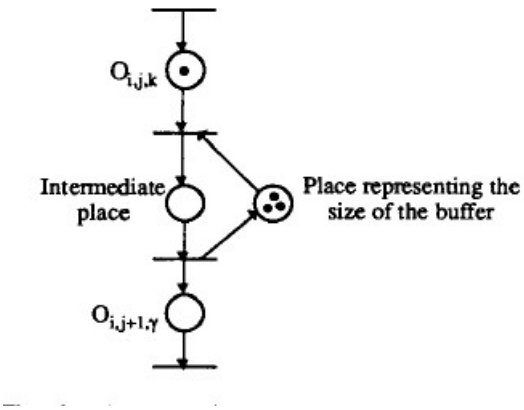


图 6 具有有限大小的缓冲器的模型

可以使用本节中讨论的方法类似地构建示例1的J_2、J_3和J_4的Petri网模型。

根据作业拆分整个模型并通过子网来构建Petri网进行作业并可使其变得简单。由于J_1、J_2、J_3和J_4共享公共资源，即库所，J_1、J_2、J_3和J_4的子网通过库所彼此连接。

使用本节描述的方法对柔性制造系统的作业需求进行建模，给出了一个包含路径灵活、资源共享和大小批量的 Petri 网模型。由于 Petri 网建模还结合了活动间的并发性和优先级约束，因此它有助于维持问题模型的条理性。

三 调度模型

一旦构建了系统的 Petri 网模型，系统的演化就可以用网标识的变化来描述，换句话说系统所有可能的行为都可以被网络的可达性图完全跟踪，因此从理论上讲，可以通过生成可达性图并找到从初始标识到结束标识的最佳路径来获得最佳调度，这个路径也就是 Petri 网模型变迁的触发序列，但是即使对于一个简单的 Petri 网来说，可达性图可能因为太大而不能完整地生成，因此我们使用启发式搜索算法却不用生成整个可达性图。根据所使用的启发式函数，该算法仅生成可达性图的必要部分以找到最优或接近最优的路径，通过这种方式，就可以在一个合理的时间内生成一个合理的良好的调度方案。

找到最佳路径的调度算法 L1 如下：算法 L1 改编自众所周知的图形搜索算法 A* [34]，[38]。给定一个 Petri 网模型，L1 从初始标识开始扩展可达性图，直到可达性图的生成部分接触到最终标识，一旦找到最终标识，通过跟踪表示标识的父结点指针来构造最佳路径，从最终标识回溯到初始标识，然后路径的变迁序列提供了活动的触发顺序即调度表。

算法 L1：

步骤 1：将初始M_0放在列表 OPEN 中。

步骤 2：如果 OPEN 列表为空，则算法终止并失败。

步骤 3：从 OPEN 中删除第一个标记 m 并将 m 放入列表 CLOSED 中。

步骤 4：如果 m 是最终标识，则从初始标识到最终标识构建出最优路径并终止算法。

步骤 5：找到标识 m 的具有发生权的变迁。

步骤 6：为每个具有发生权的变迁生成下一个标识或后继，并从后面的标识到标识 m 设置指针，计算每个后继m'的$g(m')$。

步骤 7：对于 m 的每个后继m'执行以下操作：

a：如果m'已经在 OPEN 表中，则沿着路径引导指针向后走，产生最小的$g(m')$。

b：如果m'已经在 CLOSED 表中，则沿着路径引导指针向后走，产生最小的$g(m')$，如果m'需要指针重定向，将m'移动到 OPEN 表中。

c：计算$h(m')$和$f(m')$并将m'移入 OPEN。

步骤 8：以 f 的递增量对 OPEN 列表重新排序。

步骤 9：转到步骤 2。

算法 L1 使用标识 m 的一个函数，即$f(m) = g(m) + h(m)$，$f(m)$是对路径花费时间的估计值，即从初始标识到沿着通过标识 m 的最优路径的最终标识的最大完工时间，$g(m)$是从初始标识到当前标识 m 的最低路径花费时间，$h(m)$是从标识 m 到沿着通过标识 m 的最优路径的最终标识的路径花费时间的估计。

调度算法是被允许[34]当$h(m)$满足以下条件时始终可以找到最优路径，

$$0 < (m) \leq h(m), \textit{对任意}m\textit{均成立}$$

其中$h(m)$是从 m 到最终标识的最优路径的所花时间，本文采用的启发式算法不能保证受这种下限条件的约束，它们旨在在合理的时间内获得合理的解决调度方案，未来的研究中如果启发式算法得到发展那就可以保证受这种下限的条件的约束。

可达性图的每个弧对应于一个变迁的触发，并且具有相应的时间花销或时间延迟，在可以触发变迁之前，必须使得这个变迁的输入库所的托肯传递至少有一定延时，而这个延时则与输入库所有联系，因此我们可以认为，可达性图的弧形$c(m,m')$的时间花销可被视为是输入库所的最大延时时间，即$c(m,m') = \max(d_{1j}, d_{2j}, \dots, d_{kj})$，在这里$d_{ij} = 1,2,\dots,k$是变迁$t_j$的输入库所$p_{ij}$的时延且标识 m 通过变迁$t_j$转移到了标识 m’。但是如下所述，这种论点并不是正确的，我们让$d_{max} = \max(d_{1j}, d_{2j}, \dots, d_{kj})$，当

标识 m 转换为 m’时，这些不是变迁t_j的输入库所的库所中的托肯也会通过d_{max}同时有一段时延，每个标识的更新都是如此，因此，可达性图的弧的时间花销实际上不是输入库所的最大时间延迟，而是输入库所的残余时延的最大值，即：$c(m,m') = \max(r_{1j}, r_{2j}, \dots, r_{kj})$，其中$r_{ij}, i = 1,2,\dots,$ k 是变迁t_j的输入库所p_{ij}的残余时延，在 L1 的实施中我们解释了这个观测，以便在 L1 产生下面的标识时 L1 的实施可以追踪库所的残余时延。

OPEN 列表保留已生成但尚未探索的标识，这些标识构成了可达性图的边界，可达性图从最初的标识向图边沿外延伸直到到达最后的标识，因此程序不会生成整个可达性图，而是逐层生成可达性图直到算法找到到达最终标识的最优路径。

CLOSED 列表保留了迄今为止已生成和探索的所有标记，即对照最终标识进行检查并找出不是最终标识的标识，在算法 L1 的第 7 步中，如果一个新生成的标识m'既不在 OPEN 中也不在 CLOSED 中，那么它就被置于 OPEN 表中以备后续延伸搜索，如果m'处于 OPEN 表，那么这意味着该算法已经找到了从初始标识到m'的新路径，新路径的时间耗费$g(m')$与旧路径的时间耗费进行比较，然后路径更新以产生最小的耗费时间，因为m'还在 OPEN 上，因此以后还可以被搜索到。

如果m'在 CLOSED 表中，则表示m'已经被探索并且已找到了一条通向m'的新路径。因为m'已经被搜索过了，所以m'的一些后代标识已经生成，如果通向后代标识的路径一直向下，则每条路径都以位于 OPEN 列表中的标识结束。如果通往m'的新路径的耗费时间低于通向旧路径的耗费时间，有两种方法可以处理这种情况，首先，到m'的后代标识的路径可以被重定向，将此变化一直传播到 OPEN 上的标识；其次，就像 L1 算法一样，只有通向m'的路径可以被重定向并且m'可以放入 OPEN 中被重新搜索。

第一种方法的优点是不会丢弃m'的后代标识和已经生成的到这些后代标识的路径，但第一种方法的缺点是需要额外费时来将重定向这个变化传播到m'的所有后代标识，这些标识的数量可能非常大。第二种方法

的优点是节省变化传播的工作，但其缺点是它抛弃了m'的先前生成的后代标识，并重复再次生成m'的一些后代标识的工作，但是m'的一些后代标识可能根本不需要重新生成，在文献[34]，[38]中更加详细地讨论了上述两种方法应用于 A*算法的优缺点。

如果在步骤 5 中没有发现具有发生权的变迁，标识 m 则表示一个死锁。然后在步骤 6 中不会产生标识 m 的下一个标记，紧接着步骤 7 和步骤 8 不会产生任何效果，然后该算法返回到步骤 2 以探索列表 OPEN 上的其他标识，即探索最终可能到达最终标识的其他替代路径，这样的话如果存在这种情况就可以找到避免潜在死锁的路径。如果在步骤 2 中标识已用尽，则给定的初始标识和最终标识之间就没有连接路径了。

由于 L1 算法在任意给定的初始和最终标识对之间找到了一条路径，因此通过适当地设置初始和最终标识可以生成从任意给定状态到任意期望状态的调度表，因此可以在不修改模型的情况下处理部分调度表。在本文中因为重点是最大完工时间因此耗费函数 g(m)是时间的累积，但是对 g(m)来说还可以使用其他函数，例如如果将机器的运行时间用于 g(m)，则可以得到总运行时间最优规划，因此可以通过在调度算法中为 g(m)和 h(m)使用适当的函数或函数组合来调整其他性能标准或多个性能标准。

四 算法 L1 的实现以及调度实例

L1 算法是在一个在 DEC 站点 5000/200 上上用 C 语言实现的，在步骤 8 中重新排序 OPEN 列表实际上是与在 OPEN 上插入标识这个动作相结合的，因此 OPEN 总是按照 f 的递增顺序排列的。通过在步骤 3 中从 OPEN 中选取第一个标识，L1 总是选择具有最佳路径的最小耗费时间估计的标识。

为了在步骤 5 中找到具有发生权的变迁，算法 L1 的实现识别出被标记的库所，对于每个被标记库所，程序可以识别出其输出变迁；对于每个输出变迁，程序检查其是否所有输入库所都被标记，如果是，则在步

骤 6 中使得此变迁有发生权且可以被使用来生成下一个标识，当生成下一个标识时，程序更新标识并计算下一个标识的时间耗费函数 g 和 f，同时该程序还更新了第 3 节中讨论的库所的剩余操作时间。

例 2： 问题是调度一个有三台机器M_1，M_2和M_3的柔性制造系统，该系统将处理两个作业J_1和J_2. 表 3 表述了作业需求，操作时间如表四所示，带有初始标识的系统的 Petri 网模型如图 7 所示。

	J_1	J_1
1	M_1/M_2	M_1/M_3
2	M_2/M_3	$M_1/M_2/M_3$

表 3 例 2 中作业的需求

Op.	Time
$O_{1,1,1}$	3
$O_{1,1,2}$	4
$O_{1,2,2}$	3
$O_{1,2,3}$	2
$O_{2,1,1}$	4
$O_{2,1,3}$	2
$O_{2,2,1}$	3
$O_{2,2,2}$	4
$O_{2,2,3}$	4

表 4 例 2 的操作时间

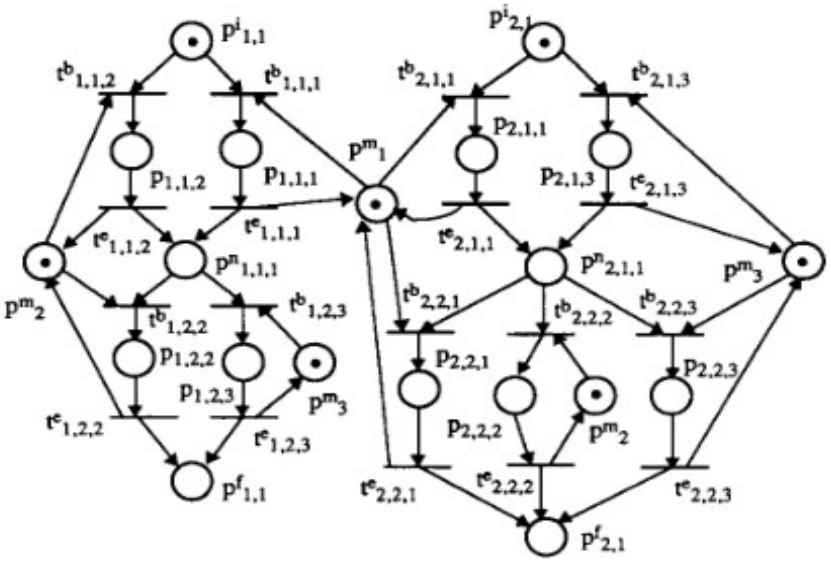


图 7 例 2 系统的 Petri 网模型

例 2 中给出的调度问题可以通过 L1 算法中使用不同的 h 函数来得到解决，用下述 4 个 h 函数实施 L1 算法。

$h_1(m)=0$,对所有的标识m

$h_2(m)=$ -$dep(m)$,在这里$dep(m)$为可达图中标识m的深度

$h_3(m)=\min(rt_{1m},rt_{2m},\cdots,rt_{km})$,这里$rt_{im}$，$i=1,2,\cdots,k$为标识$m$下有一个托肯的库所$p_{im}$的剩余操作时间

$h_4(m)=h_2(m)+h_3(m)=\min(rt_{1m},rt_{2m},\cdots,rt_{km})$-$dep(m)$

$h_1(m)$使$f(m)$等于$g(m)$且程序不使用启发式信息，它选择实际花费时间最小的下一个标识来从初始标识到达该标识，这种非预见的最优优先搜索对应于统一时间花费搜索，并可以保证找到最佳路径[38]。

$h_2(m)$使程序偏向于在可达图中更深层的标识即：

$$f(m)=g(m)+h(m)=g(m)+h_2(m)=g(m)-dep(m) \qquad (5)$$

因此即使标识 m 具有较大时间耗费，如果 m 在可达性图中较深那么该时间耗费也会因此得到补偿，此补偿考虑一个可达图中更深的标识可能更接近最终标识。

$h_3(m)$偏爱一个即将结束操作的标识，完成一项操作可以使得这些资源可用于其他操作。$h_4(m)$结合$h_2(m)$和$h_3(m)$，除了$h_1(m)$以外，启发函数的目的是在较短的时间内找到一个接近最优的调度，表五显示了使用上述四个 h 函数的执行结果。

	Number of iterations	Depth of final marking	Cost of solution path	Transition firing sequence
$h_1(m)$	50	8	6	$t^b2,1,3t^b1,1,1t^c2,1,3$ $t^c1,1,1t^b2,2,1t^b1,2,3$ $t^c1,2,3t^c2,2,1$
$h_2(m)$	13	8	6	$t^b2,1,3t^b1,1,1t^c2,1,3$ $t^b2,2,2t^c1,1,1t^b1,2,3$ $t^31,2,3t^2,2,2$
$h_3(m)$	54	8	6	$t^b2,1,3t^b1,1,2t^c2,1,3$ $t^b2,2,1t^c1,1,2t^b1,2,3$ $t^c2,2,1t^c1,2,3$
$h_4(m)$	53	8	6	$t^b2,1,3t^b1,1,1t^c2,1,3$ $t^c1,1,1t^b2,2,1t^b1,2,3$ $t^c1,2,3t^c2,2,1$

表 5 算法执行结果

迭代的数量是程序重复其主要路径的次数，最终标识的深度表示从初始标识到达最终标识的过渡触发的总数，路径的成本是生成调度的最大完工时间，由于新标识的探索需要一次迭代，因此迭代的数量始终大于或等于最终标识的深度。变迁触发序列给出了调度安排。

从 Petri 网模型中可以直接理解调度表，如表 6 到表 9 所示，操作顺序说明了在给定机器上开始每个操作的顺序。

在表 5 中，所有生成调度都有相同的总耗费，即最大完工时间为 6。在表 6 到表 9 中，每个调度操作时间的序列总和是 10 或 11，这表明每个调度都促进了操作的并发性。

$h_1(m)$	Operation sequence	Operation time
1	$O_{2,1,3}$	2
2	$O_{1,1,1}$	3
3	$O_{2,2,2}$	4
4	$O_{1,2,3}$	2
Sum of operation times		11
Makespan of schedule		6

表 6 在例 1 中使用$h_1(m)$生成的调度

$h_1(m)$	Operation sequence	Operation time
1	$O_{2,1,3}$	2
2	$O_{1,1,1}$	3
3	$O_{2,2,2}$	4
4	$O_{1,2,3}$	2
Sum of operation times		11
Makespan of schedule		6

表 7 在例 1 中使用$h_2(m)$生成的调度

$h_3(m)$	Operation sequence	Operation time
1	$O_{2,1,3}$	2
2	$O_{1,1,2}$	4
3	$O_{2,2,1}$	3
4	$O_{1,2,3}$	2
Sum of operation times		11
Makespan of schedule		6

表 8 在例 1 中使用$h_3(m)$生成的调度

$h_4(m)$	Operation sequence	Operation time
1	$O_{2,1,3}$	2
2	$O_{1,1,1}$	3
3	$O_{2,2,1}$	3
4	$O_{1,2,3}$	2
Sum of operation times		10
Makespan of schedule		6

表 8 在例 1 中使用$h_4(m)$生成的调度

注意到$h_2(m)$极大地减少了迭代次数，即比$h_1(m)$的情况(即非预知的最佳优先搜索)少 74%，这意味着程序使用函数$h_2(m)$能在更短的时间内生成具有相同最大完工时间的调度，这也表明了启发函数$h_2(m)$的值。另

一方面，$h_3(m)$稍微增加了迭代次数然后生成了一个具有相同最大完工时间的调度，这事实上表明在资源共享的交互环境中，只偏向于即将结束的操作几乎没有启发价值。我们也注意到$h_3(m)$比$h_2(m)$更占优势，如$h_4(m)$中案例的结果所示。

例 3:问题是安排一个具有三个多用途机器M_1，M_2和M_3的柔性制造系统，现有五个作业J_1，J_2，J_3，J_4和J_5，每个作业有四个流程且批量均为 10。表 10 列出了作业需求，操作时间见表 11。

	J_1	J_2	J_3	J_4	J_5
1	M_1/M_3	M_1/M_2	$M_1/M_2/M_3$	M_2/M_3	M_1/M_3
2	M_2	M_3	M_2/M_3	M_1/M_3	M_2/M_3
3	M_1/M_3	M_1/M_2	M_1/M_3	M_2/M_3	M_1/M_2
4	M_1/M_2	M_1/M_3	M_1/M_2	$M_1/M_2/M_3$	$M_1/M_2/M_3$

表 10 例 3 的作业需求

Operation	Time	Operation	Time
$O_{1,1,1}$	7	$O_{3,4,1}$	6
$O_{1,1,3}$	4	$O_{3,4,2}$	3
$O_{1,2,2}$	3	$O_{4,1,2}$	9
$O_{1,3,1}$	3	$O_{4,1,3}$	5
$O_{1,3,3}$	6	$O_{4,2,1}$	6
$O_{1,4,1}$	2	$O_{4,2,3}$	2
$O_{1,4,2}$	4	$O_{4,3,2}$	7
$O_{2,1,1}$	8	$O_{4,3,3}$	12
$O_{2,1,2}$	12	$O_{4,4,1}$	9
$O_{2,2,3}$	4	$O_{4,4,2}$	6
$O_{2,3,1}$	7	$O_{4,4,3}$	3
$O_{2,3,2}$	14	$O_{5,1,1}$	10
$O_{2,4,1}$	8	$O_{5,1,3}$	15
$O_{2,4,3}$	4	$O_{5,2,2}$	7
$O_{3,1,1}$	10	$O_{5,2,3}$	14
$O_{3,1,2}$	15	$O_{5,3,1}$	5
$O_{3,1,3}$	8	$O_{5,3,2}$	8
$O_{3,2,2}$	2	$O_{5,4,1}$	4
$O_{3,2,3}$	6	$O_{5,4,2}$	6
$O_{3,3,1}$	2	$O_{5,4,3}$	8
$O_{3,3,3}$	4		

表 11 例 3 的操作时间

系统的 Petri 网模型根据每个作业进行分解，带有初始标识的J_1部分如图 8 所示，J_2，J_3，J_4和J_5的部分可以用第 2 节中描述的方法类似地构建，具有相同位置标注的库所实际上是相同的库所，我们为了清晰呈现它而分

开绘制，总之该系统的 Petri 网模型共有 69 个库所和 82 个变迁，一个库所里面的数字代表托肯的数量。

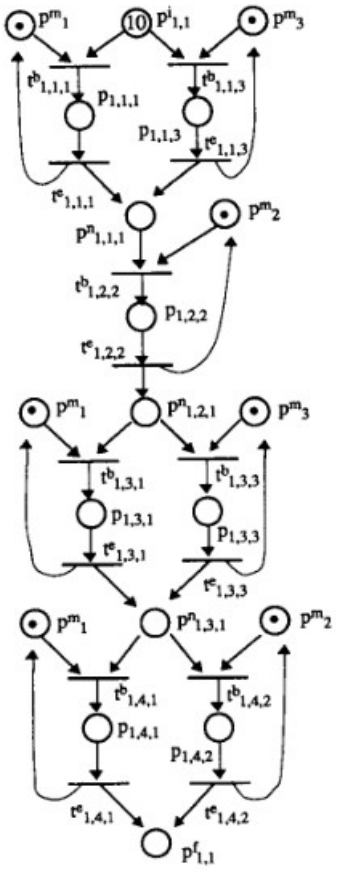


图 8 例 3 中的作业J_1的模型

使用下面的启发函数可以解决例 3 的问题：

$$h(m) = -w \cdot dep(m) \qquad (6)$$

其中w是权重因子，这个启发式函数通过标识的加权深度补偿标记 m 的时间耗费成本，我们假设深度较大的标识更接近最终标识，标识耗费时间的多少与操作次数的多少、批量大小以及作业和操作的数量有关，因此应当有一个适当的权重值反映上述因素。

表 12 展示了w使用不同值的执行结果，当w的值小于 2 时，程序在合理的时间内没有找到一条路径，这是因为所用的计算机大部分时间都在交换内存，我们使用的电脑是 DECstation 5000/200，有 16 兆内存，在后台跑了几天这个程序就停止了执行，当程序找到解决方案时大约需要 5 到 10 分钟的时间。当w的值从 2 变为 5 时，迭代次数显著减少，但解的最大完工时间增加，当w的值从 5 变为 10 或更大时，迭代次数减少一些，解的最大完工时间稍微减少。结果表明，当w的值为 5 或更大时，由于迭代次数

接近其下限 400，并且路径的时间消耗的增加相对较小，因此程序的执行符合我们的期望。

w	Number of iterations	Depth of final marking	Cost of solution path
2	758	400	426
5	412	400	447
10	401	400	439
20	401	400	439
50	401	400	439

表 12 例 3 的执行结果

例 4： 问题是安排一个有多用途机器M_1，M_2，M_3，M_4，M_5和机器人R_1，R_2，R_3的柔性制造系统，现有 10 个作业J_k，$k = 1,2,\cdots 10$，每个作业都有不同数量的流程。表 13、14 和 15 列出了作业需求，操作时间如表 16 所示。Petri 网模型是使用第 2 节中讨论的方法构建的，此外该 Petri 网模型总共有 154 个库所和 182 个变迁。

	J_1	J_2	J_3	J_4
1	M_1R_2/M_3	M_2R_2/M_5R_1	$M_2R_1/M_4R_1/M_5$	M_2R_2/M_3
2	M_2R_1/M_5R_1	$M_3R_2/M-4$	$M_2R_3/M_3R_1/M_4R_1$	M_3R_1/M_5R_2
3	M_2R_3/M_4	M_1R_2/M_5R_3	M_1R_3	M_4R_2
4	NA	$M_1/M_3R_1/M_5$	M_2R_1	NA
5	NA	NA	M_4/M_5R_2	NA

表 13 例 4 中作业J_1到J_4的需求

	J_5	J_6	J_7	J_8
1	M_2R_1/M_3R_2	M_1R_3/M_3R_2	M_2R_2/M_4R_1	$M_2R_2/M_3/M_4$
2	M_5	M_4R_1/M_3	M_3/M_5	M_4R_3/M_5R_1
3	M_1/M_3R_1	M_3R_3/M_5R_2	NA	$M_2/M_3/M_4$
4	M_2R_1/M_4R_1	$M_1/M_2/M_3$	NA	M_5
5	M_4R_1	M_2R_1/M_4	NA	M_2R_3/M_4R_1
6	NA	$M_2R_1/M_3/M_5$	NA	$M_1/M_2/M_3R_3$
7	NA	NA	NA	$M_1/M_4/M_5R_1$
8	NA	NA	NA	M_2R_1/M_3R_2

表 14 例 4 中作业J_5到J_8的需求

	J_9	J_{10}
1	$M_2R_2/M_3/M_5$	M_5
2	M_2	M_1/M_2
3	M_1R_1/M_4	M_5
4	M_4R_1/M_5	M_2R_1/M_3
5	M_1/M_3	NA

表 15 例 4 中作业J_9和J_{10}的需求

Op.	Time	Op.	Time	Op.	Time	Op.	Time
$O_{1,1,1}$	3	$O_{3,5,4}$	7	$O_{8,8,3}$	7	$O_{8,6,3}$	4
$O_{1,1,3}$	5	$O_{3,5,5}$	5	$O_{6,5,2}$	4	$O_{8,7,1}$	9
$O_{1,2,2}$	4	$O_{4,1,2}$	4	$O_{6,5,4}$	5	$O_{8,7,4}$	8
$O_{1,2,5}$	5	$O_{4,1,3}$	6	$O_{6,6,2}$	3	$O_{8,7,5}$	7
$O_{1,3,2}$	2	$O_{4,2,3}$	7	$O_{6,6,3}$	7	$O_{8,8,2}$	5
$O_{1,3,4}$	3	$O_{4,2,5}$	8	$O_{6,6,5}$	8	$O_{8,8,3}$	6
$O_{2,1,2}$	4	$O_{4,3,4}$	5	$O_{7,1,2}$	8	$O_{9,1,2}$	3
$O_{2,1,5}$	4	$O_{5,1,2}$	4	$O_{7,1,4}$	8	$O_{9,1,3}$	4
$O_{2,2,3}$	4	$O_{5,1,3}$	3	$O_{7,2,3}$	9	$O_{9,1,5}$	4
$O_{2,2,4}$	5	$O_{5,2,5}$	6	$O_{7,2,5}$	7	$O_{9,2,2}$	6
$O_{2,3,1}$	8	$O_{5,3,1}$	9	$O_{8,1,2}$	5	$O_{9,3,1}$	5
$O_{2,3,5}$	7	$O_{5,3,3}$	7	$O_{8,1,3}$	7	$O_{9,3,4}$	7
$O_{2,4,1}$	6	$O_{5,4,2}$	10	$O_{8,1,4}$	8	$O_{9,4,4}$	2
$O_{2,4,3}$	3	$O_{5,4,4}$	9	$O_{8,2,4}$	10	$O_{9,4,5}$	4
$O_{2,4,5}$	5	$O_{5,5,4}$	4	$O_{8,2,5}$	10	$O_{9,5,1}$	5
$O_{3,1,2}$	3	$O_{6,1,1}$	6	$O_{8,3,2}$	4	$O_{9,5,3}$	7
$O_{3,1,4}$	4	$O_{6,1,3}$	6	$O_{8,3,3}$	5	$O_{10,1,5}$	4
$O_{3,1,5}$	5	$O_{6,2,3}$	7	$O_{8,3,4}$	6	$O_{10,2,1}$	5
$O_{3,2,2}$	5	$O_{6,2,4}$	5	$O_{8,4,5}$	7	$O_{10,2,2}$	6
$O_{3,2,3}$	5	$O_{6,3,3}$	4	$O_{8,5,2}$	4	$O_{10,3,5}$	7
$O_{3,2,4}$	5	$O_{6,3,5}$	4	$O_{8,5,4}$	3	$O_{10,4,2}$	2
$O_{3,3,1}$	6	$O_{6,4,1}$	7	$O_{8,6,1}$	5	$O_{10,4,3}$	3
$O_{3,4,2}$	3	$O_{6,4,2}$	8	$O_{8,6,2}$	5		

表 16 例 4 的操作时间

式(6)的启发函数可用于解决此问题，表 17 显示了当每个作业的批量固定为 5 时的执行结果，接下来进行不同批量的实验，表 18 显示了每项工作的批量，把执行结果放在了表 19，在表 17 和表 19 中，当权重因子 w 分别小于 2 和 1.5 时程序不能在合理的时间内找到一条路径。

w	Number of iterations	Depth of final marking	Cost of solution path
2	465	450	304
3	465	450	304
4	453	450	304
5	451	450	298
10	451	450	298
100	451	450	298

表 17 例 4 中批量为 5 的执行结果

Job	J_1	J_2	J_3	J_4	J_5	J_6	J_7	J_8	J_9	J_{10}
Lot size	5	6	4	6	4	5	7	3	5	5

表 18 例 4 中对于批量的二次实验

w	Number of iterations	Depth of final marking	Cost of solution path
1.5	1121	420	273
1.51	1012	420	273
1.52	1012	420	273
1.55	885	420	273
1.6	778	420	273
2	499	420	281
5	421	420	293
10	421	420	293

表 19 例 4 中批量为变量时的执行结果

从表 17 和表 19 所示的结果来看，当权重 w 大于或等于 5 时程序理想地执行，在 w 达到 5 之后，有关迭代次数减少的性能提升就已达到一个瓶颈。迭代次数和权重 w 之间有很强的相关性，特别从表 19 中我们可以看到这种关联性。

5 结论

本文提出的新调度方法用时延库所 Petri 网模型规划了一个调度问题，一旦构建了系统的完整模型，调度算法就使用该 Petri 网模型来搜索部分可达图，根据所使用的启发式函数，调度算法可根据 Petri 网模型变迁的触发序列找到全局最优或接近最优的可行调度。

该规划明确且容易处理柔性制造系统的一些重要特征，例如路径灵活性、资源共享、并发性和批量性，并且还要有便于维护问题模型的一致性。通过适当设置初始和最终标识，此方法可以在不修改模型的情况下处理部分调度。通过在调度算法中使用适当的函数或函数的组合，系统的其他性能标准或多个性能标准可以因此得到改进和适配。

我们由于在规划问题期间建立了系统完整的 Petri 网模型，根据变迁的触发序列生成的调度可以通过在模型系统的管控系统中被 Petri 网控制器拿来直接使用，一旦生成了所期望的最终状态的调度就可以通过遵循该调度来避免潜在的死锁，因此这消除了保证模型和系统活性的分析成本。

我们还进行实验对一些启发式函数的效率做了研究。式(6)的函数具有适当调整的权重w，它是一个潜在的好的启发式函数。我们还在做进一步的研究来探索其他可用于减少搜索的启发式函数。本文没有明确讨论机器之间的物料转移，但零件的流动对调度来说也很重要。自动导向车辆系统(AGVS)越来越多地用于物料搬运，人类也正在进行进一步研究以协同处理 AGVS 和零件加工设备例如机器和机器人[27]，[28]。

致谢

作者非常感谢三位匿名审稿人重要和具有建设性的意见。

参考文献

[1]S. Ahmad and B. Li, “Robot control computation in microprocessor systems with multiple arithmetic processors using a modified DF/HIS scheduling algorithm,” *IEEE Transactions on Systems, Man, and Cybernetics,* vol. 19, no. 5, pp. 1167-1178, 1989.
[2]M. Aicardi, A. D. Febbraro, and R. Minciardi, “Analysis of deterministic discrete event systems via minimax algebra,” in *Proceedings of the I991 IEEE International Conference on Systems, Man, and Cybernetics,* Charlottesville, VA, Oct 13-16, 1991, pp. 321-328.
[3]R. Y. AI-Jaar and A. A. Desrochers, “Performance evaluation of automated manufacturing systems using generalized stochastic Petri nets,” *IEEE Transactions on Robotics and Automation,* vol. 6, no. 6. pp. 621-639, 1990.
[4]K. R. Baker, *Introduction to Sequencing and Scheduling.* New York: John Wiley & Sons, 1974.
[5]E. Bowman, “The schedule-sequencing problem,” *Operations Research,* vol. 7, no. pp. 621-624, 1959.
[6]P. J. Brucker, “Scheduling problems in connection with flexible production system s,” in *Proceedings of the I991 IEEE International Conference on Robotics and Automation,* Sacramento, CA, April 1991, pp.1778-1783.
[7]J. Carlier and P. Chretienne, “Timed Petri net schedules,” in *Advances in Petri Nets,* Rozenberg, Ed., Berlin: Springer-Verlag, 1988, pp. 62-84.
[8]C. L. P. Chen, “Time lower bound for manufacturing aggregate scheduling problems,” *Proceedings of the I991 IEEE International Conference on Robotics and Automation,* Sacramento, CA, April 1991, pp. 83C835.
[9]R. Conway, W. Maxwell, and L. Miller, *Theory of Scheduling.* Reading, MA: Addison-Wesley, 1967.
[10]E. Falkenauer and S. Bouffouix, “A genetic algorithm for job shop,” in *Proceedings of the 1991 IEEE International Conference on Robotics and Automation,* Sacramento, CA, April 1991, pp. 824-829.
[11]S. French, *Sequencing and Scheduling: An Introduction to the Mathematics of the Job-Shop.* Ellis Horwood, 1982.
[12]S. B. Gershwin and R. R. Hildebrant, et al., “A control perspective on recent trends in manufacturing systems,” *IEEE Control Systems Magazine,* vol. 6, no. 2, pp. 3-15, 1986.
[13]H. P. Hillion and J. Proth, “Performance evaluation of job-shop systems using timed event-graphs,” *IEEE Transactions on Automatic Control,* vol. 34, no. 1, pp. 3-9, 1989.
[14]D. J. Hoitomt, J. B. Perkins. and P. B. Luh, “Distributed scheduling of job shops,” in *Proceedings of the 1991 IEEE International Conference on Robotics and Automation,* Sacramento, CA, April 1991, pp. 1067-1072.
[15]D. J. Hoitomt, P. B. Luh, and K. R. Pattipati, “A Lagrangian relaxation approach

to job shop scheduling problems," in *Proceedings ofthe* 1990 *IEEE International Conference on Robotics and Automation,* Cincinnati, OH, May 1990, pp. 1944-1949.
[16]D. J. Hoitomt, P. B. Luh, and K. R. Pattipati, "Job shop scheduling," in *Proceedings of the Firsf International Conference on Automation Technology,* Taipei, Taiwan, July 1990, pp. 565-574.
[17]M. A. Holliday and M. K. Vernon, "A generalized timed Petri net model for performance analysis." *Proceedings of the IEEE International Workshop on Timed Petri Nets,* Torino, Italy, July 1-3, 1985, pp.18 1- 190.
[18]L. E. Holloway and B. H. Krogh, "Synthesis of feedback control logic for a class of controlled Petri nets," *IEEE Transactions on Automatic Control,* vol. 35, no. 5, pp. 514-523, May 1990.
[19]E. Horowitz and S. Sahni, *Fundamentals of Computer Algorithms.* Rockville, MD: Computer Science Press, 1978.
[20]E. S. H. Hou and H. Y. Li, "Task scheduling for flexible manufacturing systems based on genetic algorithms," in *Proceedings of the 1991 IEEE International Conference on Systems. Man, and Cybernetics,* 1991, pp.397-402.
[21]C. Q. Jiang, M. G. Singh and K. S. Hindi, "Optimized routing in flexible manufacturing systems with blocking," *IEEE Transactions on Systems, Man, and Cybernetics,* vol. 21, no. 3, pp. 589-595, 1991.
[22]P. Kapasouris and D. Serfaty, *et al.,* "Resource allocation and performance evaluation in large human-machine organizations," *IEEE Transactions on Systems, Man, and Cybernetics,* vol. 21, no. 3, pp.521-532, 1991.
[23]R. Larson and A. Odoni, *Urban Operations Research.* Englewood Cliffs, NJ: Prentice-Hall, 1981, pp. 360-361.
[24]A. E. J. Lee, "Integrated tooling and scheduling of flexible machines: Theory and algorithms," Ph.D. Dissertation, Rensselaer Polytechnic Institute, Troy, NY, May 1989.
[25]D. Y. Lee and F. DiCesare, "FMS scheduling using Petri nets and heuristic search," in *Proceedings of the 1992 IEEE International Conference on Robotics and Automation.* Nice, France, May 1992, pp. 1057-1062.
[26]D. Y. Lee and F. DiCesare, "Experimental study of a heuristic function for FMS scheduling," in *Proceedings of the 1992 Japan-USA Symposium on Flexible Automation,* San Francisco, CA, July 13-15, 1992, pp.1171-1 177.
[27]D. Y. Lee and F. DiCesare, "Integrated models for scheduling flexible manufacturing systems," in *Proceedings of the I993 IEEE Intl. Cotf. On Robot.* K *Auto.,* Atlanta, Georgia, May 2-7, 1993, pp. 827-832.
[28]D. Y. Lee and F. DiCesare, "Scheduling of flexible manufacturing systems employing automated guided vehicles," in *Proceedings of the 9th International Conference on CADICAM, Robotics, and Factories of the Future.* Newark, New Jersey, USA, August 17-20, 1993, in press.
[29]P. S. Liu and L. C. Fu, "Planning and scheduling in a flexible manufacturing system using a dynamic routing method for automated guided vehicles," in *Proceeditigs ofthe 1989 IEEE International Conference on Robotics and Automation.* Scottsdale, AZ, 1989, pp. 1584-1589.
[30]Z. P. Lo and B. Bavarian, "Job scheduling on parallel machines using simulated

annealing," *Proceedings of the I991 IEEE International Conference on Systems, Man, and Cybernetics,* 1991, pp. 391-396.
[31]P. B. Luh and D. J. Hoitomt, *et al..* "Schedule generation and reconfiguration for parallel machines," *IEEE Transactions on Robotics and Automation,* vol. 6, no. 6, pp. 687-696, Dec. 1990.
[32]A. Manne, "On the job-shop scheduling problem," *Operations Research.* vol. 8, no. pp. 219-223, 1960.
[33]P. Mellor, "A review of job shop scheduling," *Operational Research Quarterly.* vol. 17, no. 2, pp. 161-171, June 1966.
[34]N. Nilsson, *Principles of Artificial Intelligence.* Palo Alto, CA: Tioga, 1980.
[35]J. O'Brien, *Schedding Handbook.* New York: McGraw-Hill, 1969.
[36]K. Onaga, M. Silva and T. Watanabe, "On periodic schedules for deterministically timed Petri net systems," in *Proceed/ngs of the 4th International Workshop on Petri Nets and Performance Models,* Melbourne, Australia, Dec. 2-5, 1991, pp. 21G215.
[37]T. A. Owens and P. B. Luh, "A job completion time estimation method for work center scheduling," in *Proceedings of the 1991 IEEE International Conference on Robotics and Automation.* Sacramento, CA, April 1991, pp. 11G115.
[38]J. Pearl. *Heuristics: Intelligent Search Strategies for Conipurer Problem Soli.ing.* Reading, MA: Addison-Wesley. 1984.
[39]C. Ramamoorthy and G. Ho, "Performance evaluation of asynchronous concurrent systems using Petri nets," *IEEE Transactions on Sofmare Engineering,* vol. 6, no. *5,* pp. 440449. 1980.
[40]J. Rickel, "Issues in the design of scheduling systems," in *Expert Systems and Intelligent Manufac.turing,* Oliff, Ed. Elsevier, 1988. pp. 70-89.
[41]F. Rodammer and J. K. White, "A recent survey of production scheduling," *IEEE Transactions on Systems, Man. and Cybernetics.* vol. 18, no. 6, pp. 841-851, 1988.
[42]R. V. Rogers and K. P. White, "Algebraic, mathematical programming, and network models of the deterministic job-shop scheduling problem," *IEEE Transactions on SJ>stemsM, an. and Cybernetics.* vol. 21, no. 3, pp. 693497, 1991.
[43]R. Sengupta and S. Lafortune, "Optimal control of a class of discrete event systems," *Proceedings of the IFAC International Symposium on Distributed Intelligence Systems.* Arlington, VA, Aug 13-15, 1991, pp. 25-30.
[44]L. Shen and Q. Chen, *et al.,* "Truncation of Petri net models of scheduling problems for optimum solutions," *Proceedings of the JapanlUSA Symposium on Flexibie Automation.* San Francisco. CA, July 13-15, 1992, pp. 1681-1688.
[45]H. Shih and T. Sekiguchi, "A timed Petri net and beam search based on-line FMS scheduling system with routing flexibility," *Proceedings* of *the I9YI IEEE International Conference on Robotics and Automation.* Sacramento, CA, April 1991, pp. 2548-2553.
[46]J. Sifakis, "Performance evaluation of systems using nets," *Net Theory and Applications.* Brauer, Ed. Berlin: Springer-Verlag, 1980, pp. 307-319.
[47]M. Silva and R. Valette, "Petri nets and flexible manufacturing," *Advances in Petri Nets.* Berlin: Springer-Verlag, 1989, pp. 374417.
[48]A. S. Spachis and J. R. King, "Job-shop scheduling heuristics with local

neighbourhood search," *International Journal of Production Research,* vol. 17, no. 6, pp. 507-526, 1979.
[49]A. Vasquez and P. B. Mirchandani, "Concurrent resource allocation for production scheduling," *Proceedings of the I991 IEEE International Conference on Robotics and Automation,* Sacramento, CA, April 1991. pp. 1060-1066.
[50]N. Viswanadham and Y. Narahari, "Stochastic Petri net models for performance evaluation of automated manufacturing systems,'' *Information and Decision Technologies,* vol. 14, no. pp. 125-142, 1988.
[51]D. Zhang, "Planning using timed PrR nets," in *Proceedings of the JapanlUSA Symposium on Fle.uible Automation,* San Francisco, CA, July 13-15, 1992, pp. 1179-1183.
[52]Y. Zhang, "Solution to job-shop scheduling of FMS by neural networks," in *Proceedings of the 1991 IFAC Workshop* on *Discrete Event System Theory and Applications in Manufacturing and Social Phenomena.* Shenyang, China, June 25-27, 1991, pp. 261-266.

李斗勇:获得首尔国立大学控制与仪器工程系的学士学位,是纽约特洛伊伦斯勒理工学院电子、计算机和系统工程系的博士后研究助理,他的研究兴趣包括离散事件系统的建模、分析和控制,Petri 网理论及其应用,柔性制造系统和自动导向车辆系统的调度和监督控制以及多媒体理论和应用。他于 1993 年获得伦斯勒理工学院查尔斯·密斯博士奖,是美国电气和电子工程师学会机器人和自动化学会以及美国电气和电子工程师学会系统、人和电子学会的成员。

弗兰克迪斯卡尔:分别于 1960 年和 1962 年获得马萨诸塞州波士顿西北大学电气工程学士和硕士学位,1969 年他加入了纽约州特洛伊市的伦斯勒理工学院,目前是电气、计算机和系统工程教授,目前的研究重点和研究生的教学领域是离散事件系统的建模、分析和控制,应用于制造和智能车辆高速公路系统。在本科教学领域他教授微处理器系统和嵌入式控制课程,从 1956 年到 1960 年,他在麻省理工学院、剑桥和麻省理工学院仪器实验室担任合作学生工程师,从 1960 年到 1962 年,他是东北大学的研究员和教学助理。作为一名美国陆军信号部队的军官,他 1962 年至 1964 年在艾尔亨茨维尔红石兵工厂的法令导弹学校担任教官和课程主任,在此之后,于 1964-1969 年在卡梅吉梅隆大学担任电气工程讲师和运输研究所的研究工程师,为各级政府和私营部门的许多公司和机构提供信息和控制系统方面的咨询服务。他是 1965 年科珀斯奖学金的获得者,1984 年获得了电气和电子工程师协会百年奖和奖章,1987 年被制造工程师协会授予伦斯勒理工学院领导奖,1993 年获得了电气和电子工程师协会系统、人和控

制论协会的富兰克林·泰勒奖。他在电气和电子工程师协会的出版物和会议上都非常活跃，目前担任系统长期规划和财务副总裁，同时也是 1994 年在德克萨斯州圣安东尼奥召开的管理控制会议的联合主席。

Scheduling Flexible Manufacturing Systems Using Petri Nets and Heuristic Search

Doo Yong Lee, *Member, IEEE,* and Frank DiCesare, *Member, IEEE*

***Abstract*—Petri net modeling combined with heuristic search provides a new scheduling method for flexible manufacturing systems. The method formulates a scheduling problem with a Petri net model. Then, it generates and searches a partial reachability graph to find an optimal or near optimal feasible schedule in terms of the firing sequence of the transitions of the Petri net model. The method can handle features such as routing flexibility, shared resources, lot sizes and concurrency. By following the generated schedule, potential deadlocks in the Petri net model and the system can be avoided. Hence the analytical overhead to guarantee the liveness of the model and the system is eliminated. Some heuristic functions for efficient search are explored and the experimental results are presented.**

I. Introduction

FLEXIBLE manufacturing systems (FMS's) can produce multiple types of products using various resources such as robots, multipurpose machines, etc. While the increased flexibility of an FMS provides a greater number of choices of resources and routings, and allows greater productivity, it imposes a challenging problem; i.e., the allocation of given resources to different processes required in making each product and the scheduling of the sequence of activities to accomplish the best efficiency. This paper presents a new method to schedule FMS's, which uses a Petri net formulation and heuristic search.

Scheduling problems are known to be complex even for simple formulations [4], [5], [9], [19], [32] and are NP-hard in many cases [11], [24]. Various scheduling methods have been developed to tackle the ever increasing complexity and flexibility of manufacturing [12], [33], [40], [41], [42], [49].

Traditionally, the routing of a part to complete the sequence of required processes is considered planning and the assignment of resources according to the determined routing is considered scheduling. Planning and scheduling are often separately carried out with little interaction. But in an FMS, the planning and the scheduling should be collectively carried out to take full advantage of the flexibility of the system. Since a machine in the FMS can be used for multiple jobs and there are choices of resources to be used, the assignment of resources, or scheduling, must be considered when the routing of a part is planned.

Manuscript received November 3, 1992; revised July 30, 1993.

D. Y. Lee is with Information Technology Services, Rensselaer Polytechnic Institute, Troy, NY 12180-3590.

F. DiCesare is with the Department of Electrical, Computer, and Systems Engineering, Rensselaer Polytechnic Institute, Troy, NY 12180-3590.

IEEE Log Number 9215779.

Because of the complexity of generative scheduling methods, evaluative methods such as simulation combined with heuristic dispatch rules have recently been given much attention. However, comprehensive simulation models that effectively treat flexible and automated manufacturing environments are expensive and difficult to develop. Simulation models are often specific to particular applications so that the models and the simulation results are difficult to generalize. Moreover, instead of systematic handling of the constraints of FMS's, present heuristic dispatch rules often rely on empirical experience. This again hinders the generalization of the simulation methods.

A complete and general scheduling method must be able to formulate explicitly and concisely a scheduling problem, and provide an efficient and general technique to solve the formulated problem. Current scheduling methods in the literature have difficulties with the formulation and/or the solution techniques. Methods such as integer programming [14], [15], [16], [31], [37], search [10], [20], [29], [30], [48], [52], and network models (CPM/PERT, queuing networks, graphs) [1], [6], [8], [21], [22], [43] can provide efficient solution techniques but have formulation difficulties in accounting for shared resources, concurrency, routing flexibility, lot sizes, etc. Methods such as algebraic models [2], [42], and control theoretic approaches [12], [18], [41] have difficulties in providing efficient solution techniques.

Petri nets are well suited to model the dynamics of flexible manufacturing systems. Petri nets can concisely represent the activities, resources and constraints of the system in a single coherent formulation. And since Petri nets are a graphical tool, designers can also better understand and formulate scheduling problems. Larson and Odoni [23] discuss the merits of graphical models. Petri nets have been widely used for performance analysis of given schedules [3], [7], [13], [17], [39], [46], [47], [50].

Although Petri nets showed promise as an effective tool to formulate and solve the scheduling problems of FMS's, the actual generation of schedules has not been given much attention in the Petri net community. Recently, there have been some independent efforts to use Petri nets to generate schedules for FMS's. Shih and Sekiguchi [45] present an FMS scheduling system which simulates the evolution of the FMS as modeled by Petri nets. The scheduling system calls for a beam search routine whenever there is a conflict. The beam search routine then constructs partial schedules within the beam-depth and evaluates them to choose the best one. The cycle is repeated until a complete schedule

is achieved. This method based on partial schedules does not guarantee global optimization. Onaga *et al.* [36] present a linear programming approach for periodic scheduling of systems modeled by Petri nets. Shen *et al.* [44] present a scheme which starts with an arbitrary schedule and applies branch and bound search to find an optimal schedule. Zhang [51] presents a method which translates rules of a rule based planning system into a timed Predicate/Transition net model and applies the A^* algorithm to find an optimal schedule.

The scheduling method presented in this paper formulates a scheduling problem using a Petri net model, and employs global search and limits the search space by the use of heuristic functions [25], [26]. The methods generates an optimal or near optimal feasible schedule in terms of the firing sequence of the transitions of the Petri net model. This method is also event driven as opposed to time driven, i.e., the schedule is provided as an order of the initiations of the activities. Most of the current scheduling approaches can be considered as time driven, i.e., the schedule is a list of time instants when certain activities are to happen [35]. This approach may not always be best for the scheduling of flexible manufacturing systems that are, by nature, discrete event driven [40]. Event driven scheduling focuses on the precedence constraints of the activities and is robust to disturbances.

There are many targets for optimization in manufacturing. For example, the minimization of makespan and/or tardiness is one of the frequently adapted goals. The maximum utilization of critical machines is also often considered. Generating a schedule with the minimum or near minimum makespan is the focus in this paper.

In timed Petri nets, time can be associated with either places (timed place Petri nets), or transitions (timed transition Petri nets). Generally, a timed transition removes tokens from its input places and takes some time before it introduces tokens to its output places. Therefore, between the initiation and the termination of firing, the marking (the state of the system) is uncertain. Depending on whether timed transitions or timed places are used, activities are associated with transitions or places, respectively. In the case of timed transitions, multiple initiation or firing of transitions must be allowed to represent concurrency of activities. Therefore, the time associated with each initiated transition must be tracked in order to correctly update the marking, or state. Since the initiated transitions may not be tracked in applications of Petri net modeling, an additional tracking method is required.

But when timed places are used, multiple marked places naturally represent the concurrency of activities associated with the places and it is only necessary to keep track of time for the marked places. And at any given instant, there is no ambiguity in the marking. In this paper, time is associated only with places, and all transitions are immediate. Transitions represent the initiation or the termination of the involved activities. Places, therefore, represent activities or resources [18], [45]. Each token in the places represents either the availability of a resource, the readiness of a part for the next process, or a part being processed.

II. Petri Net Modeling for Scheduling

For the discussion of this paper, every part or partially finished product which enters and moves inside an FMS is called a part. A finished part that leaves the FMS is called a product. Each product is the result of a sequence of *processes* according to its technological requirements. Resource requirements are not considered in the processes. A sequence of processes defines a *product type* or a *job type*. The FMS can produce multiple products of the same product type, i.e., a lot of the same job type. A process can be carried out more than one way. For instance, consider the following scenario of two machines M_1 and M_2, a robot R, and a part. The part can be treated at M_1 or at M_2 with R, accomplishing the same result. Treating the part at M_1 or at M_2 is an *operation*. Hence resource requirements are considered in an operation. An operation $O_{i,j,k}$ represents the jth process of the ith job type being performed at the kth machine. All the operations are assumed to be non-preemptive. The job, therefore, can be completed by carrying out different sequences of operations although the generic sequence of processes remains the same. These different sequences of operations are vital to the flexibility of the FMS; i.e., capability to produce multiple types of product and share resources. The requirements of a job can be given by the description of alternative operations and the necessary resources as shown in the following example.

Example 1 An FMS has three machines M_1, M_2, M_3 and a robot R. Four job types J_1, J_2, J_3, J_4 are to be carried out and Table I shows the requirements for each job.

The first process of J_1 can be carried out at either M_1 with R or M_2 with R. The second process of J_1 can be carried out only at M_3 but does not require the use of R. This table also gives the technological precedence constraints among the processes. For instance, the second process of J_1 can be carried out only after the first process of J_1 is complete. No technological precedence constraints are assumed among different job types. Notice that J_3 has only two processes to be completed. Fig. 1 shows the Petri net model of J_1 with an initial marking.

The transitions are immediate. $t^b_{i,j,k}$ and $t^e_{i,j,k}$ represent the beginning and the end of an operation $O_{i,j,k}$, respectively. $p^i_{j,k}$ is the kth *initial place* of J_j representing the beginning of the job type J_j, i.e., a part for J_j is ready. The number of tokens in an initial place represents the lot size of the corresponding job. $p^f_{j,k}$ is the kth *final place* of J_j representing the end of J_j, i.e., a product J_j is completed. $p_{i,j,k}$ is an operation place representing $O_{i,j,k}$. A token in an operation place represents the specific operation being performed.

$p^n_{i,j,k}$ is the kth *intermediate place* of J_i after the jth process. A token in an intermediate place represents the readiness of the part for the next process, i.e., an intermediate place represents a buffer. p^r_i and p^m_i are *resource places* for the ith robot and the ith machine, respectively. A token in a resource place represents the availability of the corresponding resource. p^r_1 is drawn twice just for clear presentation and is actually a single place.

As indicated, a job representation may have more than one initial and/or final place. For example, assume that the

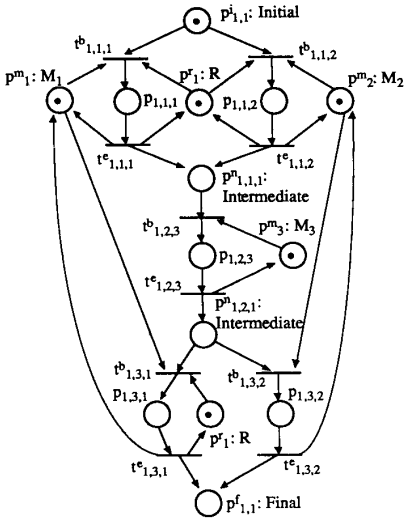


Fig. 1. The Petri net model of J_1.

TABLE I
JOB REQUIREMENTS OF EXAMPLE 1

	Job			
Process	J_1	J_2	J_3	J_4
1	M_1R/M_2R	M_1	M_3	M_2R
2	M_3	M_2R/M_3	M_1/M_2	M_3
3	M_1R/M_2	M_3	N/A	M_1R/M_2R

TABLE II
THE OPERATION TIMES OF J_1 OF EXAMPLE 1

Operation	Operation Time
$O_{1,1,1}$	10
$O_{1,1,2}$	12
$O_{1,2,3}$	7
$O_{1,3,1}$	5
$O_{1,3,2}$	8

operation represented by $p_{1,2,3}$ requires an additional partially finished part since the operation is an assembly. Then the corresponding Petri net model of J_1 can be modified as shown in Fig. 2. $p^i_{1,2}$ is an additional initial place.

In addition, if a new job type J_5 shares the same generic sequence of processes with J_1 up to the operation represented by $p_{1,2,3}$ and does not require further processes, the corresponding part of the Petri net model can be modified as shown in Fig. 3. $p^f_{5,1}$ is an additional final place for J_5. This situation can be found in chemical processes where a process generates a byproduct.

Time delays are associated with places. A token in an intermediate place or a resource place is readily available. A token at an operation place becomes available after a delay corresponding to the duration time of the operation. The duration times of operations are given as shown in Table II.

A potential deadlock is implicitly avoided by using the intermediate places. The potential deadlock happens when

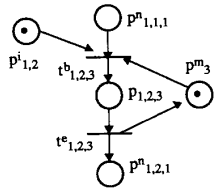


Fig. 2. The modified part of the model with an additional initial place.

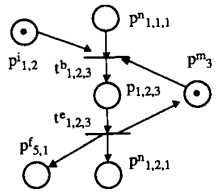


Fig. 3. The modified part of the model with an additional final place.

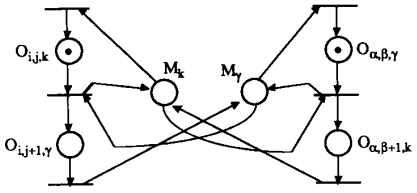


Fig. 4. A potential deadlock.

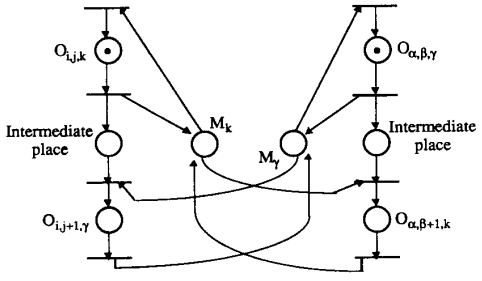


Fig. 5. The solution to the potential deadlock.

the operation $O_{i,j,k}$ is completed at the machine M_k and $O_{i,j+1,\gamma}$ requires the machine M_γ, and the operation $O_{\alpha,\beta,\gamma}$ is completed at the machine M_γ and $O_{\alpha,\beta+1,k}$ requires the machine M_k. This is illustrated in Fig. 4.

By simply swapping the resources, i.e., M_k and M_γ, the operations $O_{i,j+1,\gamma}$ and $O_{\alpha,\beta+1,k}$ can be initiated. But the model of Fig. 4 fails to achieve this. Putting intermediate places can solve this problem as shown in Fig. 5. The model of Fig. 5 corresponds to having a buffer at the machine. By unloading (by either a robot or a human operator) the part from the machine to the buffer after the operation, the machine becomes available and the deadlock does not occur. At each machine, the loading (unloading) of the part onto (from) the machine before (after) the operation is regarded as part of the operation.

This eliminates a particular type of potential deadlocks. The scheduling algorithm presented in Section III may encounter a

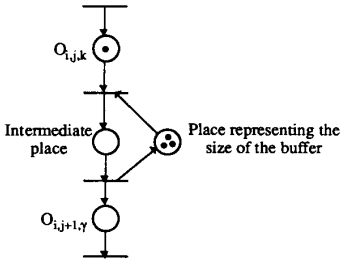


Fig. 6. A model for a buffer with a finite size.

deadlock during the search. The algorithm keeps on looking for alternative ways to get around the deadlock and chooses those paths. If there is absolutely no way to get around the deadlock, then, the desired final marking is unreachable from the initial marking. In this case, the algorithm halts and reports it.

In Fig. 5, the intermediate places, i.e., buffers, have infinite sizes. When the buffer size is finite, the model can be modified as shown in Fig. 6. The buffer size is represented by the number of tokens, e.g., the buffer size of the model shown in Fig. 6 is three.

The Petri net models of J_2, J_3 and J_4 of Example 1 can be similarly constructed using the method discussed in this section. Decomposing the entire model according to jobs and building the Petri net by subnets for jobs make it easy. Since J_1, J_2, J_3 and J_4 share common resources, i.e., places, the subnets for J_1, J_2, J_3 and J_4 are connected with each other through the places.

Modeling the job requirements of an FMS using the method described in this section gives a Petri net model which incorporates routing flexibility, shared resources and lot sizes. Since the Petri net modeling also incorporates concurrency and precedence constraints among activities, it facilitates maintaining the coherency of the problem model.

III. Scheduling Algorithm

Once the Petri net model of the system is constructed, the evolution of the system can be described by changes in the marking of the net. In other words, all possible behaviors of the system can be completely tracked by the reachability graph of the net. Theoretically, therefore, an optimal schedule can be obtained by generating the reachability graph and finding the optimal path from the initial marking to the final marking. The path is a firing sequence of the transitions of the Petri net model. But even for a simple Petri net, the reachability graph may be too large to generate in its entirety. Instead of generating the entire reachability graph, a heuristic search algorithm is employed. Depending on the heuristic functions used, this algorithm generates only the necessary portion of the reachability graph to find an optimal or near optimal path. In this way, a reasonably good schedule can be generated in a reasonable amount of time.

The scheduling algorithm $L1$ that finds the optimal path is as follows. The algorithm $L1$ is adapted from the well known graph search algorithm A^* [34], [38]. Given a Petri net model, $L1$ expands the reachability graph from the initial marking until the generated portion of the reachability graph touches the final marking. Once the final marking is found, the optimal path is constructed by tracing the pointers that denote the parenthood of the markings, from the final marking to the initial marking. Then, the transition sequence of the path provides the order of the initiations of the activities, i.e., the schedule.

Algorithm L1:
Step 1: Put the initial marking m_0 on the list OPEN.
Step 2: If OPEN is empty, terminate with failure.
Step 3: Remove the first marking m from OPEN and put m on the list CLOSED.
Step 4: If m is the final marking, construct the optimal path from the initial marking to the final marking and terminate.
Step 5: Find the enabled transitions of the marking m.
Step 6: Generate the next marking, or successor, for each enabled transition, and set pointers from the next markings to m. Compute $g(m')$ for every successor m'.
Step 7: For every successor m' of m, do the following.
a: If m' is already on OPEN, direct its pointer along the path yielding the smallest $g(m')$.
b: If m' is already on CLOSED, direct its pointer along the path yielding the smallest $g(m')$. If m' requires pointer redirection, move m' to OPEN.
c: Calculate $h(m')$ and $f(m')$, and put m' on OPEN.
Step 8: Reorder OPEN in the increasing magnitude of f.
Step 9: Go to Step 2.

$L1$ uses a function of the marking m, $f(m) = g(m)+h(m)$. $f(m)$ is an estimate of the cost, i.e., the makespan from the initial marking to the final marking along an optimal path which goes through the marking m. $g(m)$ is the current lowest cost obtained from the initial marking to the current marking m. $h(m)$ is an estimate of the cost from the marking m to the final marking along an optimal path which goes through the marking m.

The scheduling algorithm is admissible [34], i.e., it always finds an optimal path if $h(m)$ satisfies the following condition,

$$0 \leq (m) \leq h^*(m) \text{ for all } m,$$

where $h^*(m)$ is the cost of the optimal path from m to the final marking. The heuristic functions experimented with in this paper do not guarantee this lower bound condition. They are designed to get a reasonably good solution in a reasonable amount of time. Future research includes the development of heuristic functions that guarantee this lower bound condition.

Each arc of the reachability graph corresponds to firing of a transition and has a corresponding cost or time delay. Before a transition can be fired, the tokens of the input places of that transition must be delayed for at least the time delays associated with the input places. Hence, the cost of an arc of the reachability graph, $c(m, m')$ can be considered to be the maximum time delay of the input places, that is,

$$c(m, m') = \max(d_{1j}, d_{2j}, \cdots, d_{kj})$$

where $d_{ij}, i = 1, 2, \cdots, k$ is the time delay of the input place p_{ij} of the transition t_j, and the marking m converts to the marking m' through the transition t_j. But this argument is

not quite correct as explained in the following. Let $d_{\max}$ be $\max(d_{1j}, d_{2j}, \cdots, d_{kj})$. When the marking m converts to m', the tokens of the places that are not input places of t_j, are also concurrently delayed by $d_{\max}$. This is true for every marking update. In fact, therefore, the cost of an arc of the reachability graph is not really the maximum time delay of the input places, but rather the maximum of the remaining time delays of the input places, that is,

$$c(m, m') = \max(r_{1j}, r_{2j}, \cdots, r_{kj})$$

where $r_{ij}, i = 1, 2, \cdots, k$ is the remaining time delay of the input place p_{ij} of t_j. This observation is accounted for in the implementation of $L1$ so that the implementation of $L1$ keeps track of the remaining time delays of the places when it generates the next marking.

The list OPEN maintains the markings generated but not yet explored. These markings form the frontier of the reachability graph. The reachability graph grows pushing this frontier outward from the initial marking until it touches the final marking. Hence, the program does not generate the entire reachability graph, but rather generates the reachability graph layer by layer until the algorithm finds an optimal path to the final marking.

The list CLOSED maintains all the markings generated and explored thus far, i.e., the markings that have been checked against the final marking and found not to be the final marking. In Step 7 of $L1$, if a newly generated marking m' is not on either OPEN or CLOSED, it is put on OPEN to be explored later. If m' is on OPEN, it means a new path to m' from the initial marking has been found. The cost, $g(m')$ of the new path is compared with the cost of the old path. The path, then, is updated to yield the smallest cost. Since m' is still on OPEN, it can be explored in the future.

If m' is on CLOSED, it means m' is already explored and a new path to m' has been found. Since m' is already explored, some descendant markings of m' are already generated. If the paths leading to the descendant markings are followed all the way down, each path ends with a marking which is on OPEN. If the new path to m' has a cost lower than the cost of the old path, there are two options to handle this situation. First, the paths to the descendant markings of m' can be redirected propagating changes all the way down to the markings that are on OPEN. Second, as the algorithm $L1$ does, just the path to m' can be redirected and m' can be put on OPEN for re-exploration.

The first approach has the advantage that the descendant markings of m' and the paths to them that have already been generated will not be discarded. The disadvantage of the first approach is the extra effort to propagate changes to all the descendant markings of m' that may be very large in number. The second approach has the advantage of saving the effort of change propagation. Its disadvantage is that it discards previously generated descendant markings of m' and repeats the effort of generating again some of the descendant markings of m'. But some of the descendant markings of m' may not need to be regenerated at all. The advantages and disadvantages of the above two approaches applied to A^* is discussed in more detail in [34], [38].

TABLE III
JOB REQUIREMENTS OF EXAMPLE 2

	J_1	J_1
1	M_1/M_2	M_1/M_3
2	M_2/M_3	$M_1/M_2/M_3$

If no enabled transition is found in Step 5, the marking m represents a deadlock. Then, no next marking for the marking m would be generated in Step 6. Subsequently, Step 7 and Step 8 would not result in any effects. The algorithm then returns to Step 2 to explore another marking on the list OPEN, i.e., to explore another alternative path that could eventually lead to the final marking. In this way, a path that avoids the potential deadlocks can be found if one exists. If the markings are exhausted in Step 2, there is no path connecting the given initial and final markings.

Since the algorithm $L1$ finds a path between any given pair of the initial and the final markings, a schedule from any given state to any desired state can be generated by appropriately setting the initial and the final markings. Therefore, partial scheduling can be handled without any modification of the model. In this paper, the cost function $g(m)$ is the accumulation of time because the focus of this paper is the makespan. But other functions can be used for $g(m)$. For example, if the operating cost of machines is used for $g(m)$, a schedule with the optimal total operating cost can be obtained. Therefore, other performance criteria or multiple performance criteria can be adapted by using appropriate functions or combinations of functions for $g(m)$ and $h(m)$ in the scheduling algorithm.

IV. THE IMPLEMENTATION OF L1 AND SCHEDULING EXAMPLES

The algorithm $L1$ is implemented in C on a DECstation 5000/200. Reordering OPEN in Step 8 is actually combined with putting a marking on OPEN. Hence, OPEN is always in the increasing order of f. By picking up the first marking from OPEN in Step 3, $L1$ always selects the marking that has the smallest estimate of the cost of an optimal path.

In order to find enabled transitions in Step 5, the implementation of $L1$ identifies marked places. For each marked place, the program then identifies its output transitions. For each output transition, the program checks if all its input places are marked. If they are, the transition is enabled and used in Step 6 to generate the next marking. When generating the next marking, the program updates the marking and calculates the cost functions g and f for the next marking. The program also updates the remaining operation times of the places as discussed in Section III.

Example 2 The problem is to schedule an FMS which has three machines, M_1, M_2 and M_3. The system will handle two jobs, J_1 and J_2. Table III shows the job requirements. The operation times are shown in Table IV. The Petri net model of the system with the initial marking is shown in Fig. 7.

The scheduling problem given in Example 2 is solved by $L1$ using different h functions. $L1$ is executed using the following

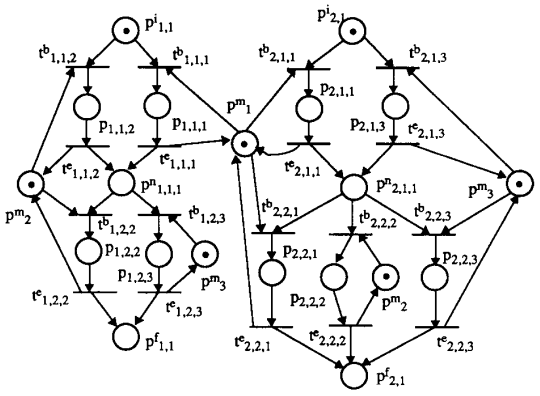


Fig. 7. The Petri net model of the system of Example 2.

TABLE IV
OPERATION TIMES OF EXAMPLE 2

Op.	Time
$O_{1,1,1}$	3
$O_{1,1,2}$	4
$O_{1,2,2}$	3
$O_{1,2,3}$	2
$O_{2,1,1}$	4
$O_{2,1,3}$	2
$O_{2,2,1}$	3
$O_{2,2,2}$	4
$O_{2,2,3}$	4

four h functions.

$$h_1(m) = 0 \text{ for all marking } m. \quad (1)$$

$$h_2(m) = -\ \text{dep}(m), \text{ where dep(m) is the depth of the marking } m \text{ in the reachability graph.} \quad (2)$$

$$h_3(m) = \min(rt_{1m}, rt_{2m}, \cdots, rt_{km}), \text{ where } rt_{im},\ i = 1, 2, \cdots, k \text{ is the remaining operation time of the place } p_{im} \text{ that has a token under the marking } m. \quad (3)$$

$$h_4(m) = h_2(m) + h_3(m) = \min(rt_{1m}, rt_{2m}, \cdots, rt_{km}) - \text{dep}(m). \quad (4)$$

$h_1(m)$ makes $f(m)$ equal to $g(m)$ and the program utilizes no heuristic information. It selects the next marking that has the smallest actual cost to reach the marking from the initial marking. This uninformed best-first search corresponds to the uniform-cost search and is guaranteed to find the optimal path [38].

$h_2(m)$ makes the program prefer markings that are deeper in the reachability graph, that is,

$$f(m) = g(m) + h(m) = g(m) + h_2(m) = g(m) - \text{dep}\ (m). \quad (5)$$

Hence, even if a marking m has a large cost, the cost is compensated if m is deep in the reachability graph. The compensation takes into account that a marking deeper in the reachability graph may be closer to the final marking.

TABLE V
THE EXECUTION RESULTS

	Number of iterations	Depth of final marking	Cost of solution path	Transition firing sequence
$h_1(m)$	50	8	6	$t^b2,1,3t^b1,1,1t^e2,1,3$ $t^e1,1,1t^b2,2,1t^b1,2,3$ $t^e1,2,3t^e2,2,1$
$h_2(m)$	13	8	6	$t^b2,1,3t^b1,1,1t^e2,1,3$ $t^b2,2,2t^e1,1,1t^b1,2,3$ $t^31,2,3t^2,2,2$
$h_3(m)$	54	8	6	$t^b2,1,3t^b1,1,2t^e2,1,3$ $t^b2,2,1t^e1,1,2t^b1,2,3$ $t^e2,2,1t^e1,2,3$
$h_4(m)$	53	8	6	$t^b2,1,3t^b1,1,1t^e2,1,3$ $t^e1,1,1t^b2,2,1t^b1,2,3$ $t^e1,2,3t^e2,2,1$

TABLE VI
THE SCHEDULE OF EXAMPLE 1 GENERATED USING $h_1(m)$

$h_1(m)$	Operation sequence	Operation time
1	$O_{2,1,3}$	2
2	$O_{1,1,1}$	3
3	$O_{2,2,1}$	3
4	$O_{1,2,3}$	2
Sum of operation times		10
Makespan of schedule		6

TABLE VII
THE SCHEDULE OF EXAMPLE 1 GENERATED USING h_2(m)

$h_1(m)$	Operation sequence	Operation time
1	$O_{2,1,3}$	2
2	$O_{1,1,1}$	3
3	$O_{2,2,2}$	4
4	$O_{1,2,3}$	2
Sum of operation times		11
Makespan of schedule		6

$h_3(m)$ favors a marking which has an operation ending soon. Completing an operation makes resources available for other operations. $h_4(m)$ combines $h_2(m)$ and $h_3(m)$. Except for $h_1(m)$, the heuristic functions are intended to find a near optimal schedule in a shorter time. Table V shows the results of the executions employing the above four h functions.

The number of iterations is the number of times the program repeats its main routine. The depth of the final marking indicates the total number of transition firings to reach the final marking from the initial marking. The cost of the path is the makespan of the generated schedule. Since the exploration of a new marking requires an iteration, the number of iterations is always greater than or equal to the depth of the final marking. The transition firing sequence gives the schedule.

The interpretation of the schedule is straightforward from the Petri net model and is given in Table VI–IX. The operation sequence tells the order in which each operation should be initiated at the given machine.

In Table V, all the schedules generated have the same total cost, or makespan, of 6. In Table VI–IX, the sequential sum of operation times of each schedule is 10 or 11. This demonstrates that each schedule fosters concurrency of operations.

TABLE VIII
THE SCHEDULE OF EXAMPLE 1 GENERATED USING h_3(m)

$h_3(m)$	Operation sequence	Operation time
1	$O_{2,1,3}$	2
2	$O_{1,1,2}$	4
3	$O_{2,2,1}$	3
4	$O_{1,2,3}$	2
Sum of operation times		11
Makespan of schedule		6

TABLE IX
THE SCHEDULE OF EXAMPLE 1 GENERATED USING h_4(m)

$h_4(m)$	Operation sequence	Operation time
1	$O_{2,1,3}$	2
2	$O_{1,1,1}$	3
3	$O_{2,2,1}$	3
4	$O_{1,2,3}$	2
Sum of operation times		10
Makespan of schedule		6

TABLE X
THE JOB REQUIREMENTS OF EXAMPLE 3

	J_1	J_2	J_3	J_4	J_5
1	M_1/M_3	M_1/M_2	$M_1/M_2/M_3$	M_2/M_3	M_1/M_3
2	M_2	M_3	M_2/M_3	M_1/M_3	M_2/M_3
3	M_1/M_3	M_1/M_2	M_1/M_3	M_2/M_3	M_1/M_2
4	M_1/M_2	M_1/M_3	M_1/M_2	$M_1/M_2/M_3$	$M_1/M_2/M_3$

Notice $h_2(m)$ reduced the number of iterations quite significantly, i.e., 74% less than the case of $h_1(m)$, i.e., the uninformed best-first search. This means that the program generated a schedule with the same makespan in a much shorter time using function $h_2(m)$. This suggests the value of heuristic function $h_2(m)$. On the other hand, $h_3(m)$ slightly increased the number of iterations to generate a schedule with the same makespan. This shows that only favoring an operation ending soon, in fact, suggests little heuristic value in a resource-sharing interactive environment. Notice that $h_3(m)$ dominates over $h_2(m)$ as shown in the result of the case of $h_4(m)$.

Example 3: The problem is to schedule an FMS with three multipurpose machines, M_1, M_2, and M_3. There are five jobs, J_1, J_2, J_3, J_4, and J_5 that have four processes each. Each job has a lot size of 10. Table X shows the job requirements. Operation times are given in Table XI.

The Petri net model of the system is decomposed according to each job and the part for J_1 with the initial marking is shown in Fig. 8. The parts for J_2, J_3, J_4, and J_5 can be similarly constructed using the method described in Section II. The places with the same place notations are actually the same places and are drawn separately for clear presentation. The Petri net model of the system has a total of 69 places and 82 transitions. The number inside a place represents the number of tokens.

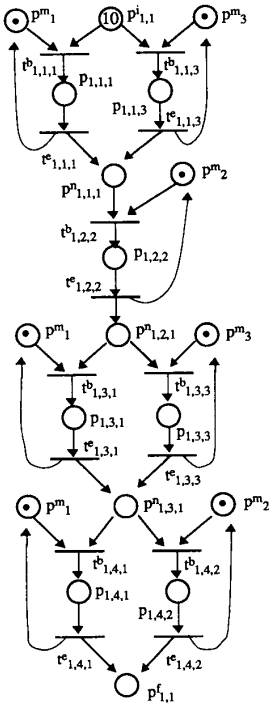


Fig. 8. The model of J_1 of Example 3.

TABLE XI
THE OPERATION TIMES OF EXAMPLE3.

Operation	Time	Operation	Time
$O_{1,1,1}$	7	$O_{3,4,1}$	6
$O_{1,1,3}$	4	$O_{3,4,2}$	3
$O_{1,2,2}$	3	$O_{4,1,2}$	9
$O_{1,3,1}$	3	$O_{4,1,3}$	5
$O_{1,3,3}$	6	$O_{4,2,1}$	6
$O_{1,4,1}$	2	$O_{4,2,3}$	2
$O_{1,4,2}$	4	$O_{4,3,2}$	7
$O_{2,1,1}$	8	$O_{4,3,3}$	12
$O_{2,1,2}$	12	$O_{4,4,1}$	9
$O_{2,2,3}$	4	$O_{4,4,2}$	6
$O_{2,3,1}$	7	$O_{4,4,3}$	3
$O_{2,3,2}$	14	$O_{5,1,1}$	10
$O_{2,4,1}$	8	$O_{5,1,3}$	15
$O_{2,4,3}$	4	$O_{5,2,2}$	7
$O_{3,1,1}$	10	$O_{5,2,3}$	14
$O_{3,1,2}$	15	$O_{5,3,1}$	5
$O_{3,1,3}$	8	$O_{5,3,2}$	8
$O_{3,2,2}$	2	$O_{5,4,1}$	4
$O_{3,2,3}$	6	$O_{5,4,2}$	6
$O_{3,3,1}$	2	$O_{5,4,3}$	8
$O_{3,3,3}$	4		

The problem of Example 3 is solved using the following heuristic function.

$$h(m) = -w \bullet \text{dep}(m), \text{ where w is a weight factor.} \quad (6)$$

This heuristic function compensates the cost of the marking m by the weighted depth of the marking. A marking with a

TABLE XII
THE EXECUTION RESULTS OF EXAMPLE 3

w	Number of iterations	Depth of final marking	Cost of solution path
2	758	400	426
5	412	400	447
10	401	400	439
20	401	400	439
50	401	400	439

TABLE XIII
THE REQUIREMENTS OF J_1 TO J_4 OF EXAMPLE 4

	J_1	J_2	J_3	J_4
1	M_1R_2/M_3	M_2R_2/M_5R_1	$M_2R_1/M_4R_1/M_5$	M_2R_2/M_3
2	M_2R_1/M_5R_1	$M_3R_2/M-4$	$M_2R_3/M_3R_1/M_4R_1$	M_3R_1/M_5R_2
3	M_2R_3/M_4	M_1R_2/M_5R_3	M_1R_3	M_4R_2
4	NA	$M_1/M_3R_1/M_5$	M_2R_1	NA
5	NA	NA	M_4/M_5R_2	NA

bigger depth is hypothesized to be nearer to the final marking. The magnitude of the cost of the marking is related to the magnitude of operation times, the lot sizes, and the number of jobs and operations. Therefore, an adequate value of the weight should reflect the mentioned factors.

Table XII shows the results of executions using different values for w. When the value of w was less than 2, the program did not find a path in a reasonable amount of time. This was because the computer used spent most of the time swapping memory. The computer used, a DECstation 5000/200, has 16 megabytes of RAM. The execution was killed after running in the background a few days. When the program found a solution, it took about 5 to 10 minutes. When the value of w changes from 2 to 5, the number of iterations significantly reduces but the makespan of the solution increases. When the value of w changes from 5 to 10 or greater, the number of iterations reduces a little and the makespan of the solution decreases slightly. The results suggest that the program performs desirably when the value of w is 5 or greater since the number of iterations is close to its lower bound, 400, and the increase of the cost of the path is relatively small.

Example 4 The problem is to schedule an FMS which has multipurpose machines, M_1, M_2, M_3, M_4, M_5, and robots, R_1, R_2, R_3. There are 10 jobs, $J_k, k = 1, \cdots, 10$, each with varied number of processes. Tables XIII, XIV, and XV show the job requirements. The operation times are shown in Table XVI. The Petri net model is constructed using the method discussed in Section II. The Petri net model has a total of 154 places and 182 transitions.

The heuristic function of (6) is used to solve the problem. Table XVII shows the execution results when the lot size is fixed at 5 for each job. Next, an experiment with varied lot sizes is conducted. Table XVIII shows the lot size of each job and the execution results are shown in Table XIX. In Table XVII and Table XIX, the program did not find a path in a reasonable amount of time when w is less than 2 and 1.5, respectively.

TABLE XIV
THE REQUIREMENTS OF J_5 TO J_8 OF EXAMPLE 4

	J_5	J_6	J_7	J_8
1	M_2R_1/M_3R_2	M_1R_3/M_3R_2	M_2R_2/M_4R_1	$M_2R_2/M_3/M_4$
2	M_5	M_4R_1/M_3	M_3/M_5	M_4R_3/M_5R_1
3	M_1/M_3R_1	M_3R_3/M_5R_2	NA	$M_2/M_3/M_4$
4	M_2R_1/M_4R_1	$M_1/M_2/M_3$	NA	M_5
5	M_4R_1	M_2R_1/M_4	NA	M_2R_3/M_4R_1
6	NA	$M_2R_1/M_3/M_5$	NA	$M_1/M_2/M_3R_3$
7	NA	NA	NA	$M_1/M_4/M_5R_1$
8	NA	NA	NA	M_2R_1/M_3R_2

TABLE XV
THE REQUIREMENTS OF J_9 AND J_10 OF EXAMPLE 4

	J_9	J_{10}
1	$M_2R_2/M_3/M_5$	M_5
2	M_2	M_1/M_2
3	M_1R_1/M_4	M_5
4	M_4R_1/M_5	M_2R_1/M_3
5	M_1/M_3	NA

TABLE XVI
THE OPERATION TIMES OF EXAMPLE 4

Op.	Time	Op.	Time	Op.	Time	Op.	Time
$O_{1,1,1}$	3	$O_{3,5,4}$	7	$O_{8,8,3}$	7	$O_{8,6,3}$	4
$O_{1,1,3}$	5	$O_{3,5,5}$	5	$O_{6,5,2}$	4	$O_{8,7,1}$	9
$O_{1,2,2}$	4	$O_{4,1,2}$	4	$O_{6,5,4}$	5	$O_{8,7,4}$	8
$O_{1,2,5}$	5	$O_{4,1,3}$	6	$O_{6,6,2}$	3	$O_{8,7,5}$	7
$O_{1,3,2}$	2	$O_{4,2,3}$	7	$O_{6,6,3}$	7	$O_{8,8,2}$	5
$O_{1,3,4}$	3	$O_{4,2,5}$	8	$O_{6,6,5}$	8	$O_{8,8,3}$	6
$O_{2,1,2}$	4	$O_{4,3,4}$	5	$O_{7,1,2}$	8	$O_{9,1,2}$	3
$O_{2,1,5}$	4	$O_{5,1,2}$	4	$O_{7,1,4}$	8	$O_{9,1,3}$	4
$O_{2,2,3}$	4	$O_{5,1,3}$	3	$O_{7,2,3}$	9	$O_{9,1,5}$	4
$O_{2,2,4}$	5	$O_{5,2,5}$	6	$O_{7,2,5}$	7	$O_{9,2,2}$	6
$O_{2,3,1}$	8	$O_{5,3,1}$	9	$O_{8,1,2}$	5	$O_{9,3,1}$	5
$O_{2,3,5}$	7	$O_{5,3,3}$	7	$O_{8,1,3}$	7	$O_{9,3,4}$	7
$O_{2,4,1}$	6	$O_{5,4,2}$	10	$O_{8,1,4}$	8	$O_{9,4,4}$	2
$O_{2,4,3}$	3	$O_{5,4,4}$	9	$O_{8,2,4}$	10	$O_{9,4,5}$	4
$O_{2,4,5}$	5	$O_{5,5,4}$	4	$O_{8,2,5}$	10	$O_{9,5,1}$	5
$O_{3,1,2}$	3	$O_{6,1,1}$	6	$O_{8,3,2}$	4	$O_{9,5,3}$	7
$O_{3,1,4}$	4	$O_{6,1,3}$	6	$O_{8,3,3}$	5	$O_{10,1,5}$	4
$O_{3,1,5}$	5	$O_{6,2,3}$	7	$O_{8,3,4}$	6	$O_{10,2,1}$	5
$O_{3,2,2}$	5	$O_{6,2,4}$	5	$O_{8,4,5}$	7	$O_{10,2,2}$	6
$O_{3,2,3}$	5	$O_{6,3,3}$	4	$O_{8,5,2}$	4	$O_{10,3,5}$	7
$O_{3,2,4}$	5	$O_{6,3,5}$	4	$O_{8,5,4}$	3	$O_{10,4,2}$	2
$O_{3,3,1}$	6	$O_{6,4,1}$	7	$O_{8,6,1}$	5	$O_{10,4,3}$	3
$O_{3,4,2}$	3	$O_{6,4,2}$	8	$O_{8,6,2}$	5		

From the results shown in Table XVII and Table XIX, the program performs desirably when the weight w is greater than or equal to 5. There is little room for improvement in terms of reducing the number of iterations after w reaches 5. A strong correlation between the number of iterations and the weight w is suggested especially from Table XIX.

V. CONCLUSION

The new scheduling method presented in this paper formulates a scheduling problem with a timed place Petri net

TABLE XVII
THE EXECUTION RESULTS OF EXAMPLE 4 WITH LOT SIZE OF 5

w	Number of iterations	Depth of final marking	Cost of solution path
2	465	450	304
3	465	450	304
4	453	450	304
5	451	450	298
10	451	450	298
100	451	450	298

TABLE XVIII
THE LOT SIZES FOR THE SECOND EXPERIMENT OF EXAMPLE 4

Job	J_1	J_2	J_3	J_4	J_5	J_6	J_7	J_8	J_9	J_{10}
Lot size	5	6	4	6	4	5	7	3	5	5

TABLE XIX
THE EXECUTION RESULTS OF EXAMPLE 4 WITH VARIED LOT SIZES

w	Number of iterations	Depth of final marking	Cost of solution path
1.5	1121	420	273
1.51	1012	420	273
1.52	1012	420	273
1.55	885	420	273
1.6	778	420	273
2	499	420	281
5	421	420	293
10	421	420	293

model. Once a complete model of the system is constructed, the scheduling algorithm uses this Petri net model to search a partial reachability graph. Depending on the heuristic functions used, the scheduling algorithm finds a globally optimal or near optimal feasible schedule in terms of the firing sequence of the transitions of the Petri net model.

The formulation explicitly and easily handles the important characteristics of flexible manufacturing systems, such as routing flexibility, shared resources, concurrency and lot sizes, and also facilitates maintaining the coherency of problem models. By setting the initial and the final markings appropriately, partial scheduling can be handled without any modification of the model. By using appropriate functions or combinations of functions in the scheduling algorithm, other performance criteria or multiple performance criteria can be adapted.

Since a complete Petri net model of a system is obtained during the problem formulation, the generated schedule in terms of the firing sequence of the transitions can be used directly by employing Petri net controllers in the supervisory control of the modeled system. Once a schedule to reach a desired final state is generated, potential deadlocks can be avoided by following the schedule. Hence the analytical overhead to guarantee the liveness of the model and the system is eliminated.

An experimental study of the efficiency of some heuristic functions is conducted. The function of (6) with properly tuned weight w is a potentially good heuristic function. Further research is under way to explore other heuristic functions which can be used to reduce the search. In this paper, the material transfer among machines is not explicitly discussed but the flow of parts is also important to scheduling. Automated guided vehicle systems (AGVS's) are increasingly used for the material handling. Further research is under way to handle collectively the AGVS's and the part processing facilities such as machines and robots [27], [28].

ACKNOWLEDGMENT

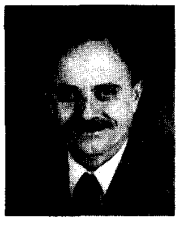
The authors are very grateful to the three anonymous reviewers for their important and constructive comments.

REFERENCES

[1] S. Ahmad and B. Li, "Robot control computation in microprocessor systems with multiple arithmetic processors using a modified DF/IHS scheduling algorithm," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 19, no. 5, pp. 1167–1178, 1989.
[2] M. Aicardi, A. D. Febbraro, and R. Minciardi, "Analysis of deterministic discrete event systems via minimax algebra," in *Proceedings of the 1991 IEEE International Conference on Systems, Man, and Cybernetics*, Charlottesville, VA, Oct 13–16, 1991, pp. 321–328.
[3] R. Y. Al-Jaar and A. A. Desrochers, "Performance evaluation of automated manufacturing systems using generalized stochastic Petri nets," *IEEE Transactions on Robotics and Automation*, vol. 6, no. 6, pp. 621–639, 1990.
[4] K. R. Baker, *Introduction to Sequencing and Scheduling*. New York: John Wiley & Sons, 1974.
[5] E. Bowman, "The schedule-sequencing problem," *Operations Research*, vol. 7, no. pp. 621–624, 1959.
[6] P. J. Brucker, "Scheduling problems in connection with flexible production systems," in *Proceedings of the 1991 IEEE International Conference on Robotics and Automation*, Sacramento, CA, April 1991, pp. 1778–1783.
[7] J. Carlier and P. Chretienne, "Timed Petri net schedules," in *Advances in Petri Nets*, Rozenberg, Ed., Berlin: Springer-Verlag, 1988, pp. 62–84.
[8] C. L. P. Chen, "Time lower bound for manufacturing aggregate scheduling problems," *Proceedings of the 1991 IEEE International Conference on Robotics and Automation*, Sacramento, CA, April 1991, pp. 830–835.
[9] R. Conway, W. Maxwell, and L. Miller, *Theory of Scheduling*. Reading, MA: Addison-Wesley, 1967.
[10] E. Falkenauer and S. Bouffouix, "A genetic algorithm for job shop," in *Proceedings of the 1991 IEEE International Conference on Robotics and Automation*, Sacramento, CA, April 1991, pp. 824–829.
[11] S. French, *Sequencing and Scheduling: An Introduction to the Mathematics of the Job-Shop*. Ellis Horwood, 1982.
[12] S. B. Gershwin and R. R. Hildebrant, et al., "A control perspective on recent trends in manufacturing systems," *IEEE Control Systems Magazine*, vol. 6, no. 2, pp. 3–15, 1986.
[13] H. P. Hillion and J. Proth, "Performance evaluation of job-shop systems using timed event-graphs," *IEEE Transactions on Automatic Control*, vol. 34, no. 1, pp. 3–9, 1989.
[14] D. J. Hoitomt, J. B. Perkins. and P. B. Luh, "Distributed scheduling of job shops," in *Proceedings of the 1991 IEEE International Conference on Robotics and Automation*, Sacramento, CA, April 1991, pp. 1067–1072.
[15] D. J. Hoitomt, P. B. Luh, and K. R. Pattipati, "A Lagrangian relaxation approach to job shop scheduling problems," in *Proceedings of the 1990 IEEE International Conference on Robotics and Automation*, Cincinnati, OH, May 1990, pp. 1944–1949.
[16] D. J. Hoitomt, P. B. Luh, and K. R. Pattipati, "Job shop scheduling," in *Proceedings of the First International Conference on Automation Technology*, Taipei, Taiwan, July 1990, pp. 565–574.
[17] M. A. Holliday and M. K. Vernon, "A generalized timed Petri net model for performance analysis," *Proceedings of the IEEE International Workshop on Timed Petri Nets*, Torino, Italy, July 1–3, 1985, pp. 181–190.
[18] L. E. Holloway and B. H. Krogh, "Synthesis of feedback control logic for a class of controlled Petri nets," *IEEE Transactions on Automatic Control*, vol. 35, no. 5, pp. 514–523, May 1990.
[19] E. Horowitz and S. Sahni, *Fundamentals of Computer Algorithms*. Rockville, MD: Computer Science Press, 1978.
[20] E. S. H. Hou and H. Y. Li, "Task scheduling for flexible manufacturing systems based on genetic algorithms," in *Proceedings of the 1991 IEEE International Conference on Systems, Man, and Cybernetics*, 1991, pp. 397–402.

[21] C. Q. Jiang, M. G. Singh and K. S. Hindi, "Optimized routing in flexible manufacturing systems with blocking," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 21, no. 3, pp. 589–595, 1991.
[22] P. Kapasouris and D. Serfaty, *et al.*, "Resource allocation and performance evaluation in large human-machine organizations," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 21, no. 3, pp. 521–532, 1991.
[23] R. Larson and A. Odoni, *Urban Operations Research*. Englewood Cliffs, NJ: Prentice-Hall, 1981, pp. 360–361.
[24] A. E. J. Lee, "Integrated tooling and scheduling of flexible machines: Theory and algorithms," Ph.D. Dissertation, Rensselaer Polytechnic Institute, Troy, NY, May 1989.
[25] D. Y. Lee and F. DiCesare, "FMS scheduling using Petri nets and heuristic search," in *Proceedings of the 1992 IEEE International Conference on Robotics and Automation*, Nice, France, May 1992, pp. 1057–1062.
[26] D. Y. Lee and F. DiCesare, "Experimental study of a heuristic function for FMS scheduling," in *Proceedings of the 1992 Japan-USA Symposium on Flexible Automation*, San Francisco, CA, July 13–15, 1992, pp. 1171–1177.
[27] D. Y. Lee and F. DiCesare, "Integrated models for scheduling flexible manufacturing systems," in *Proceedings of the 1993 IEEE Intl. Conf. on Robot. & Auto.*, Atlanta, Georgia, May 2–7, 1993, pp. 827–832.
[28] D. Y. Lee and F. DiCesare, "Scheduling of flexible manufacturing systems employing automated guided vehicles," in *Proceedings of the 9th International Conference on CAD/CAM, Robotics, and Factories of the Future*, Newark, New Jersey, USA, August 17–20, 1993, in press.
[29] P. S. Liu and L. C. Fu, "Planning and scheduling in a flexible manufacturing system using a dynamic routing method for automated guided vehicles," in *Proceedings of the 1989 IEEE International Conference on Robotics and Automation*, Scottsdale, AZ, 1989, pp. 1584–1589.
[30] Z. P. Lo and B. Bavarian, "Job scheduling on parallel machines using simulated annealing," *Proceedings of the 1991 IEEE International Conference on Systems, Man, and Cybernetics*, 1991, pp. 391–396.
[31] P. B. Luh and D. J. Hoitomt, *et al.*, "Schedule generation and reconfiguration for parallel machines," *IEEE Transactions on Robotics and Automation*, vol. 6, no. 6, pp. 687–696, Dec. 1990.
[32] A. Manne, "On the job-shop scheduling problem," *Operations Research*, vol. 8, no. pp. 219–223, 1960.
[33] P. Mellor, "A review of job shop scheduling," *Operational Research Quarterly*, vol. 17, no. 2, pp. 161–171, June 1966.
[34] N. Nilsson, *Principles of Artificial Intelligence*. Palo Alto, CA: Tioga, 1980.
[35] J. O'Brien, *Scheduling Handbook*. New York: McGraw-Hill, 1969.
[36] K. Onaga, M. Silva and T. Watanabe, "On periodic schedules for deterministically timed Petri net systems," in *Proceedings of the 4th International Workshop on Petri Nets and Performance Models*, Melbourne, Australia, Dec. 2–5, 1991, pp. 210–215.
[37] T. A. Owens and P. B. Luh, "A job completion time estimation method for work center scheduling," in *Proceedings of the 1991 IEEE International Conference on Robotics and Automation*, Sacramento, CA, April 1991, pp. 110–115.
[38] J. Pearl, *Heuristics: Intelligent Search Strategies for Computer Problem Solving*. Reading, MA: Addison-Wesley, 1984.
[39] C. Ramamoorthy and G. Ho, "Performance evaluation of asynchronous concurrent systems using Petri nets," *IEEE Transactions on Software Engineering*, vol. 6, no. 5, pp. 440–449, 1980.
[40] J. Rickel, "Issues in the design of scheduling systems," in *Expert Systems and Intelligent Manufacturing*, Oliff, Ed. Elsevier, 1988, pp. 70–89.
[41] F. Rodammer and J. K. White, "A recent survey of production scheduling," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 18, no. 6, pp. 841–851, 1988.
[42] R. V. Rogers and K. P. White, "Algebraic, mathematical programming, and network models of the deterministic job-shop scheduling problem," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 21, no. 3, pp. 693–697, 1991.
[43] R. Sengupta and S. Lafortune, "Optimal control of a class of discrete event systems," *Proceedings of the IFAC International Symposium on Distributed Intelligence Systems*, Arlington, VA, Aug 13–15, 1991, pp. 25–30.
[44] L. Shen and Q. Chen, *et al.*, "Truncation of Petri net models of scheduling problems for optimum solutions," *Proceedings of the Japan/USA Symposium on Flexible Automation*, San Francisco, CA, July 13–15, 1992, pp. 1681–1688.
[45] H. Shih and T. Sekiguchi, "A timed Petri net and beam search based on-line FMS scheduling system with routing flexibility," *Proceedings of the 1991 IEEE International Conference on Robotics and Automation*, Sacramento, CA, April 1991, pp. 2548–2553.
[46] J. Sifakis, "Performance evaluation of systems using nets," *Net Theory and Applications*. Brauer, Ed. Berlin: Springer-Verlag, 1980, pp. 307–319.
[47] M. Silva and R. Valette, "Petri nets and flexible manufacturing," *Advances in Petri Nets*. Berlin: Springer-Verlag, 1989, pp. 374–417.
[48] A. S. Spachis and J. R. King, "Job-shop scheduling heuristics with local neighbourhood search," *International Journal of Production Research*, vol. 17, no. 6, pp. 507–526, 1979.
[49] A. Vasquez and P. B. Mirchandani, "Concurrent resource allocation for production scheduling," *Proceedings of the 1991 IEEE International Conference on Robotics and Automation*, Sacramento, CA, April 1991, pp. 1060–1066.
[50] N. Viswanadham and Y. Narahari, "Stochastic Petri net models for performance evaluation of automated manufacturing systems," *Information and Decision Technologies*, vol. 14, no. pp. 125–142, 1988.
[51] D. Zhang, "Planning using timed Pr/T nets," in *Proceedings of the Japan/USA Symposium on Flexible Automation*, San Francisco, CA, July 13–15, 1992, pp. 1179–1183.
[52] Y. Zhang, "Solution to job-shop scheduling of FMS by neural networks," in *Proceedings of the 1991 IFAC Workshop on Discrete Event System Theory and Applications in Manufacturing and Social Phenomena*, Shenyang, China, June 25–27, 1991, pp. 261–266.

Doo Yong Lee (S'90–M'93) earned the B.S. degree from the Department of Control and Instrumentation Engineering, Seoul National University, Seoul, Korea, and the M.S. and Ph.D. degrees from the Department of Electrical, Computer, and Systems Engineering, Rensselaer Polytechnic Institute, Troy, NY. He is currently a postdoctoral Research Associate at Information Technology Services, Rennselaer Polytechnic Institute. His research interests include the modeling, analysis, and control of discrete event systems; Petri net theory and application; scheduling and supervisory control of flexible manufacturing systems and automated guided vehicle systems; and multimedia theory and application. He received the Charles M. Close Doctoral Prize from Rensselaer Polytechnic Institute in 1993. He is a member of the IEEE Robotics and Automation Society and the IEEE Systems, Man, and Cybernetics Society.



Frank DiCesare (S'60–M'62) received the B.S. and M.S. degrees in Electrical Engineering from Northwestern University, Boston, MA, in 1960 and 1962, respectively.

In 1969 he joined the Rensselaer Polytechnic Institute, Troy, NY, where he is presently a Professor of Electrical, Computer, and Systems Engineering. His current research focus and graduate-level teaching is in the area of modeling, analysis, and control of discrete event systems, with application to manufacturing and intelligent vehicular highway systems. At the undergraduate level he teaches courses in microprocessor systems and embedded control. From 1956 to 1960 he worked as a Cooperative Student Engineer at the M.I.T. Instrumentation Laboratory, Cambridge, MA, and from 1960 to 1962 he was a graduate Research and Teaching Assistant at Northeastern University. As an officer in the U.S. Army Signal Corps, he served from 1962 to 1964 as an Instructor and Course Director at the Ordinance Guided Missile School, Redstone Arsenal, Huntsville, AL. Following this, he spent 1964–1969 at Carnegie-Mellon University as an Instructor in Electrical Engineering and as a Research Engineer in the Transportation Research Institute. He has consulted on information and control systems for numerous companies and agencies at all levels of government and the private sector.

He was a recipient of the 1965 Koppers Fellowship, received the IEEE Centennial Award and Medal in 1984, was cited in the 1987 LEAD Award to Rensselaer Polytechnic Institute by the Society of Manufacturing Engineers, and received the Franklin V. Taylor Award from the IEEE Systems, Man, and Cybernetics Society in 1993. He is very active in the IEEE, both in publication and conferences, and currently serves as Vice President of Long-Range Planning and Finance of the Systems, Man, and Cybernetics Society, and is Co-Chair of the 1994 SMC Conference in San Antonio, TX.