国际生产研究杂志
卷。49，第 23 期，2011 年 12 月 1 日，6963–6980

Taylor & Francis
Taylor & Francis Group

研究文章

用于解决多标准灵活作业车间调度问题的禁忌搜索算法

Geoffrey Vilcot [ab] * 和 Jean-Charles Billaut [b]

[A]
Volume Software，20 rue Dublineau BP2630, 37026 Tours Cedex 1，法国；
[b]法国-拉伯雷图尔大学，信息实验室，
64 Avenue Jean Portalis, 37200 图尔, 法国

（最终版本于 2010 年 8 月收到）

我们在本文中考虑的问题是来自印刷和纸板行业的柔性作业车间调度问题。必须最小化两个标准：完工时间和最大延迟时间。提出了两种禁忌搜索算法来寻找一组非支配解：第一种是基于受第二个标准（-约束方法）限制的一个标准的最小化，第二个是基于线性的最小化标准的组合。这些算法在文献中的基准实例上进行了测试，并讨论了结果。总迟到被视为第二次禁忌搜索的第三个标准，并对结果进行了介绍和讨论。

关键词：调度；加工车间; 禁忌搜索；多目标

一、简介

在本文中，我们考虑来自印刷和纸板行业的一个问题。在这种行业中，车间由多台并行的机器组成，主要是印刷机、梳理机和折页机。通常，不止一台印刷机能够执行印刷操作，不止一台折叠机能够执行折叠操作等。因此，除了调度操作的问题，即确定每个操作的开始时间的问题之外，存在分配问题，即确定执行机器的问题。由于所有要执行的作业并不以相同的顺序访问车间的机器，因此作业的路由并不相同，我们说我们面临着作业车间环境。这个问题在文献中是已知的，称为灵活作业车间调度问题（FJSSP）。此外，决策者通常不仅对单个目标函数表现良好的解决方案感兴趣，而且还对几个标准（如完工时间或流程时间）和涉及迟到度量的标准之间的最佳折衷解决方案感兴趣。下面，我们首先考虑目标函数中的两个标准，最后考虑三个标准。双标准问题涉及完工时间和最大迟到最小化，并且考虑的附加标准是总迟到。目的是为决策者提供一组与非支配标准向量相对应的解决方案，为了更方便也称为帕累托最优解或非支配解决方案。

*通讯作者。电子邮件：g.vilcot@volume-software.com

ISSN 0020–7543 印刷版/ISSN 1366–588X 在线版

http://dx.doi.org/10.1080/00207543.2010.526016 http://www.tandfonline.com

文献中包含大量涉及车间调度问题、调度中的多标准方法（T'kindt 和 Billaut 2006）以及调度和分配问题的文章。涉及多个标准的 FJSSP 的文章较少。

Jurisch (1992) 针对多用途机器作业车间调度问题提出了分支定界算法和一些启发式算法。对于 FJSSP，可以在 Paulli (1995) 中找到分层方法，其中第一步是解决分配问题，第二步是解决作业车间调度问题。Dauze` re-Pe´ re` s 和 Paulli (1997) 提出了以完工时间为目标函数的 FJSSP 禁忌搜索。卡塞姆等人。(2002) 提出了进化算法和模糊逻辑的混合。Mastrolilli 和 Gambardella (2000) 提出了 FJSSP 的两种邻域结构。对所提出的邻域的兴趣之一是任何具有空邻域的可行解决方案都是最优的。阿尔瓦雷斯-瓦尔德斯等人。(2005) 为 FJSSP 提出了一种启发式算法，专门用于具有特殊约束（如无等待或重叠）的玻璃工厂行业。目标是找到一个时间表，其标准基于提前和迟到处罚。作者使用了两步算法。第一步使用优先级规则来解决问题，第二步使用本地搜索改进该解决方案。何等人。(2007) 提出了一种架构，其中将学习模块引入到求解 FJSSP 的进化算法中。完工时间被最小化，作者表明，对于某些文献实例，他们的方法优于现有方法（Brandimarte 1993，Kacem 等人 2002）。乔比内等人。(2006) 针对具有序列相关设置时间的单机问题提出了一种多目标禁忌搜索算法。我们参考 T'kindt 和 Billaut (2006) 对多标准调度问题进行更精确的调查。

我们在本文中提出了两个版本的禁忌搜索算法来解决该问题。第一个版本包括解决问题的约束方法。这意味着假设第二个标准以 为界，则第一个标准被最小化。通过迭代地修改界限，可以确定一组非支配解（是否受权衡曲线支持）。禁忌搜索算法的第二个版本包括寻找最小化标准线性组合的解决方案。通过修改与标准相关的权重，还可以确定一组非支配解决方案，仅限于“受支持”的解决方案。

第 2 节用符号给出了问题的正式定义。第 3 节解释了两种禁忌搜索算法。第 4 节介绍并讨论了计算实验。第三个标准与实验一起介绍，因为它不影响算法描述。第五节总结了本文并给出了一些未来的研究方向。

2. 问题定义和符号

我们考虑的问题是 FJSSP，它是经典作业车间调度问题的推广。我们认为，必须在由 m 个分离资源组成的 M 组上调度由 n 个作业组成的 J 组作业。每个作业 J_j 需要 n_j 个连续操作才能完成，并且作业 J_j 的操作 i 由 $O_{i,j}$ 表示。每个操作 $O_{i,j}$ 可以由集合 $R_{i,j}$ 中的任何资源处理，集合 $R_{i,j}$ 是包含可以执行 $O_{i,j}$ 的资源的 M 的子集。我们假设操作 $O_{i,j}$ 的处理时间（用 $p_{i,j}$ 表示）不依赖于执行资源。这个假设可以在不改变方法有效性的情况下放宽。每个作业 J_j 还关联有一个截止日期，用 d_j 表示。

表 1. 符号列表。

中号	：资源集
米	：资源数量，m ¼jMj
MK	：资源编号 k
J	： 工作组
n	：职位数量，n ¼jJj
杰杰	：作业编号 j
nj	：J_j 中的操作数
$O_{i,j}$	：J_j 的操作编号 i
$R_{i,j}$	：能够执行 $O_{i,j}$ 的资源集
,j	：$O_{i,j}$ 的处理时间
dj	：J_j 的预产期

Cj : J j 的完成时间
C max : 完工时间

LP max T j :: 最大迟到总迟到

问题是为每个操作设置执行资源和开始时间。我们用 C j 表示作业 J j 的完成时间。为了衡量调度的质量，需要考虑三个经典标准：由 C max ¼ max J j 2 J C j 定义的完工时间 C max，由 L max ¼ max J j 2 J ðC j d j 定义的最大迟到 L max Þ 以及总迟到情况
PP
T j 由 j2J T j ¼ j2J max(0, C j d j) 定义。

这个问题是强 NP 困难的。表 1 总结了这些符号。

3. 禁忌搜索算法

禁忌搜索算法的工作原理如下。它需要一个初始解决方案和一个邻域结构，并通过使用移动从一个解决方案转换到另一个解决方案来进行。检查当前解决方案的所有邻居并选择最佳可接受的移动。请注意，此举动可能会降低解决方案的质量。禁忌列表存储了所有以前被利用但现在被禁止的动作。禁忌搜索算法的一般描述可以在 Glover (1989) 中找到，并且 FJSSP 的首次应用可以在 Brandimarte (1993) 中找到。

提出了两种禁忌搜索算法：

。第一个对应于 -constraint 方法并用 TS 表示。
。第二个最小化标准的线性组合并用 TS'c 表示。

下面，我们首先介绍两种算法的共同点，然后继续介绍每种算法的具体部分。

3.1 两种算法的共同部分

共同的元素是解的编码、初始解、邻域结构以及最后的禁忌列表定义。

3.1.1 解决方案的编码

解由经典的合取图 G=(V,E) 表示。顶点集等于 V={O i,j , 1jn, 1in j }[{s,t}，其中 s 为源顶点，t 为汇顶点。

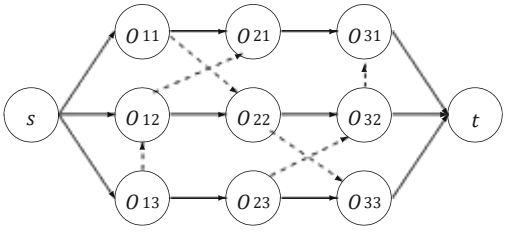

图 1. 解决方案的编码。

如果相应操作之间存在路由约束或资源约束，则两个顶点之间存在边。$O_{n_j,j}$ 和 t 之间有一条边，长度为 $p_{n_j,j}$；$_{s\text{和}}O_{1,j}$之间长度为 0 的边。对于其他边，长度等于边原点处操作的处理时间。该图不包含任何环，并且所有弧的长度均为正。相应调度的完工时间或最大延迟时间的评估可以在多项式时间内精确完成（图 1）。

3.1.2 初始解

初始解由两步贪心算法给出。第一步包括解决每个工序的分配问题，第二步包括解决作业车间问题，而不修改工序的分配。

第一步 – 首先，将操作按 $jR_{i,j}$ j 非递减顺序排序，并按非递增 $p_{i,j}$ 顺序排序以打破平局。操作列表称为 O。一项资源的工作负载由分配的操作的处理时间之和来定义。资源按其工作量非递减顺序排序，该列表称为 R。根据列表 O 进行操作，对于每个操作 $O_{i,j}$，将 R 中属于 $R_{i,j}$ 的第一个资源分配给操作时，该资源的工作负载以及资源的排序都会更新。该过程不断迭代，直到 O 的所有操作都分配给一个资源。

第二步——使用基于“松弛”规则的贪心算法解决作业车间问题。我们用 S 表示候选操作集。一开始，S={$O_{1,j}$, 1jn}。候选操作按其发布时间非递减顺序排序（以及松弛非递减顺序以打破平局）。操作 $O_{i,j}$ 的释放时间是执行资源可用于处理操作 $O_{i,j}$ 的空闲时间与操作 $O_{i1,j}$（如果存在）的完成时间之间的最大值。操作 $O_{i,j}$ 的松弛是作业 J_j 的到期日期与该作业的完成时间之间的差值，就好像剩余操作在 $O_{i,j}$ 之后没有空闲时间处理一样。安排第一个候选操作，更新发布日期，更新 S，并且在 S6/4; 期间迭代该过程。

3.1.3 邻居的定义

邻居是通过移动操作获得的。这种独特的技术允许修改分配和序列。在里面

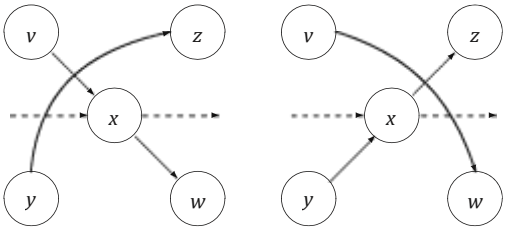

图 2. 在 y 和 z 之间移动操作 x。

联合图，在操作 y 和 z 之间移动操作 x（此移动用 {x,y,z} 表示）意味着：

。删除边 (v,x) 和 (x,w)，其中 v（分别为 w）是前趋执行资源上的 x（分别为后继者），

。添加一条边 (v,w),

。删除边 (y,z)，. 添加边 (y,x) 和 (x,z)。

图 2 说明了图中的移动。

3.1.4 禁忌表

禁忌列表 T' 用于防止搜索在解决方案之间循环。我们的禁忌列表有固定的大小，当列表已满时，最旧的元素将被新元素替换（列表的先进先出管理）。

Dauze` re-Pe´ re` s 和 Paulli (1997) 提出了三种类型的禁忌列表，我们使用他们提出的最佳类型的列表。该禁忌列表的工作原理如下：在 y 和 z 之间移动 x 后，(x,y) 和 (z,x) 将添加到禁忌列表中。如果 (x ^{0} , y ^{0})2T' 或 (z ^{0} , x 0)2T'^{则}禁止移动 { x ^{0} ,y $^{0,z\,0}$ }。

3.1.5 邻域结构

邻域 N 定义如下：对于每个操作 O $_{i,j}$，如果该操作不在第一位置，则该操作与其执行资源上的前一个操作交换。为了改变分配，对于 R $_{i,j}$ 中的每个资源（当前分配除外），O $_{i,j}$ 被移动到在其实际开始时间或紧接着其实际开始时间开始的第一个可能的插入位置。图 3 说明了分配修改（虚线表示未修改的路由约束）。

算法 1 描述了邻域探索。Seq $_{k\text{是在 }M\,k}$ 上排序的操作集合， Seq $_{k,l\text{表示资源 }M\,k}$ 上位置 l 的操作。

SearchInsertion(r,o) 查找操作 o 在资源 M $_{r}$ 上的插入位置。

算法 1：

1：对于所有 M $_{k}$ 2M 所做的事情

2： forl=1 到 jSeq $_{k}$ j do

3： 如果 l2 且 (Seq $_{k,l}$,Seq $_{k,l2}$) 2= T' 且 (Seq $_{k,l1}$,Seq $_{k,l}$) 2= T' 则

4：在 Seq $_{k,\ l2}$ 和 Seq $_{k,\ l1}$ 之间移动 Seq k，1； 评估邻居；撤消移动

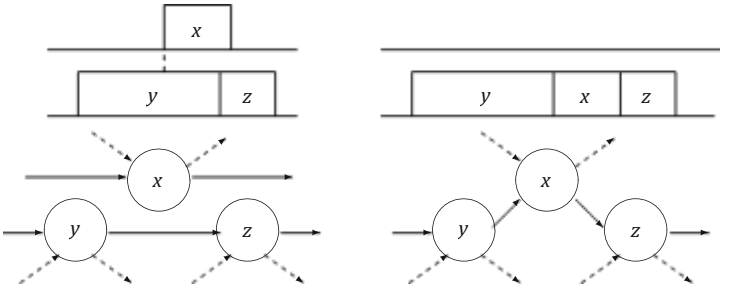

图 3. x 的赋值修改。

5： 结束如果

6： 对于所有 M $_{k\,0}$ 2R $_{Seq\,k,l}$ do

7： l ^{0} SearchInsertion(k ^{0} ,Seq $_{k,l}$) 8： 如果 M $_{k\,0}$ 6¼ M $_{k}$ 和 ðSeq $_{k,l}$,Seq $_{k\,0,l\,0\,1}$ Þ2= T' 和 ðSeq $_{k,l\,0}$,Seq $_{k\,0,l}$ Þ2= T' 则

9：在 Seq $_{k\,0,l\,0\,1}$ 和 Seq $_{k\,0,l\,0}$ 之间移动 Seq $_{k}$,l； 评估邻居；撤消移动

10： 结束如果

11： 结束

12： 结束

13：结束

过程 Evaluate_the_neighbour 存储邻居及其评估（如果它是迄今为止最好的）。对于第一位置的工作，仅探讨了任务修改。

命题 3.1: 邻域 N 是连通的。

证明：考虑初始解 S^i 和目标解 S^t。我们必须证明，经过有限次数的移动后，可以从 S^i 获得 S^t。我们考虑与这些时间表相对应的联合图，用 G^i 和 G^t 表示。从 S^i 开始，始终可以更改操作的分配，使其与 S^t 中的相同。显然，这样的一组移动不能在连接图中生成循环。符号 S^i 用于表示修改阶段中的当前调度。现在我们可以假设在 S^i 中，操作的分配与 S^t 中的相同。

假设 G^t 是拓扑排序的。我们首先考虑 G^t 中的一阶运算。由于这些操作没有前导，因此始终可以通过使用交换移动将它们放在 G^i 中的第一顺序。

现在考虑 G^t 中等级 r 的操作 x 并假设 G^t 中等级 r^0 5r 的操作在 G^i 中处于相同等级。图 4 说明了“唯一的情况”，即 x 与其前任在资源上的交换生成循环（虚线表示路由约束）。请注意，从 w 到 y 的路径可以由多个边组成。x 在 G^i 中的交换移动无法生成图 4(b) 的循环，因为图 4(a) 中 x 之前的操作 z、w 和 y 的秩为 r^0 5r，因此已经在 G^i 中调度。因此，始终可以将 x 与其前任交换，直到其在 G^i 中的等级与 G^t 中的等级相同。通过按照 G^t 的拓扑顺序移动 G^i 的操作，经过有限次数的移动后，我们可以从 S^i 获得 S^t。H

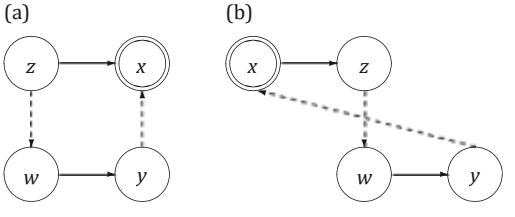

图 4. 交换移动后的循环生成。

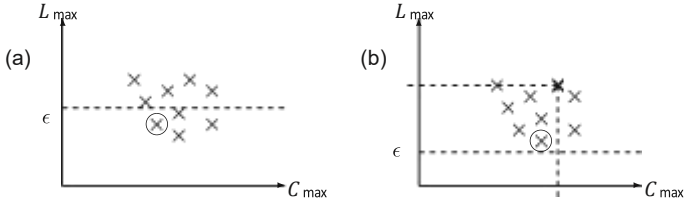

图 5. 最佳邻居和搜索方向。

3.2 TS 算法

使用 -constraint 方法，禁忌搜索算法需要对邻居进行特定的评估。在这里，我们假设最小化的目标是完工时间，并受到最大延迟的限制。

由于我们试图解决约束问题，因此最佳邻居的定义取决于邻居集的结构。解决方案的最佳邻居是 L_{max} 5 的邻居中具有最小 C_{max} 的解决方案（参见图 5 (a)，其中最佳邻居用圆圈表示）。如果所有邻居都满足 L_{max}，则最好的邻居是延迟时间最小的邻居，如果可能的话，其生产时间值不大于当前解决方案的生产时间值（参见图 5 b）。在所有其他情况下，我们选择最小化 L max 的邻居。通过使用最佳邻居的定义，我们首先尝试尊重界限，然后在约束下提高完工时间。

3.3 TS'c 算法

通过标准的线性组合来评估解决方案。考虑的标准是：完工时间 (C_{max}) 和最大延迟时间 (L_{max})。

这种方法的困难之一在于标准范围。例如，完工时间可以等于 1000，最大延迟时间可以等于 10。在这种情况下，完工时间会超过最大延迟时间。此外，不可能先验估计每个标准的范围。为了绕过这个困难，线性组合的系数已被归一化。解决方案的评估由下式给出

$$Z(S) = \frac{C_{最大}(S) - L_{最大}(S)}{\max(C_{max}) - \min(C_{max}) - \max(L_{max}) - \min(L_{max})}$$

和

- {,} 系数，
- $C_{max}(S)$ 和 $L_{max}(S)$ 解决方案的完工时间和最大延迟时间，
- 以及 $\max(C_{max})$、$\min(C_{max})$、$\max(L_{max})$和 $\min(L_{max})$，其计算如下。

我们用 S^{i}_{best} 表示迭代 i 时选择的邻居，令 k 为 TS'c 算法的当前迭代次数。标准的最大值和最小值计算如下：

$$Y = \{S^{i}_{最佳},\ 8i,\ i \le k\};\ \max(C_{max}) = \max_{y \in Y} C_{max}(y);\ \min(C_{max}) = \min_{y \in Y} C_{max}(y)$$

$$\max(L_{max}) = \max_{y \in Y} L_{max}(y);\ \min(L_{max}) = \min_{y \in Y} L_{max}(y)$$

这种标准化方法受到 Deb 等人的启发。（2002）用于计算拥挤距离。

有时，禁忌搜索会陷入深度局部最优，无法离开该解。在这种情况下，我们使用多元化运营商，其灵感来自于战略振荡（其中包括允许在短时间内探索不可行的解决方案，（Downsland 1998））。我们实施的多元化工作如下。在 TS'c 算法中，经过一些没有改进的迭代后，搜索将使用最佳已知解 S^{+}和 随机选择用于线性组合的新系数重新开始。通过使用这种技术，可以探索一些用初始系数永远无法探索的解决方案。继续使用这些新系数进行搜索，直到找到比 S^{+}更好的解（具有正确的系数）或达到极限。在第一种情况下，TS'c 继续从该解开始并使用初始系数，否则搜索结束。

4. 计算实验

使用四个数据集来评估禁忌搜索算法。这些数据集基于 Hurink 的实例（Hurink 等人，1994），源自 Lawrence 的实例。为了在这些数据集中引入截止日期，所使用的方法受到以下启发

德米尔科尔等人。（1998）。J_j 的到期日定义为

其中 T 和 R 是两个参数，已固定为 T=0.3 和 R=0.5。四个数据集如下：

- sdata： Lawrence 的实例（每个操作单个资源），没有分配问题，

。edata： 灵活性很少的实例，接近劳伦斯的原始实例，

。rdata： 平均每个操作具有两个可能资源的实例。vdata： 平均每个操作具有 m/2 可能资源的实例。

实例 la01 到 la05 的大小为 510，即 m=5，n=10，实例 la06 到 la10 为 515，实例 la11 到 la15 为 520，实例 la16 到 la20 为 1010，实例 la21 到 la25 为 1015，实例 la26 到 la30 为 1020 ，实例 la31 至 la35 为 1030，实例 la36 至 la40 为 1515。实验已使用 Intel Core Duo T2400、1.84 GHz、2Go RAM（仅使用一个核心）进行。

4.1 指标定义

4.1.1 表现较弱。如果 S 中没有一个解受 S 中的一个解支配，并且 S 中至少有一个解支配 S 中的一个解，则一组非支配解 S 弱胜于集合 S。图 6 说明了弱表现的概念：集合 S (解决方案由十字表示）弱优于集合 S （解决方案由圆圈表示）。如果 S 弱于 S，则 Ow(S,S) 等于 1 ，否则等于 0。请注意，Ow(S,S)þOw(S,S) 可能等于 0 或 1（但不能等于 2）。

令 exp_i 和 exp_j 为两个测试序列，$S_{exp\,i}$（分别为 $S_{exp\,j}$ ）为 exp_i（分别为 exp_j ）的非支配解集。对于所有 j6=i，SOw(exp_i ,exp_j) 是所有考虑的 OwðS exp_i ,S exp_j Þ 实例的总和。指标 SOw(exp_i) 是以下各项之和

SOw(exp_i ,exp_j) 对于所有 j6=i。如果 N 是测试系列的数量，那么我们有

$$SOwðexp_i Þ¼ \sum_{j=1,j6=i}^{N} SOwðexp_i ,exp_j Þ$$

该指标表示针对所考虑的参数集 exp_i 获得的非支配解集弱于其他解的次数。该指标越大，$S_{exp\,i}$ 越好。

4.1.2 净前沿贡献。考虑两组非支配解 S 和 S，S 表示 S[S 中的非支配解集合。NFC(S,S) 是 S 中 S 的解的比例 。NFCðexp $_i$,exp $_j$ Þ 表示所有考虑实例的 NFCðS $_{exp\,i}$,S $_{exp\,j}$ Þ 的平均值。NFCðexp $_i$ Þ 定义为

$$\overline{NFCðexp_i Þ} \frac{1\times N}{N \quad 1} \sum_{j=1,j6=i} \overline{NFCðexp_i ,exp_j Þ}$$

图 6. 表现较弱的概念。

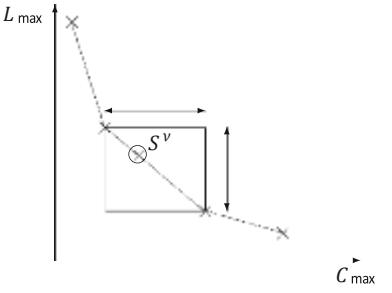

图 7. 拥挤距离的计算。

NFC 指标对应于平均净前沿贡献。该指标越大，$S_{exp\,i}$ 越好。

4.1.3 拥挤距离。拥挤距离（Deb et al. 2002）可用于评估帕累托前沿的多样性。让我们考虑标准空间中的解 S。拥挤距离是“右侧”解和“左侧”解之间矩形的宽度和高度之和（图 7）。极端解的拥挤距离等于无穷大。

指标 CD 是曲线中除极值解之外的所有解之间的平均拥挤距离。该指标越大，非支配解集越好。然而，请注意，生成很少解决方案的方法可能具有重要意义

平均拥挤距离。该指标也被认为是“仅次于”SOw 和 NFC。

4.2 初步实验

为了确定禁忌搜索算法的参数，已经进行了初步实验。参数是 #Max 在停止搜索之前没有改进的迭代次数 (#Max2{100,200,500,800,1000}) 和 jT'j 禁忌列表的大小 (jT'j2{10,40,70,100,150})。这些初步实验已经针对 TS 和 TS'c 算法实现。用于测试的实例是 sdata 、 edata 、 rdata 和 vdata 的 la01、la06、la11、la16、la21、la26、la31 和 la36（每个数据集对应于一个问题大小）。

4.2.1 TS 算法结果

表 2 显示了 TS 算法的结果取决于 #Max 的变化。对于这些实验，T' 的大小已固定为 100。

由于算法的计算时间与参数 #Max 密切相关，因此我们必须修复此参数，以便在计算质量之间取得良好的折衷。

解和计算时间。图 8 说明了 SOw 和 NFC 随计算时间的变化。

我们可以看到，500 秒计算后的改进对于两种性能指标来说并不是真正显着：#Max=500 和 #Max=800 之间的改进

SOw 为 41%，NFC 为 15%，但计算时间增加了 42%。因此对于 TS 算法，#Max 已固定为 500。

表 2. TS 算法的#Max 变化。

#最大限度	母猪	NFC ð%Þ	Average computation 时间（秒）
100	3	35	53
200	13	51	137
500	24	66	第 339 章
800	41	78	第 593 章
1000	48	81	818

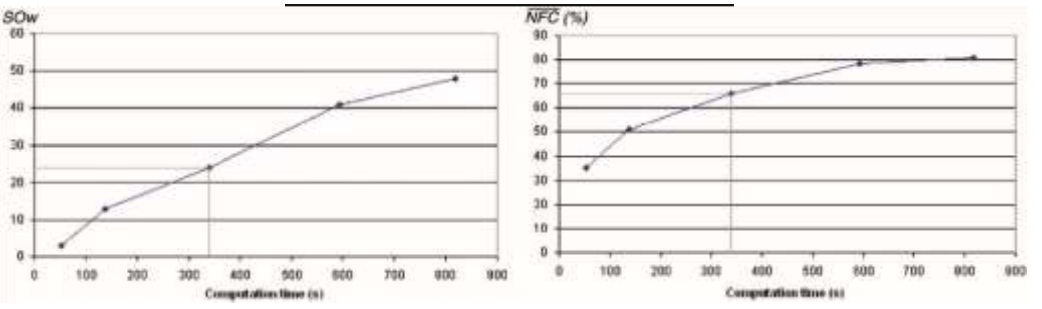

图 8. TS 的 #Max 变化的质量与计算时间。

表 3. TS 算法的 jT'j 变化。

jT‘j	SOw	NFC ð%þ
10	0	15
40	37	55
70	46	67
100	53	70
150	53	71

表 3 显示了根据禁忌列表大小的 TS 结果。对于这些实验，没有改进的最大迭代次数已固定为 500。

由于禁忌列表的大小对解决方案的质量有影响，图 9 显示 SOw 和 NFC 随 jT'j 的变化。参数 jT'j 已固定为 100。

4.2.2 TS'c 算法结果

表 4 显示了 TS'c 算法根据 #Max 值的结果。对于这些实验，禁忌列表大小固定为 100。图 10 显示了 SOw 的变化

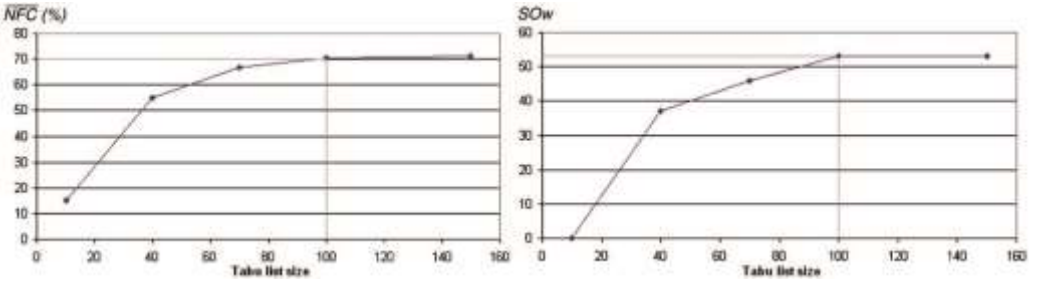

图 9. TS 的 jT'j 变化的质量与 jT'j 的关系。

表 4. TS'c 算法的 #Max 变化。

#最大限度	母猪	NFC ð%Þ	Average computation 时间（秒）
100	1	19	77
200	20	43	155
500	47	68	第 369 章
800	48	74	第 547 章
1000	63	81	650

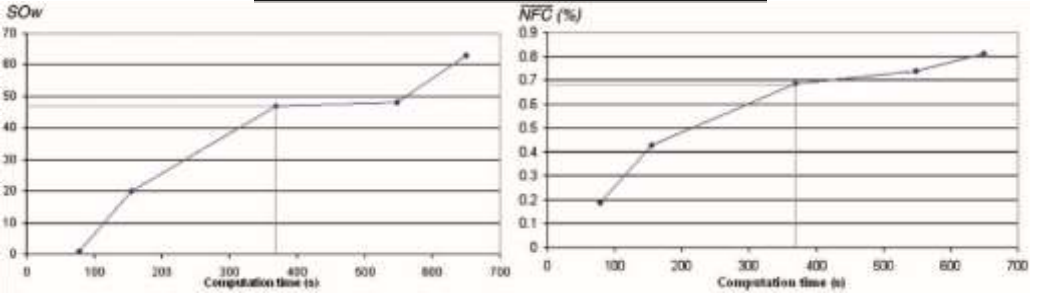

图 10. TS'c 的 #Max 变化的质量与计算时间。

和 NFC 以及计算时间。对于算法 TS，参数#Max 已固定为 500。

表 5 显示了使用线性组合方法对禁忌列表大小变化进行禁忌搜索的结果。对于这些实验，没有改进的最大迭代次数已固定为 500。

图 11 显示了 TS'c 的结果取决于禁忌列表的大小。对于这些实验，没有改进的最大迭代次数固定为 500。我们可以看到，大小 100 是最佳值。对于两种禁忌搜索算法，参数值都固定为#Max=500 和 jT'j=100。

表 5. TS'c 算法的 jT'j 变化。

jT'j	母猪	NFC ð%Þ

10	4	20
40	23	50
70	37	67
100	47	69
150	44	64

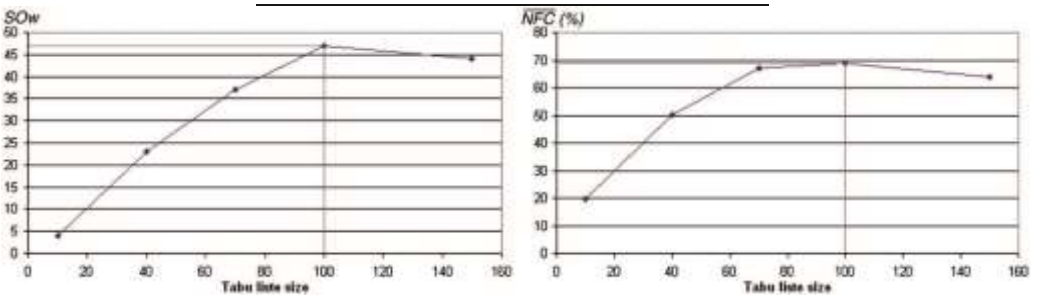

图 11. TS'c 的 jT'j 变化的质量与计算时间。

4.3 结果

在本节中，我们给出三种类型测试的计算结果：考虑单个标准（完工时间）、考虑两个标准（完工时间和最大延迟时间）以及考虑三个标准（加上总延迟时间）。

4.3.1 完工时间最小化

在本节中，我们将 TS 和 TS' c 与 Dauze` re-Pe´ re` s 和 Paulli (1997) 中提出的禁忌搜索算法进行比较。该算法仅优化完工时间。对于线性组合方法，参数为 1/41 和 1/40。对于-constraint 方法，为 1/4。表 6 显示了我们的禁忌搜索的两个版本与 Dauze` re-Pe´ re` s 和 Paulli (1997) 中的禁忌搜索的平均完工时间偏差，用 TSdpp 表示：

$$D\eth TSÞ/\eth C_{max}\ \eth TSÞ\ C_{max}\ \eth TSdppÞÞ{=}C_{max}\ \eth TSÞ$$

与 TS2{TS,TS'c}。如果该值为正，则算法 TSdpp 的性能优于 TS。列 C_{max} (TS)C_{max} (TS'c) 表示算法 TS 执行优于或等于 TS'c 的次数（超过 40 个实例）。D 列表示这两种算法之间的平均偏差：

$$D=\frac{\max\eth C_{max}\ \eth TSÞ,C_{max}\ \eth TS'cÞÞ\ \min\eth C_{max}\ \eth TSÞ,C_{max}\ \eth TS'cÞÞ\ \max\eth C_{max}\ \eth TSÞ,C_{max}\ \eth TS'cÞÞ}{}$$

表 6. 平均完工时间与 TSdpp 的偏差。

数据集	D(TS)(%)	D(TS'c)(%)	最大C (TS)最大C (TS'c)	密 度 (%)
数据	10.5	9.8	11	1.7
电子数据	11.1	11.0	15	2.64
数据	8.5	9.1	24	2.22
虚拟数据	5.3	6.7	28	2.34

我们可以看到 TSdpp 优于 TS 和 TS'c。这并不奇怪，因为我们的禁忌搜索算法是面向多标准的并实现基本的邻域结构，而 TSdpp 仅考虑完工时间最小化并实现用于评估邻域的特定技术。此外，我们可以看到，对于将完工时间最小化视为单一标准，TS 和 TS'c 的表现类似。

4.3.2 双标准问题

为了找到帕累托前沿的近似值，根据禁忌搜索算法使用了两种方法。

对于 TS，一 开始固定为 1。在给定的步骤 k:

。 被固定为 $L^{k}_{max}{}^{1,}$ 其中 $L^{k}_{max}{}^{1}$ 是在步骤 k1 获得的 $L_{max\,值}$。步骤 k 的初始解是步骤 k1 找到的最佳解。

如果找不到可行解，则使用贪心算法（第 3.1 节）获得的初始解和 ¼ $L^{k}_{max}{}^{1}$ 重新启动禁忌。这个过程不断迭代，直到找到可行的解决方案。

对于 TS'c，已经测试了不同的 和值（表 7）。

表 8-10 显示了 TS'c 和 TS 对于所考虑的三个因素的比较性能指标：分别为 SOw、NFC 和 CD。在表 8 中，请注意每个值不能超过 5（大小实例数 (mn)）。表 9 中各值不能超过 100%。最佳值在表 8-10 中以粗体表示。表 8 的结果表明 TS'c 总体上弱于 TS。

表 9 的结果表明 TS 对 Pareto 前沿的贡献大于 TS'c。这意味着在大多数情况下，TS 返回的解决方案集在质量标准方面优于 TS'c 返回的解决方案。该结果证实了中发现的结果表 8 表明，当 TS'c 弱于 TS 时，NFC 指标通常优于 TS'c，反之亦然。

表 10 的结果表明，TS'c 算法的平均拥挤距离高于 TS 算法。这些结果证实 TS'c 优于 TS。

为了在双标准背景下总结这些实验，我们可以说 TS'c 在解决方案质量方面优于 TS。这个结果可能令人惊讶，因为通常认为 -constraint 方法是获得一组非支配解的更好方法（所有非支配解都可以通过使用精确方法来达到），而线性组合算法仅允许获得受支持的解那些。关于计算时间，两种方法的性能相似，即使 TS'c 比 TS 花费的时间稍长。

表 7. 线性组合的测试参数集。

0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
1	0.9	0.8	0.7	0.6	0.5	0.4	0.3	0.2	0.1	0

表 8. SOw 绩效指标。

		TS‘c				TS			
m	n	sdata	edata	rdata	vdata	sdata	edata	rdata	vdata
5	10	2	3	0	0	2	1	0	0

515	1	1	0	0	2	1	1	1
520	0	0	0	0	2	1	1	1
1010	0	0	1	0	1	1	0	0
1015	1	2	1	0	0	1	0	0
1020	1	3	0	0	1	1	1	0
1030	3	3	4	3	2	1	0	0
1515	3	2	0	2	0	2	0	0

Table 9. $\overline{\text{NFC}}$ performance indicator.

		TS‘c				TS			
m	n	sdata(%)	edata(%)	rdata(%)	vdata(%)	sdata(%)	edata(%)	rdata(%)	vdata(%)
5	10	68	73	47	38	85	45	56	62
5	15	62	56	36	30	87	55	64	70
520		38	37	34	34	78	77	66	67
1010		53	42	48	52	52	63	52	48
1015		52	64	43	25	58	36	57	75
1020		47	72	28	37	53	28	72	63
1030		70	65	92	81	40	35	8	19
1515		74	50	49	68	33	50	51	32

表 10. CD $\overline{\text{性能}}$指标。

		TS'c			TS			
米尼	数据	电子数据	数据	虚拟数据	数据	电子数据	数据	虚拟数据
510	1.47	1.25	0.79	0.64	1.29	1.00	0.56	0.69
515	0.92	1.23	0.90	0.59	0.88	0.75	0.61	0.38
520	1.40	1.38	0.67	0.71	1.69	1.14	0.44	0.40
1010	1.17	1.19	0.81	1.22	1.61	1.38	1.15	1.47
1015	1.18	1.17	1.32	0.93	0.87	1.36	1.17	0.65
1020	1.01	0.62	0.61	0.94	1.05	0.40	1.22	0.98
1030	0.72	0.80	0.99	0.84	0.40	0.65	0.60	0.60
1515	1.16	0.93	1.42	1.19	1.55	1.41	0.75	0.60

表 11. 三标准实验的线性组合的测试参数集。

1	0	0	0.5	0.5	0	0.8	0.1	0.1	0.3	0.4	0.3
0	1	0	0.4	0.6	0	0.6	0.2	0.2	0.4	0.2	0.4
0.5	0.5 0.9	0	0.3	0.7	0	0.4	0.3	0.3	0.5	0	0.5
	0.1 0.8	0	0.2	0.8	0	0.2	0.4	0.4	0.1	0.1	0.8
0.2	0.7 0.3	0	0.1	0.9	0	0	0.5	0.5	0.2	0.2	0.6
0.6	0.4	0	0	0	1	0.1	0.8	0.1	0.3	0.3	0.4
		0	0.3	0.3	0.3	0.2	0.6	0.2	0.4	0.4	0.2

表 12. 结果 代表 翁比纳 h ›线性 c 蒂 与 三暴击 奚里亚。

	sdata			电子数据			rdata			虚拟数据		
锰	#索尔	CPU	光盘	#索尔	CPU	CD	#索尔	CPU	CD	#索尔	CPU	光盘
510	6	8.9	0.600	6	12.9	0.451	12	27.7	0.361	12.2	43.1	0.337
515	5.8	29.8	0.440	7	36.1	0.402	13.4	71.2	0.316	15	96.4	0.262
520	3.6	63.6	0.137	7.8	75.4	0.295	11.8	142.6	0.332	11.2	191.8	0.362
1010	8.6	59.7	0.364	5.8	66.5	0.329	4.4	141.1	0.450	4.2	327.9	0.602
1015	3.8	174.8	0.063	7.6	228.6	0.352	7.4	490.7	0.440	6.6	1204.8	0.502
1020	5.2	334.1	0.402	4	444.1	0.387	7	1014.5	0.460	10.4	2337.1	0.313
1030	3.4	733.9	0.049	4.8	966.2	0.556	7.6	2578.8	0.207	11	5805.3	0.337
1515	5.2	425.8	0.411	4.8	479.9	0.280	5.2	1182.9	0.700	6.2	4129.3	0.336

4.3.3 考虑三个标准

仅在三个标准上使用线性组合禁忌搜索算法进行了实验：完工时间、最大迟到时间和迟到时间总和。

$$C_{max}\,\delta S\text{Þ}\;L_{max}\,\delta S\text{Þ}\;PT_j\,\delta S\text{Þ}$$

$$Z\delta S\text{Þ}\diagup \max\delta C_{max}\text{ Þ }\min\delta C_{max}\text{ Þþ }\max\delta L_{max}\text{ Þ }\min\delta L_{max}\text{ Þþ }\max\delta PT_j\text{ Þ }\min\text{ ð }PT_j\text{ Þ}$$

和：

- {„} 系数，
- C_{max} (S)、L_{max} (S) 和 PT_j (S) 分别是解决方案的完工时间、最大迟到时间和总迟到时间，
- max(C_{max}) 和 min(C_{max}) 是在此之前找到的较大和较低的完工时间值（L_{max} 和 PT_j 相同）。

表 11 显示了已测试的 28 个参数值。因此，sdata [edata [rdata [vdata (160 个实例) 的每个实例运行 28 次。

在表 12 中，#Sol 列表示已找到的不同解决方案的平均数量（最多 28 个），CPU 列表示 28 次启动的平均计算时间（以秒为单位），CD 列表示平均多样性。

我们可以注意到，不同主导解决方案的数量明显低于最大值（28），并且这个数量随着灵活性的增加而增加（1010 个问题除外）。对于小型实例来说，计算时间非常短：在

sdata 510 上完成 28 次启动平均需要 8.9 秒，这意味着每次启动平均需要 317 毫秒。在最大的实例 vdata 1030 上，完成 28 次启动需要大约 5805 秒（平均），这意味着每次启动需要 207 秒。

多样性（CD 指标）似乎并不依赖于实例的大小或其灵活性。

为了看出 TS'c 算法引入的多样化的有趣之处，我们计算了搜索过程中的重新启动次数。对于 48% 的测试问题，至少实现了一次重启。这意味着在一半的情况下，该方法有助于提高最终解决方案的质量。对于这 48% 的实例，最大重启次数为 13 次，平均重启次数为 2.01 次，53% 的实例仅重启 1 次。

5. 结论

在本文中，我们解决了多标准 FJSSP。已经提出了两个版本的禁忌搜索算法来提供一组非支配解决方案。一种算法基于 -constraint 方法，另一种算法基于标准的线性组合。分别对单准则问题、二准则问题和三准则问题进行了实验。在单一标准（C max ）的情况下，这些方法不是很有竞争力。对于两个标准，解决方案质量的最佳方法是基于标准线性组合的方法。

在印刷和纸板行业中，在解决问题时通常必须考虑一些额外的资源。未来的研究方向包括考虑某些操作需要不止一种资源才能执行。问题变成了具有多处理器操作的 FJSSP。我们将尝试扩展所提出的算法来考虑这种附加约束。

参考

Alvarez-Valdes, R., et al., 2005。玻璃工厂灵活车间调度的启发式方法。欧洲运筹学杂志，165，525–534。

Brandimarte, P., 1993。通过禁忌搜索在灵活作业车间中进行路由和调度。运筹学年鉴，22，158-183。

Choobineh, F.、Mohebbi, E. 和 Khoo, H.，2006。针对具有序列相关设置时间的单机调度问题的多目标禁忌搜索。欧洲运筹学杂志，175, 318–337。

Dauze` re-Pe´ re` s, S. 和 Paulli, J.，1997。使用禁忌搜索建模和解决一般多处理器作业车间调度问题的集成方法。运筹学年鉴，70，3-20。

Deb, K., et al., 2002。快速且精英的多目标遗传算法：NSGA-II。IEEE 进化计算汇刊，6 (2), 182–197。

Demirkol, E.、Mehta, S. 和 Uzsoy, R.，1998。车间调度问题的基准。欧洲运筹学杂志，109, 137–141。

Downsland, K.，1998。具有禁忌搜索和策略振荡的护士调度。欧洲运筹学杂志，106，393–407。

Glover, F., 1989。禁忌搜索 - 第 I 部分。ORSA 计算杂志，1, 190–206。

Ho, NB、Tay, JC 和 Lai, EM-K.，2007 年。用于学习和发展灵活作业车间计划的有效架构。欧洲运筹学杂志，179，316-333。

Hurink, J.、Jurisch, B. 和 Thole, M.，1994。 Tabu 搜索多用途机器的作业车间调度问题。或 Spektrum，15, 205–215。

Jurisch, B., 1992。在具有多功能机器的商店中安排工作。论文博士学位。德国：奥斯纳布吕克大学。

Kacem, I.、Hammadi, S. 和 Borne, P.，2002。灵活作业车间调度问题的帕累托最优方法：进化算法和模糊逻辑的混合。模拟中的数学和计算机，60，245–276。

Mastrolilli, M. 和 Gambardella, L.-M., 2000。灵活作业车间问题的有效邻域函数。调度杂志，3 (1), 3–20。
Paulli, J., 1995。FMS 调度问题的分层方法。欧洲运筹学杂志，86，32-42。
T'kindt, V. 和 Billaut, J.-C.，2006 年。多标准调度。理论、模型和算法。第二版。柏林：施普林格。