变邻域结构 Tabu 搜索算法及其在 Job Shop 调度问题上的应用

孙元凯,刘 民,吴 澄
(清华大学自动化系,北京 100084)

摘 要: 本文针对最小化完工时间的 Job Shop 调度问题提出一种变邻域结构 Tabu 搜索算法,该算法使用的邻域结构随算法的进程而改变,不仅邻域规模小,而且仍保持了可达性这一重要的属性. 对不同规模的实例进行了数值计算,计算结果表明,该算法具有非常高的效率,且初始解对算法的影响很小.

关键词: 调度; Job Shop; 局部搜索; Tabu 搜索; 邻域结构
中图分类号: TP18 **文献标识码**: A **文章编号**: 0372-2112 (2001) 05-0622-04

Tabu Search Algorithm with Varying Neighborhood and Its Application to Job Shop Scheduling Problem

SUN Yuan-kai, LIU Min, WU Cheng
(*Department of Automation, Tsinghua University, Beijing* 100084, *China*)

Abstract: This paper presents a tabu search algorithm with varying neighborhood for Job Shop scheduling problem of minimizing the makespan. The neighborhood used can be changed with the process of the algorithm, and the scale of the neighborhood is not only small, but also the reachability can be kept. The computational results of different scale problems show that the efficiency of the algorithm is high, and that the influence of the initial solution on the algorithm is small.

Key words: scheduling; job shop; local search; Tabu search; neighborhood

1 引言

Job Shop 问题可以简单地描述如下(French 1982):有一组工件和机器设备;每个工件都包含一系列的操作,每个操作都要占用某一台设备进行加工;操作一旦开始,就不能中途停止;每台机器在任一时刻最多只能加工一个工件. 所谓调度,就是在每台机器上为每个操作分配一段加工时间. 最小化完工时间 Job Shop 问题就是要寻找一种调度方案,使得所有工件的完工时间最短. Job Shop 调度问题已被证明属于 NP 难题. 由于该问题具有广泛的应用背景,许多学者都对该问题进行了研究,并提出了多种优化算法和近优算法.

本文提出一种基于 Tabu 搜索的快速而容易实现的算法. 该算法的核心是根据关键路径上的操作块而定义的一种变结构邻域. 该邻域不仅规模小,而且仍保持了可达性这一重要属性.

2 问题的描述

采用分枝定界算法中常用的非连接图模型来对问题进行描述. 如图 1 所示. 图中顶点 O_{13} 代表第一个工件的第三个操作,其他符号类推.

图 1 中,同一工件的操作,如 $\{O_{11}, O_{12}, O_{13}\}$, $\{O_{21}, O_{22}\}$, $\{O_{31}, O_{32}, O_{33}\}$ 和 $\{O_{41}, O_{42}\}$,这些操作之间由有向弧串联,表示工件的工艺路线,是固有的次序约束. 在同一台机器上加工的所有操作,如 $\{O_{11}, O_{21}, O_{31}, O_{41}\}$, $\{O_{12}, O_{32}\}$ 和 $\{O_{22}, O_{33}, O_{42}\}$,这些操作之间以双向弧连接. 图 1 中所描述的是 4 个工件在 4 台设备上进行加工时的非连接图模型. 也可以将操作的加工时间以权重的方式表示在图中. 调度方案就是要确定图中双向弧的取向问题. 可行的调度方案可表述为: ①图中所有的双向弧均固定为单向弧; ②最终的图中不包含回路. 图 2 就是因图 1 中双向弧取向固定而得到的一种调度方案.

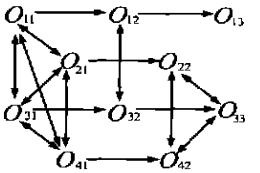

图 1 非连接图模型

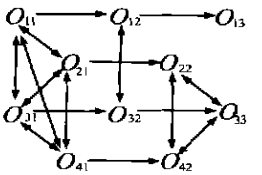

图 2 一种可行的调度方案

收稿日期:1999-11-09;修回日期:2000-12-18
基金项目:国家自然科学基金(No. 60004010)和国家教育部博士点基金

对于某个调度方案,最小完工时间就等于图中最长的加权路径的长度. 该路径通常就称为关键路径(以 CP 来表示),以 Length(CP) 来表示关键路径的长度. 若假设图 2 中所有权重均为 1,则可得其关键路径如下(有两条):

$$CP1: O_{11} \rightarrow O_{31} \rightarrow O_{21} \rightarrow O_{41} \rightarrow O_{42} \rightarrow O_{33}$$

$$CP2: O_{11} \rightarrow O_{31} \rightarrow O_{21} \rightarrow O_{22} \rightarrow O_{42} \rightarrow O_{33}$$

即 Length($CP1$) = Length($CP2$) = 6.

称关键路径上在同一机器上进行加工的所有连续操作为关键块(操作数目需不小于 1). 因此,在 $CP1$ 上有 $O_{11} \rightarrow O_{31} \rightarrow O_{21} \rightarrow O_{41}$ 和 $O_{42} \rightarrow O_{33}$ 两个关键块;在 $CP2$ 上有 $O_{11} \rightarrow O_{31} \rightarrow O_{21}$ 和 $O_{22} \rightarrow O_{42} \rightarrow O_{33}$ 两个关键块.

3　变邻域结构 Tabu 搜索算法

Tabu 搜索算法有几个要素:初始解,移动,邻域,邻域规模和禁忌列表. 下面对这些要素做一些简单的说明."移动"就是从一个解转向另一个解. 对于任一可行解 S,只要定义了其移动的方式,就定义了从该解经过一步移动所能到达的所有其他解,这个所有一步可达的解集就称为 S 的邻域. 邻域规模就是指所定义邻域中元素的数量. 对 S 的邻域进行搜索,按某种标准找出某个元素,替换 S,转而重复刚才的动作,直到达到算法的停止条件. 为了防止震荡(即从 S_1 到 S_2,又从 S_2 到 S_1 反复进行)现象的发生,使用禁忌列表来控制元素的选择. (在下面的叙述中,以 $n \times m$ 的形式表示调度问题的规模,其中 n 代表工件数,m 代表机器数.)

3.1　邻域的构造

Van Laarhoven 等人[2]定义的移动操作是:交换所有关键路径上关键块中相邻的两个操作. 由此所生成的邻域记做 H_2,由于关键路径上可能有 m 个关键块(即 m 台机器),每个关键块上可能有(n - 1)个交换操作(即每台机器上可能会有 n 个工件),所以其邻域规模为 $O(mn)$. 该移动的优点在于,所生成邻域中的解都是可行解. 然而,该移动所定义的邻域相当大,尤其是在大规模问题中. 因为 Tabu 搜索算法的效率主要取决于对每个邻域搜索的效率,Taillard(1989)采用一种计算下界的方法来减少计算的规模. Nowichi 等人采用另外的方法来减少邻域规模[1],他们所定义的移动操作为:任取一条关键路径,仅交换关键块两端的操作,而且对于第一个关键块,仅交换后面的两个操作,对于最后一个关键块,仅交换前面的两个操作. 所生成的邻域记做 H_1,其关键路径上可能有 m 个关键块(即 m 台机器),每个关键块最多有两个交换操作,所以其邻域规模不大于(2m - 2). 以前面的图 2 为例,H_2 = {(O_{11}, O_{31}), (O_{31}, O_{21}), (O_{21}, O_{41}), (O_{42}, O_{33}), (O_{11}, O_{31}), (O_{31}, O_{21}), (O_{22}, O_{42}), (O_{42}, O_{33})},其中前 4 个是 $CP1$ 所产生的,后 4 个是 $CP2$ 所产生的(其中有重复). 若选择 $CP1$,则 H_1 = {(O_{21}, O_{41}), (O_{42}, O_{33})};若选择 $CP2$,则 H_1 = {(O_{31}, O_{21}), (O_{42}, O_{33})}. 显然,H_1 是 H_2 的子集. 在大规模问题中,H_1 将比 H_2 小得多. 在文[1]中,作者还证明了在一步移动过程中,$H_2 \setminus H_1$(差集)中的元素不会对解质量的提高有所帮助. 然而,一步移动中不能提高解的质量并不意味着后续移动中也不会提高解的质量,因此,使用 H_1 的算法比使用 H_2 的算法更快地陷入局部极值点. 更重要的是,H_2 具有可达性而 H_1 不具有可达性[4].

结合 H_2 和 H_1,提出一种变结构邻域 H,其移动操作定义为:选定初始解 S_0,从其 H_1 邻域中选择好的元素进行迭代,直到某个 S_k,其 H_1 邻域中无更好的解. 此时,求 S_k 的 H_2 邻域,从 H_2 邻域中随机选择一个元素作为新的初始解,重复计算,直到满足停止条件. 因此,所定义的邻域结构在计算过程中是变化的. 先使用 H_1 进行计算,陷入极值点后采用 H_2 邻域. 该邻域既具有 H_1 规模小的优点,同时又保留了 H_2 可达性的优点(见附录). 显然,H 是 H_2 的子集,而 H_1 是 H 的子集. 因此,H 中的元素都是可行解.

3.2　禁忌列表

禁忌列表的目的是防止震荡现象的发生. 然而,并不能把所有已经走过的路径都记录下来. 一是因为在具体算法中,不太可能这样实现;另外,在所禁止的移动操作中,可能某些移动会导致更好的解. 因此,一般都将禁忌列表设定为某一长度,随着新解的加入,"老"的解将从列表中删除,从而实现自动解禁.

3.3　邻域搜索策略

对邻域进行搜索的目的是要选出用于下一次迭代的元素. 该元素的选择也有不同的标准. 在算法中,采用了两种方法. 一是最速下降法,即计算邻域中所有的解,从中选择下降幅度最大的一个解,这需要计算邻域中的所有解;另一种方法是就近下降法,即在计算邻域的过程中,只要发现解的质量有所下降,就停止计算邻域中的其他解,选定刚才所计算的元素作为下一步迭代的起点. 另外,若邻域中没有下降解,则看是否能进行平移操作(即解值与迭代起点相同的元素). 若可平移,则进行平移操作. 若无法下降,也无法平移,则表明陷入了局部极值点,此时,改变邻域的结构,从新的邻域中随机选择一个解作为新的迭代起点.

3.4　算法

该算法的计算机实现是比较简单的. 先确定一个初始解. 可采用各种调度规则来生成初始解. 在算法中,为了便于比较,采用了 INSA[1]启发式规则来生成初始解. 下面给出算法的基本流程.

step1:确定初始解. 将初始解作为迭代的起点.

step2:判断是否满足结束条件. 若满足结束条件,给出当前最好解作为最优解.

step3:计算起点的关键路径和关键块.

step4:计算起点的邻域.

step5:根据最速下降法或就近下降法,计算邻域中的解. 若可下降或平移,则将选定的元素作为新的起点,更新禁忌列表,转至 step2. 若无法平移和下降,则重新计算起点的邻域,从新邻域中随机选择一个元素作为新的起点,更新禁忌列表,转至 step2.

算法中,给出的算法停止条件为限定它的总迭代次数. 由于该算法中采用了具有可达性的邻域结构,而且在陷入极值点时采用随机的方式跳出,因此,这种"随机游动"使得初始解

在进行了一定的迭代次数后,有可能达到解集中的所有元素,这当然也包括最优解在内.因此,当迭代次数足够多时,该算法将以概率 1 搜索到最优解(概率收敛性).

4 计算结果分析

为了充分观察算法的效率以及算法各个组成部分对算法效率的影响,设计了多组算法,分别针对不同的问题在小规模和大规模实例上进行了计算实验.所有算法均在 PII266 Windows 98 平台上采用 VC++5.0 工具实现.所采用的小规模问题一共 66 个计算实例:实例 abz5-abz9 是文献[3]中提出的;car1-car8 是 J. Carlier 在"Ordonnancements a contraintes disjonctives"一文中提出的;实例 la01-la40 是 S. Lawrence 在"Resource constrained project scheduling: an experimental investigation of heuristic scheduling techniques"一文中所提出的;{mt06, mt10, mt20}来自 Muth 和 Thompson 编写的"Industrial Scheduling";orb1-orb10 是 1986 年提出的.大规模问题共 20 个实例,编号为 ta61-ta80,是 Taillard E. 在"Benchmarks for basic scheduling problems"一文(European Journal of Operational Research, 64, 1993)中公布的,前 10 个问题的规模是 50×20,后 10 个问题的规模是 100×20.

4.1 可达性属性对算法的影响

(1)为了观察初始解对该算法的影响,针对 66 个小规模实例,设计了两个算法:算法 1 以 FCFS 调度规则生成初始解,算法 2 采用 INSA[1]调度规则生成初始解.两个算法的其他部分均相同:禁忌列表长度均取为机器数量的二分之一,采取普通的搜索控制策略(即按先入先出规则简单地更新禁忌列表),分别运行了 30 次,迭代次数为 10000 次,取平均值作为运算的结果,如表 1 所示(限于篇幅,仅列出了 la31-la40 这 10 个实例的计算结果).在表中,"规模"一栏以 n/m 的形式来表示问题的结构,其中 n 代表工件数量,m 代表机器数量;"最好结果"一栏中列出了到目前为止,该问题所获得的最好解,以上下界的形式表示(上下界相等则表示已获得了最优解).

表 1 不同初始解对算法的影响

问题	规模	已知最好解		算法 1 的平均结果 A(采用 FCFS 初始解)	算法 2 的平均结果 B(采用文献[1]初始解)	二者相对误差 $\lvert A-B\rvert/A$
		下界	上界			
La31	30/10	1784	1784	1810	1806	0.2%
La32	30/10	1850	1850	1935	1967	1.6%
La33	30/10	1719	1719	1784	1746	1.9%
La34	30/10	1721	1721	1835	1853	0.9%
La35	30/10	1888	1888	1946	1940	0.3%
La36	15/15	1268	1268	1382	1365	1.2%
La37	15/15	1397	1397	1517	1524	0.4%
La38	15/15	1184	1217	1312	1296	1.2%
La39	15/15	1233	1233	1303	1297	0.4%
La40	15/15	1222	1222	1311	1310	0.1%

从表 1 中可以看出,初始解的不同,并没有使计算结果有明显的差别,即初始解对算法的计算效果影响不大.

(2)为了进一步观察可达性属性对算法的影响,对于小规模和大规模实例,分别做了比较实验,设计了两个算法:在算法 1 中,使用所定义的具有可达性属性的变结构邻域,在算法 2 中使用了文献[1]中所定义的没有可达性属性的邻域.两个算法的其他部分相同,如均采用 FCFS 生成初始解和采用简单的搜索控制策略(即简单更新禁忌列表)等,分别运行了 30 次,迭代次数限制为 10000 次,取平均值进行比较.小规模实例的计算结果如表 2 所示(其中算法 1 采用了表 1 中的数据,不必重新计算),限于篇幅,仅列出 la31 - la40.大规模实例的计算结果如表 3 所示.

表 2 可达性属性对算法的影响(小规模实例的计算结果)

问题	规模	已知最好解		算法 1 的平均结果 A(邻域具有可达性)	算法 2 的平均结果 B(邻域不具有可达性)	二者相对误差 $(B-A)/B$
		下界	上界			
La31	30/10	1784	1784	1810	1843	1.8%
La32	30/10	1850	1850	1935	1980	2.3%
La33	30/10	1719	1719	1784	1843	3.2%
La34	30/10	1721	1721	1835	1867	1.7%
La35	30/10	1888	1888	1946	1982	1.8%
La36	15/15	1268	1268	1382	1412	2.1%
La37	15/15	1397	1397	1517	1536	1.2%
La38	15/15	1184	1217	1312	1335	1.7%
La39	15/15	1233	1233	1303	1340	2.8%
La40	15/15	1222	1222	1311	1339	2.1%

表 3 可达性属性对算法的影响(大规模实例的计算结果)

问题	规模	已知最好解		算法 1 的平均结果 A(邻域具有可达性)	算法 2 的平均结果 B(邻域不具有可达性)	二者相对误差 $(B-A)/B$
		下界	上界			
ta61	50/20	2868	2868	3045	3656	16.7%
ta62	50/20	2848	2902	3177	3636	12.6%
ta63	50/20	2755	2755	2961	3365	12.0%
ta64	50/20	2697	2702	2926	3312	11.7%
ta65	50/20	2725	2725	2938	3453	14.9%
ta66	50/20	2845	2845	3134	3562	12.0%
ta67	50/20	2812	2841	2999	3414	12.2%
ta68	50/20	2764	2784	2980	3468	14.1%
ta69	50/20	3071	3071	3377	3764	10.3%
ta70	50/20	2995	2995	3342	3618	7.6%
ta71	100/20	5464	5464	6011	6413	6.3%
ta72	100/20	5181	5181	5694	6344	10.2%
ta73	100/20	5552	5568	6221	6660	6.6%
ta74	100/20	5339	5339	5734	6667	14.0%
ta75	100/20	5392	5392	6265	6787	7.7%
ta76	100/20	5342	5342	5856	6307	7.2%
ta77	100/20	5436	5436	6049	6498	6.9%
ta78	100/20	5394	5394	5907	6651	11.2%
ta79	100/20	5358	5358	5921	6146	3.7%
ta80	100/20	5183	5183	5841	5958	2.0%

从表 2 和表 3 中可以看出,算法 1 的效果普遍要好于算法 2,在大规模实例上效果更加明显.如果将所有可行解集比喻为一个"海洋",则使用不具有可达性属性邻域结构的算法就相当于在"海洋"中的一个个"孤岛"上进行搜索,其搜索过程要从一个"孤岛"转移到另一个"孤岛",只能依靠算法重新启动并随机选择初始解,或者依靠复杂的搜索控制策略转移到其他的"孤岛"上;而使用了具有可达性属性的邻域,就相当于在各个"孤岛"之间架设了桥梁,算法的搜索过程就不会限于某个"孤岛",而是可以从一处转移到另一处,从而就有更多的机会找到更好的解,这也说明了算法为什么对初始解并不

敏感的问题.对比表 2 和表 3 可以发现,对于大规模问题,可达性属性显得更加重要,使用具有可达性邻域结构的算法,其效果比不使用可达性邻域的算法高出 10 %左右.通过进一步的分析发现,在大规模问题上,文献[1]中所采用的不具有可达性属性的邻域结构,由于其邻域规模小,使得解集空间割裂的状况加剧,又由于邻域结构不具有可达性,从而在解集的"海洋"中"制造"了更多的"孤岛",这些实际上就是"牺牲"邻域结构的规模来换取算法速度的提高.

4.2　与文献[1]中算法的比较

为了进一步与文献[1]中的算法进行比较,设计了两个算法:算法 1 就是严格按照文献[1]中方法来实现的,算法 2 采用所定义的具有可达性的邻域,两个算法的其他部分均相同,即采用文献[1]中的初始解,都采用复杂的搜索控制策略.对于大规模问题 ta71 - ta80 进行了计算实验,分别运行 30 次,迭代次数限制为 10000 次,取平均值进行比较,计算结果如表 4 所示.

表 4　与文献[1]中的算法进行比较(大规模实例的计算结果)

问题	规模	已知最好解		初始解	算法 1 的平均结果 A	算法 2 的平均结果 B	二者相对误差 $(B-A)/B$
		下界	上界				
ta71	100/20	5464	5464	6234	5643	5567	1.3 %
ta72	100/20	5181	5181	5849	5428	5313	2.1 %
ta73	100/20	5552	5568	6367	5941	5674	4.5 %
ta74	100/20	5339	5339	5920	5442	5391	0.9 %
ta75	100/20	5392	5392	6335	5576	5518	1.0 %
ta76	100/20	5342	5342	5965	5479	5429	0.9 %
ta77	100/20	5436	5436	6107	5792	5689	1.8 %
ta78	100/20	5394	5394	6095	5637	5476	2.9 %
ta79	100/20	5358	5358	5935	5441	5393	0.9 %
ta80	100/20	5183	5183	5975	5263	5247	0.3 %

从表 4 中可以看出,算法 2 的效果比算法 1 好,说明邻域结构的可达性对算法的效率确有影响.而且对比表 4 的第 6 列(使用文献[1]中的算法)和表 3 的第 5 列(除了搜索控制策略外,与文献[1]中的算法相同),可以看出,使用复杂的搜索策略确实可以大大提高算法的搜索效率.因此,有理由相信,文献[1]中的算法具有较高的效率,与它使用的复杂搜索控制策略有很大的关系.

5　结论

本文所提出的变结构邻域的本质就是在到达局部极值点时候,能够跳出局部极值点.为了更容易跳出局部极值点,在改变邻域结构时,可以从 $H_2 \setminus H_1$(差集)中进行随机选择.另外,该算法也可以采用并行实现的机制,采用不同的方法求得各自的初始解.

数值计算的结果表明,由于变结构邻域不仅规模小(算法未陷入局部极值点时采用 H_1 邻域,其规模不大于 $2m$;只有在跳出局部极值点时才采用 H_2 邻域,其规模为 $O(mn)$),而且保有了可达性这一重要属性,从而使得算法具有很高的搜索效率,同时也说明了当邻域规模很小时,可达性对于算法的效率有不可忽视的影响.如果能结合现有的一些技术,如变长禁忌列表和复杂的搜索控制策略,相信该算法的效率还会相应提高.

附录: H 邻域具有可达性.

证明:假设最优解为 S_{opt},最优解的关键路径记为 $CP(S_{opt})$.任一给定初始非最优解 S_0,使用 H_1 邻域求得极值点 S_0',假设 S_0 不是最优解,其关键路径记为 $CP(S_0)$,令 $\pi' = \{CP(S_0) \setminus CP(S_{opt})\}$,令 $\pi = \{\pi$ 中次序约束与 S_{opt} 中次序约束不同的弧线$\}$.先证明 π 非空.反正法:若 π 为空,则说明 S_{opt} 中也包含了 $CP(S_0)$ 这条路径,从而根据关键路径的定义,在 S_{opt} 中就有 $\mathrm{Length}(CP(S_0)) < \mathrm{Length}(CP(S_{opt}))$,但 S_0' 非最优解,因此又有 $\mathrm{Length}(CP(S_0)) > \mathrm{Length}(CP(S_{opt}))$,从而矛盾.既然 π 非空,从中任选一弧,按 S_{opt} 中的约束关系固定,得到一个新解 S_1.显然,新解 $S_1 \in H_2(S_0)$.同理,若 S_1 不是最优解,又可重复上面的动作,直到所获得的 π 为空集.又 S_{opt} 中弧线的数量是有限的,因此,必可在有限步内达到 π 为空集的状态,从而达到最优解.

参考文献:

[1] Eugeniusz Nowichi & Czeslaw Smutnichi. A fast taboo search algorithm for the Job shop problem [J]. Management Science, 1996, 42(6): 797 - 813.

[2] Van Laarhoven P. J. M., E. H. L. Aarts, and J. K. Lenstra. Job shop scheduling by simulated annealing [J]. operations Research, 1992, 40(1): 113 - 125.

[3] Adams J., E. Balas, and D. Zawark. The shifting bottleneck procedure for job shop scheduling [J]. Management Science, 1988, 34(3): 391 - 401.

[4] Peter Bruker. Scheduling Algorithm [M]. Springer-Verlay Berlin, Heidelberg, 1998(2nd).

[5] Carlier J. And E. Pinson. An algorithm for solving the job-shop problem [J]. Management Sci., 1989, 35(2): 164 - 176.

作者简介:

孙元凯　1972 年生.1991 年 9 月进入清华大学自动化系学习,1996 年 7 月获得学士学位.1996 年 9 月进入清华大学自动化系,在国家 CIMS 工程研究中心攻读博士学位.研究方向主要是车间生产过程的建模、调度和优化控制.

刘　民　1965 年生,清华大学自动化系国家 CIMS 工程技术研究中心讲师、博士,现为中国自动化学会名词委员会副主任委员,已在国内外刊物和会议上发表学术论文 20 多篇,目前感兴趣的领域为复杂制造系统智能优化理论与方法、人工生命、进化计算等.

吴　澄　中国工程院院士,国家 863 计划自动化领域首席科学家,清华大学自动化系教授,博士生导师,国家 CIMS 工程技术研究中心主任.复杂制造系统智能优化理论与方法、人工生命、进化计算等.