第20卷 第4期
2011年8月
运 筹 与 管 理
OPERATIONS RESEARCH AND MANAGEMENT SCIENCE
Vol. 20,No. 4
Aug. 2011

并行机问题的模拟退火调度算法研究

史 烨[1,2], 李 凯[1]

(1. 合肥工业大学 管理学院,安徽 合肥 230009; 2. 中国科技大学 管理学院,安徽 合肥 230026)

摘 要:研究了一类调度目标是最小化最大完成时间的并行机调度问题。考虑到此问题的 NP-hard 特性,引入模拟退火算法思想以获取高质量近优解。分析了现有此问题模拟退火算法的缺陷,定义了关键机器和非关键机器,设计了一个包含局部优化的模拟退火算法。除了交换变换,还引入插入变换以改变各子调度中作业个数。大量的随机数据实验用于验证算法解的质量和计算效率,实验结果表明该模拟退火算法能够在有限时间内为大规模问题求得高质量满意解。

关键词:调度; 并行机; 最大完工时间; 模拟退火

中图分类号:TP301;O223　　文章标识码:A　　文章编号:1007-3221(2011)04-0104-04

Research on Simulated Annealing Scheduling Algorithm for Parallel Machine Problem

SHI Ye[1,2], LI Kai[1]

(1. *School of Management*, *Hefei University of Technology*, *Hefei* 230009, *China*; 2. *School of Management*, *University of Science and Technology of China*, *Hefei* 230026, *China*)

Abstract: This paper considers a kind of parallel machine scheduling problem with minimizing makespan. Considering the NP-hardness of this problem, the simulated annealing approach is introduced to obtain the near-optimal solutions with high quality. The limitation of the existing simulated annealing algorithm for this problem is analyzed. Two kinds of machines, crucial machine and non-crucial machine, are identified. A simulated annealing algorithm including a process of local optimization is designed. Apart from the swap operation, the insertion operation is presented to change the number of the sub-schedules. A large set of randomly generated instances are made to test the solution quality and the efficiency of the algorithm. Computational results show that the simulated annealing algorithm proposed in this paper can solve the problems with large size in a reasonable time.

Key words: scheduling; parallel machine; makespan; simulated annealing

0 引言

并行机问题是一类重要的调度问题并一直受到广泛的关注,相关的研究成果也较丰富。Lee et al.[1]、Cheng & Sin[2]、Mokotoff[3]等对现有研究成果进行了总结。Li & Yang[4]从建模方法、松弛技术和算法设计等方面对一类并行机调度问题进行了较为详细地综述。

收稿日期:2009-12-02

基金项目:国家高技术研究发展计划(863)重点项目(2008AA042901);国家自然科学基金项目(70631003,70871032,70971035);安徽省自然科学基金资助项目(11040606Q27);合肥工业大学博士专项科研资助基金项目(GDBJ2010-025).

作者简介:史烨(1987),男,江苏常州人,硕士研究生,研究方向为生产调度,优化算法;李凯(1977),男,安徽蒙城人,讲师,博士,研究方向为生产调度,企业能源管理,优化算法,人工智能等。

本文将针对一类并行机调度问题进行研究。其中给定一组作业：$J_1, J_2, \cdots, J_n$，每个作业均可调度到一组机器 $M_1, M_2, \cdots, M_m$ 中的任意某一机器上加工。一个作业在每一时刻只能被一个机器加工，并且一个机器在同一时刻只能加工一个作业。作业不允许中断。所有机器的处理能力均相同，假定处理速度均为 1。因此，作业 $J_j(j=1,2,\cdots,n)$ 在任意机器上加工对应的处理时间都相同，设为 p_j。设 $\prod$ 为调度方案的全集，$\pi \in \prod$ 为某一可行调度方案，则 π 可以表示为 $\pi = \{\pi^1, \pi^2, \cdots, \pi^m\}$，其中 $\pi^i = \{\pi_1^i, \pi_2^i, \cdots, \pi_{ni}^i\}$ 为 M_i 机器上的子调度，π_k^i 为 π 调度方案中机器 M_i 上加工的第 k 个作业，n_i 为 π^i 子调度中的作业个数，$\sum_{i=1}^{m} n_i = n$。用 C_k^i、$C_{\max}^i$、$C_{\max}$ 分别表示作业 π_k^i 的完成时间、π^i 子调度中的最大作业完成时间和全部作业中的最大完成时间，则 $C_{\max} = \max_{i=1}^{m} C_{\max}^i = \max_{i=1}^{m} \max_{k=1}^{n_i} C_k^i$，调度的目标是找到能够使得最大完工时间 $C_{\max}$ 最小化的调度方案 π^*。此问题通常符号表示为 $p_m \| C_{\max}$。$p_m \| C_{\max}$ 是 NP-hard 问题，因为即使是其仅包含 2 个机器的简单特例都是 NP-hard 问题[5]。

$p_m \| C_{\max}$ 问题一类常见的并行机调度问题，也是其他更复杂调度问题的基础。早期关于 $p_m \| C_{\max}$ 问题的研究成果通常是针对某一启发式规则构造相应的启发式算法，或对这些启发式算法进行改进，或证明这些启发式算法的最坏误差界。例如，Graham[6] 将求解 $C_{\max}$ 常用的 LPT（Longest Processing Time first，长作业优先）规则应用到此问题；Coffman et al.[7] 根据装箱问题与此问题的对应关系构造了 MULTIFIT 算法；文献[8,9]证明了 MULTIFIT 算法的最坏误差界优于 LPT 算法；文献[10～12]则对这两种启发式算法进行了改进，以获得更高质量的解。

考虑到此问题的 NP-hard 特性，以及模拟退火（SA，Simulated Annealing）算法在许多生产调度问题中的有效应用[13～15]，我们拟引入 SA 算法求解 $\mathrm{p_m} \| \mathrm{C}_{max}$ 问题。此问题现有求解效果最好的算法是 Lee et al.[16] 给出的 SA 算法，其解的质量远高于前面两种启发式算法。但它仍然存在如下不足：(1) 该算法仅采用交换邻域，无法改变初始解中各子调度中的作业个数，无法保证收敛到最优解；(2) 该算法采用盲目的搜索方法。因此，我们认为有必要引入改变子调度中作业个数的策略和快速寻找比当前解更优解的机制。为此，我们为问题构造了一个新的模拟退火算法。为区别起见，本文用 $\mathrm{SA_1}$ 表示文献[16]中的模拟退火算法，用 $\mathrm{SA_2}$ 表示本文构造的模拟退火算法。

本文剩余章节作如下安排：下一节将详细介绍本文算法 SA_2 的思路，并对构造的模拟退火算法进行描述；第 2 节通过大量的随机实验数据从两个方面验证 SA_2 算法的性能：(1) 比较 SA_1 与 SA_2 的相对质量；(2) 比较 SA_2 解与问题下界的绝对质量，说明其收敛到最优解的程度；最后对论文所做的工作进行总结并分析进一步的研究工作。

1　模拟退火算法

1.1　算法思想

首先，我们给出关键机器和非关键机器的定义。

定义 1　在一个调度解 π 中，若 M_i 机器对应的子调度 π^i 的最大完成时间 $C_{\max}^i = C_{\max}$，则称该机器为关键机器，否则称为非关键机器。

虽然在一个给定的解中关键机器的个数未必为 1，但只有减少关键机器对应子调度的最大完成时间才有可能减少所有作业中的最大完成时间。然而，在文献[16]的模拟退火算法 $\mathrm{SA_1}$ 中并没有考虑到关键机器与非关键机器的区别，仅是盲目地从中任选两个不同的子调度实时交换变换。

由关键机器和非关键机器的定义，易知定理 1 是成立的。

定理 1　若当前调度解 π 不是最优解，则必存在不同机器 M_i 和 M_h，$i \neq h$，$i, h \in \{1,2,\cdots,m\}$，对应的子调度 π^i 和 π^h 满足条件 $C_{\max}^i \neq C_{\max}^h$。

显然，此定理的逆否命题是成立的。即，若对于 $\forall i, h \in \{1,2,\cdots,m\}$，$i \neq h$，均有 $C_{\max}^i = C_{\max}^h$，则此解 π 必为最优解。

可见，只要当前解不是最优解，则其中必存在关键机器和非关键机器。因此我们在构造模拟退火算法 $\mathrm{SA_2}$ 时，首先针对关键机器和子调度中最大完成时间最小的非关键机器，对其中的作业实施交换或插入变

换，从而尽快缩小关键机器对应的最大完成时间。

定义 2　交换：将当前解中不同子调度中的两个作业互换机器及其加工位置。

定义 3　插入：将当前解中某作业插入到其他子调度的某位置进行加工。

Lee et al.[16] 仅采用交换变换为当前解生成邻域，而此种方法无法改变任意子调度中作业个数。显然，若初始解各子调度中作业个数不是恰巧与最优解中对应子调度中作业个数相等，则 SA_1 将无法收敛到问题的最优解。而在 SA_2 中，我们引入插入变换将能够改变各子调度中作业个数，使得算法从理论上有收敛到问题最优解的可能性。

1.2　算法描述

本文为 $p_m \| C_{\max}$ 问题构造的模拟退火算法 SA_2 描述如下：

Step 1　用 LPT 算法获得一个初始解 π，并计算对应的 $C_{\max}$；

Step 2　计算下界 LB：$\sum_{j=1}^{n} p_j / m$；

Step 3　如果 $(C_{\max} - LB)/LB < \varepsilon_1$（这里，取 ε_1 为 0.0001），则返回 π 及对应 $C_{\max}$ 的，结束；

Step 4　选择某一关键机器 M_i，$i := \text{ar max}\{C^l_{\max} \mid l = 1,2,\cdots,m\}$，和某一非关键机器 M_h，$h =: \arg\min\{C^l_{\max} \mid l = 1,2,\cdots,m\}$，记由此 2 个子调度组成的局部调度为 π_{Local}，对应的局部 $C_{\max}$ 值为 Z_{Local}；

Step 5　计算局部 $LB_{Local} := (C^i_{\max} + C^h_{\max})/2$；

Step 6　设定初始温度 $T := (Z_{Local} - LB_{Local}) \times K$（这里 K 取经验值 3000）；

Step 7　如果 $T < \varepsilon_2$（这里，取 ε_2 为 0.0001），或同一温度下未接受新解，则转 Step3；

Step 8　设定同一温度下的迭代长度 $L := 5$；

Step 9　产生一个随机数 r_1 $(0 \le r_1 < 1)$，如果 $r_1 < 0.5$，则转 Step10；否则，转 Step11；

Step 10　选择 π^i 子调度中处理时间最大的作业 π^i_j，$j := \arg\max(p^i_k \mid k = 1,2,\cdots,n_i)$；选择 π^h 子调度中处理时间最小的作业 π^h_q，$q := \arg\min(p^h_k \mid k = 1,2,\cdots,n_h)$；交换作业 π^i_j 和 π^h_q 位置，得到新的局部解 π_{Local}' 和对应的 Z_{Local}'，转 Step12；

Step 11　在子调度 π^i 和 π^h 中随机选择某一作业插入到另外一个子调度中，得到新的局部解 π_{Local}' 和 Z_{Local}' 对应的；

Step 12　计算 $\Delta Z_{Local} := Z_{Local}' - Z_{Local}$，如果 $\Delta Z_{Local} < 0$，则 $\pi_{Local} := \pi_{Local}'$，$Z_{Local} := Z_{Local}'$，转 Step14；

Step 13　产生一个随机数 r_2 $(0 \le r_2 < 1)$，如果 $\text{Exp}(\Delta Z_{Local}/T) > r_2$，则 $\pi_{Local} := \pi_{Local}'$，$Z_{Local} := Z_{Local}'$；

Step 14　$L := L - 1$，如果 $L = 0$，则 $T := T \times \alpha$，转 Step7；否则，转 Step9。

2　实验及其结果分析

考虑到 Lee et al.[16] 中的 SA 算法获得的解质量明显优于其他启发式算法，本节将通过大量随机数据将本文 SA 算法与其进行比较。我们考虑的问题情形包括 2，4，6，8 个机器和 200，300，400，500 个作业。所有算法用 C++ 语言实现。实验的环境为 CPU：Genuine Intel(R) T2050 1.60GHz、内存：1.00GB、操作系统：Microsoft Windows XP Home Edition。所有实验数据均由计算机随机生成，作业处理时间的取值范围为 [101，200]。此生成随机数据的方法部分借鉴了文献 [16] 的方法。为了使得算法性能比较更加客观，对于任意相同的问题规模（相同的机器数和作业数），我们分别采用了 10 组不同的随机算例进行测试，所有计算结果在表 1 中给出。其中 $Cap(H) = (C_{\max}(H) - LB) \times 100/LB$，表示算法 H 对应的 $C_{\max}$ 与问题下界相对误差的百分比；Time(H) 为算法 H 占用的 CPU 时间，以秒为单位；Ave、Best、Worst 分别表示同一规模 10 组不同算例求解中的平均、最好和最坏情形；统计值一栏分别对平均值、最好值和最坏值进行了统计。

从表 1 中数据可以看出：本文提出的模拟退火算法 SA_2 所获得的解的质量明显优于 SA_1，但其所占用的 *CPU* 时间大于 SA_1。对于所有共 $4 \times 4 \times 10 = 160$ 组随机实验，SA_1 所获得的解平均误差为 2.158%，而 SA_2 解的平均误差仅为 0.005%，并且其最坏误差仅为 0.01%，说明其收敛到最优解的程度非常高。SA_1 占用的 CPU 时间较长，平均占用 CPU 时间为 13.65 秒。但是对于所有问题，SA_2 最坏的 *CPU* 占用时间为 20.031 秒，而平均占用 *CPU* 时间为 2.688 秒，这在现实生产生活中的绝大多数环境中是能够接受的。

通过对表 1 中数据分析，我们可以看到 SA_2 具有另外一个优势就是其占用 *CPU* 时间并非完全依赖于问题的规模(问题的作业数和机器数) 。从各种问题规模 10 组不同随机算例的平均占用 *CPU* 时间能够明显看出，文献[16] 给出的 SA_1 占用的 *CPU* 时间依赖于问题的规模。以 4 机器情形为例，作业数从 500 到 200 的 4 种情形，SA_1 平均占用 *CPU* 时间秒数分别为 26. 17、14. 38、8. 317、4. 228，明显呈下降趋势; 而 SA_2 平均占用的 *CPU* 时间秒数分别为 1. 308、3. 308、2. 23、2. 617，其下降趋势不明显。原因在于，SA_1 搜索的时间与其交换邻域中的个数相关，所以问题规模越大，其搜索时间越长; 而 SA_2 的搜索速度与局部模拟退火优化相关，因此与问题规模没有较强的依赖关系，这也为 SA_2 求解超大规模的问题提供了可能性。

表 1　实验数据

n	m	GAP(SA_1)			TIME(SA_1)			GAP(SA_2)			TIME(SA_2)		
		Ave	Best	Worst	Ave	Best	Worst	Ave	Best	Worst	Ave	Best	Worst
200	2	1. 275	0. 006	3. 824	4. 117	3. 953	4. 735	0. 005	1. 04E-05	0. 01	1. 828	0. 156	3. 343
200	4	2. 193	0. 847	4. 003	4. 228	3. 984	4. 781	0. 005	0. 000901	0. 008	2. 617	0. 5	9. 188
200	6	3. 459	1. 343	9. 072	3. 995	3. 968	4. 016	0. 005	0. 000331	0. 01	2. 825	0. 547	9. 781
200	8	4. 809	1. 88	9. 9	4. 153	4	4. 61	0. 005	0. 000276	0. 01	3. 347	0. 422	10. 688
300	2	0. 621	0. 118	1. 233	8. 572	8. 219	9. 719	0. 004	0	0. 009	1. 925	0. 297	9. 25
300	4	1. 696	0. 401	3. 907	8. 317	8. 297	8. 329	0. 005	0. 00057	0. 009	2. 23	0. 344	5. 187
300	6	2. 021	1. 01	4. 702	8. 038	8. 016	8. 047	0. 003	0. 000236	0. 006	2. 301	0. 359	4. 484
300	8	3. 454	1. 354	8. 234	8. 005	7. 985	8. 016	0. 006	0. 001463	0. 01	2. 509	0. 375	7. 718
400	2	0. 501	0. 037	1. 528	14. 44	14. 38	14. 5	0. 005	0. 000428	0. 009	1. 831	0. 125	3. 953
400	4	1. 432	0. 617	3. 153	14. 38	14. 33	14. 41	0. 004	0. 000211	0. 009	3. 308	0. 437	8. 297
400	6	2. 708	1. 184	6. 331	13. 92	13. 89	13. 95	0. 006	0. 000874	0. 009	1. 581	0. 219	5. 703
400	8	3. 01	1. 183	5. 753	13. 61	13. 58	13. 66	0. 007	0. 00341	0. 01	4. 681	0. 484	10. 641
500	2	0. 726	0. 337	1. 42	24. 08	22. 34	29. 2	0. 006	0. 000994	0. 01	1. 906	0. 313	5. 531
500	4	1. 496	0. 397	3. 147	26. 17	22. 31	52. 28	0. 005	0. 001007	0. 008	1. 308	0. 094	4. 422
500	6	2. 828	1. 144	4. 249	30. 01	22. 36	59. 34	0. 004	0. 000557	0. 01	4. 428	0. 187	13. 219
500	8	2. 29	0. 734	4. 635	32. 33	21. 81	86. 88	0. 005	0. 000884	0. 009	4. 387	0. 546	20. 031
统计值		2. 158	0. 006	9. 9	13. 65	3. 952	86. 88	0. 005	0	0. 01	2. 688	0. 094	20. 031

3　结论及进一步的工作

本文为 $p_m \parallel C_{max}$ 问题构造了一个模拟退火算法。通过对现有该问题模拟退火算法缺陷的分析，我们在构造模拟退火算法时定义了关键机器和非关键机器。针对关键机器上的子调度进行局部优化，使得搜索具有较强的目的性。考虑到初始解中各子调度中作业个数未必能够恰巧等于最优解中各相应的子调度作业个数，我们不仅可以对当前解进行交换变换，也可以进行插入变换。大量的随机数据实验表明，本文构造的模拟退火算法能够在大约 3 秒的时间求解 500 个作业规模的问题，解相对于下界的平均误差仅为 0. 010% 。

本文进一步的工作将致力于该算法计算效率的提高，验证其对超大规模(例如作业数大于 1000) 问题求解的适用性，同时也可以考虑对本文算法思想进行拓展，求解更复杂并行机调度问题。

参考文献:

[1] Lee C Y，Lei L，Pinedo M. Current trends in deterministic scheduling [J]. Annals of Operations Research，1997. 70: 1-41.

[2] Cheng T C E，Sin C. A state-of－the－art review of parallel－machine scheduling research [J]. European Journal of Operational Research，1990，47: 271-292.

[3] Mokotoff E. Parallel machine scheduling problems: a survey [J]. Asia-Pacific Journal of Operational Research，2001，18 (2): 193-242.

[4] Li K，Yang S L. Non-identical parallel-machine scheduling research with minimizing total weighted completion times: models，relaxations and algorithms [J]. Applied Mathematical Modelling，2009，33(4): 2145-2158.

[5] Lenstra J K，Rinnooy Kan A H G，Brucker P. Complexity of machine scheduling problems [J]. Annals of Discrete Mathematics，1977，1: 343-362.

(下转第 112 页)

参考文献:

[1] Garey M, Tarjan R, Wilfong G. One-processor scheduling with symmetric earliness and tardiness penalties [J]. Mathematics of Operations Research, 1988, 13: 330-348.

[2] Baker K, Scudder G. Sequencing with earliness and tardiness penalties: a review [J]. Operations Research, 1990, 38: 22-36.

[3] Conway R W, Maxwell W L, Miller L W. Theory of scheduling [M]. Reading, MA: Addison-Wesley, 1967.

[4] Qi X, Tu F S. Scheduling a single machine to minimize earliness penalties subject to the SLK deadline determination method [J]. European Journal of Operational Research, 1998, 105: 502-508.

[5] Chand S, Schneeberger H. Single machine scheduling to minimize weighted earliness subject to no tardy jobs [J]. European Journal of Operational Research, 1988, 34: 221-230.

[6] Pathumnakul S, Egbelu P J. Algorithm for minimizing weighted earliness penalty in single - machine problem [J]. European Journal of Operational Research, 2005, 161: 780-796.

[7] Valente J M S. Local and global dominance conditions for the weighted earliness scheduling problem with no idle time [J]. Computers and Industrial Engineering, 2006, 51: 765-780.

[8] Kanet J J, Sridharan V. Scheduling with inserted Idle time: problem taxonomy and literature review [J]. Operations Research, 2000, 48(1): 99-110.

[9] Mandel M, and Mosheiov G. Minimizing maximum earliness on parallel identical machines [J]. Computers and Operations Research, 2001, 28: 317-327.

[10] Gonzalez T, Sahni S. Open shop scheduling to minimize finish time [J]. Journal of the ACM, 1976, 23(4): 665-679.

(上接第 107 页)

[6] Graham R L. Bounds on multiprocessor timing anomalies [J]. SIAM Journal on Applied Mathematics, 1969, 17: 416-429.

[7] Coffman E G, Garey M R, Johnson D S. An application of bin-packing to multipossor scheduling [J]. SIAM Journal on Computing, 1978, 7: 1-17.

[8] Friesen D K. Tighter bounds for the multifit processor scheduling algorithm [J]. SIAM Journal on Computing, 1984, 13: 170-181.

[9] Yue M. On the exact upper bound for MULTIFIT processor algorithm [J]. Annals of Operations Research, 1990, 24: 233-259.

[10] Lee C Y, Massey J D. Multiprocessor scheduling: combining LPT and MULTIFIT [J]. Discrete Applied Mathematics, 1988, 20: 233-242.

[11] Fatemi G, Jolai G. A pairwise interchange algorithm for parallel machine scheduling [J]. Production Planning and Control, 1998, 9: 685-689.

[12] Gupta J N D, Ruiz - Torres A J. A LISTFIT heuristic for minimizing makespan on identical parallel machines [J]. Production Planning and Control, 2001, 12: 28-36.

[13] Crauwels H A J, Potts C N, Wassenhove L N V. Local search heuristics for single machine scheduling with batch set-up times to minimize total weighted completion time [J]. Annals of Operations Research, 1997, 70: 261-279.

[14] 范晔,周泓. 作业排序模拟退火算法影响因素分析和一种多次淬火模拟退火算法 [J]. 系统工程理论方法应用,2003,11(1):72-76.

[15] 吴大为,陆涛栋,刘晓冰,孟永胜. 求解作业车间调度问题的并行模拟退火算法 [J]. 计算机集成制造系统,2005,11(6):847-850.

[16] Lee W C, Wu C C, Chen P. A simulated annealing approach to makespan minimization on identical parallel machines [J]. International Journal of Advanced Manufacturing Technology, 2006, 31: 328-334.