Advances in Computer Science Research (ACSR), volume 61
7th International Conference on Education, Management, Computer and Society (EMCS 2017)

Multiprocessor Scheduling Problem by Wading across Stream Algorithm

Shang Gao[1, a*] and Yong Liu[2, b]

[1] School of Computer Science and Engineering, Jiangsu University of Science and Technology, Zhenjiang 212003, China;

[2]Artificial Intelligence of Key Laboratory of Sichuan Province, Zigong 643000, China.

[a]Gao_shang@just.edu.cn [b]susergzn@163.com

Abstract. The Wading across Stream Algorithm (WSA) act a solution as a start point, then search several solutions randomly near the start point, and find the best solution of these solutions. This best solution is to take as next start point, and then search several solutions randomly near this start point, and so on. For solving the Multiprocessor Scheduling problem, two methods to selected the initial solution as start solution were given. In order to search neighborhood trial solution, four strategies are put forward.. It is proved that Wading across Stream Algorithm is a simple and effective algorithm.

Keywords: Random optimization algorithm; Multiprocessor Scheduling problem; wading across stream algorithm

解多处理机的摸石头过河算法

高尚[1]，刘勇[2]

(1.江苏科技大学计算机科学与工程学院，江苏 镇江 212003；2.人工智能四川省高校重点实验室,自贡 643000)

摘要：依据“摸石头过河”的思想，提出一种快速、高效的随机优化算法。摸石头过河算法是以一个解为起点，向该起点附近邻域随机搜索若干个解，找出这些解中的最好的一个解，以此解为下次迭代的结果，然后以此点为起点，再向附近邻域随机搜索若干个解，以此类推。解多处理机题时，采用了 2 种选取初始解方法;找邻域解时,提出 4 种策略，测试表明,摸石头过河算法是有效

关键词：随机优化算法；多处理机问题；摸石头过河算法

中图分类号： **TP391** **文献标志码**： **A**

引言

所谓多处理机调度问题[1][2][3]（Multiprocessor Scheduling Problem：MSP）是指有 n 台相同的处理机 $P_1, P_2, \cdots, P_n$，处理 m 个独立的作业 $J_1, J_2, \cdots, J_m$，以互不相关的方式工作，任何作业可以在任何处理机上运行，未完工的作业不允许中断。作业也不能拆分成更小的子作业。调度的任务是给出一种作业调度方案，使 m 个作业尽可能短的时间内由这 n 台相同的处理机完成。本文依据“摸石头过河”的思想，提出一种快速、高效的随机优化算法–摸石头过河算法，来解决离散优化问题。

1 数学模型

已知作业 J_i 需要的处理机时间为 $t_i, (i = 1,2,\cdots,m)$，若作业 i 分配到处理机 j 上处理，则令 $x_{ij} = 1$，

否则令 $x_{ij}=0$。$\sum_{i=1}^{m}x_{ij}t_i$ 表示处理机 j 完工时间，$\sum_{j=1}^{n}x_{ij}=1$ 表示作业 i 只能分配到一个处理机上，因此多处理机调度问题的数学模型为：

$$\begin{aligned}&\min \max_{1\le j\le n}\sum_{i=1}^{m}x_{ij}t_i\\ &s.t.\quad \sum_{j=1}^{n}x_{ij}=1\quad (i=1,2,\cdots,m)\\ &\qquad x_{ij}=0,1\end{aligned}\tag{1}$$

把（1）变换如下：

$$\begin{aligned}&\min v\\ &s.t.\quad \sum_{j=1}^{n}x_{ij}=1\quad (i=1,2,\cdots,m)\\ &\qquad \sum_{i=1}^{m}x_{ij}t_i\le v\quad (j=1,2,\cdots,n)\\ &\qquad x_{ij}=0,1\end{aligned}\tag{2}$$

上式实质是线性 0-1 整数规划问题，属于 $NP-$ 难题，目前没有有效的算法解此问题。目前采用贪心法[1][2]，其思路为先将作业按运行时间的长短从大到小排成非递增序，然后给空闲的处理机依次分配作业。但此方法虽然简单，但往往得不到最优解。

2 “摸石头过河”算法的思想

摸石头过河算法[4]（Wading across Stream Algorithm， WSA）的思想来源于“摸石头过河”的思想，摸到一个“石头”后，向该“石头”周围摸索其它石头，当摸到一个“石头”后，再向该“石头”周围摸索其它石头，以此类推进行搜索。摸石头过河算法思路是以一个解为起点，向该起点附近邻域随机搜索若干个解，找出这些解中的最好的一个解，以此解为第 2 次迭代的结果。然后以此点为起点，再向附近邻域随机搜索若干个解，找出这些解中的最好的一个解，以此解为第 3 次迭代的结果。后面的步骤以此类推，达到最大迭代次数或其它停止条件为止。其迭代过程如图 1 所示。

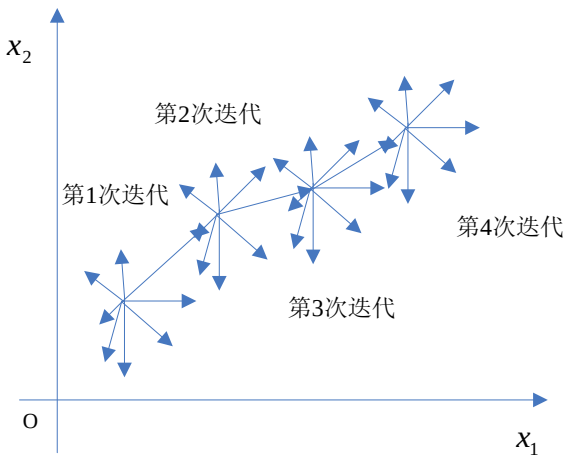

图 1 摸石头过河算法迭代过程

“摸石头过河算法”与模拟退火算法有点类似，但效果比模拟退火算法好，并且算法比模拟退火算法

简单。模拟退火算法是从 1 个解搜索下一个解，而“摸石头过河算法”从周围若干个解中摸索比较好的解作为下一个解，利用了“群”优势。模拟退火算法根据 Metropolis 准则来接受或舍弃下一个解，而“摸石头过河算法”直接选取周围若干个解中比较好的解为下一个解，操作简单。

3 解处理机调度问题的摸石头过河算法

处理机调度问题的解用矩阵 $X=(x_{ij})_{m\times n}$ 表示，$X=(x_{ij})_{m\times n}$ 满足每行和为 1。

解处理机调度问题的摸石头过河算法的框架：

步骤 1 设置算法参数：搜索解的个数 M，迭代次数 $n_{\max}$，初始一个解 X_0；

步骤 2 在 X_0 的邻域内产生 M 个解 $X_1, X_2, \cdots, X_M$；

步骤 3 计算这 M 个解的目标值 $f_1, f_2, \cdots, f_M$，找出最好 X^* 及其目标值 f^*，保留最好解，$k=0$；

步骤 4 在 X^* 的邻域内产生 M 条路径 $X'_1, X'_2, \cdots, X'_M$；

步骤 5 计算这 M 个解的长度 $f'_1, f'_2, \cdots, f'_M$，找出最好 X^* 及其目标值 f^*，保留最好解，$k=k+1$；

步骤 6 若 $k>n_{\max}$ 算法则结束，输出保留的最好解；否则执行步骤 4。

该摸石头过河算法的时间复杂性估算如下：以计算目标的操作花时间最多，所以时间复杂性大约为 $O(M\cdot n_{\max})$。

“摸石头过河”时，总要在岸边考察一下，慎重选择初始起点。同样道理，初始选择的解对整个算法会产生影响。选择初始解有很多方法：

方法 1 随机产生一个解作为初始解。

方法 2 采用随机初始解，方法是先产生 N 随机个解，从中找出最好解作为起点解。

摸石头过河算法要从邻域中随机产生另一个解，对于多处理机调度问题，它的邻域是指解矩阵除局部有差别外，大多数数据相同。在上面算法中，在 C_0 的邻域内产生另一条解 C_1，方法较多，这里给出 4 种简单的策略。

策略 1：对于解矩阵，随机挑选一行，将 1 改为 0，再随机的挑选某一列，改其为 1；

假如解 C_0 为 $\begin{bmatrix}1&0&0&0&1&0&0&0&1\\0&1&0&1&0&1&0&1&0\\0&0&1&0&0&0&1&0&0\end{bmatrix}^T$，随机挑选一，如第 2 行，在随机挑选 1 列，如第 3 列，其领域解为 $\begin{bmatrix}1&0&0&0&1&0&0&0&1\\0&0&0&1&0&1&0&1&0\\0&1&1&0&0&0&1&0&0\end{bmatrix}^T$；

策略 2：对于解矩阵，随机挑选两行，互换这两行数据；

假设解 C_0 为 $\begin{bmatrix}1&0&0&0&1&0&0&0&1\\0&1&0&1&0&1&0&1&0\\0&0&1&0&0&0&1&0&0\end{bmatrix}^T$，随机挑选两行，如第 3 和第 8 行，其领域解为 $\begin{bmatrix}1&0&0&0&1&0&0&0&1\\0&1&1&1&0&1&0&0&0\\0&0&0&0&0&0&1&1&0\end{bmatrix}^T$。

策略 3：计算各处理机的完工时间，把最长完工时间的处理机的任务进行调整，随机将其中的一个任务调整给完工时间最少的处理机。

假设有 3 台处理机和 9 个作业[1]，作业需要的运行时间分别为 81，40，26，4，65，98，53，71，15。解 C_0 为 $\begin{bmatrix}1&0&0&0&1&0&0&0&1\\0&1&0&1&0&1&0&1&0\\0&0&1&0&0&0&1&0&0\end{bmatrix}^T$，其调度结果为 P_1（81，65，15），P_2（40，4，98，71）和 P_3（26，53），完工时间为 213。P_2 是最长完工时间 213，P_3 是最少完工时间，将 P_2 随机一个任务给 P_3，如将 P_2 完工时间为 40 给 P_3，此时完工时间为 183。

策略 4：计算各处理机的完工时间，将最长完工时间的处理机和最快完工的处理机的任务交叉调整，从 1 到 n 中随机挑一个数，如为 k，那么将 $X(1:k,i\max)$ $X(1:k,i\min)$ 交换数据，$i\max$ 为最长完工时间的处理机所在的列，$i\min$ 为最快完工时间的处理机所在的列。

上面的例子中，解 C_0 为 $\begin{bmatrix}1&0&0&0&1&0&0&0&1\\0&1&0&1&0&1&0&1&0\\0&0&1&0&0&0&1&0&0\end{bmatrix}^T$，其调度结果为 P_1（81，65，15），P_2（40，4，98，71）和 P_3（26，53），完工时间为 213。P_2 是最长完工时间 213，P_3 是最少完工时间，将 P_2 与 P_3 的任务作调整，如随机产生 $k=4$，交换的结果为 $\begin{bmatrix}1&0&0&0&1&0&0&0&1\\0&0&1&0&0&1&0&1&0\\0&1&0&1&0&0&1&0&0\end{bmatrix}^T$，此时的调度为 P_1（81，65，15），P_2（26，98，71）和 P_3（40，4，53），此时完工时间为 195。

4 算法测试

仍然以上例为测试对象，设有 3 台处理机和 9 个作业，作业需要的运行时间分别为 81，40，26，4，65，98，53，71，15。为说明算法的性能，这里将与贪心法、模拟退火算法、粒子群群优化算法作对比。

利用贪心法解决该问题，其调度结果为 P_1（98，40，4），P_2（81，53，26）和 P_3（71，65，15），完工时间为 160。

模拟退火算法的参数如下[5-7]：起始温度 $T=20000$，终止温度 $T_0=1$，退火速度 $\alpha=0.95$；蚁群

算法参数如下[8]：$\rho = 0.8$，$Q = 100$；粒子群算法参数设置如下[9][10]：粒子数$n_p = 30$，最大迭代次数$n_{\max} = 50$；摸石头过河算法的参数如下，$M = 30$，$n_{\max} = 100$，$N = 100$。算法各测试 100 次，统计数据如表 1 所示。

表 1 几种算法测试结果

算法	测试问题		
	最好解	平均值	最差解
贪心算法	160	160	160
模拟退火算法	151	165.2	177
蚁群算法	151	163.6	167
粒子群优化算法	151	151.56	155
摸石头过河算法（方法 1+策略 1）	151	161.0	199
摸石头过河算法（方法 1+策略 2）	151	169.2	177
摸石头过河算法（方法 1+策略 3）	151	165.4	176
摸石头过河算法（方法 1+策略 4）	151	160.4	175
摸石头过河算法（方法 2+策略 1）	151	156.4	169
摸石头过河算法（方法 2+策略 2）	151	154.4	162
摸石头过河算法（方法 2+策略 3）	151	153.5	161
摸石头过河算法（方法 2+策略 4）	151	153.2	161

从表 1 可以看出，摸石头过河算法的性能均比较理想，特别采用方法 2 的摸石头过河算法的效果更令人满意。

5 结束语

根据“摸石头过河”的思想，提出一种摸石头过河算法来求多处理机问题，该算法易于设计和实现，设置的参数也少。测试结果表明，该算法具有搜索速度快、精度高和不易陷入局部极值点的特点，因而具有较好的全局搜索能力，其应用前景非常广泛。该方法是有一定潜力，值得推荐。

6 致谢

论文得到人工智能四川省重点实验室开放基金(2016RYJ03)支持。

Acknowledgement

This work was supported by t Artificial Intelligence of Key Laboratory of Sichuan Province (2016RYJ03).

参考文献：

[1] 王晓东.计算机算法设计与分析[M].北京：电子工业出版社，2001：102-104

[2] Herbert S. Wilf. Algorithms and Complexity. New York ：Addison Wesley publishing company, INC, 1994:30-66.

[3] D. E. Knuth. The art of computer programming, Vol. 3: Sorting and searching. New York ： Addison Wesley publishing company, INC, 1973:1-10.

[4] Shang Gao, Hualong Yu,Ling Qiu, Cungen Cao. The wading across stream algorithm. international Journal of Computers and Applications, 2014,36(4):127-132.

[5]高国华，沈林成，常文森. 求解 TSP 问题的空间锐化模拟退火算法[J].自动化学报，1999,25(3)：425-428.

[6]Kirkpatrick S， Gelatt Jr C D， Vecchi Jr M P. Optimization by simulated annealing[J]. Science， 1983，220:671-680.

[7]康立山，谢云，尤矢勇等.模拟退火算法[M]].科学出版社，1994：150-151.

[8]高尚,钟娟 ,莫述军. 多处理机调度问题的蚁群算法[J]. 微型电脑应用,2003,19(4): 9-10,16.

[9]高尚,杨静宇. 多处理机调度问题的粒子群优化算法[J]. 计算机工程与应用,2005,41(27)：72-73，104.

[10]高尚，杨静宇.群智能算法及其应用[M]. 北京：中国水利水电出版社,2006.

References

[1] Wang Xiaodong. Algorithms Design and Analysis [M]. Beijing: Publishing House of electronics industry, 2001:102-104.

[2] Herbert S. Wilf. Algorithms and Complexity[M]. New York: Addison Wesley publishing company, INC, 1994:30-66.

[3] D. E. Knuth. The art of computer programming, Vol. 3: Sorting and searching[M]. New York: Addison Wesley publishing company, INC, 1973:1-10.

[4] Gao Shang, Yu Hua-long, Qiu Ling, Cao Cun-gen: “The wading across stream algorithm”[J], International Journal of Computers and Applications, Vol.36(2014),No.4,p.127-132.

[5] Gao Guo-hua Shen Lin-cheng Chang Wen-sen:“ Using Simulated Annealing Algorithm With Search Space Sharpening To Solve Traveling Salesman Problem”[J], Acta Automatica Sinica, Vil.25(1999),No.3,p.425-428. (in Chinese)

[6] Kirkpatrick S, Gelatt Jr C D, Vecchi Jr M P. Optimization by simulated annealing[J]. Science, Vol.220(1983),pp.671-680.

[7] Kang Lishan, Xie Yun, et al. Simulated annealing algorithm [M]]. Science Press, 1994:150-151. (in Chinese)

[8] Gao Shang, Zhong juan Mu Shu-jun: “An Ant Colony Algorithm for the Multiprocessor Scheduling Problem”[J], Microcomputer Applications, Vol.19(2003)p. 9-10,16. (in Chinese)

[9] Gao Shang, Yang Jing-yu: “Solving Multiprocessor Scheduling Problem by Particle Swarm Optimization Algorithm”[J], Computer Engineering and Applications, Vol.41(2005), No. 27, p.72-73. (in Chinese)

[10] Gao Shang, Yang Jing-yu. Swarm intelligence algorithm and its application [M]. Beijing: China Water and Power Press, 2006(in Chinese)