Computer Science and Application 计算机科学与应用, 2015, 5, 62-73
Published Online March 2015 in Hans. http://www.hanspub.org/journal/csa
http://dx.doi.org/10.12677/csa.2015.53009

Hans汉斯

Multi-Objective Scheduling of Jobs on Parallel Batch Machines with Pareto Algorithm

Chao Wang, Zhaohong Jia, Hao Song

School of Computer Science and Technology, Anhui University, Hefei Anhui
Email: wangchao6439@foxmail.com

Received: Apr. 5th, 2015; accepted: Apr. 22nd, 2015; published: Apr. 27th, 2015

Abstract

In this paper, the batch scheduling problem is extended to the multi-objective (sumCj, MOC) batch scheduling problem. The scheduling problem is divided into two stages: batching and batch scheduling. In the batching process using the traditional BFLPT batch rule to obtain the batching results; while in the batch scheduling process, for the multi-objective function, this paper not only presents improved evolutionary algorithm Improved-NSGA-II to solve the multi-objectives minimization problem, but also lists the algorithms of NSGA-II and SPEA2 as the contrast. Through the simulation experiment, to compare the three algorithms in three aspects, respectively from the number, the quality and the running time of Pareto solution set, this paper proves the effectiveness of the Improved-NAGS-II algorithm.

Keywords

Ulti-Objective, Batch Scheduling, Pareto Solution Set, MOC, $\sum Cj$

使用帕累托方法解决多目标平行机批调度问题

王　超，贾兆红，宋　浩

安徽大学计算机科学与技术学院，安徽　合肥
Email: wangchao6439@foxmail.com

收稿日期：2015年4月5日；录用日期：2015年4月22日；发布日期：2015年4月27日

摘　要

本文将批调度问题扩展到针对多目标($\sum Cj$，MOC)的批调度问题，这一调度问题分为两个阶段：分批和批调度。分批过程使用的是传统的BFLPT分批规则，得到分批结果；而批调度过程中，针对多个目标函数，本文提出了改进型进化算法Improved-NSGA-II来完成多目标的极化问题，同时列举了算法NSGA-II和SPEA2作为对比。通过仿真实验，分别从帕累托解集的数量、质量和算法运行时间三个方面对三种算法进行比较，从而证明算法Improved-NAGS-II的有效性。

关键词

多目标，批调度，帕累托解集，MOC，$\sum Cj$

1. 引言

批处理机(Batch processing machine, BPM)调度问题兴起于二十世纪九十年代初，是从半导体生产过程的最后阶段中提炼出来的一种新型调度问题，具有广泛的应用背景(Uzsoy, 1994 [1]; Dupont and Ghazvini, 1998 [2]; Melouk et al., 2004 [3])。平行机批处理是一个非常传统的研究分支，也是调度问题的一个重要分支。于经典调度问题不同的是，一个机器加工多个工件。因为这类问题不仅要把工件按要求分配到机器上，还要把工件按照不同的规则分批。这类问题要比经典调度复杂的多，已知的批调度问题都是 NP 难问题。从现有的文献调查平行机批调度问题去解决的目标有最小化最大完工时间、总加工时间、总成本等等。找到正确的工件分配序列来满足上面的多个目标问题的约束仍是个具有挑战的任务。

目前有大量的研究人员对上述的目标问题进行研究，发现的各种确定性和不确定性类型方法来解决上述问题。如线性规划、遗传算法[4]、动态规划、整数编程、蚁群优化[5]、微粒群(Patrical Swarm Optimization, PSO)算法[6]、模拟退火等等。

现在关于批调度的研究通常集中在生产效率和完工时间相关的目标上，此类目标倾向于找到较早或按时完工的方案，总体上与客户满意度相关。而在另一方面，制造商所关注的利润成本上的研究较少。在当前的绿色背景下，能源所占的比重较增大，成本更应成为评价调度的重要指标。

在考虑到成本目标的双目标平行机批调度比传统的单目标批调度更为复杂的组合优化问题，本文在此基础上提出双目标($\sum Cj$，MOC)。其中 $\sum Cj$ 为工件的总加工时间，MOC 是每台机器上工件加工时所产生的成本。工件加工分为两个阶段：分批，批调度。本文的重点集中在批调度中，采用两种最新的多目标进化算法进行求解最优帕累托解集。并在此基础上提出一种新的进化算法 Improved-NSGA-II。

本文的第二章对问题进行详细描述，介绍了问题的一些概念和对应的数学模型；第三章介绍三种进化算法(NSGA-II、SPEA2 及 Improved-NSGA-II 算法)；第四章设计算例，进行实验仿真，对比各算法的实验结果，分析得出结论；文章的最后说明了接下可以继续研究的方向。

2. 问题描述

2.1. 问题模型

本文研究的多目标调度问题可具体描述如下：

(1) 给定 m 台机器平行机的集合 M，n 个工件的集合为 N，每个工件可以分配到任一个机器上加工。

(2) 机器为批处理机，其容量为 C，单个工件尺寸不会超过机器容量且批内工件尺寸之和小于批的容量。

(3) 分批后用 B 表示批集合，b 表示批。批的加工时间为批内工件的最大的加工时间，批的尺寸为批内工件尺寸之和，批的加工成本为批内工件加工成本之和。且批在加工过程中不允许中断，也不允许加入新的工件。

(4) 优化目标位($\sum Cj$, MOC)，求出在两目标最优的解集。

2.2. 数学模型 Pm|s_j，c_j，batch| ($\sum Cj$, MOC)

$$\text{Min Z1} = \sum_{i=1}^{M}\sum_{j=1}^{k}(k-j+1)\,p_j \tag{1}$$

$$\text{Min Z2} = \max(c_1, c_2, \cdots, c_m) \tag{2}$$

$$\text{s.t.} \quad \sum_{b=1}^{k} x_{jb} = 1 \quad j = 1, \cdots, n \tag{3}$$

$$\sum_{i=1}^{m} y_{ib} = 1 \quad b = 1, \cdots, k \tag{4}$$

$$P^b \geq x_{jb}\,p_j \quad j = 1, \cdots, n; b = 1, \cdots, k \tag{5}$$

$$C^b = \sum_{j \in b} c_j \quad b = 1, \cdots, k \tag{6}$$

$$\sum_{i=1}^{n} x_{jb} s_i \leq C \tag{7}$$

$$x_{jb} = \begin{cases} 1 & \text{工件}j\text{属于批}b \\ 0 & \text{其他} \end{cases} \tag{8}$$

$$y_{ib} = \begin{cases} 1 & \text{批}b\text{属于机器}i \\ 0 & \text{其他} \end{cases} \tag{9}$$

其中(1)最小化第一个目标函数，总完工时间等于所有机器上的总完工时间。(2)最小化第二个目标函数，最大完工时间等于 m 台机器中最大的完工时间。(3)表示每个工件只能由一个机器加工。(4)表示每个批只能由一个机器加工。(5)批的加工时间等于批内工件的最大加工时间。(6)批的加工成本等于批内所有工件加工成本。(7)批内所有工件的尺寸之和小于等于批的空间。(8)和(9)约束决策变量只能取 0 或 1。

2.3. 多目标优化的基本概念

设有r个优化目标，且这些目标间互相冲突，在一个目标上优化会使另一个目标变差。优化目标表示为 $f(X) = (f_1(x), f_2(x), \cdots, f_r(x))$ 多目标优化中的最优解一般称为pareto最优解。

给定一个多目标问题 $\min f(X)$，这个问题的解空间 Ω。若 $X \in \Omega$，且不存在其他的 $X \in \Omega$ 使 $f_i(X) \geq f_j(X)(j = 1, 2, \cdots, r)$ 成立，且其中至少有一个严格的不等式成立，则X是 $\min f(X)$ 的pareto最优解。

个体间的支配关系：p和q是多目标问题 $\min f(X)$ 的两个不同解，若同时满足以下两个条件，则称p支配q，符号表示为 $p \prec q$。

(1) 对所有的优化目标，p不比q差，即为 $f_i(p) \leq f_i(q), (j = 1, 2, \cdots, r)$。

(2) 至少有一个目标，p 在该目标上的值小于 q。即为 $\exists j = 1, 2, \cdots, r$，使 $f_j(p) < f_j(q)$。

3. 算法介绍

多目标批调度问题分为两个步骤，一是分批过程，一个是批处理过程，以优化 MOC 和 makespan 为目标。这两个问题都是 NP 难问题，本文在分批中使用 BFLPT 规则，以减轻计算的复杂度。文章的侧重点在批调度过程，下面介绍本文在批调度阶段使用的几个算法的流程。

3.1. BFLPT 介绍

多目标调度中分批过程也是很重要的阶段，介于本文研究问题的特征，选择了 BFLPT (best-fit longest processing time)启发式规则，它是由 L. Dupont 等[7]已提出的，并且在后续的研究中被广泛使用，能够得到较优的分批方案，具体步骤如下：

算法 BFLPT：

(1) 加工件按加工时间 p_i 非递增排序，得到工件序列 L；

(2) 从序列 L 选择工件，放入可行批中，使该批的剩余空间最小；如果没有可以放入的批，则重新创建一个批，将工件放入进去，更新批属性。重复步骤 2 直到所有工件都分配到批中。

(3) 将批排序，随机分配到机器上加工。

这一启发式式算法，前两步完成分批操作，通常根据不同的问题修改第三步的机器选择策略，从而能得到较优的目标函数解。

3.2. NSGA-II 介绍

NSGA-II (Non-dominated Sorting Genetic Algorithm II) [8]是 Deb et al. [9]在 2012 年提出的基于多目标的进化算法，相对于此前版本 NGSA [10] (Srinivas and Deb, 1994)的一点缺点，主要有以下几个优点。一是它改进了 Pareto 最有解集的构造，同时降低的时间复杂度。二是加入的精英保留机制[11]，在搜索过程中更好的保留住非支配解集。三是解决多目标问题的参数共享问题。

3.2.1. 编码与解码

每个染色体都代表所有批的序列，每个批都有原始的序列号 $1,2,\cdots,n$ 。则这个染色体的长度为 $n\times(n/2+1)$，染色体上每$(n/2+1)$位上存储一个批的序列号。在编码时将按一定初始顺序将每个批的序列号转为$(n/2+1)$位的二进制码。在解码时进行反转即可。

3.2.2. 算法的基本流程

首选随机产生一个新的初始种群 A。然后对其进行竞标赛选择，变异，交叉操作后形成一个新的种群 B。将种群 A 和种群 B 合并成为一个具有 $2n$ 个染色体的种群 C。

对种群 C 进行构造其边界集[12]。并在其中某个边界集上计算所有个体的聚集距离，以此来建立这个边界集上的个体偏序关系。以此为基础，从 C 中选择出 n 个染色体进入新一代种群 A。重复上述过程直到到达预定的迭代次数或指定的条件。

3.2.3. 边界集的构建(图 1)

边界集的定义：群体 C 中有 N 个解，按照某种策略分类为 m 个子集 $F_1,F_2,F_3,\cdots,F_m$，若满足下列心智，则这些子集构成群体 C 的 m 级边界集：

(1) $\bigcup_{p\in\{F_1,F_2,\cdots,F_m\}} p=C$；

(2) $\forall i,j\in\{1,2,\cdots,m\}$ 且$i\neq j, F_i\cap F_j=\varnothing$;

(3) $F_1\succ F_2\succ\cdots\succ F_m$, 即 F_{i+1} 中的个体直接受F_i中的某个个体支配。

在NSGA-II中，为了从现有的种群中选择高质量的个体进入新一代种群，需要对种群构造边界集。

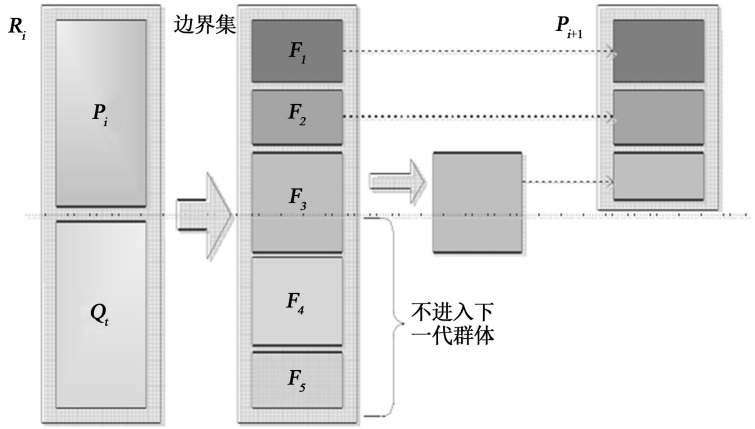

Figure 1. Construction of boundary set
图 1. 边界集构建

设对于每个个体 $i \in C$ 有两个变量s_i和t_i，其中s_i记录种群中支配个体i的个体数，t_i记录被i支配的个体集合，即

$$s_i = \left|\left\{ j \middle| j \succ i, i, j \in C \right\}\right|$$

$$t_i = \left|\left\{ j \middle| i \succ j, i, j \in C \right\}\right|$$

构造边界集时，先为每个个体计算出s_i和t_i，算法的伪代码如下，然后构造边界集。

```
//计算s_i和t_i
do while  p ∈ C
   do while  q ∈ C
         if  (p ≻ q)  {t_p = t_q ∪ {q};}
         else if  (p ≺ q) {s_p = s_p + 1;}
    end while
end while
//构造边界集
i = 1;
While  (F_i ≠ ∅)
     H = ∅;
     do while  q ∈ t_p
          s_q = s_q − 1;
if  (s_q == 0)
                H = H ∪ {q};
   End while
   i = i + 1;
   F_i = H;
End while
```

3.2.4. 聚集距离计算

设F_i代表边界集i内的染色体数目，其满足以下条件：

1) $F_1+F_2+\cdots+F_i-1<100$

2) $F_1+F_2+\cdots+F_i\geq 100$

如图上所示比F_i高的边界集内的染色体全部进入下一代，并在F_i内选择$100-\left(F_1+F_2+\cdots+F_{i-1}\right)$个染色体进入下一代，选择规则即是聚集距离(图2)。

聚集距离用于区分同一边界集上的个体好坏，公式为：

$$D(i)=\sum_{k=1}^{r}\left(f_k(i+1)-f_k(i-1)\right)$$

根据目标 k 上的值为每个个体排序，r 代表目标个数，$f_k(i+1)$ 与 $f_k(i-1)$ 代表目标 k 在与 i 相邻的两个个体的函数值。

3.3. SPEA2 介绍

SPEA2 [13]算法是在SPEA (Strength Pareto Evolutionary Algorithm) [14]基础上进行改进，解决多目标问题的帕累托算法。最初的SPEA算法，将每个个体的适应度称为Pareto强度，首先把算法每次迭代过程中产生的所有个体的值进行适应度定义，并设定一个外部的种群来保留当前进化群体中较优秀的个体。本文用此算法解决调度此调度问题，其编码与解码和NSGA-II相同。

算法SPEA2：群体P的规模为N，存档集Q为M，迭代次数为T；

(1) 随机产生一个存档集和初始种群Q_0，P_0。迭代器$t=0$；

(2) 为PT与QT中的个体计算适应度；(3.3.1)

(3) 将PT与QT里的所有非支配个体保持入下一代存档集Q_{T+1}中，若此时$\left|Q_{T+1}\right|>M$，则利用修剪去除多余的个体；若$\left|Q_{T+1}\right|<M$，则在PT和QT中按一定的规则选择一些个体加入Q_{T+1}，使$\left|Q_{T+1}\right|=M$；(3.3.2)

(4) 若$t=T$或满足其他终止条件，输出Q_{T+1}中的非支配解为算法结果；

(5) 对Q_{T+1}执行竞标赛选择，交叉和变异，把结果保持到PT+1中，$t=t+1$，转入(2)。

3.3.1. 适应度分配

为了使每个个体拥有不同的适应度值，同时考虑在迭代中的群体和外部群体的所有的个体的情况，通过计算个体与它相邻的个体间的距离来确定拥挤情况，即计算群体PT(初始集)和QT(存档集)中每个个体的适应度，本文分为两个方面讨论。总适应度$F(i)$由$R(i)$和$D(i)$综合决定：

$$F(i)=R(i)+D(i)$$

其中$R(i)$ (图3)是整数部分，$R(i)$的计算公式如下：

$$R(i)=\sum_{j\in P_t\cup Q_t,\, j\prec i}S(j)$$

其中$S(j)$为群体P_t和Q_t中j支配的个体数。$R(i)$越低代表解的质量越好，$D(i)$是小数部分，其计算公式如下：

$$D(i)=\frac{1}{\sigma_i^k+2}$$

其中 σ_i^k 是个体i到其第k个最近个体的距离，分母加2是为了是距离不为0，且 $D(i)<1$，其中k为 $k=\sqrt{|P_t|+|Q_t|}$。

对外部种群的维护，选择当前迭代种群和外部种群的非支配解集，当外部种群的数量大于我们预设的值时，删除外部种群中较差的个体。否则选取迭代中的种群中较好的个体进行补充。重复这个过程，直到外部种群的规模达到我们预先设定的值。虽然没有降低算法的复杂度，但由于考虑了每个个体的拥挤情况，可以得出的结果有着更均匀的分布。

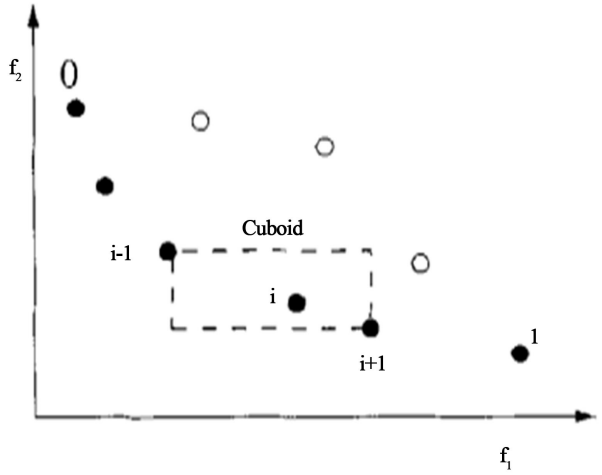

Figure 2. Gathering distance calculation
图 2. 聚集距离计算

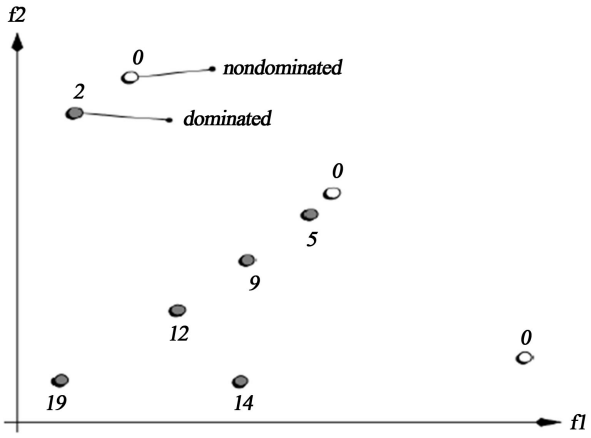

Figure 3. Fitness R_i calculation
图 3. 适应度 R_i 计算

3.3.2. 环境选择

从群体 PT 和存档集 QT 中选择合适的个体存入下一代存档集 Q_{T+1} 中，首选对 PT 和 QT 中 $F(i)<1$ 的(即非支配个体)存入 Q_{T+1}。如果 $|Q_{T+1}| \le M$，则从 PT 和 QT 中选择剩余 $F(i)$最小的加入，直到 $|Q_{t+1}| = M$。如果 $|Q_{t+1}| > M$，则利用存档修剪不断删去 Q_{t+1} 中的个直到 $|Q_{t+1}| = M$。利用 σ_i^k 发现个体间的距离，删去与选择个体距离最小的个体(图 4)。

3.4. 基于改进型 NSGA-II 的批调度算法

前文介绍过在批调度多目标算法中，我们将算法分为两个阶段。而在批调度阶段，用进化算法来求问题的解。本文在此 NSGA-II 的基础的上进行改进，在批调度的种群迭代中加入一个启发式算法，优化种群中的每个个体。

3.4.1. 改进策略

在批处理阶段，将批分批到每台机器上加工。对本文的目标问题(XQMOC)，我们可以通过分级思想，来确保第一个目标不变的情况下来优化第二个目标。在 NSGA-II 的进化算法中，对于每代种群中的个体

我们都将对其进行优化。在种群的迭代过程中，通过进化算法在每次种群迭代结束后，都能得到更优的个体，然后将更优的个体放于下一代中。

算法 Improved-NSGA-II 的流程如图 5 所示。

3.4.2. 批处理阶段的启发式算法

在批分配阶段，应用启发式算法主要考虑以下三个方面：

①种群的迭代过程中得到的各个个体的批序列；

②基于分级思想调整每级上的工件顺序，将排好序的批分配到各机器上；

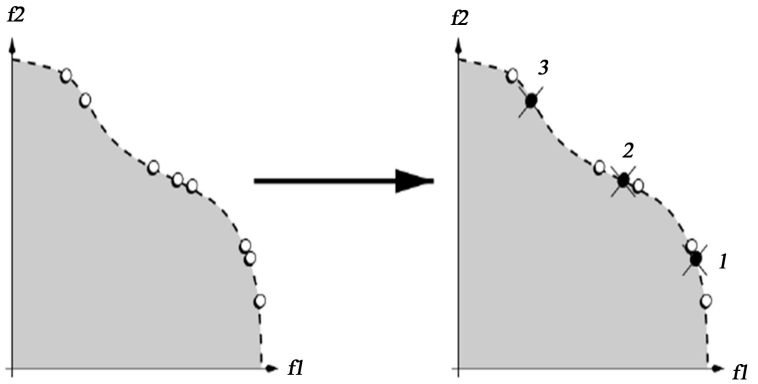

Figure 4. Trim archive collection
图 4. 存档集的修剪

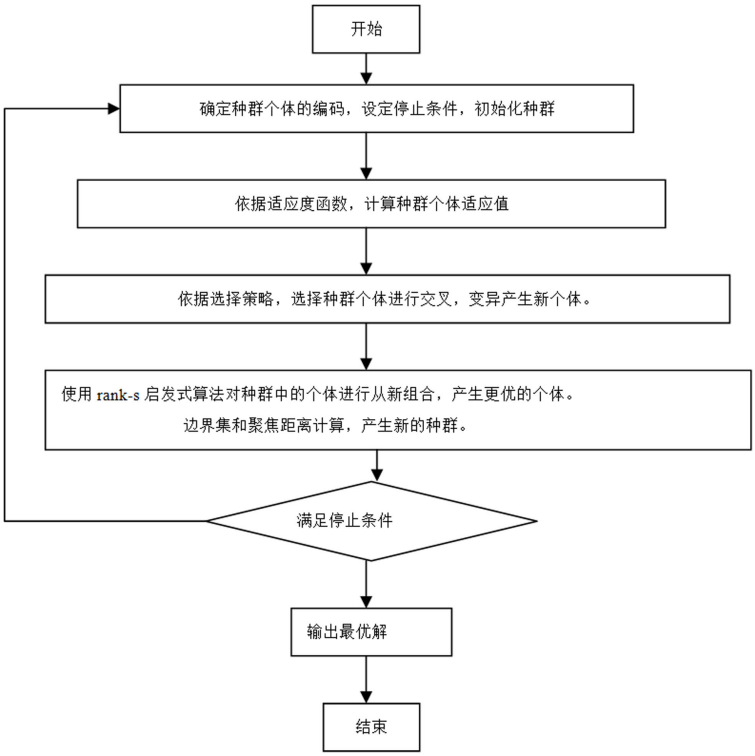

Figure 5. Algorithm process diagram
图 5. 算法流程

③我们使用启发式算法调整得到：在保证第一目标总完工时间不变的基础。

上考虑如何最小化第二目标：机器的最大加工成本 MOC。

算法步骤如下：

(1) 在经过一次交叉变异后，我们得到一个种群。解码得到每个种群个体的批序列，将批按照自定好的序安放在机器上，则批数量 $n=(R-1)m+x$；

(2) 将最后 m 个批分配到最后一级上，将倒数第 2 m 个批分配到倒数第二级上，基于帕累托方法的多目标平行机批调度问题直到分完 $(R-1)m$ 个批到 $R-1$ 级上，最后将剩下的 x 个工件分配到第一级上；

(3) 从第二级开始，将当前级上加工成本最大的批放到前一级中机器加工成本最小的机器上，执行此过程直到最后一级第 R 级为止。

从启发式算法 rank − s 对每个种群的个体进行改进，可以保证个体的第一个目标不变的情况下，达到优化第二个目标的目的。再把优化后的群体按照 NSGA-II 进行选择，对于每一代的群体都要进行优化，这样我们每代可以得到更优的个体。

4. 实验设计与结果分析

4.1. 实验设计

通过仿真实验来比较几个多目标算法。实验中算例的生成主要考虑一下几个方面：工件数量，加工时间，加工成本，机器数量(表 1)。

4.2. 算法参数设置

通过初步的实验，我们得到三种算法在以下参数的设置情况可以得到较为理想的结果参数设置参照以下列表(表 2)：

所有算法在 eclipse 平台下运行，硬件为 dual core 2.3Ghz 处理器，2G 内存。每种算法运行 10 次实例。

4.3. 算法的评价指标

为了衡量三种算法在不同情况下的表现，我们在实验中采用以下的几种指标来评价算法的性能。

(1) 非支配解集的质量，各个算法得到非支配解集的质量是一个非常直观的指标，能很清晰表明算法在某种情况下的表现。优质的帕累托解集更加接近边界的近似值。

(2) 帕累托的解集数量，即是各个算法得到的非支配解集的规模。解集的数量越多，我们也就有着越大的选择空间，从而可以更好的构造帕累托曲线，得到更优的解决方案。

(3) 算法所运行的时间，因为这类问题都是 NP 难问题，求解的时间很大程度上可以看出一个算法的好坏。

Table 1. Parameter settings of jobs and machines
表1. 工件机器参数设置表

因素	取值	不同取值数目
工件数量	50,100,150	3
机器数量	2,3,4	3
工件尺寸	[1,10]	1
加工时间	[1,10]	1
加工成本	[10,50]	1
机器容量	10	1

4.4. 实验结果与讨论

在表 3 中我们列出了几种算法在不同情况下找到的非支配解的个数。在类型列第一个数值代表工件

Table 2. Algorithm parameters
表2. 算法参数表

Improved NSGA-II	NSGA-II	SPEA2
种群规模：100 (20 工件)	种群规模：100 (20 工件)	种群规模：100 (20 工件)
种群规模：150 (50 工件)	种群规模：150 (50 工件)	存档规模：100 (20 工件)
种群规模：200 (80 工件)	种群规模：200 (80 工件)	种群规模：150 (50 工件)
		存档规模：150 (50 工件)
		种群规模：200 (80 工件)
		存档规模：200 (80 工件)
迭代次数：500	迭代次数：500	迭代次数：500
交叉概率：1.0	交叉概率：1.0	交叉概率：1.0
变异概率：0.1	变异概率：0.1	变异概率：0.1

Table 3. The compare of three algorithms' results
表3. 三种算法解数量的实验结果对比

类型	Improved NSGA-II			NSGA-II			SPEA2		
	Max	Avg	Min	Max	Avg	Min	Max	Avg	Min
20-3-小	12	8.7	5	11	8.1	4	14	9.7	5
20-3-大	10	9.1	8	12	8.3	5	13	8.6	5
20-4-小	12	11.5	10	13	11	10	12	8	6
20-4-大	13	10.4	8	11	8.8	7	13	11.5	10
20-5-小	9	8.5	7	12	10.4	8	11	8.8	7
20-5-大	9	7.5	4	13	8.2	4	10	6.6	3
50-3-小	20	18.1	16	19	17.5	15	20	16.1	14
50-3-大	33	29.4	25	32	27.9	23	30	23.8	21
50-4-小	16	13.2	11	14	11.4	9	14	10.7	6
50-4-大	37	31.7	27	30	20.8	12	26	21.7	19
50-5-小	15	13.5	10	13	11.6	10	13	11	10
50-5-大	17	14.8	12	14	11.6	10	15	11.8	9
80-3-小	28	25.3	21	26	22.3	19	25	22.5	18
80-3-大	46	30	24	39	31.4	23	30	20.6	14
80-4-小	18	15	9	16	12.9	9	17	13.2	9
80-4-大	29	25.9	22	20	15.4	9	25	18.9	15
80-5-小	33	30.9	26	32	28.4	21	31	26.3	22
80-5-大	47	38.2	32	36	33.7	27	30	28.6	25

数，第二个数值代表机器数，第三个数值代表工件尺寸。其中 MAX、AVG、MIN，在列表中示几种算法在运行过程中找到非支配解的最大值、平均值、和最小值。从数据中，我们可以看出，在工件数较小的情况下，几种算法的性能都差不多。但当随着工件的增加，在大规模的工件中 NSGA-II 相对于 SPEA2 可以找到更多的解集。而 Improved-NSGA-II 又比 NSGA-II 要有更好的效果。因为存档集的原因，SPEA2 的非支配解集是从存档集中找出的，因此我们得到的非支配解集的规模是不可能大于存档集的规模的。总体来看，找到的非支配解集的数量是随着工件规模的增加而增加的，因为解空间的增加会找到更多符合帕累托曲线的解集。

表 4 中列出了三种算法的运行时间，单位为秒(s)。从表中我们可以看到，但工件数位 20 时，三种算法的运行时间并没有太大的区别。但随着问题规模的增大，Improved-NSGA-II 的运行时间相较与 NSGA-II 和 SPEA2 有着明显的增大。这是因为，我们在 NGSA-II 中加入了 mnk-s 的分级算法，此算法在随着层次的增加，加工时间会曾显指数级增长。NSGA-II 和 SPEA2 只需通过交叉变异来选出优秀个体。这两则间，SPEA2 在所有的算例的时间上都要长于 NSGA-II，这是因为 SPEA2 的复杂度是由存档集 M 和种群 N 共同决定的，而 NSGA-II 的时间开销只是由种群的规模 N 决定的。这是 NSGA-II 要更快点。

5. 总结

本文从该问题的模型特征入手，针对调度过程中的两个阶段分批阶段和批调度阶段。使用了两种进

Table 4. The compare of three algorithms' running time
表4. 三种算法运行时间的实验结果对比

类型	Improved NSGA-II			NSGA-II			SPEA2		
	Max	Avg	Min	Max	Avg	Min	Max	Avg	Min
20-3-小	10.98	10.66	10.44	11.69	11.21	11.03	25.68	24.73	23.95
20-3-大	16.70	17.35	16.13	17.88	17.29	16.95	28.55	26.21	25.73
20-4-小	20.19	17.84	14.86	20.72	18.65	17.78	28.06	25.75	24.48
20-4-大	24.22	23.42	23.05	23.58	23.15	22.48	27.67	26.69	25.17
20-5-小	25.22	24.42	24.05	24.58	24.15	23.48	29.86	28.73	28.09
20-5-大	27.59	26.95	25.87	26.35	25.87	25.05	30.11	29..36	28.85
50-3-小	42.78	42.13	41.88	41.15	40.97	40.43	60.32	58.64	56.27
50-3-大	47.73	45.28	43.17	45.84	43.66	42.53	65.93	64.81	63.15
50-4-小	57.63	55.82	52.26	54.33	52.13	51.84	74.23	72.59	71.94
50-4-大	62.49	60.31	58.79	60.54	58.72	56.86	80.98	75.66	71.69
50-5-小	70.05	68.58	62.95	65.15	63.13	60.81	85.55	81.74	76.36
50-5-大	75.31	72.22	69.81	72.94	70.42	67.41	93.97	90.41	88.39
80-3-小	358.08	351.70	346.35	120.19	114.84	100.38	147.81	146.52	145.81
80-3-大	367.16	348.27	335.91	142.24	140.77	136.11	190.19	188.07	186.03
80-4-小	472.24	447.08	429.64	189.14	173.38	162.67	220.79	218.98	217.69
80-4-大	592.11	573.27	561.41	203.43	198.29	196.05	283.98	267.17	259.59
80-5-小	640.34	623.82	612.91	274.48	273.66	272.73	327.47	324.66	321.93
80-5-大	882.92	875.38	857.13	317.14	315.09	313.77	460.14	451.56	449.41

化算法 NSGA-II 和 SPEA2，并在此基础上提出了 Improved-NSGA-II 算法。这三个算法主要用的批调度阶段，我们从解的质量，解的数量和算法运行时间三个方面进行了比较。实验结果表明在工件集较小的时候三个算法的加工时间基本相同，而 Midified-NSGA-II 有较优的解。而随着工件数量的增加，Midified-NSGA-II 的运算时间较其他两个进化算法也会快速增加。本文得出的结论，Midified-NSGA-II 在工件集较小，机器台数较多时，获得的非支配解集优于 SPEA2 和 NSGA-II。而 SPEA2 和 NSGA-II 在工件规模大的时候有着更优的运行时间。

对于未来的研究，可以从这几个方面进行：在工件加工时间，加工成本和工件尺寸三个维度上进行进一步的研究，使分布更加均匀，来提高解集的质量；本文提出 Midified-NSGA-II 算法在面对工件集数量较多时，运行时间方面的效果并不是很好，还有着较大的改进空间；对于工件分级思想上可以找到新的方法来降低时间复杂度，这也是接下来工作的主要方向。

参考文献 (References)

[1] Uzsoy, R. (1994) Scheduling a single batch processing machine with nonidentical job sizes. *International Journal of Production Research*, **32**, 1615-1635.

[2] Dupont, L. and Ghazvini, F.J. (1998) Minimizing makespan on a single batch processing machine with non-identical job sizes. *European Journal of Automation*, **32**, 431-440.

[3] Melouk, S., Damodaran, P. and Chang, P.Y. (2004) Minimizing makespan for single machine batch processing with non-identical job sizes using simulated annealing. *International Journal of Production Economics*, **87**, 141-147.

[4] 席裕庚, 柴天佑, 恽为民 (1996) 遗传算法综述. *控制理论与应用*, **06**, 697-708.

[5] 刘彦鹏 (2007) 蚁群优化算法的理论研究及其应用. 浙江大学, 杭州.

[6] 杨维, 李歧强 (2004) 粒子群优化算法综述. *中国工程科学*, **05**, 87-94.

[7] Dupont, L. and Dhaenens-Flipo, C. (2002) Minimizing the makespan on a batch machine with non-identical job sizes: An exact procedure. *Computers & Operations Research*, **29**, 807-819.

[8] Deb, K., Pratap, A., Agarwal, S., et al. (2002) A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, **6**, 182-197.

[9] Deb, K., Agrawal, S., Pratap, A. and Meyarivan, C. (2000) A fast elitist non-dominated sorting genetic algorithm for multiobjective optimization: NSGA-II. *Proceedings of the Parallel Problem Solving from Nature VI Conference*, 849-858.

[10] Deb, K. and Srinivas, N. (1994) Multiobjective optimization using nondominated sorting in genetic algorithms. *Evolutionary Computation*, 221-248.

[11] Zitzle, E. and Thiele, L. (1999) Multiobjective evolutionary algorithms: A comparative case study and the Strength Pareto approach. *IEEE Transactions on Evolutionary Computation*, **3**, 257-271.

[12] 舒锋 (2011) SAGA 算法在差异工件平行机批调度问题中的应用研究. 中国科学技术大学, 合肥.

[13] Zitzler, E., Laumams, M. and Thiele, L. (2001) SPEA2: Improving the strength Pareto evolutionary algorithm: Evolutionary Methods for Desigri. *Optimization and Control with Applications to Industrial Problems*, Athens, 95-100.

[14] 杜冰 (2011) 批处理机调度问题的模型与优化方法研究. 中国科学技术大学, 合肥.