学校代码：10200　　学号：1272410045

东北师范大学

本科毕业论文

基于带变邻域搜索的自适应模拟退火算法解决零空闲流水线调度优化问题

An adaptive simulated annealing algorithm with variable neighborhood search for optimization no-idle flow shop scheduling problem

作者：**王云鹤**

指导教师：**殷明浩　副教授**

学科专业：**计算机科学与技术**

学　　院：**计算机科学与信息技术学院**

东北师范大学学位评定委员会

2014 年 5 月

摘　　要

零空闲流水线调度问题（no-idle flow shop scheduling problem，NFSP）是一个有广阔应用背景的 NP 难度问题。本篇论文提出了一个高效率的带变邻域搜索的自适应模拟退火算法（adaptive simulated annealing algorithm with variable neighborhood search，ASAvns）解决零空闲流水线调度优化问题，即目标是让工作调度序列最大完成时间（makespan）最小化。第一，该论文提出一个以自适应模拟退火为算法基本框架的有效局部搜索算法，叫做变邻域搜索（variable neighborhood search，VNS）；其次，为了提高收敛速度，在算法中使用两个搜索阶段进行局部搜索；最后，算法所需要的参数，如局部搜索循环次数，温度冷却系数，强制接受新解的条件，都要根据具体 NFSP 规模进行调节。实验结果中用 12 组不同规模的 NFSP 测试样例将该算法和其它构造启发式算法如 NEH 等、元启发式算法如 PSOvns 等相比，证明该算法求解的高效性和鲁棒性。目前，用该算法求解出的相应问题结果在零空闲流水线调度领域和工程领域等领域内效果很好。

关键词： 自适应；模拟退火；变邻域搜索；零空闲流水线调度；最大完成时间

Abstract

The no-idle flow shop scheduling problem (no-idle flow shop scheduling problem, NFSP) is an NP-hard problem with an extensive application background. In this paper, we present an adaptive simulated annealing algorithm with variable neighborhood search (adaptive simulated annealing algorithm with variable neighborhood search, ASAvns) which is an effective local search algorithm to solve this problem with the objective of minimize the maximum completion time (makespan) of the scheduling sequences. First, the proposed algorithm based on the standard adaptive simulated annealing algorithm with a very efficient local search which we call it the variable neighborhood search (variable neighborhood search, VNS). Second, two search phases are used during its search, for the sake of speeding up the convergence rate of it. Third, parameters like the times of repetition of the local search, the cool coefficient of the temperature and the condition of compulsively accepting a new solution are adaptive according to the scale of the specific NFSP examples. The results of simulation experiments that are the 12 different size NFSP benchmarks compared with other constructive heuristic algorithms such as NEH and meta-heuristic algorithms such like PSOvns proved the effectiveness, the high search quality and the robustness of ASAvns. The present value of the new algorithm is very fine in the no-idle flow shop scheduling region, engineering field and so on.

Key words: Adaptive; Simulated annealing; Variable neighborhood search; No-idle flow shop; Makespan

目　　录

第一章 绪论

1.1 研究背景

生产调度在实现先进制造业和提高生产效率中扮演一个基本和关键的角色，涉及到很多学科及领域，例如：数学、控制工程、计算机工程和信号处理领域等。其中车间工作调度问题是最典型和最具难度的组合优化问题之一。根据不同的约束条件，车间工作调度问题包括零空闲流水线调度、无等待流水线调度问题等。在现实生产调度中，有的时候，一旦流水线生产开始，机器的运转是不允许停止的，这就是零空闲流水线调度问题。如在生产化学肥料过程中，一些大型设备的开启和停止都需要耗费大量的财力，因此希望设备在运行时是零空闲，无停歇的；再例如在生产玻璃纤维的过程中，需要使用成型装置拉丝成型，一旦成型装置停止工作，玻璃纤维的生产将功亏一篑。

Baraz 和 Mosheiov 已经证明 3 个机器的零空闲流水线调度问题是 NP 难度问题[1],精准算法如分支限界法只适合解决小规模问题。为了解决大规模问题，研究者们提出了各种各样的算法如构造启发式算法，元启发式算法等。

构造启发式算法如 NEH，KK，改进贪婪方法 IG 等，虽然可以快速构建一个问题的解，在时间上有高效性，但是解的质量不太令人满意。随着智能优化算法的发展和计算机科学的进步，又有更多的学者提出了元启发式算法如差分进化算法 DE，踏步搜索 TS，基因算法 GA，模拟退火算法 SA 等提高解的质量。

1.2 论文的主要工作

论文提出的算法是带变邻域搜索的自适应模拟退火算法，它是基于模拟退火算法提出的一个新算法，我们使用 C 语言实现新算法，并使用它解决零空闲流水线调度问题，让最大完成时间最小化。

1. 理论基础

1953 年，N. Metropolis 等人提出了最先的模拟退火算法（Simulated Annealing，SA）思想。1983 年，S. Kirkpatrick [1]等人在组合优化问题中成功地引入了模拟退火思想。它从某一较高的初始温度开始，伴同冷却系数，温度将继

续下降，在一固定时间段内，联合概率突跳特征，随机搜索目标函数的全局最优解，其中搜索范围是一个大的搜索空间。模拟退火算法是一种具有通用性的优化算法，理论上该算法具有依概率收敛到全局最优解的机能。近几年，该算法广泛应用到各种各样不同的组合优化问题中，例如图着色问题，经典的旅行商问题等。它有很强的通用性，同时需要的内存空间较少。

只要初始温度充分大，降温过程较缓慢，该算法就可以收敛找到一个全局最优解。然而，与其他元启发式算法相比，它需要跟多的计算时间，而且有时候易陷入局部最优，因为在实际情况中，很难达到严格的理论上的收敛条件。

2. 模拟退火算法的基本思想

算法的基本思想步骤详见图 1.1

它包含解空间、初始解和目标函数三部分。以目标最小化问题为例：

（1） 初始化初始温度 T（足够大），初始解状态 S 也是算法每次迭代的起始点，每个 T 值的迭代次数 time；

（2）对 k=1，……，time 做第（3）到第（6）步；

（3）产生新解 S^1（使用局部搜索算法）；

（4）计算增量$\Delta f=F(S^1)-F(S)$，其中 F（S）为目标函数的计算结果，即为适应度值；

（5） 若$\Delta f<0$，表示新解较好，则接受 S^1 作为新的当前解，否则以一定概率接受 S^1 作为新的当前解；

（6） 当满足终止条件时，输出当前解作为最优解，结束此程序；

终止条件通常取为温度下降至某一较低温度或者连续若干个新解都没有被接受。

（7）T 逐渐减少，若 T^1（下降后温度）>结束温度，则转向第 2 步。

图 1.1 模拟退火算法的基本思想

3. 新算法实现

为了获得具体问题（NFSP）更好的解，在该论文中提出一种带变邻域搜索的自适应模拟退火算法（ASAvns）解决零空闲流水线调度问题。首先该算法使用的参数是根据具体 NFSP 例子大小及算法进行的程度来调节的，这就是所谓的“自适用”。其次，该算法提出两阶段局部搜索的策略，提高解的收敛速度以获得一个全局最优解。局部搜索算法采用的是高效率的变邻域局部搜索方法。最后用 C 语言实现该算法，经结果测试，ASAvns 在 12 组不同规模 NFSP 测试样例中会产生很多比其它算法适应度高，即最大完成时间比其它算法小的解。

1.3 论文安排

该论文提出一种带变邻域搜索的自适应模拟退火算法，每章的内容安排如下：

第一章，介绍生产调度问题的研究背景，及本文的主要工作和论文内容安排。

第二章，描述零空闲流水线调度问题（NFSP），并讲解该问题的相关概念，最后介绍一种最大完成时间（makespan）的计算方法。

第三章，说明带变邻域搜索的自适应模拟退火算法（ASAvns）及相应的参数设置，介绍变邻域搜索算法框架。

第四章，展示用 ASAvns 解决 12 个 NFSP 测试样例的结果及用该算法和其它算法解决 12 组不同规模测试样例的对比结果，并对实验结果进行分析。

论文最后，总结全文并说明下一步的研究方向。

第二章 零空闲流水线调度问题

零空闲流水线调度问题是一个 NP 难度难题，属于 NP 完全问题。

2.1 问题描述及基本概念

根据名字特点，这个调度问题可以描述为：[3]在一个生产系统中，有 n 个工作和 m 个机器，所有的工作在所有的机器上的处理顺序是相同的。假设每个工作的准备时间为 0 或者包含在处理时间中，要求每一个工作在某一时刻只能在一个机器上处理；一台机器某一刻只能处理一个工作；机器上的工作一旦开始，那么机器的运转是不可以停止的。这就是所谓的零空闲，即每个机器上的操作处理工作阶段是连续的，不可以停止。下面介绍算法中用到的一些重要变量的含义及表示方法：

变量 1：一个完整工作调度序列 $\pi=\{\pi_1, \pi_2, ..., \pi_n\}=\pi^{(p)}(n)$

变量 2：一个部分工作调度序列 $\pi^{(p)}(i)=\{\pi_1, \pi_2, ..., \pi_i\}$

变量 3：受零空闲约束条件的限制，对于某一个工作调度序列$\pi^{(p)}(i)$,它最后一个工作在机器 J+1 和机器 J 上处理完成的最小时间差:$E(\pi^{(p)}(i), j, j+1)$

变量 4：工作 i 在机器 j 上的处理时间：$t_{i,j}$ $(1\leq i\leq n, 1\leq j\leq m)$

变量 5：可行解π的最大完成时间：$C_{max}(\pi)$

变量 6：所有可行解即序列的集合：$\prod$

变量 7：最优工作调度序列即有最小适应度的解：π^*

了解这些变量后，通过下一小节给出的相应公式使用它们，就可以计算出makespan也就是最大完成时间。其中$\prod$为模拟退火算法中提到的解空间，目标函数为makespan，适应度值是makespan的大小。本问题是最小化最大完成时间，所以适应度值越小表示该解越优。

2.2 最大完成时间的计算方法

目标函数 makespan 即最大完成时间 $C_{max}(\pi)$，可以使用以下提出的几个公式计算。

公式（2.1）：$E(\pi^{(p)}(1),j,j+1)=t_{j_1,j+1}$, j=1,2,…m

公式（2.2）：$E(\pi^{(p)}(2),j,j+1)=\max\{ E(\pi^{(p)}(1),j,j+1)-t_{j_2,j},0\}+t_{j_2,j+1}$,

j=1,2,…m-1

公式（2.3）：$E(\pi^{(p)}(i),j,j+1)=\max\{ E(\pi^{(p)}(i-1),j,j+1)-t_{j_i,j},0\}+t_{j_i,j+1}$,

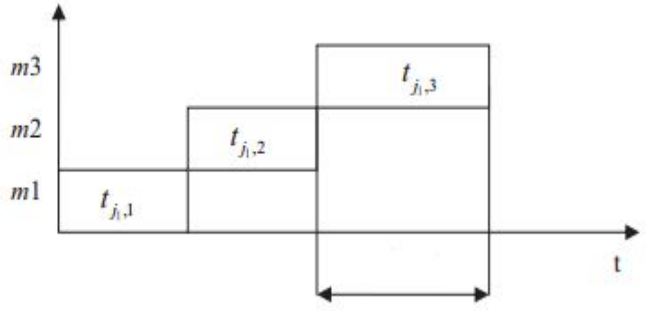
i=2,3,…n;j=1,2,…m-1.

公式（2.4）：$C\max(\pi)=\sum_{j=1}^{m-1}E(\pi,j,j+1)+\sum_{i=1}^{n}t_{j_i,1}$

公式（2.5）：$\pi^*=\min(C_{max}(\pi)),\forall\pi\in\prod$

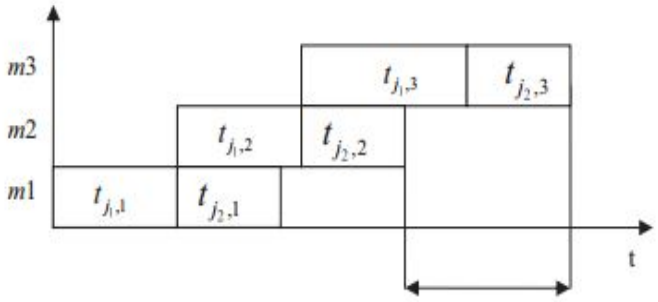
其中公式（2.4）表示零空闲流水线调度问题的目标函数，公式（2.5）表示最优解条件。

其中 n 为工作的个数，m 是机器的个数，其它变量为上一小节提出的变量。下面用图 2.1，2.2，2.3 形象展示一下当 n=2，m=3，工作调度序列为：$\pi=\{\pi_1,\pi_2\}$时，怎样计算出零空闲流水线调度问题在不同条件下的 $E(\pi^{(p)}(1),2,3)$，$E(\pi^{(p)}(2),2,3)$的大小。



$E(\pi^{(p)}(1),2,3)$

图 2.1 计算出 $E(\pi^{(p)}(1),2,3)$



$E(\pi^{(p)}(2),2,3)$

图 2.2 当 $E(\pi^{(p)}(1),2,3) > t_{j_2,2}$ 时，计算出 $E(\pi^{(p)}(2),2,3)$

$E(\pi^{(p)}(2),2,3)$

图 2.3 当 $E(\pi^{(p)}(1),2,3) < t_{j_2,2}$ 时，计算出 $E(\pi^{(p)}(2),2,3)$

其中，最大完成时间用公式求解为：

$makespan=t_{j1,1}+t_{j2,1}+E(\pi^{(p)}(2),1,2)+E(\pi^{(p)}(2),2,3)$。

第三章 带变邻域搜索的自适应模拟退火算法

最基本的模拟退火算法有一个随着温度降低的基础循环过程，如第一章中介绍的模拟退火算法的基本思想中提到的循环过程一样，首先，使用某种局部搜索算法随机产生一个问题解，即工作调度；其次，需要计算出 NFSP 的适应度值即 makespan 的大小，判断解的好坏，最后，决定接受该解还是舍弃这个解。

这个循环将一直进行，直到温度降低至结束温度或者到达停止条件。这种有效的模拟退火算法曾适用于解决旅行商问题（TSP）[4]，在 CPU 时间和准确性方面达到了很好的计算效果。在本文，ASAvns 算法中融合这个循环到两个搜索部分里，更加快速地收敛解空间中的解，高效率地找到效果更佳的全局最优解。

3.1 参数设置

在该算法中，需要用到很多参数而且它们需要在算法初始化阶段赋予一个初值。在自适应模拟退火算法解决 TSP 问题的论文[5]中给出的自适应参数具体值的基础上，通过对不同的参数值进行多次的实验，得到最佳的参数设置值，参数赋值情况如表 3.1：

表 3.1 参数设置值

参数	意义	值
N	工作个数	样例中大小
M	机器个数	样例中大小
Initial_t	初始温度	1000.0
cool_t1	第一阶段温度冷却系数	0.99999
cool_t2	第二阶段温度冷却系数	0.9999
vns_t1	第一阶段变邻域搜索循环次数	N/3
vns_t2	第二阶段变邻域搜索循环次数	3×N/2
end_t1	结束温度	0.0025
end_t2	第二阶段变邻域搜索的开始温度	0.005
current_t	当前温度	current_t =current_t×cool_t

3.2 变邻域搜索算法

新算法中的变邻域局部搜索算法是在最原始的变邻域搜索算法解决流水线调度问题[6]的框架上进行改善的。它包含两个基本的邻居结构。从本质上来说，变邻域局部搜索的最关键部分就是如何定义工作调度序列的邻居结构。大部分智能算法解决工作调度问题，广泛采用的邻居结构有两种即：交换和插入。然而通过借鉴以前的文献和实验结果，我们发现交换的邻居结构在获得解的质量和效率上没有插入的这种邻居结构效果好。所以我们决定把交换这种邻居结构放在算法初始化阶段，给在局部搜索中的初始序列 π 一个轻微的扰动，改变初始解，使整个算法不会很快陷入局部最优而无法获得全局最优解。当然，我们选择了邻居结构 insert 进入邻居结构集合。另外，从文献[7]中我们发现另一种优秀的邻居结构，序列插入。经过在多种流水线调度问题上的试验发现，该结构获得高质量解，因此这个邻居结构也被采用，放到邻居结构集合中。该算法中使用到的邻居结构说明如下所示：

插入（insert）：在工作调度序列中移除在 i 位置的工作，把它插入到该序列 j 位置中。（i,j 是 1 到 N 的随机数，而且 i≠j）

序列插入（block insert）：移除一个在 i 位置和 j 位置之间的工作调度序列，把它插入该序列的 k 位置中。（i,j,k 是 1 到 N 的随机数，而且 i≠j≠k）

交换（swap）：交换工作调度序列中 i 位置和 j 位置的工作。（i,j 是 1 到 N 的随机数，而且 i≠j）

以上所说的邻居结构集合中的两种邻居结构就是插入和序列插入，将它们应用到最初提到的变邻域局部搜索基本结构中，可以得到 VNS 的伪代码如下：

ASAvns 中对工作调度序列π进行的变邻域局部搜索算法的伪代码

```
Input: the permutation π
Begin
s0=π;
i=rand(1,N);j=rand(1,N);i≠j;
s1=swap(s0,i,j);
   For ii=1:(N×(N-1)/2)
        k_count=0;
     While (k_count<2)
              If (k_count=0)
                  i=rand(1,N);j=rand(1,N);i≠j;
                  s2=insert(s1,i,j);
              End if
              If (k_count=1)
```

i=rand(1,N);j=rand(1,N);k=rand(1,N);i≠j≠k;
s_2=block insert(s_1,i,j,k);
End if
If *(fitness(s_2)<fitness(s_1))*
s_1=s_2;
k_count=0;
Else
k_count=k_count+1;
End if
End while
End for
If *(fitness(s_1) ≤fitness(s_0))*
s_0= s_1;
End if
output: the permutation s_0 and the fitness(s_0)
End

3.3 带变邻域搜索的自适应模拟退火算法

应用上节提出的变邻域局部搜索算法，基于基本模拟退火算法框架，可以得到我们提出的 ASAvns 算法框架。

为了进一步得出更好的解，我们采用两阶段局部搜索方式。在算法的开始阶段，执行第一阶段的局部搜索，使用变邻域局部搜索算法获得一些可行解。当温度降低至 0.005 时，第二阶段局部搜索开始进行。在这阶段中，重新初始化参数是必须的，目的是快速收敛解空间中的解。同时，循环过程再一次被使用，为了收获在第一阶段获得解基础上的更好质量的解，即更优的工作调度序列。需要特别注意的是，在算法中需要设置一个标志参数，保证在第二阶段中只重新初始化参数一次。循环中，如果解比上一次循环中的解适应度小的话，表示新解更优，新解将会替换旧解；否则，当循环次数为 N 或者 N 的倍数的时候，循环中的解将会被强制接受，就如第一章中模拟退火算法提到的基本思想一样，这里也是自适应的体现。如此是为避免算法陷入局部最优而无法获得理想中的全局最优解。与此同时，如果算法完成完整的整个循环，需要耗费太长时间，所以，在循环中我们提出一个时间段上限（time limit），若循环时间超过它，循环就终止。这样就保证此算法的 NFSP 样例实验时间和其它算法耗费的时间相同或者相近。这个时间段上限是根据具体的 NFSP 样例和其它算法耗费时间大小规定的。具体的大小会在下面讲解实验结果对比时提及。

最后，算法结束条件是温度到达终止温度或者循环时间到达时间段上限。整个 ASAvns 算法可以循环 10 到 20 次，并将其中最好的解作为最终解。

整个 ASAvns 算法框架伪代码如下所示：

ASAvns 算法解决 NFSP 伪代码

```
Begin
Input: the data of the NFSP;
Initialize parameters including Initial_t, cool_t1, vns_t1;
Induce one solution s randomly;
count=0;
current_t=Initial_t;
While (current_t≥end_t1 or the stopping criterion is not satisfied )
    If (this statement is first executed and the current_t<end_t2)
        Reinitialize the parameters including Initial_t, cool_t2, vns_t2;
    End if
    For times=1: vns_t1 or vns_t2
      s'=s;
      If(runtime>time limit)
        break;
      End if
      Carry out the variable neighborhood search for s' and obtain a new
solution s'';
      If(fitness(s'')<fitness(s'))
        s'=s'';
      End if
    End for
    If (fitness(s')<fitness(s))
      s=s';
    Else
    {
      count=count+1;
      If(count%N=0)
        s=s';
      End if
    }
    End if
    current_t =current_t× cool_t;
End while
s*=s;
Output: the best solution s* and the final result as the fitness(s*)
End
```

第四章 实验结果

实验是检验真理的唯一标准，验证一个算法的性能只能靠实验获得。本章将该算法与其它构造启发式算法和元启发式算法进行对比，得到最后实验结果显示ASAvns 算法在性能和效率上明显优于其它算法。

4.1 开发工具及测试环境

我们提出的解决 NFSP 的 ASAvns 算法是用标准 C 语言进行编码的。

开发工具：DEV-C++ 4.9。

测试环境：

1. 操作系统：Windows 7;
2. 处理器：Intel core i7-2600 3.4 GHz CPU;
3. 内存：4GB。

4.2 实验结果分析

为了测试 ASAvns 算法的效率和解决 NFSP 的能力，12 组不同规模的 NFSP 测试样例将被采用来检验实验结果。它们是由 Taillard[8]设计的。每一组测试样例包含 10 个相同规模的具体的 NFSP 样例。规模是从 20 个工作×5 个机器到 50 个工作×20 个机器变动，标号从 Ta001 到 Ta120。

4.2.1 ASAvns 和 NEH 算法对比

在这一部分，为了保证对比的效率和正确性，NEH 算法和 ASAvns 算法会在相同的运行环境下进行。防止出现即使是相同算法，在不同电脑环境下运行，实验结果也会有差别的情况。我们定义了一个统计参数 PIP 来衡量 ASAvns 算法相对 NEH 算法性能提高的百分比，计算公式为：|ASAvns-NEH |/NEH×100%，计算结果越小表明算法性能相对 NEH 算法提高的越高。表 4.1 显示了 ASAvns 与 NEH 算法最大完成时间对比结果及 PIP 的值。

表 4.1 NEH 和 ASAvns 对比结果

Problem	Scale	NEH	ASAvns	PIP
	n×m	Makespan	Makespan	

Ta001	20×5	1 413	**1 380**	**-2.335%**
Ta011	20×10	2 266	**2 188**	**-3.442%**
Ta021	20×20	3 482	**3 205**	**-7.955%**
Ta031	50×5	3 028	**3 014**	**-0.462%**
Ta041	50×10	3 579	**3 426**	**-4.275%**
Ta051	50×20	5 813	**5 408**	**-6.967%**
Ta061	100×5	5 874	**5 836**	**-0.647%**
Ta071	100×10	6 893	**6 756**	**-1.988%**
Ta081	100×20	9 667	**9 367**	**-3.103%**
Ta091	200×10	11 926	**11 684**	**-2.029%**
Ta101	200×20	15 330	**15 004**	**-2.127%**
Ta111	500×20	30 600	**30 049**	**-1.801%**
Average	PIP:	**-3.094%**		

从表 4.1 中得出最好的 PIP 达到-7.955%，平均的 PIP 有-3.094%。这样的结果显示 ASAvns 算法性能明显优于 NEH 算法。从加黑数据也可以明显看出 12 个 NFSP 不同规模的测试样例，经 ASAvns 算法计算得到的结果均比 NEH 的对应结果优秀。

由 ASAvns 算法获得的表 4.1 中的 12 个不同规模 NFSP 测试样例[8]的具体工作调度安排序列及对应的最大完成时间如表 4.2 所示。

表 4.2 ASAvns 算法工作调度序列

	Scheduling program	Makespan
Ta001	8 9 17 15 10 14 5 19 1 16 3 6 7 13 4 2 18 20 12 11	1380
Ta011	17 19 10 9 15 5 4 6 12 8 14 20 3 7 2 11 13 18 1 16	2188
Ta021	18 9 16 15 10 5 19 8 3 7 17 4 11 2 12 13 1 20 6 14	3205
Ta031	19 26 4 8 46 36 20 34 11 48 38 32 21 44 30 13 27 6 1 7 22 18 24 25 17 39 28 2 31 35 12 5 41 9 29 43 15 14 10 23 40 45 16 42 33 47 49 37 50 3	3014
Ta041	41 44 33 15 48 36 14 20 49 23 42 8 3 11 6 45 40 21 18 9 28 43 31 22 19 1 16 10 2 29 7 4 35 26 12 27 37 38 46 32 5 50 13 47 17 34 25 30 39 24	3426
Ta051	43 31 5 6 45 15 24 13 29 10 35 33 20 16 48 34 18 17 3 50 1 26 2 7 39 22 47 23 46 49 28 42 38 36 27 11 19 40 9 8 25 30 21 4 32 44 14 41 37 12	5408
Ta061	83 10 71 94 87 72 85 96 81 65 82 31 2 28 69 41 39 76 36 97 15 21	5836

	40 27 95 16 34 55 73 20 7 44 78 59 58 50 74 25 3 32 93 57 26 13 90 75 67 38 35 77 52 30 45 18 5 9 14 91 61 64 60 89 68 56 6 23 53 19 84 22 43 51 80 4 46 98 99 29 100 42 8 92 24 49 86 37 11 12 54 70 47 1 62 66 63 48 33 79 88 17	
Ta071	93 58 15 95 6 72 56 21 44 37 68 10 54 61 48 96 71 65 57 85 80 97 86 64 40 41 34 3 7 77 83 67 66 32 60 49 31 22 53 17 36 42 25 39 35 70 38 50 13 1 75 55 24 92 82 90 88 30 100 52 79 23 29 18 4 19 94 51 87 26 74 46 14 11 5 73 78 99 28 12 33 91 2 89 43 98 76 20 59 63 69 16 9 27 8 81 62 84 47 45	6756
Ta081	98 75 58 74 99 89 18 10 54 63 22 86 3 39 14 34 90 1 88 41 43 42 46 76 8 45 91 72 20 35 15 9 19 40 96 2 49 47 31 55 33 5 83 51 56 12 61 38 26 87 94 64 29 67 82 62 77 78 32 100 81 84 21 11 71 53 65 25 50 6 16 97 69 36 85 66 44 70 4 30 60 28 27 57 92 79 7 24 80 13 23 68 95 73 17 93 48 37 52 59	9367
Ta091	96 197 1 75 162 172 154 29 99 15 10 183 142 104 134 123 67 138 189 102 2 194 44 73 150 49 57 107 173 131 39 137 124 127 186 48 117 91 46 19 152 22 195 160 111 108 9 79 59 121 76 196 153 24 33 143 85 70 18 130 171 62 27 148 72 193 175 156 113 132 3 17 105 16 43 66 110 128 35 191 4 188 126 135 122 83 90 159 13 68 36 77 170 30 6 151 144 115 74 81 190 185 60 161 98 106 20 89 63 93 187 45 181 56 52 146 147 80 87 92 200 12 71 54 125 116 34 174 78 42 82 141 84 112 165 58 47 25 11 21 69 166 101 139 86 169 164 7 179 114 97 95 198 5 158 31 26 145 177 163 129 199 8 176 40 14 167 37 140 38 51 28 178 32 182 50 109 61 53 119 180 94 118 41 88 184 192 136 64 120 149 55 155 157 103 168 100 23 133 65	11684
Ta101	83 69 183 89 65 190 90 109 104 76 141 95 137 49 59 107 177 126 2 66 147 134 11 151 61 57 37 13 25 50 81 176 132 93 91 113 188 29 105 48 127 172 111 140 195 22 173 178 92 129 115 74 9 179 108 144 106 199 62 75 150 154 85 70 77 20 45 156 63 162 67 164 123 165 152 43 47 5 170 187 192 100 168 24 3 196 26 98 130 103 180 16 94 120 38 88 118 87 80 160 175 68 41 6 122 124 46 186 28 101 117 198 21 33 78 36 139 200 1 71 27 184 56 4 149 31 35 121 79 23 7 157 15 135 191 146 194 17 143 99 174 19 163 153 155 96 42 181 58 136 161 40 131 158 53 18 145 102 128 110 30 8	15004

	197 39 34 44 10 116 73 166 64 72 84 182 169 112 12 185 97 55 114 159 119 133 142 193 148 54 52 60 51 125 189 167 138 32 86 171 82 14	
Ta111	307 70 475 55 37 298 158 22 86 452 387 374 399 94 180 306 383 356 472 385 353 481 85 114 166 56 226 243 185 135 280 369 316 184 465 384 231 51 362 467 367 42 162 338 290 410 145 303 331 343 282 346 147 6 139 159 377 471 198 407 109 491 97 477 194 313 344 152 402 210 368 26 325 148 302 165 323 250 143 255 496 335 100 447 29 274 153 224 370 473 213 89 62 91 95 458 488 389 149 99 469 288 128 479 381 319 237 311 435 202 64 111 463 350 110 483 217 415 214 448 146 349 395 222 167 314 341 16 333 275 183 248 487 492 216 205 15 378 238 339 358 345 52 130 96 304 181 449 32 498 433 400 138 74 57 422 129 408 359 321 108 480 120 10 246 478 334 388 28 4 115 191 296 386 132 54 79 101 464 420 125 43 245 58 131 318 227 31 373 229 315 119 186 284 283 41 154 285 372 27 438 459 256 267 116 484 411 489 431 196 127 295 401 179 398 73 454 177 34 427 142 254 337 33 212 47 141 497 84 234 270 342 456 494 60 126 258 293 66 8 312 233 468 474 174 63 230 88 151 239 336 264 470 327 204 1 263 106 366 332 173 30 140 414 121 286 364 413 244 113 242 396 355 428 430 69 192 61 281 161 46 289 215 50 107 278 382 391 348 13 78 72 297 423 460 466 320 24 424 261 82 299 347 123 376 171 45 294 418 209 122 39 279 178 409 189 90 172 92 442 137 187 53 287 265 406 223 18 144 324 301 176 499 21 203 445 105 9 361 451 495 437 49 309 363 236 93 365 225 201 273 220 375 67 380 485 262 328 441 228 310 300 112 136 219 157 118 134 421 38 416 253 195 207 257 156 461 268 77 440 390 65 80 436 357 482 218 392 40 394 308 232 25 133 193 190 371 453 206 124 476 354 211 439 247 2 490 432 500 102 269 163 405 434 221 252 271 155 182 23 76 48 493 169 404 68 486 277 175 305 14 457 36 71 443 19 117 259 450 35 240 44 170 330 425 444 87 419 266 17 393 322 403 12 417 7 352 329 351 446 379 98 104 200 397 5 291 20 188 272 81 241 164 412 292 249 251 462 276 326 168 11 197 317 59 208 235 426 260 3 429 75 360 160 103 455 83 150 340 199	30049

4.2.2 ASAvns 和 IG，KK 算法对比

在 2007 年，Baraz 和 Mosheiov[9]提出 IG 算法，它是一个构造性启发式算法，叫做改进贪婪方法。它以一个简单的贪婪算法为基础，会在每个步骤中的当前工作调度序列上添加一个后继工作，在每个改进步骤中,使用成对交换操作。而 KK 算法是由 Kalczynski 和 Kamburowski[10]提出的。它们的数据测试结果表明相比由 Saasani et al.[11,12] 提出的启发式算法和专门针对零空闲流水线调度问题进行改良后的 NEH[13]算法来说，这些启发式算法性能更高一些。

同样，为了对比 ASAvns 算法和 IG，KK 算法，我们采用了上文使用到的 12 组不同规模的 NFSP 测试样例[8]进行对比。在衡量算法表现的过程中，我们使用到的统计参数有平均相对差错百分比 ARE（the average relative percent error），平均标准差（the average standard deviation）和平均时间 AT（the average time）。当 ARE 越小时，表示算法的具体问题结果就越好。表 4.3 给出了算法 ASAvns 和算法 IG，KK 的 ARE，ASD 和 AT 的对比结果。

表 4.3 IG，KK 和 ASAvns 算法结果对比

Scale	IG	KK	ASAvns		
n×m	ARE	ARE	ARE	ASD	AT
20×5	9.19	0.87	**-3.15**	0.23	1.0S
20×10	8.37	2.00	**-7.73**	0.08	1.0S
20×20	5.40	1.29	**-7.28**	0.03	1.0S
50×5	11.53	-0.03	**-1.37**	0.02	5.0S
50×10	12.34	-1.79	**-6.12**	0.03	5.0S
50×20	12.44	-0.55	**-7.58**	0.03	5.0S
100×5	16.40	-0.24	**-1.21**	0.01	10.0S
100×10	13.98	0.08	**-2.45**	0.01	10.0S
100×20	14.78	-2.57	**-5.47**	0.01	10.0S
200×10	17.18	-0.55	**-2.15**	0.00	20.0S
200×20	16.38	-2.23	**-4.15**	0.01	21.0S
500×20	18.68	-1.65	**-1.18**	0.00	67.4S
Average	13.055	-0.447	**-4.15**	0.04	13.07S

从表中观察可以得到，ASAvns 算法获得的解比之前的构造性启发式算法 IG，KK 的解优化很多。ASAvns 算法得到的 ARE 均比 IG，KK 的小，表示 ASAvns 算法有着比它们更高的性能。进一步讲，表中 ASAvns 得到的 ASD 也较小。总体来说，ASAvns 算法是一种在解决优化 NFSP 中，有着高效率和高性能的元启发式算法。

4.2.3 ASAvns 和 PSOvns, HDPSO 算法对比

在文献[14]中，一种专门为解决 NFSP 而改进的粒子群算法 PSOvns 被提出，它的数据测试结果显示，该算法比单纯的变邻域局部搜索算法和基因算法具有更高的性能。在文献[15]中，提出了一种混合离散粒子群算法（HDPSO）解决 NFSP，它获得的解优于粒子群算法和其它元启发式算法，通过观察这个算法的相关实验结果就可以看出。在这部分中，我们采用与上面实验对比相同的对比方式，完成了 PSOvns 和 HDPSO 算法与 ASAvns 算法的实验对比。12 组不同规模的 NFSP 测试样例平均对比结果由表 4.4 展示。以上提到的时间段上限，和下面结果中用到的平均时间 AT 相近，不同规模的上限时间分别为 1.0s，5.0s，10.0s，20.0s，70.0s。这样保证 ASAvns 算法的 AT 与 PSOvns 算法和 HDPSO 算法的 AT 相同或者相近，尽可能保证实验在相同条件下进行对比。

表 4.4 PSOvns，HDPSO 和 ASAvns 算法结果对比

Scale	PSOvns			HDPSO			ASAvns		
n×m	ARE	ASD	AT	ARE	ASD	AT	ARE	ASD	AT
20×5	-3.01	0.04	2.0S	-3.03	0.00	2.0S	**-3.15**	0.23	1.0S
20×10	-7.16	0.15	2.0S	-7.28	0.02	2.0S	**-7.73**	0.08	1.0S
20×20	-6.81	0.27	2.0S	-7.04	0.03	2.0S	**-7.28**	0.03	1.0S
50×5	-0.75	0.00	5.0S	-0.75	0.00	5.0S	**-1.37**	0.02	5.0S
50×10	-5.75	0.22	5.0S	-5.88	0.10	5.0S	**-6.12**	0.03	5.0S
50×20	-6.65	0.42	5.0S	-7.27	0.21	5.0S	**-7.58**	0.03	5.0S
100×5	-0.72	0.03	10.0S	-0.73	0.00	10.0S	**-1.21**	0.01	10.0S
100×10	-1.40	0.11	10.0S	-1.55	0.03	10.0S	**-2.45**	0.01	10.0S
100×20	-4.66	0.33	10.0S	-5.36	0.08	10.0S	**-5.47**	0.01	10.0S
200×10	-1.41	0.12	20.0S	-1.54	0.02	20.0S	**-2.15**	0.00	20.0S
200×20	-3.23	0.31	20.0S	-3.71	0.13	20.0S	**-4.15**	0.01	21.0S
500×20	-1.85	0.15	50.0S	-2.16	0.10	50.0S	-1.18	0.00	67.4S
Average	-3.62	0.18	11.75S	-3.86	0.06	11.75S	**-4.15**	0.04	13.07S

从加黑实验结果可以看出在相同或者相近基准上，ASAvns 算法有 11 组测试样例即 110 个 NFSP 的具体问题结果优于 PSOvns 和 HDPSO 算法。当然，12 组测试样例的 ARE 的平均值小于 HDPSO 有 0.29%，小于 PSOvns 有 0.53%。这表示，ASAvns 算法在更广泛地规模上获得更优化的结果即更小的最大完成时间。ASD 的平均值比 PSOvns 和 HDPSO 小，表明该算法有着更强的鲁棒性。在平均时间方面，第一组测试样例即前 10 个 NFSP 具体问题测试样例的平均时间为 1.0s, 比 HDPSO 和 PSOvns 算法少 1s，第二组到第十一组测试样例的平均时间和 HDPSO 和 PSOvns 相同，最后一组测试样例时间花费比 HDPSO 和 PSOvns 多 27.4s。虽然这样，由于 ARE 的平均值高于 HDPSO 和 PSOvns 算法，我们仍可以总结说 ASAvns 算法显著提高了算法性能，优化了 NFSP 具体问题计算结果。它比 PSOvns 算法和 HDPSO 算法有着更优越的效率和更强的鲁棒性。

总结与展望

本论文中是要解决零空闲流水线调度优化问题，在以最小化最大完成时间为目标的前提下，我们提出了一种带变邻域搜索算法的自适应模拟退火算法。

零空闲流水线调度问题，简单地说是机器一旦开始工作就不能停止。正是由于问题这样的特殊性，在研究深度方面没有其它普通调度问题广泛。最初，我们简要介绍了模拟退火算法，其次，具体解释了零空闲流水线调度问题，接着阐述了 ASAvns 算法是如何解决零空闲流水线调度问题（NFSP）的。ASAvns 算法有两个优势特点，第一个是基本模拟退火算法基础上的两阶段局部搜索和自适应参数调整，第二个是该算法将两种简单搜索方法包括插入和段插入局部搜索方式，嵌入到变邻域搜索算法基本框架里，一种独特的高效变邻域搜索算法就此形成。然而，这些特点需要具体的实验来证明，因此在论文的最后，我们对 12 组不同规模的 NFSP 测试样例进行实验，用统计参数 ARE，ASD，AT 来表明 ASAvns 算法的高性能，高效率以及广泛的鲁棒性。

接下来，我们的工作是思考更高效的局部搜索策略和智能优化算法，应用它们，进一步提高算法性能，并试验采用此算法解决更多调度问题。同时，我们希望有更多的学者应用此算法解决更多有着类似目标函数的组合优化问题，让它成为一个广泛使用并有着独特吸引力的算法。

参考文献

[1] D. Baraz, G. Mosheiov, A note on a greedy heuristic for the flow-shop makespan minimization with no machine idle-time, Eur. J. Oper. Res. 184 (2) (2008) 810–813.

[2] S. Kirkpatrick, C.D. Gelatt Jr., M.P. Vecchi, Optimization by simulated annealing,Science 220 (1983) 671–680.

[3] Q.K. Pan, L. Wang, No-idle permutation flow shop scheduling based on a hybrid discrete particle swarm optimization algorithm, Int. J. Adv. Manuf. Technol. 39 (7–8) (2008) 796–807.

[4] Wang Z, Geng X, Shao Z. An effective simulated annealing algorithm for solving the traveling salesman problem[J]. Journal of Computational and Theoretical Nanoscience, 2009, 6(7): 1680-1686.

[5] Geng X, Chen Z, Yang W, et al. Solving the traveling salesman problem based on an adaptive simulated annealing algorithm with greedy search[J]. Applied Soft Computing, 2011, 11(4):3680-3689.

[6] Tasgetiren M F, Liang Y C, Sevkli M, et al. A particle swarm optimization algorithm for makespan and total flowtime minimization in the permutation flowshop sequencing problem[J]. European Journal of Operational Research, 2007, 177(3): 1930-1947.

[7] Geng X, Chen Z, Yang W, et al. Solving the traveling salesman problem based on an adaptive simulated annealing algorithm with greedy search[J]. Applied Soft Computing, 2011, 11(4):3680-3689.

[8] E. Taillard, Benchmarks for basic scheduling problems, Eur. J. Oper. Res. 64 (2) (1993) 278–285.

[9] D. Baraz, G. Mosheiov, A note on a greedy heuristic for the flow shop makespan minimization with no machine idle-time, Eur. J. Oper. Res. 184 (2) (2008) 810–813.

[10] P.J. Kalczynski, J. Kamburowski, A heuristic for minimizing the makespan in no-idle permutation flow shop, Comput. Ind. Eng. 49 (1) (2005) 146–154.

[11] Saadani NEH, Guinet A, Moalla M(2005) A traveling salesman approach to solve the F/no-idle/Cmax problem. Eur J Oper Res161:11–20.

[12] Saadani NEH, Guinet A, Moalla M (2001) A traveling salesman approach to solve the F/no-idle/Cmax problem. Proceedings of the international conference on industrial engineering and production management (IEPM'2001) Quebec (Vol. 2) pp. 880–888.

[13] Nawaz M, Enscore EE Jr., Ham I (1983) A heuristic algorithm for the m-machine, n-job flow shop sequencing problem. OMEGA 11:91–95.

[14] Su Shoubao, Wang Jiwen, Zhang Ling, et al., An invasive weed optimization algorithm for constrained engineering design problems, J. Univ. Sci. Technol. China 39 (8) (2009) 885–893.

[15] Q.K. Pan, L. Wang, No-idle permutation flow shop scheduling based on a hybrid discrete particle swarm optimization algorithm, Int. J. Adv. Manuf.Technol. 39 (7–8) (2008) 796–807.

致　谢

衷心感谢我的导师殷明浩老师对我的支持和帮助，他教会我很多学习方法，令我收获颇丰。同时，感谢殷明浩老师和李向涛师兄在整个设计过程中对我的指导，帮助我解决了设计过程中遇到的各种难题。感谢我的毕业设计小组的所有成员给予我的帮助。

最后，感谢所有关心和帮助过我的老师，亲人和朋友以及在设计中被我引用或参考的论著的作者。