一种求解车间作业调度问题的混合邻域结构搜索算法*)

曾立平　黄文奇

(华中科技大学计算机科学与技术学院　计算机理论研究所　武汉430074)

摘　要　车间作业调度问题是优化组合中一个著名的难题,问题的目标是在满足约束条件的前提下,使调度的加工周期尽可能小。文章中提出了利用新的混合邻域结构进行搜索来求解车间作业调度问题。对于算法关键的邻域构造问题以及跳坑策略给出了提高算法优度的解决方案。采用43个不同规模和难度的国际标准算例做为本算法的测试实验集,39个算例找到了最优解,其中包括著名的难例FT10。与当前国外学者提出的一种先进算法进行了比较,算法的优度高于被比较的先进算法。

关键词　车间作业调度,邻域结构,局部搜索,跳坑策略

A Local Search Method with Hybrid Neighborhood for the Job Shop Scheduling Problem

ZENG Li-Ping　HUANG Wen-Qi

(Theoretical Computer Science Institute, School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan 430074)

Abstract　Job shop scheduling problem is well known to be a difficult, strongly NP-complete problem, the objective of this is minimizing the completion time of all the jobs, called the make span, subject to the constraints of this kind of problem. A local search method with hybrid neighborhood is presented for job shop scheduling problem. The key problems for local search method, such as neighborhood structure and off-trap strategy, the solutions are made to decrease the makespan of the schedule. Our approach is tested on a total of 43 standard problem instances, 39 instances are found optimal solutions, including the notorious instances FT10. Our approach yields better solutions than a particularly efficient algorithms discussed in the literature.

Keywords　Job shop scheduling, Neighborhood, Local search, Off-trap strategy

1　引言

生产调度是CIMS研究领域生产管理的核心内容和关键技术,车间作业调度问题(JSSP)是最困难的组合优化问题和典型的NP难问题[1],这类问题的精确求解所需的计算时间会随着问题规模的增大而呈现指数膨胀,没有一个有效的算法能在多项式时间内求出其最优解。现代经济日益激烈的竞争趋势和不断变化的用户需求要求生产者不断降低生产成本,如更短的产品加工周期等,而JSSP是利用有限的资源满足被加工任务的各种约束,并确定工件在相关设备上的加工顺序和时间,以保证所选择的性能指标最优,能够提高企业的经济效益,JSSP具有很多实际应用背景,开发有效而精度高的调度算法是调度和优化领域重要的课题。

从问题开始提出之日起,许多学者就一直从事着该问题的研究,并对该问题提出了各种最优化算法和近似算法。最优化算法包括混合整数线性规划(MILP)、分枝定界(R&B)法以及拉氏松弛法等,近似算法通常为迭代算法,如邻域搜索[2]、移动瓶颈(SB)[3]、禁忌搜索(TS)[4]、模拟退火(SA)[5]、遗传算法(GA)[6]、神经网络(NN)等。

最优化算法虽然能从理论上求得最优解,但由于其计算时间太长的原因,不能获得真正的实用。与最优化算法相比,近似算法的明显优势在于计算效率高,算法灵活多变但近似算法有明显不足之处,即有可能出现所产生的解比全局最优解差很多的情况,而且差的程度总是不够明确。因此,合理的计算时间和所求出的解的最优性就成为衡量启发式算法性能的标准。单独使用一种近似方法不如两种近视方法结合起来形成的混合近似算法取得的结果好。近年来,一些近似方法的综合应用,比如局部邻域搜索结合移动瓶颈过程的算法[7]、禁忌搜索结合移动瓶颈过程[8],取得了很好的效果。

实际上,移动瓶颈(SB)、禁忌搜索(TS)、模拟退火(SA)、遗传算法(GA)都属于邻域搜索算法。邻域搜索以一定的概率接受劣解,从而跳出局部极小值,模拟退火和禁忌搜索是两种著名的跳出局部极小值的框架。邻域搜索结果的好坏与邻域的构造、跳出局部极小值方法的选择有很大关系,本文将讨论处理这些问题的具体方法,提出了一种新的邻域搜索算法。

2　问题

给定一组工件和机器,每个工件都包含若干道给定的按先后次序加工的工序,每道工序需要占用某一台机器进行加工。问题需要满足两个约束条件:a. 每个工件的工序加工先后次序是确定的;b. 每台机器在任意时刻最多只能加工一个工序,某个工序的加工一旦开始,就不能中途停止。车间作业调度就是在满足上述约束条件的前提下,所有工件的所有工序都分配到了时间段在相应的机器上进行加工,并使从最早开始加工时刻开始到所有工序完成加工的时刻结束这个时间段最小,这个时间段我们称之为加工周期。

*)本课题得到国家重点基础研究发展规划973项目资助(No. G1998030600)。曾立平　博士,主要研究方向是NP难问题。黄文奇　教授,博士导师,主要研究方向是可计算性与计算复杂性理论及应用。

假定：$J=\{1,2,\cdots,n\}$代表工件集合，$M=\{1,2,\cdots,m\}$代表机器集合，$N=\{0,1,\cdots,n^*,n^*+1\}$代表工序集合，在集合$N$中，0和$n^*+1$分别表示虚拟的开始和结束工序，集合$A$表示受给定工件工序先后次序约束的成对的工序集合，集合E_k表示在第k台机器上加工的工序集合。$d_i(d_i>0$，常量）表示第i个工序在相应机器上加工的时间，t_i（变量）表示第i个工序在相应机器上加工的开始时间。于是问题可以描述如下[5]：

minimize t_{n^*+1}. 约束条件如下：

$t_j-t_i\geqslant d_i,(i,j)\in A$，

$t_i\geqslant 0,i\in N$，

$t_j-t_i\geqslant d_i \vee t_i-t_j\geqslant d_j,(i,j)\in E_k,k\in M.(P)$

对(P)的任何一个可行的解称为一个调度。

为了更好地理解Job-Shop调度问题，我们以一个4工件、3机器、12工序问题来介绍调度问题的几何模型。

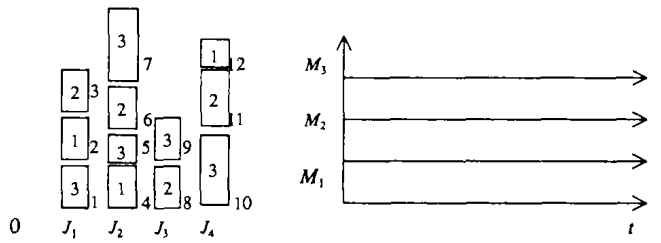

图1　一个调度问题的几何模型

在图1中，每个矩形块代表一道工序，矩形块中间的数字代表加工这道工序的机器序号，矩形块右侧的数字表示矩形块的统一编号，矩形块的高度表示加工这道工序所需时间，左边的每一列矩形块代表一个工件所需要加工的全部工序。J_1，J_2，J_3，J_4是工件序号，M_1，M_2，M_3是机器序号。t轴为时间轴。

需要解决的问题是如何把这些矩形块调度到图1右边所示的横线上，使形成的最终格局的加工周期尽可能短。在调度过程中需要满足两个条件：a. 在横线上矩形块的相对位置关系和原来这些矩形块在工件上的相对顺序关系必须一致，比如说工件J_2中矩形块6至少比矩形块5的底高出d_5，那么这两个矩形块在放进块子篓时，它们需满足关系$t_5+d_5\leqslant t_6$；b. 同一横线上的任何两个矩形块不能重叠。图2表示图1在按照某种算法完成调度之后的一个最终格局。

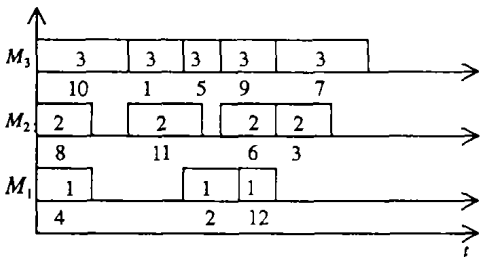

图2　图1所示问题的最终调度格局

3　混合邻域结果搜索算法

邻域搜索算法的基本过程是从一个初始解s出发，确定满足特定条件的解s的邻域$N(s)$，然后按照某种搜索策略从邻域$N(s)$中选出一个可行的解s'，从s移动到s'，从新的出发点再进行搜索。为了避免陷入循环和局部极小值，如果邻域$N(s)$中所有解的加工周期都大于或者等于初始解的加工周期，则需要按照某种跳坑策略随机选择一个可行解s'，从s移动到s'，从新的出发点再进行搜索。在阐述本文算法之前，先定义与本文算法相关的几个重要的概念。

对于车间作业调度问题，我们是按照先后顺序对每道工序进行调度，假定S是所有工序被调度时的先后顺序序列，$MACHIINE_S$是相对于机器的工序被调度时的先后顺序序列集合，$MACHINE_S=\{S_k|S_k$是第k台机器上的所有工序被调度时的先后顺序序列$\},k\in M$。在我们的算法中，假设i与j是同工件位置相邻的两个工序，i的位置在j之前，只有当i已经被调度后，才能去调度j，根据这个规定，下面引入“前沿工序”的定义。

定义1（前沿工序）　假定i与j是同属于工件n的位置相邻的两个工序，i的位置在j之前，当算法进行到某一格局时，i已被调度而就j未被调度，则j被称为当前格局下工件n的前沿工序。对于几何模型，前沿工序就是位置最低的矩形块。

定义2（前沿贪心方法）　在每次在调度工序时，只考虑调度每个工件的前沿工序，在调度到相应的机器上时，在满足合法性的前提下，使其开工时间为最小值。对于几何模型，就是在调度前沿矩形块到横线上时尽量靠左调度。

定理1　按照前沿贪心方法给出的调度集合一定包含Job-Shop问题的最优解。

证明：任意给定的合法的调度*Schedule* 1，我们按照调整顺序采用前沿贪心方法来对*Schedule* 1的所有工序进行调整，调整完成后的调度为*Schedule* 2。调整顺序的排序规则为一个二元组：〈工序的加工开始时间，工序所在机器的序号〉。其中工序的加工开始时间是小优先，在工序的加工开始时间相同的情况下，再考察工序所在机器的序号，工序所在机器的序号小的工序别优先被排序。

我们按照加工开始时间小优先的规则来排序，那么调整顺序符合确定的工件n的工序加工先后次序，因此，当我们按照调整顺序去调整每一个工序时，每一个工序都是前沿工序。现在我们按照调整顺序对*Schedule* 1的每一道工序采用贪心方法进行调整，即对于每道工序在满足约束条件的前提下使其开工时间最小，$\forall i\in N,t'_i$为调整后的开工时间，显然$t_i\leqslant t'_i$，所以调整完成后*Schedule* 2的加工周期一定小于或等于*Schedule* 1的加工周期。

因为调整的顺序是前沿序，我们又用贪心的方法对每一道工序进行调整，所以*Schedule* 2是一个按照前沿贪心方法来调度所有工序得到的调度。所以对于任意给定的调度，我们按照前沿贪心方法对同一个问题重新给出调度后，新调度的加工周期一定小于或等于给定调度的加工周期。因此，按照前沿沉底方法给出的调度集合一定包含Job-Shop问题的最优解。

定义3（关键路径和关键路径上的块）　对于一个按照前沿贪心方法给出的调度，关键路径定义为：$(i_1,i_2,\cdots,i_{m-1},i_m)$：其中$i_1$是所有工序中开工时间最早的工序，$i_m$是最晚完工的工序，满足$t_{im}=t_{im-1}+d_{im-1}$，$t_{im-1}=t_{im-2}+d_{im-2}$，$\cdots$，$t_{i2}=t_{i1}+d_{i1}$，$t_{i1}=0$。如果$i_k$在关键路径上，$j_1$是$i_k$的同工件工序，$j_2$是$i_k$的同机器工序，并且$t_{ik}=t_{j1}+d_{j1}=t_{j2}+d_{j2}$，那么我们把$j_1$而不是$j_2$作为关键路径上的工序。

设$(i_1,i_2,\cdots,i_{m-1},i_m)$为关键路径，从$i_m$开始，依次将关键路径上的工序依次划分为若干个块，使得满足$t_{ik}=t_{ik-1}+d_{ik-1}(1<k\leqslant m)$并且在同一台机器上加工的关键路径上的工序属于同一块，这样的块我们称为关键路径上的块。

3.1 邻域的构造

邻域结构1：对于给定的一个合法的解，计算出当前状态下所有的关键路径和关键上的块。假定 i 与 j 是属于同一关键路径上的块上的任意两个工序，i 与 j 在机器 k 上加工，在 S_k 上，i 在 j 之前，把 i 的位置移至 j 的位置后称为向后插入，把 j 的位置移至 i 的位置前称为向前插入。通过执行不同的向前或向后插入操作，并保留执行操作之后仍然满足约束条件的解，可以得到一个可行解的集合，这个集合为给定解的第一种邻域。

我们在对关键路径上的块中的工序进行向前插入或向后插入时，包括以下四种情况，假设 i 在机器 k 上加工：

在 S_k 上，假设 i 是关键路径上的块中的最后一个工序：

(1) i 可以向前插入到同一关键路径上的块中的任何工序前面。

(2) i 前属于同一关键路径上的块中的任何工序可向后插入到 i 的后面。

在 S_k 上，假设 i 是关键路径上的块中的第一个工序：

(3) i 可以向后插入到同一关键路径上的块中的任何工序后面。

(4) i 后属于同一关键路径上的块中的任何工序可向前插入到到 i 的前面。

在以上的四种情况中，除去重复的移动。如果是关键路径上的第一个块，不包括(1)、(2)两种情况，如果是关键路径上的最后一个块，不包括(3)、(4)两种情况。

邻域结构1的基本思想是移动关键路径上的工序，考察关键路径上的块中第一个和最后一个工序，改变它们的位置。(1)、(2)两种情况使块中的第一个工序后移。下面介绍算法中另外一种通过单机调度来构造邻域的方法。

假定 $SET(k)=\{POST\_MACHINE\_S \mid$ 机器 k 上的工序被调度时的先后序列 S_k 改变为 S_k'，而其他机器上的工序被调度时的先后序列都保持不变$\}$。

定义4(单机调度中工序的 i 的 $r(i),q(i)$) $\forall\ POST\_MACHINE\_S \in SET(k)$，我们用前沿贪心方法按照 $POST\_MACHINE\_S$ 的顺序，所得到的调度中从第一个工序的开工时间开始到工序 i 的开工时间为止，这个时间段的最小值，我们记为 $r(i)$。

$\forall\ POST\_MACHINE\_S \in SET(k)$，我们用前沿贪心方法按照 $POST\_MACHINE\_S$ 的顺序，所得到调度中从工序 i 的完工时间开始到所有工序的完工时间为止，这个时间段的最小值，我们记为 $q(i)$。

$r(i),q(i)$的计算方法：

假设我们对机器 k 进行单机调度，当前格局下的相对于机器的工序被调度的先后顺序序列集合为 $MACHINE\_S$，$n(k)$为机器 k 上工序的数目。将机器 k 由1台改为 $n(k)$台，仍然按照 $MACHINE\_S$ 的顺序用前沿贪心法摆放完所有工序，在所得到的调度下，机器 k 上的工序 i 的开工时间即为当前格局下的 $r(i)$。

$q(i)$的计算是和 $r(i)$的计算是对称的。将每台机器上的工序被调度的先后顺序序列颠倒，则 $MACHINE\_S$ 变为 $REVERSE\_MACHINE\_S$，将机器 k 由1台改为 $n(k)$台，按照 $REVERSE\_MACHINE\_S$ 的顺序用前沿贪心法摆放完所有工序，在所得到的调度下，机器 k 上的工序 i 的开工时间即为当前格局下的 $q(i)$。

单机调度中工序 i 的最早开工时间 $min(t_i)$：i 是在机器 k 上加工的工序，在某一格局下对机器 k 进行单机调度时，工序 i 的最早开工时间 $min(t_i)$为满足两个约束条件下的最小值。a. 工序 i 不能和机器 k 上已摆放的工序加工时间冲突；b. $min(t_i)\geqslant r(i)$。

单机调度算法为：

(1)在当前格局下，机器 k 上所有的工序都还没有被调度。

(2)在对当前格局的机器 k 进行单机调度时，采用如下的的优先规则来调度机器 k 上的所有工序：$(min(t_i),q(i),d_i)$，$min(t_i)$小的工序优先，$min(t_i)$相同则 $q(i)$大的工序优先，$min(t_i)$和 $q(i)$都相同则 d_i 小的工序优先。如果 d_i 仍然相同，则工序所在工件序号小的优先。

(3)按照如上规则调度完机器 k 上的所有工序。

邻域结构2：对于给定的一个合法的解，任选机器 k 使用上述单机调度算法进行调度，并保留单机调度之后仍然满足约束条件的解，可以得到一个可行解的集合，这个集合为给定解的第二种邻域。

算法在构造邻域时，移动关键路径上的块和单机调度时都是基于工序被调度时的先后次序，而不是工序被调度后的开工时间先后次序，在某个工序被移动的情况下，根据前沿贪心方法，属于同一个工件的其他工序的开工时间可能会自动调整。在文[4,5,7,8]中移动关键路径上的块时是基于工序被调度后的开工时间先后次序，在某个工序被移动的情况下，属于同一个工件的其他工序的开工时间是不能变的。

3.2 邻域搜索策略

对邻域进行搜索的目的是要得到下一次进行邻域搜索的出发点。在本文的算法中，采用的是就近下降法，即在邻域搜索的过程中，只要发现邻域中某个解的加工周期小于出发点所对应的解的加工周期，就停止计算邻域中其他的解，选定刚才所计算的解作为下一次邻域搜索的出发点。如果搜索完邻域中所有的解都没有某个解的加工周期小于出发点所对应的解的加工周期，则进行跳坑。

3.3 禁忌控制跳坑策略

跳坑是指当搜索陷入局部极小值后，按照某种策略随机选择一个解作为新的邻域搜索的出发点。此策略也就称为跳坑策略。

算法使用单机调度作为一种跳坑手段。在算法使用第二种邻域结构搜索时，如果在邻域中找不到某个解的加工周期比出发点所对应的解的加工周期小，则随机选择某台进行单机调度后得到的合法的解作为新的邻域搜索的出发点。

为了避免跳坑时出现重复的解和提高效率，禁忌控制跳坑策略引入了一个记忆装置，即长度为 l 的禁忌表，表中记录了最近进行的 l 个跳坑之前的进行单机调度的机器，在当前的跳坑步骤中，这些跳坑是被禁止的。

在跳坑过程中，算法会选择没有被禁止的跳坑，这个选择要受到存储在禁忌表中信息的制约，因此，适当选择禁忌表的长度是必要的，根据机器数目的不同，l 的取值范围为1～3。

此外，算法还采用了我们称为“随机同工件工序移动”的跳坑方法，方法是：随机选择同一个工件的一半数目的工序，在满足合法性的前提下，先后对工序在相应加工机器上随机移动它们各自的被调度时顺序位置，移动完成后，我们就完成了一次“试跳”。如果试跳后调度的加工周期不超过跳坑前调度的加工周期的110%，试跳被接受，得到一个新的解作为下一次邻域搜索的出发点，如果在算法设定的试跳次数内找不

到这样的解，则在所有的试跳中选择一个最好的解，跳坑，作为下一次邻域搜索的出发点。

算法所采用的跳坑方法，均是从历史最好解起跳，希望能够继承一些对工序不错的调度，能够更快、更有效地寻找更优解。

3.4 算法

我们的算法被命名为 HLS，下面是 HLS 的基本步骤：

Step 1　用 INSA 调度规则[6]生成一个解作为初始解，将初始解作为 HLS 迭代的出发点。

Step 2　判断是否满足停机条件。若满足停机条件，给出历史最好解作为最终解。

Step 3　计算出发点的关键路径和关键路径上的块，计算邻域1中的解。若一旦搜索到邻域1中的某个解的加工周期比出发点所对应的解的加工周期小，则将搜索到的解作为新的出发点，转 Step 2。

Step 4　单机调度，计算邻域2中的解。若一旦搜索到邻域2中的某个解的加工周期比出发点所对应的解的加工周期小，则将搜索到的解作为新的出发点，转 Step 2。

Step 5　若已跳坑次数不能被(2×机器数)整除或者跳坑次数为0，则根据“禁忌控制单机调度”方法跳坑；若能被(2×机器数)整除，则根据“随机同工件工序移动”方法跳坑。将跳坑后得到解作为新的出发点，转 Step 2。

算法给出的停机条件为限定 HLS 的总迭代次数。理论上，如果迭代次数足够多时，有可能搜索到解集中的所有元素，算法将以概率1搜索到最优解(概率最优性)。

表1　所有 FT 和 LA 问题的计算及比较结果

问题	n	m	最优解	TSSB 加工周期	TSSB t/s	HLS 加工周期	HLS t/s
FT6	6	6	55	55*	<1	55*	<1
FT10	10	10	930	930*	80	930*	138
FT20	20	5	1165	1165*	115	1165*	81
LA1	10	5	666	666*	9.8	666*	<1
LA2	10	5	655	655*	9.8	655*	<1
LA3	10	5	597	597*	9.8	597*	<1
LA4	10	5	590	590*	9.8	590*	<1
LA5	10	5	593	593*	9.8	593*	<1
LA6	15	5	926	926*	<1	926*	<1
LA7	15	5	890	890*	<1	890*	<1
LA8	15	5	863	863*	<1	863*	<1
LA9	15	5	951	951*	<1	951*	<1
LA10	15	5	958	958*	<1	958*	<1
LA11	20	5	1222	1222*	<1	1222*	<1
LA12	20	5	1039	1039*	<1	1039*	<1
LA13	20	5	1150	1150*	<1	1150*	<1
LA14	20	5	1292	1292*	<1	1292*	<1
LA15	20	5	1207	1207*	<1	1207*	<1
LA16	10	10	945	945*	61.5	945*	202
LA17	10	10	784	784*	61.5	784*	64
LA18	10	10	848	848*	61.5	848*	22
LA19	10	10	842	842*	61.5	842*	144
LA20	10	10	902	902*	61.5	902*	595
LA21	15	10	1046	1046*	115	1046*	1656
LA22	15	10	927	927*	115	927*	469
LA23	15	10	1032	1032*	115	1032*	<1
LA24	15	10	935	938	115	937	837
LA25	15	10	977	979	115	977*	245
LA26	20	10	1218	1218*	105	1218*	11
LA27	20	10	1235	1235*	105	1235*	1240
LA28	20	10	1216	1216*	105	1215*	101
LA29	20	10	1152	1168	105	1164	662
LA30	20	10	1355	1355*	105	1355*	1
LA31	30	10	1784	1784*	<1	1784*	<1
LA32	30	10	1850	1850*	<1	1850*	<1
LA33	30	10	1719	1719*	<1	1719*	<1
LA34	30	10	1721	1721*	<1	1721*	<1
LA35	30	10	1888	1888*	<1	1888*	<1
LA36	15	15	1268	1268*	141	1268*	2260
LA37	15	15	1397	1411	141	1397*	6624
LA38	15	15	1196	1201	141	1201*	2135
LA39	15	15	1233	1240	141	1235	2579
LA40	15	15	1222	1233	141	1229	1644

注：t/s 时间/秒。

4　计算结果与比较

为了测试 HLS 的性能，我们采用了43个著名的 Job-Shop 问题的实例作为测试实验集。所有的测试实例来自 OR-Library (http://mscmga.ms.ic.ac.uk/jeb/orlib/jobshopinfo.html)。算法用 C 语言编写，运行平台为 Intel 赛扬733。

HLS 与当前国外学者提出的一种著名的先进算法进行了比较，它是 Pezzella 的禁忌搜索结合移动瓶颈过程算法[8]

(下转第189页)

行更加深入的研究。

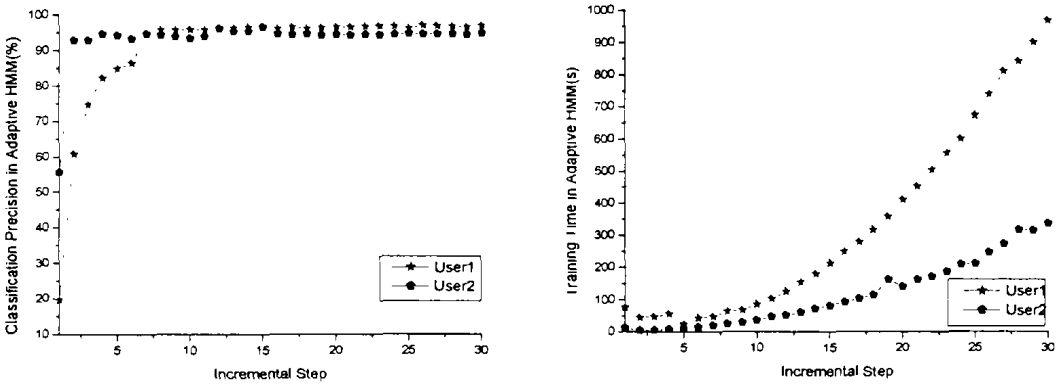

图8 自适应HMM用户适应效果

参考文献

1 Fish J, Scrivener S. Amplifying the mind's eye:Sketching and visual cognition [J]. Leonardo, 1990,23(1):117～126

2 孙正兴,徐晓刚,孙建勇,金翔宇. 支持方案设计的图形输入工具. 计算机辅助设计与图形学学报,2003,15(9):1145～1152,1159

3 Hu Jianying, Brown M K, Turin William. HMM Based On-Line Handwriting Recognition. IEEE Transactions on Pattern Analysis and Machine Intelligence,1996,18(10)

4 Lee J J, Kim J W, Jin H. Kim Data-driven Design of HMM Topology for On-line Handwriting Recognition. In: Proc. of 7th Intl. Workshop on Frontiers in Handwriting Recognition, Amsterdam, Sep. 2000

5 Nakai M, Akira N, Shimodaira H, Sagayama S. Substroke Approach to HMM-based On-line Kanji Handwriting Recognition. In: Sixth Intl. Conf. on Document Analysis and Recognition (ICDAR 2001), 2001.491～495

6 Calhoun C, Stahovich T F, Kurtoglu,et al. Recognizing Multi-Stroke Symbols. In: AAAI Spring Symposium, Sketch Understanding, 2002

7 刘伟,查建中,徐晓慧,郭明成. 用RCR特征和NN识别实时手绘工程草图. 计算机辅助设计与图形学学报,2003,15(6):692～696

8 Yang Jie, Xu Yangsheng. Hidden Markov Model for Gesture Recognition:[Technical Report of Robotics Institute]. Carnegie Mellon University, 1994

9 Rubine D. Specifying gestures by example. Computer Graphics, 1991,25:329～337

10 孙建勇,金翔宇,彭彬彬,孙正兴. 一种快速在线图形识别与规整化方法. 计算机科学,2003,25(12):172～176

(上接第180页)

(TSSB)。

表1给出了我们对OR-Library中所有FT和LA问题的计算结果。LA1-LA40来自Lawrence,其中LA2、LA19、LA21、LA24、LA25、LA27、LA29、LA36、LA37、LA38、LA39以及LA40被认为是这40个LA问题中最难的问题[7]。FT6、FT10以及FT20来自Fisher和Thompson。其中带*的表示算法找到了最优解。对于3个LA问题,HLS都找了最优解,其中包括著名的难例FT10,对于40个LA问题,HLS找到了其中36个问题的最优解,TSSB找到了其中33个问题的最优解,对于HLS未找到最优解的其余4个问题,HLS的计算结果都好于TSSB。

从表1我们看出,在这43个不同规模大小和难度的问题上,HLS在可接受的时间内取得了很好的求解效果。

TSSB[8]是在Pentium 133 MHz的PC机上做的实验,对每个LA算例TSSB的计算时间为同规模大小的同组算例的的平均计算时间。

结论 本文提出的混合邻域结构的本质就是不同的邻域结构的组合不但能使局部搜索具有更高的效率,而且在计算陷入局部极小值陷阱时,有助于使计算跳出陷阱,走向前景更好的区域中去。在混合邻域结构的基础上,本文提出的跳坑策略的实施也是本文算法取得令人满意的求解效果的直接原因。

对43个不同规模和难度的标准算例的实算结果以及与当前一种先进算法的比较验证了本文算法的寻优性能。

参考文献

1 Garey M, Johnson D, Sethy R. The complexity of flow shop and job shop scheduling. Mathematics of Operations Research, 1976, 1(1):117～129

2 Aarts E H L, van Laarhoven P J M, Lenstra J K, Ulder N L J. A computational study of local search algorithms for job shop scheduling. ORSA J. Computing, 1994, 6: 108～117

3 Adams J, Balas E, Zawack D. The shifting bottleneck procedure for job shop scheduling. Management Sci., 1988,34:391～401

4 Nowicki E, Smutnicki C. A fast taboo search algorithm for the job shop scheduling problem. Manabement Sci.,1996,42(6):797～813

5 van Laarhoven P J M, Aarts E H L, Lenstra J K. Job shop scheduling by simulated annealing. Oper. Res.,1992,40:113～125

6 Dorndorf U, Pesch E. Evolution based learning in a job shop scheduling environment. Comput. Oper. Res.,1995,22:25～40

7 Balas E, Vazacopoulos A. Guided local search with shifting bottleneck for job shop scheduling. Managemeng Sci.,1998,44:262～275

8 Pezzela F, Merelli E. A tabu search method guided by shifting bottleneck for the job shop scheduling problem. European Journal of Operation Research, 2000,120(2):297～310

9 Carlier J. The one-machine sequencing problem. European J. Oper. Res., 1982,11: 42～47

10 Aarts E, Lenstra J K. Local search and combinatorial optimization. New York: Wiley, 1997

11 王书锋,邹益仁. 车间作业调度(JSSP)技术问题简明综述. 系统工程理论于实践, 2003,1