BAB 算法中集成 CPT 求解 job-shop 调度问题①

冯 欣, 唐立新, 王梦光
(东北大学信息科学与工程学院, 沈阳 110004)

摘要: CSP (constraint satisfactory problem) 的优势在于能够处理复杂约束,获得一个满足约束的解,但难以保证解的质量. OR (operation research) 的优点是获得最优解或近优解,但它求解复杂约束的优化问题非常困难. CPT (constraint propagation technique) 是 CSP 的主要搜索技术,BAB (branch-and-bound) 是 OR 常用的优化算法. 提出了一种将 CPT 集成于 BAB 中的混合算法,从一个新的角度解决具有一般性与挑战性的 job-shop 调度问题. 其主要特点是,通过在 BAB 算法中嵌入动态可调的时间窗口约束和加强一致性 CPT 搜索方法,融合 BAB 的优化能力和 CPT 处理复杂约束的能力,提高 BAB 的优化性能及实际应用能力. 实验结果令人满意,证明了算法的有效性.

关键词: 约束传播技术; 分支定界算法; job-shop 调度
中图分类号: TP29 **文献标识码**: A **文章编号**: 1007 - 9807(2005)03 - 0081 - 09

0 引 言

调度问题大致可以分为两个研究领域[1]: 人工智能(AI)和运筹学(OR). CSP 是 AI 的主要研究方法,它采用有效的变量排序、值排序和 CPT 搜索策略,目的是为各种一般性的问题尽快得到一个满足所有约束的解. 由于 CSP 方法具有处理实际问题中各种复杂约束的良好柔性,近年来越来越引起各国学者的重视[2,3]. 不足的是在求解过程中没有确定的评价函数与搜索参数,因而难以保证解的质量[4]. OR 研究大多基于数学模型,以最优化技术为核心,通过开发问题的组合结构来提高求解效率. BAB 是 OR 中经典的精确算法,对小规模优化问题有效,但面对组合爆炸、复杂问题建模、动态响应、应用领域等问题的挑战束手无策[5],因而对 BAB 的研究仍然停留在推导复杂的分支模式与改进最小下界方面.

本文提出在 BAB 中集成 CPT 的混合算法,从新的角度解决具有一般性与挑战性的 job-shop 调度问题. 通过在 BAB 算法中嵌入动态可调的时间窗口约束和加强一致性 CPT 搜索方法,融合 BAB 的优化能力和 CPT 处理复杂约束的能力,用简单易行的搜索结构提高 BAB 的搜索效率及实际应用能力,为 AI 与 OR 领域的混合研究提供成功的范例.

1 BAB 与 CPT 的基本思想

1.1 Job-shop 问题描述

Job-shop 调度问题是最困难的组合优化问题之一,是典型的 NP-hard 问题,它作为包括多种调度问题的一般模型而著称,常用来检验各种求解方法的性能[5,6]. 加工车间的大多数调度问题可以公式化为 job-shop 调度问题,描述如下: 考虑 n 个工件 $J_1, \cdots, J_n$ 及 m 个不同的机器 $M_1, \cdots, M_m$, 其中每个工件 J_i 包括 n_i 道工序 $O_{i1}, \cdots, O_{in_i}$, 各工序加工顺序已知,工序 O_{ik} 仅可以由机器 μ_{ik} ($i = 1, \cdots, n$; $k = 1, \cdots, n_i$) 加工,机器始终有效, p_{ik} 表示工序 O_{ik} 在机器 μ_{ik} 上的加工时间,且一台机器一次只加工一道工序,同一工件不能同时被两台机器加工,调度目标通常为最大完工时间最小化,即在每台机器 M_j 上,为所有的工序 O_{ik} ($\mu_{ik} = M_j$) ,找到可行的加工顺序,使得所有工件的最大完工时间 $C_{\max}$ 最小.

① 收稿日期: 2003 - 05 - 09; 修订日期: 2005 - 03 - 17.
基金项目: 国家自然科学基金资助项目(70171030; 60274049); 高等学校优秀青年教师教学科研奖励计划资助项目(教人司[002]383); 霍英东青年教师基金资助项目(81073).
作者简介: 冯 欣(1965 —),女,博士生.

$$C^*_{\max} = \min(C_{\max}) = \min_{\text{feasible schedules}}(\max(C_i): \forall_i \in J) \quad (1)$$

1.2 活动调度与 BAB 算法

最大完工时间最小化的 job-shop 调度问题,最优调度一定在活动调度集合中[7]. 活动调度是没有工序在不延迟其它工序的条件下而能够提前开工的可行调度,其数量随问题规模的增大成指数级别增长,因此寻遍活动调度集合得到最优解十分困难. BAB 算法,即一种深度优先带有回溯的树搜索方法,通过部分列举活动调度的方式求出最优调度[5]. 首先采用树形结构构造活动调度,树中的节点对应部分调度,节点的每个分支对应同一层次有冲突的工序,即竞争同一机器的工序,树的叶子是活动调度的集合. 令 P 为部分调度集合,S 为各工件中无紧前工序或紧前工序已调度完的工序集合,对任何 S 中的工序 i, $T_m(i)$ 为所需机器有效的时间, $T_p(i)$ 为紧前工序的最大完工时间, p_i 为加工时间, est_i 为最早开始时间, eft_i 为最早完工时间,且 $est_i = \max_{i \in S}\{T_m(i), T_p(i)\}$, $eft_i = est_i + p_i$,则生成活动调度的流程如下:

1) 初始化 P 为空集, S 为各个工件的第一道工序
2) repeat
3) $eft_{i^*} \leftarrow \min_{i \in S}\{eft_i\}$ 选取工序 i 及加工 i 的机器 M^*
4) 根据指定的优先规则选取工序 j, $j \in S$ 满足 $M_j = M^*$,且 $est_j < eft_{i^*}$
5) 将工序 j 从 S 移到 P, j 的紧后工序放入 S
6) until S 为空

在步骤 4) 中,可能包括多个工序,即若干个分支,对这多个工序的选择按照最大流程时间 $F_{\max}$ 的最小下界进行. 在到达一个叶子后,回溯遍历所有可行的节点,获得最优解. 最大流程时间的最小下界的确定方法:

$$B_1(S_t) = \max_{1 \le i \le n}\{T_i + R_i\} \quad (2)$$

$$B_2(S_t) = \max_{1 \le k \le m}\{U_k + V_k\} \quad (3)$$

$$B_{12}(S_t) = \max\{B_1(S_t), B_2(S_t)\} \quad (4)$$

其中: $B_1(S_t)$, $B_2(S_t)$ 分别表示在部分工序的活动调度 S_t 下按工件及按机器确定的 $F_{\max}$ 下界;对工件 i, T_i 为可安排工序的最早可能开始时间, R_i 为未安排工序的加工时间之和;在机器 M_k 上, U_k 为加工可安排工序的最早可能开始时间, V_k 为未安排工序的加工时间之和. 得到混合下界 $B_{12}(S_t)$.

1.3 CPT 策略

CPT 是一种用于缩减组合搜索与优化问题的搜索空间的方法,其基本思想是通过重复地分析与评价变量、定义域以及描述指定问题实例的若干约束,来检测并移去不能加入任何可行解中的不一致的变量赋值[1].

在 CSP 中,不同的文献对 job-shop 调度问题有着不同的描述形式[2,4],一般是将每道工序的开始时间 st_i 作为变量集,每个变量对应一个可取值集合,求解过程就是为每个变量赋值,并满足下面的约束:

1) 优先级约束 $st_i + p_i \le st_j$, i 与 j 为同一工件内,任意具有优先级关系 $i \rightarrow j$ 的工序对, p_i 为 i 的加工时间;

2) 机器能力约束 $st_i + p_i \le st_j \vee st_j + p_j \le st_i$, $i \leftrightarrow j$, i 与 j 为任意两个需求同一机器的工序;

3) 到达时间与交货期约束 $r_J \le st_i$, $st_i + p_i \le d_J$, r_J 与 d_J 分别为工件 J 的到达时间与交货期.

CPT 的作用就是保证连续的瞬时一致性. 例如,如果选取了 $i \rightarrow j$,则修正 est_j 与 lft_i 为 $est_j = \max\{est_j, est_i + p_i\}$ 与 $lft_i = \min\{lft_i, lft_j - p_j\}$. 在传播过程中,如果某个工序 k 有 $est_k + p_k > lft_k$,则检验到了不一致,说明当前决策不可行. 继续检验其它可能的选择,直到找到一个解或证明该问题无可行解. 文献[8]验证了在活动调度中嵌入时间窗口的方法可以有效地获得一个可行解.

2 在 BAB 中集成 CPT 的算法

本文提出的 CPT-integrated BAB 算法适合解决一般性的调度问题,但为了与 BAB 比较,以验证算法的有效性,假定所有的工件具有相同的 $r_J = 0$ 及 d_J,开始时间变量 st_i 取整数值. 另外,为使最大完工时间最小化,每道工序应尽可能早地被调度,在实例化开始时间变量时,首选取时间窗口中最小的值. 下面结合表 1 中实例描述算法工作过程.

表 1 一个简单的 job-shop 调度问题实例

Table 1 A simple example of job-shop scheduling problem

Job	工序 1		工序 2		工序 3	
	加工时间	机器	加工时间	机器	加工时间	机器
1	7	1	3	3	9	2
2	4	1	3	2	3	3
3	9	3	2	1	9	2

2.1　带有时间窗口的 BAB 搜索树的构造

本文提出了在搜索树构造过程中,时间窗口上、下界动态地异步生成及修正的方法,以构造带有时间窗口的 BAB 搜索树:

1) 松弛机器能力约束,在满足工件 r_J 的条件下,应用 CPT 计算 J 的每道工序 i 的 est_i 作为时间窗口下界. 按照 BAB 的定界方法,深度优先到达一个叶子,得到一个活动调度的最大完成时间 C_1;

2) 松弛机器能力约束,在满足工件 $d_J = C_1 - 1$ 的条件下,应用 CPT 计算 J 的每道工序 i 的 lst_i 作为时间窗口上界,est_i 与 lst_i 之间可能的取值作为工序 i 的初始时间窗口,对应生成开始时间变量 st_i 的可取值域;

3) 每扩展一个分支节点,首先应用 CPT,修正未调度工序开始时间变量的时间窗口下界,削减不可行的取值,同时进行一致性检查. 对通过了检查的节点,计算该节点的最大流程时间最小下界,削去不一致的节点;

4) 每到达一个新的叶子 C_l,重新应用 CPT 计算 J 的每道工序 i 的 lst_i 修正时间窗口上界,直到完成对所有有效节点的遍历,得到最优调度.

用 J_i-O_j-M_k 表示工件 i 的第 j 道在机器 k 上加工的工序,图 1 给出了实例最优调度部分的搜索树分支节点生成情况. 在整个搜索过程中,BAB 经历了 9 次回溯,CPT-integrated BAB 为 6 次.

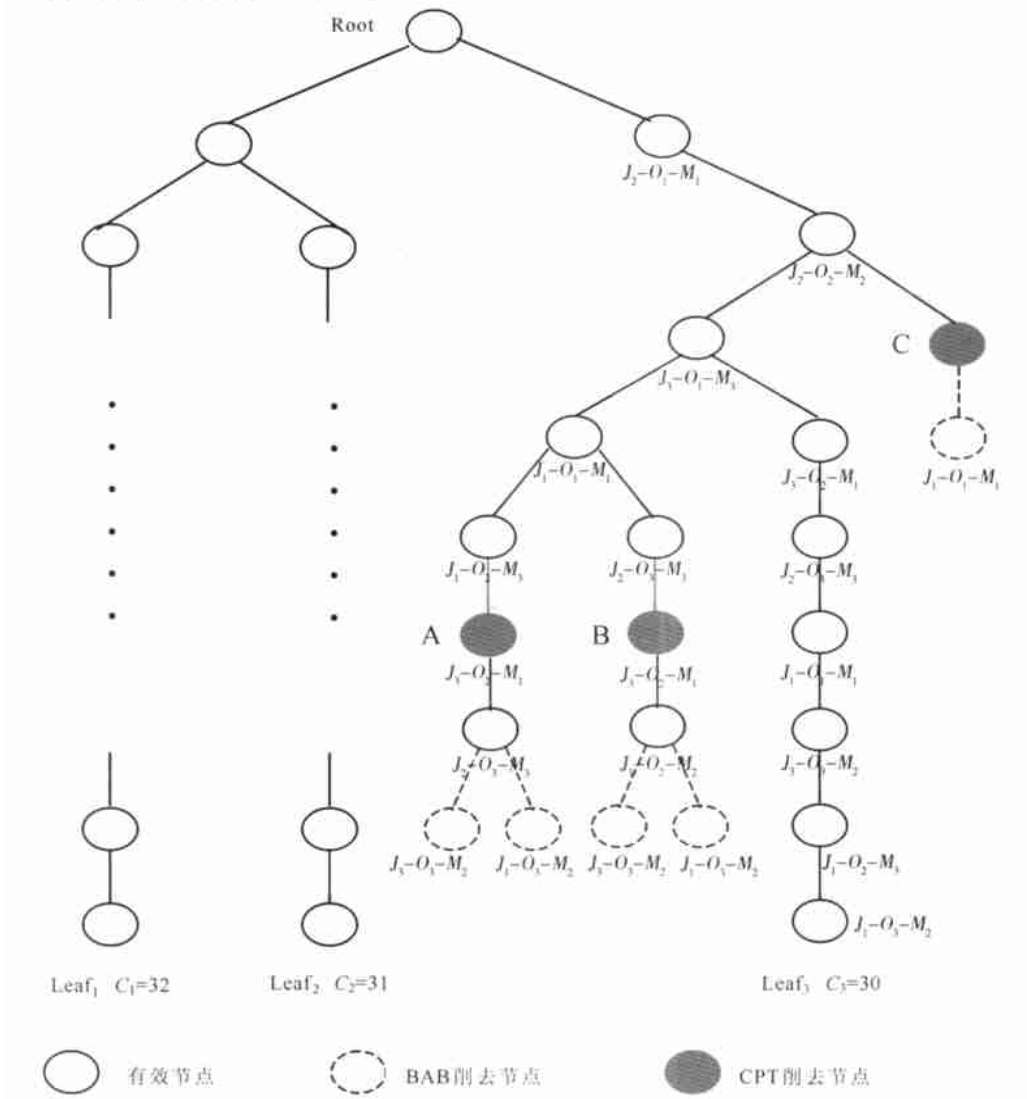

图 1　最优调度部分搜索树

Fig. 1 Partial search tree of optimal schedule

2.2 搜索树中的瞬时 CPT

构造搜索树的每个步骤,都要应用 CPT. 采用动态加强弧一致前向检查确定不一致性,令 LB_i、UB_i 分别表示工序 $i(i = 1, \cdots, n)$ 的开始时间窗口的下界与上界,一旦选定了被实例化的变量,并赋值为该变量的时间窗口的 LB,立即执行瞬时约束传播.

实例中各工序的约束关系见图 2. 为满足优先级约束进行的传播,可修正与被实例化变量 st 相应的工序所属的工件内,所有未被实例化工序的开始时间变量的 LB;为满足机器约束进行的传播,可修正与被实例化变量 st 相应的工序使用同一台机器的所有未被实例化工序的开始时间变量的 LB. 过程见图 3、图 4.

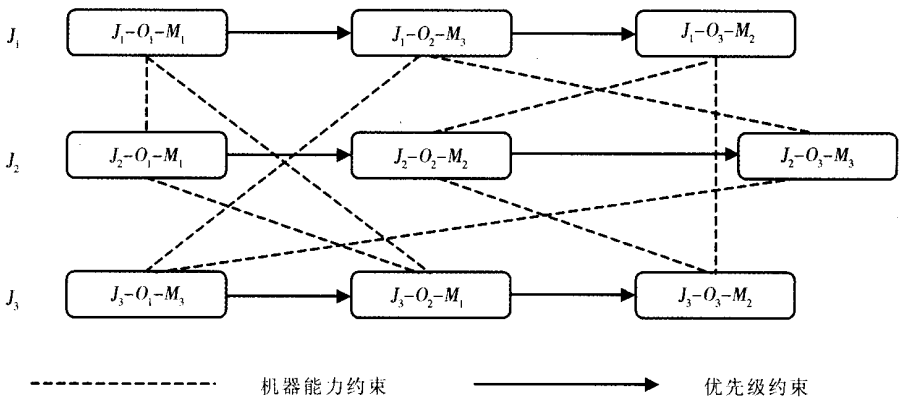

图 2 约束关系

Fig. 2 Constraints connection

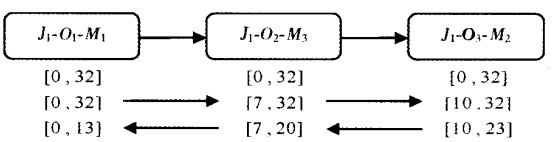

图 3 优先级约束一致性传播

Fig. 3 Consistency propagation of precedence constraints

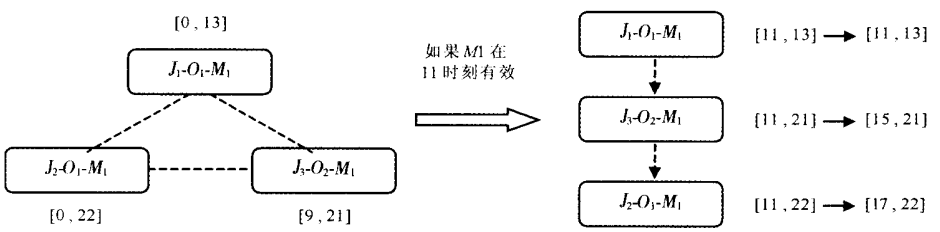

图 4 机器能力约束一致性传播

Fig. 4 Consistency propagation of machine capacity constraints

这些约束的传播起两个方面的作用:一是滤掉互相约束着的相关变量值域中的一些无效值;一是检验一致性,如有某个窗口 $LB_i > UB_i$,则表明产生了不合理的时间窗口,说明实例化该变量将产生不一致,则削去该节点及其下面所有的分支,从而达到削减搜索空间的目的.

2.3 瞬时 CPT 削支过程的实现

搜索过程中,每实例化一个工序的开始时间变量 st_i,即到达一个新的搜索状态. 需要保存有效节点的状态信息以备回溯. 随着不断到达搜索树的新叶子,$F_{\max}$ 的最小下界被逐步下调,各变量的时间窗口减小,变量间的约束缩紧,CPT 在搜索过程中的作用越来越显著. 在回溯到某个保存的节点时,根据当前的搜索情况,重新评价该节点,确定该节点的取舍.

表 2　被削去节点的变量取值及窗口状态

Table 2 Variables values and windows states of cut nodes

Note	下界	J_1-O_1-M_1	J_1-O_2-M_3	J_1-O_3-M_2	J_2-O_1-M_1	J_2-O_2-M_2	J_2-O_3-M_3	J_3-O_1-M_3	J_3-O_2-M_1	J_3-O_3-M_2
A	25	4	11	[14,21]	0	4	[14,27]	0	11	[13,21]
B	25	4	[12,18]	[14,21]	0	4	9	0	11	[13,21]
C	29	[4,10]	[7,17]	[10,20]	0	4	7	[0,9]	[9,18]	[11,20]

表 2 是图 1 中三个被 CPT 削去的节点的状态信息. 如 A 节点,实例化工序 J_3-O_2-M_1 的开始时间变量为 11,约束传播使 J_3-O_3-M_2 的时间窗口由{[11,21]}成为{[13,21]}. 此时, 由于工序 J_3-O_3-M_2 与 J_1-O_3-M_2 共同竞争 M_2,从表 1 得知,两者的加工时间均为 9,无论先实例化哪道工序,都造成另一工序的开始时间窗口为空集,所以 A 节点将被削去. 而 BAB 算法在 A 节点的 F_{max} 下界为 25,小于当前最好的解 31,仍需继续扩展 A 节点.

时间窗口的更新与修正,计算简单易行,几乎不增加计算的负担. LB_i 修正计算复杂性仅为 $O(l)$, l 为未被实例化的变量个数, UB_i 更新只有在到达了新的叶子时才进行. 图 5 给出实例的最优调度结果.

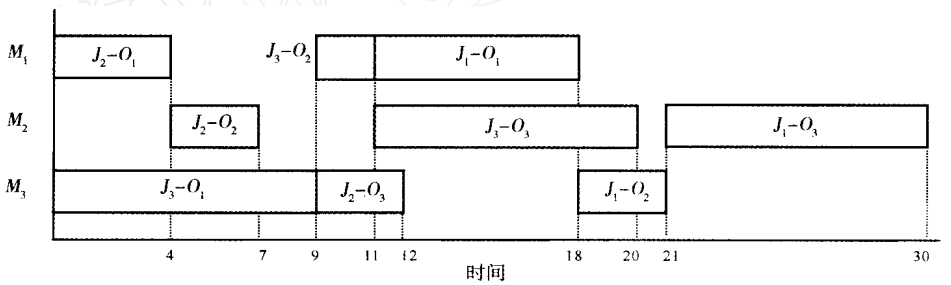

图 5　最优调度

Fig. 5 The optimization schedule

2.4　算法流程

令 *INFOR* 表示分支节点状态信息集合, *Lbf* 表示最大流程时间下界, *Ubc* 表示最大完工时间上界, p 表示部分调度集合,其它表示与前面提到的相同. CPT-integrated BAB 算法流程如下:

1) 初始化 P 与 *INFOR* 为空, S 为每个工件的第一道序,生成初始时间窗口 LB_i
2) repeat
3) 在 S 中选择实例化的工序 j,将 k 的状态信息存入 *INFOR* 集合, $k \in S$, $k \neq j$
4) 更新 S,将 j 从 S 移到 P,将 j 的紧后工序放入 S
5) 由 CPT 修正未实例化变量的 LB
6) until 到达搜索树的一个叶子,计算 C_{max}
7) 生成初始时间窗口 UB_i
8) repeat
9) $Ubc = C_{max} - 1$,更新 UB_i
10) repeat
11) 按后入先出方式取出 *INFOR* 中信息,经 CPT 检验,直到取出的节点 l
12) 满足 $Lbf_l \leq Ubc \cap LB_l \leq UB_l$ 并扩展该节点
13) until 到达搜索树的一个新叶子
14) until 获得最优解,或达到指定的计算时间、回溯次数

3　实验结果与分析

根据 Taillard[9] 提出的数据生成方法,随机产生了 10 种规模,每种规模 10 例数据,共 100 例数据进行实验,将 CPT-integrated BAB 算法与 BAB 比较. 其中 5 种选用小规模问题,比较最优情况下的实验结果,见表 3. 另外 5 种复杂问题则比较在规定的计算时间内解的情况,见表 4. 相应的时间种子(time seed)与机器种子(machine seed)一并列入结果数据表. 实验在 P4 处理器、主频为 1.8 G 的 PC 机上运行.

表中, n 为工件数, m 为机器数,运行时间以 s 计(这里选用 1 000 s),列"SL*"表示最优解,"SL"表示在限定时间内所得到的最好解,其中有"*"标志的数据为最优解,列"CPU"与"BT"分别给出了得到最优解或最好解所用的 CPU 时间及所经历的回溯次数.

表 3 小规模问题获得最优解性能

Table 3 The performance to obtain optimal schedule for small size problems

n/ m	Time Seed	Machine Seed	SL *	BAB		CPT-integrated BAB	
				CPU	BT	CPU	BT
5/ 5	841 613 311	1 524 105 215	354	0.047	145	0.000	51
	1 618 870 271	232 128 511	410	0.015	76	0.000	20
	1 400 831 999	739 704 831	410	0.000	30	0.000	23
	2 112 028 671	804 519 935	520	0.063	342	0.000	24
	1 948 123 135	1 540 751 359	361	0.032	208	0.015	64
	857 931 775	849 084 415	319	0.063	217	0.000	41
	2 072 903 679	1 273 954 303	389	0.016	74	0.015	27
	999 817 215	1 061 355 519	368	0.031	76	0.000	34
	2 037 317 631	1 497 104 383	409	0.016	43	0.000	27
	1 066 401 791	602 079 231	474	0.063	171	0.015	67
6/ 6	126 681 087	1 560 281 087	442	0.516	2 391	0.438	2 143
	898 170 879	375 848 959	474	0.219	881	0.078	291
	728 104 959	1 932 394 495	502	1.750	8 422	0.312	1 477
	1 206 124 543	1 700 659 199	472	0.625	3 009	0.484	2 092
	1 224 867 839	1 495 269 375	403	0.078	306	0.047	207
	1 376 583 679	1 181 679 615	529	3.765	18 962	0.657	2 807
	223 543 295	1 338 834 943	541	0.140	507	0.047	143
	1 733 689 343	694 747 135	560	0.234	747	0.094	314
	1 022 230 527	1 542 062 079	638	0.813	3 519	0.062	233
	1 702 559 743	1 354 170 367	543	0.750	3 340	0.016	63
7/ 7	1 624 244 223	994 115 583	677	20.359	89 288	1.437	5 476
	802 815 999	399 507 455	622	1.688	6 125	0.219	678
	50 069 503	558 366 719	583	29.000	132 013	9.359	34 054
	1 081 671 679	623 575 039	536	17.890	63 043	1.797	7 537
	745 799 679	1 479 999 487	521	1.938	8 036	0.500	1 951
	411 172 863	1 669 464 063	661	78.297	344 485	3.735	13 701
	808 845 311	1 347 878 911	542	10.547	38 571	3.359	10 685
	1 865 220 095	55 902 207	681	2.906	11 193	1.109	3 909
	61 472 767	2 104 885 247	593	2.922	9 303	0.781	2 357
	147 390 463	746 979 327	596	7.313	25 292	2.984	9 672
8/ 7	658 767 871	1 460 404 223	641	21.766	83 469	2.984	8 322
	1 850 802 175	1 558 773 759	569	22.563	86 223	5.531	19 075
	2 021 457 919	557 645 823	613	27.688	113 386	3.656	14 039
	811 859 967	71 041 023	666	136.078	577 989	14.781	45 749
	562 167 807	1 014 890 495	672	259.953	1 124 142	18.641	68 972
	617 218 047	1 240 662 015	541	30.485	136 950	4.547	16 789
	613 482 495	650 969 087	576	128.968	471 729	70.829	241 480
	1 437 073 407	52 297 727	622	307.266	1 798 854	0.625	2 938
	2 086 666 239	1 189 019 647	597	68.734	280 836	13.594	52 219
	855 769 087	970 915 839	647	254.172	1 080 283	4.296	17 792
8/ 8	1 383 202 815	974 979 071	696	33.656	127 983	8.063	26 821
	1 017 839 615	675 414 015	598	204.047	579 141	44.000	153 760
	1 277 558 783	2 147 287 039	644	186.625	564 913	15.406	41 357
	1 599 471 615	473 628 671	645	17.140	56 752	3.438	9 409
	1 087 963 135	1 239 613 439	690	109.359	396 305	26.610	88 520
	38 273 023	1 319 043 071	786	141.828	581 588	11.000	36 140
	689 373 183	962 658 303	710	183.016	815 263	6.359	20 416
	823 263 231	1 842 610 175	698	142.218	469 497	24.203	77 547
	819 396 607	381 878 271	644	581.594	2 440 728	48.203	165 947
	1 492 516 863	1 727 987 711	695	62.109	233 767	23.672	59 742

表 4　复杂问题在规定时间内求解性能

Table 4　The performance for complex problems in pre-designated time

n/ *m*	Time Seed	Machine Seed	BAB			CPT-integrated BAB		
			SL	CPU	BT	SL	CPU	BT
10/ 10 1 000 s	343 474 175	388 562 943	1 023	998.91	3 239 206	999	820.02	2 469 809
	331 218 943	344 915 967	775	974.84	3 258 830	727	953.08	3 365 959
	481 558 527	939 130 879	840	442.89	1 266 669	840	140.25	410 036
	273 219 583	1 487 208 447	817	12.64	40 056	808	340.70	889 607
	186 712 063	1 966 014 463	886	899.64	2 909 190	882	65.33	180 643
	201 916 415	327 483 391	923	772.02	3 332 448	878 *	111.27	315 504
	1 846 280 191	1 764 884 479	814	730.66	2 264 920	801	792.27	2 302 495
	752 615 423	1 874 198 527	915	401.03	1 402 164	900	364.75	1 229 900
	555 220 991	547 880 959	912	1.20	2 571	852	832.30	2 171 829
	1 985 871 871	1 614 872 575	784	164.77	560 882	776	832.38	3 036 413
20/ 10 1 000 s	378 863 615	1 913 520 127	1 510	309.75	748 057	1 504	165.72	420 707
	1 216 610 303	708 116 479	1 642	15.42	30 662	1 642	4.60	8 312
	1 122 631 679	2 090 336 255	1 501	168.66	300 803	1 481	231.83	461 958
	1 781 989 375	520 486 911	1 514	360.95	876 853	1 439	583.83	1 257 602
	1 426 325 503	160 563 199	1 539	495.59	1 109 829	1 513	647.06	1 356 782
	736 821 247	2 051 342 335	1 647	610.05	1 081 972	1 647	132.14	207 452
	1 286 668 287	1 026 686 975	1 499	714.34	1 728 559	1 493	376.03	973 410
	1 453 850 623	167 313 407	1 423	928.84	2 469 286	1 410	282.89	504 789
	266 469 375	1 319 829 503	1 686	615.33	1 651 074	1 649	111.83	273 117
	1 683 816 447	228 982 783	1 465	594.22	1 357 991	1 455	494.66	901 744
30/ 10 1 000 s	1 796 931 583	2 117 468 159	2 270	614.70	863 880	2 270	105.30	154 630
	194 772 991	1 365 835 775	1 957	531.16	968 094	1 957	83.80	144 339
	1 762 525 183	991 625 215	2 058	897.13	1 781 302	1 981	949.31	1 343 316
	1 527 119 871	1 962 213 375	2 107	407.39	720 917	2 100	150.95	320 045
	327 352 319	1 675 558 911	1 992	344.34	601 442	1 992	175.23	243 787
	1 775 828 991	1 759 772 671	2 062	485.55	972 431	2 033	985.09	1 542 294
	481 951 743	1 813 250 047	2 342	6.25	10 186	2 289	54.19	94 322
	1 696 923 647	16 121 855	2 103	785.41	1 275 262	2 103	78.73	126 533
	513 540 095	2 145 189 887	2 012	396.22	662 341	2 009	630.86	1 026 590
	1 026 818 047	738 590 719	2 129	696.22	1 067 883	2 111	993.94	1 707 959
30/ 15 1 000 s	1 345 454 079	549 715 967	2 429	450.84	396 531	2 418	201.48	200 023
	1 040 908 287	1 194 655 743	2 440	0.13	43	2 429	309.28	314 569
	1 160 773 631	1 245 970 431	2 297	704.58	854 048	2 259	532.94	485 799
	1 080 950 783	522 518 527	2 339	3.92	4 836	2 334	388.09	595 982
	1 314 652 159	1 053 032 447	2 451	518.06	572 276	2 435	91.45	121 892
	613 285 887	1 977 090 047	2 504	260.95	402 097	2 455	964.73	1 265 832
	325 255 167	2 139 553 791	2 585	110.50	107 862	2 585	35.14	30 643
	1 881 604 095	1 986 265 087	2 490	99.86	84 514	2 484	895.13	697 019
	571 473 919	427 294 719	2 447	492.13	612 448	2 407	727.41	814 000
	639 238 143	1 636 696 063	2 359	0.09	9	2 359	0.11	9
50/ 15 1 000 s	153 747 455	1 812 463 615	3 839	8.03	5 194	3 778	5.83	3 926
	173 408 255	740 163 583	3 629	346.59	266 187	3 594	998.92	588 562
	1 909 260 287	1 643 839 487	3 497	149.23	105 646	3 480	108.11	80 288
	1 736 835 071	792 133 631	3 631	526.97	409 488	3 624	251.03	206 317
	350 093 311	737 935 359	3 740	6.73	4 296	3 699	311.91	259 160
	1 306 918 911	573 177 855	3 578	144.95	139 421	3 522	516.97	336 466
	408 551 423	1 517 813 759	3 477	249.16	176 178	3 477	25.97	16 702
	1 634 729 983	586 481 663	3 692	224.81	258 128	3 668	875.66	704 448
	1 202 257 919	7 579 238 393	3 792	618.39	463 516	3 792	95.61	68 537
	828 047 359	702 808 063	3 737	69.08	45 734	3 731	240.66	137 114

观察表 3 中的数据,所有的实例采用 CPT-integrated BAB算法获得最优解时的计算时间与回溯次数都比 BAB 有明显的改善. 表 4 中 50 个实例中有 40 个实例在解值上有所改善,另外 10 个实例则在获得相同的解时,计算时间与回溯次数明显减少. 表 5 给出了各种规模问题获得最优解或相同解的计算时间与回溯次数平均值间的比较. 图 6、图 7 分别显示了平均值间的对数比较关系.

表 5 获得最优解或相同解的计算时间与回溯次数平均值

Table 5 The average values of calculating time and track backing times when optimization or same solution are obtained

n/ *m*	BAB		CPT-integrated BAB	
	CPU(avg)	BT(avg)	CPU(avg)	BT(avg)
5/5	0.035	138	0.005	38
6/6	0.889	4 208	0.224	977
7/7	16.386	72 735	2.528	9 002
8/7	125.767	575 386	13.948	48 738
8/8	166.159	626 594	21.059	67 966
10/10	539.86	1 827 694	45.09	132 243
20/10	481.32	1 135 509	45.29	88 280
30/10	516.44	892 374	70.92	114 549
30/15	264.11	303 466	48.23	54 008
50/15	234.40	187 379	32.25	21 043

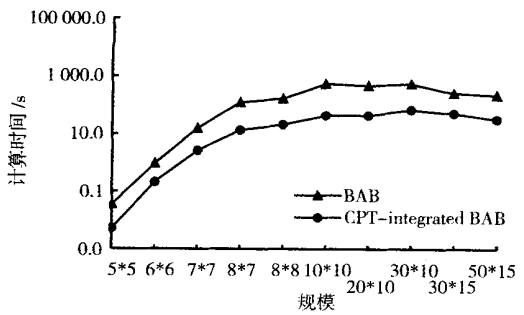

图 6 获得最优解或相同解的计算时间平均值对数比较

Fig. 6 The compared logarithm results of average values of calculating time when optimization or same solution are obtained

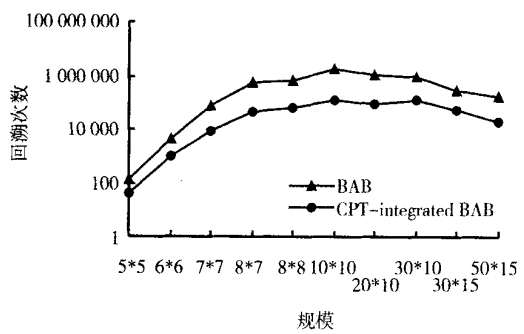

图 7 获得最优解或相同解的回溯次数平均值对数比较

Fig. 7 The compared logarithm results of average values of track backing times when optimization or same solution are obtained

在图 6 与图 7 中,前半部分由于问题都已得到了最优解,曲线随问题规模的扩大呈上升趋势;后半部分问题规模越大,从一个叶子到达另一个叶子需耗费更多的计算时间,使得在限定的计算时间内获得最好解的计算时间与回溯次数曲线下降. 经计算,100 个实例总的 CPU 平均缩短 9.48 倍,BT 平均缩小 8.60 倍. 将 CPT-integrated BAB 算法对 BAB 算法的平均改进率列于表 6,总的 CPU 平均改进率为 78.8 %,BT 平均改进率为 79.6 %. 同样由于计算时间的限制,SL 的平均改进率随问题规模增大而减小.

表 6 CPT-integrated BAB 算法对 BAB 算法的平均改进率

Table 6 The average ratio which CPT-integrated BAB algorithm versus BAB algorithm (%)

n/ *m*	SL(avg)	CPU(avg)	BT(avg)
5/5	—	73.6	62.2
6/6	—	62.3	62.9
7/7	—	76.9	79.1
8/7	—	83.9	85.7
8/8	—	82.7	84.8
10/10	2.58	86.2	87.4
20/10	1.26	88.6	89.1
30/10	0.65	83.4	84.9
30/15	0.73	65.7	73.2
50/15	0.67	84.2	87.1

CPT-integrated BAB 算法基于 BAB 搜索树,因此最坏的情况是混合算法没能为 BAB 削去任何的分支,由于时间窗口的加入与 CPT 进行的窗口修正、更新,几乎不增加计算的负担,也可以得到与 BAB 同样的计算结果,这从实验结果可以证实. 另外,随着问题规模的增大,BAB 中的分支数量呈指数级增长,即使混合算法可以削去大量的分支,也需要访问众多的节点,使短时间内对解的改进率不会很大.

可以得出这样的结论:CPT 可以集成到任何其它改进了下界与分支模式的更高级的 BAB 算法中发挥作用,获得更好的结果.

4 结束语

本文提出了在 BAB 中集成 CPT 求解调度问题的 CPT-integrated BAB 算法,并给出了如何在调

度过程中通过集成 CSP 与 BAB 各自的优势提高原有方法的求解性能及求解复杂问题的实现方法,即通过在 BAB 中加入时间窗口,在 BAB 搜索树的每个节点采用动态加强弧一致前向检查确定不一致,减少搜索空间,提高算法实用性.

本文检验了算法在求解具有最大完工时间最小化的 job-shop 调度问题时的性能,结果显示:

1) 算法削减了大量的搜索树分支,减少了搜索空间,加速了搜索过程,改善了搜索效率;

2) 时间窗口的加入增强了算法解决实际问题中各种约束的能力,提高了柔性与实用性;

3) 简单的搜索结构既容易实现又不增加计算负担;

4) 算法为提高传统优化算法的性能及应用能力提供了一个新的角度,值得在理论上及实际中得到进一步的研究.

参 考 文 献:

[1] Baptiste P, Pape C L, Nuijten W. Incorporation Efficient Operations Research Algorithms in Constraint-based Scheduling[R]. ILOG S A, 1996. 1—12.

[2] Nuijten W P M, Aarts E H L. A computational study of constraint satisfaction for multiple capacitated job shop scheduling[J]. European Journal of Operational Research, 1996, 90: 269—284.

[3] Varela R, Vela C R, Puente J, *et al*. A knowledge-based evolutionary strategy for scheduling problems with bottlenecks[J]. European Journal of Operational Research, 2003, 145: 57—71.

[4] Brilsford S C, Potts C N, Smith B M. Constraint satisfaction problem: Algorithms and applications[J]. European Journal of Operational Research, 1999, 119: 557—581.

[5] Brucker P, Jurisch B, Sievers B. A branch and bound algorithm for the job-shop scheduling problem[J]. Discrete Applied Mathematics, 1994, 49: 107—127.

[6] Blazewicz J, Domschke W, Pesch E. The job-shop scheduling problem: Conventional and new solution techniques[J]. European Journal of Operational Research, 1996, 93: 1—33.

[7] Baker K. Introduction to Sequencing and Scheduling[M]. New York: John Wiley & Sons, 1974.

[8] Chu W W, Ngai P H. Embedding temporal constraint propagation in machine sequencing for job shop scheduling[J]. AI EDAM, 1993, 7(1): 37—52.

[9] Taillard E. Benchmarks for basic scheduling problems[J]. European Journal of Operational Research, 1993, 64: 278—285.

Integrating CPT into BAB algorithm for job shop scheduling

FENG Xin, *TANG Li-xin*, *WANG Meng-guang*

Department of Systems Engineering, Northeastern University, Shenyang 110004, China

Abstract: CSP (constraint satisfactory problem) has the advantage of solving complicated constraints to obtain satisfying solutions, but does not guarantee the quality of the solution. In contrast, OR (operation research) has the advantage of obtaining optimization solution or near optimization solution. But it is very difficult to solve the optimization problems with complicated constraints. CPT (constraint propagation technique) is the main search approach of CSP. BAB (branch-and-bound) is one of the optimization algorithms that are usually used in OR. In this paper, we propose a hybrid algorithm that integrated CPT into BAB to solve the job-shop scheduling problem with universality and challenge it from a new point of view. The characteristics of this hybrid algorithm focus on the improving of the optimization performance and the enlarging application area of BAB, by imbedding the dynamic adjustable time window constraints of CSP and search methods of CPT into BAB. Experiments show that hybrid algorithm is promising.

Key words: constraint propagation techniques; branch-and-bound; job-shop scheduling