基于粒子群优化的异构多处理器任务调度算法*

李静梅, 张 博, 王 雪
(哈尔滨工程大学 计算机科学与技术学院, 哈尔滨 150001)

摘 要: 为提高异构多处理器任务调度的执行效率,充分发挥多处理器并行性能,提出一种基于粒子群优化的异构多处理器任务调度算法——FPSOTTS 算法。该算法以求得任务最短完成时间为目标,首先通过建立新的编码方式和粒子更新公式实现粒子搜索空间到离散空间的映射,使连续的粒子群优化算法适用于离散的异构多处理器任务调度问题;同时通过引入禁忌算法进行局部搜索,克服粒子群算法的早熟收敛现象,避免陷入局部最优。实验结果表明,FPSOTTS 算法的执行效率优于 Min-min 算法和遗传算法,有效地降低任务的执行时间。FPSOTTS 算法很好地解决了异构多处理器任务调度问题,并且适合于大规模并行任务调度。

关键词: 异构多处理器; 任务调度; 粒子群优化算法; 禁忌搜索
中图分类号: TP303 **文献标志码**: A **文章编号**: 1001-3695(2012)10-3621-04
doi:10.3969/j.issn.1001-3695.2012.10.005

Heterogeneous multiprocessor task scheduling algorithm based on PSO

LI Jing-mei, ZHANG Bo, WANG Xue
(*School of Computer Science & Technology*, *Harbin Engineering University*, *Harbin* 150001, *China*)

Abstract: This paper proposed a task scheduling algorithm named FPSOTTS, which based on particle swarm optimization to improve the task scheduling execution efficiency of heterogeneous multi-processor and make the full use of the performance of parallel multi processors. FPSOTTS algorithm took the shortest task completion time as the goal. Firstly, the FPSOTTS algorithm realized the mapping from particle searching space to discrete space by building new coding method and particle update formula. And the mapping above made the continuous particle swarm optimization algorithm apply to the discrete heterogeneous multiprocessor task scheduling problem probably. At the same time, the FPSOTTS algorithm overcame the particle swarm algorithm premature convergence phenomenon and avoided the obtained solution trapping into the local optimum by introducing tabu search algorithm for local search. The experimental results show that the execution efficiency of FPSOTTS algorithm is superior to Min-min algorithm and genetic algorithm for reducing the task execution time effectively. FPSOTTS algorithm provides a better solution to the task scheduling problems of heterogeneous multi-processor. And it is suitable for large scale parallel task scheduling.

Key words: heterogeneous multi-processor; task scheduling; PSO algorithm; tabu search

0 引言

异构多处理器系统由多个具有不同计算性能的处理器构成。这些具有不同计算性能的处理器通过接收并执行不同运算量和计算类型的任务,不仅可以加快应用程序的执行,重要的是进一步增强了多处理器本身的并行性。所以,如何将不同的任务分配到不同计算能力的处理器上进行处理便成为能否充分发挥多处理器性能的首要问题,这也是目前对任务调度策略的研究成为多处理器技术研究热点问题的原因。但是,目前比较成熟的任务调度算法研究大多基于同构环境[1,2],任务在同构多处理器中每个处理器上的执行时间相同,在任务调度时只需要将任务调度到上一任务完成时间最早的处理器上即可。而异构多处理器上各处理器的处理能力各不相同,在分配任务时不能简单地将任务分配到执行速度最快的处理器上,还要考虑任务在不同处理器上的执行效率和处理器整体的执行时间。因此,异构多处理器上的任务调度问题相对更加复杂,采用同构多处理器任务调度算法不能充分发挥每一个处理器的性能和整体的并行性。

异构多处理器环境下的任务调度是指在满足一定的约束条件下,将一组任务分配到合适的处理器上执行,并使整个应用程序的完成时间最短。任务调度问题是一种组合优化问题,其已被证明是 NP 完全问题[3],不能在多项式时间复杂度内找到最优解。为求解此类 NP 完全问题,一些启发式算法如遗传算法(genetic algorithm, GA)等,开始被应用到任务调度研究中[4~8]。但是,遗传算法运行时需要设置过多的参数,交叉和变异操作的有效配合使用难以控制,同时存在着早熟和大量计算等问题,当求解到一定范围时,求得最优解的效率显著降低。

基于上述背景,本文提出了一种基于粒子群优化(particle swarm optimization, PSO)的任务调度算法——FPSOTTS 算法,并在 MATLAB 仿真平台上与遗传算法和 Min-min 算法进行了

收稿日期: 2012-03-15; **修回日期**: 2012-04-23 **基金项目**: 国家自然科学基金资助项目(61003036, 60873138);黑龙江省自然科学基金资助项目(F201124)

作者简介: 李静梅(1964-),女,黑龙江佳木斯人,教授,博士,主要研究方向为计算机系统结构、多核处理器性能优化等(lijingmei@hrbeu.edu.cn);张博(1986-),男,内蒙古通辽人,硕士,主要研究方向为多处理器任务调度;王雪(1989-),女,黑龙江齐齐哈尔人,硕士,主要研究方向为多核处理器任务调度.

性能比较测试,证明其收敛速度和求解精度均优于现有算法。

1 任务调度问题描述

为了描述问题,本文研究设多处理器系统包含 m 个不同的处理器,一个应用程序被划分成 n 个子任务,并且子任务之间相互独立。定义 $T=\{T_1, T_2, \cdots, T_n\}$ $(i=1, 2, 3, \cdots, n)$ 为 n 个独立任务序列;$P=\{P_1, P_2, \cdots, P_m\}$ $(j=1, 2, 3, \cdots, m)$ 为 m 个处理器序列。当 $n \leqslant m$ 时,即待执行任务数小于处理器的数目,可以根据任务最早执行完成时间分配任务上到指定处理器执行即能获得较优调度方案;当 $n > m$ 时,则可以通过一些调度方案分配任务到相应处理器上执行。本文仅考虑后一种情况。

任务 i 的计算量为 T_i,任务 i 在处理器 j 上的执行时间 cpt_{ij} 已知,cpt 为全部任务执行完的时间,则任务调度问题的数学描述为

$$SL=\min\{cpt^s\} \tag{1}$$

其中:cpt^s 为调度方案 s 下全部任务执行完的时间,SL 即为任务序列 T 在处理器集合 P 上的最小花费时间。

2 粒子群优化算法

PSO 算法是一种基于群体智能理论的全局优化方法,种群(swarm)中粒子通过相互间的合作与竞争进行优化搜索,种群由若干个粒子构成,粒子的个数称做种群大小或规模。每个粒子都有一个位置矢量和速度矢量,其维数代表了问题解空间的维数。它从随机解出发,进行迭代寻找最优解,通过适应度来评价解的质量,粒子追随当前搜索到的最优值寻找全局最优。PSO 算法实现容易、求解精度高、收敛速度快,在解决 TSP、流水车间调度和路径规划等问题中展示了其良好的优越性[9]。

假设问题的搜索空间是一个 d 维空间,则第 i 个粒子的位置矢量和速度矢量可以表示为

$$\begin{aligned} X_i &= [x_{i1}, x_{i2}, \cdots, x_{id}] \\ V_i &= [v_{i1}, v_{i2}, \cdots, v_{id}] \end{aligned} \tag{2}$$

PSO 算法运行时初始化为一群随机粒子(随机解),每个粒子在搜索空间内不断更新速度和位置,然后通过迭代找到最优解。在每一次迭代中,粒子通过跟踪两个极值和不断调整自己的位置进行更新。一个是粒子本身所找到的最优解,被称为个体最优解,第 i 个粒子的个体最优解记为 $\text{pbest}_i=[\text{pbest}_{i1}, \text{pbest}_{i2}, \cdots, \text{pbest}_{id}]$;另一个是整个种群目前找到的最优解,称为全局最优解,记为 $\text{gbest}=[\text{gbest}_1, \text{gbest}_2, \cdots, \text{gbest}_d]$。另外,粒子的所有邻居中的最优解是局部最优解。

在找到个体最优解和全局最优解时,粒子根据状态更新公式更新自己的速度和位置,PSO 算法速度和位置更新公式[10]如下:

$$\begin{aligned} v_{ik}(t+1) &= \omega v_{ik}(t) + \\ & c_1 r_1(\text{pbest}_{ik}(t) - x_{ik}(t)) + \\ & c_2 r_2(\text{gbest}_k - x_{ik}(t)) \\ x_{ik}(t+1) &= x_{ik}(t) + v_{ik}(t+1) \\ 1 \leqslant i \leqslant n, & 1 \leqslant k \leqslant d \end{aligned} \tag{3}$$

其中:n 是种群大小;t 是迭代次数;c_1 和 c_2 为加速常数,一般取值为 2;r_1 和 r_2 为 0 ~ 1 的随机数;ω 是惯性权重,用于平衡收敛的全局性和收敛速度,计算方法如下:

$$\omega^t = \omega_{\min} + \frac{\omega_{\max} - \omega_{\min}}{N} \times (N - t) \tag{4}$$

其中:N 为最大迭代次数;$\omega_{\max}$ 和 $\omega_{\min}$ 分别为 ω 的最大值和最小值,通常取值为 0.9 和 0.4;t 是迭代次数。

PSO 算法实际运行过程中,由于粒子更新过程中总会考虑周围粒子的信息,随着迭代次数的增加就会产生“聚堆”现象,粒子会在问题过程中逐渐丧失多样性,致使过早地收敛,容易陷入局部最优。为了解决此问题,PSO 算法多使用一些元启发式算法进行局部搜索,以加强 PSO 算法的收敛性能。

3 基于粒子群优化的调度算法

3.1 粒子编码方案

异构多处理器环境中,任务调度算法需要解决的是在提高资源利用率和系统效率的同时,如何最大限度地减少完成时间的问题。PSO 算法是一种连续算法,算法的更新公式和过程是面向连续空间并为其设计的,而调度问题属于离散问题。本文根据异构多处理器调度问题特点,重新构建粒子表达方式,对粒子的位置和速度进行编码,将 PSO 算法映射到离散空间,使其适用于解决任务调度问题。

PSO 算法中每一个粒子都代表一个任务调度问题的潜在解。粒子位置矢量定义为一个 $n \times m$ 矩阵 $\boldsymbol{X}$,每一列代表一个任务分配情况,每一行代表一个处理器执行情况。式(5)为粒子位置编码方案,约束条件为式(6)。

$$\boldsymbol{X}=\begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1n} \\ x_{21} & x_{22} & \cdots & x_{2n} \\ \vdots & \vdots & & \vdots \\ x_{m1} & x_{m2} & \cdots & x_{mn} \end{bmatrix} \tag{5}$$

$$\text{s. t. } x_{ij} \in \{0,1\}, \sum_{i=1}^{m} x_{ij} = 1 \tag{6}$$

根据约束条件式(6):位置矩阵 $\boldsymbol{X}$ 的元素 $x_{i,j}$ 取值为 0 或 1,每一行可以有多个 1 出现,每一列任意一个元素都可以为 1,每一列只允许仅有一个元素值为 1,即

a)矩阵中 $x_{i,j}=1$ 表示任务 T_i 分配到处理器 P_j 上执行,否则 $x_{i,j}=0$。

b)每一个处理器都可以执行多个任务。

c)任务可以分配到任意一个处理器上执行,且任务必须被分配到一个处理器上执行。

d)一个任务不允许同时分配到多个处理器上,即任务执行不能被中断。

速度定义为粒子达到目标状态所需要对其当前位置执行的交换次序。速度 $\boldsymbol{V}$ 为 $n \times n$ 矩阵,如式(7)所示,速度矩阵 $\boldsymbol{V}$ 中的元素 v_{ij} 只取值为 0 或 1。

$$\boldsymbol{V}=\begin{bmatrix} v_{11} & v_{12} & \cdots & v_{1n} \\ v_{21} & v_{22} & \cdots & v_{2n} \\ \vdots & \vdots & & \vdots \\ v_{n1} & v_{n2} & \cdots & v_{nn} \end{bmatrix} \tag{7}$$

$$\text{s. t. } v_{ij} \in \{0,1\}, \sum_{i,j=0}^{n} v_{ij} + v_{ji} = 0 \text{ 或 } 1 \tag{8}$$

此时,粒子的位置和速度状态不再是同维连续矢量。粒子

的速度和位置不能再参照式(3)进行更新,为了利用此编码方式,本文重新定义粒子速度和状态更新公式中的加减法、乘法运算规则:

a)定义 PSO 算法中的加减法操作为交换操作,即 $v_{ij}=1$ 表示将位置矩阵 X 中第 i 列取值为 1 的元素和第 j 列值为 1 的元素所在的行序号进行交换,两个交换操作可以合并成一个新的交换。例如,矩阵 X 中,$x_{22}=1$,$x_{35}=1$,若 $v_{25}=1$,交换后有 $x_{32}=1$,$x_{25}=1$,即相互交换任务 i 和任务 j 分配到的处理器,根据速度约束条件式(8),速度矩阵里的元素 v_{ij} 和 v_{ji} 不能够同时为 1,避免了一次更新过程中无效的重复交换现象。

b)定义速度与随机数的数乘为以随机数对应概率值决定是否按照速度矩阵进行交换操作。

重新定义的运算规则使用符号"⊕"表示,则重新定义后的粒子状态更新公式为式(9),各变量含义同式(3)。

$$\begin{aligned} v_i(t+1) &= v_i(t)\oplus c_1\oplus[\text{pbest}_i - x_i(t)]\oplus \\ &\quad c_2\oplus[\text{gbest} - x_i(t)] \\ x_i(t+1) &= v_i(t+1)\oplus x_i(t) \end{aligned} \tag{9}$$

根据式(10)所示进行位置更新的结果如式(11):

$$X(t)=\begin{bmatrix}1&1&0\\0&0&0\\0&0&1\end{bmatrix},V(t+1)=\begin{bmatrix}0&0&1\\1&0&0\\0&0&0\end{bmatrix} \tag{10}$$

$$X(t+1)=X(t)+V(t+1)=\begin{bmatrix}0&1&1\\0&0&0\\1&0&0\end{bmatrix} \tag{11}$$

由此可见,粒子的这种编码方案和约束简单明了,符合异构多处理器任务调度规则,能够表示出所有可能的任务调度方案,并且将粒子搜索空间和任务调度方案一一映射起来,同时避免了冗余搜索。

3.2　适应度

粒子位置好坏的评价标准由粒子的适应度进行评价。粒子适应度通过适应度函数计算得到,本文使用粒子的 makespan作为其适应度评价标准。粒子的 makespan 是指粒子中全部任务的最晚完成时间,makespan 越小,粒子位置越接近最优粒子,其所代表的潜在调度方案越好。第 y 个粒子适应度值计算方法如下:

$$f(y)=\max\{\text{makespan}(p_j)\}\qquad j=1,2,\cdots,m \tag{12}$$

其中:j 为处理器编号,$\text{makespan}(p_j)$ 表示第 j 个处理器上所有任务的完成时间。$\text{makespan}(p_j)$ 通过式(13)计算得到。

$$\text{makespan}(p_j)=\sum_{i=0}^{n}T_i\times x_{ij}\qquad j=1,2,\cdots,m \tag{13}$$

由式(12)(13)得到适应度函数为

$$f(y)=\max\{\sum_{i=0}^{n}T_i\times x_{ij}\}\qquad j=1,2,\cdots,m \tag{14}$$

3.3　FPSOTTS 算法异构多处理器任务调度流程

PSO 算法搜索后期,由于粒子向最优点聚集而导致种群多样性迅速降低,如果是局部最优点,则会使 PSO 算法陷入局部最优解,产生早熟现象。

禁忌搜索(tabu search,TS)算法是局部搜索算法的一种扩展,通过引入一个被称为禁忌表的存储结构和相应的禁忌规则来有效探索以实现全局优化。本文通过融入了 TS 算法对 PSO 算法局部搜索性能进行改进,提出 FPSOTTS 算法。

FPSOTTS 算法将 PSO 算法的全局搜索能力和 TS 算法的局部搜索能力相结合,一定程度上改善了 PSO 算法的求解精度和全局收敛性。同时,由于在 PSO 进化后期群体质量已经较好,此时调用 TS 极大地提高了 TS 算法的搜索速度,减少了 FPSOTTS 算法的执行时间。

FPSOTTS 算法首先使用 PSO 算法进化整个种群,随着迭代次数的增加,粒子比较稳定地分布在解空间,种群多样性程度较低,这时从 PSO 粒子的当前位置出发,开始执行局部 TS 搜索,因此种群多样性程度的判断对 FPSOTTS 算法的收敛性能极为重要。

种群多样性采用种群的适应值方差 σ^2 来衡量。设种群规模为 N,第 i 个粒子的适应值为 makespan_i,种群平均适应值为 $\text{makespan}_{\text{avg}}$,根据方差公式有

$$\delta^2=\sum_{i=1}^{N}\frac{(\text{makespan}_i-\text{makespan}_{\text{avg}})^2}{N} \tag{15}$$

其中:σ^2 动态跟踪种群适应值的整体变化,其大小直接反映了 PSO 的多样性程度,σ^2 越小,则种群聚集程度就越大,多样性越低;λ 为初始种群的多样性程度,随着迭代次数的增加,σ^2 逐渐降低,通过多次实验分析,选定当 $\sigma^2<\lambda/10$ 时,使用 TS 进行局部搜索。

局部搜索以每次迭代的 gbest 位置矢量和其作为全局极值的次数构造 TS 算法的禁忌表。为防止 TS 搜索过程陷入循环状态,本文采用文献[11]中的欧几里德距离判断一个粒子是否被禁忌,如式(16)所示:

$$|x-x'|\leqslant h_0 \tag{16}$$

其中:$h_0=0.01\times(x_{\text{best}}-x_{\text{bad}})$;$x$ 为当前粒子;x'为禁忌表里的其他任何粒子,x_{best}为禁忌表里位置最好粒子,x_{bad}为禁忌表里位置最差粒子。根据式(11),如果 x 在一定范围 h_0 内接近 x',那么就认为点 x 是禁忌的。对于禁忌的粒子 x,若 $\text{makespan}(x)<\text{makespan}(x_{\text{best}})$,则认为点 x 符合特赦准则,此时解禁粒子 x,并更新当前找到的最优粒子。

FPSOTTS 算法的异构多处理器任务调度算法完整流程可以概括如下:

a)初始化。设置有关 FPSOTTS 算法参数:随机初始化每个粒子的位置矢量和速度矢量;设定粒子群的规模和最大速度 $V_{\max}$;初始化禁忌表和禁忌次数。

b)根据目标函数计算各个粒子的适应值,初始化粒子的个体最优位置和种群的全局最优位置,设置粒子当前位置为 pbest,种群最佳位置为全局最优值 gbest。

c)运用式(9)更新粒子的位置和速度。

d)使用适应度函数式(14)计算出每一个粒子的适应值。

e)找出个体最优粒子和全局最优粒子,并将求得的适应度分别与 pbest 和 gbest 的适应度比较,若较好,则更新 pbest 或 gbest。

f)更新迭代变量。

g)计算种群多样性程度,若 $\sigma^2<\lambda/10$,则执行下一步;否则执行步骤 c)。

h)若禁忌表为空,则执行下一步;否则,若当前粒子满足特赦规则 $\text{makespan}(x)<\text{makespan}(x_{\text{best}})$,更新最优粒子和禁忌表,否则执行下一步。

i)计算当前全局最优粒子所属区域,若当前 gbest 满足式

(16),则该分量对应的被选为全局极值的禁忌次数增加一次,更新禁忌表;否则转入 h)。

j)判断是否满足结束条件(达到 TS 的最大迭代次数),如果满足,继续下一步, 否则执行 h)。

k)输出 gbest,算法运行结束。

4 仿真实验

本文在 MATLAB 仿真环境中,对 FPSOTTS 算法、Min-min 算法和 GA 进行性能[12]对比验证。基准测试用例采用 Watson 等人[13]提出的基准测试套件来进行调度算法的性能测试。

FPSOTTS 和 GA 初始种群大小为 100,最大迭代次数为 500,惯性权重 ω 取值 0.7,禁忌表长度 L 为 8,交叉和变异概率分别为 0.8 和 0.03。为了抵消数据随机性对测试结果的影响,更好地反映算法的性能,完成时间取 20 次测试中得到最优解的平均完成时间。

本文实验中测试选定处理器数为 5,在任务数目为 50、100、150、200、250、300 情况下,在 MATLAB 仿真平台上执行以上三种算法。

图 1 为不同任务数情况下,三种算法的执行时间,即 makespan 变化曲线。通过图 1 可以看出,任务数较少时,FPSOTTS 算法的执行时间稍优于 Min-min 算法和 GA,但随着任务数目的增加,差距越来越明显。在任务数较多时,FPSOTTS 算法具有更短的执行时间,性能明显优于 Min-min 算法和 GA。这是因为 Min-min 算法本质上是一种贪心算法,其片面追求完成时间最早的局部最优解,只考虑将任务调度到最先执行的处理器上,导致个别处理器上任务分配过多,造成了整体时间跨度的增加;GA 在任务数较多时,交叉和变异操作的多样性会受到限制,效率低下;而 PSO 算法受问题维数的影响更小,并且具有更好的并行搜索性能。因此,FPSOTTS 算法更适合于大规模并行任务调度。

图 2 为使用 FPSOTTS 算法和 GA 在 100 个任务和 5 个处理器条件下进行任务调度的完成时间—迭代次数曲线。由图 2 可以看出,相对于 GA,FPSOTTS 算法找到最优值的迭代次数更少,解的质量更高,即适应度更低,全部任务完成时间更少。这是因为 GA 复杂的交叉和变异操作增加了算法的时间复杂度,算法后期容易错过最优解;FPSOTTS 算法具有 PSO 算法收敛速度快的特点,同时融入了 TS 算法对质量较好解进行局部搜索,增加了种群多样性,保证收敛到全局最优,提高了算法的求解质量。

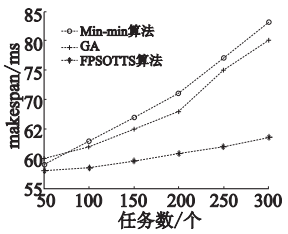

图1 FPSOTTS算法、Min-min和GA的makespan对比

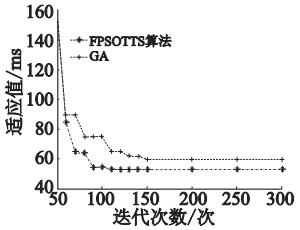

图2 FPSOTTS算法和GA的迭代次数比较

5 结束语

为提升异构多处理器系统中的任务调度效率,本文提出了基于 PSO 算法的 FPSOTTS 算法来求得最短任务完成时间。FPSOTTS 算法充分利用了 PSO 算法收敛速度快和并行性的特点,通过建立粒子编码方式和重新定义粒子更新公式,使连续的 PSO 算法适用于异构多处理器任务调度,并采用 TS 算法在种群多样性较低时进行局部搜索,避免了 PSO 算法的早熟现象。通过与 Min-min 算法和 GA 进行性能测试比较,FPSOTTS 算法比 GA 能在更少的迭代次数内搜索到最优解,并且最优解的质量更高,算法的执行速度也优于 Min-min 算法和 GA,能够在更短的时间内完成任务调度,很好地解决了异构多处理器系统中的任务调度问题,在大规模运算环境中有很好的研究价值和应用前景。

参考文献:

[1] 赵鹏,严明,李思昆. 异构多处理器 SoC 的应用算法性能优化方法[J]. 软件学报,2011,22(7): 1475-1487.

[2] FATIH-TASGETIRENA M, LIANG Y C, SEVKLI M, *et al*. Particle swarm optimization and differential evolution for the single machine total weighted tardiness problem[J]. International Journal of Production Research,2006,44(22): 4737-4754.

[3] ULLMAN J D. NP-complete scheduling problems[J]. Journal of Computer and System Sciences,1975,10(3): 384-393.

[4] THANUSHKODI K, DEEBA K. On performance analysis of hybrid algorithm (improved PSO with simulated annealing) with GA, PSO for multiprocessor job scheduling[J]. WSEAS Trans on Computers,2011,10(9): 287-300.

[5] WEN Yun, XU Hua, YANG Jia-dong. A hybrid heuristic-genetic algorithm for task scheduling in heterogeneous processor networks[J]. Journal of Parallel and Distributed Computing,2011,71(11): 1518-1531.

[6] MOGHADDAM M E, BONYADI M R. An immune-based genetic algorithm with reduced search space coding for multiprocessor task scheduling problem[J]. International Journal of Parallel Programming,2011,40(2): 225-257.

[7] 姚丽丽, 史海波, 刘昶. 基于遗传算法的混合流水线车间调度多目标求解[J]. 计算机应用研究, 2011,28(9): 3264-3271.

[8] ABDEL-KADER R F. An improved discrete PSO with GA operators for efficient QoS-multicast routing[J]. International Journal of Hybrid Information Technology,2011,4(2): 23-38.

[9] 李娅, 李丹, 王东. 改进的混沌粒子群算法求解车辆路径问题[J]. 计算机应用研究, 2011,28(11): 4108-4110.

[10] SHI Y, EBERHART R C. Fuzzy adaptive particle swarm optimization[C]//Proc of IEEE Conference on Evolutionary Computation. [S. l.]: Institute of Electrical and Electronics Engineers Inc, 2001:101-106.

[11] CHELOUAH R, SIARRY P. Tabu search applied to global optimization[J]. European Journal of Operational Research,2000,123(2):256-170.

[12] LI Zhi-qiang. Optimization for the parallel test task scheduling based on GA[C]//Proc of the 2nd International Conference on Information Science and Engineering. 2011:5223-5226.

[13] WATSON J P, BARBULESCU L, WHITLEY L D, *et al*. Contrasting structured and random permutation flowshop scheduling problems: search space topology and algorithm performance[J]. ORSA Journal of Computing,2002,14(2):98-123.