一种基于 DQCGA 算法的软硬件动态划分方法*)

范乐君　李　斌　庄镇泉　傅忠谦
(中国科学技术大学智能信息处理研究室　合肥 230026)

摘　要　软硬件划分是嵌入式系统设计的高层抽象环节中最重要的关键步骤之一。在某些数据相关的应用领域中，划分环境是动态变化的，因此我们提出了一种解决动态软硬件划分的方法。这种方法基于一种名为 DQCGA 的演化算法。DQCGA 算法受自然界中对称和互补机制的启发，操纵一对互补的概率向量来适应动态变化的环境。我们系统地完成了建模，动态环境定义等环节，然后通过和已有方法的比较，有针对性地设计了实验。试验结果很好地证明了该方法对于解决软硬件划分问题的可行性和有效性，并且较之以往的方法有着更好的表现。

关键词　嵌入式系统，软硬件协同设计，动态划分，DQCGA

An Approach for Dynamic Hardware /Software Partitioning Based on DQCGA

FAN Le-jun　LI Bin　ZHUANG Zhen-quan　FU Zhong-qian
(Nature Inspired Computation and Applications Laboratory, University of Science and Technology of China, Hefei 230026, China)

Abstract　HW/SW partitioning is one of key steps in high level abstraction of embedded system co-design. In some kinds of data-dependence applications, the partitioning environment is dynamic. In the paper we present an approach to solve dynamic HW/SW partitioning problem based on a kind of evolutionary algorithm called DQCGA. DQCGA was inspired by Dualism or complementarity in the nature and operated on a pair of dual probability vectors to adapt itself to the dynamic environments. We systematically construct the system model, define the dynamic environment and conduct experimental analysis via comparison with the existing method. Experimental results show that our approach is feasible and efficient for dynamic HW/SW partitioning, and had much better performance than traditional ones.

Keywords　Embedded system, HW/SW co-design, Dynamic partitioning, DQCGA

1　引言

嵌入式系统设计在过去的几年中发生了深刻的变化，设计者在很多方面，包括芯片尺寸、功耗、代价、系统执行和相应时间以及越来越短的设计时限等等，都面临着严峻的挑战。设计周期的缩短加大了对早期设计的需求，设计需求的多样化又导致了异质系统结构的大量引入。因此，软硬件划分作为系统高层抽象体系结构设计中的一个关键环节，逐渐成为嵌入式系统设计中最重要的问题之一。

一般来说，软硬件划分可以描述为决定系统行为模型中各个功能对象向着选定的系统实际架构的映射的过程。在过去的大量深入的研究和探索的基础上，软硬件划分工作已经得到了一系列可行的结果和方法。

在大多数传统的软硬件划分方法中，系统的属性描述都是基于固定的状态的。然而在许多实际的应用中(例如图像处理等)，某些对于划分算法非常关键的参数(比如任务的执行时间等)，是根据待处理的数据流的大小而动态变化的。而这一类的应用往往对时间还有着严格的约束条件，因此要求算法必须具有处理动态问题的能力。对于这类特殊的问题，相关的工作和文献比较匮乏。文献[1]中提出了基于软硬件实时重分配的方法，对于系统中时间消耗最大的循环，根据功能块的执行时间对系统的软硬件映射进行重新的配置。文献[2]提出了一种对动态配置进行人工选择的方法。文献[3]提出了另一种在线划分方法，通过周期性的操作在执行速度优先和系统实现代价优先两者之间寻找合适的平衡点。

在本文中，我们将研究动态软硬件划分问题的建模方法以及 DQCGA 算法(Dual Quantum Codeing Genetic Algorithm)在动态软硬件划分模型中的应用。DQCGA 算法是一种 EA 算法(Evolutionary Algorithm)，并且已经被证明在处理动态优化问题中有着突出的表现[4]。我们将观察该算法在一系列不同的动态环境设定下的表现，并且和文献[3]中提出的基于动态重配置的 OPA 算法进行比较。试验结果表明，DQCGA 方法对于 OPA 算法有着明显的优势。

下面我们将首先介绍动态软硬件划分问题的描述方法，然后是 DQCGA 算法的细节和动态环境的定义，接下来是实验结果和分析，最后是总结。

2　问题描述

2.1　系统模型和目标架构

在嵌入式系统设计中，系统的高层抽象模型通常用任务图(Task Graph) $G(N, E)$来描述，包含点集$|N|$和边集$|E|$。N表示系统的功能节点，E表示节点之间的通讯和数据流向。一般来说，目标系统的架构可以简化为图 1，包含一个软件处理单元(例如 RISC)和一个硬件处理单元(例如带有重配置能力的 FPGA)。RISC 和 FPGA 通过系统总线相互通信。我们假设 RISC 和 FPGA 能够同时执行任务，并且在

*)本文由国家自然基金(No. 60401015, No. 60572012)、安徽省自然基金(No. 050420201)支持。范乐君　硕士；李　斌　博士，副教授；庄镇泉　博士，教授；傅忠谦　博士，副教授。

FPGA 上动态分配的任务通过共享内存机制相互通信(所谓共享内存机制是指根据内存需求的不同将片上内存和片外的独立存储器分别地映射为逻辑内存)。在这样的目标架构下,一个任务节点既可以被映射到硬件处理单元来获得更快的执行速度,也可以被映射到软件处理单元来降低系统代价。软硬件划分的目的就是在执行时间系统代价之间寻找平衡点。

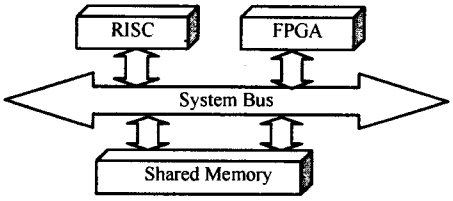

图 1 常见的目标系统体系结构

2.2 计算模型的提取

根据上述对软硬件划分的描述,每一个节点可以被描述为 $N_i=\{T_{hw}(i),\ T_{sw}(i),\ C_{hw}(i),\ C_{sw}(i),\ x(i)\}$,前 4 个参数分别表示该节点在硬件和软件上的执行时间和代价,最后一个参数表示该节点所选定的映射模式。$x(i)=1$ 表示映射到硬件,$x(i)=0$ 表示映射到软件。

总的系统代价可以表示为

$$\begin{aligned}\mathrm{Cost}(x)&=\sum C_{hw}(i)\times x(i)+\sum C_{sw}(i)\times(1-x(i))\\&=\sum C_{sw}(i)+\sum(C_{hw}(i)-C_{sw}(i))\times x(i)\end{aligned}\qquad(1)$$

总的执行时间可以表示为

$$\begin{aligned}\mathrm{Time}(x)&=\sum T_{hw}(i)\times x\ (i)+\sum T_{sw}(i)\times(1-x(i))\\&=\sum T_{sw(i)}+\sum(T_{hw}(i)-T_{sw}(i))\times x(i)\end{aligned}\qquad(2)$$

这里,由于参数 E 通常是由系统设计的另一个步骤"任务调度"所使用,为了系统描述的简化和增强对划分问题的针对性,本文没有引入任务调度带来的对实际总执行时间的增益,因此没有使用参数 E,而是用功能节点执行时间的总和来表示系统的时间函数。考虑到实际问题中通常对总执行时间有严格的要求,我们定义了总执行时间的约束值,表示为常量 T_{cons}。

这样,软硬件划分问题就可以被抽象为

问题目标:求 $x(i)$,$i=1,2,\cdots,N$,使得 $\mathrm{Cost}(x)$ 最小

约束条件:$\mathrm{Time}(x)<T_{\mathrm{cons}}$ (3)

由公式(1),(2)及问题定义(3)可以看出,软硬件划分可以抽象为一个标准的 0-1 背包问题。

2.3 动态环境生成

对于动态软硬件划分而言,由于问题的特殊性,任务节点的属性受输入数据流的影响是动态变化的,通常是代表执行时间的 T 参数在系统运行的时候发生不规则的变化,因此动态的系统参数必须考虑进来。为了简化问题,我们假设 T 参数是在最大值和最小值之间波动,即 T 参数不规则地被 T_{max} 和 T_{min} 取代。

我们采用了一种和文献[5]类似的动态环境生成技术,但是我们做了一些改进,使得它能更合理地描述动态环境的特点。这种技术包含两个动态环境参数:变化的速度和变化的幅度。第一个参数和环境变化的周期相关,表示为 τ,定义为两次变化之间演化算法的执行的周期数(代数),即每 τ 代适应度前景发生变化。第二个参数和变化的幅度相关,代表每一个环境变化的周期分别生成的二进制模板 $E(k)\in\{0,1\}^L$(L 表示染色体长度)中"1"的取值概率,表示为 r。对于每一个环境变化的周期 $k=[t/\tau]$,我们生成另一个二进制模 $F(k)\in\{0,1\}^L$ 并且按照(4)式迭代:

$$F(k)=F(k-1)\oplus E(k)\qquad(4)$$

对于第一个周期,$F(1)$ 初始化为一个全零向量。$E(k)$ 在每个新的周期开始时随机生成,其中包含 $r\times L$ 个"1"。"⊕"是位异或操作符(例如 $1\oplus1=0$,$1\oplus0=1$,$0\oplus0=0$)。

在文献[5]中,$F(k)$ 和 $E(k)$ 是用来改变每一代中的个体。在我们的方法中,与之不同的是,$F(k)$ 和 $E(k)$ 是用来改变适应度函数中的 T 参数。主要原因是适应度函数代表了个体对环境的适应程度。改变适应度函数中的 T 参数,将改变解空间中适应度函数的前景,进而改变个体的评估标准。这是对实际动态环境更自然的模拟。$F(k)$ 模板是 T 参数的标志。当 $F(k)(i)=1$,相关位的 T 参数就被取代为 $T_{\max}$;当 $F(k)(i)=0$,相关位的 T 参数就被取代为 $T_{\min}$。$E(k)$ 用来通过异或操作翻转 $F(k)$ 中的标志位。当 $E(k)(i)=1$,相应的标志位就被翻转,反之则不变。由此可以看出,r 参数控制着环境变化的幅度,r 越大,$E(k)$ 中就有更多的"1",环境变化也就越剧烈。

此外,这种方法对于演化操作没有特殊的要求,这使得它适合使用很多启发式优化方法,包括 EDA 算法(estimation of distribution algorithms),用概率模型估计解空间中出现最优解希望较大的区域,并且对概率模型采样生成新的个体,构成新的群体,取代了传统演化算法中的交叉和变异操作,同时还能有效地保持个体的多样性。下面的 DQCGA 算法就是其中的一种。

3 动态划分问题中的 DQCGA

QCGA 最先在文献[7]中提出,其核心思想是用量子比特串将解空间转化为类似遗传算法的染色体形式,然后通过操作一个概率向量来控制量子比特串的属性,用一种旋转门技术来修正该概率向量。QCGA 算法已经被证明在大量标准的优化问题中有着突出的表现。

在量子计算中,用来储存信息的单位是一个具有两个状态的量子系统,称为量子比特(qubit)。量子比特和经典比特的不同之处在于,它是两个量子状态的组合,一般可以表示为

$$\Phi=\alpha|0\rangle+\beta|1\rangle$$

(α,β)是一对复数变量,成为量子比特的概率幅度,它们之间满足

$$|\alpha|^2+|\beta|^2=1$$

$|0\rangle$ 和 $|1\rangle$ 分别表示两个不同的量子比特状态,所以一个量子比特可以同时表示两个状态的信息。

一个标准的染色体可以定义为式(3),其中我们用更多的量子比特来表示两个以上的状态:

$$q_j^t=\left(\begin{array}{c|c|c|c|c|c|c|c|c|c|c|c|c|c|c}\alpha_{11}^t & \alpha_{12}^t & \cdots & \alpha_{1k}^t & \alpha_{21}^t & \alpha_{22}^t & \cdots & \alpha_{2k}^t & \cdots & \alpha_{m1}^t & \alpha_{m2}^t & \cdots & \alpha_{mk}^t\\ \beta_{11}^t & \beta_{12}^t & \cdots & \beta_{1k}^t & \beta_{21}^t & \beta_{22}^t & \cdots & \beta_{2k}^t & \cdots & \beta_{m1}^t & \beta_{m2}^t & \cdots & \beta_{mk}^t\end{array}\right)$$

变量 1　　变量 2　　变量 m

其中 m 是染色体中基因的数目，k 是用来编码一个基因所需的量子比特的数目，可以通过 $k=\text{ceil}(\log_2 n)$ 得到，n 是每个基因的状态总数，ceil(x) 函数是大于 x 的最小整数，基因的进化操作通过如下的旋转门操作来实现：

$$\begin{pmatrix}\alpha_1'\\ \beta_1'\end{pmatrix}=\begin{pmatrix}\cos(\theta_i) & -\sin(\theta_i)\\ \sin(\theta_i) & \cos(\theta_i)\end{pmatrix}\begin{pmatrix}\alpha_i\\ \beta_i\end{pmatrix}$$

从(4)式可以得出，令 $\alpha=\cos(t)$，$0<t<90°$，则 α 通过旋转门操作后变为 $\cos(t+\theta)$，

θ_i 是对不同问题进行分别定义的，用来控制不同的进化步长。通常，为了保证收敛性，定义为

$$t_i+\theta_i=(1-s)\times(t_{i-1}+\theta_{i-1})+90°\times s\times\alpha_{best}[i]$$

其中 s 为步长控制系数，α_{best} 为当前代最优个体的状态测量值。

对称性和互补性是自然界中非常普遍的现象。例如生物学中，DNA 分子就包含两个对称的串，相互缠绕形成一个双链。受到自然界中这种对称机制的影响，针对前文提出的动态划分问题，我们利用对称性对 QCGA 算法进行了改进，得出一种新的 DQCGA 算法。首先，给定一个长度为 L 的概率向量 $\alpha=(\alpha[1],\cdots,\alpha[L])\in I=[0,1]^L$，它的对称向量可以定义为：$\beta=(\beta[1],\cdots,\beta[L])\in I=[0,1]^L$，其中 $\alpha[i]^2+\beta[i]^2=1$（$i=1,\cdots,L$）。也就是说这两个向量在解空间中关于中心点对称。根据这种定义，我们将一对这样的对称向量表示的两个量子比特串 α 和 β 引入到 DQCGA 算法中。DQCGA 算法的主要步骤如下：

1）初始化两个对称概率向量、群体规模，以及各向量的样本规模配额。

2）分别根据两个向量的样本规模对两个对称向量进行一次测量，产生一代群体，然后计算每个个体的适应度函数，分别从每个向量的样本中选出适应度最大的个体。

3）比较这两个最佳的个体，选出较优的一个，然后通过旋转门的方法，按照设定的步长，用产生较优个体的向量来学习该个体的概率，同时相应地改变另一个向量，以保持两个向量之间的对称性。

4）修正两个向量的样本规模，将产生第 3 步中较优个体的向量的样本规模适当增大，但是要保持其对称向量的样本规模配额，以保证产生个体的对称性。

5）如果未达到终止条件，则返回第二步，产生新一个的群体，向下重新循环执行。

在动态软硬件划分中，根据我们在前文中的设定，在新的变化周期到来时，某些随机节点的 T 参数将会激烈地改变，这意味着该节点的执行时间突然变化，或者相对的执行代价的突然变化，这将可能导致这些节点的软硬件配置完全改变。对于算法而言，最优解的相应位也有可能完全翻转，尤其是环境变化比较剧烈的时候，这种现象更加明显。因此，单概率向量的设定将无法满足动态软硬件划分的设计需求。在 DQCGA 算法中，一对对称的概率向量能比单向量更好地适应动态环境，主要因为对称机制增强了在解空间中搜索的目的性。通过双对称概率向量对解搜索能力的增强，能有效地应对动态环境下最优解的位翻转造成的在解空间中的跳跃性变化，求解的收敛速度和陷入局部最优解的概率都有明显的改善。这些特性在接下来的试验中得到了很好的证明。

4 实验与分析

我们采用的实验模型是一个具有 30 个节点的任务图，各节点的参数由文献[6]提出的 TGFF 技术随机产生。执行时间参数 T 设定成：$T_{min}=T$，$T_{max}=2T$，终止条件为最大执行代数 100 代，然后采用前文中的技术构建了一系列动态环境。我们分别用不同的 $r=0.1$，0.5，0.75，0.95 来模拟环境变化的程度由轻微向剧烈转变的过程，用不同的 $\tau=10$，20，50 来模拟环境变化由慢到快的过程。对于这样 4×3 个动态问题，我们分别使用不同的随机种子对每一种动态参数的配置执行 30 次独立的实验来获得结果的平均值。实验结果的分析将在下一段中列出。

实验平台是 P4 2.4GHz 的主机，实验集成环境是 VC++6.0 IDE。为了衡量我们的方法的表现，我们将实验结果与文献[3]中的 OPA 方法进行了对比。OPA 方法基于一种在线重分配技术，通过执行速度优先和执行代价优先两种策略的迭代作为主要的优化步骤。我们通过下面有效性、稳定性、实用性、适应性四个方面的比较来对两种方法的性能进行评估。

4.1 有效性

我们使用 F_{avg} 来表示算法的有效性，它定义为每一代中找到的最佳适应度的平均值，由指定动态环境下所有独立实验的所有代的最佳适应度得出。对于 2.2 节中式(3)所定义的软硬件划分问题而言，适应度函数的值越小，意味着算法找到的解越接近最优解，算法的有效性也就越高。因此，对于每一组动态环境参数，我们都使用 F_{avg} 来估计算法的有效性。

根据 2.2 节中的计算模型，DQCGA 算法的适应度函数定义如下：

$$fit=\begin{cases}\text{Cost}(x) & ,\text{when}\quad \text{Time}(x)<T_{cons}\\ \text{Cost}_{HW} & ,\text{when}\quad \text{Time}(x)\geqslant T_{cons}\end{cases}\tag{5}$$

Cost_{HW} 表示当总执行时间超过约束值时适应度函数的惩罚项，其值为一常量，由所有节点都划分为硬件时的总代价计算得出。

平均适应度则被定义为

$$F_{avg}=\frac{1}{R}\sum_{r=1}^{R}\frac{1}{N}\sum_{g=1}^{N}fit_{best}(r,g)\tag{6}$$

N 是总代数，R 是总独立实验执行次数。$fit_{best}(r,g)$ 是第 r 次独立实验、第 g 代的最优 fit 值。表 1 是实验结果的一个概要。

表 1 DQCGA 和 OPA 的总体表现

序号	参数设定	DQCGA	OPA
1	$\tau=10$, r=0.1	1572	2020
2	$\tau=10$, r=0.5	1910	2394
3	$\tau=10$, r=0.75	1979	2411
4	$\tau=10$, r=0.95	2106	2588
5	$\tau=25$, r=0.1	1244	1777
6	$\tau=25$, r=0.5	2019	2220
7	$\tau=25$, r=0.75	2052	2505
8	$\tau=25$, r=0.95	2232	2602
9	$\tau=50$, r=0.1	980	1295
10	$\tau=50$, r=0.5	1078	1392
11	$\tau=50$, r=0.75	1166	1358
12	$\tau=50$, r=0.95	1158	1476

从表 1 中可以看出，DQCGA 的表现要明显地优于 OPA，几乎在所有的动态环境设定下 DQCGA 都有较小的 F_{avg} 值。而且，当“τ”的值固定时，两种算法平均适应度值的差值随着“r”从 0.1 增加到 0.5，然后 0.95 基本保持稳定。这意味着

当环境变化越来越剧烈的时候 DQCGA 仍然能够比 OPA 工作得更好。主要原因就是 DQCGA 在每一代中使用了对称的概率向量来对解空间分别采样，这增加了算法在每一代中搜索的区域，并且对称的概率向量有针对性地适应了动态环境的变化方式，使得搜索解空间时的目的性大大增强，因此也有更多的机会来寻找到动态环境下的较优解。

4.2 稳定性

对于算法稳定性的表现，我们使用适应度函数在单独实验中的曲线的波动来评估。图 2 是一个典型的例子，$r=0.95$，$\tau=10$，表示的是变化最激烈的动态环境。

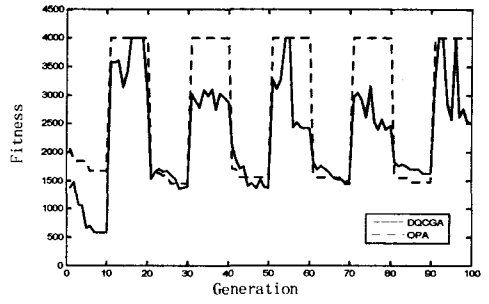

图 2 $r=0.95$，$\tau=10$ 时的单次变化曲线

从图 2 中可以看出，虚线表示的 OPA 算法适应度函数曲线有着明显的不足：在每个偶数的动态周期中，OPA 的适应度函数曲线就被锁定在 $Cost_{HW}$，这是因为算法在前一个奇数的动态周期中找到了较优解后，因为环境的突然剧烈变化，无法快速地找到新的较优解在解空间的可能位置，搜索能力急剧下降。而我们的 DQCGA 算法明显地能够快速适应动态环境的急剧变化，这主要得益于对称的概率向量的存在。当环境变化使得最优解的位置几乎在解空间中完全反转时，另一个向量携带的概率分布信息能够很快地找到变化了的最优解位置导向的信息，从而将适应度值迅速地向新的最优解稳定下来，减轻了环境变化对算法求解性能的影响。图中的实线很好地说明了我们的方法在剧烈的动态环境中的表现。

4.3 实用性

OPA 方法的主要实际步骤包括划分、预估、调度和数据库通信，并且应用于一个实际的 ICAM 系统的设计中，其实用效果已在多方面进行了评价。在表 2 中，我们进行了两种方法的一些比较。

表 2 DQCGA 和 OPA 的实用性比较

方法	划分时间	额外资源	离线功能	可调整性	实时控制
DQCGA	0.160	无	较强	有	无
OPA	0.330	有	受限	有	无

从表 2 中可以看出，两种方法各自有着不同的优势和劣势。对于算法性能的评估，主要包括执行时间和资源消耗等因素，OPA 方法是在特殊的平台上实现其在线反馈机制的，因此需要额外的资源来对动态环境做精确的模拟，相应的划分算法的执行速度就会下降；而本文算法对动态环境的变化做了部分合理的预测（T 参数的设定），因此在大多数动态环境中能对划分目标有一定的先验知识，保证了算法的执行速度，减少了跟踪动态环境的资源消耗。但是，OPA 的在线反馈机制所提供的实时控制的能力在少数特定的场合还是有独特的用处，能更精确地对动态环境进行跟踪。总之，我们的方法有着更快的划分速度和更少的资源消耗，而 OPA 方法有在线控制的能力。

4.4 适应性

为了考查算法对问题的适应性，我们利用 TGFF 方法设计了一些不同规模的问题。节点数由 30 开始，每次增加 30 个，直至 300。动态参数设定为较为平均的 $r=0.75$，$\tau=20$，来观察算法的求解速度和 F_{avg} 的变化情况。实验结果按照 30 个节点的结果归一化，如图 3 所示。

随着节点数的线性增长，平均适应度保持较平稳的线性增长，算法执行时间呈较小阶次的指数增长，在实际可能的情况中处于可以接受的范围，说明算法对于不同规模的问题具有较好的适应性。

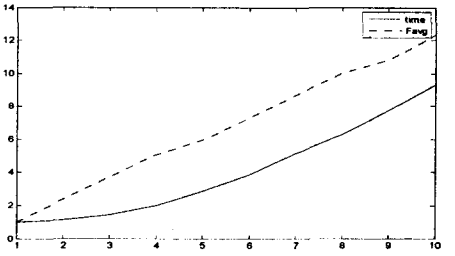

图 3 随问题规模增加算法性能曲线

结束语 本文研究了动态软硬件划分问题，提出了一种基于 DQCGA 算法的解决方法。通过和一种已有方法 OPA 的比较，分析了该方法在有效性、稳定性、实用性方面的性能。一系列实验结果表明，我们的方法对于解决动态软硬件划分问题是可行而高效的，在有效性、稳定和划分时间上都更胜一筹，对于不同规模的问题也有较好的适应性，但是在动态要素的实时控制和反馈上还稍显不足。

参考文献

[1] Lydecky R, Vahid F. A Configurable Logic Architecture for Dynamic Hardware/Software Partitioning // Proc of the DATE 2004 Conference. Paris, 2004: 480-485

[2] Stitt G, Lydecky R, Frank V. Dynamic Hardware/Software Partitioning: A First Approach// Proc of 40th ACM/IEEE Conference on Design Automation(DAC), 2003:250-255

[3] Fakhreddine G, Michel A, Mohamed A. An Adaptive On-line HW/SW Partitioning for Soft Real Time Reconfigurable Systems// Proceedings of the 2005 8th Euromicro Conference on Digital System Design (DSD'05). 2005: 379-382

[4] Yang Shengxiang, Yao Xin. Experimental study on population-based incremental learning algorithms for dynamic optimization problems. Soft Comput ,2005,9: 815-834

[5] Yang S. Non-Stationary Problem Optimization Using the Primal-Dual Genetic Algorithm// Proc of the 2003 Congress on Evolutionary Computation. vol 3. 2003:2246-2253

[6] Dick R P, Rhodesy D L, Wolf W. TGFF: Task Graphs for Free// 6th International Workshop on Hardware/Software Co-design. 1998:97-101

[7] 李斌，庄镇泉，等. 量子概率编码遗传算法及其应用. 电子与信息学报，2005,27 (5) :805