基于混合杂交与间歇变异的约束优化演化算法*)

周永华　毛宗源
(华南理工大学自动化科学与工程学院　广州510640)

A Constrained Evolutionary Algorithm Based on Hybrid Crossovers and Intermittent Mutation

ZHOU Yong-Hua　MAO Zong-Yuan
(College of Automation Science and Engineering, South China University of Technology, Guangzhou 510640)

Abstract In solving constrained optimization problems with genetic algorithms, more emphases are laid on handling constraints than increasing the search capability of algorithms, which often leed to unsatisfied results as reported in most literatures. This paper proposes a new evolutionary algorithm for constrained optimization, emphasizing more on increasing the search capability of the algorithm by means of hybrid crossovers and intermittent mutation while adopting a simple constraint handling technique called direct comparison. Numerical experiments and comparisons show the effectiveness of the proposed algorithm.

Keywords Constrained optimization, Evolutionary algorithms, Hybrid crossovers, Intermittent mutation

1　引言

许多实际的优化问题带有不等式和/或等式约束，这类问题一般可以描述为如下的形式：

$$\begin{aligned}&\max \quad f(x)\\&\text{s. t.} \quad g_j(x)\geqslant 0;\ j=1,2,\cdots,J\\&\qquad h_k(x)=0;\ k=1,2,\cdots,K.\\&\qquad l_i\leqslant x_i\leqslant u_i;\ i=1,2,\cdots,n.\end{aligned}\tag{1}$$

其中，$x=(x_1,x_2,\cdots,x_n)$是n维实向量。目标函数f；约束函数$g_1,g_2,\cdots,g_J$；$h_1,h_2,\cdots,h_K$为R^n上的实值函数。称(1)式为约束优化问题。

通常认为，用遗传算法求解约束优化问题时，处理好约束是获得好的优化结果的关键。现有的许多研究是以此为出发点展开的，并已提出了若干种适合于遗传算法的约束处理方法，它们可分为以下四类：罚函数法；可行解和不可行解区别处理法；解的可行性保持法；混合法[1]。然而，从有关文献，如文[2]所报告的实算结果来看，绝大多数算法存在以下问题，解的精度低、不稳定，甚至还存在不同程度的违约情况。

显然，只有好的约束处理方法是不够的。一种搜索算法要获得好的搜索效果最重要的是设法提高其搜索能力。基于这一点，本文在实数编码遗传算法的基础上提出了一种新的约束优化演化算法。在算法中，通过混合使用多种扩展杂交算子并辅之以间歇变异来提高搜索能力；根据约束条件构造一个违约函数并按照它直接比较个体的违约程度，简化对约束的处理，而且使用遗传操作算子时也无需顾及约束条件。

2　基于混合杂交与间歇变异的约束优化演化算法

2.1　约束的处理

对约束优化问题式(1)，定义如下的违约函数：

$$G(x)=\sum_{j=1}^{K+J}\max\{0,-g_j(x)\}\tag{2}$$

在(2)式中，已将问题(1)的等式约束转化为不等式约束来处理：

$$g_{k+J}(x)=\varepsilon-|h_k(x)|\geqslant 0;\ k=1,2,\cdots,K.\tag{3}$$

其中ε为给定的正小量。个体x的违约程度用$G(x)$的值来衡量。

两个个体优劣的比较准则如下：按(2)式计算两个体的违约函数值，如果两者的违约函数值不相等，则违约函数值小者为优；如果相等，则按(1)式的目标函数值比较，目标函数值大者为优。

上述处理约束的方法称为直接比较法。采用这种方法时，群体的演化过程显然由目标函数的最大化和违约函数的最小化两个子过程组成。

2.2　杂交算子

基于混合杂交与间歇变异的约束优化演化算法(简记作CEA/HCIM)混合使用了十个扩展杂交算子。为便于介绍，设两个父体为$p^1=(p_1^1,p_2^1,\cdots,p_n^1)$，$p^2=(p_1^2,p_2^2,\cdots,p_n^2)$，杂交后得到的两个子体为$s^1=(s_1^1,s_2^1,\cdots,s_n^1)$，$s_2=(s_1^2,s_2^2,\cdots,s_n^2)$。

扩展整体算术杂交算子。在$[-d,1+d]$(本文取$d=1$，下同)内均匀独立地生成n个随机数$a_1,a_2,\cdots,a_n$，则两个子体定义为

$$s_i^1=a_ip_i^1+(1-a_i)p_i^2=p_i^2+a_i(p_i^1-p_i^2)\tag{4}$$

$$s_i^2=a_ip_i^2+(1-a_i)p_i^1=p_i^1+a_i(p_i^2-p_i^1)\tag{5}$$

其中$i=1,2,\cdots,n$。当取$a_1=a_2=\cdots=a_n=a$，就得到扩展线性杂交算子。

扩展部分算术杂交算子。在两父体中选择第k分量以后的所有分量，然后在$[-d,1+d]$内均匀独立地生成$n-k$个随机数$a_{k+1},a_{k+2},\cdots,a_n$，则两个子体定义为

$$s_i^1=\begin{cases}p_i^1 & i=1,\cdots,k\\ a_ip_i^1+(1-a_i)p_i^2 & i=k+1,\cdots,n\end{cases}\tag{6,7}$$

$$s_i^2=\begin{cases}p_i^2 & i=1,\cdots,k\\ a_ip_i^2+(1-a_i)p_i^1 & i=k+1,\cdots,n\end{cases}\tag{8,9}$$

当取$a_{k+1}=a_{k+2}=\cdots=a_n=a$，就得到扩展线性部分算术杂交

*)基金项目：广东省自然科学基金(011626)项目资助。周永华　高级工程师，博士研究生，研究领域为演化计算。毛宗源　教授，博士生导师，研究领域为人工智能与智能控制，模糊控制，演化计算等。

算子。

类似地，可以定义互补扩展部分算术杂交算子和互补扩展线性部分算术杂交算子。在两父体中选择第 k 分量以前的所有分量(包括第 k 分量)，然后按上述方法定义。

扩展均匀算术杂交算子。随机生成一个与父体等长的 n 位二进制串，称之为模板，其中每一位均按相同的概率(本文取为0.5)置为1或0。若某一位为1，则两子体对应分量按(11)和(13)式计算，否则保持不变。假设模板中有 m 位为1，分别是 $i_1,i_2,\cdots,i_m$，在$[-d,1+d]$内均匀独立地生成 m 个随机数 $a_{i_1},a_{i_2},\cdots,a_{i_m}$，则两个子体定义为

$$s_i^1=\begin{cases}p_i^1 & i\notin(i_1,i_2,\cdots,i_m) \quad (10)\\ a_{i_k}p_{i_k}^1+(1-a_{i_k})p_{i_k}^2 & i_k\in(i_1,i_2,\cdots,i_m) \quad (11)\end{cases}$$

$$s_i^2=\begin{cases}p_i^2 & i\notin(i_1,i_2,\cdots,i_m) \quad (12)\\ a_{i_k}p_{i_k}^2+(1-a_{i_k})p_{i_k}^1 & i_k\in(i_1,i_2,\cdots,i_m) \quad (13)\end{cases}$$

当取 $a_{i_1}=a_{i_2}=,\cdots,=a_{i_m}=a$，就得到扩展均匀线性杂交算子。

同样地，对于同一个模板，可以定义互补扩展均匀算术杂交算子和互补扩展均匀线性杂交算子。在两父体中选择模板为零的位所对应的分量，然后按上述相同的方法定义。

2.3 混合杂交和选择过程

从当前群体中无重复地随机选择两个个体，设为 p^1 和 p^2。依次执行上述十种杂交操作，从产生的20个子代中选择出最优的一个子代，并判断其是否优于 p^1 和 p^2 中最差的一个，是，则取代之，否则不取代。主要的算法步骤如下：

Step1 从当前群体中无重复地随机选择两个个体 p^1 和 p^2；

Step2 p^1 和 p^2 进行扩展线性杂交；

Step3 p^1 和 p^2 进行扩展整体算术杂交；

Step4 p^1 和 p^2 进行扩展部分算术杂交；

Step5 p^1 和 p^2 进行扩展线性部分算术杂交；

Step6 p^1 和 p^2 进行互补扩展部分算术杂交；

Step7 p^1 和 p^2 进行互补扩展线性部分算术杂交；

Step8 p^1 和 p^2 进行扩展均匀线性杂交杂交；

Step9 p^1 和 p^2 进行互补扩展均匀线性杂交；

Step10 p^1 和 p^2 进行扩展均匀算术杂交；

Step11 p^1 和 p^2 进行互补扩展均匀算术杂交；

Step12 选出杂交后得到的最优的子代 c_{best}；

Step13 若 c_{best} 优于 p^1 和 p^2 中最差的，则用 c_{best} 替换之；

Step14 结束。

扩展杂交不保证子代的每个分量都在规定的区间内，因此每次杂交后要检查子代各分量是否在区间内，如不在则令它等于区间的上限或下限。以上算法没有示出这一步。

2.4 间歇变异

CEA/HCIM 只使用一种变异算子即均匀变异，且变异不是每代都发生的，而是按给定的间隔发生，因而称之为间歇式变异。这是不同于常规的遗传算法的。

2.5 算法描述

CEA/HCIM 的主要步骤如下：

Step1 初始化，确定群体规模 N，变异概率 p_m，变异间隔 M，迭代次数 g，随机产生一个均匀分布的初始群体；

Step2 计算每个个体的目标函数值和违约函数值；

Step3 若迭代次数大于 g，转 Step8；

Step4 执行2.3的混合杂交和选择算法；

Step5 保存最优可行个体及其目标函数值；

Step6 如果迭代次数 $g/M=0$，则：

按概率 p_m 执行均匀变异；

计算每个个体的目标函数值和违约函数值；

Step7 迭代次数增1，转 Step3；

Step8 输出最优可行个体及其目标函数值，结束。

2.6 算法讨论

杂交算子是遗传算法的主要搜索算子，因此 CEA/HCIM 主要从杂交算子入手来提高算法的搜索能力。在 CEA/HCIM 中，使用了10个扩展杂交算子，因而局部搜索方式丰富，有助于充分搜索解空间内各局部区域的不同部位，从而提高算法的局部搜索能力，达到提高算法求解精度的目的。变异是间断发生的，不是每代都发生。虽然变异可避免陷入局部最优解，但是，频繁的变异会干扰杂交阶段群体的最大化(对于目标函数)和最小化(对于违约函数)进程。此外，扩展杂交算子本身也具有一定的探索能力[3]，采用间歇变异能更好地配合它们的搜索；选择算子是遗传算法的基本算子之一，它将搜索导向解空间内有希望的区域，加速算法的收敛，但是也容易引起算法的过早收敛。因此，在 CEA/HCIM 中，杂交前不使用选择算子，两个杂交父体的选择是在群体中无重复地随机选取的，但杂交后父体和子代间要按个体优劣的比较准则进行局部竞争选择，以确保群体的进化。CEA/HCIM 还采用了杂交后保存最优可行个体的策略，以防因最优可行个体变异时可能发生蜕变而无法得到搜索过程的最优可行个体。该策略也是算法收敛的一个前提条件。下面通过数值实验来考察 CEA/HCIM 的性能。

3 数值实验与比较

3.1 测试函数与算法参数设置

1) $\min f_1(x)=5\sum_{i=1}^{4}x_i-5\sum_{i=1}^{4}x_i^2-\sum_{i=5}^{13}x_i$

$$\begin{aligned}\text{s.t.}\quad & 2x_1+2x_2+x_{10}+x_{11}\leqslant 10\\ & 2x_1+2x_3+x_{10}+x_{12}\leqslant 10\\ & 2x_2+2x_3+x_{11}+x_{12}\leqslant 10\\ & -8x_1+x_{10}\leqslant 0\\ & -8x_2+x_{11}\leqslant 0\\ & -8x_3+x_{12}\leqslant 0\\ & -2x_4-x_5+x_{10}\leqslant 0\\ & -2x_6-x_7+x_{11}\leqslant 0\\ & -2x_8-x_9+x_{12}\leqslant 0\\ & 0\leqslant x_i\leqslant 1,i=1,\cdots,9\\ & 0\leqslant x_i\leqslant 100,i=10,11,12\\ & 0\leqslant x_{13}\leqslant 1\end{aligned}$$

函数的全局最小值是-15。

2) $\min f_2(x)=5.3578547x_3^2+0.8356891x_1x_5+37.293239x_1-40792.141$

$$\begin{aligned}\text{s.t.}\quad & 0\leqslant 85.334407+0.0056858x_2x_5+0.0006262x_1x_4-0.0022053x_3x_5\leqslant 92\\ & 90\leqslant 80.51249+0.0071317x_2x_5+0.0029955x_1x_2+0.0021813x_3^2\leqslant 110\\ & 20\leqslant 9.300961+0.0047026x_3x_5+0.0012547x_1x_3+0.0019085x_3x_4\leqslant 25\\ & 78\leqslant x_1\leqslant 102\\ & 33\leqslant x_2\leqslant 45\\ & 27\leqslant x_i\leqslant 45,i=3,4,5\end{aligned}$$

函数的全局最小值是-30665.5。

3) $\min f_3(x)=(x_1-10)^2+5(x_2-12)^2+x_3^4+3(x_4-11)^2+10x_5^6+7x_6^2+x_7^4-4x_6x_7-10x_6-8x_7$

$s.t.\ 127-2x_1^2-3x_2^4-x_3-4x_4^2-5x_5\geqslant 0$

$282-7x_1-3x_2-10x_3^2-x_4+x_5\geqslant 0$

$196-23x_1-x_2^2-6x_6^2+8x_7\geqslant 0$

$-4x_1^2-x_2^2+3x_1x_2-2x_3^2-5x_6+11x_7\geqslant 0$

$-10\leqslant x_i\leqslant 10, i=1,\cdots,7$

函数的全局最小值是680.6300573。

4) $\min f_4(x)=x_1+x_2+x_3$

$s.t.\ 1-0.0025(x_4+x_6)\geqslant 0$

$1-0.0025(x_5+x_7-x_4)\geqslant 0$

$1-0.01(x_8-x_5)\geqslant 0$

$x_1x_6-833.33252x_4-100x_1+83333.333\geqslant 0$

$x_2x_7-1250x_5-x_2x_4+1250x_4\geqslant 0$

$x_3x_8-1250000-x_3x_5+2500x_5\geqslant 0$

$100\leqslant x_1\leqslant 10000$

$1000\leqslant x_i\leqslant 10000, i=2,3$

$10\leqslant x_1\leqslant 1000, i=4,\cdots,8$

函数的全局最小值是7049.330923。

5) $\min f_5(x)=x_1^2+x_2^2+x_1x_2-14x_1-16x_2+(x_3-10)^2+4(x_4-5)^2+(x_5-3)^2+2(x_6-1)^2+5x_7^2+7(x_8-11)^2+2(x_9-10)^2+(x_{10}-7)^2+45$

$s.t.\ 105-4x_1-5x_2+3x_7-9x_8\geqslant 0$

$-10x_1+8x_2+17x_7-2x_8\geqslant 0$

$8x_1-2x_2-5x_9+2x_{10}+12\geqslant 0$

$-3(x_1-2)^2-4(x_2-3)^2-2x_3^2+7x_4+120\geqslant 0$

$-5x_1^2-8x_2-(x_3-6)^2+2x_4+40\geqslant$

$-x_1^2-2(x_2-2)^2+2x_1x_2-14x_5+6x_6\geqslant 0$

$-0.5(x_1-8)^2-2(x_2-4)^2-3x_5^2+x_6+30\geqslant 0$

$3x_1-6x_2-12(x_9-8)^2+7x_{10}\geqslant 0$

$-10\leqslant x_i\leqslant 10, i=1,\cdots,10$

函数的全局最小值是24.3062091。

6) $\min f_6(x)=e^{x_1x_2x_3x_4x_5}$,

$s.t.\ x_1^2+x_2^2+x_3^2+x_4^2+x_5^2=10$,

$x_2x_3-5x_4x_5=0$,

$x_1^3+x_2^3=-1$,

$-2.3\leqslant x_i\leqslant 2.3, i=1,2.$

$-3.2\leqslant x_i\leqslant 3.2, i=3,4,5.$

函数的全局最小值是0.0539498478。

7) $\max f_7(x)=\left|\dfrac{\sum_{i=1}^{n}\cos^4(x_i)-2\prod_{i=1}^{n}\cos^2(x_i)}{\sqrt{\sum_{i=1}^{n}ix_i^2}}\right|$,

$s.t.\ \sum_{i=1}^{n}x_i\leqslant 7.5n$,

$\prod_{i=1}^{n}x_i\geqslant 0.75$,

$0\leqslant x_i\leqslant 10, i=1,2,\cdots,n.\ n=20,50$

函数的全局最大值是未知的，文[2]给出的最好结果是：$n=20$时，为0.80351067；$n=50$时，为0.83319378。

运行CEA/HCIM要预先设置好算法参数，这里给出一些经验的设置界限。群体规模一般不小于200，变异概率$0<p_m<0.5$；变异间隔一般不小于100；迭代次数g一般为几千次以上。对于具体的函数，执行算法几次即可获得一组较好的参数。

对于以上7个函数，群体规模均设为200；变异间隔：对f_1、f_2设为100，对$f_3\sim f_8$设为50000，对$f_7(n=20$和$50)$均设为10000；变异概率：对f_1、f_2设为0.01，对$f_3\sim f_6$设为0.001，对$f_7(n=20$和$50)$均设为1/3；迭代次数：对f_1、f_2设为5000，对$f_3\sim f_5$和$f_7(n=20)$设为200000，对f_6和$f_7(n=50)$设为400000。对f_6，将等式约束转化为不等式约束时，取$\varepsilon=0.0001$。

3.2 结果与比较

数值实验在Celeron III/1GH兼容机上进行，对每个函数，独立运行CEA/HCIM 50次。

表1列出了CEA/HCIM的实算统计结果（全部满足约束）。由于CEA/HCIM被设计为搜索目标函数的最大值，对于最小化问题目标函数要乘－1转换成最大化问题，输出的结果再乘－1还原成最小值。对于最小化问题，将50个结果由小到大排序，第1个最好，第25个中等，第50个最差。对于最大化问题也作类似处理。e≤0.01（优于）项表示求解结果与已知最优值的绝对误差小于等于0.01的次数，或优于已知最优值的次数（括号内）。

表1 CEA/HCIM优化$f_1\sim f_7$的结果

函数	最好	中等	最差	e≤0.01(优于)	函数计算次数
f_1	−14.9999999	−14.9999999	−14.9999999	50	110200
f_2	−30665.5386717	−30665.5386717	−30665.5386717	50	110200
f_3	680.6300573	680.6300573	680.6300573	50	4001000
f_4	7049.2480208	7049.2486827	7469.0461543	(32)	4001000
f_5	24.3062101	24.3088759	24.5382680	29	4001000
f_6	0.0539415	0.4388026	0.4388028	22	8001800
$f_7(20)$	0.8036189	0.8036173	0.7926070	(44)	4004200
$f_7(50)$	0.8350360	0.8317846	0.8220702	(19)	8008200

CEA/HCIM求解f_4和$f_7(n=20$和$50)$的最好结果优于已知的最优值。下面列出相应的解，供参考。

f_4新的最优解为：

$x^*=$（579.3169714，1359.9913019，5109.9397474，182.0185587，295.6024101，217.9814412，286.4161486，395.6024101）。

$f_7(n=20)$新的最优解为：

$x^*=$（3.16228768，3.12841692，3.09488882，3.06140062，3.02788657，2.99381178，2.95850164，2.92181976，0.49501933，0.48845560，0.48214841，0.47691088，0.47135109，0.46542272，0.46171490，

0.45695777, 0.45257384, 0.44824200, 0.44417031, 0.44037421)。

限于篇幅,f_7($n=50$)新的最优解不列出了。f_6的最好结果虽优于已知的最优值,但三个约束的满足情况不如已知最优解,前者只达到0.0001,而后者达到0.0000001,考虑到这点,不能认为这里给出的结果优于已知最优值。

表2列出了 Michalewicz 的 GENOCOP II 求解 f_1,f_3~f_6 10次运行的统计结果和 GENOCOP III 求解 f_7的最好结果[2]。其中 f_5的最好结果对应的解有一些违约。比较表1和表2可见,除了 f_1外,CEA/HCIM 的求解质量优于 GENOCOP 系统。

表2 GENOCOP 优化 f_1,f_3~f_7的结果

函数	最好	中等	最差
f_1	−15.000	−15.000	−15.000
f_3	680.642	680.718	680.955
f_4	7377.976	8206.151	9652.901
f_5	18.917	24.418	44.302
f_6	0.054	0.064	0.557
f_7(20)	0.80351067		
f_7(50)	0.83319378		

表3 Deb 算法优化 f_1~f_7的结果

函数	最好	中等	最差	δ≤1%
f_1	−15.000	−15.000	−13.000	47
f_2	−30665.537	−30665.535	−29846.654	47
f_3	680.634460	680.641720	680.650879	50
f_4	7060.221	7220.026	10230.834	17
f_5	24.37248	24.40940	25.07530	41
f_6	0.053950	0.241289	0.507761	19
f_7(20)	0.7139	0.6623	0.6371	

表3是 Deb 在文[4]报告的求解 f_1~f_6的50次运行统计结果和在文[5]报告的求解 f_7($n=20$)的10次运行统计结果,其中 δ≤1%项表求解结果与已知最优值的相对误差小于等于1%的次数。比较表1和表3可见,CEA/HCIM 的求解质量也优于 Deb 的算法。Deb 在他的算法中采用了简单的约束处理技术,但是没有采取有效措施提高算法的搜索能力,优化的效果也不理想。

由于有关文献没有提供各函数计算次数,算法之间的求解效率无法进行比较。就 CEA/HCIM 自身而言,有些函数的求解效率较高,有些较低。如何更好地兼顾求解质量和求解效率仍需进一步的研究。

结论 本文从提高算法的搜索能力,简化对约束的处理入手,提出了一种新的约束优化演化算法。在算法中,采用混合杂交和间歇变异来提高算法的搜索能力,采用违约函数和个体的直接比较来处理约束,数值实验和比较显示了所提算法的有效性。本文的研究也表明,求解约束优化问题时,不应只限于处理好约束,提高算法的搜索能力对获得好的优化效果也是至关重要的。

参 考 文 献

1 Michalewicz Z, Schoenauer M. Evolutionary algorithms for constrained parameter optimization problems. Evolutionary Computation 1996, 4(1): 1~32

2 [美] Z. 米凯利维茨著,周家驹,何险峰译. 演化程序——遗传算法和数据编码的结合. 北京:科学出版社,2000

3 Herrera F, Lozano M. Gradual Distributed Real-Coded Genetic Algorithms. IEEE Trans on Evolutionary Computation, 2000, 5 (1): 43~63

4 Deb K. An efficient constraint handling method for genetic algorithms. Computer methods in applied mechanics and engineering, 2000, 186: 311~338

5 Deb K, Agrawal S. A niched-penalty approach for constraint handling in genetic algorithms. In: Montana, D ,ed. Proceedings of the ICANNGA-99. Portoroz, Slovenia, 1999. 234~239

(上接第18页)

7 Yao Weili, He Xudong, Mapping Petri Nets to concurrent programs in CC++. Information and Software Technology, 1997, 39: 485~495

8 Hong J-E, Bae D-H. Software modeling and analysis using a hierarchical object-oriented Petri net. Information Sciences, 2000, 130: 133~164

9 Peterson J L. Petri Net Theory and the Modeling of System. Prentice-Hall, April 1981

10 Goldsack S J, Kent S J H. Formal Methods and Object Technology. Springer-Verlag, London Limited, 1996

11 Janicki R, Koutny M. Semantics of inhibitor nets. Information and Computation, 1995, 123: 1~16

12 Gerogiannis V C, Kameas A D, Pintelas P E. Comparative study and categorization of high-level pertri nets. The Journal of System and Software, 1998, 43: 133~160

13 He Xudong. PZ nets --- a formal method integrating Petri nets with Z. Information and Software Technology, 2000, 43: 1~18

14 袁崇义. Petri Net 网原理. 北京:电子工业出版社,1998

15 Chandy K M, Misra J. Parallel Program Design: A Foundation. Addison-Wesley, 1989

16 Queinnec. Tailoring UNITY to Distributed Program Design. In: Intl. Workshop on Formal Methods for Parallel Programming: Theory and Applications, April 1998. 820~832

17 Yuan Chongyi, Qu Wanling. UNITY and its Missing Structure. In: Proc. of 3rd publishing company, 1998. Workshop on Advanced Parallel Processing Technologies, Changsha, China, Publising House of Electronics Industry, 1999. 172~176

18 Booch G. Object-oriented Design with Application. Menlo Park, CA, Benjamin Cummings, 1993

19 付弘宇. 从 Petri net, UNITY 到程序语义模型:UniNet:[硕士论文]. 北京大学,2001

20 Zhou Guofu, Yuan Chongyi. Mapping PUNITY to UniNet. To be appeared in Journal of computer science and technology

21 Annette L, Matthias L, Daniel M, Ivana T. Modelling Intra- and Inter-Object Control using Reference Nets. In: Jürgen Ebert and Ulrich Frank, eds. Modellierung 2000, St. Goar, Band 15 von Koblenzer Schriften zur Informatik, Fölbach Verlag, 2000. 89~102

22 Reisig W. Petri nets in software engineering. In advances in Petri nets 1986. Number 225, 1993. 77~87