第 53 卷 第 6 期 厦门大学学报(自然科学版) Vol. 53 No. 6
2014 年 11 月 Journal of Xiamen University (Natural Science) Nov. 2014
doi:10.6043/j.issn.0438-0479.2014.06.011

VLSI 标准单元阵列布局问题的一个高效遗传算法

陈雄峰[1],吴景岚[1,2],朱文兴[2]*
(1.闽江学院计算机科学系,福建 福州 350108;2.福州大学离散数学与理论计算机科学研究中心,福建 福州 350108)

摘要:研究可有效处理几万至百万个单元规模 VLSI 标准单元阵列布局问题的遗传算法,使之能在合理的时间内获得高质量的布局结果.为了提高布局质量,针对布局的二维特性设计了新型线网交叉算子和局部搜索技术,并提出了三阶段算法框架以协调算法的全局搜索和局部搜索.为了降低算法的时间和空间复杂度,使算法可处理大规模问题,采用了交叉算子局部化和小规模种群的思想,同时使用了多种保持种群多样性的策略以提高小规模种群的进化性能.对 Peko suite3、4 标准测试电路的实验结果表明,基于这些策略的遗传算法是有效的.

关键词:标准单元阵列布局;遗传算法;线网交叉;局部搜索.
中图分类号:TP 18 **文献标志码**:A **文章编号**:0438-0479(2014)06-0797-07

超大规模集成电路(very large scale integration, VLSI)的布局模式根据对布线位置和布局模块的限制不同,可分为全定制、标准单元、门阵列等.单元阵列(cell array)是一种改进的门阵列模式,每个电路单元大小相同,但去掉一般门阵列的垂直通道,跨行电路单元之间的布线必须利用行间布线通道或布线单元,布线单元为电路单元之间的间隔区域且与电路单元大小相同,因而又与标准单元布局模式相似,通常划归标准单元布局模式[1-5].标准单元是指具有相等高度、相等或不等宽度的电路单元,在 ASIC (application specific integrate circuit)设计中较为常用.其中等宽的标准单元通常用于寄存器、高速缓存(cache)和 CMOS 等存储型电路,布局时采用单元阵列布局模式[2-6].文献[3]的研究表明,以新一代移动设备所用计算结构电路为代表,为了满足低成本、低能耗和短设计周期等要求,采用单元阵列布局模式是必然的发展趋势.

就包含阵列模式在内的标准单元布局问题而言,目前主要的算法有解析算法、启发式算法和划分算法等.最新的解析算法通常是基于非线性规划的算法[6-7].启发式算法包括遗传算法和模拟退火等,不采用并行计算的情况下通常仅适用于处理几十至几千个单元规模电路,文献[10]采用了并行计算使得遗传算法处理电路规模达到 6 万~7 万单元,且在合理的时间内布局结果质量均没有超过最新的解析算法[1-12].

文献[13-15]的相关研究成果表明,根据启发式算法本身的优缺点和具体问题的特点,采用针对性策略扬长避短,是使得启发式算法可应用于大规模乃至超大规模问题的有效途径.包括文献[1,8-9,16-19]在内的大量文献表明,未采用针对性设计策略情况下遗传算法所能处理的组合优化问题规模一般在 2 万以下.文献[13]使用了一系列针对性的策略,特别是根据问题特点设计高性能的交叉算子,使得遗传算法在合理的时间内可处理 1 万~20 万规模的旅行商问题并获得高质量的近优解.从 1990 年 Shahookar 等[8]将遗传算法应用于标准单元布局问题以来,基本上没有采用针对性策略,而是与现场可编程门阵列(field-programmable gate array, FPGA)或标准单元布局问题等同处理[1-12,17].本文根据标准单元阵列布局问题的特点,采用了一系列针对性设计策略,使得遗传算法在合理的时间内可有效处理问题规模达几万至百万个单元,且可获得明显优于现有解析算法的布局结果.

1 问题描述

1.1 问题定义

布局前的电路设计阶段已经确定布局所需的信

收稿日期:2013-12-16
基金项目:国家自然科学基金(61170308)
*通信作者:wxzhu@fzu.edu.cn

息,包括:1) 标准单元的个数 n,每个单元 v_i 的高度 h 和宽度 w_i;2) I/O 固定端点(pins) 位置;3) 线网个数 m,以及每个线网 e_j 所连接的单元或 I/O 固定端点;4) 矩形布局区域高度 H 和宽度 $W\left(W\times H\geqslant\sum w_i\times h,\ 1\leqslant i\leqslant n\right)$,以及布局区域上所划分的布局行数 p 和行高 $h_r(h_r\geqslant h)$. 其中,一个线网连接两个或以上的单元或固定端点,一个单元或固定端点可属于多个线网.

布局时将每个单元 v_i 放置在布局区域的某一行 r_k 上,确定单元 v_i 中心位置的坐标(x_i,y_i),要求单元之间互不重叠. 相等宽度$(w_i=w,1\leqslant i\leqslant n)$的标准单元布局即为标准单元阵列布局,如图 1(a)所示,大小相同的电路单元和布线单元放置在布局行上,形成单元阵列模式布局.

1.2 优化目标与适应值函数

目前,布局的所有线网半周长之和(half-perimeter wire-length,HPWL)是最常用的优化目标. 因为优化 HPWL 可间接地优化其他几个指标,如可布线性、时延、功耗和面积等[2,6],HPWL 减小幅度达 2% 即可认为明显提高了布局质量[20]. 一个线网 e_j 的半周长 l_j 是指包围其连接的所有单元或固定端点的最小矩形的半周长,即:

$$l_j=\max_{p,q\in e_j}|x_p-x_q|+\max_{p,q\in e_j}|y_p-y_q|=\max_{p\in e_j}x_p-\min_{p\in e_j}x_p+\max_{p\in e_j}y_p-\min_{p\in e_j}y_p. \tag{1}$$

本文以最小化 HPWL 作为优化目标,表示为:

$$\min\ \mathrm{HPWL}=\sum_{1\leqslant j\leqslant m}l_j=\sum_{1\leqslant j\leqslant m}\left(\max_{p\in e_j}x_p-\min_{p\in e_j}x_p+\max_{p\in e_j}y_p-\min_{p\in e_j}y_p\right). \tag{2}$$

一个布局 P 的 HPWL 越小表示其布局质量越高,所以作为评估布局 P 质量的适应值函数可定义为其 HPWL 的反比例函数,算法中适应值函数定义为:

$$\mathrm{fitness}(P)=[\mathrm{HPWL}_{\text{Theoretical Opt.}}|\mathrm{HPWL}_{\text{Known Opt.}}]/\mathrm{HPWL}_P, \tag{3}$$

其中 $\mathrm{HPWL}_{\text{Theoretical Opt.}}$ 为某一电路实例理论最优布局的 HPWL. 没有 $\mathrm{HPWL}_{\text{Theoretical Opt.}}$ 的实例可取其已知最优值 $\mathrm{HPWL}_{\text{Known Opt.}}$. 二者取值只要保证不同规模布局的适应值可稳定在相同区间如(0.0,1.5),以便于算法的实现,没有其他特别限制.

2 算法设计

2.1 染色体编码与存储策略

遗传算法中的个体通常以一维线性结构形式进行染色体编码,以利于交叉操作的定位和交换,如图 1 所示. 其中 1~12 为大小相同的电路单元编号,斜体 21、22、23、24 为同样大小的布线单元编号. 以蛇形排列的目的是为了使二维布局中物理位置靠近,特别是蛇形弯曲处的单元在一维染色体中也能彼此靠近,以利于提高交叉算子的进化效果. 单元编号顺序编码方式完全满足完备性、健壮性和非冗余性的编码原则[15].

图 1(b)"布局行索引"用于存储二维布局每一行开始位置所对应的一维染色体编码位置的下标,因此可用一维数据结构存储二维布局. 这为算法处理超大规模电路提供了必要的基础条件,实现细节详见 2.4.2 节.

2.2 种群初始化策略

本文算法选用文献[6-7]算法第一阶段非线性规划方法生成的非法布局,以就近原则合法化后(下文简称非法布局)作为初始种群的主体. 10 万个单元以下规模电路实例的初始种群全部由非法布局组成,10 万~21 万个单元规模电路实例的初始种群由非法布局与文献[6-7]解析算法合法化或优化后的布局(下文简称解析算法优化布局)组成,22 万~160 万个单元规模电路实例的初始种群由解析算法优化布局组成.

2.3 算法框架与保持种群多样性策略

1) 算法基本框架

本文遗传算法为交叉算子与局部搜索混合的算法,其中适应值阈值 μ_1、μ_2 的设定对进化效率有重要的影响,经过详细的实验分析,算法中设定 $\mu_1=0.35$,

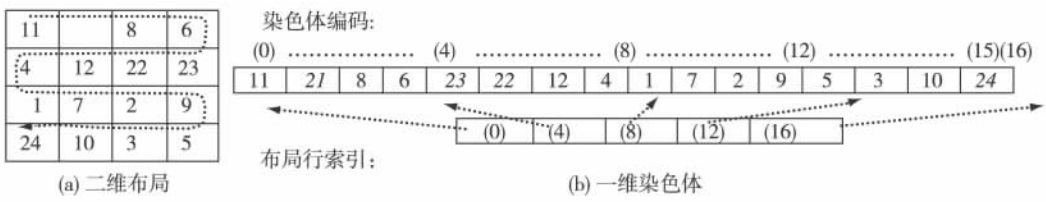

图 1 二维标准单元布局蛇形对应一维染色体

Fig. 1 2-D standard cell placement serpentine transform into 1-D chromosome

$\mu_2=0.64$. 第三阶段中“种群失去多样性”也可依据个体的适应值确定，如“所有 fitness(P_i)/fitness(P_{best})>99.5%”，算法结束条件“终止条件”设定为“一定时间间隔内 HPWL 进化减小量少于 0.01×10^6”. 算法中变异算子采用常用的随机交换单元位置方法，交叉算子、局部搜索策略将分别在后面 2.4、2.5 节中做详细介绍. 算法基本框架如下：

标准单元阵列布局问题遗传算法框架
输入:附有布局所需信息的电路$C(V,E)$
输出:优化布局的所有电路单元位置坐标(x^*,y^*)
1. 初始化布局种群；
2.阶段1: 偏重全局搜索
  Repeat
    (1) 选定双亲；(2) 交叉；(3) 选择生成新种群；
  Until fitness(P_{best})>$\mu1$；
3. 阶段2: 平衡全局与局部搜索
  Repeat
    (1) 选定双亲；(2) 交叉；(3) 选择生成新种群；
    (4) 仅对当前最优个体进行变异和局部搜索；
  Until fitness(P_{best})>$\mu2$；
4. 阶段3: 偏重局部搜索
  Repeat
    (1) 选定双亲；(2) 交叉；(3) 选择生成新种群；
    (4) 对所有个体进行变异和局部搜索；
    (5) 如果“种群失去多样性”就更新种群；
  Until满足“终止条件”；
5. 输出优化布局.

2) 后代选择策略

为了能处理大规模问题，本文遗传算法采用小规模种群[9]；为了能持续在搜索区域寻优，采用了竞争性和爬山(hill climbing)法后代选择策略. 但爬山法会使得种群个体经过若干代进化后均达到相近质量水平而失去多样性，进化速度迅速变慢，因此在使用爬山法的同时须要采用一些有效策略，以使种群在进化过程中尽可能保持多样性[13].

3) 保持种群多样性策略

算法在以下 3 个方面采用了保持种群多样性的策略，其中动态更新种群是主要策略.

i) 动态更新种群策略：种群一旦失去多样性，即所有个体近似于当前最优个体，就更新种群个体. 更新时用新初始个体替换除当前最优个体之外的其他所有个体.

ii) 双亲选择策略：在每一代开始时，对种群个体重新进行随机排列，依次逐个选择当前个体为母亲个体，其循环相邻的下一个个体为父亲个体. 这样，下一代个体仅可能与各自母亲个体相同或相似，与其他个体保持一定的差异性.

iii) 交叉算子使用策略：新型的线网交叉算子(net crossover，NetX)和常用的部分匹配交叉算子(partially-matched crossover，PMX)随机交替使用. 不同类型的交叉算子生成不同类型的孩子个体，使得种群可更长时间保持多样性.

2.4　交叉算子策略

标准单元阵列布局具有明显的二维特征，通用型的 PMX 是针对一维染色体而设计的交叉算子，PMX 交叉时的位置变化不能直接体现二维布局的位置优化. 显然将具有二维结构的线网作为交叉的基因片段，更有利于交叉时保留或生成优良的部分布局，进而利于获得高质量的布局结果并提高进化速度. 基于二维结构和边交叉[13]的思想，本文设计了高性能的 NetX，并作为主要交叉算子与 PMX 随机交替使用. PMX 为常用交叉算子，具体算法流程这里不做赘述，使用时也应依据交叉算子局部化的思想将交叉片段限制为较少个数的连续单元. 实验表明限制在 10 个左右单元其进化效果最佳. 下面详细介绍 NetX.

2.4.1　NetX

NetX 交叉过程如图 2 所示. P_f、P_m 上 1、2、3、4 为将要交叉的一个线网所连接的单元，P_m 上 5、6、7、8 为分别与 P_f 上 1、2、3、4 对应位置的单元. 交叉时将单元 1、2、3、4 在 P_f 上的位置作为双亲进行交叉的对应位置，采用与 PMX 相同机制处理交叉后单元重复或缺失问题. 交叉过程可简化为直接在 P_m 上进行单元交换，即 P_m 上对应位置原有单元 5、6、7、8 分别与 P_m 上线网所连接单元 1、2、3、4 直接进行位置交换，如图 2(a)双箭头虚线所示，交叉生成的 P_{ch} 如图 2(b)所示. 因此，P_{ch} 可视为由 P_m 进化而来，如此可在 P_m 的 HP-

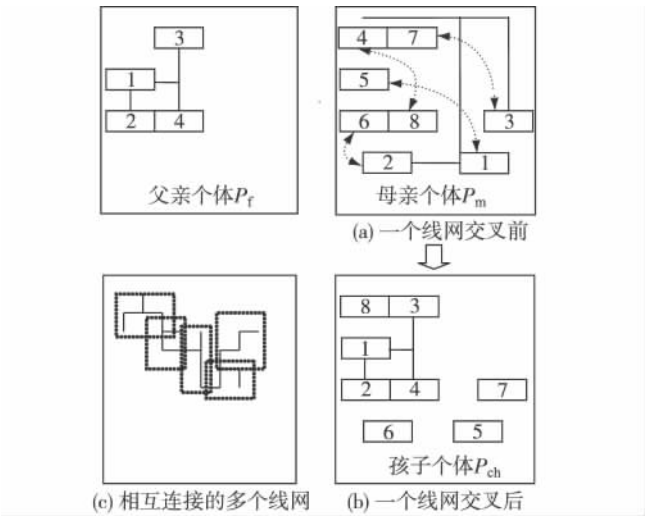

图 2　线网交叉示意图
Fig. 2　NetX diagram

WL 基础上快速计算 P_{ch} 的 HPWL 而不需要重新计算 P_{ch} 所有线网的半周长,详见 2.4.2 小节. 这是算法能在合理时间内处理超大规模标准单元阵列布局问题之关键所在. NetX 算法如下:

线网交叉算子(NetX)算法流程
输入: 附有布局所需信息的电路C(V, E), 电路连接信息 cell_netlist[],net_celllist[], 父亲和母亲个体布局P_f 和P_m, 双亲交叉线网个数crossover_netnumber.
输出: P_f和P_m交叉生成的最优孩子个体布局P_{ch}.
1. net_crossed[]=EMPTY, P_{old}= P_{ch}=P_m;
2. pf_cellposition[]=Collect_Position(P_f), pold_cellposition[]=Collect_Position(P_{old});
3. for (repeat = 1 to crossover_netnumber)
4.current_net=Randomly Select from Adjoin(net_crossed[]) or All First ;
//登记与当前要交叉线网有连接的线网
5. net_connected[] = Collect_ConnectedNet(current_net, net_celllist[], cell_netlist[]);
//登记交叉前有连接线网的HPWL之和
6. connected_oldHPWL = Calculate_PartHPWL(net_connected[], P_{old}) ;
//一个线网交叉, 相当于如图3所示的单元位置交换
7. P_{new}=Swap_CellPosition(pf_cellposition[], pold_cellposition[], P_{old}, current_net) ;
//计算交叉后新布局的总HPWL
8. connected_newHPWL=Calculate_PartHPWL(net_connected[], P_{new}) ;
9. P_{new}_HPWL =P_{old}_HPWL+(connected_newHPWL - connected_oldHPWL) ;
10. if(P_{new}_HPWL < P_{ch}_HPWL) P_{ch}=P_{new}; end if
11. P_{old} =P_{new}; //更新作为母亲个体的布局
12. end for
13. returnP_{ch}.

标准单元阵列布局问题中单元大小相同,单元之间的位置交换不会影响整个布局的合法性,一对双亲进行线网交叉而生成的孩子个体布局一定是合法的. 因此,让一对双亲进行连续多个线网(如 15～60)交叉,可有效保留或生成多个线网所构成的优良的部分布局. 实验观察,尽可能选择相互连接的多个线网更为有效,不容易陷入局部优解. 相互连接的多个线网如图 2(c)所示,其中一个虚矩形框表示一个线网. 一对双亲连续交叉线网个数 crossover_netnumber 的设定对算法效率有非常重要的影响,取值范围在 15～60 之间,布局质量和运行时间相对均衡. 为尽快提高其他个体的布局质量,以利于交叉生成更高质量的孩子个体,当前最优个体作为母亲个体时 crossover_netnumber 取值固定为 15,其他个体依据其与当前最优个体的适应值差距在 15～60 之间适当加大取值,差距越大取值越大. crossover_netnumber 分别取 15、1 的 NetX 与 PMX 收敛性能比较详见 3.2 节.

2.4.2 NetX 的高效实现

1) 用一维数据结构存储二维布局

算法实现时使用 2.1 节所述存储策略,且标准单元阵列布局的每行单元个数 max_col 相同. 因此,每个单元在一维数组存储位置的下标 i(从 0 开始)与其在二维布局所处位置的行、列下标(从 0 开始)之间可进行快速换算或索引查找,其时间和空间代价几乎可忽略不计,具体为:

i) Row[i]=(int)(i/max_col);

ii) Row[i]为偶数时,Col[i]=i MOD max_col; Row[i]为奇数时,Col[i]=(max_col−1)−(i MOD max_col);

2) 快速交换单元位置与计算 HPWL 变化量

收集单元在一维数组存储位置的信息 pf_cellposition[]和 pold_cellposition[](NetX 算法第 2 行),该步骤时间和空间复杂度均仅为 O(n). 而后利用 pf_cellposition[]、pold_cellposition[]进行单元快速定位并交换位置坐标(NetX 算法第 7 行),该步骤时间复杂度仅为 O(1).

HPWL 变化量显然仅与少数几个交换单元所邻接的少数线网有关. 收集登记少数线网(NetX 算法第 5 行)的时间和空间复杂度为 O(1),其中参数 cell_netlist[]、net_celllist[]在算法初始化时一次性收集或计算并暂存于内存. 计算这些少数线网 HPWL 变化量即可快速获得孩子个体的总 HPWL (NetX 算法第 9 行),此时与 HPWL 变化量有关的平均线网个数通常仅是总线网个数的 1%～5%,因此 Calculate_PartHPWL()(NetX 算法第 6,8 行)的计算时间也只是重新计算全部线网总 HPWL 的 1%～5%.

2.5 局部搜索策略

算法根据布局的二维特征将局部搜索邻域设定为一个矩形窗口,在窗口内分别对每一列或行上 4～5 个连续单元进行穷尽搜索. 穷尽搜索较为耗费计算时间,因而在保持适当的进化速度前提下,搜索窗口应尽可能小. 经过实验,当前最优个体的搜索窗口随着布局质量的提高而逐步加大(4×(max_col/12)～4×(max_col/4)),其他个体的搜索窗口固定为相对较小(4×(max_col/12)),其中 max_col 为每行的单元个数. 窗口内每一列进行 1 次穷尽搜索;每一行进行"窗口宽度/5"次穷尽搜索,每次随机选择 5 个连续单元. 每次穷尽搜索的内部算法流程及其高效实现方法与 NetX 算法类似,所不同的只是每趟循环针对的是一

列或一行上 4～5 个连续单元的一种排列，最后在所有可能的排列中保留最优排列.

3　实验结果与分析

算法使用 C 语言编程实现，实验软件环境为 Windows XP，硬件环境为 Intel Core2 CPU 2.10 GHz，RAM 2.0 G. 使用 UCLA VLSI CAD LAB 公布的标准测试电路 Peko suite3、4[11] 进行测试.

3.1　算法优化性能

算法种群规模 $N_p=5$，交叉概率 $R_c=0.8$，变异概率 $R_m=0.05$，算法结束条件为“40 min 的时间间隔 HPWL 进化减小量小于 0.01×10^6”. 实验结果如表 1 和表 2 所列，其中：“非法布局”、“非法布局与解析算法优化布局”和“解析算法优化布局”是指种群初始化策略，如 2.2 节所述；“解析算法布局结果 HPWL”是文献[6-7]中的解析算法所获得的 5 个不同布局结果 HPWL 平均值；“HPWL”是本文算法所获得的 5 个不同布局结果 HPWL 平均值；“HPWL 改进”是“HPWL”较之与“解析算法布局结果 HPWL”的减小百分比.

1）“非法布局”初始种群：10 万个单元以下规模电路均可在 15 h 内得到良好的布局结果，“HPWL 改进”均在 4%以上. 10 万个单元以上规模电路无法在在合理时间(24 h)内获得优于“解析算法布局结果 HPWL”的结果.

2）“非法布局与解析算法优化布局”初始种群：21 万个单元及以下规模电路均可在 4.5 h 内得到较好的布局结果，“HPWL 改进”均在 3%以上. 22 万个单元以上规模电路不能在在合理时间(24 h)内获得优于“解析算法布局结果 HPWL”的结果. 与“非法布局”初始种群相比，处理相同规模电路的“运行时间”缩短 60%～90%，可处理电路规模扩大至 21 万个单元，但“HPWL 改进”下降了 2%左右. 显然，解析算法优化布局使得遗传进化陷入了相对低质量的局部最优解.

3）“解析算法优化布局”初始种群：21 万个单元以上规模电路选取 suite4 Peko07、13、15 分别作为 22

表 1　Peko suite3 测试结果

Tab. 1　Test results of Peko suite3

电路	单元	HPWL/10^6	解析算法布局结果 HPWL/10^6	非法布局		非法布局与解析算法优化布局	
				HPWL/10^6	HPWL 改进/%	HPWL/10^6	HPWL 改进/%
Peko01	12 506	0.822	1.01	0.92	8.91	0.95	5.94
Peko02	19 342	1.27	1.59	1.46	8.18	1.50	5.66
Peko03	22 853	1.51	1.90	1.75	7.89	1.79	5.79
Peko04	27 220	1.76	2.21	2.02	8.60	2.09	5.43
Peko05	28 146	1.95	2.46	2.33	5.28	2.34	4.88
Peko06	32 332	2.07	2.57	2.38	7.39	2.45	4.67
Peko07	45 639	2.89	3.58	3.29	8.10	3.41	4.75
Peko08	51 023	3.15	3.93	3.70	5.85	3.76	4.33
Peko09	53 110	3.65	4.51	4.28	5.10	4.31	4.35
Peko10	68 685	4.75	5.89	5.64	4.24	5.67	3.74
Peko11	70 152	4.72	5.88	5.52	6.12	5.62	4.42
Peko12	70 439	5.02	6.20	5.92	4.52	5.98	3.55
Peko13	83 709	5.89	7.37	6.99	5.16	7.04	4.48
Peko14	147 088	9.03	11.55			10.95	5.19
Peko15	161 187	11.60	14.59			13.91	4.66
Peko16	182 980	12.50	15.71			14.96	4.77
Peko17	184 750	13.50	16.87			16.12	4.45
Peko18	210 341	13.20	16.61			15.76	5.12

万～50 万、50 万～100 万、100 万～160 万规模的代表,均可在 7～10 h 获得优于"解析算法布局结果 HPWL"的结果,"HPWL 改进"均在 2%～3%.

表 2 Peko suite4 中 3 个不同规模电路的测试结果

Tab. 2 Test results of 3 benchmark circuits with different scale in Peko suite4

电路	单元	理论最优布局 HPWL/10^6	解析算法布局结果 HPWL/10^6	解析算法优化布局 HPWL/10^6	解析算法优化布局 HPWL 改进/%
Peko07	456 390	28.9	34.91	33.86	3.01
Peko13	837 090	58.9	72.19	70.43	2.44
Peko15	1 611 870	116.0	147.65	143.11	3.07

3.2 收敛性

为进一步验证三阶段算法框架和 NetX 的有效性,以及 crossover_netnumber 取值的合理性,分别对 suite3 的 Peko01～13 电路进行收敛性测试.所有测试结果均体现出本文三阶段算法框架的收敛性明显优于通常使用的一阶段和二阶段算法框架,特别是在布局结果 HPWL 小于"解析算法布局结果 HPWL"4%之前的收敛性有较大优势;所有测试结果均表明 NetX 收敛性大大优于 PMX,crossover_netnumber=15 时 NetX 的收敛性与 crossover_netnumber=1 时相比具有较大优势.其中对 Peko01 电路的收敛性对比如图 3 所示.

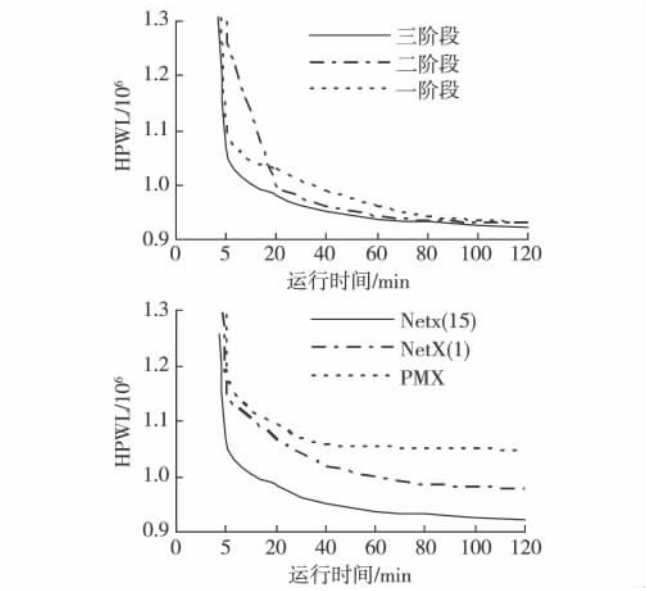

图 3 对 suite3 Peko01 电路的收敛性对比

Fig. 3 Comparisons of convergence for the circuit suite3

4 结 论

本文充分考虑标准单元阵列布局的二维特性,将具有二维结构的线网作为交叉的基因片段,设计并使用新型高效的 NetX,使得布局个体之间交叉时更为有效地保留或生成优良的部分布局,进而提高算法的收敛性能并获得高质量的布局结果.本文提出的三阶段算法框架与常用的一、二阶段算法框架相比,可更为协调地控制算法的全局搜索和局部搜索,从而在更短的运行时间内获得高质量的布局结果.采用交叉算子局部化和高效实现的思想以及小规模种群的策略,可非常有效地降低遗传算法的时间和空间复杂度,动态更新种群策略可大幅提高小规模种群遗传算法的交叉进化效率,从而使得遗传算法可有效处理问题达几万至百万个单元规模.实验表明,这些策略是使得遗传算法能高效地解决 VLSI 标准单元阵列布局问题的关键所在,其中针对性策略的思想也可为研究其他类型大规模问题启发式算法所借鉴.深入分析标准单元非阵列布局问题的特点,提出相应的策略以使得遗传算法能同样高效地获得高质量的非阵列布局结果,将是后续研究的重点.此外,遗传算法的并行实现能够处理更大规模的标准单元阵列布局问题也将是后续研究方向之一.

参考文献:

[1] Kaur J, Kauri M. A survey on various genetic approaches for standard cell placement[J]. International Journal of Computer Technology and Applications, 2013, 4(3): 533-536.

[2] Chu C. Chapter 11: placement in electronic design automation: synthesis, verification, and testing[M]. San Francisco, CA: Elsevier/Morgan Kaufmann, 2008.

[3] Khawam S, Nousias I, Milward M, et al. The reconfigurable instruction cell array[J]. IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 2008, 16(1): 75-85.

[4] Fudos I, Kavousianos X, Markouzis D, et al. Placement and routing in computer aided design of standard cell arrays by exploiting the structure of the interconnection graph[J]. Computer-Aided Design and Applications, 2008, 5(1/2/3/4): 325-337.

[5] 李康. VLSI 物理设计中布局及有约束的布局优化[D]. 成都:电子科技大学, 2010.

[6] Chen J L, Zhu W X. An analytical placer for VLSI stand-

ard cell placement[J]. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2012, 31(8): 1208-1221.

[7] Chang Y W, Jiang Z W, Chen T C. Essential issues in analytical placement algorithms[J]. IPSJ Transactions on System LSI Design Methodology, 2009, 2: 145-166.

[8] Shahookar K, Mazumder P. VLSI cell placement techniques [J]. ACM Computing Surveys, 1991, 23(2): 195-212.

[9] Rcía-Martínez C, Lozano M. Local search based on genetic algorithms: advances in metaheuristics for hard optimization[M]. Berlin: Springer, 2008: 199-221.

[10] Subbaraj P, Sanka S S, Anand S. Parallel genetic algorithm for VLSI standard cell placement[C]// The 2009 International Conference on Advances Computer, Control, and Telecommunication Technologies. India: Institute of Electrical and Electronics Engineers, 2009: 80-84.

[11] Chang C C, Xie M. PEKO suite (placement example with known optimal wirelength). [2013-03-20]. [2011-06-15]. http: // cadlab. cs. ucla. edu/～ pubbench/placement/dw. htm.

[12] Chopra V, Singh A. Multi-objective genetic algorithm for testing MCNC FPGA benchmark circuits[J]. International Journal of Computer Science and Communication Engineering, 2013, 2(1): 74-78.

[13] Nata Y, Kobayashi S. A powerful genetic algorithm using edge assembly crossover for the traveling salesman problem [J]. INFORMS Journal on Computing, 2013, 25(2): 346-363.

[14] Chen J L, Zhu W X, Ali M M. A hybrid simulated annealing algorithm for nonslicing VLSI floorplanning[J]. IEEE Transactions on Systems Man, and Cybernetics-Part C: Applications and Reviews, 2011, 41(4): 544-553.

[15] 郭文忠，陈国龙，彭少君，等. 求解电路划分问题的混合粒子群优化算法[J]. 软件学报, 2011, 22(5): 833-842.

[16] Chen J L, Zhu W X, Peng Z. A heuristic algorithm for the strip packing problem[J]. Journal of Heuristics, 2012, 18: 677-697.

[17] Bunglowala A, Singhi B M, Verma A. Multi-objective optimization of standard cell placement using memetic algorithm[J]. International Journal of Computer Applications, 2011, 19(7): 31-34.

[18] Tang M, Yao X. A memetic algorithm for VLSI floorplanning[J]. IEEE Transactions on Systems Man, and Cybernetics-Part B: Cybernetics, 2007, 37(1): 62-69.

[19] Baghe M, Agrawal S, Silakari S. Survey of metaheuristic algorithms for combinatorial optimization[J]. International Journal of Computer Applications, 2012, 58(19): 21-31.

[20] Kim M, Viswanathan N, Alpert C J, et al. MAPLE: multilevel adaptive placement for mixed-size designs[C]// The 2012 ACM International Symposium on International Symposium on Physical Design. New York, USA: ACM, 2012: 193-200.

An Effective Genetic Algorithm for VLSI Standard Cell Array Placement

CHEN Xiong-feng[1], WU Jing-lan[1,2], ZHU Wen-xing[2]*

(1. Department of Computer Science, Minjiang University, Fuzhou 350108, China; 2. Center for Discrete Mathematics and Theoretical Computer Science, Fuzhou University, Fuzhou 350108, China)

Abstract: This paper presents a genetic algorithm for solving the problem of VLSI standard cell array placement with up to tens of thousands to millions of cells, and for obtaining high quality placement results in a reasonable running time. For producing high quality placement results, a new kind of crossover operator on nets and a new type of local search method for the 2-D placement problem are designed, and an innovative algorithm framework with three phases is proposed to coordinate the global search and the local search of this algorithm. For the purpose of enabling the algorithm to hundle large-scale problem, the ideas of crossover localization, and small size population are adopted to reduce time and space complexities of the genetic algorithm. Meanwhile, various strategies for maintaining population diversity are used to improve the evolution performance of a small size population. Experimental results on Peko suite3 and suite4 benchmark circuits verify that the genetic algorithm with these strategies is efficient.

Key words: standard cell array placement; genetic algorithm; net crossover; local search.