浙江工业大学

本科毕业设计开题报告

（2017 届）

论文题目　基于遗传算法的音乐作曲演示系统研究与实现

作者姓名　陈文帆

指导教师　盛伟国

专业班级　计自 1302

所在学院　计算机科学与技术学院

提交日期　2017 年 3 月

基于遗传算法的音乐演示系统研究与实现

一、选题的背景与意义

1.1 研究开发的目的

算法作曲（Algorithm Composition）或称自动作曲（Automated Composition），是试图使用某种形式的过程，以使人类在利用计算机进行音乐创作时的介入程度达到最小的研究[1]。算法作曲是计算机艺术最重要的领域之一，其代表了计算机应用领域一个真正的历史性突破，因为它第一次试图复制纯粹的、人类所独有的能力：创造力[2]。

目前，算法作曲领域采用的主要技术有马尔可夫链（Markov Chain）、随机过程、专家系统、音乐文法、遗传算法和人工神经网络。

遗传算法（Genetic Algorithm，GA）是一种启发式搜索算法。因此，利用遗传算法来控制乐曲的生成过程，首先必须将作曲过程建模为一个优化问题。在作曲过程中，对给定音乐片段进行一定形式的编码，编码后音乐片段即为遗传算法的概念染色体，然后采用交叉、突变、选择等算子对其进行进化，用适应度函数来衡量进化的结果，如此不断进化，直到找到最终满意的解。

由于遗传算法评估和演化给定音乐片段的能力，其在算法作曲方面扮演着越来越重要的角色。然而，算法作曲存在两个巨大的挑战：

1） 人为地再现人类作曲家的天才与能力的巨大挑战；

2） 对于自动生成的乐曲给定一个正式的、标准的美学评估的困难性。[1]

本课题，是基于遗传算法的音乐演示系统研究与实现，力图充分利用现有的音乐作曲技巧，弥补算法作曲在创造力方面的不足；同时基于乐理知识，并选择性借助交互的方式，来对自动生成的乐曲作符合人类音乐美学的评估。

1.2 国内外研究发展现状

将计算机应用于作曲的事迹，最早可以追溯到上世纪 50 年代：1956 年，Lejaren Hiller 创造了最早的计算机作曲作品《ILLIAC 弦乐四重奏组曲》[3]。

半个多世纪过去，随着计算机技术的发展，算法作曲已经取得了不俗的成

就。各时期、各阶段，均有适应当时科研水平的作曲系统产生。以进化作曲系统论，在过去的几十年间就出现了多个成熟的作曲系统。

1）Horner 与其计算机辅助作曲程序

Horner 最早将遗传算法应用到了计算机作曲中，并设计实现了一个简单的计算机辅助作曲程序[4]。

该程序能够完成主题桥接，从一个原始音乐动机出发，经过一系列变换，最终生成目标动机。系统中定义的变换包括删除音符、旋转、突变、交换等操作。从原始动机到目标动机所经历过的变换操作的集合就是桥。系统的适应度分两步计算，首先衡量经过桥变换后的动机与目标动机的匹配度，这个匹配度既考虑相同音符的比例，也考虑音符顺序的匹配程度；然后进一步计算适应度，考虑音符长度的匹配程度，如果长度也是精确匹配，个体的适应度将达到最大值。

Horner 辅助作曲系统实现了一个卡农式的五声部轮唱主题的生成，在给定原始动机下，各个声部独立进行进化，最终输出经过桥接的音乐序列。

2）Prisco 与其十二音体系作曲

Prisco 利用遗传算法实现了一个基于十二音作曲体系的乐曲创作[5]。

十二音体系是 Arnold Schonberg 创立的无调性音乐体系[6]。在这种作曲方法中，1 个八度里的所有 12 个半音（C，C#，D，D#，E，F，F#，G，G#，A，A#，B）都是平等的，在一个音出现以后，其他 11 个半音出现之前，该音不能重复使用，以免形成调性中心。十二音作曲法中，首先要有一个由 12 个半音组成的音列，称之为种子序列或原形 O（Original），从原形又可以派生出序列的几种变形，包括逆行 R（Retrograde）、倒影 I（Inversion）、倒影逆行 RI（Retrograde of Inversion）等。

3）Biles 与其 GenJam

Biles 应用遗传算法实现了一个实时交互式演奏系统——GenJam[7,8]。其可以对一段爵士旋律片段进行进化，进化的结果用于爵士独奏的材料。

GenJam 中的算法设计了 2 个分层关联种群：乐节种群和乐句种群。一个乐节定义为由 8 个音符组成的小节，而一个乐句包含 4 小节。这 2 个种群各自进行进化，有各自的变异和交叉运算。

GenJam 是目前为止最为成功的进化作曲系统之一，从 1994 年最初设计的一个简单概念模型开始，Biles 不断地对它进行修改完善。现在的 GenJam 已经是一个颇为成熟的交互式实时演奏系统，可以与人类表演者一起进行各种形式的爵士乐演奏。

4）E. Muñoz 与模因作曲

由于各种遗传算法都存在不同程度的缺陷，E. Muñoz 等人为解决单独使用某种遗传算法作曲而产生的问题，提出了模因作曲法[2]。该方法同时使用多种遗传算法，对音乐的多元进化过程进行模仿，组成不同的模因作曲代理。从实验结果来看，该方法有效解决了传统遗传算法的问题，是进化音乐领域一项突破性的技术，不失为未来算法作曲方发展的方向之一。

本课题，将充分考虑上述研究提及的作曲方法，结合本课题实际情况辩证地采纳吸收，以解决算法作曲的固有挑战为目标，实现一个可定制、可扩展、功能强大的音乐演示系统。

二、研究开发的基本内容、目标，拟解决的主要问题或技术关键

2.1 研究目标

本课题旨在，基于遗传算法，结合现有的作曲方法，设计并实现一个可定制、可扩展、功能强大的音乐演示系统，力图以作曲技巧弥补算法作曲在创造力方面的不足；以乐理知识为准则，结合人机交互，解决乐曲评估的困难。

2.2 研究的基本内容

项目的主要设计思路如下。

1）分层关联种群

项目采用了 GenJam 系统的分层种群思想，预分层情况同 GenJam：第一层为小节种群，第二层为乐句种群，两个种群分别进化。由于一个乐句由若干小节组成，安排乐句的进化在小节的进化完成之后，两者之间是相对独立的，各自编码，有着各自的遗传算子以及适应度函数。小节对乐句有影响，但乐句对小节而言是不可知的，即乐句的进化对小节是无影响的。

2）人机交互

为有效解决乐曲评估难的问题，项目在引入乐理知识作为约束规则的同时，加入了交互功能。由于人力资源的紧张，听众只会参与到乐句的评分，而不必对每一个小节做评估。因此，乐句的总体适应度将由基于乐理知识的适应度 $F_1(X)$ 和基于听众评价的适应度 $F_2(X)$ 组成，记作：

$$F(X) = F_1(X) + W_n \bullet F_2(X)$$

其中 W_n 表示基于听众评价的适应度的权重，是关于进化次数 n 的单调递增函数，表示随着进化的不断迭代，听众的打分占比将更大，影响更大。

3）卡农

卡农是一种相对比较适合算法作曲的音乐形式。它是一种音乐谱曲技法，其所有声部虽然都模仿一个声部，但不同高度的声部依一定间隔进入，造成一种此起彼伏，连绵不断的效果。

本项目将首先模仿卡农的作曲技法，参照 Horner 的计算机辅助作曲程序，定义一系列删除音符、旋转、突变、交换等操作，对小节进行进化。 因为不存在目标动机，因此适应度的计算不以 Horner 的计算机辅助作曲程序为依据。

2.3 需要解决的技术难点

（1）音乐的知识表达困难性；

（2）将作曲过程建模为优化问题的技术难点：

1. 编码的困难性；

2. 定义适应度函数的困难。

三、研究开发的方法、技术路线和步骤

(1) 系统平台：Ubuntu 16.04 LTS

(2) 编程语言：Python 2.7.12

Python 是一种解释型、动态语言。其最大的特点是：简单而强大！

Python 具有清晰的语法结构，与 C 语言、Java 等相比，简单得像可执行的伪代码一样。这使得研究开发可以专注于算法与实现本身，而不必纠结于语法细节。

Python 已经自带了许多高级数据类型，如列表（List）、元祖（Tuple）、字典（Dictionary）、集合（Set）、队列（Que）等，无需进一步编程就可以使用。这使得实现抽象的数学概念非常简单。

除标准库之外，Python 还有着极丰富的第三方模块库，为研发工作提供了极大的便利。尤其是 Python 在科学与金融领域的应用，使得一系列科学函数库，如 SciPy 和 NumPy 等得以产生。这些函数库使用底层语言（C 或 Fortran）编写，提供了计算性能；而它们所提供的 Python 语言接口，又增加了代码的可读性。

一年前，Google 开源了 Tensorflow，支持多种语言，但对 Python 支持最好；一个月前，Facebook 开源了 PyTorch，将 Python 置于首要地位（PyTorch is a deep learning framework that puts Python first.）；几天前，Facebook 又开源了数据预测工具 Prophet，同时提供了对 Python 的支持。可以毫不夸张地说，在数据科学领域，Python 基本坐稳了头把交椅。

(3) 系统开发工具：PyCharm Community 2016.3

PyCharm 是由 JetBrains 开发的一款 Python IDE，其带有一整套可以帮助用户提升开发效率的工具，包括：调试、语法高亮、项目管理、代码跳转、

智能提示、自动补全、单元测试、版本控制等等。PyCharm 社区版提供了对数据科学的完整支持，将为本次项目开发带来极大便利。

(4) Main Lib： DEAP

DEAP（**D**istributed **E**volutionary **A**lgorithms in **P**ython）是一个创新型进化算法框架，可用于快速开发原型或验证猜想。它力图使算法更清晰，使数据结构更透明。此外，它还与并行机制完美协调，比如多处理（Multi-Processing）。DEAP 包含但不限于以下功能：

- 基于多种数据结构的遗传算法
- 使用前缀树的遗传编程
- 进化策略（包括 CMA-ES）
- 多目标优化（NSGA-II，SPEA2，MO-CMA-ES）
- 多种群共同演化（协作与竞争）
- 并行进化

(5) 实现步骤

编码：编码的目的是建立表现型到基因型的映射关系，将实际问题转化为算法形式。

本项目，涉及到小节与乐句的同时编码，一种可行的编码方式是，均采用数字列表的编码形式。以小节编码为例：

小节由音符组成，要使音符能较好地表现音乐形式，其至少需要涉及音级与时值的编码。采用三位十进制编码方式，前两位表示音级，第三位表示时值，比如小节 $\underline{\underline{4\dot{3}3}}\,6$ 可编码为 [423，320，320，615][9]。

种群初始化：种群初始规模大小对遗传算法有一定影响：初始规模太大，有利于增大解的搜索空间，但计算量将成几何倍增加；初始规模太小，程序运行速度快，但搜索容易陷入局部最优。

对于本课题，种群初始规模大小，有待实验的进一步研究。

项目采用分层的两层关联种群，需要对小节和乐句分别进行种群初始化，乐句的初始化是基于小节的初始化的。以小节的初始化为例：

初始化小节种群可以有两种方式：一种是采用现有的某音乐的小节，以

同类别音乐为优；一种是使用计算机随机生成。实际上，现有的音乐片段基本上已经比较成熟了，再进行进化的意义不大[10]。

适应度计算：适应度函数是遗传算法作曲的关键之一，它可以直接衡量算法的优劣，并且对算法的性能也起到了决定性最用。

对本项目而言，小节的适应度，完全按照乐理知识计算得，记作 $G(X)$；乐句的适应度，由基于乐理知识的适应度和听众打分两部分组成，记作 $F(X)=F_1(X)+W_n \bullet F_2(X)$。

选择：基于适应度在种群中按照一定规则对优秀的个体进行挑选，直接选入下一代，或需进行遗传操作，产生新个体。

对于本项目，选择可以模拟音乐家根据自己的知识和经验选择用于后续创作的种子。

交叉：通过交叉可将不同个体中的遗传信息进行随机交换，产生新个体，大大提升搜索能力。

对于本项目，交叉可以模拟音乐家综合不同乐曲的长处，来创作出更好的乐曲。

突变：配合交叉操作，变异可以让算法跳出局部最优解。此外，作为辅助算子，通过改变染色体中局部基因位的值，能改善算法的局部搜索能力。

对于本项目，突变可以模拟音乐家再创作乐曲时的灵感。

终止条件：项目设置两种终止的方式：一是进化迭代到预设的代数，二是算法提前得到满足要求的解，此处即为得到满足预期的乐曲。终止条件采用“或”的关系，任一条件被满足，就终止程序，输出结果。

四、研究工作总体安排与时间进度

任务序号	起 止 时 间	阶 段 任 务 要 点
1	2017.1.10-2017.1.25	对项目课题背景知识进行充分了解，并查找相关文献资料
2	2017.1.26-2017.3.1	查阅中英文文献资料，完成文献综述、开题报告和外文翻译
4	2017.3.2-2017.4.15	学习 DEAP 库，程序设计
5	2017.4.16-2017.5.20	算法改进，系统扩展
6	2017.5.26-2017.5.29	整理相关资料、完成毕业论文
7	2017.5.30-2017.6.16	上交毕业论文、准备毕业答辩

参考文献：

[1] Alpen A. Techniques for algorithmic composition of music. 1995. http://alum.hampshire.edu/~adaF92/algocomp/algocomp95.html

[2] E. Munoz, J. Cadenas, Y. Ong, G. Acampora, "Memetic music composition", IEEE Transactions on Evolutionary Computation, vol. 20, no. 1, pp. 1-15, 2016.

[3] Järvelainen H. Algorithmic musical composition. 2000. http://www.tml.tkk.fi/Studies/Tik-111.080/2000/papers/hanna/alco.pdf

[4] HORNERA, GOLDBERGDE . Genetic algorithm and computer-assisted music composition [C] . Proceeding of the 4th International Conference on Genetic Algorithm. San Diego:Morgan Kaufman Publishers, 1991.

[5] DE-PRISCOR , GIANLUCA Z , ROCCO Z . A Genetic Algorithm for Dodecaphonic Compositions [M] . Applications of Evolutionary Computation, Heidelberg:Springer, 2011.

[6] 郑英烈. 序列音乐写作教程 [M] . 上海:上海音乐出版社, 2007.

[7] BILES J A . GenJam: A Genetic Algorithm for Generating Jazz Solos [C] . Proceedings of the International Computer Music Conference. San Francisco:International Computer Music Association 1994.

[8] BILES J A. Autonomous GenJam: Eliminating the Fitness Bottleneck by Eliminating Fitness [C] . Proceedings of the 2001 Genetic and Evolutionary Computation Conference Workshop Program . San Francisco:GECCO, 2001.

[9] 黄澄宇. 运用遗传算法进行智能音乐作曲研究[J]. 微型电脑应用, 2014, 30(3).

[10] 曹西征,张爱丽,徐久成.基于遗传算法的智能作曲技术研究,计算机工程与应用,2014,(32):79-84.