

FriendlyMusic: Fixing the un-Friendliness of Classical Music with Constraint Satisfaction Programming

Graham Percival

1 Introduction

Western classical music is generally seen as elitist and difficult to perform. Many children spend a few years learning classical music on the piano, but few teenager and adults play classical music for fun. By and large, this genre of music is relegated to a few hundred professional musicians in each city. This is an unusual state of affairs – two centuries ago, a typical evening’s entertainment for a middle-class family would consist of playing the music of Mozart and Haydn as amateur musicians! The public’s appetite for playing music has not diminished – the explosive popularity of games such as Guitar Hero and Rock Band are evidence of this.

Unfortunately, most classical music for small ensembles was written for a strict set of instruments with a uniform level of ability. The first violin or flute parts will be more difficult than other instruments, but that is the extent of the variation in ability. There are no string quartets with very challenging cello parts but easy violin parts, for example!

As a result, amateur musicians must find other players of similar ability and specific instruments. For example, viola player must find two violinists and one cellist. If the viola player has a good friend who plays trumpet, they cannot play in small groups together – the smallest ensemble which contains both viola and trumpet is a chamber orchestra of 30 people!

This aspect of classical music makes it very difficult for members of the same social group to play together. There is nothing wrong with making new friends by organizing musical ensembles, but the chances of any pre-existing group

of friends being able to play music together is very low. *FriendlyMusic* solves this problem by creating sheet music in the style of the composer desired, with any number of instruments played by musicians of any skill level.

2 Technical Overview

FriendlyMusic is based on David Cope’s Experiments in Musical Intelligence (EMI) and our previous work on ability level specific Constraint Satisfaction Programming. First, a musical phrase is generated with the algorithms in EMI. Second, that musical phrase is analyzed according to our ability constraints and is rejected if it breaks a constraint.

For example, consider the generation of a work for solo violin, in the style of Mozart, for an amateur violinist with two years of experience. The first phrase that EMI generates has the violinist shifting into fourth position on the E string – but our amateur violinist is not comfortable playing that high. The ability constraint module rejects this musical phrase, so EMI generates a new phrase. This new phrase stays in first and third position, which does not break any of our ability constraints. The phrase is accepted and the process continues to the next musical phrase.

This high-level description glosses over two vitally important details: first, the specification of ability constraints, and two, the optimizations we performed to create sheet music within a reasonable period of time. These will be detailed in the next two sections.