DEPARTMENT OF ENVIRONMENT,
TECHNOLOGY AND TECHNOLOGY MANAGEMENT

A variable neighbourhood search algorithm to generate first species counterpoint musical scores

Dorien Herremans & Kenneth Sörensen

A variable neighbourhood search algorithm to generate first species counterpoint musical scores

Dorien Herremans & Kenneth Sörensen

RESEARCH PAPER 2011-017
OCTOBER 2010

D/2011/1169/017

A variable neighbourhood search algorithm to generate first species counterpoint musical scores

Dorien Herremans & Prof. Dr. Kenneth Sörensen

University of Antwerp
Faculty of Applied Economics
Operations Research Group ANT/OR

Working Paper

October 26, 2011

In this paper a variable neighbourhood search (VNS) algorithm is developed that can generate musical fragments of arbitrary length consisting of a *cantus firmus* and a *first species counterpoint* melody. The objective function of the algorithm is based on a quantification of existing counterpoint rules. The VNS algorithm developed in this paper is a local search algorithm that starts from a randomly generated melody and improves it by changing one or two notes at a time.

A thorough parametric analysis of the VNS reveals the significance of the algorithm's parameters on the quality of the composed fragment, as well as their optimal settings.

The VNS algorithm has been implemented in a user-friendly software environment for composition, called *Optimuse*. Optimuse allows a user to specify a number of characteristics such as length, key, and mode. Based on this information, Optimuse "composes" both a cantus firmus and a first species counterpoint melody. Alternatively, the user may specify a cantus firmus, and let Optimuse compose only an accompanying first species counterpoint melody.

1. Introduction

Computers and music go hand in hand these days: music is stored digitally and often played by electronic instruments. *Computer aided composing* (CAC) is a relatively new research area that takes the current relationship between music and computers one step further, allowing the computer to aid a composer or even generate an original score. The idea of automatic music generation is not new, and one of the earliest "automatic composition" methods is due to Mozart. In his *Musikalisches Würfelspiel* (Musical Dice Game), a number of small musical fragments are combined by chance to generate a Minuet [Boenn et al., 2009].

In this paper an algorithm is developed for automatic composition of first species counterpoint music. It is argued that composing music can—at least partially—be regarded as a *combinatorial optimisation problem*, in which one or more melodies are "optimised" to adhere to the "rules" of their specific musical style. To this end, these rules need to be formalised and quantified, so that the algorithm can judge whether one automatically composed melody is better than another. Although every musical genre has its own rules, these have generally not been formally written down, nor are they usually very strict. The main reason for the choice of the counterpoint style, is that in this style there do exist formal, written rules [Fux and Mann, 1971] that are not too difficult to quantify. This paper therefore focuses on automatic composition of one or two melodies that adhere to all Fuxian counterpoint rules. A similar algorithm could then in principle be used to compose melodies from other styles, using rules that have been data-mined from a musical database, but this is beyond the scope of this paper.

Composing music is computationally complex, especially since the number of possible melodies rises exponentially with the length (number of notes) of the melody to compose. *Heuristic* or *metaheuristic* optimisation algorithms therefore present the most promising approaches to generate high-quality melodies in a reasonable amount of time. Contrary to *exact methods*, (meta)heuristics do not necessarily return the *optimal* (i.e., best possible) solution [Blum and Roli, 2003], but use a variety of strategies, some of which seemingly unrelated to optimisation (such as natural selection or the cooling of a crystalline solid), to find good solutions. A large number of metaheuristics has been proposed, which can be roughly divided into three categories. *Local search* metaheuristics (tabu search, variable neighbourhood search, iterated local search, ...) iteratively improve a single solution. *Constructive* metaheuristics (GRASP, ant colony optimisation, ...) build a solution from its constituting parts. *Population-based* metaheuristics (genetic/evolutionary algorithms, path relinking, ...) maintain a set (usually called population) of solutions and *combine* solutions from this set into new ones.

Most methods for CAC found in the literature belong to the class of population-based, more specifically *evolutionary*, algorithms [Nierhaus, 2009] that find inspiration in the process of natural evolution. The use of other types (constructive and/or local search) of metaheuristics has remained relatively unexplored.

Several CAC methods rely on user input. GenJam [Biles, 2001] uses an evolutionary algorithm to optimise monophonic jazz solos, relative to a given chord progression. The quality of a solo is calculated based on feedback from a human mentor. This creates a bottleneck because the mentor needs to listen to each solo and take the time to evaluate it. A comparable human fitness bottleneck also arises in the CONGA system, a rhythmic pattern generator based on an evolutionary algorithm. According to the authors of the system, the lack of a quantifiable fitness function slows down the composing process and places a psychological burden on the user who listens and has to determine the fitness value [Tokui and Iba, 2000].

Other CAC algorithms have attempted to eliminate the human bottleneck and define an objective function that can be automatically calculated. Geis and Middendorf [2007] develop a function that indicates how well a given fragment adheres to five harmonic rules of baroque music. This function is then used as the objective function of an ant colony metaheuristic. Orchidée is a multiobjective genetic algorithm that combines musical fragments (from a database) in order to efficiently orchestrate them. The goal of this approach is to find a combination of samples that, when played together, sounds as close as possible to the target timbre [Carpentier et al., 2010].

Notwithstanding the above examples, very little research can be found on the automatic scoring of a melody. As mentioned, formal, written-down rules for a given musical style typically do not exist. An exception is counterpoint music: the formalised and restrictive nature of this style is exemplified by the fact that very formal rules exist for both a single melody and the harmony between several melodies. Counterpoint music starts from a single melody called the *cantus firmus* ("fixed song"), composed according to a set of melodic rules and adds a second melody (the *counterpoint*). The counterpoint is composed according to a similar set of melodic rules, but also needs to adhere to a set of *harmonic* rules, that describe the relationship between the cantus firmus and the counterpoint [Norden, 1969]. Aguilera et al. [2010] use probabilistic logic to generate counterpoint music in C major, based on a given cantus firmus. This application only assesses the harmonic characteristics of counterpoint, the melodic aspect is ignored. GPmuse implements a genetic algorithm that "optimises florid counterpoint in C major, by using a fitness function based on the homework problems described by Fux [Polito et al., 1997].

Although other studies have used Fux's rules in order to calculate the counterpoint-quality of a musical fragment, the exact way in which they are quantified is usually not mentioned. Some studies only focus on a few of the most important rules instead of looking at the entire theory [Anders, 2007]. The next section will discuss the Fuxian rules for first species counterpoint and how they have been quantified to determine the "quality" (i.e., adherence to the rules) of a fragment consisting of a cantus firmus and a counterpoint melody. This quality score will be used as the objective function of a local search metaheuristic, that is developed in Section 3.

2. Quantifying musical quality

This paper focuses on a specific type of polyphonic classical music called first species counterpoint. A polyphonic musical fragment consists of two or more voices, also called parts or *melodies*. The term "counterpoint" refers to the relationship between those melodies. The complexities that arise from playing different notes at the same time has given rise to a very restrictive and formalised set of rules on how to compose polyphonic music.

The species counterpoint rules written by Johann Fux in 1725 are one of the most restrictive sets of rules for composing classical music. This system was originally developed as a pedagogical tool, for students learning how to compose. It consists of five *species* or levels (first, second, third, fourth and the most complex is called *florid* counterpoint) that are taught in sequence. Each species adds more complexity to the music, e.g., more notes per part, or different rhythmical structure. The Fuxian counterpoint rules shape the basis of much of the classical music that is composed today [Norden, 1969].

First species counterpoint is the most restrictive species. It is often referred to as "note-against-note" counterpoint because only whole notes are allowed [Adiloglu and Alpaslan, 2007]. The rules of first species counterpoint apply to a polyphonic musical fragment consisting of two parts or melodies: a *cantus firmus* and a *counterpoint* melody. The cantus firmus is a base melody to which the counterpoint melody is added. This latter melody takes into account not only the melodic transitions between notes within the melody (i.e., the *horizontal* aspect of the music), but also the harmonic interplay between the two melodies (i.e., the *vertical* aspect) [Aguilera et al., 2010]. The fact that Fux's rules are the most restrictive set of rules that exist in music theory, makes them ideal to use as quantifiers of quality in an optimisation context.

In this work, the Fuxian rules as described by Salzer and Schachter [1969] are used to model the objective function of a music composition optimisation problem. Contrary to existing work, the objective function developed in this paper spans Fux's complete theory, and is based on the qualitative Fuxian rules formalised by Salzer and Schachter [1969]. Full details are given in appendix A. Some examples rules are listed below.

- Each large leap should be followed by stepwise motion in the opposite direction.
- Only consonant intervals are allowed.
- The climax should be melodically consonant with the tonic.
- All perfect intervals should be approached by contrary or oblique motion.

In order to determine how well a generated musical fragment s, consisting of a cantus firmus (CF) and a counterpoint (CP) melody, adheres to the counterpoint style, the Fuxian descriptive rules are quantified. The objective functions for the CF and CP melodies are represented in equations (1) and (2) respectively. The objective function for the entire fragment is the sum of these two, as shown in equation (3). Both objective functions are the sum of several subscores, one per rule, whereby each subscore takes a value between 0 and 1. The lower the score, the better the fragment adheres to the rule. A "perfect" musical fragment therefore has an objective function value of 0. The relative importance of a subscore is determined by its weight. The weights a_i (for the subscores of "horizontal" rules) and b_i (for the subscores of "vertical" rules) can be set by the composer to calculate the total score. The score for the CF melody (f_{CF}) only consists of a horizontal aspect, whereas that of the CP melody (f_{CF}), takes into account the horizontal scores of the CP melody as well as the vertical scores, that evaluate the interaction between the two melodies.

$$f_{\mathrm{CF}}(s) = \underbrace{\sum_i a_i.\mathrm{subscore}_i^H(s)}_{\text{horizontal aspect CF}} \tag{1}$$

$$f_{\mathrm{CP}}(s) = \underbrace{\sum_i a_i.\mathrm{subscore}_i^H(s)}_{\text{horizontal aspect CP}} + \underbrace{\sum_j b_j.\mathrm{subscore}_j^V(s)}_{\text{vertical aspect CP}} \tag{2}$$

$$f(s) = f_{\mathrm{CF}}(s) + f_{\mathrm{CP}}(s) \tag{3}$$

The rules used in the above objective function, can be seen as "soft" constraints. Although the algorithm developed in the next section tries to find a musical fragment that violates as few of these rules as possible (and by as little as possible) a fragment that breaks some of the rules is not necessarily "infeasible". For pieces of arbitrary length, it is additionally not known whether it is possible to adhere to all the rules. Music theory also imposes a set of "hard rules", that have been implemented as constraints in the optimisation problem. These rules, such as "all notes are from the tonal set" and "all notes are whole notes", always need to be satisfied, in order to obtain a feasible fragment.

The very restrictive nature of counterpoint and the large number of rules that all need to be satisfied at the same time, make it a difficult craft for human composers to master. For several of the rules, it is not easy to see or hear if they are fulfilled. E.g., it takes an extremely trained ear or a laborious manual counting of the intervals to check whether the rule "the beginning and the end of all motion needs to be consonant" has been satisfied. However, the quantification of the counterpoint rules allows them to be objectively assessed by a computer, and allows a

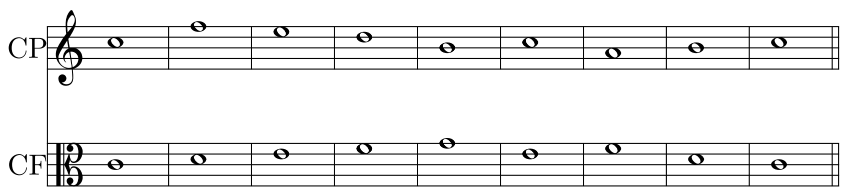


Figure 1: Cantus firmus and counterpoint example with objective score $f(s) = 1.589$

computer to judge whether one fragment is "better" than another. This information is used in the algorithm developed in the next section.

As an example, the score on the objective function of the counterpoint example given by Salzer and Schachter [1969] (see Figure 1) is calculated. As expected, both the horizontal (respectively 0.44 and 0.64) and the vertical part (0.5) of the score are very good and close to the optimum ($f(s) = 0$), yet not totally optimal. This shows that even experienced composers have difficulty in composing a fragment that obeys all Fuxian rules.

3. Variable neighbourhood search to generate counterpoint fragments

Most of the existing research on the use of metaheuristics for CAC revolves around evolutionary algorithms. The popularity of these algorithms can partially be explained by their "black-box" character. Essentially, an evolutionary algorithm only requires that a "fitness" (objective) function can be calculated. Other components, such as the recombination operator or the selection operator, do not require any knowledge of the problem that is being solved. This makes it very easy to implement an evolutionary algorithm, but the resulting algorithm is not likely to be efficient, compared to other metaheuristics that exploit the specific structure of the problem. Moreover, evolutionary algorithms generally use a lot of randomness, which brings them more into the "computer art" domain.

Local search heuristics, on the other hand, are generally more focused on the specific problem and use far less randomness to search for good solutions. These methods typically operate by iteratively performing a series of small changes, called *moves*, on a *current solution s*. The *neighbourhood $N(s)$* consists of all feasible solutions that can be reached in one single move from the current solution. The type of move therefore defines the neighbourhood of each solution. The local search algorithm always selects a better fragment, i.e., a solution with a better objective function value, from the neighbourhood. This solution becomes the new current solution, and the process continues until there is no better fragment in the neighbourhood. It is said that the search has arrived in a *local optimum* [Hansen et al., 2001]

To escape from a local optimum, the local search metaheuristic called VNS (variable neighbourhood search) uses two mechanisms. First, instead of exploring just one neighbourhood, the algorithm switches to a different neighbourhood (defined by a different move type) if it arrives in a local optimum [Mladenovic and Hansen, 1997]. The underlying idea is that a local optimum is specific to a certain neighbourhood. By allowing other move types, the search can continue. A second mechanism used by VNS is a so-called *perturbation*, that randomly changes a relatively large part of the current solution.

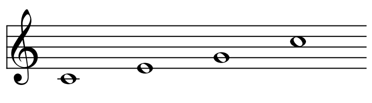
3.1. Components of the VNS algorithm

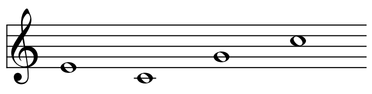
The VNS algorithm that is developed in this paper, operates in two sequential phases. First, the cantus firmus is generated, then the first species counterpoint melody. The algorithm used to generate both melodies is identical, and differs only in the objective function used. A “solution” in our algorithm is a musical fragment consisting of a cantus firmus and a first species counterpoint melody. The algorithm takes as its input the key and length of the fragment that is to be composed. A fragment that has the correct number of notes, all of which are in the specified key, is called *feasible*. The functioning of the VNS algorithm ensures that no infeasible fragments are generated.

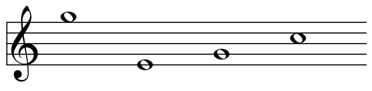
Three types of moves have been defined, they are represented in Table 1 and Figure 2.

The first neighbourhood (N_1) is defined by the Swap move type and is generated by swapping every pair of notes, starting from the current fragment. An example of a swap move is shown in Figure 2(b). N_1 will thus contain all possible fragments that can be obtained by swapping two notes in the current fragment.

Neighbourhoods N_2 and N_3 are respectively defined by move types Change1 and Change2. The Change1 move will change the pitch of any one note to any other allowed pitch. The last move, Change2, is an extension of the previous one whereby the pitch of two sequential notes is changed simultaneously to all possible allowed pitches. These last two moves are illustrated in Figures 2(c) and 2(d).


(a) Original


(b) Swap move


(c) Change1 move

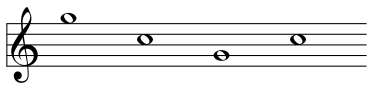

(d) Change2 move

Figure 2: Moves

For each neighbourhood, N_1, N_2 and N_3, the algorithm generates all possible feasible fragments s by applying the corresponding move and selects the one with the best value in the objective function $f(s)$. This strategy is called a *steepest descent* and typically ensures a fast improvement of the value of the objective function.

When no improving fragments can be found in any of the neighbourhoods of the current

N_i	Name	Description
N_1	Swap	Swap two notes
N_2	Change1	Change one note
N_3	Change2	Change two notes

Table 1: neighbourhoods

fragment, the VNS algorithm uses a *perturbation* strategy to allow the search to continue. The perturbation is implemented by reverting back to the current best fragment and changing a predefined percentage of the notes to a new, random note from the key. The reason for performing the perturbation move from the current best fragment, and not from the current fragment, is that this strategy was found to lead to better fragments faster.

Often, the current fragment scores optimally with respect to a large majority of subscores, but performs badly with respect to some others. The VNS therefore also uses an *adaptive weight adjustment* mechanism in order to steer the optimisation in the right direction. This mechanism adapts the weights of the subscores of the objective function at the same time a random perturbation is performed. The default setting for all of the weights is 1, although this can be changed by the user, in order to favour specific rules. The adaptive weight mechanism works by increasing the weight of the subscore that has the highest value (i.e., the subscore on which the current fragment has worst performance) by 1 immediately after the perturbation move. The algorithm then uses the score based on these new weights (called the *adaptive score* $f^a(s)$) to determine the quality of fragments in the neighbourhoods. In order to determine whether a fragment is considered as the new current best, however, the algorithm always considers the original weights. This weight adjustment strategy helps to keep the search strategy in the right direction, by increasing the impact of otherwise insignificant moves with little impact on the objective function. After the perturbation, the VNS always uses the Swap move first. The reason for this is that this neighbourhood does not introduce new notes, and therefore the search cannot immediately converge back to the previous local optimum.

To prevent the algorithm from getting trapped in cycles (i.e., revisiting the same local optimum again and again), a simple short term memory structure is introduced. This tabu list [Glover and Laguna, 1993] prohibits specific moves, more specifically, the ones that have been performed within the previous iterations. These “forbidden” moves are referred to as *tabu active*. The number of iterations that a move remains tabu active is called the *tabu tenure*. Each neighbourhood has its own tabu list, including its own tabu tenure. The tabu tenure is expressed as a fraction of the length of the melody in number of notes.

3.2. General outline of the implemented VNS algorithm

The VNS algorithm starts by generating a fragment s consisting of random notes from the key. This fragment is set as the initial global best solution s_{best}.

For the current solution s, the Swap neighbourhood (N_1) is generated. Moves on the tabu list of N_1 and infeasible fragments are excluded. The best solution s' of this neighbourhood is selected as the new current solution s, if its objective function with adaptive weights is better than that of s ($f^a(s') < f^a(s)$). The move applied to obtain s' is added to the tabu list of the Swap neighbourhood on a *first in first out* basis. This procedure is repeated as long as a better current solution is found, based on the objective function with adaptive weights. Each time a move is performed, the current solution s is compared to the best global solution s_{best}, based on the original objective function $f(s)$. If $f(s)$ is better than $f(s_{\text{best}})$, the current solution s becomes the new global best solution s_{best}.

If no better current solution, based on $f^a(s)$, is found in the first neighbourhood, the algorithm switches to the Change1 neighbourhood (N_2) and repeats the same procedure. If it again cannot find a better current solution in the Change1 neighbourhood, it switches to the Change2 neighbourhood (N_3).

If the algorithm goes through all three of the neighbourhoods without finding a better current solution (based on $f^a(s)$), then a perturbation is performed. $r\%$ of the notes of s are randomly changed to form the new current solution s. The weight of the subscore of the best solution (s_{best}) that has the highest value is also increased by one. If $f(s)$ is better than $f(s_{\text{best}})$, the current solution s becomes the new global best solution s_{best}.

The algorithm continues to repeat these steps (generating the three types of neighbourhoods), until either the optimum is found or it has reached the maximum number of iterations without improving the best global solution s_{best}, as specified by the user through the *maxiters* parameter.

4. Implementation

The VNS algorithm has been implemented in `C++`. To improve the usability of the software, a plug-in for the open source music composition and notation program MuseScore has been developed in JavaScript using the QtScript engine (Figure 3). This allows for easy interaction with Optimuse (the implementation of the VNS algorithm) through a graphical user interface. A cantus firmus can either be generated from a menu link, or composed on screen by clicking on the staff. When the cantus firmus has been generated, the counterpoint melody can be generated from a second menu link. The user can specify the input parameters such as the key (i.e., G# minor) and the weights for each subscore in an input file (*input.txt*). Other parameters of the algorithm, such as the ones discussed in the next section can optionally be passed as command line arguments. The resulting counterpoint music is displayed in score notation and can immediately be played back. MuseScore also provides easy export to midi, pdf, lilypond and other popular music notation formats. The fragment is also automatically exported in the MusicXML format, a relatively new format that is designed to facilitate music information retrieval. MuseScore can be downloaded at `musescore.org`. Optimuse is available for download at `antor.ua.ac.be/optimuse`.

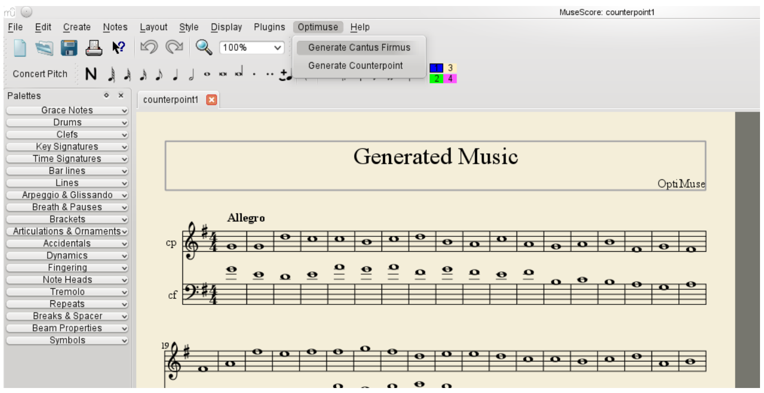


Figure 3: MuseScore

5. Experiments

The VNS algorithm described in Section 3 has several components. In this section, an experiment is described that has been set up to test the effectiveness of the different parts of the developed VNS and to determine their optimal parameter settings. In this way, components of the algorithm that do not contribute to the final solution quality can be removed. Separate experiments were performed for the generation of cantus firmus and of counterpoint. The different parameters that have been tested are displayed in Table 2. In Section 5.3, a small fragment is generated with the best parameter settings.

Parameter	Values	Nr. of levels
N_1 - Swap	on with tt_1=0, tt_1=$\frac{1}{4}$, tt_1=$\frac{1}{2}$, off	4
N_2 - Change1	on with tt_2=0, tt_2=$\frac{1}{4}$, tt_2=$\frac{1}{2}$, off	4
N_3 - Change2	on with tt_3=0, tt_3=$\frac{1}{4}$, tt_3=$\frac{1}{2}$, off	4
Random move (randsize)	$\frac{1}{4}$ changed, $\frac{1}{8}$ changed, off	3
Adaptive weights (adj. weights)	on, off	2
Max. number of iterations (iters)	10, 50, 100	3
Length of music (length)	16, 32, 48, 64 notes	4

tt_i = tabu tenure of the tabu list of neighbourhood N_i, expressed as a fraction of the total number of notes

Table 2: Parameters

A full-factorial experiment was performed for each of the experimental groups (cantus firmus and counterpoint). The total number of runs for both groups is $n = 4068$ $(2 \times 3^2 \times 4^4)$. The results of these 4068 runs of the algorithm were analysed by performing a Multi-Way ANOVA (Analysis of Variance). Using the open source statistical software R [Bates D., 2011], a model was estimated that takes into account these basic variables (see Tables 3 and 4) and analyses their impact on the musical quality of the end result, as well as on the running time of the algorithm.

5.1. Cantus Firmus

Measure	Value
R^2	0.2605
R^2 Adj	0.2577
F-statistic	95.1
p-value	$< 2.2e^{-16}$

Table 3: Model without interactions (CF) - Summary of Fit

When including only the main effects, the R^2 statistic of the linear regression (Table 3) shows that approximately 26% of the total variation around the mean value of the objective function can be explained by the model. However, the value of the R^2 statistic and therefore also the quality of the model can be increased by including interaction effects. In order to determine which parameters have a significant influence on the quality of the generated music, a model was calculated that takes into account the interaction effects of the significant factors ($p < 0.05$). This high quality model has an R^2 statistic of approximately 96% (Table 5), which means that it can

Parameter	Df	Sum Sq	Mean Sq	F value	Prob (>F)
N_1	1	37.47	37.465	220.3070	$< 2.2e^{-16}$ *
N_2	1	62.78	62.778	369.0956	$< 2.2e^{-16}$ *
N_3	1	124.50	124.501	731.9956	$9.702e^{-06}$ *
tt_1	2	0.01	0.006	0.0373	0.9634
tt_2	2	0.21	0.107	0.6274	0.5340
tt_3	2	0.22	0.110	0.6477	0.5233
randsize	2	37.93	18.964	111.4955	$< 2.2e^{-16}$ *
adj. weights	1	3.34	3.335	19.6100	$9.718e^{-06}$ *
iters	2	4.82	2.412	14.1792	$7.261e^{-07}$ *
length	3	3.71	1.235	7.2613	$7.406e^{-05}$ *

Table 4: Multi-Way ANOVA model without interactions (CF)

explain 96% of the variation. The most important influential factors of the model are displayed in Table 10 in appendix B.

Measure	Value
R^2	0.9612
R^2 Adj	0.9556
F-statistic	171.8
p-value	$< 2.2e^{-16}$

Table 5: Model with interactions (CF) - Summary of Fit

A similar Multi-Way ANOVA test has been performed, using the computation time of the algorithm as the dependent variable. The results of this analysis are displayed in Table 6 and show that all factors, except for the tabu list tenure for N_1 and N_3, have a significant influence on the running time.

Examination of the results of the ANOVA Table with interaction effects reveals that most of the factors have a very low p-value. This means that they have a significant influence on the result. The large number of small p-values indicates that most of the factors make a contribution to the model. Only tt_1, the tabu tenure of N_1 does not seem to have a significant influence on the quality of the result. We will take a more extensive look at the exact effect that the different parameters settings have on the objective function of the end result, by also taking into account their means plots (Figure C in appendix C). The means plots visualise the impact of each parameter setting in one graphical overview.

The ANOVA Table (Table 10) shows that parameters N_1, N_2 and N_3 all have a significant influence on the quality of the generated music (p-value $< 2e^{-16}$). Figures 6(a), 6(b) and 6(c) show that activating one of the neighbourhoods will have a decreasing effect on the score. However, they also show that adding a neighbourhood increases the mean running time for both N_1 and N_3. According to Table 6 this effect on the running time is significant. In deciding the optimal parameter settings, the primary objective was the musical quality; the computing time was considered a secondary objective. The three neighbourhoods will therefore be included in the optimal parameter set.

From Tables 6 and 10, it is clear that the tabu tenure for N_1 does not have a significant effect

Parameter	Df	Sum Sq	Mean Sq	F value	Prob ($> F$)
N_1	1	37397	37397	36.6801	$1.503e^{-9}$ *
N_2	1	13568	13568	13.3081	0.0002672 *
N_3	1	22921	22921	22.4816	$< 2.186e^{-6}$ *
tt_1	2	617	308	0.3025	0.7389918
tt_2	2	7782	3891	3.8164	0.0220770 *
tt_3	2	3225	1612	1.5815	0.2057780 *
randsize	2	445875	222938	218.6638	$< 2.2e^{-16}$ *
adj. weights	1	18610	18610	18.2537	$1.973e^{-5}$ *
iters	2	429998	214999	210.8771	$< 2.2e^{-16}$ *
length	3	2114993	704998	691.4822	$< 2.2e^{-16}$ *

R^2=0.3981

Table 6: Multi-Way ANOVA model for Time

on the quality or the computing time (p-values 0.535733 and 0.7389918). The tabu tenures for N_2 and N_3 do have a significant influence. The means plots 6(e) and 6(f) show that tabu tenures of respectively $\frac{1}{4}$ and $\frac{1}{2}$ (i.e., 25% and 50% of the length of the number of notes of the melody respectively) contribute most to a higher result quality.

With $p < 2e^{-16}$, the random perturbation size clearly has a significant effect on the solution quality. Plot 6(g) indicates that a perturbation will result in music of better quality, and that a perturbation of 12.5% offers the best results in the objective score, although this has a significant negative effect on the computing time.

Table 10 demonstrates that the adaptive weights parameter also has a significant influence on the musical quality if we take the interaction effects into account. Figure 6(h) shows that activating this parameter has a significant lowering effect on the mean computing time as well as the objective function. This means that the adaptive weights procedure makes a positive contribution to the general effectiveness of the VNS.

The maximum number of iterations is a significant factor in the algorithm ($p < 2.2e^{-16}$). The means plot 6(i) clearly shows that solution quality improves when the maximum number of iterations is higher. However, as expected, this is paired with a significant increase in computing time.

5.2. Counterpoint

The same full factorial experiment was run on the VNS algorithm to generate the counterpoint melody. This time, the algorithm generates the counterpoint melody, given a cantus firmus melody. As discussed in Section 2, a vertical aspect comes into play when generating the objective function.

Table 7 indicates that the model with interactions can explain approximately 91% of the total variation around the mean value of the objective function. The detailed results of this model are displayed in the ANOVA Table with interactions (Table 8). The ANOVA model shows the same significant parameters as the analysis in the previous section. The mean plots for the cantus firmus (Figure D in appendix D) also show a strong resemblance to the ones with the counterpoint results (Figure C). The conclusions of the previous section can therefore be extended to the counterpoint results.

Measure	Value
R^2	0.9122
R^2Adj	0.9062
F-statistic	152.4
p-value	$< 2.2e^{-16}$

Table 7: Model with interaction effects (CP) - Summary of Fit

Parameter	Df	Sum Sq	Mean Sq	F value	Prob ($> F$)
N_1	1	323.99	323.99	1173.4292	$< 2.2^e{-16}$ *
N_2	1	723.12	723.12	2618.9755	$< 2.2^e{-16}$ *
N_3	1	1794.21	1794.21	6498.1957	$< 2.2^e{-16}$ *
randsize	2	1441.36	720.68	2610.1349	$< 2.2^e{-16}$ *
iters	2	61.69	30.84	111.7095	$< 2.2^e{-16}$ *
length	3	104.29	34.76	125.8989	$< 2.2^e{-16}$ *
tt_1	2	0.76	0.38	1.3815	0.2513093
tt_2	2	4.17	2.09	7.5519	0.0005321 *
tt_3	2	104.13	52.07	188.5756	$< 2.2e^{-16}$ *
adj. weights	1	5.13	5.13	18.5697	$1.675e^{-05}$ *

Interactions of all significant factors are included in the model, but omitted in the Table for clarity.

Table 8: Multi-Way ANOVA model with interaction effects (CP)

5.3. Optimal parameter settings

The means plots and ANOVA analysis give a good indication of the significant parameters and their optimal setting. Table 9 summarises the optimal parameter settings. As mentioned, these parameters were set in a way that always favoured solution quality over computing time. In other words, the effect of a parameter on the computing time was only taken into account if the means plot did not show any effect on the quality.

Parameter	Values
N_1 - Swap	on with tt_1=0
N_2 - Change1	on with $tt_2=\frac{1}{4}$
N_3 - Change2	on with $tt_3=\frac{1}{2}$
Random move	$\frac{1}{8}$ changed
Adaptive weights	on
Max. number of iterations	100
Length of music	64 notes

tt_i = tabu tenure of the tabu list of neighbourhood N_i, expressed as a fraction of the total number of notes

Table 9: Best parameters

The algorithm was run again with these optimal parameter settings. Figure 4 shows the evolution of the solution quality over time, which is characterised by a fast improvement of the best solution in the beginning of the algorithm. After several iterations, the improvements diminish in size, especially in the case of counterpoint. The random perturbations can be spotted on the graph as peeks in the objective score. They are usually followed by a steep descent which often results in a better "best solution", confirming the importance of the random perturbation factor.

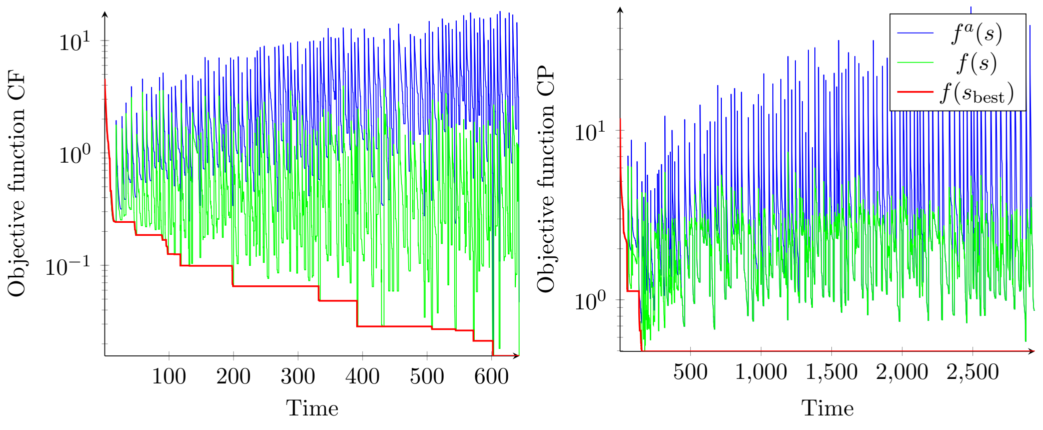
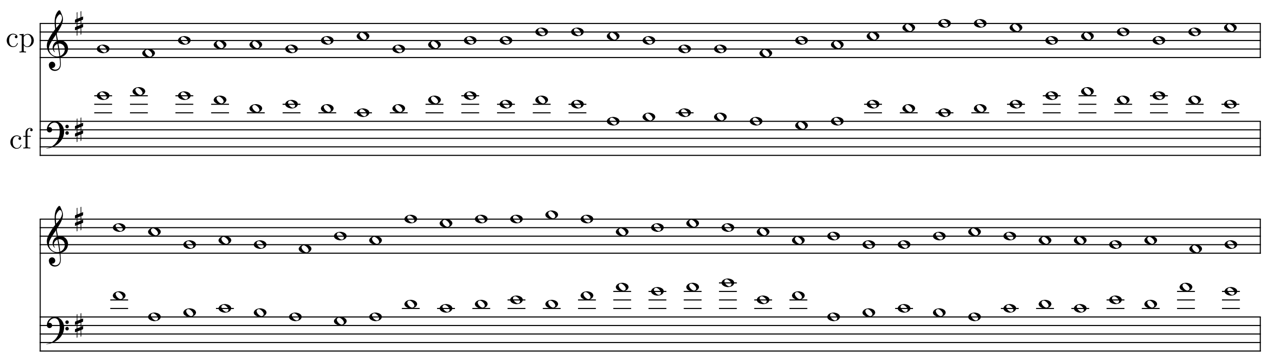


Figure 4: Evolution of the VNS with optimal parameter settings

An example of the music that is generated by the algorithm with these settings is displayed in Figure 5. With an objective score as low as 0.371394, this fragment can be considered as good first species counterpoint music.

Figure 5: Generated counterpoint with score of 0.371394



6. Conclusions

In this paper, an efficient VNS algorithm has been developed to automatically "compose" musical fragments consisting of a cantus firmus and a first species counterpoint melody. To this end the first species counterpoint rules have been quantified and used as an objective function in a local search algorithm. The different parameter settings of the VNS were extensively analysed by means of a full factorial experiment, which resulted in a set of optimal parameter settings. The resulting algorithm was then implemented in a user-friendly way. The musical output of the VNS has a good objective score and is pleasing to the ear.

An interesting extension of this work would be to evaluate different styles and types of music. A rhythmic component can be added to the music, by working with other species of counterpoint, such as florid counterpoint. The number of parts can also be increased, to allow more voices at the same time. Three part counterpoint offers a logical starting point for this expansion.

An analysis of a large database of existing music could also result in more detailed style characteristics, for instance specific to a composer. This could expand the objective function, so that more complex music can be generated. It is also interesting to compare the current VNS algorithm with other methods, such as a genetic algorithm, and compare their efficiency. This could be the subject of future research.

References

K. Adiloglu and F.N. Alpaslan. A machine learning approach to two-voice counterpoint composition. *Knowledge-Based Systems*, 20(3):300–309, 2007.

G. Aguilera, J. Luis Galán, R. Madrid, A.M. Martínez, Y. Padilla, and P. Rodríguez. Automated generation of contrapuntal musical compositions using probabilistic logic in derive. *Mathematics and Computers in Simulation*, 80(6):1200–1211, 2010.

T. Anders. *Composing music by composing rules: Design and usage of a generic music constraint system.* PhD thesis, Queen's University Belfast, 2007.

Chambers J. Bates D. The r project for statistical computing, 2011. URL `http://www.r-project.org/`.

J.A. Biles. Autonomous genjam: eliminating the fitness bottleneck by eliminating fitness. In *Proceedings of the GECCO-2001 Workshop on Non-routine Design with Evolutionary Systems*, San Francisco, California, USA, 7 2001. Morgan Kaufmann.

C. Blum and A. Roli. Metaheuristics in combinatorial optimization: Overview and conceptual comparison. *ACM Computing Surveys (CSUR)*, 35(3):268–308, 2003.

G. Boenn, M. Brain, M. De Vos, and J. Ffitch. Automatic composition of melodic and harmonic music by answer set programming. *Logic Programming*, 5366:160–174, 2009.

G. Carpentier, G. Assayag, and E. Saint-James. Solving the musical orchestration problem using multiobjective constrained optimization with a genetic local search approach. *Journal of Heuristics*, 16(5):1–34, 2010.

J.J. Fux and A. Mann. *The study of counterpoint from Johann Joseph Fux's Gradus Ad Parnassum - 1725.* Norton, New York, 1971.

M. Geis and M. Middendorf. An ant colony optimizer for melody creation with baroque harmony. In *IEEE Congress on Evolutionary Computation*, pages 461–468, 2007.

F. Glover and M. Laguna. *Tabu search.* Kluwer Academic Publishers, 1993.

P. Hansen, N. Mladenović, and D. Perez-Britos. Variable neighborhood decomposition search. *Journal of Heuristics*, 7(4):335–350, 2001.

N. Mladenovic and P. Hansen. Variable neighborhood search. *Computers & Operations Research*, 24(11):1097–1100, 1997.

G. Nierhaus. *Algorithmic composition: paradigms of automated music generation.* Springer, 2009.

H. Norden. *Fundamental Counterpoint.* Crescendo Publishing Co., Boston, 1969.

J. Polito, J. Daida, and T. Bersano-Begey. Musica ex machina: Composing 16th-century counterpoint with genetic programming and symbiosis. In *Evolutionary Programming VI*, pages 113–123. Springer, 1997.

F. Salzer and C. Schachter. *Counterpoint in composition: the study of voice leading.* Columbia University Press, 1969.

N. Tokui and H. Iba. Music composition with interactive evolutionary computation. In *Proceedings of the Third International Conference on Generative Art*, pages 215–226, 2000.

A. Detailed breakdown of objective function

Horizontal Score

These rules apply to a tonal cantus firmus consisting of L $\times$ 16 whole notes. L is the length of the fragment expressed in units of 16 notes.

1. Notes should not be repeated immediately. In the case of counterpoint, maximum two times two tied notes per L are allowed
2. Only consonant horizontal intervals are permitted. Horizontal intervals of 0, 1 ,2, 3, 4, 5, 7, 8, 9, 12, 15 and 16 semitones are consonant. The largest allowed consonant interval between two *consecutive* notes is the perfect octave.
3. Conjunct: stepwise movement should predominate, 2 to 4 leaps per L are optimal. Stepwise motion is an interval of 1 or 2 semitones. If the context is prevailing stepwise (90%), an exception can be made.
4. Each large leap should be followed by stepwise motion. A *leap* is an interval of more than 2 semitones. A *large leap* is an interval of more than 4 semitones.
5. Change direction after each large leap.
6. There should be one climax (high note).
7. Climax should be melodically consonant with tonic.
8. No more than two large leaps per L.
9. No more than two consecutive leaps.
10. Do not move too long, stepwise, in same direction.
11. The direction needs to change several times (min 3 times per L).
12. The starting note should be the tonic. In the case of counterpoint, the starting note can also be the dominant.
13. The ending note should be tonic and in same octave as starting note.
14. The penultimate note should be the leading tone for CP and the supertonic for CF. The leading tone is the 7th note of the scale. In the case of a minor scale it should be raised by half a tone. When the previous note is the sixth note of the scale, it is also allowed to be raised by half a tone.
15. The beginning and the end of all motion should be consonant. A motion interval is the interval between the start and end note of an upward or downward movement.

16. There should be a good balance between ascending and descending motion. Large motion intervals (> 9 semitones) should be avoided.
17. No frequent repetition of a single note (maximum 3 times per L).
18. No repetition of sequences within a 16 note interval. A sequence is a group of at least two notes.

$$subscore_1^H(s) = \begin{cases} \frac{\text{\#repeated notes}}{\text{L}} & \text{if \#repeated notes} \leq \text{L}, \\ 1 & \text{if \#repeated notes} > \text{L}. \end{cases} \tag{4}$$

$$subscore_1^H(s) = \frac{\text{\#2 repeated notes} - 2 \times \text{L}}{\text{length} - 2 \times \text{L}} \tag{5}$$

$$subscore_2^H(s) = 1 - \frac{\text{\#consonant intervals}}{\text{\#intervals}} \tag{6}$$

$$subscore_3^H(s) = \begin{cases} 0 & \text{if } 2{\times}\text{L} \leq \text{\#leaps} \leq 4{\times}\text{L}, \\ \frac{\text{\#leaps}-(4\times\text{L})}{\text{length}-1-(4\times\text{L})} & \text{if } 4{\times}\text{L} < \text{\#leaps}, \\ \frac{\text{\#leaps}}{\text{length}-1-(4\times\text{L})} & \text{if } 2{\times}\text{L} > \text{\#leaps}. \end{cases} \tag{7}$$

$$subscore_4^H(s) = \begin{cases} 0 & \text{if} \geq 90\% \text{ of intervals are 1 or 2 semitones}, \\ \frac{\text{\#large leaps not foll. by stepw. mot.}}{\text{\#large leaps}} & \text{if} < 90\% \text{ of intervals are 1 or 2 semitones}. \end{cases} \tag{8}$$

$$subscore_5^H(s) = \frac{\text{\#large leaps not followed by } \Delta\text{direction}}{\text{\#large leaps}} \tag{9}$$

$$subscore_6^H(s) = \frac{\text{\#occurences of highest note} - 1}{\text{length}} \tag{10}$$

$$subscore_7^H(s) = \begin{cases} 0 & \text{if climax is consonant with tonic}, \\ 1 & \text{if climax is } \textit{not} \text{ consonant with tonic}. \end{cases} \tag{11}$$

$$subscore_8^H(s) = \begin{cases} 0 & \text{if \#large leaps} \leq 2{\times}\text{L}, \\ \frac{\text{\#large leaps}-(2\times\text{L})}{\text{length}-1-(2\times\text{L})} & \text{if \#large leaps} > 2{\times}\text{L}. \end{cases} \tag{12}$$

$$subscore_9^H = \frac{\text{\# of 3 consecutive leaps}}{\text{length} - 3} \tag{13}$$

$$subscore_{10}^H(s) = \begin{cases} 0 & \text{if longest stepwise sequence} \leq 5, \\ \frac{\text{length of longest stepwise sequence}}{\text{length}-6} & \text{if longest stepwise sequence} > 5. \end{cases} \tag{14}$$

$$subscore_{11}^H(s) = \begin{cases} 0 & \text{if \# direction changes} \geq 3{\times}\text{L}, \\ 1 - \frac{\text{\# direction changes}}{3\times\text{L}} & \text{if \# direction changes} < 3{\times}\text{L}. \end{cases} \tag{15}$$

$$subscore_{12}^H(s) = \begin{cases} 0 & \text{if start note is tonic (or dominant in the case of CP)}, \\ 1 & \text{if start note is } \textit{not} \text{ tonic}. \end{cases} \tag{16}$$

$$subscore_{13}^H(s) = \begin{cases} 0 & \text{if end note is tonic in same octave as start note}, \\ 0.5 & \text{if end note is tonic, but is not equal to start note}, \\ 1 & \text{if end note is } \textit{not} \text{ tonic}. \end{cases} \tag{17}$$

$$subscore_{14}^H(s) = \begin{cases} 0 & \text{if pen. is leading note (CP) or supertonic (CF) in same oct. as end note}, \\ 0.5 & \text{if pen. is leading note (CP) or supertonic (CF), not in same oct. as end note}, \\ 1 & \text{if pen. is } \textit{not} \text{ leading note (CP) or supertonic (CF)}. \end{cases} \tag{18}$$

$$subscore^H_{15}(s) = \frac{\text{\#times that start and end of motion are dissonant}}{\text{\#motion intervals}} \tag{19}$$

$$subscore^H_{16}(s) = \frac{\text{\#motion intervals} > 9 \text{ semitones}}{\text{\#motion intervals}} \tag{20}$$

$$subscore^H_{17}(s) = \frac{\text{\#notes repeated more than } (3 \times \text{L}) \text{ times}}{\text{length}} \tag{21}$$

$$subscore^H_{18}(s) = \frac{\text{\#notes that are part of a rep. seq. within a 16 note interval}}{\text{length}} \tag{22}$$

Vertical Score

1. Only consonant vertical intervals are permitted. Vertical consonance is determined by the tonal distance between the cantus firmus (CF) and the counterpoint (CP). Vertical intervals of 0, 3, 4, 7, 8, 9, 12, 15 and 16 semitones are consonant.
2. All perfect intervals should be approached by contrary or oblique motion. Intervals of 0, 5, 7 and 12 semitones are considered perfect. There are four types of motion:
   - Contrary motion: CP and CF move in a different direction.
   - Similar motion: CP and CF move in the same direction.
   - Parallel motion: CP and CF move in the same direction at a constant (or virtually constant) distance. Thirds, sixths and tenths are considered parallel, even if the quality of the interval changes (minor vs major).
   - Oblique motion: one voice remains stationary and the other voice moves.
3. Avoid the overlapping of parts. Overlap occurs when the CF is 1 or 2 semitones higher than the previous note of the CP. Or when the CP is 1 or 2 semitones lower than the previous note of the CF.
4. The maximum allowed distance between voices is one tenth (16 semitones). With the exception of the climax.
5. There should be no crossing. Crossing occurs when the lower voice is above the upper voice.
6. Unison is only allowed in the beginning and end
7. Avoid leaps simultaneously in the cantus firmus and counterpoint. Especially leaps larger than one fourth (6 semitones) in the same direction.
8. Use all types of motion.
9. From unison to octave (and vice versa) is forbidden.
10. The ending should be unison or octave.
11. The beginning should be unison, octave or fifth above cantus.
12. Contrary motion should predominate slightly.
13. Thirds, sixths and tenths should predominate.
14. There can be no more than 3 successive parallel thirds, sixths and tenths.
15. The climax of the CF and CP should not coincide.

$$subscore^V_1(s) = 1 - \frac{\text{\#consonant intervals}}{\text{\#intervals}} \tag{23}$$

$$subscore^V_2(s) = \begin{cases} \frac{\text{\#perfect intervals not appr. by contr. or obl. motion}}{\text{\#perfect intervals}} & \text{if \#perfect intervals} \neq 0, \\ 0 & \text{if \#perfect intervals} = 0 \end{cases} \tag{24}$$

$$subscore_3^V(s) = \frac{\text{\#overlaps}}{\text{\#length-1}} \tag{25}$$

$$subscore_4^V(s) = \frac{\text{\#distances} > 16 \text{ semitones}}{\text{length}} \tag{26}$$

$$subscore_5^V(s) = \frac{\text{\#crossed notes}}{\text{length-1}} \tag{27}$$

$$subscore_6^V(s) = \frac{\text{\#unisons (excl beginning and end)}}{\text{length-2}} \tag{28}$$

$$subscore_7^V(s) = \frac{(\text{\#sim leaps not in same dir}) + 2 \times (\text{\#sim leaps} > 6 \text{ semitones in same dir})}{2 \times \text{\#intervals}} \tag{29}$$

$$subscore_8^V(s) = 1 - \frac{\text{\#different types of motion used}}{4} \tag{30}$$

$$subscore_9^V(s) = \frac{\text{\#motion from unison to octave and vsvs}}{\text{length}} \tag{31}$$

$$subscore_{10}^V(s) = \begin{cases} 1 & \text{if end is not unison or octave,} \\ 0 & \text{if end is unison or octave.} \end{cases} \tag{32}$$

$$subscore_{11}^V(s) = \begin{cases} 1 & \text{if beginning is not unison, octave or fifth above cantus,} \\ 0 & \text{if beginning is unison, octave or fifth above cantus.} \end{cases} \tag{33}$$

$$subscore_{12}^V(s) = \begin{cases} 1 & \text{if \# contrary motion} \leq \text{all other motion,} \\ 0 & \text{if \# contrary motion} > \text{all other motion.} \end{cases} \tag{34}$$

$$subscore_{13}^V(s) = \begin{cases} 0 & \text{if \# thirds, sixths and tenths} \geq 50\%, \\ 1 & \text{if \# thirds, sixths and tenths} < 50\%. \end{cases} \tag{35}$$

$$subscore_{14}^V(s) = \frac{\text{\#sequences of parallel thirds, sixths and tenths} > 3}{\text{length-3}} \tag{36}$$

$$subscore_{15}^V(s) = \begin{cases} 1 & \text{if the climax of the CF and CP coincide,} \\ 0 & \text{if the climax of the CF and CP do not coincide.} \end{cases} \tag{37}$$

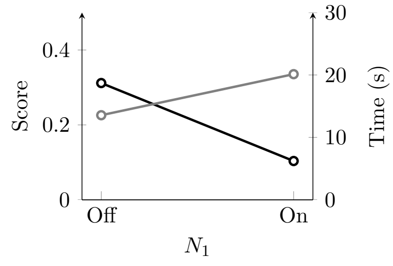
B. Multi-Way ANOVA model with interactions (CF)

Parameter	Df	Sum Sq	Mean Sq	F value	Prob ($> F$)
N_1	1	37.465	37.465	3685.4635	$< 2.2e^{-16}$ *
N_2	1	62.778	62.778	6175.4618	$< 2.2e^{-16}$ *
N_3	1	124.501	124.501	12247.2613	$< 2.2e^{-16}$ *
tt_1	2	0.013	0.006	0.6242	0.535733
tt_2	2	0.213	0.107	10.4974	$2.837e^{-05}$ *
tt_3	2	0.22	0.11	10.8363	$2.025e^{-05}$ *
randsize	2	37.927	18.964	1865.4689	$< 2.2e^{-16}$ *
adj. weights	1	3.335	3.335	328.102	$< 2.2e^{-16}$ *
iters	2	4.823	2.412	237.2371	$< 2.2e^{-16}$ *
length	3	3.705	1.235	121.4919	$< 2.2e^{-16}$ *
N_1:N_2	1	79.381	79.381	7808.7883	$< 2.2e^{-16}$ *
N_1:N_3	1	97.219	97.219	9563.5288	$< 2.2e^{-16}$ *
N_2:N_3	1	190.144	190.144	18704.564	$< 2.2e^{-16}$ *
N_1:randsize	2	0.8	0.4	39.3468	$< 2.2e^{-16}$ *
N_2:randsize	2	5.144	2.572	252.9978	$< 2.2e^{-16}$ *
N_3:randsize	2	17.73	8.865	872.0523	$< 2.2e^{-16}$ *
N_1:adj. weights	1	0.017	0.017	1.6866	0.1941202
N_2:adj. weights	1	0.025	0.025	2.4398	0.1183666
N_3:adj. weights	1	0.046	0.046	4.5202	0.033557 *
randsize:adj. weights	2	5.591	2.795	274.9924	$< 2.2e^{-16}$ *
N_1:iters	2	0.844	0.422	41.4913	$< 2.2e^{-16}$ *
N_2:iters	2	2.203	1.102	108.369	$< 2.2e^{-16}$ *
N_3:iters	2	4.32	2.16	212.4846	$< 2.2e^{-16}$ *
randsize:iters	4	0.359	0.09	8.8365	$4.225e^{-07}$ *
adj. weights:iters	2	0.122	0.061	6.0082	0.0024806 *
N_1:length	3	1.395	0.465	45.7464	$< 2.2e^{-16}$ *
N_2:length	3	0.196	0.065	6.4358	0.0002411 *
N_3:length	3	0.964	0.321	31.5985	$< 2.2e^{-16}$ *
randsize:length	6	0.168	0.028	2.7547	0.0112953 *
adj. weights:length	3	0.018	0.006	0.5819	0.6268833
iters:length	6	0.024	0.004	0.3941	0.8832212
N_1:N_2:N_3	1	238.596	238.596	23470.7999	$< 2.2e^{-16}$ *
N_1:N_2:randsize	2	2.539	1.27	124.8853	$< 2.2e^{-16}$ *
N_1:N_3:randsize	2	2.501	1.251	123.0317	$< 2.2e^{-16}$ *
N_2:N_3:randsize	2	16.622	8.311	817.5753	$< 2.2e^{-16}$ *
N_1:N_2:adj. weights	1	0.025	0.025	2.4761	0.1156635
N_1:N_3:adj. weights	1	0	0	0.0013	0.9709671
N_2:N_3:adj. weights	1	0.001	0.001	0.1372	0.711064
N_1:randsize:adj. weights	2	0.132	0.066	6.4709	0.001564 *
N_2:randsize:adj. weights	2	0.004	0.002	0.2066	0.8133344
N_3:randsize:adj. weights	2	0.205	0.102	10.064	$4.367e^{-05}$ *
N_1:N_2:iters	2	2.026	1.013	99.6471	$< 2.2e^{-16}$ *
N_1:N_3:iters	2	1.941	0.97	95.4637	$< 2.2e^{-16}$ *
N_2:N_3:iters	2	7.03	3.515	345.7817	$< 2.2e^{-16}$ *
N_1:randsize:iters	4	0.248	0.062	6.0981	$6.875e^{-05}$ *
N_2:randsize:iters	4	0.11	0.028	2.7069	0.0287158 *
N_3:randsize:iters	4	0.301	0.075	7.4022	$6.164e^{-06}$ *
N_1:adj. weights:iters	2	0.005	0.003	0.25	0.7787971
N_2:adj. weights:iters	2	0.01	0.005	0.4876	0.614123
N_3:adj. weights:iters	2	0.063	0.032	3.1004	0.0451376 *
randsize:adj. weights:iters	4	0.219	0.055	5.3779	0.0002567 *
N_1:N_2:length	3	2.794	0.931	91.6277	$< 2.2e^{-16}$ *
N_1:N_3:length	3	3.088	1.029	101.2698	$< 2.2e^{-16}$ *
N_2:N_3:length	3	0.347	0.116	11.3937	$1.942e^{-07}$ *
N_1:randsize:length	6	0.256	0.043	4.2001	0.0003224 *
N_2:randsize:length	6	1.025	0.171	16.7981	$< 2.2e^{-16}$ *
N_3:randsize:length	6	1.296	0.216	21.245	$< 2.2e^{-16}$ *
N_1:adj. weights:length	3	0.008	0.003	0.2673	0.8490325

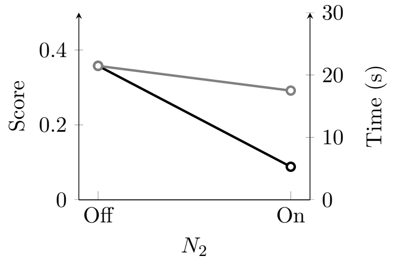
Interactions of all significant factors are included in the model, but omitted in the Table for clarity.

Table 10: Multi-Way ANOVA model with interactions (CF)

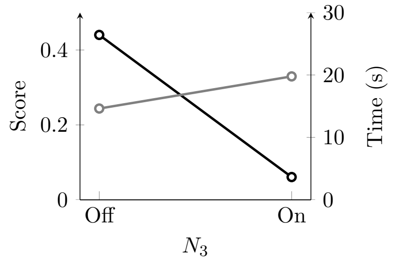
C. Means plot for cantus firmus



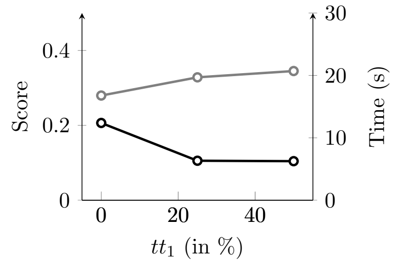
(a) Swap neighbourhood



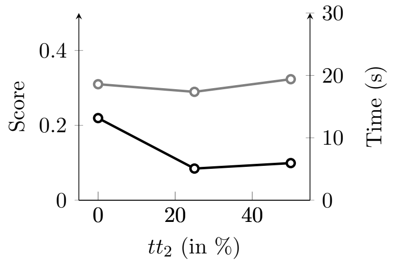
(b) Change1 neighbourhood



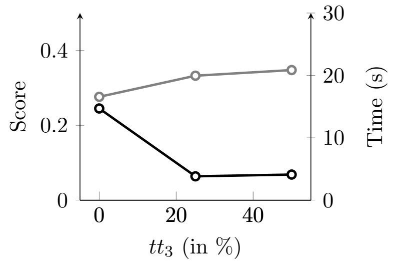
(c) Change2 neighbourhood



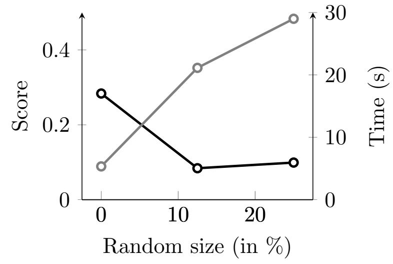
(d) Tabu tenure of Swap



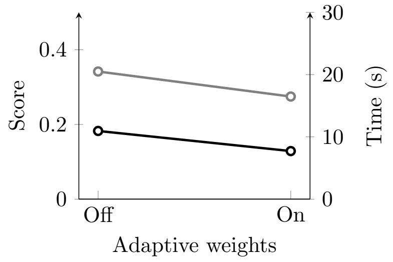
(e) Tabu tenure of Change1



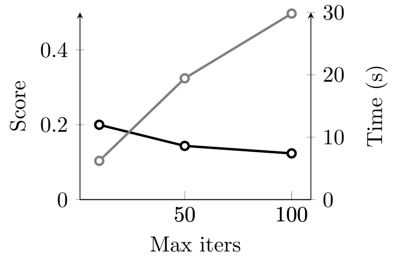
(f) Tabu tenure of Change2



(g) Size of the random perturbation



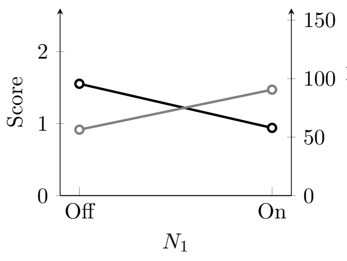
(h) Adaptive weights procedure



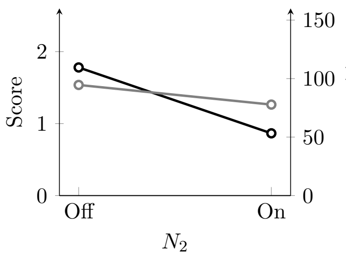
(i) Maximum iterations

—○—: running time
—○—: objective score

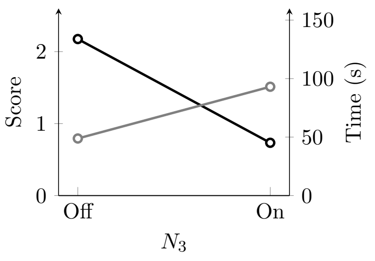
D. Means plot for Counterpoint



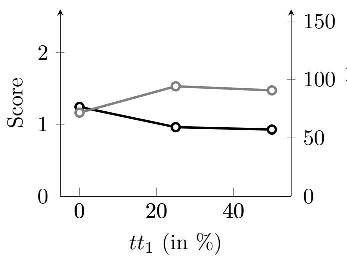
(a) Swap neighbourhood



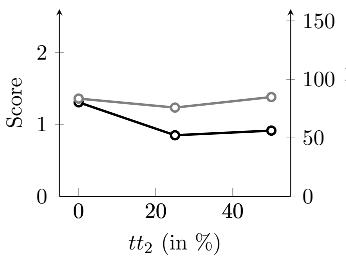
(b) Change1 neighbourhood



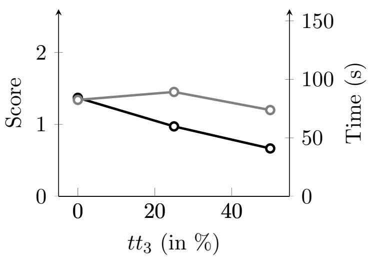
(c) Change2 neighbourhood



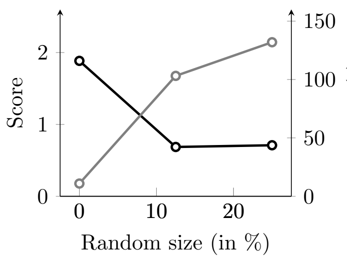
(d) Tabu tenure of Swap



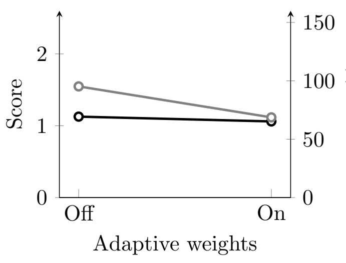
(e) Tabu tenure of Change1



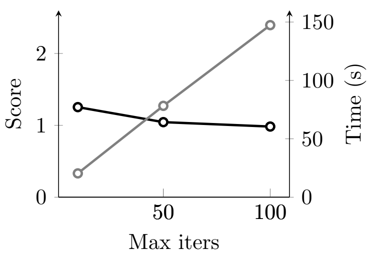
(f) Tabu tenure of Change2



(g) Size of random perturbation



(h) Adaptive weights procedure



(i) Maximum iterations

—o—: running time
—●—: objective score