Composing fifth species counterpoint music with a variable neighborhood search algorithm.

Herremans, D; Sörensen, K

doi:10.1016/j.eswa.2013.05.071

DEPARTMENT OF ENVIRONMENT,
TECHNOLOGY AND TECHNOLOGY MANAGEMENT

Composing Fifth Species Counterpoint Music With Variable Neighborhood Search

Dorien Herremans & Kenneth Sörensen

UNIVERSITY OF ANTWERP
Faculty of Applied Economics

Stadscampus
Prinsstraat 13, B.226
BE-2000 Antwerpen
Tel. +32 (0)3 265 40 32
Fax +32 (0)3 265 47 99
http://www.ua.ac.be/tew

FACULTY OF APPLIED ECONOMICS

DEPARTMENT OF ENVIRONMENT,
TECHNOLOGY AND TECHNOLOGY MANAGEMENT

Composing Fifth Species Counterpoint Music With Variable Neighborhood Search

Dorien Herremans & Kenneth Sörensen

RESEARCH PAPER 2012-020
SEPTEMBER 2012

University of Antwerp, City Campus, Prinsstraat 13, B-2000 Antwerp, Belgium
Research Administration – room B.226
phone: (32) 3 265 40 32
fax: (32) 3 265 47 99
e-mail: joeri.nys@ua.ac.be

The papers can be also found at our website:
www.ua.ac.be/tew (research > working papers) &
www.repec.org/ (Research papers in economics - REPEC)

D/2012/1169/020

Composing Fifth Species Counterpoint Music With Variable Neighborhood Search

Dorien Herremans & Kenneth Sörensen

University of Antwerp

Faculty of Applied Economics

Operations Research Group ANT/OR

Working Paper

September 12, 2012

In this paper, a variable neighborhood search (VNS) algorithm is developed and analyzed that can generate fifth species counterpoint fragments. The existing species counterpoint rules are quantified and form the basis of the objective function used by the algorithm. The VNS developed in this research is a local search metaheuristic that starts from a randomly generated fragment and gradually improves this solution by changing one or two notes at a time. An in-depth statistical analysis reveals the significance as well as the optimal settings of the parameters of the VNS. The algorithm has been implemented in a user-friendly software environment called Optimuse. Optimuse allows a user to input basic characteristics such as length, key and mode. Based on this input, a fifth species counterpoint fragment is generated that can be edited and played back immediately. This work is the expansion of a previous paper

by the authors in which first species counterpoint music is composed by a similar VNS algorithm.

1. Introduction

One of the earliest attempts at automated musical composition is due to Mozart. His *Musicalisches Würfelspiel* (Musical Dice Game) uses chance to piece together musical fragments into a Minuet (Boenn, Brain, De Vos, and Ffitch 2009). Similar experiments that use stochastics during the compositional process can be found in the work of avant-garde composers such as John Cage, Charles Dodge, Lejaren Hiller and Iannis Xenakis (Brown and Jenkins 2004). In the case of these composers the emphasis is not on the system generates a different output on each run, but on using techniques to create and improve a single composition.

A famous example is the "Atlas Eclipticalis", which Cage composed in 1961 by drawing musical staves over a star chart (Burns 2004). Dodge is known for pieces such as "Earth's Magnetic Field" composed by mapping fluctuations in the magnetic field to musical sounds. Other examples of compositions based on stochastics are Xenakis's "Pithroprakta" and "Metastaseis". These pieces are inspired by natural phenomena such as the flights of starlings and a swarm of bees (Fayers 2011).

From the very conception of computers, the idea was formed that they could be used as a tool for composers. Around 1840, Ada Lovelace, the world's first conceptual programmer (Gürer 2002) hinted at using computers for automated composition:

"[The Engine's] operating mechanism might act upon other things besides numbers [...] Supposing, for instance, that the fundamental relations of pitched sounds in the signs of harmony and of musical composition were susceptible of such expressions and adaptations, the engine might compose elaborate and scientific pieces of music of any degree of complexity or extent." – (Bowles 1970)

Starting in the mid 1900s, many systems have been developed for automated composition. One of the earliest pieces successfully composed by a computer is the "Illiac Suite" by Lejaren Hiller and Leonard Isaacson in 1957, who simulate the compositional process with a rule-based approach (Sandred, Laurson, and Kuuskankare 2009).

An automatic composition system typically wants to find a musical fragment that follows the rules of a chosen musical style as well as possible. By quantifying these rules, the "quality" of different fragments can be compared. A fragment that conforms to the style, will have a high solution quality. In this sense, composing music can be considered to be a combinatorial optimization problem. This musical composition problem is computationally complex, because the length of the piece exponentially increases the number of possible fragments. If an *exact method* like exhaustive enumeration is used to find the best possible fragment, the exponential number of solutions would make this practically impossible. For instance, a musical fragment consisting of 16 notes, without rhythm, in which each note can take on 14 different pitches, already has 14^{16} (or roughly 2.18 quintillion) possible note combinations. *Heuristics*, or rules of thumb, offer an alternative to exact methods. These simple strategies will generally find a reasonable solution in a very short computing time. *Metaheuristic* optimization algorithms offer a balance between the solution quality of exact methods, and the speed of heuristic methods. They therefore present a promising approach to generate good quality musical fragments within reasonable computing time.

Metaheuristics can roughly be classified in three categories. A first category is that of the *population-based* metaheuristics, which includes genetic and evolutionary algorithms, path relinking and others. These algorithms keep a set of solutions, usually refered to as population, and combine members of this set to form new ones (Sörensen and Glover 2012).

In 1991, the first genetic algorithm (GA) was applied in the field of music (Horner and Goldberg 1991). In the following decades, a number of GAs were developed for computer aided composition (CAC), an extensive overview is given by Burton and Vladimirova (1999) and Todd and Werner (1998).

A second category of metaheuristics is that of the *constructive* metaheuristics, such as ant colony optimization and GRASP. Constructive metaheuristics construct solutions from their constituting parts (Sörensen and Glover 2012). This category has not received as much attention as the population-based metaheuristics, especially in the domain of CAC. The first ant colony metaheuristic for harmonizing a melody was developed by (Geis and Middendorf 2007). This algorithm includes an objective function that indicates how well a fragment adheres to rules of baroque music.

Local search algorithms are the last category of metaheuristics, they include techniques such as tabu search and variable neighborhood search (VNS). These algorithms iteratively make small changes to a single solution (Sörensen and Glover 2012). Local search techniques have been used at IRCAM (Institut de Recherche et Coordination Acoustique/Musique) for solving musical constraint satisfaction problems (Truchet and Codognet 2004). The potential of local search metaheuristics in the area of CAC remains an interesting area for exploration.

In a previous paper by the authors, the first variable neighborhood search algorithm was successfully developed that could generate first species counterpoint music (Herremans and Sörensen 2012). This algorithm was implemented as a tool called *Optimuse*. In this research, the existing work is expanded to fifth species counterpoint.

2. Counterpoint and CAC

An important difference between the existing CAC algorithms is the way in which they evaluate music. GenJam, the genetic jammer, is an example of a population-based metaheuristic (Biles 2001). It uses a genetic algorithm to compose monophonic jazz fragments given a fixed chord progression. Unlike traditional genetic algorithms, the solution quality or *fitness* is not coded in the algorithm, but feedback is given for each population member individually by a *human mentor* (Biles 2003). This creates an enormous fitness bottleneck. A similar problem arises in the CONGA system, which uses a genetic algorithm to generate rhythmic patterns. Another interactive rhythmic pattern generator was developed by Horowitz (1994) using a genetic algorithm. In this case a human has to listen to each member of the population and indicate his or her preference for selection. This process has to be repeated for each population. The human fitness bottleneck slows down the composing process significantly and places a non-negligible psychological burden on the listener (Tokui and Iba 2000).

Other CAC research circumvents this human fitness bottleneck by using an *automatically calculated* objective function. Towsey, Brown, Wright, and Diederich (2001) develop a fitness function based on best practices. Their analysis of a library of existing melodies, the Western art

music repertoire, resulted in 21 features that are included in the fitness function. This library contains melodies by composers such a Bach, Tchaikovsky, Du Fay and Montiverdi as well as traditional nursery rhymes. The resulting function could be used to construct a genetic algorithm that generates short melodies in the tradition of Western diatonic music. Vox Populi, a genetic algorithm that can evolve a population of chords (Moroni, Manzolli, Zuben, and Gudwin 2000), uses a fitness function based on physical factors. These factors relate to melody, harmony and vocal range.

In a previous paper (Herremans and Sörensen 2012) the authors have implemented a variable neighborhood search algorithm that can generate cantus firmus and first species counterpoint melodies. The *counterpoint* style was chosen because it is one of the most formally defined musical styles, consisting of a set of simple rules (Rothgeb 1975). Figure 1 shows an example of a first species counterpoint fragment. In this figure, the bottom line is the cantus firmus (CF), or "fixed song". The top line is the counterpoint (CP), a melody that is composed by not only taking into account the melodic relationship between the subsequent notes, but also the harmonic balance with the cantus firmus. The term "counterpoint" refers to the relationship between these two melodies.

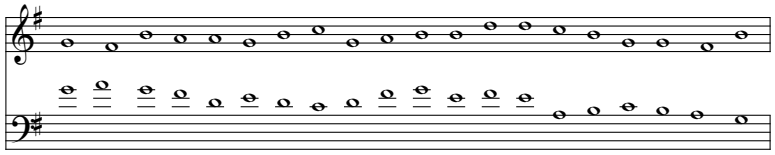


Figure 1: First species counterpoint fragment (generated by Optimuse)

Since the previously developed algorithm was successful in generating good cantus firmi and first species counterpoint fragments, Optimuse was extended to include fifth species, a more complex form of counterpoint music. In *fifth species*, a rhythmical aspect is added to the music. An example of fifth species counterpoint is displayed in Figure 2.

The amount of research done on automatically evaluating a musical fragment, to circumvent the human bottleneck, is very limited. A musical style is not typically defined in a formal way

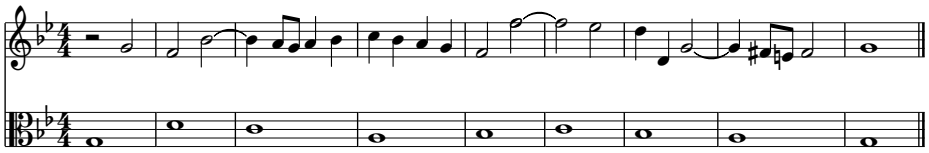


Figure 2: Fifth species counterpoint fragment (Salzer and Schachter 1969)

that can be quantified. However, the counterpoint style is an exception to this rule. It is a very formalized and restrictive style with simple rules for melody and harmony (Rothgeb 1975), which makes it perfect for quantification.

A number of studies use *counterpoint rules* to evaluate the generated music. Aguilera, Luis Galán, Madrid, Martínez, Padilla, and Rodríguez (2010) develop an algorithm that uses probabilistic logic to generate first species counterpoint music in C major. Their fitness function uses only the harmonic characteristics of counterpoint and ignores the melodic aspects. Polito, Daida, and Bersano-Begey (1997) implement a genetic algorithm that can compose fifth species counterpoint, based on an existing cantus firmus. The GA uses a piece-based adjusted fitness function based on Fux's homework problems, not a fixed set of rules. Composer David Cope's "Gradus", composes first species counterpoint given a cantus firmus. The evaluation is based on six criteria, a small subset of all counterpoint rules (Cope 2004).

Other studies have developed methods to compose *four-part* counterpoint music. For instance, Schottstaedt (1984) constructed a rule-based expert system for composing species counterpoint based on Fux's rules. The GA developed by Phon-Amnuaisuk, Tuson, and Wiggins (1999) uses a subset of the four-part rules as a fitness function. (Donnelly and Sheppard 2011) present a similar GA, based on 10 melodic, 3 harmonic and 2 rhythmic criteria. For a more extensive overview of existing research, the reader is referred to (Herremans and Sörensen 2012).

Although there is some existing research on the use of genetic algorithms to compose counterpoint, the application of the variable neighborhood search metaheuristic to generate counterpoint is new. In this paper, the first successful implementation of a VNS to generate first species counterpoint (Herremans and Sörensen 2012) is expanded to fifth species counterpoint music.

Contrary to most of the existing studies, a detailed breakdown of the objective function is given and Fux's rules are included as extensively as possible. In the next section the developed objective function that assesses how well music fits into the counterpoint style is explained. This is followed by a detailed description of the algorithm in the third section. Details on how Optimuse was implemented are discussed in "Architecture and Implementation". In the section called "Experiments", a statistical experiment is described that determines the optimal parameter settings of the VNS, this is followed by some general conclusions.

3. Quantifying musical quality

Optimuse generates fifth species counterpoint music, a very specific type of polyphonic classical music. Counterpoint music was the inspiration of many of the great composers such as Bach and Haydn and are foundational in music pedagogy, even today (Norden 1969). The counterpoint rules are a set of strict and specific rules that take into account the complexities that arise by playing multiple notes at the same time (Siddharthan 1999). Fux wrote down very specific counterpoint rules in his Gradus Ad Parnassum in 1725, a pedagogical book designed to teach musical students how to compose (Fux and Mann 1971). It starts by explaining rules for easy (first species) musical fragments and gradually moves to more complex (fifth species) music. Each of Fux's species can be seen as "levels" that add more complexity to the music, e.g. more rhythmical possibilities (Adiloglu and Alpaslan 2007).

Just like a student would start with Fux's first species, the cantus firmus and first species rules were implemented in Optimuse as a prototype (Herremans and Sörensen 2012). Each of the rules, as described by Salzer and Schachter (1969), were quantified into a *subscore* between 0 and 1. These subscores include rules such as "each large leap should be followed by stepwise motion in the opposite direction" and "the climax should be melodically consonant with the tonic". This paper brings Optimuse to the next level by extending these rules with *fifth species* counterpoint rules. The subscores now also include rules for more complex musical structures such as passing notes, ties, ottava and quinta battuta. A passing note is a non-harmonic note

that appears between two stepwise moving notes. An ottava or quinta battuta (beaten octave or fifth) occurs when two voices move in contrary motion and leap into a perfect consonance (Salzer and Schachter 1969). The rules can be divided into two categories. *Melodic rules* focus on the horizontal relationship between successive notes, whereas *harmonic rules* focus on the vertical interplay between simultaneously sounding notes. A detailed breakdown of the resulting subscores can be found in the appendix.

Given a cantus firmus and a fifth species counterpoint fragment, the objective function $f(s)$, displayed in Equation 1, calculates how good the fragment fits into the counterpoint style.

$$f(s) = \underbrace{\sum_{i=1}^{19} a_i.\text{subscore}_i^H(s)}_{\text{horizontal aspect}} + \underbrace{\sum_{j=1}^{19} b_j.\text{subscore}_j^V(s)}_{\text{vertical aspect}} \tag{1}$$

This score is used as an indicator of quality of the generated music. All subscores result in a number between 0 (best) and 1 (worst), therefore, the objective of the algorithm developed in the next section is to minimize $f(s)$. Each subscore has a weight a_i or b_j and the total score is a linear combination of the subscores with their corresponding weights. The weights are set in the beginning by the user. This allows a particular rule to be emphasized according to the preferences of the user.

The rules mentioned above are all "soft" rules. Although the aim of Optimuse is to find a musical fragment that has the lowest possible value for the objective function, it is allowed that a few of these rules are "broken", meaning that their corresponding subscore is not equal to zero. Given the large number of rules and their complexity, it not known if it is even possible to satisfy all of them at the same time for a piece of arbitrary length.

The soft rules are supplemented with a set of "hard" rules, that have been implemented as constraints. While soft rules can be violated, hard rules cannot. A violation of one or more of these constraints renders the musical fragment *infeasible*. All rhythmic criteria that are defined

by Salzer and Schachter (1969) have been implemented as hard constraints. Table 1 lists the implemented feasibility criteria.

No	Feasability criterium
1	All notes come from the correct key.
2	Only certain rhythmic patterns are allowed for a measure.
3	No rhythmic pattern can be repeated immediately or used excessively.
4	The first measure should be a half rest followed by a half note.
5	The penultimate measure is a tied quarter note, followed by two eight notes and a half note.
6	The last measure should be a whole note.
7	Ties are allowed between measures and notes of the same pitch.
8	A half note can be tied to a half note or a quarter note. No other ties are possible.
9	Maximum two measures of the same note value (duration) are allowed. Variations with eight notes do not count.

Table 1: Feasibility criteria

In the next section, these hard rules are used by the variable neighborhood search algorithm as feasibility criteria. The objective function, based on the soft rules, will be minimized by the algorithm.

4. Variable neighborhood search

Most of the existing literature on CAC proposes population-based algorithms (especially genetic/evolutionary algorithms) to compose music. In this paper, the class of local search metaheuristics is explored and a variable neighborhood search algorithm (VNS) is developed. The strength of local search metaheuristics in solving difficult combinatatorial optimization problems has been demonstrated in many different fields such as vehicle routing and scheduling (Sörensen and Glover 2012). The literature suggests that local search metaheuristics are particularly well-equipped to exploit the specific structure of the problem. This stands in contrast to the "black-box" character of many population-based metaheuristic implementations.

The core of VNS is a local search strategy, whereby the algorithm starts from an initial solution s and iteratively makes small improvements (or *moves*) to this solution in order to find a better one. The set of all solutions s' that can be reached from the *current* solution by making

one move is called the *neighborhood* of this solution. When no better solution can be found in the neighborhood of the current solution, the search has arrived in a local optimum. A variable neighborhood search strategy then switches to a different type of neighborhood to be able to continue the search escape these local optima (Mladenovic and Hansen 1997). A second mechanism used by VNS is a perturbation strategy. When no improving solutions can be found in any of the neighborhoods, a relatively large part of the current solution is randomly changed, allowing the search to continue (Hansen and Mladenović 2003).

Since the first implementations of variable neighborhood search in the late 90s, the technique has been successfully applied to a wide range of combinatorial problems (Hansen and Mladenović 2001) including finding extremal graphs (Caporossi and Hansen 2000), vehicle routing (Bräysy 2003), project scheduling (Fleszar and Hindi 2004) and graph coloring (Avanthay, Hertz, and Zufferey 2003). Hansen, Mladenović, and Perez-Britos (2001) found that for several problems, the VNS outperforms existing heuristics in an effective way and is able to find the best solutions in moderate computing time.

The previously developed application of VNS for CAC by the authors outperforms both a random search and a genetic algorithm (Herremans and Sörensen 2012). This efficient algorithm was extended to fifth species counterpoint music. A detailed description of this algorithm is given in the next section.

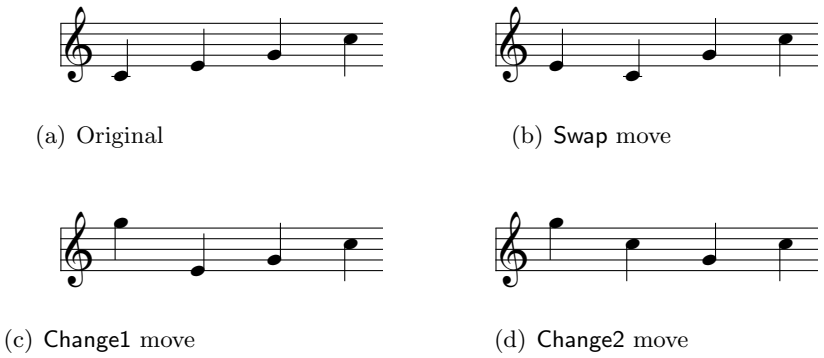
4.1. Components

The developed VNS starts from an *initial random fragment s*. Whilst generating this fragment, the hard rules from the previous section are taken into consideration to ensure that the fragment is feasible. This means, among other things, that the rhythmic patterns of the first, penultimate and last measure are set correctly. For each measure, a pattern is chosen from the set of allowed patterns and ties are applied correctly. The pattern selection mechanism ensures that there are no more than two sequential measures with the basic rhythmic pattern. This basic pattern

considers the rhythm without ties and eight note decorations. Finally, the pitches of all notes are randomly selected from the correct key. This ensures that the initial fragment s is feasible and can be used as a starting point for the VNS.

Three types of moves are defined, giving rise to three different neighborhoods. Figure 3 shows an example of a possible move for each of the three types of moves. The first move is the swap move. The swap neighborhood consists of all feasible fragments that can be created by swapping the pitch of any two notes of the current fragment. The change1 move changes the pitch of any one note by any other allowed pitch from the key. The neighborhood therefore consists of the fragments that are created by each changing any one note to any other allowed pitch. The change2 move expands this by changing 2 sequential notes to any other allowed pitch.

The algorithm starts by generating the swap neighborhood. Once a neighborhood is generated, the algorithm selects the fragment s' with the best value for the objective function $f(s')$ as the new current fragment. This strategy is called *steepest descent* and typically ensures a fast improvement of the solution quality. When no better solution can be found in one of the neighborhoods, the VNS moves to the next neighborhood.



(a) Original (b) Swap move

(c) Change1 move (d) Change2 move

Figure 3: The three types of moves

When no improving fragments can be found in any of the neighborhoods, a local optimum s_{lo} is reached (see Figure 4). In order to get out of this local optimum, a *perturbation* strategy is used. The algorithm reverts back to the best found fragment and changes a predefined percentage of the notes to a random allowed pitch. This strategy of iteratively building a sequence of solutions

leads to far better results than randomly restarting the algorithm when it reaches a local optimum (Lourenço, Martin, and Stützle 2003).

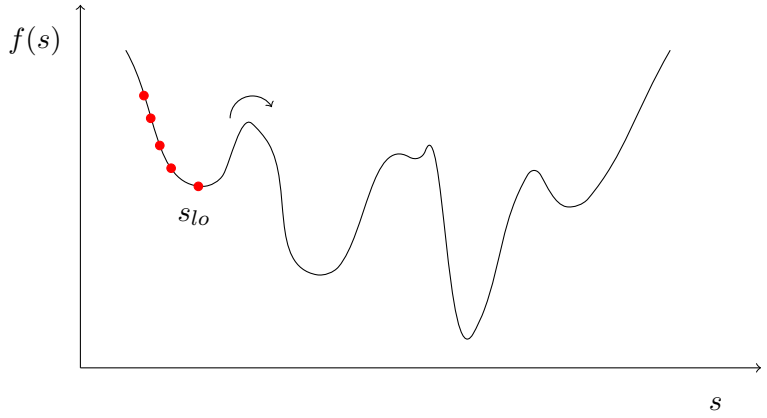


Figure 4: A perturbation move is used to "jump" out of a local optimum.

A *tabu list* was also implemented for each neighborhood, in order to keep the search from getting trapped in circles and revisiting the same fragments. This is a simple short term memory structure used to prevent moves that have been performed within the last iterations (Glover and Laguna 1993). The tabu lists are implemented as three lists (one for each neighborhood) that keep track of the notes that have recently been swapped or changed. The note positions that are on the tabu list, are called *tabu active*. Moves in tabu active positions are excluded from the neighborhood. The three tabu lists can each have a different size, or *tabu tenure*. The optimal tabu tenure is determined in the experiment described in the last section of this paper.

To achieve further improvements in solution quality, an *adaptive weight adjustment* mechanism was implemented. It is possible that a fragment has a good value for the objective function, because it scores well on a large number of subscores, but badly on others. In this case, the adaptive weights mechanism will steer the search back in the right direction, by increasing the impact of otherwise insignificant moves. Whenever a perturbation move is performed, the weight of the subscore with the worst value for the current fragment, will be increased by one. The score based on these new weights is referred to as the *adaptive score* $f^a(s)$. It is used to determine the quality of the fragment during the variable neighborhood search.

Each time a move is performed, the VNS checks if a new best fragment is found. This com-

parison is based on the objective function with original weights $f(s)$.

4.2. General outline of the implemented VNS algorithm

Figure 5 describes the structure of the algorithm in detail. The VNS starts by generating an initial random fragment according to the rules outlined in the previous section. This initial fragment is set as the current fragment s. The swap neighborhood is generated for s and its best feasible fragment s', based on the adapted score $f^a(s)$, is selected as the new current fragment s. The local search strategy in the swap neighborhood is repeated until a local optimum is reached, i.e., no more improving swap moves can be executed. The algorithm then switches to the change1 neighborhood and performs a similar local search. If it again gets trapped in a local optimum, a switch to the change2 neighborhood is made. When no more improving change2 moves are possible, the search switches back to the swap neighborhood. This process is repeated until no better fragment can be found in any of the three neighborhoods.

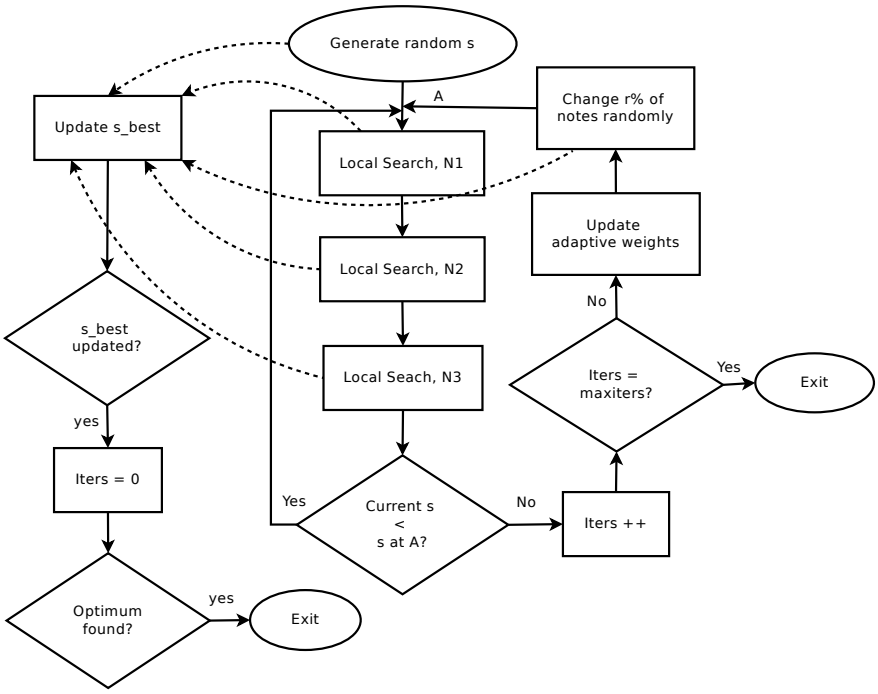


Figure 5: Overview of the developed VNS Algorithm

When all three neighborhoods do not include a better fragment, the weight adjustment mechanism increases the weight of the worst subscore of the current fragment s by one. This is followed by a perturbation whereby the pitch of a fixed percentage r of the notes is randomly changed to any other pitch in the key. This allows the search to continue.

A check for the "best fragment" s_{best} is done each time a move is performed. This check is based on the value of the original objective function $f(s)$. The algorithm stops when either an optimal fragment is found ($f(s) = 0$) or when the maximum number of successive perturbations without improvement of the best fragment, s_{best}, have been encountered. This stopping criterion is set by the user through the parameter *maxiters*.

5. Architecture and Implementation

The VNS algorithm is implemented in `C++` as Optimuse (version 2). The previous version of Optimuse optimized first species counterpoint music and only dealt with whole notes. The only information that was needed per note was the midi value of its pitch (integer value). Therefore, a musical fragment could be represented as a vector of integers. When dealing with fifth species counterpoint, this representation is no longer valid. A note is now an object with a data field for pitch, duration, measure, tied and beat (see Table 2).

Data field	Description
pitch	Midi value of the pitch of the note.
duration	Duration expressed in number of beats.
measure	The number of the measure that the note is in.
tied	0 if the note is not tied, 1 if it is the start of a tie, 2 if it is the end of a tie.
beat	Which beat the note falls on (1 to 16).

Table 2: Properties of the Note object

A musical fragment is a vector which contains all of the note objects in sequence.

The user can specify the input parameters "key" (e.g. G# major) and "weights of the subscores" in a file called `input.txt`. The parameters of the VNS that are discussed in the next

section are set to their optimal value by default and can be overwritten with command line arguments.

To allow the user to easily interact with Optimuse, a plug-in for the open source music notation and playback program *MuseScore* was written in JavaScript with QtScript Engine. This provides a drop-down menu to access Optimuse from a user-friendly interface. The generated music is displayed on the screen and can be played back in MuseScore. An export function to popular formats such as midi, pdf, lilypond is provided. The music is transferred from Optimuse to MuseScore in the MusicXML format. An XML-based music notation file format, designed to facilitate the interchange of scores (Good 2001).

The starting point for composing a counterpoint melody is a cantus firmus. The user can either input a new cantus firmus in MuseScore or choose to generate a new one from the Optimuse drop-down menu. The generated cantus firmus is displayed in editable form in MuseScore and can be modified to suit the user's expectations. When a satisfactory cantus firmus is displayed, the Optimuse drop-down menu can again be used to generate a fitting counterpoint melody.

Optimuse (including the MuseScore plug-in) is available for download at `http://antor.ua.ac.be/optimuse`.

6. Experiments

The developed VNS algorithm consists of different components, that are described in detail in the section on "Variable Neighborhood Search". As is common in metaheuristics, many of these components have one or more parameters that needs to be set, such as the tabu tenure. To thoroughly test the effectiveness of these components and their possible parameter settings, an exhaustive statistical experiment was performed. Table 3 displays the analyzed factors.

A full factorial experiment was run to test all possible combinations of the factors. This

Parameter	Values	No. of levels
N_{c1} - Swap	on with $tt_{c1} = 0$, $tt_{c1} = \frac{1}{16}$, $tt_{c1} = \frac{1}{8}$, off	4
N_{c2} - Change1	on with $tt_{c2} = 0$, $tt_{c2} = \frac{1}{16}$, $tt_{c2} = \frac{1}{8}$, off	4
N_{sw} - Change2	on with $tt_{sw} = 0$, $tt_{sw} = \frac{1}{16}$, $tt_{sw} = \frac{1}{8}$, 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 (maxiters)	5, 20, 50	3
Length of music (length)	16, 32 measures	2

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

Table 3: Parameters

resulted in 2304 runs ($2^2 \times 3^2 \times 4^3$). The cantus firmus that is used as an input for generating the counterpoint is composed by the previous version of Optimuse for each of the 2304 runs. A Multi-Way ANOVA (Analysis of Variance) was estimated with the open source software package R (Bates D. 2012). The model examines the influence of the parameter settings from Table 3 on the musical quality of the end fragment as well as the necessary computing time.

An ANOVA model was first calculated to identify the factors with a significant influence on the solution quality. This linear regression model only took into account the main effects. To improve the quality of the model, a second, ANOVA model was constructed taking into account the interaction effects between the factors that proved to be significant ($p < 0.05$) in the first model. The R^2 statistic of this improved model is 0.98 (see Table 4), which means that the model accounts for 98% of the variation around the mean value of the objective function.

Measure	Value
R^2	0.9836
R^2 Adj	0.9812
F-statistic	410.9 on 293 and 2010 DF
p-value	$< 2.2e^{-16}$

Table 4: Multi-Way ANOVA model with interactions - Summary of Fit

The p-values of the factors are displayed in Table 5. The interaction effects between more than two factors have been omitted in the table for clarity. The table reveals that all of the factors have a significant influence on the quality of the generated musical fragment ($p < 0.05$), with the exception of the tabu tenure of the change1 neighborhood. This means that this tabu list

does not have a significant influence on the solution quality. Although it is established that the other factors have a significant influence on the result, the nature of this influence still needs to be examined. The mean plots in Figure 6 clarify which parameter settings have a positive or negative influence on the result and reveal their optimal settings with respect to the objective function. The interaction plots were also drawn up in R to verify the conclusions from the mean plots for the interaction effects between parameters.

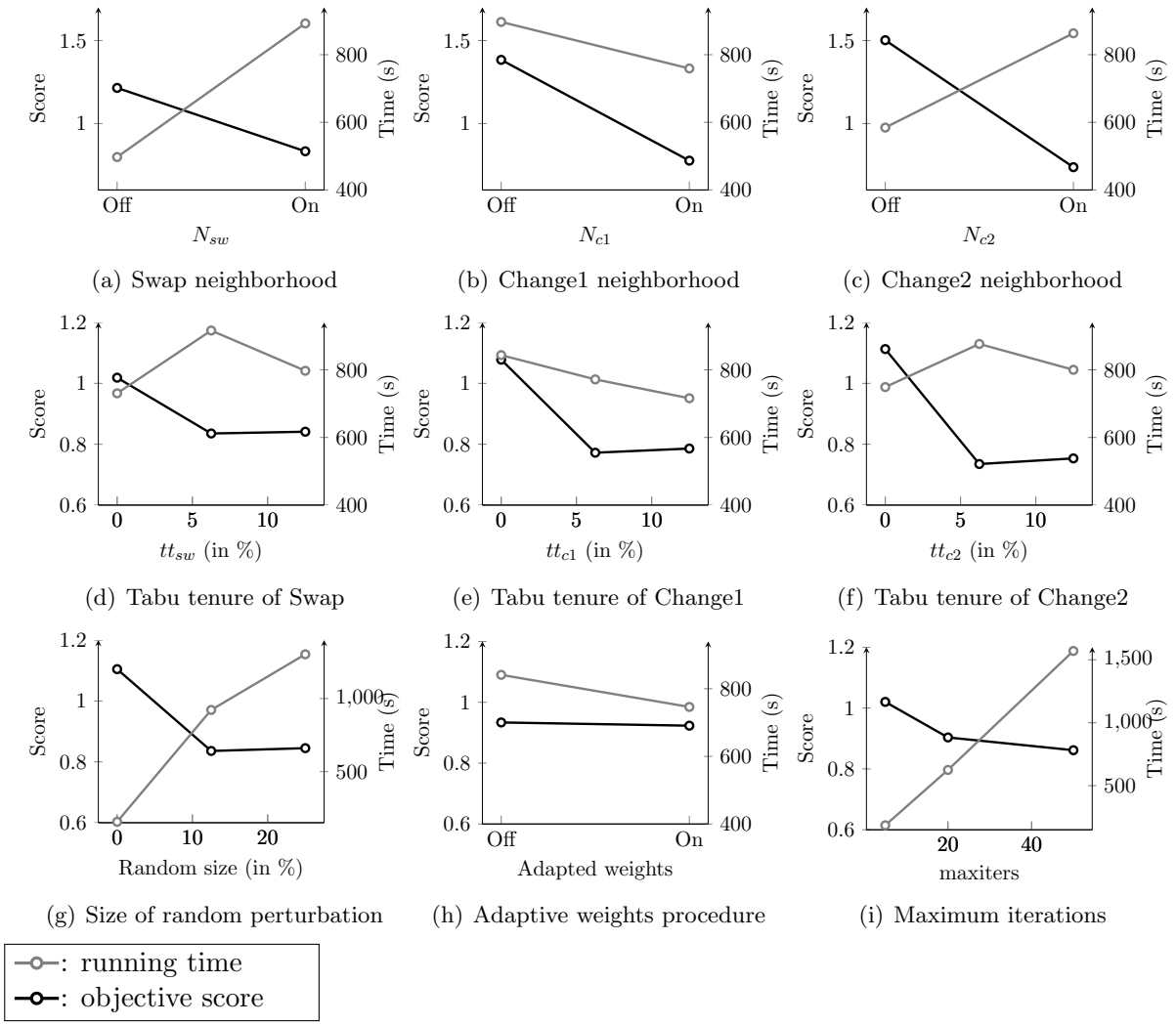
The mean plots for all three neighborhoods clearly show an improvement of the quality of the best found fragment when the respective neighborhood is active. The average value for the objective function is significantly lower when the neighborhoods are activated. This means that all three of the local search neighborhoods make a positive contribution to the solution quality. The tabu tenture for the change2 and swap neighborhood also have a significant influence on the quality of the end result. Figure 6(e) and 6(f) show that a tabu tenure of $\frac{1}{16}$ of the length of the music is optimal. The plot for the size of the permutation (see Figure 6(g)) offers another important insight. The random jump has the best effect on the solution quality when 12.5% of the notes are changed. The interaction plots support these conclusions.

Another significant means plot is that of the adaptive weights mechanism (see Figure 6(h)). The mean value of the objective function is slightly better when this mechanism is functional. A more detailed study of interaction plots reveals that the adaptive weights mechanism makes a bigger positive contribution for fragments of a smaller length. Finally, the last plot (see Figure 6(i)) shows that a higher number of allowed maximum iterations produces a better result.

A similar ANOVA model has been constructed to analyze the computing time of the algorithm. Again all factors have a significant influence, except for the tabu tenure of the change1 neighborhood and the adaptive weights mechanism. The means plots in Figure 6 reveal the nature of their influence. The computing time mostly has an inverse relationship with the solution quality, which is due to the nature of the stopping criteria. Whenever the search gets stuck in a local optimum, the quality will remain poor, but the algorithm's stopping criteria will be met sooner. This way good components of the algorithm (e.g. the random jump) cause an increase

Parameter	Df	F value	Prob (>F)
N_{c1}	1	9886.2323	¡ $2.2e^{-16}$
N_{c2}	1	15690.7234	¡ $2.2e^{-16}$
N_{sw}	1	3909.2959	¡ $2.2e^{-16}$
randsize	2	1110.1724	¡ $2.2e^{-16}$
maxiters	2	322.6488	¡ $2.2e^{-16}$
length	1	165.6053	¡ $2.2e^{-16}$
adj. weights	1	4.0298	0.0448367
tt_{c1}	2	2.2575	0.1048791
tt_{c2}	2	8.271	0.0002646
tt_{sw}	2	3.2447	0.0391833
N_{c1}:N_{c2}	1	25198.1739	¡ $2.2e^{-16}$
N_{c1}:N_{sw}	1	10264.8395	¡ $2.2e^{-16}$
N_{c2}:N_{sw}	1	10753.1655	¡ $2.2e^{-16}$
N_{c1}:randsize	2	114.2753	¡ $2.2e^{-16}$
N_{c2}:randsize	2	415.5001	¡ $2.2e^{-16}$
N_{sw}:randsize	2	117.0946	¡ $2.2e^{-16}$
N_{c1}:maxiters	2	19.0193	$6.564e^{-09}$
N_{c2}:maxiters	2	67.8841	¡ $2.2e^{-16}$
N_{sw}:maxiters	2	19.1179	$5.959e^{09}$
randsize:maxiters	4	42.7951	¡ $2.2e^{-16}$
N_{c1}:length	1	0.7116	0.3989975
N_{c2}:length	1	0.7192	0.3965164
N_{sw}:length	1	7.0076	0.0081798
randsize:length	2	1.5681	0.2086959
maxiters:length	2	4.0212	0.0180768
N_{c1}:adj. weights	1	96.6782	¡ $2.2e^{-16}$
N_{c2}:adj. weights	1	45.0273	$2.519e^{11}$
N_{sw}:adj. weights	1	0.0005	0.9813378
randsize:adj. weights	2	233.8511	¡ $2.2e^{-16}$
maxiters:adj. weights	2	0.8878	0.4117166
length:adj. weights	1	13.0046	0.0003183

Table 5: Multi-Way ANOVA model with interactions



(a) Swap neighborhood (b) Change1 neighborhood (c) Change2 neighborhood

(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

Figure 6: Mean plots CP

in computing time, because the search for better solutions can continue for a longer time, often resulting in a better solution quality.

An overview of the optimal parameter settings is given in Table 6. Improvements in solution quality was the first criterion in determining these optimal settings.

Parameter	Values
N_{sw} - Swap	on with $tt_{sw} = \frac{1}{16}$
N_{c1} - Change1	on with $tt_{c1} = \frac{1}{16}$
N_{c2} - Change2	on with $tt_{c2} = \frac{1}{16}$
Random move	$\frac{1}{8}$ changed
Adaptive weights	on
Max. number of iterations	50

Table 6: Best parameters

The VNS algorithm with the optimal settings was run on a fragment consisting of 32 measures. The evolution of the value of the objective function, both with original and adapted weights, is displayed in Figure 7. This plot shows a steep improvement of the solution quality during the first 100 moves, followed by a more gradual improvement in the next 600 moves. The many fluctuations of the score are due to the perturbation moves. Whenever a local optimum is reached, a temporary increase in the objective score leads to an eventual decrease. This confirms the importance of the perturbation move.

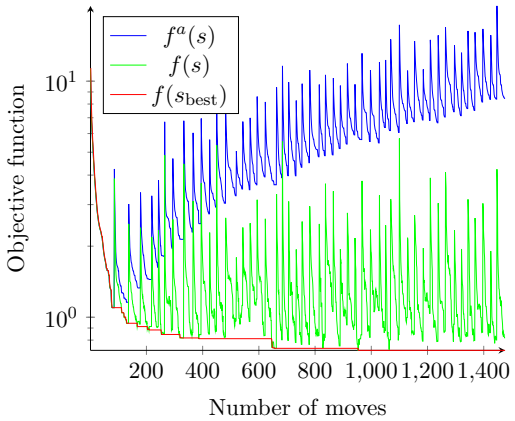


Figure 7: Evolution over time with optimal parameter settings

Figure 8 shows an example of a fifth species counterpoint fragment generated by Optimuse.

The objective score of this counterpoint fragment is 0.556776, which allows it to be considered as fifth species counterpoint according to the defined objective function. In comparison, random initial fragments typically have a score of around 10. Pdf scores and mp3 examples of Optimuse's output are available on `http://antor.ua.ac.be/optimuse`.

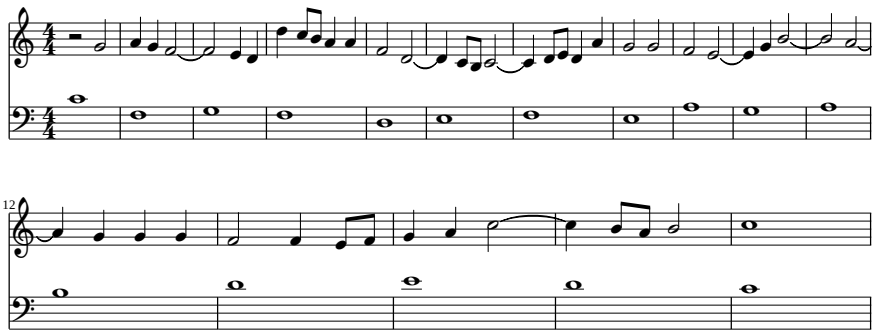


Figure 8: Fifth species counterpoint fragment (Optimuse)

The generated fragment sounds pleasing to the ear, but it can not yet be considered to be a finished composition. One of the reasons for that is its lack of theme or sense of direction, a feature that might be added to Optimuse in the future. The generated music could however, even at this point, be used by a composer as a starting point of a composition.

7. Conclusion

An efficient VNS algorithm was developed that can generate fifth species counterpoint music based on a cantus firmus. The rules of fifth species counterpoint were quantified and used as an objective function for this algorithm. The different components of the VNS were thoroughly tested and analyzed by means of a full factorial experiment. This revealed the significant components and their optimal parameter settings. The resulting algorithm was implemented as Optimuse, including a plug-in for MuseScore which provides a user-friendly environment for interacting with the VNS. The musical fragments composed by Optimuse generally have a good value for the objective function and sound pleasing.

A possible extension for the future is to allow more voices at the same time. Four-part counterpoint provides a starting point for this option. Another interesting research idea is to analyze a large existing database of music, in order to find criteria specific to a certain style or composer. These could then be implemented in, or added to, the objective function. This could, for instance, allow the generation of composer-specific music. Finally, adding evaluation criteria that enforce a sense of direction or theme could ensure the generation of more complete compositions.

A. Detailed breakdown of the objective function

These rules are valid for a fifth species counterpoint fragment of L $\times$ 16 measures. L can be seen as the length of the music, expressed in units of 16 measures.

Horizontal aspect of the objective function

$$subscore_1^H(s) = \frac{\#\text{8ths not preceded by step + 8ths not left by step}}{\#\text{8ths} \times 2} \tag{2}$$

$$subscore_2^H(s) = \frac{\#\text{highest note not as neighb. tone} - 1}{\text{length}} \tag{3}$$

$$subscore_3^H(s) = \frac{\#\text{highest notes on weak beat (except after neighbor. tones)}}{\#\text{highest notes, excl. those after neighb. tones}} \tag{4}$$

$$subscore_4^H(s) = 1 - \frac{\#\text{consonant intervals}}{\#\text{intervals}} \tag{5}$$

$$subscore_5^H(s) = \begin{cases} 0 & \text{if } 2\times\text{L} \leq \#\text{ leaps} \leq 4\times\text{L}, \\ \frac{\#\text{leaps}-(4\times\text{L})}{\text{length}-1-(4\times\text{L})} & \text{if } 4\times\text{L} < \#\text{ leaps}, \\ \frac{\#\text{leaps}}{\text{length}-1-(4\times\text{L})} & \text{if } 2\times\text{L} > \#\text{ leaps}. \end{cases} \tag{6}$$

$$subscore_6^H(s) = \frac{\#\text{large leaps not foll. by stepw. mot.}}{\#\text{large leaps}} \tag{7}$$

$$subscore_7^H(s) = \frac{\#\text{large leaps not followed by } \Delta\text{direction}}{\#\text{large leaps}} \tag{8}$$

Subscore	Description
1	Eight notes (8ths) must move in step.
2	There should be one climax, that can only be repeated after a neighboring tone. A neighboring note is positioned between two repeated notes and is approached and left by stepwise motion. The two repeated notes must always be vertically consonant.
3	The climax falls on a strong beat, except when it is repeated after a neighboring tone.
4	Only consonant horizontal intervals are permitted. Horizontal intervals of 0, 1, 2, 3, 4, 5, 7, 8, 9 and 12 semitones (or more octaves) are consonant.
5	Conjunct: stepwise movement should predominate. Two to four leaps per L are optimal. Stepwise motion is an interval of one or two semitones.
6	Each large leap should be followed by stepwise motion. A *leap* is an interval of more than two semitones. A *large leap* is an interval of more than 4 semitones.
7	Change direction after each large leap.
8	The climax should be melodically consonant with tonic.
9	No more than two consecutive leaps.
10	No more than two large leaps per L.
11	Do not move too long stepwise in same direction.
12	The direction needs to change several times (min. three times per L).
13	The ending note should be tonic.
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. Preferably in the same octave as the ending note.
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	Maximum four tied notes per L are allowed.
18	No repetition of sequences within a 16 note interval. A sequence is a group of at least two notes.
19	The largest allowed interval between two *consecutive* notes is the perfect octave.

Table 7: Description of the horizontal rules

$$subscore_8^H(s) = \begin{cases} 0 & \text{if climax is consonant with tonic,} \\ 1 & \text{if climax is } \textit{not} \text{ consonant with tonic.} \end{cases} \tag{9}$$

$$subscore_9^H = \frac{\#\text{ of 3 consecutive leaps}}{\text{length} - 3} \tag{10}$$

$$subscore_{10}^H(s) = \begin{cases} 0 & \text{if \#large leaps} \leq 2\times\text{L,} \\ \frac{\#\text{large leaps}-(2\times\text{L})}{\text{length}-1-(2\times\text{L})} & \text{if \#large leaps} > 2\times\text{L.} \end{cases} \tag{11}$$

$$subscore_{11}^H(s) = \begin{cases} 0 & \text{if longest stepwise seq.} \leq 5, \\ \frac{\text{length of longest stepwise seq.}}{\text{length}} & \text{if longest stepwise sequence} > 5. \end{cases} \tag{12}$$

$$subscore_{12}^{H}(s) = \begin{cases} 0 & \text{if \# direction changes} \geq 3\times\text{L}, \\ 1 - \frac{\#\text{direction changes}}{3\times\text{L}} & \text{if \# direction changes} < 3\times\text{L}. \end{cases} \tag{13}$$

$$subscore_{13}^{H}(s) = \begin{cases} 0 & \text{if end note is tonic,} \\ 1 & \text{if end note is } \textit{not} \text{ tonic.} \end{cases} \tag{14}$$

$$subscore_{14}^{H}(s) = \begin{cases} 0 & \text{if pen. is leading note (CP) or supertonic (CF) in same oct. as end ,} \\ 0.5 & \text{if pen. is leading note (CP) or supertonic (CF), not in same oct. as end note,} \\ 1 & \text{if pen. is } \textit{not} \text{ leading note (CP) or supertonic (CF).} \end{cases} \tag{15}$$

$$subscore_{15}^{H}(s) = \frac{\#\text{times that start and end of motion are dissonant}}{\#\text{motion intervals}} \tag{16}$$

$$subscore_{16}^{H}(s) = \frac{\#\text{motion intervals} > 9 \text{ semitones}}{\#\text{motion intervals}} \tag{17}$$

$$subscore_{17}^{H}(s) = \frac{\#\text{notes repeated more than } (2 \times \text{L}) \text{ times}}{\text{length}} \tag{18}$$

$$subscore_{18}^{H}(s) = \frac{\#\text{notes that are part of a rep. seq. within a 4 measure interval}}{\text{length}} \tag{19}$$

$$subscore_{19}^{H} = \frac{\#\text{intervals} > 12 \text{ semitones}}{\#\text{intervals}} \tag{20}$$

Vertical aspect of the objective function

$$subscore_{1}^{V}(s) = \frac{\#\text{dissonant whole notes}}{\#\text{whole notes}} \tag{21}$$

$$subscore_{2}^{V}(s) = \frac{\#\text{dissonant half notes except passing tones or weak beats}}{\#\text{half notes}} \tag{22}$$

$$subscore_{3}^{V}(s) = \frac{\#\text{dissonant quarters except passing and neighb. tones or weak beats}}{\#\text{quarters}} \tag{23}$$

$$subscore_{4}^{V}(s) = \frac{\text{pairs of eight notes with both notes dissonant}}{\#\text{pairs of eight notes}} \tag{24}$$

$$subscore_{5}^{V}(s) = \frac{\#\text{non-allowed unisons}}{\#\text{notes}} \tag{25}$$

$$subscore_{6}^{V}(s) = \frac{\#\text{distances} > 16 \text{ semitones}}{\text{length}} \tag{26}$$

$$subscore_{7}^{V}(s) = \frac{\#\text{crossed notes}}{\text{length}} \tag{27}$$

Subscore	Description
1	Whole notes should always be vertically consonant.
2	Half notes should always be consonant on first beat, unless they are suspended and continued stepwise and downward. They can be dissonant on second beat if they are a passing tone.
3	Quarter notes should always be consonant on the first beat, unless they are suspended and continued stepwise and downward. They can be dissonant on other beats if they are a passing tone or a neighboring tone.
4	Eight notes occur in pairs - at least one of them should be vertically consonant.
5	Unisons are allowed on the 1st beat through suspension if they are tied over or followed by stepwise motion.
6	The maximum allowed distance between voices is 16 semitones, except on the climax.
7	There should be no crossing.
8	Avoid the overlapping of parts.
9	From unison to octave (and vice versa) is forbidden.
10	The ending should be a unison or octave.
11	All perfect intervals (also perfect 4th) should be approached by contrary or oblique motion. Intervals of 0, 5, 7 and 12 semitones are considered perfect.
12	Avoid leaps sim. in cf and cp, especially large leaps ($>$ 6 semitones) in the same direction.
13	Use all types of motion.
14	The climax of the CF and CP should not coincide.
15	Successive unisons, octaves and fifths on first beats are only valid when separated by three quarter notes.
16	Successive unisons, octaves and fifths not on first beats are only valid when separated by at least two notes. For afterbeat fifths and octaves only separated by a single quarter this is allowed in the case of a consonant suspension of a quarter note.
17	No ottava or quinta battuda.
18	Best ending is a dissonant suspension into the leading tone. This can be decorated.
19	Thirds, sixths and tenths should predominate.

Table 8: Description of the vertical rules

$$subscore_8^V(s) = \frac{\#\text{overlaps}}{\#\text{length-1}} \tag{28}$$

$$subscore_9^V(s) = \frac{\#\text{motion from unison to octave and vsvs}}{\#\text{intervals}} \tag{29}$$

$$subscore_{10}^V(s) = \begin{cases} 0 & \text{if the ending is unison or octave,} \\ 1 & \text{otherwise.} \end{cases} \tag{30}$$

$$subscore_{11}^V(s) = \frac{\#\text{perfect intervals not appr. by contr. or obl. motion}}{\#\text{intervals}} \tag{31}$$

$$subscore_{12}^V(s) = \frac{(\#\text{sim leaps not in same dir}) + (\#\text{sim leaps} > 6 \text{ semitones in same dir})}{2 \times \#\text{intervals}} \tag{32}$$

$$subscore_{13}^V(s) = 1 - \frac{\#\text{different types of motion used}}{4} \tag{33}$$

$$subscore_{14}^{V}(s) = \begin{cases} 1 & \text{if the climax of the CF and CP coincide,} \\ 0 & \text{if the climax of the CF and CP do not coincide.} \end{cases} \tag{34}$$

$$subscore_{15}^{V}(s) = \frac{(\#\text{succ. unisons, 8ths or 5ths on first beats not separated by 3 quarters})}{\#\text{measures} - 1} \tag{35}$$

$$subscore_{16}^{V}(s) = \frac{(\#\text{succ. unisons, 8ths or 5ths not on first beats not separated by 2 unsusp. notes}}{\#(\text{notes - measures})} \tag{36}$$

$$subscore_{17}^{V}(s) = \frac{\#\text{ottava and quinta battudas}}{\#\text{measures}} \tag{37}$$

$$subscore_{18}^{V}(s) = \begin{cases} 1 & \text{if ending is dissonant suspension into the leading tone (can be decorated),} \\ 0.5 & \text{if ending is a suspension,} \\ 0 & \text{otherwise.} \end{cases} \tag{38}$$

$$subscore_{19}^{V}(s) = \begin{cases} 0 & \text{if \# thirds, sixths and tenths} \geq 50\%, \\ 1 - \frac{2\times\text{\# thirds, sixths and tenths}}{\text{\# notes}} & \text{otherwise.} \end{cases} \tag{39}$$

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.

C. Avanthay, A. Hertz, and N. Zufferey. A variable neighborhood search for graph coloring. *European Journal of Operational Research*, 151(2):379–388, 2003.

Chambers J. Bates D. The r project for statistical computing, 5 2012. 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.

J.A. Biles. Genjam in perspective: A tentative taxonomy for ga music and art systems. *Leonardo*, 36(1): 43–45, 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.

E.A. Bowles. Musicke's handmaiden: Of technology in the service of the arts. In *in H.B. Lincoln and Music*, page 4. Ithaca, NY: Cornelled., Ithaca, NY: Cornell ed., 1970.

O. Bräysy. A reactive variable neighborhood search for the vehicle-routing problem with time windows. *INFORMS Journal on Computing*, 15(4):347–368, 2003.

A. Brown and G. Jenkins. The interactive dynamic stochastic synthesizer. 2004.

C. Burns. Designing for emergent behavior: a john cage realization. In *Proc. ICMC*, 2004.

A.R. Burton and T. Vladimirova. Generation of musical sequences with genetic techniques. *Computer Music Journal*, 23(4):59–73, 1999.

G. Caporossi and P. Hansen. Variable neighborhood search for extremal graphs: 1 the autographix system. *Discrete Mathematics*, 212(1):29–44, 2000.

D. Cope. A musical learning algorithm. *Computer Music Journal*, 28(3):12–27, 2004.

Patrick Donnelly and John Sheppard. Evolving four-part harmony using genetic algorithms. In *EvoApplications (2)*, volume 6625 of *Lecture Notes in Computer Science*, pages 273–282. Springer, 2011.

E. Fayers. Utopian aesthetics: Philosophical perspectives upon the work of iannis xenakis. In *Proceedings of the Xenakis International Symposium*, Southbank Centre, London, 2011.

K. Fleszar and K.S. Hindi. Solving the resource-constrained project scheduling problem by a variable neighbourhood search. *European Journal of Operational Research*, 155(2):402–413, 2004.

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.

M. Good. Musicxml for notation and analysis. *The virtual score: representation, retrieval, restoration*, 12:113–124, 2001.

D. Gürer. Pioneering women in computer science. *ACM SIGCSE Bulletin*, 34(2):175–180, 2002.

P. Hansen and N. Mladenović. Variable neighborhood search: Principles and applications. *European journal of operational research*, 130(3):449–467, 2001.

P. Hansen and N. Mladenović. Variable neighborhood search. *Handbook of metaheuristics*, pages 145–184, 2003.

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

D. Herremans and K. Sörensen. Composing first species counterpoint musical scores with a variable neighbourhood search algorithm. *Journal of Mathematics and the Arts, Submitted for publication*, 2012.

A. Horner and D.E. Goldberg. Genetic algorithms and computer-assisted music composition. *Urbana*, 51 (61801):437–441, 1991.

D. Horowitz. Generating rhythms with genetic algorithms. In *Proceedings of the International Computer Music Conference*, pages 142–143. San Francisco: International Computer Music Association, 1994.

H. Lourenço, O. Martin, and T. Stützle. Iterated local search. *Handbook of metaheuristics*, pages 320–353, 2003.

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

A. Moroni, J. Manzolli, F.V. Zuben, and R. Gudwin. Vox populi: An interactive evolutionary system for algorithmic music composition. *Leonardo Music Journal*, 10(2000):49–54, 2000.

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

S. Phon-Amnuaisuk, A. Tuson, and G. Wiggins. Evolving musical harmonisation. In *Artificial neural nets and genetic algorithms: proceedings of the international conference in Portorož, Slovenia, 1999*, page 229. Springer Verlag Wien, 1999.

John Polito, Jason M. Daida, and Tommaso F. Bersano-Begey. Musica ex machina: Composing 16th-century counterpoint with genetic programming and symbiosis. In *Evolutionary Programming*, volume 1213 of *Lecture Notes in Computer Science*, pages 113–124. Springer, 1997.

J. Rothgeb. Strict counterpoint and tonal theory. *Journal of Music Theory*, 19(2):260–284, 1975.

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

O. Sandred, M. Laurson, and M. Kuuskankare. Revisiting the illiac suite–a rule-based approach to stochastic processes. *Sonic Ideas/Ideas Sonicas*, 2:42–46, 2009.

B. Schottstaedt. Automated species counterpoint. Technical Report STAN-M-19, Center for Computer Research in Music and Acoustics, May 1984.

R. Siddharthan. Music, mathematics and bach. *Resonance*, 4(5):61–70, 1999.

K. Sörensen and F. Glover. Metaheuristics. In *Encyclopedia of Operations Research and Management Science*. S. I. Gass and M. C. Fu, eds., Springer, New York, 3rd edition, 2012.

P.M. Todd and G.M. Werner. Frankensteinian methods for evolutionary music composition. In *Musical networks: Parallel distributed perception and performance*. MIT Press/Bradford Books, Cambridge, Mass., 1998.

N. Tokui and H. Iba. Music composition with interactive evolutionary computation. In *Proceedings of the Third International Conference on Generative Art*, volume 17:2, pages 215–226, 2000.

M.W. Towsey, A.R. Brown, S.K. Wright, and J. Diederich. Towards melodic extension using genetic algorithms. *Educational Technology & Society*, 4(2):54–65, 2001.

C. Truchet and P. Codognet. Musical constraint satisfaction problems solved with adaptive search. *Soft Computing-A Fusion of Foundations, Methodologies and Applications*, 8(9):633–640, 2004.