國立交通大學

工學院聲音與音樂創意科技碩士學位學程

碩 士 論 文

以自動簡化為基礎的音樂自動分析研究

The Study of Automatic Music Analysis Based on Musical Simplification

研 究 生：葉毅凡

指導教授：黃志方　教授

成維華　教授

中 華 民 國 九十九 年 七 月

以自動簡化為基礎的音樂自動分析研究

The Study of Automatic Music Analysis Based on Musical Simplification

研 究 生： 葉毅凡　　Student：Yi-Fan Yeh

指導教授： 黃志方　　Advisor：Chih-Fang Huang

成維華　　Wei-Hua Chieng

國 立 交 通 大 學

工 學 院 聲 音 與 音 樂 創 意 科 技 碩 士 學 位 學 程

碩 士 論 文

A Thesis

Submitted to Master Program of Sound and Music Innovative Technologies

College of Engineering

National Chiao Tung University

in partial Fulfillment of the Requirements

for the Degree of

Master

in

Engineering

July 2010

Hsinchu, Taiwan, Republic of China

中華民國九十九年七月

以自動簡化為基礎的音樂自動分析研究

學生：葉毅凡　　指導教授：黃志方

成維華

國立交通大學工學院聲音與音樂創意科技碩士學位學程

摘要

本論文首先探討各種自動作曲系統，從中歸納出對於自動作曲所需的重要音樂參數。接著利用電腦程式來進行自動分析音樂的工作，其中有些參數可直接利用統計方式得到，另外調性以及和聲的分析則是由各種相關文獻當中，找出特殊的演算法來達成。

在研究調性和和聲的分析時，我們發現了實驗中的誤差有一部份原因來自於樂曲中裝飾性的音符所產生。本文提出一個先將音樂自動簡化的演算法(AMSA, Automated Music Simplification Algorithm)來提升分析之準確度，文中比較了沒有簡化與簡化後之分析結果，在調性分析和和聲的分析上可增加約百分之六到百分之八的準確率。經由音樂自動簡化的手法，使得樂曲架構能更清楚顯現，也讓音樂參數分析能更加準確。

透過電腦自動化分析，可使需要大量人力與時間來進行的音樂分析減少成本。另外，可利用自動分析後的結果建立樂曲風格資料庫，提供各種風格的音樂規則來讓電腦進行自動作曲，達到能自動學習音樂風格並且自動創作的最終目標。

關鍵字：自動作曲，演算法作曲，自動化樂曲分析，調性分析，和聲分析，自動化樂曲簡化

The Study of Automatic Music Analysis Based on Musical Simplification

Student: Yi-Fan Yeh Advisor: Dr. Chih-Fang Huang

Dr. Wei-Hua Chieng

Master Program of Sound and Music Innovative Technologies

National Chiao Tung University

ABSTRACT

In this thesis, we investigate systems of automated composition and determine which music features are necessary for algorithm composition initially. Then this work build some automated tools to analyze music features such as pitch distribution, pitch transfer probability, key, and harmony.

In the process of analyzing music, the accuracy will generally be affected by ornaments. A musical simplification algorithm AMSA (Automated Music Simplification Algorithm) was proposed to filter ornaments out in music. The comparison results were shown that the accuracies of key finding and chord finding have been improved about 6%~8% by using simplification method. After simplification, the music structure is getting more clearly than original one.

Through the automatic music analysis, we can minimize the cost of music analysis. We can also use the analysis result to build a music style rule database. Eventually it is easily to compose music with style learning automatically using these rules.

Key Word: algorithmic composition, automated music analysis, key finding, chord finding, automatic music simplification

誌謝

很快的，在交大讀研究所的兩年過去了，雖然只有短短的兩年，但我的人生卻發生了許多變化。以往長達十幾年的學生生涯都在教我們學習前人的成果，這兩年我不但要學以致用，還要有自己的想法來解決問題；年紀增長而唸書的地方也離家較遠，很多事都要比以往更加獨立的處理；第一次為了找工作寫履歷表、面試，第一次找到真正的工作，第一次認真思考自己未來的路該怎麼走，一切都在這兩年內發生了。

回想這兩年，首先要感謝指導教授黃志方老師。兩年來除了在研究上給我幫助，當我對未來產生困惑時老師也給予我許多建議，另外也讓我插手許多相關的活動，帶著我見識許多同在這領域內努力的人們。可以在這兩年內跟隨您這位為人客氣又盡心盡力幫忙學生的好老師，真是一件幸運的事，希望未來還有機會能繼續和老師一起在這個領域內努力。

再來要感謝所有教導過我的老師。黃志方、馬孟民、鄭泗東老師以及來自國外的 Phil Winsor、Ken Paoli 老師。感謝你們讓我體驗到一個用理工的觀點看音樂，把音樂轉化為理工的世界。

除此之外，感謝一起在研究上的互相鼓勵的同學學長們。向斌學長無論在學術研究或者生活上,都讓我學到許多可貴的東西,鴻儒學長則是讓我更知道學術界是怎麼運轉。怡瑩、鶴雄、尹君、禮瑋、人慈、阿智，雖然在研究所一年級時大家並沒有常待在學校的習慣，但在學校的時間有你們總是有趣許多。而晚我們一年進來的學弟妹們，感謝有你們，讓一年前像鬼屋的實驗室現在充滿了生氣。

長庚一起考上清大交大的同學們，你們對於我來說實在太重要了。如果沒有你們我真的無法想像生活中扣掉學校的其它時間會多麼無聊。感謝你們從大學一路到現在的陪伴，能認識你們真的是我的福氣。

最後要感謝我的家人。從一開始在準備研究所考試時,是家人告知我交大有這個所。準備考試過程中，是家人給了我全心全意的支持以及幫助，使我最後得以考取。研究所兩年中，無論發生什麼事，家也永遠是我最好的避風港。沒有家人所提供的一切，這兩年的事可能都不會發生。

目錄

表目錄

圖目錄

第一章、 序論

1.1 動機

音樂可以量化嗎？我想大部份人對於這個問題的答案都是否定的。音樂、繪畫、雕塑等藝術，在一般的認知一直都和科學是兩條平行線。科學講究實證、精確以及量化數據，音樂則講求創意和情感。

的確，音樂或許很難量化。一段音樂好不好聽，除了主觀意識之外，還牽涉到表演者、氣氛、聽者的情緒等等各種複雜的因素。或許在未來所有這些因素都有辦法量化，但也有可能永遠都是無解的問題。

然而音樂雖然包含了許多無法量化的部份，但透過人類的努力，從第一件可以稱為「音樂」的作品產生直到現在，許多音樂的參數其實已經可以量化了。

樂理就是量化音樂的一門學問。或許很多人都認為作曲家要先學會樂理，接著才能成為真正的作曲家，但其實以前的音樂並沒有樂理這種規則，而是人類將發自內心的情緒轉化為各種音高、音色、音長等的排列，進而組織成為音樂。

樂理這門學問其實是後來的人們對於前人的音樂所整理出來的規則。學習音樂的人透過樂理學習前人的精華，並進一步利用這些在樂理中所學到的規則模仿，或創造新的音樂，有可能因此進一步產生新的樂理規則。樂理也因為這樣一直累積新的內容，成為一門龐大的學問。

既然人類能從音樂中整理出規則，並且利用這些規則來創作出新的音樂，那是不是也能利用電腦來分析音樂中的某些規則，甚至更進一步利用這些規則來使電腦自動創作產生音樂？很多人因為這個想法開始在這個領域開疆闢土，而這也是我們努力的目標。

1.2 背景介紹

利用電腦自動創作已漸漸形成一門稱為準則作曲或演算法作曲的學問，是利用各種演算法來執行音樂自動化創作的一種電腦自動化應用。

一般來說可將準則作曲劃分為兩類，第一類為建立特定風格的自動創作程式，這類

程式作法是將某種風格的音樂進行人工分析，之後利用分析所得到的特徵參數，轉化為各種利用程式控制的變數來產生音樂，較有名的軟體如 CWEST、CGMUSIC[2]和 MELODY GENERATOR[1]等等。

然而樂曲分析是需要大量時間及人力的投入，為了研究某位作曲家的作曲風格技巧，可能需要耗費很多時間，去分析該作曲家所有的作品，以建立樂曲風格之模型。如果研究的對象不只是一位作曲家，例如是整個時期的音樂，那需要分析更大量的曲目，花費的時間更是可觀，此外以人工分析音樂也易流於主觀與謬誤。另外這類的自動作曲軟體也因為分析是由人工進行，因此在自動創作上受到了限制，只能創作出某種特定風格的音樂。

因此另一類型的準則作曲希望利用電腦取代人工分析的工作，這樣一來當要產生某種特定風格的音樂，只需要搜集大量的音樂讓電腦自動分析音樂參數，就可讓電腦利用自動分析後的參數進一步產生音樂，如 David Cope 的 ALICE 自動作曲程式[3]。

這樣一來音樂自動分析就成為很重要的一個研究課題，除了需要瞭解該分析什麼項目才能建立最有效的規則之外，分析上的準確性也非常重要。音樂中許多參數的確是可以精確量化的，像是各種音符的使用比例，音符的轉移機率等，都是可以直接統計的項目，而且這些參數對於自動作曲也非常重要。然而如果只利用這些參數，產生出來的音樂還是會缺乏組織，因此除了分析這些基本項目之外，調性、和聲、節奏、動機甚至於音樂架構等都需要列入分析項目才能使產生出來的音樂有架構和主題，但這些參數都需要透過各種演算法才有辦法去分析結果。

其中許多人以數學的方式去自動分析音樂，將樂曲中的各種參數加以統計，如不同音高使用比例，藉此得到樂曲的某些特徵。更近一步，透過這些統計的數據，建立數學模型，來分析樂曲內隱含的訊息。 David Temperley 就利用了機率的方式分析音樂各種參數[4][5]，而 Elaine Chew 建立 Spiral Array 模型來分析音樂的調性、和聲以及同音異名等項目[6]。

另外，也有人以樂理的角度切入，把以往樂理的內容轉化為一條條的程式規則，讓電腦自動分析音樂，例如找出樂曲的動機、段落的銜接方式、旋律的行進方式等許多有用的資訊。例如 David Cope 在他的自動創作軟體中，利用 Pitch Class Set 的概念分析記錄了大量的音樂規則，並用於自動作曲當中[3]。

1.3 研究內容及貢獻

本研究的初衷在於建立一套完整的自動分析工具，希望能讓電腦自動分析許多音樂

參數，以利於未來建立一套完整的自動分析和自動創作音樂軟體。

因此研究的第一部份為建立一套自動分析工具，其中分析內容的基本項目包含了音域、音符密度、音符用量統計、音符轉移機率統計以及旋律走向。而無法由直接統計所獲得的參數當中，我們選擇了調性以及和聲兩個重要的項目來進行分析。此外也包括了一個 MIDI 轉檔程式。

在研究調性和和聲的分析時，我們發現了實驗中的誤差有一部份原因來自於樂曲中裝飾性的音符所產生。作曲家們為了讓音樂聽起來更流暢，更有個人風格，往往會在音樂內加入許多裝飾、連結性質的音符來連接樂曲的骨幹音。而這些裝飾性質的音符如果越多，在分析上往往會造成失誤。因此如果能在分析前事先過濾掉這些裝飾性質的音符，只留下較重要、較能代表樂曲架構的骨幹音，則對於某些音樂參數而言，將可提高分析上的準確率。

因此本研究第二部份提出了一套音樂簡化的演算法，稱為 AMSA(Automated Music Simplification Algorithm)，使得在分析音樂前系統能先行自動進行音樂簡化，除了希望透過音樂簡化的方式使樂曲架構更為明顯，以利樂曲架構的分析之外，也希望進一步利用簡化的手法增加自動分析的準確性。

第二章、 相關研究

2.1 背景知識

2.1.1 MIDI

MIDI(Musical Instrument Digital Interface)檔案為一種歷史悠久的音樂檔案類型。有別於一般以記錄音訊為基礎的音樂檔案(*.wave, *.mp3…)，MIDI 檔案利用紀錄音樂事件的方式來記錄音樂，而產生音訊的部份則交由硬體或軟體來幫忙。

利用 MIDI 做音樂分析的好處在於我們不需要透過繁雜的音訊處理技巧，就能直接抓取到音樂各種參數。舉例來說，要從某音訊中抓取聲音的音高，我們需要計算頻率後再轉化為對應的音高，但過程中除了計算較為麻煩外，也會因為各種干擾，如雜音、泛音等因素造成辨認失誤。但在 MIDI 檔中，一個聲音的產生只是一道訊息，訊息中包含了音高、音長、音量等參數，每種參數直接使用數字來對應，如音高 60 等同於中央 C，如此一來我們只需要正確解讀 MIDI 事件就能直接對應到我們想要的結果。

表 1 列出幾個重要且在我們的分析系統中主要使用到的 MIDI 事件：

表 1　重要 MIDI 事件

Event Name	Code(Hex)	Code(binary)	Data	Description
Note Off	8x	1000 xxxx	nn vv	nn : Note Number vv : Velocity
Note On	9x	1001 xxxx	nn vv	nn : Note Number vv : Velocity
Set Tempo	51	0101 0001	03 ttttt	ttttt: microseconds/quarter note
Time Signature	58	0101 1000	04 nn dd cc bb	Time Signature

資料來源：The MIDI Manual [7]

在表 1 中我們可利用 Note On 和 Note Off 事件來得到音高、音長、音量、觸發時間以及 Channel 這幾項資訊。利用 Set Tempo 和 Time Signature 來轉換 Tick Time 到真實的時間(秒數)。

在音樂分析上不需要太多額外資訊，有些為了增加 MIDI 音訊檔播放的豐富性所以產生的事件如音色、演奏法等等雖然在實作系統時還是需要讀取，但不需要去紀錄，所以在這邊不詳細講述所有 MIDI 的事件。

2.1.2 Pitch Class

Pitch class 的是在現代音樂所產生的概念。會有 Pitch Class 的產生其實可以從音樂調律的變化開始講起。

最初的音是利用五度相生的方式來產生，但五度相生的方式產生的音無法回到出發音，而是會和出發音的頻率有很些微的差別，造成音高一直無限制的由五度相生的方式產生，也因此有了十二平均律的概念。十二平均律把一個八度等分為十二個音，產生一個大家都能接受的音律，也就是鋼琴上一個八度內的十二個音。

而在平均律中，八度內的十二個音我們可以給予各自的代號：

表 2　Pitch Class 對應表

音名	**C**	**C# Db**	**D**	**D# Eb**	**E**	**F**	**F# Gb**	**G**	**G# Ab**	**A**	**A# Bb**	**B**
Pitch class	0	1	2	3	4	5	6	7	8	9	10	11

資料來源：Introduction to Post-Tonal Theory [8]

而在音樂分析中，我們可以把相差八度的音歸為同一類，因為相差八度的音雖然音高不同，由物理的角度來看就是頻率相差兩倍，但在和聲的角度中八度音所扮演的角色是一樣的。這樣一來在做分析時，可讓我們減少問題的複雜度但又不會造成失誤。利用 MIDI 的特性我們可以用算式(1)來算出每個音所屬的 Pitch Class:

$$PC_x = X \,\%\, 12 \tag{1}$$

現代音樂中的十二音列就是這樣產生的，而也因為這個概念產生了序列音樂等新型態的音樂。

另外由 Pitch Class 的觀念也可以衍生出音程對應的概念。一般來說音程是兩個音之間的差距，但利用 Pitch Class 的概念我們也可把音程忽略八度後簡化為十一種。

$$Interval_{x,y} = abs(X - Y) \,\%\, 12 \tag{2}$$

十一個音程的對應表：

表 3　音程對應表

音程	完全一度	小二度	大二度	小三度	大三度	完全四度	增四度	完全五度	小六度	大六度	小七度	大七度
Interval$_{x,y}$	0	1	2	3	4	5	6	7	8	9	10	11

資料來源：Introduction to Post-Tonal Theory [8]

在本系統的分析項目中，音高分布、調性分析和和聲分析等演算法都會使用到這個概念。

2.2 自動作曲

雖然本文的主旨在於音樂分析，但目的是希望能分析出對於自動作曲有幫助的項目，因此透過觀察現有的自動作曲軟體來瞭解自動作曲的流程，以及自動作曲過程中所需要的資訊是很重要的一項工作。

2.2.1 CWEST

CWEST 為 Phil Winsor 教授的作品，是一個自動產生美國鄉村音樂的準則作曲程式。其屬於先人工分析音樂規則，再把規則實做為程式的類型。

CWEST 原本是由 Visual Basic 語言寫成，圖 1 是由我們實驗室改寫為 C++後的程式介面。在 CWEST 介面中就可看出很多這個程式中所使用到的音樂參數。

在圖 1 的 1 號方塊中包括了節奏、音符密度、旋律跳躍的程度、和聲配置。2 號方塊表示音樂各個小段落的功能，因為 CWEST 產生的音樂每小段長度是固定的，使用者可選擇要使用多少個小段落，每個段落可選擇要以旋律組成還是由和聲組成。區塊 3 則是音樂風格的選擇。區塊 4 用來選擇音樂編制大小。第 5 區塊則是調性選擇的區域。

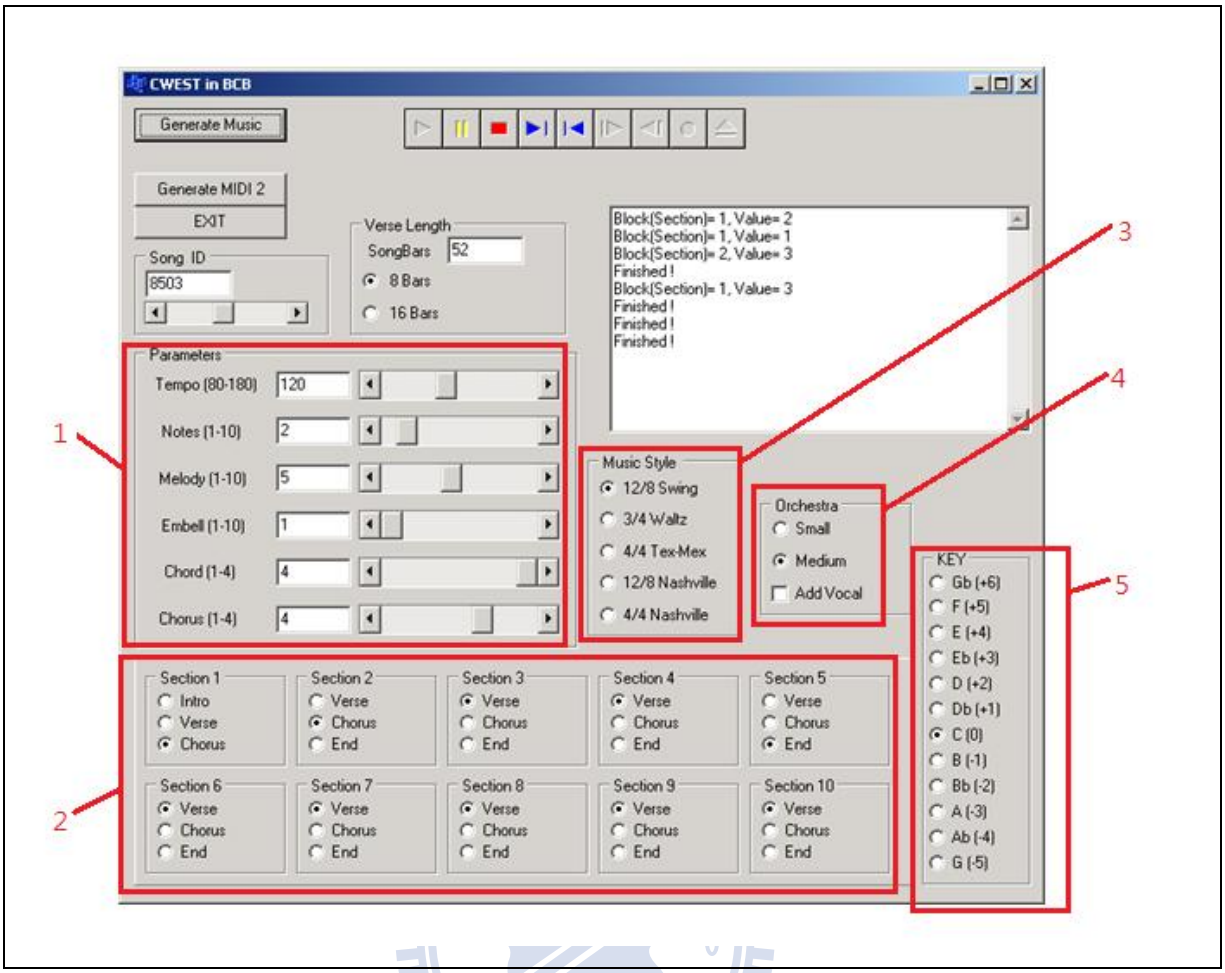


圖 1　CWEST 使用者介面

圖 2 為 CWEST 的自動作曲流程概要：

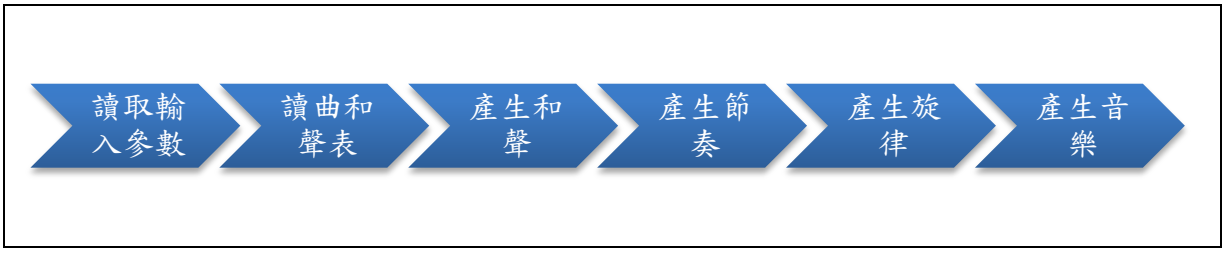


圖 2　CWEST 自動作曲流程

2.2.2 Melody Generator

Melody Generator 是 Dirk-Jan Povel 的自動作曲程式[1]。該程式並無特定音樂風格，它是利用基本的樂理和和聲知識來產生旋律。但這個程式特別的地方在於它可以一步一步讓你看出它如何產生音樂，而也可以對產生後的音樂做編輯動作。

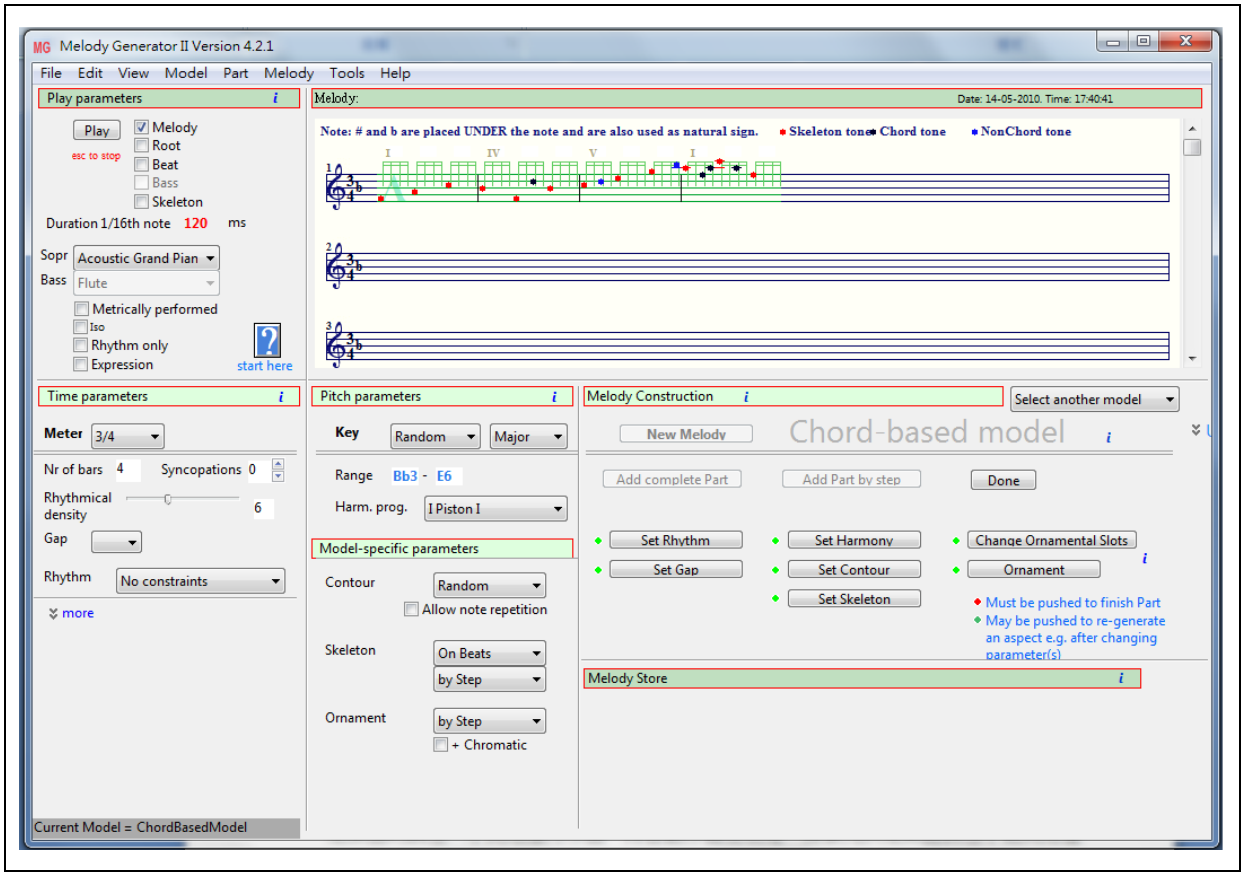
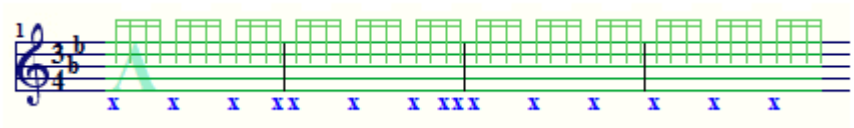
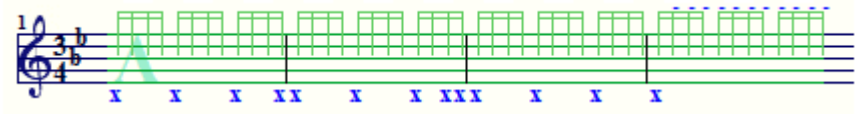
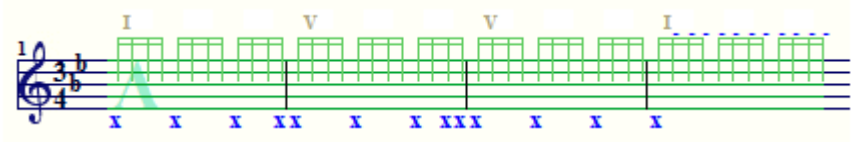
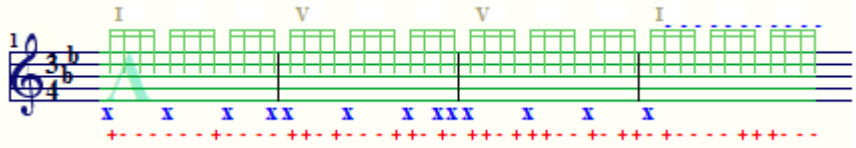
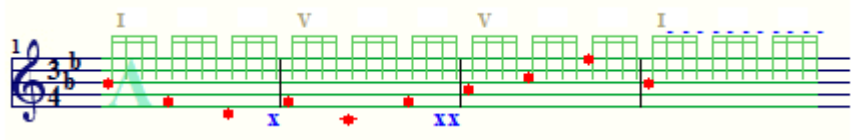
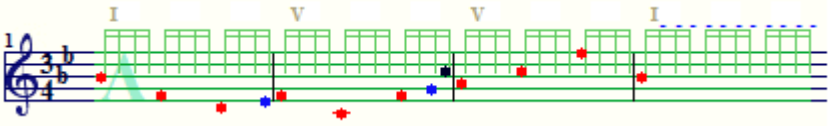


圖 3　Melody Generator 使用者介面

由圖 3 中可看出其中有許多參數可讓使用者調整，而以下簡單講解它產生旋律的步驟。

在表 4 中，第一步驟為產生節奏，產生節奏的方法是將節奏點細分到十六分音符，接著利用各節奏點的權重來決定要不要讓該節奏點出現音符。第二步驟為設定一個結束點。第三步驟決定合聲。第四步驟為決定旋律的走向。第五步驟是根據合聲、節奏、旋律走向填上骨幹音。最後步驟加入裝飾音。調性則是在一開始就會先決定。

表 4　Melody Generator 作曲流程

Step1 Set Rhythm 
Step2 Set Gap 
Step3 Set Harmony 
Step4 Set Contour 
Step5 Set Skeleton 
Step6 Ornament 

2.2.3 ALICE

以上所舉的兩個例子皆為先由人工分析音樂規則，再利用程式產生音樂的例子，但 ALICE 這個程式不一樣，ALICE 是 David Cope 的作品[3]，ALICE 是 Algorithmically Integrated Composing Environment 的縮寫，這個程式可以讓使用者輸入 MIDI 檔案，並且分析各種參數，建立音樂規則資料庫，接著利用資料庫所紀錄的規則產生音樂。

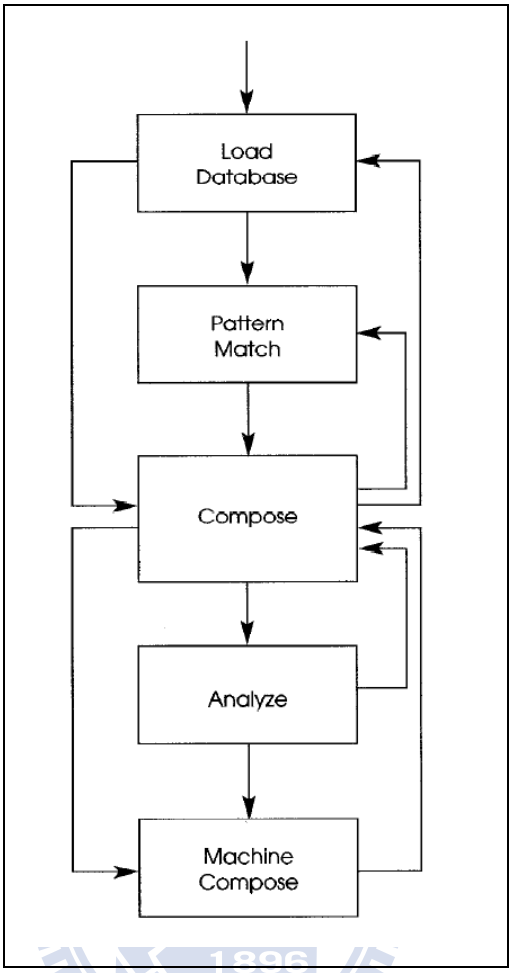


圖 4　ALICE 作曲流程圖

資料來源：The Algorithmic Composer [3]

圖 4 為 ALICE 程式作曲的可能流程圖，其中和前面所介紹的兩個程式較為不同的地方在於，它需要先從規則資料庫中取出音樂的規則和參數來進行作曲。其中有使用到的參數，一部份是基本參數，包括了音符的分布、調性、音高和音長的相互使用比例、旋律走向等等。

另一部份較為重要的是 Pattern Match，在 ALICE 的自動分析當中，利用 Pitch Class Set 儲存了很多有關於前後文連接的規則。ALCE 會將音樂切割成很多小區塊，並且紀錄每個區塊內用了哪些 Pitch Class，產生類似{{0, 4, 7},{2, 4, 6, 8}}這種規則，代表前後文的音符連接方式，這例子代表某兩連續的段落其 Pitch Class 是由 0, 4, 7 轉移到 2, 4, 6, 8。在 ALICE 的自動創作過程中，會拿當前產生片段的 Pitch Class Set 和資料庫中所紀錄的規則做比較，並選出符合的規則，進而產生出下個片段。然而在 ALICE 當中規則的紀錄並沒有經過簡化，而是選擇直接把所有資訊記錄下來，因此產生的音樂常常會有人認為並不是作曲，而只是重組。但就算如此，ALICE 仍然是目前世界上最成功利用自動分析來自動作曲的軟體，因此在分析上也是我們很重要的參考資訊。

2.3 音樂分析

在音樂分析中，本系統的分析項目包含了音域、音符密度、音符用量統計、音符轉移機率統計、旋律走向、調性以及和聲。其中調性和和聲這兩項目是無法經由直接統計來得到答案的項目，因此產生了許多尋找調性和和聲的演算法，以下分別介紹。

2.3.1 調性分析演算法

調性對於音樂分析來講是很重要的項目，一般來說，古典音樂作曲前通常都需要決定樂曲的調性，才開始進行作曲，再自動作曲上也是，需要先決定樂曲調性才能確定哪些音符是重要的，接著產生音樂。

因此在電腦自動化分析上，調性分析(Key-Finding)一直也是很熱門的題目，以下簡介幾個有名的調性分析方法。

2.3.1.1 LS Key Finding Algorithm

LS Key Finding 演算法是由 Longuet-Higgins 和 Steedman 在 1971 年所提出來的[10]，或許這也是最早出現的自動調性分析演算法。主要目的是用來找尋單聲部音樂的調性。

LS Key Finding 的做法非常直觀，它利用每種調性都有其不同的音階，並且音階內使用到的 Pitch Class 都是固定的概念來做調性分析。以 C 大調為例，音階內出現的 Pitch Class 如表 5：

表 5　C 大調音階 Pitch Class 表

音名	**C (Tonic Pitch)**	**D**	**E**	**F**	**G (Dominant Pitch)**	**A**	**B**
Pitch Class	0	2	4	5	7	9	11

當一首音樂輸入後，LS Key Finding 演算法會先列出所有音階，並從輸入音樂的第一個音開始依序處理，只要讀取到的音不存在於某個音階中，就會把該音階從答案的選項中刪除。刪除到最後如果只剩下一種調性，就猜測輸入音樂為該調性。如果剩下不只一種調性，或者全部調性都被刪去，那麼該演算法就以樂曲的第一個音為音階的主音(Tonic Pitch)或屬音(Dominant Pitch)直接猜測調性。

Longuet-Higgins 和 Steedman 輸入巴哈的平均律來測試，發現所有樂曲都能猜測到正確的調性。但事實上這個演算法很難適用於所有音樂。只要音樂出現不在音樂屬於的調性音階內的音，而且第一個音不是主音或屬音，就會產生錯誤結果；又或者音樂本身用的音符不多，最後剩下的答案還有好幾種，只要音樂的第一個音不是調性內的主音或屬音，一樣也會造成錯誤的情況。圖 5 當中的 A 狀況就是屬於上面所敘述的後者，而 B 則屬於前者。

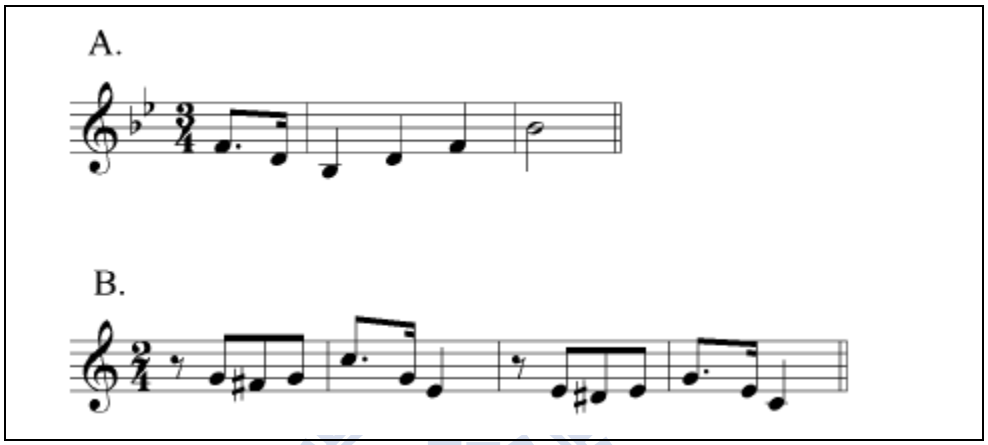


圖 5　LS Key Finding 演算法猜測錯誤音樂範例

另外 LS Key Finding 演算法在遇到小調音樂時也會出現問題，稍微熟悉樂理的人應該都會知道小調音階分為自然、和聲和旋律這三種小音階，而 LS Key Finding 演算法是直接使用和聲小音階來做，這樣一來就沒辦法考慮到其它兩種音階的可能性。

2.3.1.2 KS Key Finding Algorithm

相較於 LS Key Finding，KS Key Finding 演算法則是一個健全許多的調性猜測演算法。該演算法由 Krumhansl 和 Schmuckler 所提出，並且由 Krumhansl 在 1990 年時發表 [11]。

KS Key Finding 演算法利用「Key Profile」做為基礎來猜測調性。Key Profile 內容和十二個 Pitch Class 在每種調性伴演的角色有關。簡單的來說就是每種 Pitch Class 在某個調性之內是否為一個穩定或適合的音。

Key Profile 是建立在 Krumhasl 和 Kesslr 於 1982 年所做的一次實驗。該實驗首先會放一段調性明確的音樂，如終止式或音階，接著播放某個 Pitch Class 的音，並要求使用者對該測試音和前一段音樂的契合程度評分，測試完經過統計後把結果分為大調和小調兩種 Key Profile(或稱 KK Key Profile)，如圖 6、圖 7。

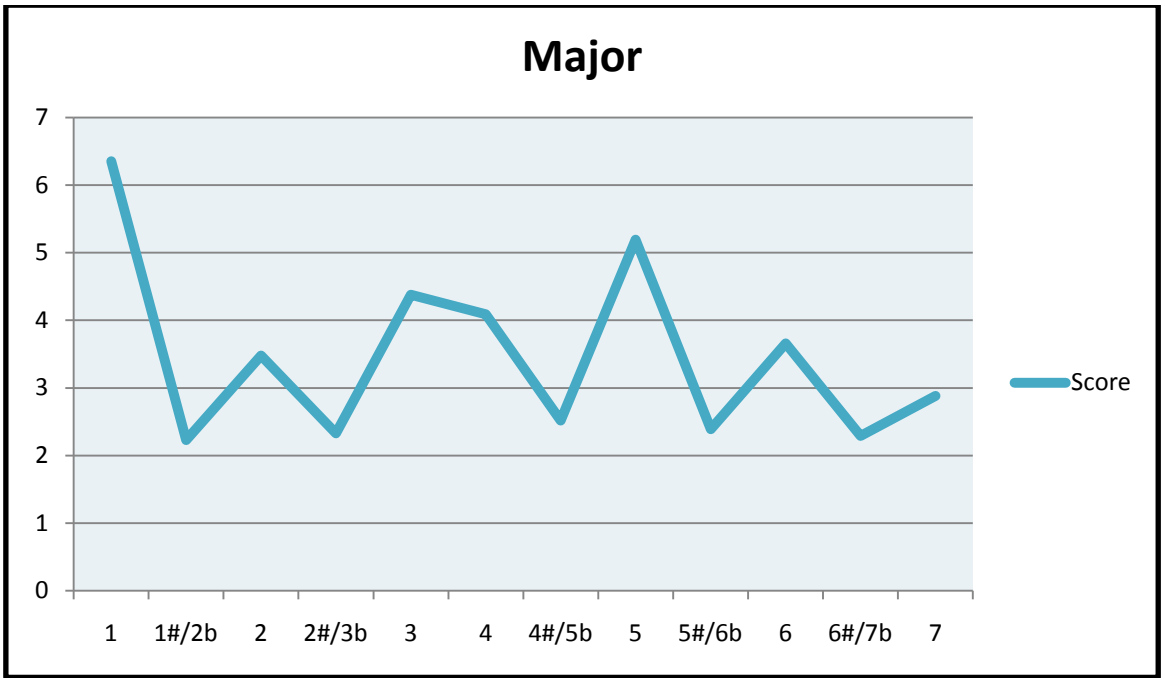


圖 6　KK Key Profile 大調部份

資料來源：Music and Probability [4]

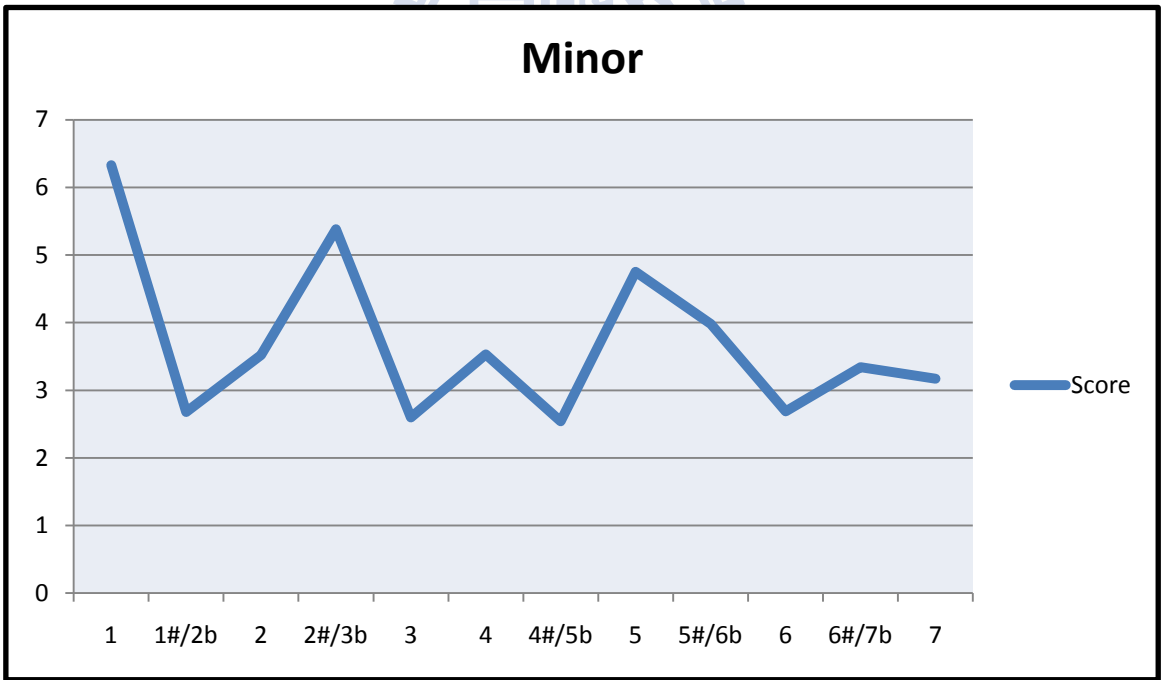


圖 7　KK Key Profile 小調部份

資料來源：Music and Probability [4]

有了大小調的 Key Profile 之後，KS Key Finding 演算法統計每種 Pitch Class 在輸入音樂的總使用量，算出每種 Pitch Class 相對應的比例，建立輸入樂曲的 Pitch Class 向量，

接著和 Key Profile 所提供的向量進行比對，比對的方式是利用標準的相關公式算出相關系數(correlation value: r)。

$$r = \frac{\sum(x-\bar{x})(y-\bar{y})}{(\sum(x-\bar{x})^2 \sum(y-\bar{y})^2)^{1/2}} \quad (3)$$

x: 輸入資料 $\bar{x}$: 輸入資料平均值 y: Key Profile 對應值 $\bar{y}$: Key Profile 資料平均值

經計算過後，所有的調性都會產生一個對應的相關系數，由 24 種調性中可比較出一個相關系數最高的調性，就可猜測該調性為輸入音樂的調性。

KS Key Finding 演算法比起 LS Key Finding 演算法，可使用的範圍更廣，而對於無轉調的音樂也有相當程度的準確率，本篇論文的實驗內容在調性部份也有利用這個演算法來做為比較的對像。

KS Key Finding 演算法提出後，許多人提出看法來嘗試改進這個演算法，如 David Temperley 在 2001 年所出版的書「The Cognition of Basic Musical Structure」中提到改變 Key Profile 的值、改變輸入音樂的權重計算方法可增加 KS Key Finding 演算法的準確率[5]，但無論如何，KS Key Finding 演算法在調性猜測這個題目中所利用的手法現在看來還是一個相當經典的解法，所以才會有許多的討論、改進及模仿出現。

2.3.1.3 The Spiral Array

利用 Spiral Array 來分析音樂的方法，是由 Elaine Chew 在 2000 年所發表的「Toward a Mathematical Model of Tonality」中所提出[6]。並且在接下來一系列論文利用 Spiral Array 來分析音樂各種參數。

Elaine Chew 所提出的 Spiral Array 其應用範圍很廣，Key Finding、Chord Finding、Pitch Spelling 和 Key Boundary 等項目都可利用 Spiral Array 進行分析。但本篇論文內調性分析部份主要輸入資料為無轉調音樂，因此我們只利用 Spiral Array 來分析音樂調性。

Spiral Array 延伸樂理五度圈的概念建立一個數學模型，並利用該模型做各種音樂分析。五度圈數學模型為

$$FC_x = 12i + 7X \% 12 \qquad X: \text{Pitch Class, } i: \text{integer} \quad (4)$$

在樂理上，五度圈是利用五度相生的概念所建立，以 C 為出發點為例，C – G –D – A – E – B…就是一連串五度相生的例子，而樂理上利用這個概念加上十二平均律會回到原點

的特性，建立了五度圈：

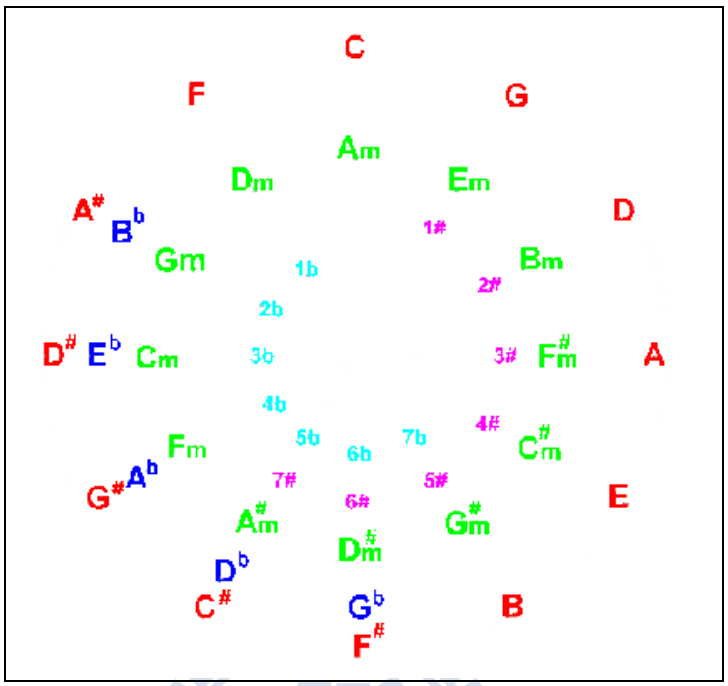


圖 8　五度圈

而 Elaine Chew 則是進而利用這個概念建立了螺旋狀的立體模型：

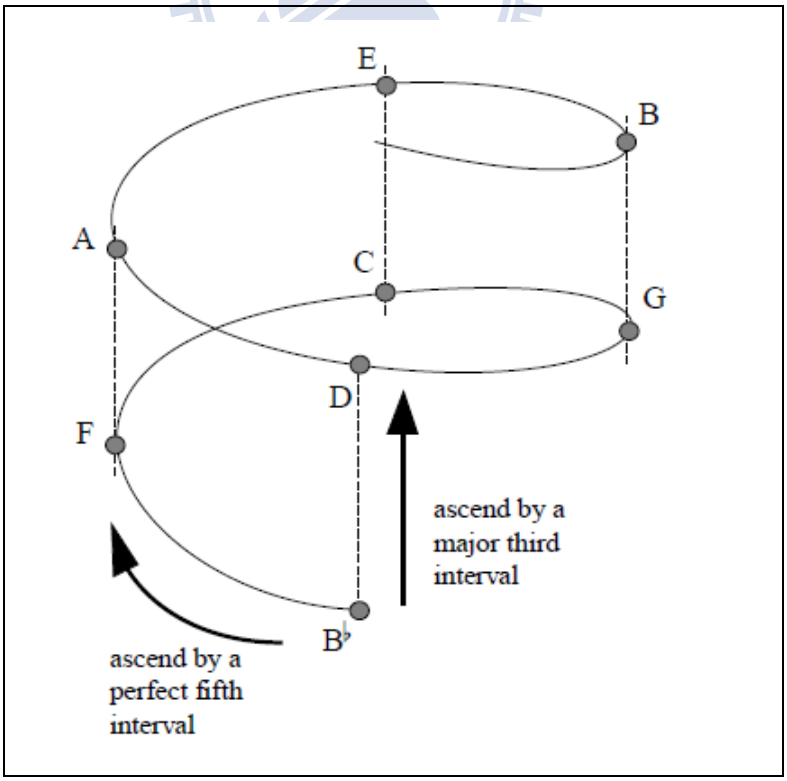


圖 9　Spiral Array Model

資料來源：Toward a Mathematical Model of Tonality [6]

在 Spiral Array 這個模型中將本來一圈就會回到原點的五度圈延長為三圈才會回到原點，但也因為這樣使這個模型多了一個特性，那就是上下音的音程為大三度音程。

建立這樣的模型後，就可利用這個模型算出不同調性的重心位置。方法是利用音階內最重要的三個和旋，分別是 I(Tonic Chord), IV(Subdominant Chord), V(Dominant Chord)級所個別產生的重心點，再算出這三點的重心，就產生了調的重心。

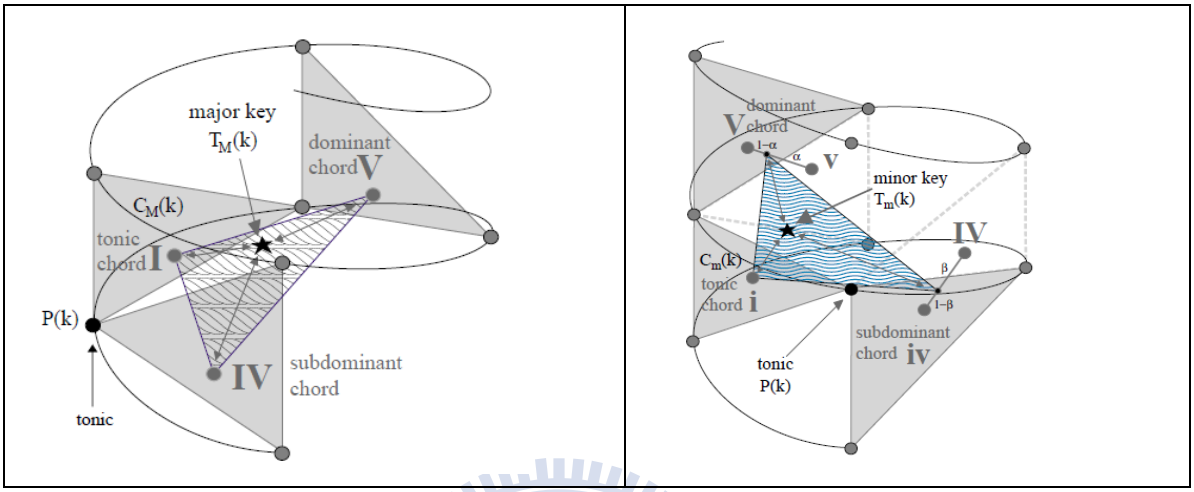


圖 10　Spiral Array 中大小調重心示意圖

資料來源：Toward a Mathematical Model of Tonality [6]

以這種方式，可計算出 24 種調性在 Spiral Array 中重心的位置，計算出之後，要猜測輸入音樂的調性時只需要計算輸入音樂的重心，計算公式如公式(5)所示。

$$c_i = \frac{1}{D_j}\sum_{j=1}^{i} d_j \cdot p_j \qquad (5)$$

公式(5)中 i 代表輸入音樂的所有音符數量，d_i為當前音符的長度，p_i為當前音符在 Spiral Array 當中的位置，c 就是當前音符的重心。就這樣子整首音樂依序處理，重心會在 Spiral Array 中一直移動，處理完整首音樂後看重心距離哪個調性的重心最接近，就可猜測出調性。

2.3.2 和聲分析演算法

和聲的分析是另一項重要的音樂參數分析。調性分析是用來分析整首音樂，或者樂曲的某一段落的參數，因此調性分析是傾向於分析音樂整體給人的感覺，但和聲的分析是對於音樂的細節去分析。如果用調性比喻一張圖片給我們的整體感覺，那麼和聲就是那張圖片內到底包含了什麼東西，以及這些東西是怎麼樣組織運用。因此兩者對於音樂分析都是不可或缺的部份。

在樂理上對和聲的分析範圍很廣，世界上存在著各式各樣的音樂，直到今天還不斷的在發展，也因此有各式各樣不同的和弦型態，除此之外，和弦的分析和音樂的脈絡有著很重要的關係，因為這些原因，造成創造一個完美的自動和聲分析演算法的難度。

和聲的分析當中除了準確率之外，有一點很重要的是如何去將音樂分段。以往的和聲分析中雖然提供了判斷和弦的辦法，但大部份相關文獻並沒有提到如何分段，如 Temperley 和 Sleator 在 1999 年所提出的方式中就存在這樣的狀況。這樣一來會產生一個問題，在音樂中和弦的使用並不會是固定長度，有可能好幾個小節只使用一個和弦，但也可能一個小節出現好幾個和弦，因此如果不解決分段的問題也沒辦法使和聲分析完全自動化。

Bryan Pardo 和 William P. Birmingham 在「Algorithm for Chordal Analysis」這篇論文中提出了一種分析音樂和聲的方式[12]。

這篇論文和以往方式較不同之處在於，它將和聲的分析和音樂分段整合成為一種新的和聲分析方法。在論文中除了闡述他們如何分析一個段落內的和聲之外，還將找尋和聲的方式和分段做結合，而且分段方式是有彈性的，並不會從頭到尾都是以固定的長度來分段，而是可以有長有短、找出整體最佳的分段方式，這樣一來解決了分段問題，也因此可以實現自動化的和聲分析。另外他們的方式當中，和弦型態是可以擴充的，也解決了和聲型態會增加的問題，所以我們選擇這個方式來實現分析系統當中的和聲分析項目。

此論文的方式分為兩個階段，分別為和弦標示(Labeling)和分段(Segmentation)。

在和弦標示(Labeling)部份，首先需要準備各種和弦樣版做為比對對象，在該論文內選擇了六種常見和弦做為比對對象，而如果使用者有需要也可以自行新增其它和弦的樣版。

表 6　和弦樣版

Chord Name	**Template Representation**
Major Triad	<0, 4, 7>
Dom7	<0, 4, 7, 10>
Minor Triad	<0, 3, 7>
Half Diminished 7th	<0, 3, 6, 10>
Diminished Triad	<0, 3, 6>
Fully Diminished 7th	<0, 3, 6, 9>

資料來源：Algorithm for Chordal Analysis [12]

表 6 中只列出每種和弦音與音之間的相對位置，這也是和弦的基本定義，然而在比對時，每種和弦樣版都需要位移到以十二個 Pitch Class 為根音的型態來做比對，以大三和弦來說，樣版{0, 4, 7}表示和弦內三個音所相差的距離，但在比對時就需要比對{0, 4, 7}, {1, 5, 8}, {2, 6, 9}…等十二種樣版，所以以六個樣版為例，在比對時就需要比對七十二次。

由於和弦的轉換只會發生在一個音開始(Note On)或結束(Note Off)的地方，因此利用這個特性把所有的 Note On /Off 點定為一個分割點(Partition Point)，以圖 11 為例，該樂曲就存在六個分割點。

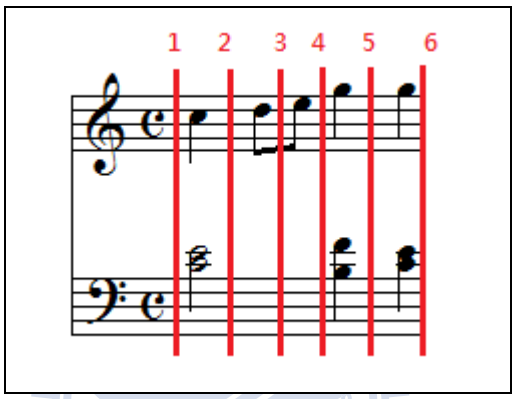


圖 11　分割點示意圖

接著開始計算各種可能的段落分割方式的分數，各種分割方式的意思是，例如圖 11 當中有六個分割點，可能的分割方式有((1,2)(2,3)(3,4)(4,5)(5,6))、((1,2,3)(3,4,5)(5,6))、((1,2,3,4)(4,5,6))…等，因此如果以六個分割點為例，可能的方式有 6+5+4+3+2+1 種。

計算每種分段分數的公式為：

$$S = P - (M + N) \tag{6}$$

其中 S 代表總分，P(Positive Evidence)代表比對的段落內有出現在正在比對的和弦樣版中的權重總合，M(Misses)代表出現在比對中的和弦樣版內，但沒出現在段落內的數目總和，N(Negative Evidence)代表出現在段落內，但沒出現在正在比對的和弦樣版中的權重總合。

以這種方式來計算每段落對應到每種和弦樣版的分數，圖 11 為例，假設正在比對的段落為 1 和 4 這兩個分割點所組成的段落，正在比對的樣版為以 A 音為根音的小三和弦(A Minor Triad: 9,0,4)，段落內出現的音為{C4, E4, C5, D5, E5}，所對應到的權重為{3, 3, 1, 1, 1}，因為 C4 和 E4 的時間長度跨越了 1 到 4 這個大段落內的三個小段，因此權重

為 3，C5, D5, E5 則都沒有跨越任何小段落，因此權重為 1。可算出 P=Weight(C4)+Weight(E4) +Weight(C5)+Weight(E5)=3+3+1+1=8，N=Weight(D5)=1，M=1，得到 S=8-(1+1)=6。

計算完每種分段方式的分數後，接下來要進行分段(Segmentation)的工作。在這篇論文中，是利用圖學的方式來找出分段方法。

首先需要建立一個有向圖的資料結構，並且把每個分割點當做一個點，給予每個點連往其後所有點的有向邊。

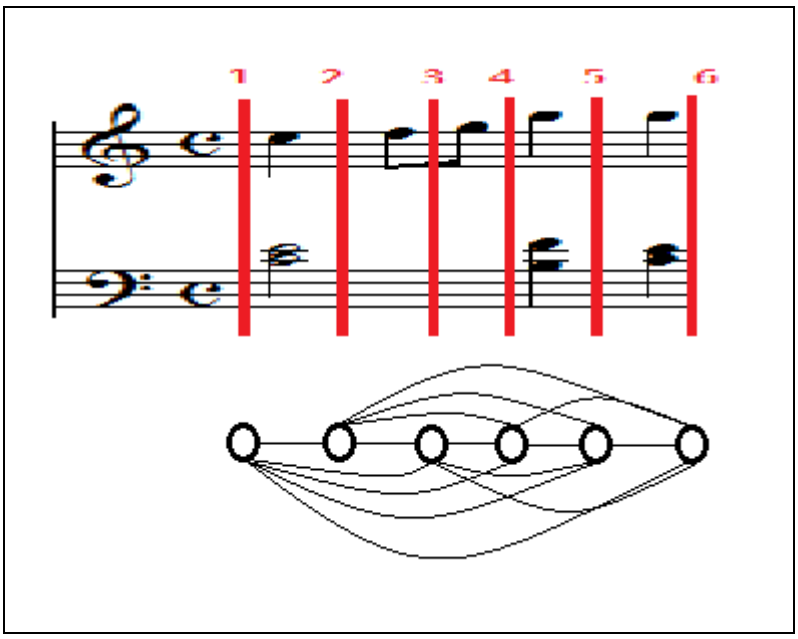


圖 12　分割點轉為圖學示意圖

由於已經計算過所有可能分段的分數，因此每個邊都可以利用分數來標示其權重，作者假設以這種方式所計算的權重，分數越高代表是越合適的和弦及分段方式，而以整體來看，如果能找出一條最高分的路徑，或許就是最佳的分段方式。

在演算法的動態規劃(Dynamic Programming)手法當中，Relaxation 演算法就是用來在有向的拓樸當中找出最佳路徑的演算法，在這篇論文當中就以這個演算法來尋找最佳路徑，以下簡單用圖 13 為例說明 Relaxation 演算法如何運作。

由圖 13 中可看出，如此一來在找尋和弦時也一起把分段方式給找出來，相較於以往和弦找尋演算法分段方式並無明確說明，這個方式我們認為這是很好的方法，因此在本論文分析系統當中使用這個演算法。

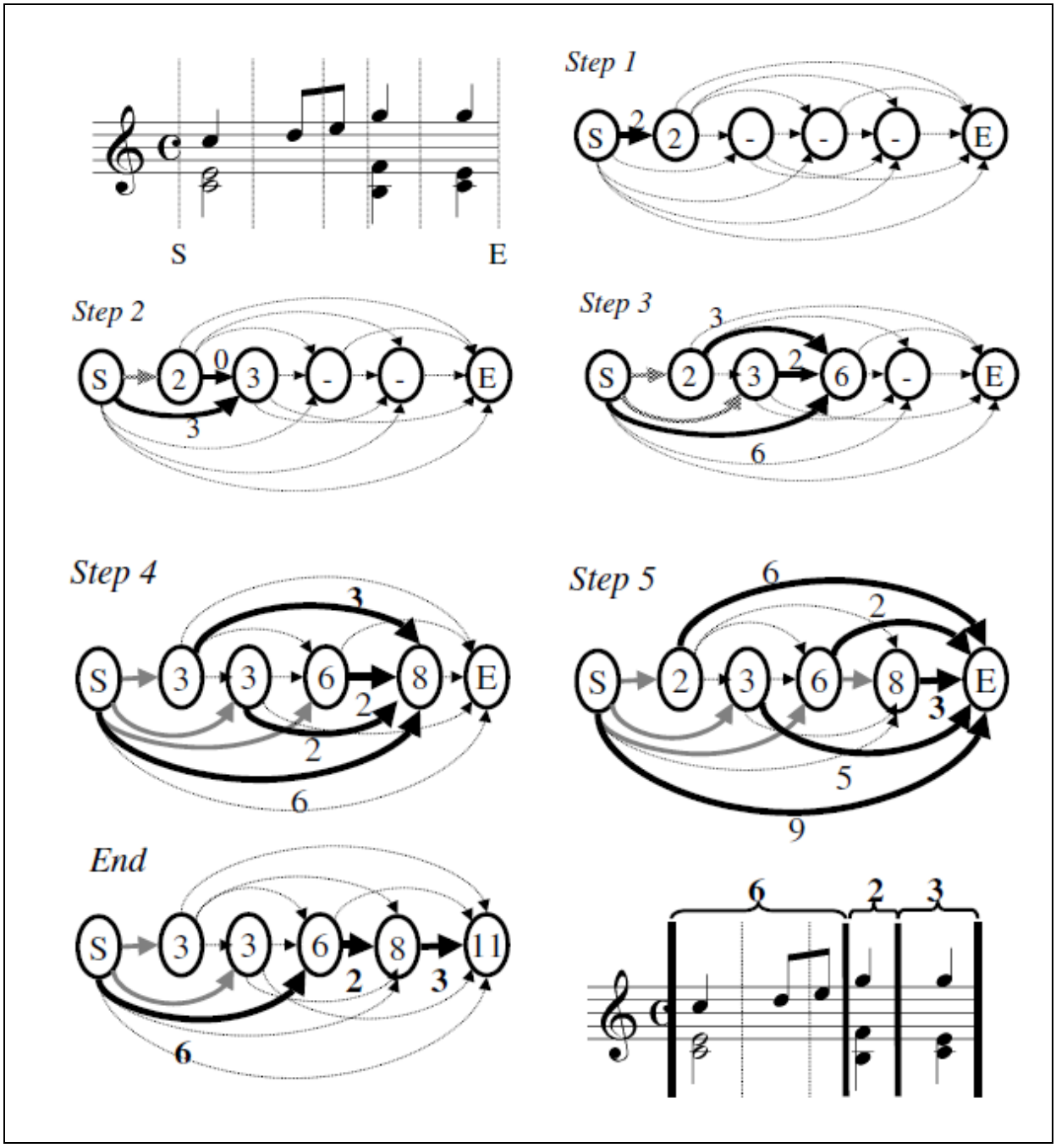


圖 13　最佳路徑尋找範例

2.4 音樂簡化

樂曲的簡化對於樂曲分析是必要的，簡化後的樂曲可顯示出旋律、節奏與和弦的基本架構，也更清楚得看出音樂連接的關係。因此簡化不但對於學習音樂的人非常重要，我們任為在電腦自動化分析中也是很重要的一步。

以往樂理上並沒有固定的規則來的教導如何簡化音樂，在學習樂理的過程中，會學到一首樂曲如何由簡單的結構(如和聲的行進)，一步一步建構出龐大的樂曲。在分析簡化作品時，則是利用這種經驗來反向找出樂曲的骨架。

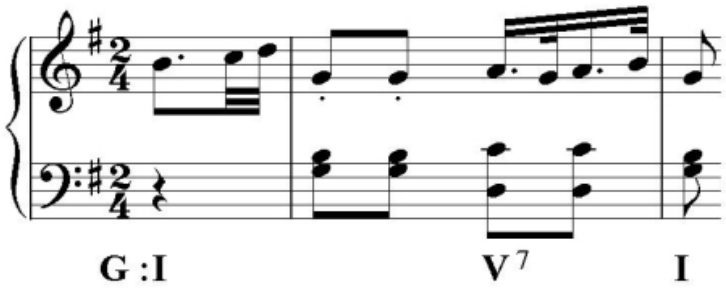
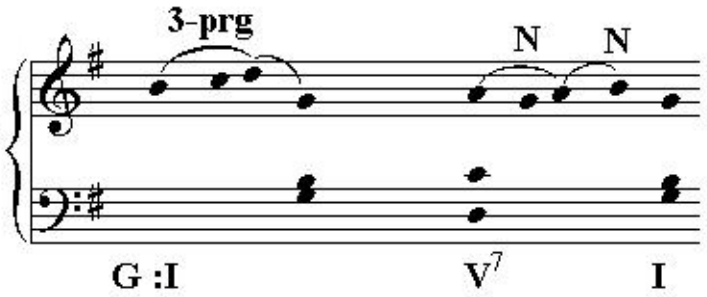
然而 Schenker 提出一種較有系統的樂曲簡化方式，稱為 Schenkerian Analysis[13]。這種分析方式雖然靠的還是對於音樂的知識，但至少以一步一步的方式來教導該如何判別什麼音可以被簡化，什麼可以被留下，我們希望利用電腦自動簡化音樂，因此參考 Schenkerian Analysis，以及以往人們利用電腦實作 Schenkerian Analysis 的例子對我們來說是很重要的。

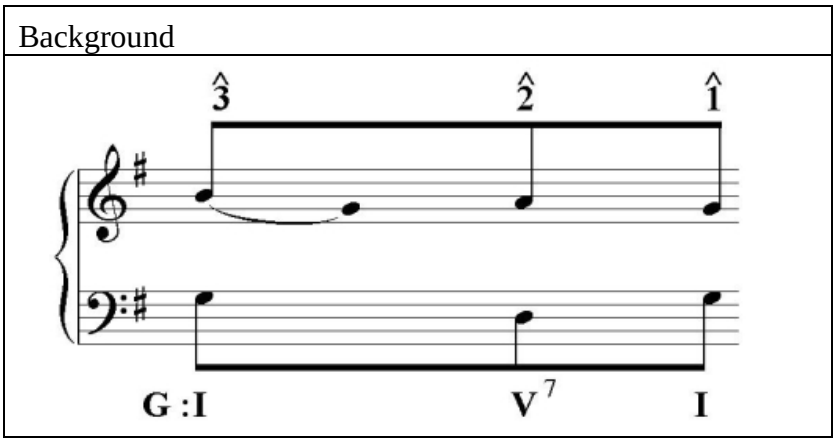
2.4.1 Schenkerian Analysis

Schenkerian Analysis 分析方式把樂曲分為 Foreground、Middleground 和 Background 三種階段[13]。

其中 Foreground 是指完全未經簡化過的樂曲。分析者可利用調性、和弦以及判斷裝飾音來找出樂曲內各種相依性，並標示出來。例如 X 音和 Y 音的相依性是，因為 X 音的存在，Y 音才會存在，則可判斷 X 在樂曲中的結構重要性比 Y 來的大，可以把 Y 刪去，經過刪減後的樂曲就成了 Middleground。而 Middleground 這個階段可以不斷的重覆，刪減到最後沒有音符可以刪去，留下來的都是骨幹音，這個結果就是 Background，也就是簡化後的音樂。

表 7　Schenkerian Analysis 步驟示意圖

Foreground (原始輸入)

Middleground




資料來源：Tom Pankhurst's Guide to Schenkerian Analysis [13]

Schenkerian Analysis 雖然步驟簡單，但其實是把樂理中簡化音樂的方式用精華的方式呈現，因此在電腦自動分析上也有許多人以實做 Schenkerian Analysis 為題提出很多相關演算法。

2.4.2 Automated Schenkerian Analysis

2.4.2.1 IVI Algorithm

IVI 演算法是由 Philip. B. Kirlin 和 Paul E. Utgoff 在 2008 年所發表的論文中提到的自動化 Schenkerian Analysis 演算法[14]。會取名為 IVI 的原因是 Schenkerian Analysis 中認為所有的樂曲最終都可簡化為 I-V-I 這樣簡單的和弦組成。

IVI Algorithm 利用圖學的方式來實做 Schenkerian Analysis，首先輸入資料的每個音都是一個點，連接每個點的邊可能有兩種，第一種為有向邊，表示依賴關係，另一種為無向邊，表示同階層的關係。以圖 14 為例，圖中上方兩個點表示同為第二階層的音，因此以沒有方向性的邊相連，下方的點則為階層一的音，而上方的兩個點指向它則表示上方的兩個點是依賴下方的點而生，因此在簡化時就可把上方的兩個點給簡化掉。

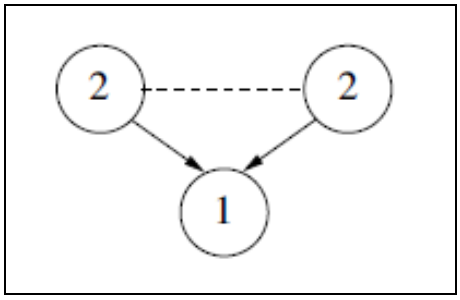


圖 14　IVI 資料結構示意圖

IVI 演算法中，一開始每個點階層都是相同的，因此在簡化前需要先進行一些前置作業，如和聲分析、旋律分析、節奏分析以及終止式分析等等。分析完後每個點會有一個對應的權重，再依據這個權重自動找出每個音所屬的階層，進一步找出最具結構代表性的路徑。

2.4.2.2 SchenkeR Algorithm

SchenkeR Algorithm 是 Christy Keele 在 2006 年所提出的自動 Schenkerian Analysis 方法[16]，其方法內容是列出一條條規則，包括判斷節奏、音長、重複音、旋律和低音的關係、和聲之後給定權重，最後設定閥值來過濾掉不想要的音，再由剩下的音找出結構。

2.4.3 自動簡化系統討論

上述提到的論文中，皆為實作 Schenkerian Analysis 的演算法，也因此需要許多先行分析。然而本文中所提到的簡化系統是希望能先簡化音樂，再進行其它項目的分析，而其中很重要的就是希望能先簡化樂曲後再行調性和和弦的分析，因此並不希望採用一般事先分析調性和和弦的方法，反而希望利用簡化的結果使調性和和弦的分析能更為準確。

然而 IVI 演算法需要先行判斷和弦、調性、終止式；SchenkeR 演算法則需要有另一個聲部(低音部)的輔助，且也需要事先判斷調性、和旋才能給定權重。

雖然本文簡化作法和 SchenkeR 的方式有些類似，同樣都是先依據某些規則判斷每個音的權重之後再利用閥值判斷可刪去的音，但 SchenkeR 中並無提出一個可以自動找出適合的閥值的方式，本論文有提出自動尋找閥值的演算法，且本論文給予權重的方式也不需要依賴低音聲部或和弦和調性的判斷。

第三章、 研究方法

在文獻回顧中，第一部份的自動作曲系統中我們可歸納出對於自動作曲較為重要的音樂參數項目。由於我們的分析系統是希望以分析自動作曲所需的參數為主，因此經由研究這些自動作曲的軟體，表 8 中整理出一些對於自動作曲所需要的重要的音樂分析項目：

表 8　音樂分析項目表

基本分析項目	音域分析、音樂密度分析、音符使用比例分析、音符轉移機率分析、音樂旋律走向分析
進階分析項目	調性分析、和聲分析、節奏分析、和旋轉移機率分析、終止式分析
架構分析項目	動機分析、樂曲架構分析

在這些項目中，基本分析項目雖然重要，但都是可經由統計所得到的項目，因此在我們的分析系統中會包含所有的基本分析項目。

進階分析項目，在我們的系統中有調性和和聲的分析，而節奏的分析雖然我們也有做相關的研究，但因為和論文中實驗部份較無相關，因此還未整合到這個系統當中，另外和弦轉移機率和終止式分析，這兩項目其實是包含在和聲分析當中，因此在我們的系統中也可進行分析。

在進行分析的研究時，我們發現在分析上有很多的不準確是因為音樂裝飾音所造成，作曲家們為了讓音樂聽起來更流暢，更有個人風格，往往會在音樂內加入許多裝飾、連結性質的音符來連接樂曲的骨幹音。而這些裝飾性質的音符如果越多，則在分析上往往會造成更多的失誤，例如調性的分析、和聲的分析等，如果加入太多裝飾音則會更難以電腦來判定。因此我們也提出了樂曲的簡化演算法 AMSA，希望能經由簡化使得音樂分析更為準確。

至於架構分析，這部分的研究目前較少相關研究，但我們也已著手進行，不過目前還未整合到系統中，但在我們系統中，簡化系統部份其實對架構分析有很重要的意義，希望以後能更進一步使用簡化系統來進行架構分析。

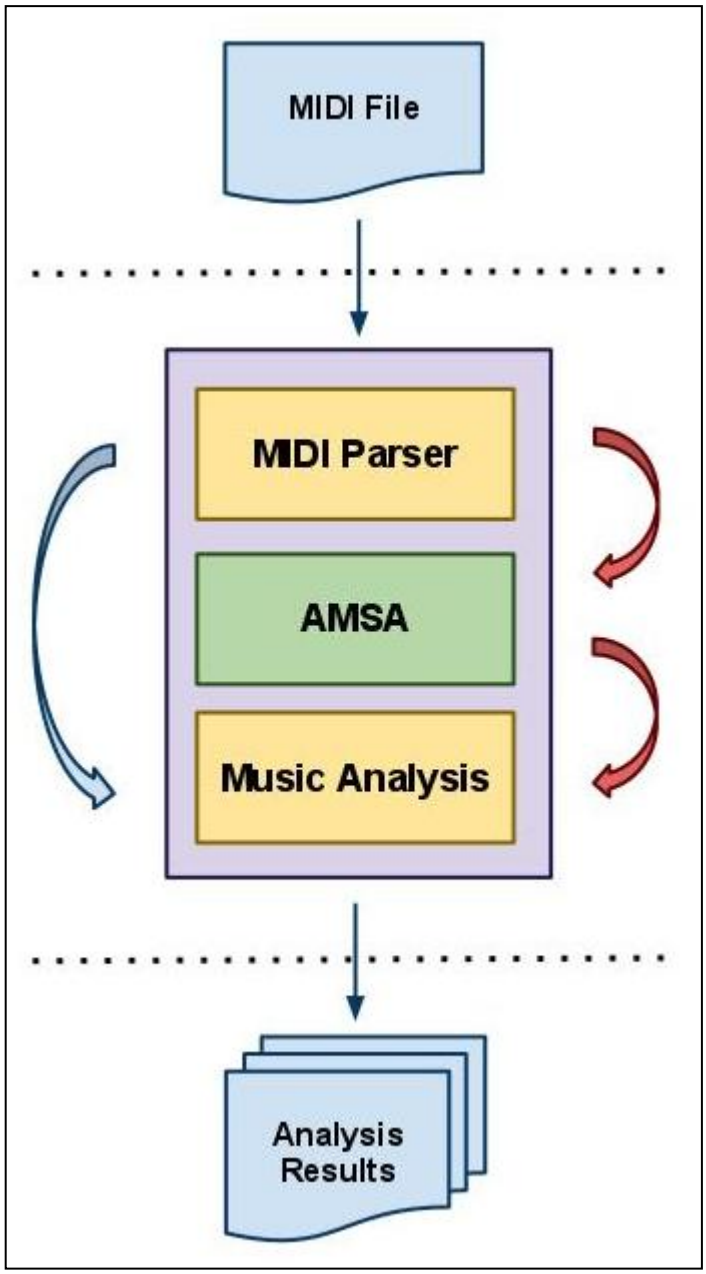


圖 15　系統流程圖

圖 15 系統流程圖中，我們使用的輸入檔案使用 MIDI 檔案，其理由在相關研究中已提過，再此不詳細說明。

輸入 MIDI 檔案後會產生一個中介檔案方便我們進行分析，接著可以直接進行音樂分析，也可先進行音樂簡化後再進行音樂分析。

以下我們分兩個部份說明這個系統，第一個部份先說明音樂分析系統和 MIDI 轉檔系統如何運作，第二部份詳細說明簡化音樂的演算法 AMSA 和簡化系統。

3.1 音樂分析與 MIDI 轉檔系統

3.1.1 系統架構

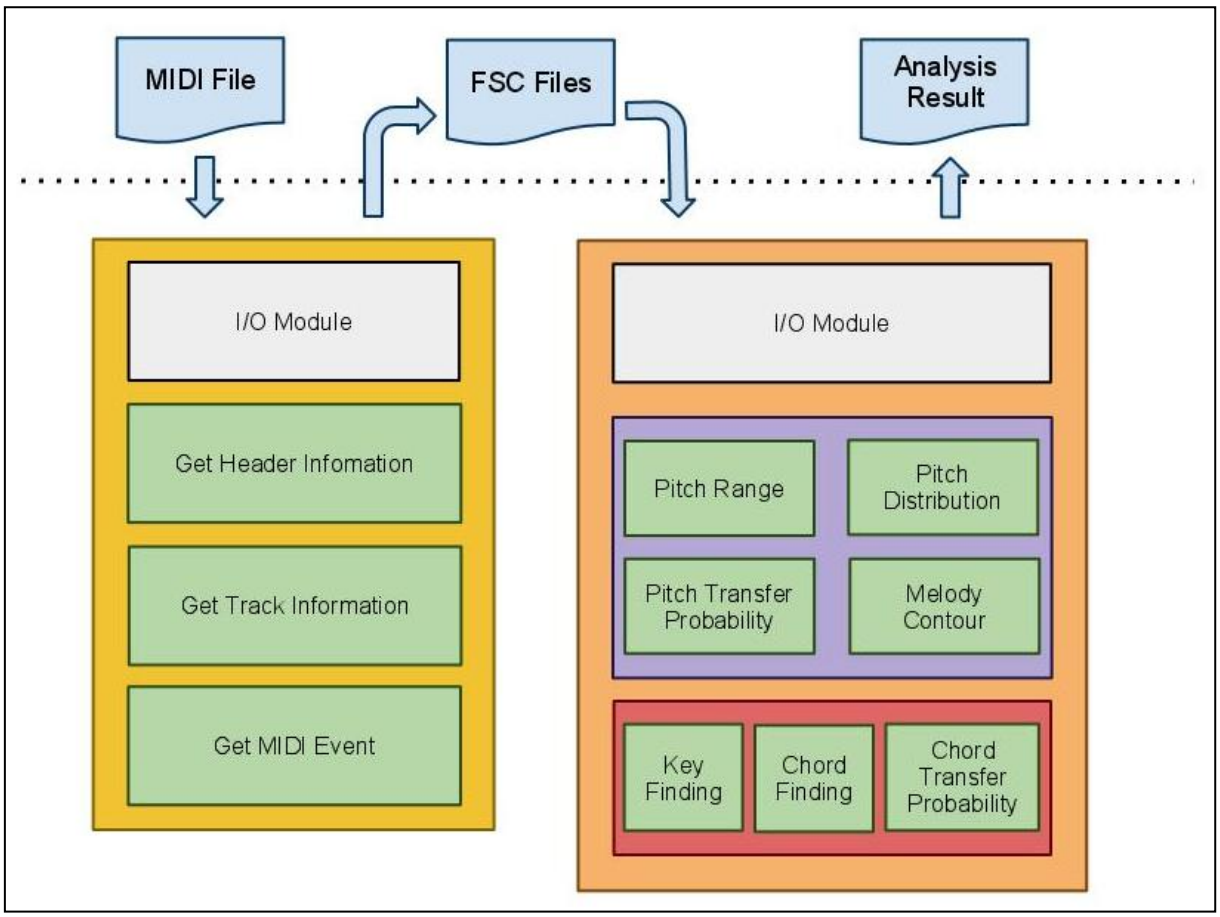


圖 16　音樂分析系統架構及流程圖

圖 16 為本論文的分析系統架構圖，其中未包含簡化系統部份。

分析系統主要分為兩部份，第一部份為 MIDI 檔的轉換(MIDI Parser)，第二部份為 MIDI 分析(Music Analysis)。

MIDI 轉換主要工作為輸入 MIDI 檔案，利用關鍵的 MIDI 事件轉換出我們所需要的中介檔案(FSC 檔案)。其中所需的資訊只要對 MIDI 檔案夠熟悉就能完整取得，因此在這邊不詳談。圖 17 為中介檔的格式示意圖：

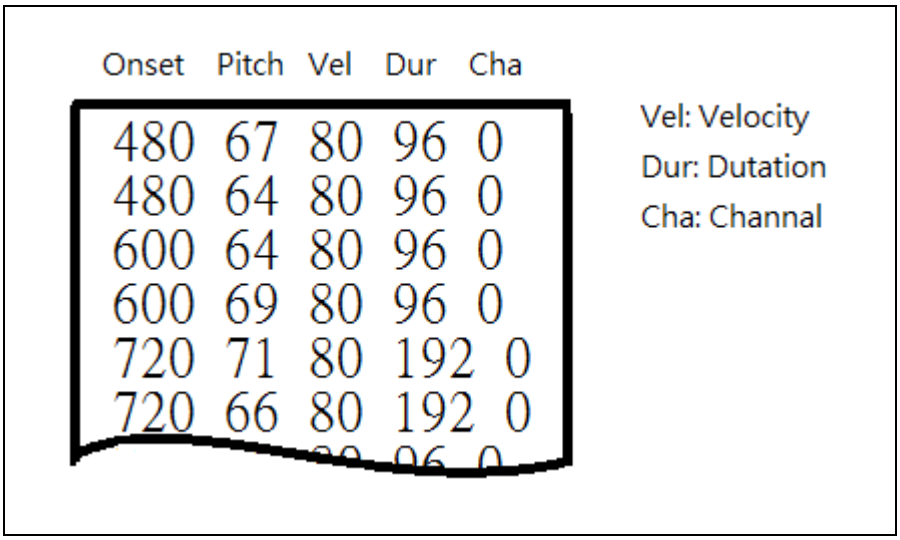


圖 17　FSC 檔案格式示意圖

而音樂分析部份因為較為重要，以下分別說明基本分析項目和進階分析項目。

3.1.2 基本分析項目

表 9　基本分析項目功能說明

Function Name	功能
PitchRange	1. 回傳最大最小 Pitch 值 2. 回傳 Pitch 數量及密度
PitchClassDistribution	1. 回傳各 pitch class 的使用次數 2. 回傳各 pitch 的使用次數 3. 回傳各 pitch class 的使用總長度 4. 回傳各 pitch 的使用總長度
PitchTransitionMatrix	回傳 pitch 轉換的機率矩陣
MelodyContour	1. 利用時間長度為單位統計每段落的 Pitch 平均值 2. 利用 pitch 數量為單位統計每段落的 pitch 平均值

表 9 內包含了分析系統當中的基本分析項目，這些函數的功能做法都很簡單，大部份都是利用簡單的統計就可達到目的。但是其中所統計的參數對於自動作曲都是很重要的依據，在常見的自動作曲軟體中大部份都需要大量利用機率來控制各種參數，例如下個音該產生什麼音，會利用到音符轉換機率矩陣來決定，而決定旋律走向需要用到 Melody Contour 參數等等，因此雖然取得這些參數的做法簡單，但如果想要利用自動分析的結果進行自動作曲，這些參數還是非常重要且必要的分析項目。

3.1.3 Key Finding

在我們的分析系統中，調性分析部份我們實作 KS Key Finding[11]，Spiral Array[6] 部份雖然不存在本系統但有另外實作在其它系統當中，因此比較實驗結果實我們兩種演算法都會比較。

KS Key Finding 的做法在相關研究討論中已詳細說明，因此這裡只利用流程圖的方式來表示 KS Key Finding 這個函數如何運作。

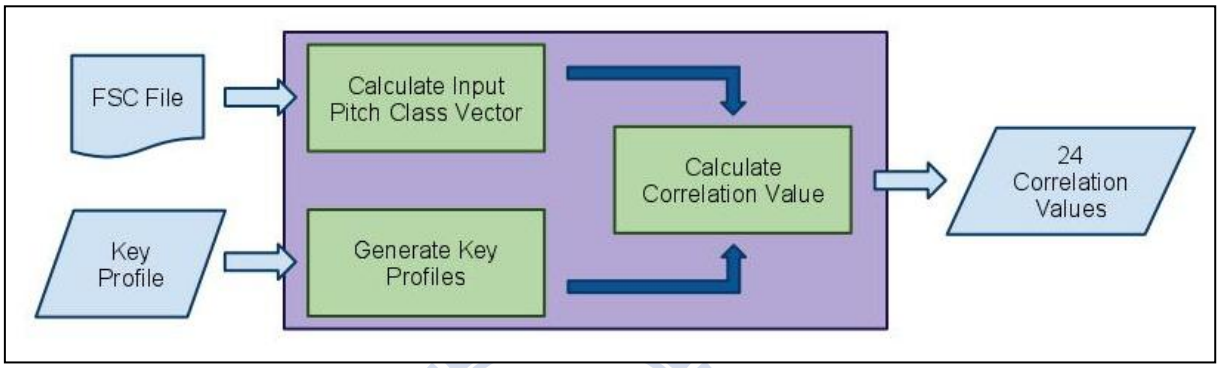


圖 18　KS Key Finding 流程圖

圖 17 當中，輸入 FSC 檔後會利用每個音的音長來計算出十二種 Pitch Class 分別的比例向量，而 Key Profile 輸入後需要位移到 24 種調性所對應到的 Pitch Class 向量來比對。

根據產生出的 24 個 Key Profile 和輸入的 Pitch Class 向量，利用公式(3)算出 24 個相對應的相關系數，最後輸出 24 種調性的計算結果。可從中取出相關系數最高的調性做為答案，也可觀察輸入音樂和每種調性的相關性高不高。

3.1.4 Chord Finding

本音樂分析系統中的和聲分析演算法是實作「Algorithm of Chordal Analysis」[12] 這篇論文的方式，其詳細做法已在相關研究中討論過，因此在這利用流程圖表示實做的方式及流程。

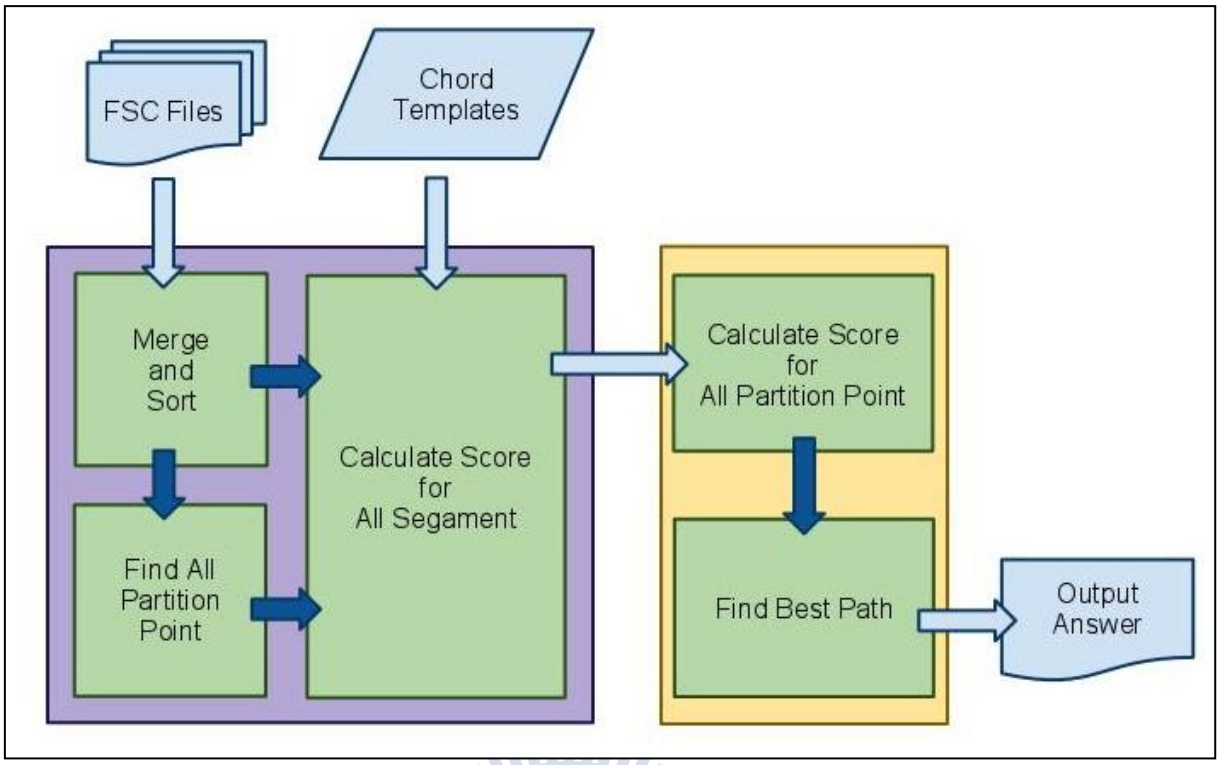


圖 19　和聲分析流程圖

如圖 19 所示，過程中第一階段會先讀取輸入的 FSC 檔案，由於和聲分析有時可能會需要輸入好幾個 MIDI 音軌的檔案，因此設計為可輸入多個 FSC 檔。

輸入後會先進行 FSC 檔案的合併以及排序，接著找出所有的分割點，找完之後就會開始對每種分割組合，利用公式(6)進行分數計算，並取出最高分和弦紀錄下來，第一階段工作到此完成，產生一個中介檔案包含所有分割方式的分數及和弦。

第二階段利用第一階段所產生的檔案，計算每個分割點的最佳路徑以及最高分，最後從最後一個分割點開始往回找出整段音樂的最佳路徑，完成分段以及和弦猜測的工作。

本系統也利用 Chord Finding 的結果統計出和弦轉移機率，做為另一種音樂參數資料。

3.2 AMSA

在音樂的自動分析中，特別是和弦的分析，很容易因為裝飾音的出現造成判斷上的錯誤，以我們所使用的和弦分析方式來說，很有可能因為裝飾音造成和弦辨識錯誤或者是誤認裝飾音為一個單獨的和弦。而在前面所提到的調性分析方法當中，KS Key Finding 有可能因為調性特徵的音在統計上不夠明顯造成誤差，Spiral Array 也可能因為裝飾音而使得重心偏移太多造成判斷錯誤。

以往音樂的簡化需要先進行和聲和調性的分析，但我們的想法則是希望反過來先進行簡化，再進行調性和和聲的分析，希望能增加這兩個分析項目的準確率。

我們參考了樂理的簡化方式，發現主要是經過分析和聲、調性之後，找出音樂的相依性，進一步找出什麼音是裝飾音什麼音是骨幹音，而裝飾音在作曲上有許多常見的手法，如重複音(Repeated Tone)、鄰音(Neighbor Tone)和經過音(Passing Tone)都有明確的定義，因此我們把裝飾音內這幾個項目加入了簡化系統的判斷規則當中，另外還加入了音階這個判斷規則。

因為我們無法先行判斷調性和和聲，如果單純利用上述的裝飾音規則判斷會產生許多的誤判情形，所以我們加入了音長和節奏這兩種參數做為判斷的依據，希望可以利用這兩種參數來平衡權重減少誤判的情況。

將這些想法整合，我們提出了 AMSA 這個音樂自動簡化演算法，也實作出了音樂自動簡化的系統。

3.2.1 AMSA：系統流程

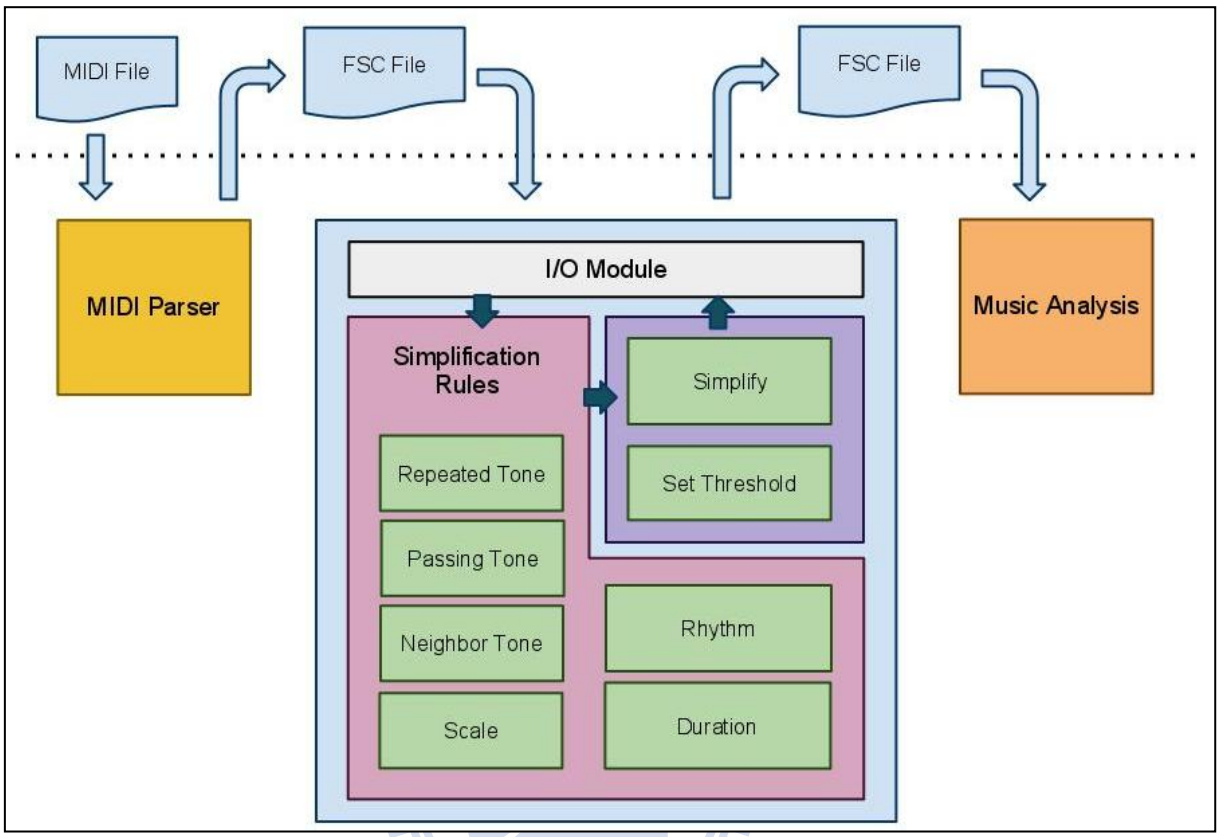


圖 20　音樂簡化系統流程圖

在簡化系統中我們使用的輸入檔案也是 MIDI 轉換出來的中介檔，輸入後會開始由各種規則判斷權重，這些規則可以選擇使用或不使用。

判斷完之後輸入音樂內每個音會產生一個權重，我們可利用這些權重來過濾我們不要的音，留下我們想要的音。而權重可直接設定固定的閥值來進行過率，也可利用我們提出的方式來動態尋找閥值。

最後可輸出簡化後的檔案，檔案格式一樣為中介檔的格式，方便後續的音樂分析工作。

3.2.2 AMSA：資料結構

在簡化系統中，我們給每個音符一個結構(struct)，內容除了包含本來中介檔所存在的觸發時間(onset)、音高(pitch)、音長(duration)、音量 velocity 和聲道(channel)之外，增加了權重(weight)，用來紀錄每個音在經過判斷後得到的權重。

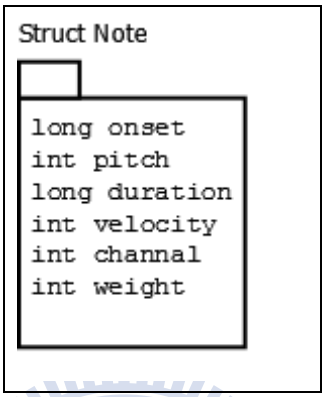


圖 21　音樂簡化資料結構

而音樂中常常有同時發聲的情況出現，然而在 MIDI 當中就算是同時發出聲音，MIDI 事件也是循序產生，如果我們依序讀取很可能在簡化上造成判斷錯誤，因此在讀取的同時，我們遇到同時產生的事件時我們會建立有兩種方向的資料結構，水平方向為發聲時間不同時的音符，垂直則為同時發聲的音，以利簡化規則的判斷。圖 22 為一個簡單的範例。

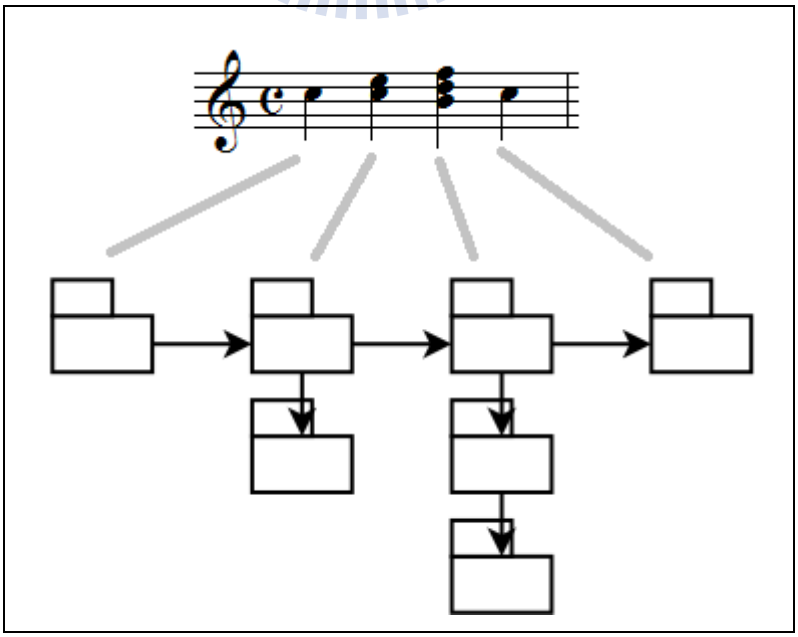


圖 22　音樂簡化資料結構範例

3.2.2 AMSA 簡化規則：重複音

重複音(Repeated Tone)在音樂上常用來強調某個音的重要性，或者是為了增加音樂的華麗，把一個長音拆解為好幾個較短的音來演奏，是一種利用節奏改變來裝飾音樂的手法。如果某個音連續的重複，那麼在音樂架構的分析上我們通常會把連續重複的音合併視為一個長度較長的音。圖 23 為重覆音的範例：

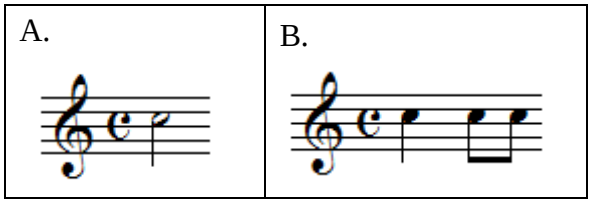


圖 23　重複音

圖 23(B)中可能是在樂譜上我們所看到的內容，但在音樂分析上其實我們可以把三個重複音合併視為圖 23(A)中的樣子，這樣一來可更清楚看到音樂的架構，因此在自動簡化上如果系統遇到重複音我們會給予第一個出現的音較高的權重，後面出現的音較低的權重，並且將後面的音的音長加到第一個音當中，如此一來在簡化後就能將重複音合併為同一個音：

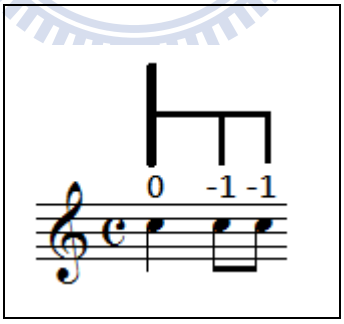


圖 24　重複音權重示意圖

然而在本次論文的實驗中並沒有用到重複音的判別，原因在於本次論文主要實驗的對象是調性分析及和聲分析，在 KS Key Finding 中同樣 Pitch Class 的音是以樂曲內的總長度來計算，因此有無合併並不會影響實驗結果。而和聲分析因為是對樂曲細微的部份做分析，如果將重複音合併可能會影響到和聲分析演算法中分段的結果，造成不可預期的錯誤，因此在本篇論文的實驗中皆不對樂曲做重複音的判斷。未來如果利用 AMSA 對樂曲進行架構上的分析，我們就會加入重複音的判斷。

3.2.3 AMSA 簡化規則：經過音

經過音(Passing Tone)在樂理上有明確的定義，其功能在於填滿由一個音行進到下一個音時所產生的三度音程[17]。

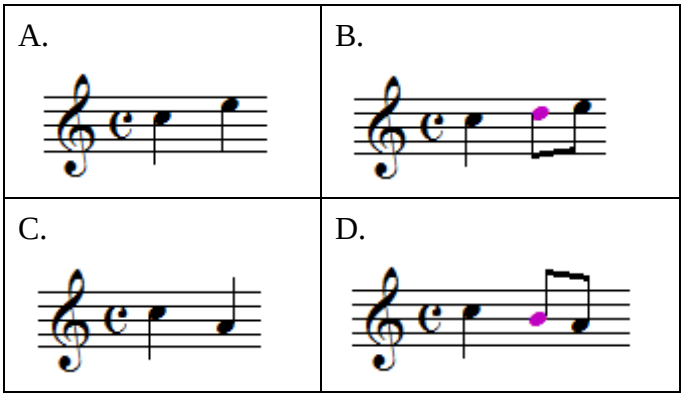


圖 25　經過音

圖 25(A)、圖 25(C)為未利用經過音裝飾前的音樂，圖 25(B)、圖 25(D)則是利用經過音裝飾後的音樂，圖 25 中顏色不同的音符為經過音。利用經過音裝飾可增加樂曲的豐富性以及圓滑性，是非常常見的作曲手法。

因此在樂曲的簡化系統中，遇到經過音的狀況時，系統會給予經過音較低的權重，如圖 26 所示：

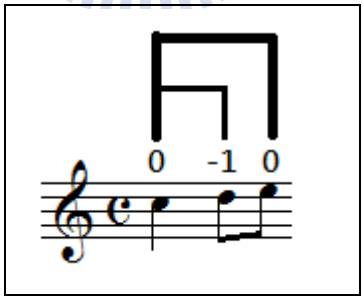


圖 26　經過音權重示意圖

而其中判斷的條件為當第一個音和第三個音相差為大三度(音高相減等於 4)或小三度(音高相減等於 3)時，如果第二個音不等於前後兩音，且音高在兩音之間，那麼第二個音就會被視為經過音，並給予較低的權重。

```
If ( abs(n1-n3) = 3 or 4)
        if ( n2<n1 and n2>n3 )
                n2->weight --;
        else if ( n2>n1 and n2<n3 )
                n2->weight --;
```

n1,n2,n3 分別代表正在判斷的三個連續音

圖 27　經過音判斷演算法

3.2.4 AMSA 簡化規則：鄰音

鄰音(Neighbor Tone)在樂理上亦有其明確定義，鄰音的功能為利用上下級進(Stepwise Progression)的方式來修飾一個靜止的音[17]，通常可分為上鄰音(UN, Upper Neighbor Tone)和下鄰音(LN, Lower Neighbor Tone)兩種：

	UN	LN
A.	B.	C.

圖 28　鄰音

圖 28(A)為未利用鄰音裝飾之前的樂譜，圖 28 (B)為上鄰音，圖 28C)為下鄰音，其中圖 28 (B)、(C)中顏色不同的部份為鄰音。做為一個增加音樂豐富性的手法，鄰音也是很常見的一種裝飾音。

在樂曲簡化系統中，會給予鄰音較低的權重：

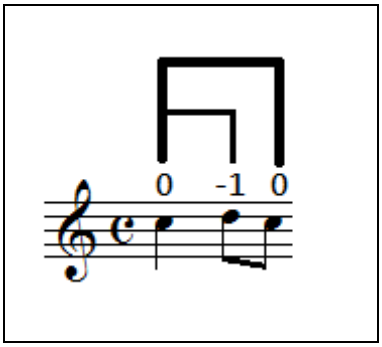


圖 29　鄰音權重示意圖

判定條件為當第一音和第三音相同，第二音和第一音不相等且相差不超過大二度(pitch 相減小於 2)時，會判定第二音為鄰音，給予較低權重。

```
If ( n1 = n3 )
        if( abs(n2-n1) < 2 and n2! =n1 )
                n2->weight--;
```

n1,n2,n3 分別代表正在判斷的三個連續音

圖 30　鄰音判斷演算法

3.2.5 AMSA 簡化規則：音階

在裝飾音的分析中，並沒有一種叫音階的裝飾音，但其實這種手法在樂曲中又常常出現。在樂曲中如果出現需要跳躍較大音程時，通常會以音階的方式來連接這些音符，這樣的做法可增加音樂的豐富性及圓滑性，如圖 31 所示：

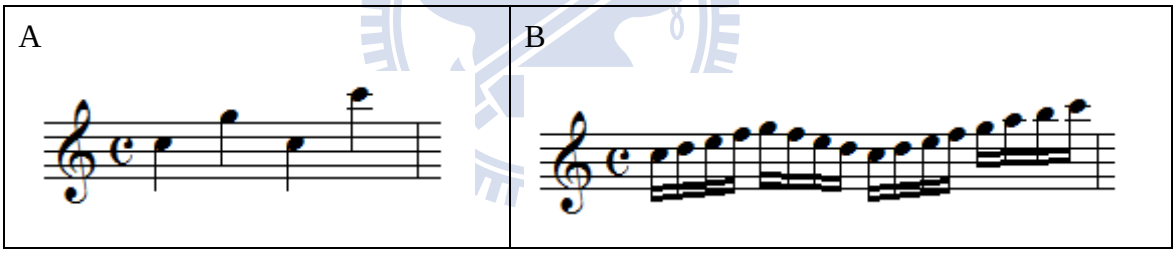


圖 31　利用音階裝飾的音樂

圖 31(A)為音樂的原始架構，圖 31 (B)則為利用音階裝飾後的音樂，可看出樂曲因為這樣而更加豐富。然而如果在一段音樂中發現類似圖 31 (B)中的型式，其實十六個音當中地位最重要的音符只有四個，分別是開頭、結尾以及中間的兩個轉折點，這些音在聽者的耳中也會留下較深的印象，所以我們在音樂簡化的判斷上可給予這四個音較高的權重，其餘的音較低的權重：

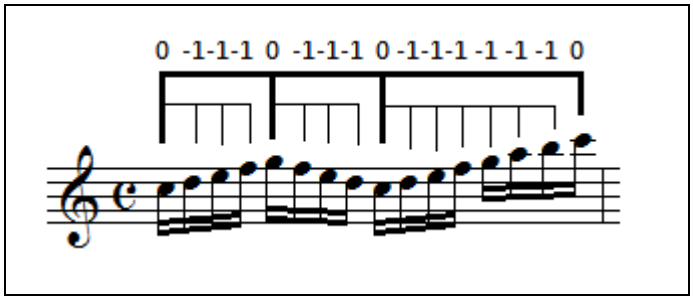


圖 32　音階權重示意圖

在簡化系統中判定音階的方式為，當一段音樂連續上升或下降超過四個音(如果只有三個音則屬於經過音)，且兩音的距離各不超過大二度，則判定為音階，並會給予起點及終點以外的音較低的權重，演算法如圖 33：

```
start = n1;
end = n2;
flag=1; //判斷上行或下行
if( n1-n2<0 )
        flag=-1;
while( n1! =n2 and 0<(n1-n2)*flag < 2 )
{
        end=n1=n2;
        n2=n2->next;
        count++;
}
if( count>=4 )
{
        temp=start->next;
        while( temp! =end )
                temp->weight--;
}
```

圖 33　音階判斷演算法

3.2.6 AMSA 簡化規則：節奏

節奏(Rhythm)對於音樂是很重要的一個因素，它讓音樂能更穩定更有組織。由音樂整體來看節奏這項參數或許可稱為速度，固定的速度讓音樂風格可更為突顯，也使音樂覺有組織，然而由細部來看節奏則由強弱拍組成，每種不同的節奏，如 4/4 拍、6/8 拍等，都有其對應的強弱拍。表 10 簡單以 4/4 拍來舉例說明：

表 10　4/4 拍強弱拍對應表

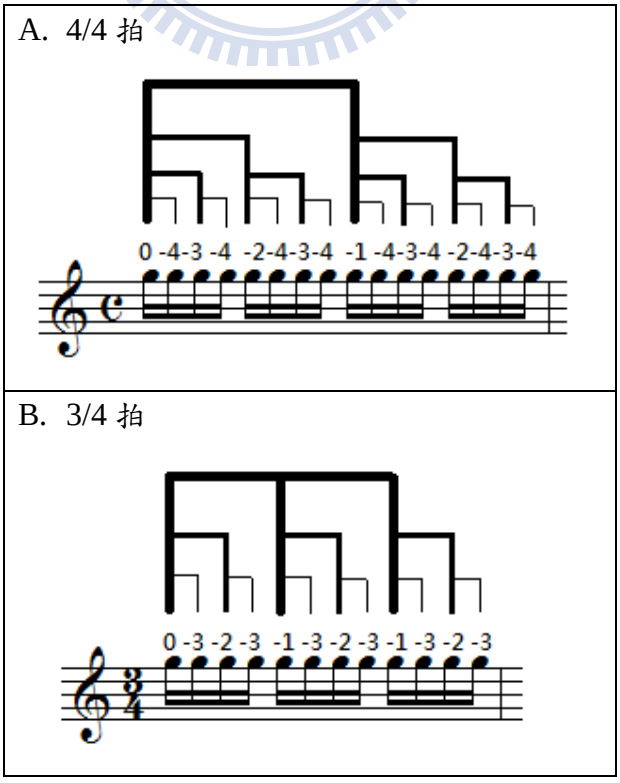
第一拍	第二拍	第三拍	第四拍
強拍	弱拍	次強拍	弱拍

如表 10 所示，一個 4/4 拍音樂就會一直重複強弱拍的循環。雖然強弱拍不一定要從頭到尾固定，但當大部份的時候以這種方式來演奏，可增加樂曲的韻律感，增加音樂的組織性，所以強弱拍在音樂上對於建立節奏感是很重要的參數。

也因為有拍子有強弱之分，作曲家在作曲時通常會傾向在強拍放上重要的音或骨幹音，而弱拍上較傾向於放裝飾音或輔助性的聲音。我們因此想利用這個特性來增加另一項簡化上的參數。

而如果只以一拍為最小單位來對於節奏給予參數，對我們來說是不夠的，因此我們參考 Melody Generator 產生節奏的方式，發現它利用 Longuet-Higgins 和 C. S. Lee 的研究[18]，產生了每個節奏點對於聽者建立節奏感的重要性給予權重，這樣的方式可對於更細部的每個節奏點給予權重。圖 34 簡單列出幾個較常見的節奏型態，並且利用譜例來表示 Longuet-Higgins 和 C. S. Lee 的研究所給予每個位置的權重。

雖然這種權重的給法很仔細，我們也可以直接拿來利用，但經過研究後發現直接利用現有的權重存在一個問題，就是會讓節奏所佔的權重比例過高，先前的經過音、鄰音等判斷，每種判斷最低只會給予-1 的權重，但節奏以 4/4 拍為例，這種權重的給法會讓有些音直接有-4 的權重，這樣一來會造成權重的不平衡。因此經過討論後我們將節奏的權重給法簡化為兩種方式，分別為最常見的以四分音符為單位以及以八分音符為單為兩種。並且在權重的給予上也分為較少的階層，如圖 35。



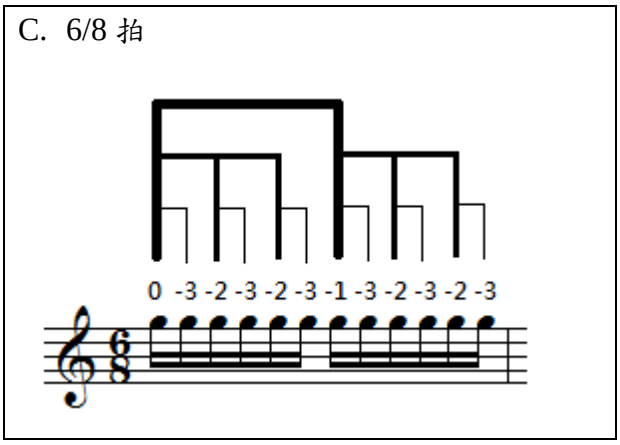


圖 34　節奏點對於建立節奏感的權重

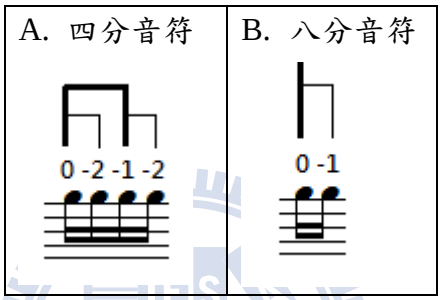


圖 35　AMSA 節奏權重示意圖

利用這種方式來代入音樂中給予權重，圖 36 可看出和原本給予權重方式的差異。

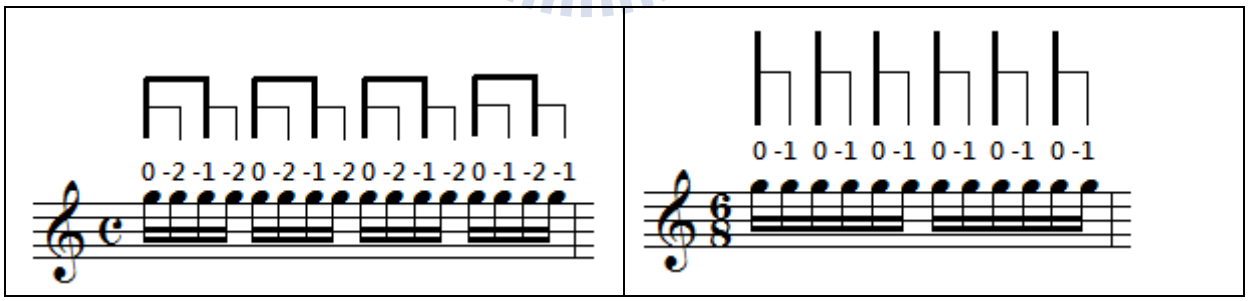


圖 36　節奏權重範例

在節奏權重的給予上，我們直接利用 MIDI 資訊中的 ONSET 時間來判斷某個音所發生的時間點，並且利用圖 36 的結果來給予權重。

3.2.7 AMSA 簡化規則：音長

音長(Duration)也是很重要的參數之一，在音樂中較長的音，或可說停留較久的音通常可使聆聽者聽起來會較有印象。另外，樂曲作曲的手法其實就是一連串的從穩定到不穩定再回到穩定這樣的變化，所以作曲家在作曲時經過一連串不穩定後會讓音樂回到穩定的狀態停留，而通常穩定的狀態就是指和諧的狀態，和諧的狀態通常指的是較為重要的和弦。所以依據這個原則，我們對每個音的長度也給予權重。

音常的權重做法是我們會找出音樂內所有的音長，並且依照排序先給予每種不同的音長一個權重，以圖 37 例子說明：

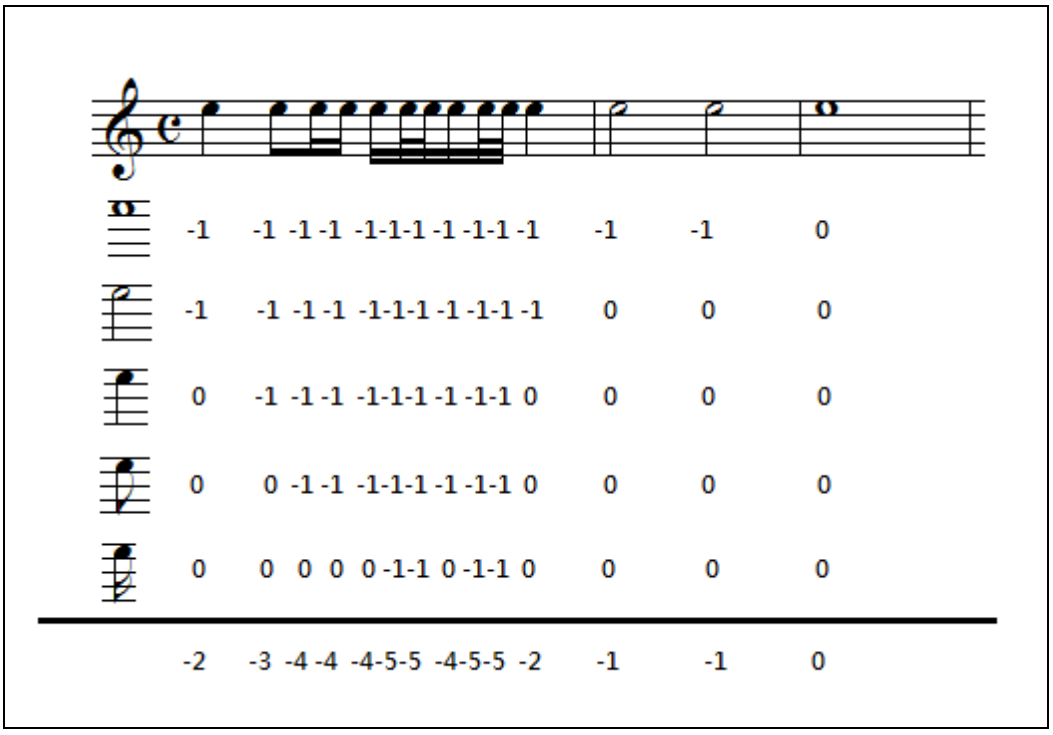


圖 37　音長權重範例

可看出圖 37 中的譜例，音長最長為全音符，最短則為三十二分音符，所以在這個例子當中全音符就會有最高的權重，依序遞減到三十二分音符，三十二分音符則是權重最低的。

然而這樣又會產生權重不平衡的問題，因此我們在找出每個音的權重後會再對這些權重做重新分配，並且以預設為三個階層(0, -1, -2)的方式來平均調整權重，如圖 38 所示：

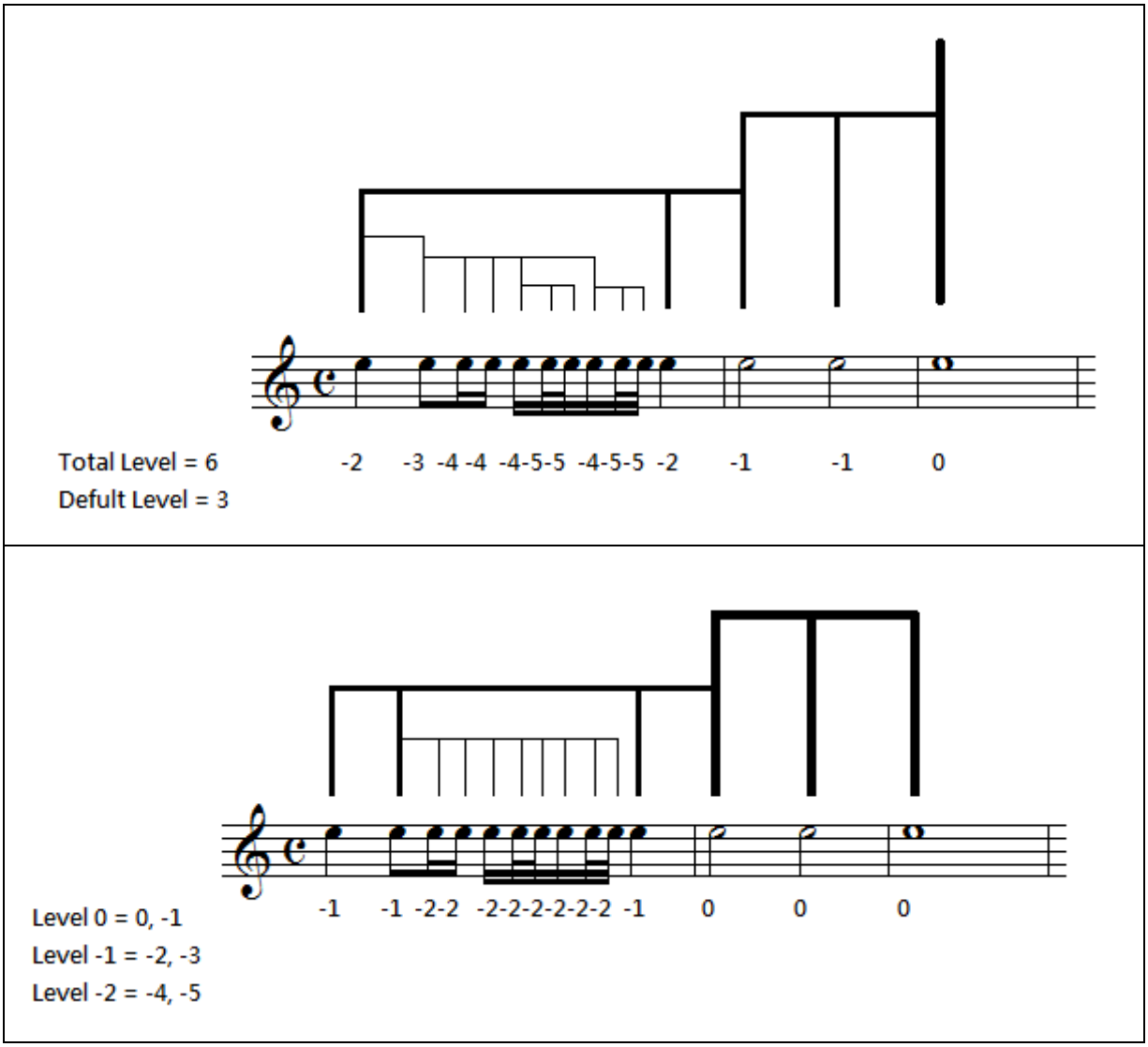


圖 38　音長權重範例 II

如此一來就可平衡權重，我們也依據這個方式給予每個音權重，這就是音長權重的給予方式。

3.2.8 AMSA：範例說明

以下簡單用一個例子說明 AMSA 簡化的過程，在這邊我們使用巴哈 a 小調小提琴協奏曲(BWV 1041)的其中一個片段來說明。

由於這個範本內所有的音符長度相等，因此不需要對長度這個項目給予權重，另外這個範例內也不存在重複音，而我們在後續的實驗中也不使用重複音判斷，所以也不需利用重複音的規則來判斷。

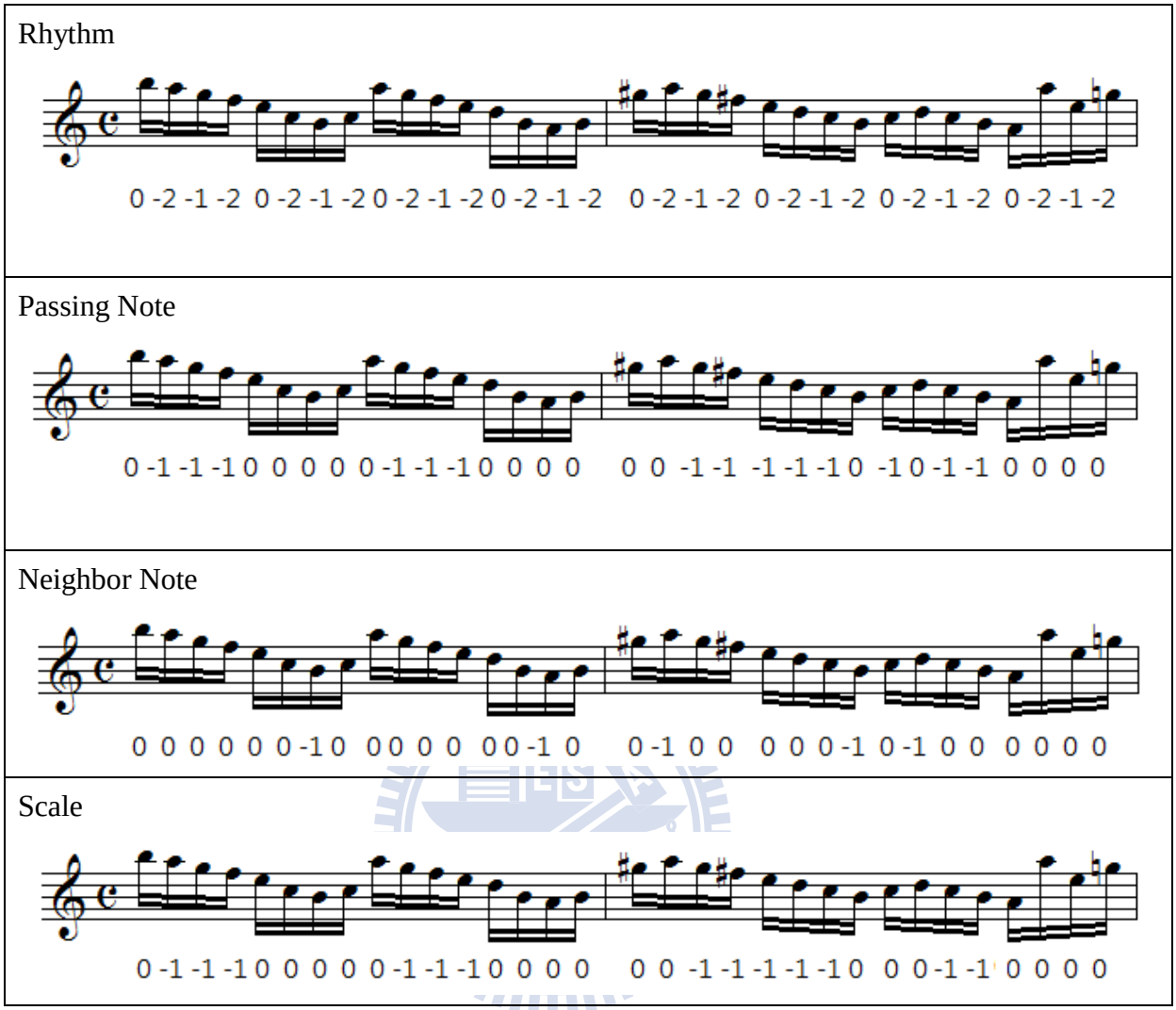


圖 39　AMSA 簡化過程範例

圖 39 可看出經過各種方式判斷後，每個音的權重變化，判斷完之後統計每個音的權重加總。

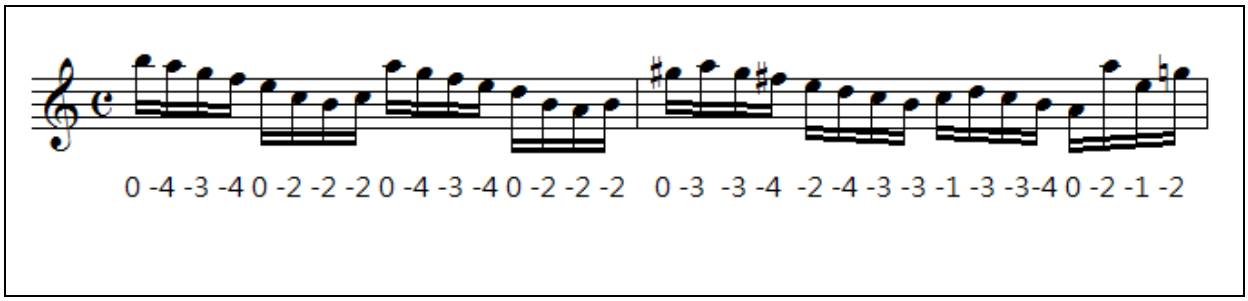


圖 40　AMSA 簡化範例權重結果

統計出每個音的總權重後，我們就可以設定各種不同的權重來觀察簡化結果，由於

圖 40 中最低的權重是-4，所以我們以下利用閥值為-4 到-1 來觀察結果。

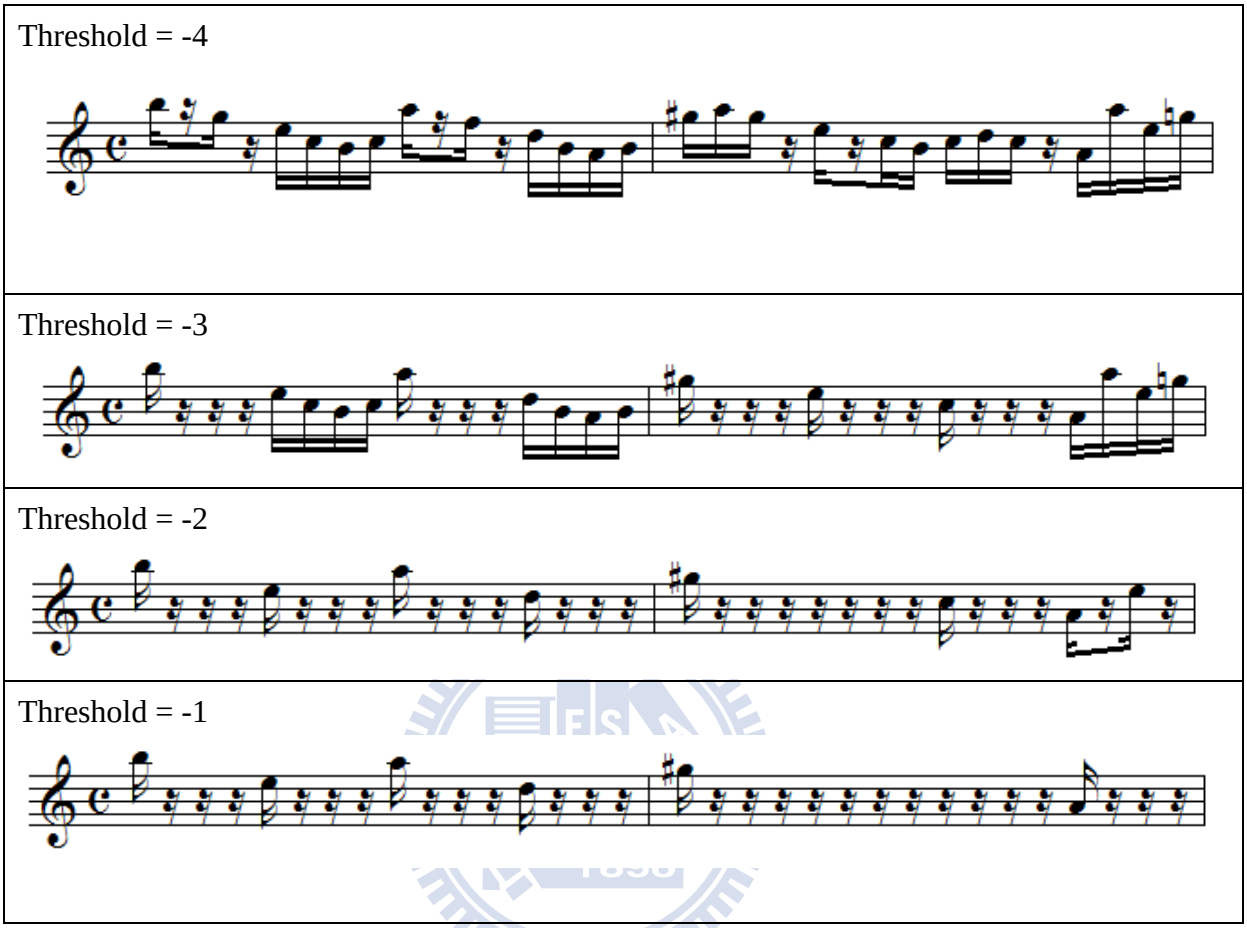


圖 41　AMSA 簡化範例各閥值結果

由上面過程可清楚看出簡化的步驟，由規則判斷權重，接著統計權重，最後設定閥值來做簡化。

觀察圖 41 的簡化結果，可發現閥值為-3, -2 時產生的結果在樂理上最為符合樂曲的架構。然而這就產生了另一個問題，如果要求系統的自動化，那麼勢必需要想一個設定閥值的方式，當然最直觀的方式就是設定固定的值，但在實驗過程中我們發現並沒有一個固定的值會讓每首輸入音樂出現最好的分析結果，因此我們提出了一個動態找尋閥值的方式，在下個小節內會詳細討論。

3.2.9 AMSA：Dynamic Threshold

由圖 41 的範例中我們可觀察每個閥值所刪除的音，-4 時刪去了 7 個音，-3 時多刪去了 8 個音，-2 時多刪去了 9 個音，-1 時多刪去了 2 個音。而其中閥值為-2 或-3 時的結果是最好的結果，也就是有保留住骨幹音，刪去裝飾音。

另外，在觀察我們其它的簡化實驗中也可發現一個有趣的特性。以下簡單以其中一個實驗歌曲來說明，我們使用的是 Czerny OP.599 No.95 這首曲子：

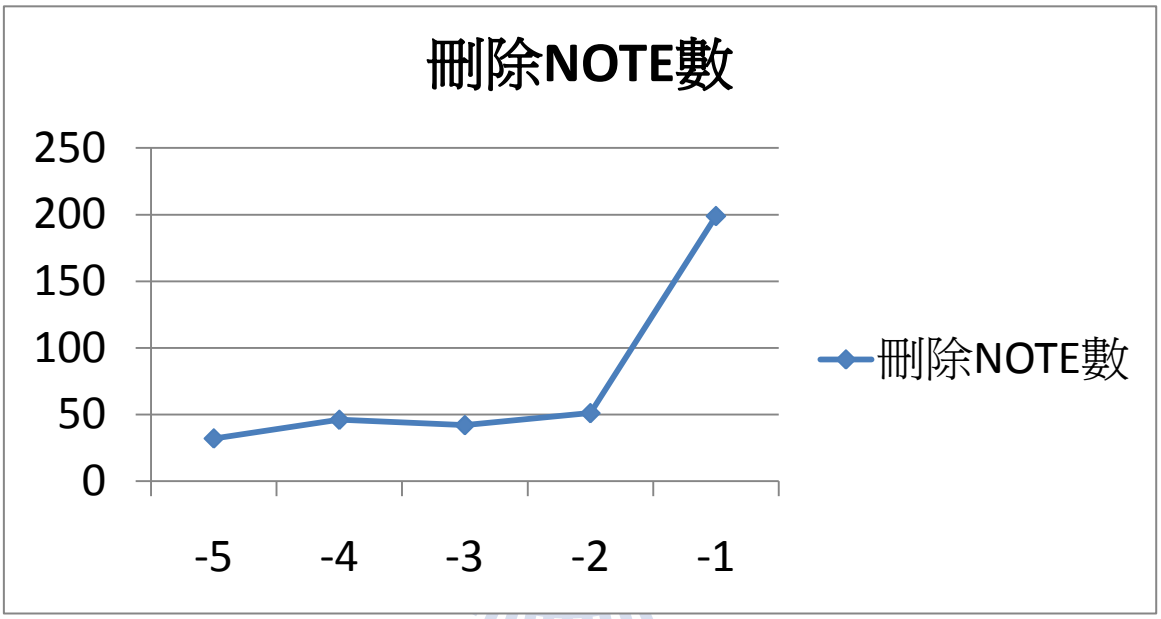


圖 42　Czerny OP.599 No.95 各閥值刪除音符數量折線圖

由圖 42 可看出刪除音符的數量在閥值為-1 時突然增加很多，而在我們所使用的輸入資料內大部份的資料都有這種特性，然而每筆資料遇到這種情況的閥值並不固定。

我們分析後發現，當達到突然刪掉很多音的閥值時，會把音樂內許多重要的分析依據給刪除，像低音部的伴奏部份可能對和聲和調性分析都很重要，但有可能在該閥值被一次全刪掉，整首曲子只下幾個音，這樣一來會造成分析上的資訊不足。因此可以得到：當在某個閥值出現大量音符被刪除的情況並不是刪掉裝飾性的音符，反而是把大部份樂曲重要的骨幹音都刪掉了，只剩下最簡單的音樂架構。這個結論。

我們利用這個特性把自動找尋閥值的方法定為，將閥值設定在刪去最多音的閥值的上一層，以此方式來完成電腦動態地找尋閥值，並可利用找到的閥值來進行簡化。

舉圖 41 的例子來說，閥值在-2 時刪去的音最多，因此我們自動判定閥值為-3，所以會得到-3 的結果。而圖 42 的範例則會自動判斷閥值為-2。簡單整理自動判斷閥值的演算法可利用下面幾個步驟來表示：

步驟一：計算出每個音的權重。
步驟二：找到整首輸入音樂最低的權重。
步驟三：計算每個權重所會刪去的音符數量
步驟四：找到會刪除最多音符的閥值 X。
步驟五：將閥值設定在(X-1)

在第三章的內容我們介紹了我們的系統所包含的分析項目及做法，而後半也介紹了我們的音樂簡化方式以及如何使電腦自動簡化音樂，接下來實驗結果部份我們將針對自動簡化對於調性和和聲分析的準確率來討論。

第四章、 實驗結果及討論

4.1 調性分析實驗結果

調性分析的實驗我們利用 KS Key Finding 以及 Spiral Array 兩種方式來比較。除了以兩種方式來測試原始輸入資料之外，也測試了各種不同的閥值所產生的結果。

測試資料一開始我們是使用許多鋼琴名曲，但發現測試後準確率偏低，仔細分析後認為是因為很多曲子雖然註明為某調性，但其實過程中不斷的轉調變化，也因為這樣造成分析上的誤差。如果遇到這樣的曲子，其實不應該單純說這首曲子的調性為何，反而應該利用和聲來分析，因為我們也有實做和聲的分析，因此在 Key Finding 的實驗中我們使用整首曲子調性明顯且較無轉調問題的曲子做為輸入檔。我們所使用的輸入檔案大部份為鋼琴曲，共 80 首[附錄一]。以下為實驗結果。

表 11　調性分析實驗結果

Threshold Value	KS	Spiral Array
Without Simplify	75%	87.5%
-5	77.5%	93.75%
-4	80%	90%
-3	80%	88.75%
-2	78.75%	91.25%

表 11 中閥值只有出現-5, -4, -3, -2 的原因在於，實驗中每首曲子的音，會得到最低的權重為-7，但並不是每首曲子都會出現那麼低的權重，在我們的測試資料中只有不到一半的曲子才會出現權重低於-5 的音符，且改善的幅度也很小，因此我們只比較-5 以上的權重。另外-1 不比較的原因在於如果閥值設定為-1，表示音符只要符合一個規則就會被刪去，這樣一來我們所做的權重平衡手法就失去意義，而實驗結果也發現閥值設為-1 準確性有可能比未經簡化還低，因此在這邊也不列出來討論。

4.2 和聲分析實驗結果

和聲分析部份，實驗我們所選擇的測試資料為 Tonal Harmony[19]和 Harmony and Voice Leading[17]這兩本和聲教科書中的 25 個範例[附錄二]，除了因為許多論文中的測試資料也出自於這本書之外，利用教科書上的範例也方便我們對答案，而雖然只使用 25 個範例，但我們自動分段的方式將這 25 個範例的音樂總共分了 244 個段落，可說是測試了 244 筆和聲分析的結果。

另外要說明的是如何統計和聲分析的實驗結果。由於我們和聲分析演算法是實作「Algorithms for Chordal Analysis」[12]內提出的方法，因此我們也利用同樣的方式進行統計，該論文所使用的計算方式如圖 43 所示。

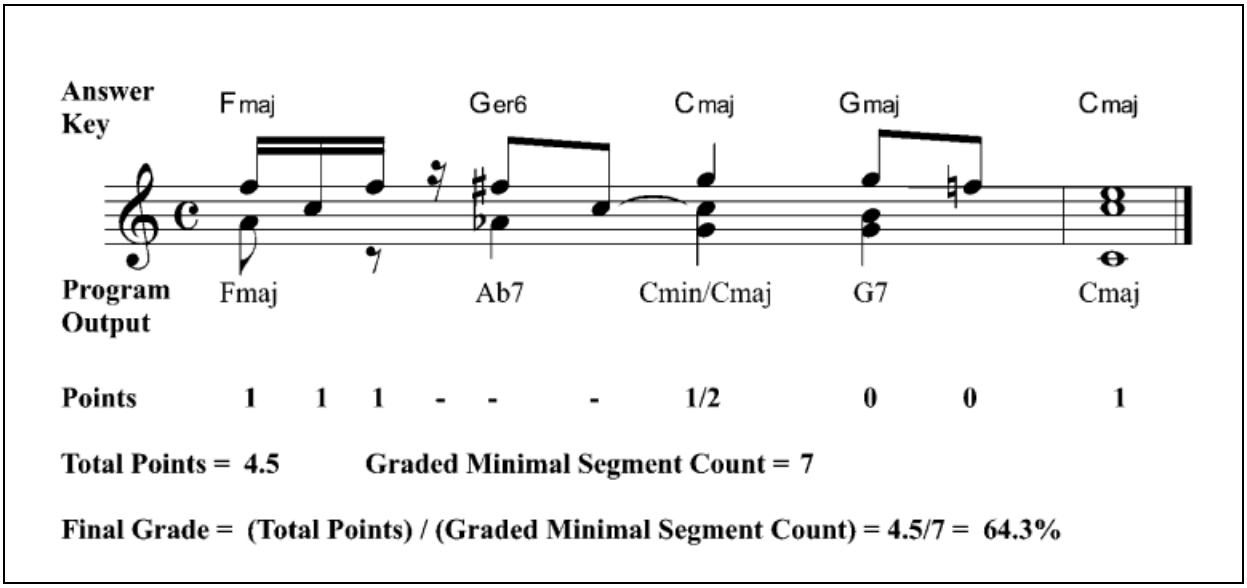


圖 43　和聲分析分數計算方式

在圖 43 當中，所輸入的音樂片段中使用了五個和弦，而程式輸出的結果卻把音樂分為十段。在「Algorithms for Chordal Analysis」[12]這篇論文中所使用的計算方式為，在圖中程式所輸出的前三點都是屬於同一個和弦，雖然被分為三段但其猜測的結果都是對的，因此視為三段都是對的，各得到 1 分。而接下來三個點因為答案所出現的和弦並沒有出現在程式中所比對的樣版內，因此不列入計算。接著在 Cmaj 這個和弦中，程式產生了兩種可能的答案，其中一個是對的另一個是錯的，因此得到 1/2 分。Gmaj 這個和弦被程式分為兩段，且都猜測錯誤，因此這兩段都是零分。最後 Cmaj 和弦只被分為一段且猜測正確，因此得到一分。

最後，本來分為十段，但其中三段因為標準答案沒有出現在比對樣版當中，所以不

列入計算，所以只剩下七段，而這個音樂片段的總分是利用分段數和得分相除，得到一個正確率的百分比，以這個百分比來判斷程式對於輸入音樂判別的準確率。

而下列實驗結果中的百分比則是把每個音樂片段的實驗結果加總並計算平均後所得到的平均準確率。

另外計算分數的方式上，我們和原本論文使用的方式稍有不同，如果答案出現了不在樣版中的和弦，我們會直接視為錯誤，而我們的程式也設計為每個分段只會有一個答案，因此不會有 1/2 這種分數出現。所以最後，未經簡化的輸入資料平均實驗結果，比本來論文中的結果稍微低了一點(76.5% → 70.1%)，除了因為輸入測試資料不相同外，也因為這些原因讓我們認為是合理的誤差。

以下為各閥值的實驗結果，其中最後一欄的 Best Answer 表示在 25 個輸入資料當中，如果一筆資料的最高分有出現在該閥值的結果當中，就列入計算，以沒有經過簡化的實驗結果來說，最後一欄的 8 代表著在 25 筆資料中，不經由簡化就可以有八筆資料所產生出來的結果是最好的。而以同一筆資料來說，最好的結果不一定會出現在單一閥值當中，有可能同時好幾個閥值的實驗結果都出現相同的準確率。Top 10 則代表在 25 比資料當中取出前十名來平均的結果。

表 12　和聲分析實驗結果

	All Data	Top 10	Best Answer
Without Simplify	70.05115%	89.88889%	8/25
-5	72.20988%	91.4798%	11/25
-4	74.07287%	93.81313%	14/25
-3	76.46825%	95.25%	14/25
-2	74.45714%	91.83333%	13/25

4.3 自動尋找閥值的實驗結果

在上個直章節中我們有提出自動找尋閥值的方式，因此以下我們也做了實驗比較。

調性分析實驗結果：

表 13　調性分析利用動態閥值方法實驗結果

	KS	Spiral Array
Best	80%　(t=-4,-3)	93.75% (t=-5)
Dynamic Threshold	81.25%	92.5%

表 13 內 Best 代表在固定閥值時得到的最佳結果，而 Dynamic Threshold 則是利用自動找尋閥值的方式所得到的結果。

和聲分析實驗結果：

表 14　和聲分析利用動態閥值方法實驗實驗結果

	All	Top10	Best Answer
Threshold = -3	76.46825%	95.25	14/25
Dynamic Threshold	77.05859%	93.18%	17/25

在和聲分析實驗中我們取出閥值為-3 來比較，因為當閥值為-3 時所得到的實驗結果是固定閥值裡最好的結果。

4.4 實驗結果討論

在表 11 中，我們可看出在 KS Key Finding 我們利用固定閥值的方式可增加約百分之五的準確率，而在 Spiral Array 的方法中可增加約百分之六的準確率。在表 12 的和聲分析當中，所有的範例，利用固定閥值的方式亦可增加約百分之六的準確率，

在調性分析中，其實原本的演算法都有考慮到裝飾音的問題，KS Key Finding 並不會因為某個調性的音階不存在某些音，就不把這些音列入考慮，而是還有一定比例存在。在 Spiral Array 的方法中只要影響調性的音所佔的比例不高，重心就不容易偏移太遠，因此在以前人所做的調性分析演算法中其實我們一開始不預期會改善太多，但從實驗結果中發現，簡化後還是能讓樂曲的結構更加明顯，也因為這樣可增加一定程度的準確率。

和聲分析在固定閥值時，平均狀況最好也可增加約百分之六的準確率。而會在比較結果時增加了一個比較項目，就是在所有測試資料當中得到最高準確率的數量，意義在於從整體來看簡化的效益，也可看出簡化對於輸入資料的整體，比起未經簡化的資料，的確是有較高的機會找出更好的答案。

另外，對於我們所提出的自動尋找閥值的方式，實驗結果在調性分析當中，利用 KS Key Finding 實驗結果發現可找出比固定閥值更多的正確答案，在 Spiral Array 當中雖然沒有比準確率最高的閥值要好，但也能達到接近的水準。在 Chord Finding 實驗當中，自動尋找閥值的方式不但整體的準確率較高，而且也能在所有輸入的資料中找出最多個最準確的選擇。因此證明了利用我們所提出的自動尋找閥值的辦法，是可以使用在自動分析上的。

然而在我們的實驗當中，發現簡化能達到效益其實是更高的，如果不論自動化與否，單純討論簡化是不是能增加分析的準確率，那麼就不需要考慮設定閥值的問題，只要在簡化的過程中有出現正確答案就列入計算的話，可統計出另一個準確率，如表 15 所示。

在「Algorithm of Chordal Analysis」[12]這篇論文中，作者列出了會造成實驗誤差的原因，如圖 44。

表 15　利用簡化所能達到最好的分析準確率

	Key Finding - KS	Key Finding – Spiral Array	Chord Finding
Without Simplify	75%	87.5%	70.05115%
Best Case of Static or Dynamic Threshold	81.25%	93.75%	77.05859%
Best Case without Considering Threshold Issue	90%	97.5%	83.5824%

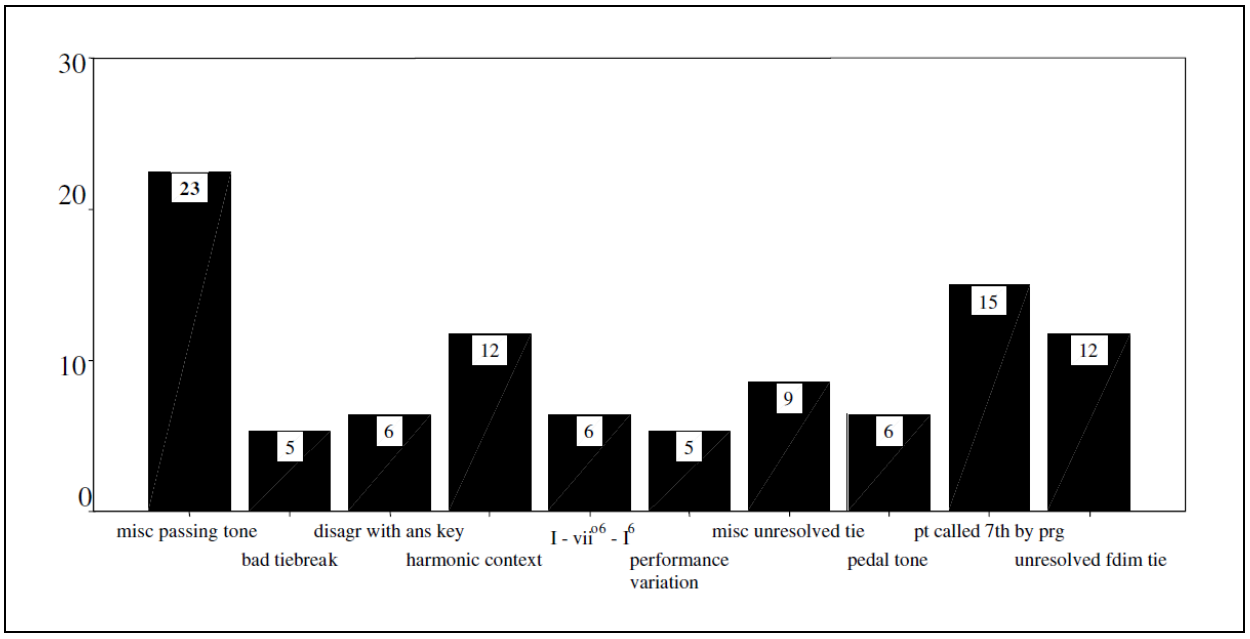


圖 44　Chord Finding 產生錯誤的十一個原因

資料來源：Algorithm of Chordal Analysis [12]

圖 44 中比例最高的兩項錯誤原因分別是把經過音誤認為和弦，佔了百分之二十三，而把經過音誤認為七和弦佔了百分之十五。這兩個項目也都是我們簡化後最想要改善的項目，也就是裝飾音所造成的失誤部份，可由此推算出裝飾音在錯誤的比例當中約佔有百分之三十八的比例。而經過我們簡化後，準確率可從原本的 70.05%增加到 83.28%，可換算出大約解決了百分之三十三的錯誤，證明了我們簡化的方式的確可以有效解決樂曲中音裝飾音所造成的和聲判斷誤差。

而在兩種調性分析演算法的實驗結果當中，表 15 也可發現簡化其實可以大大的增加分析的準確率。

雖然在固定閥值之下沒辦法達到簡化的最佳效益，利用我們所提出的自動尋找閥值的方式也沒辦法達到那麼好的效果，但我們還是相信經由簡化可以很有效的改善電腦自動分析音樂的準確性，也因此，未來我們希望透過更詳細的研究不同閥值的簡化結果和正確答案之間的關聯，找出更好的自動化尋找閥值辦法，或者增加新的簡化規則，使得這套簡化系統能更加完美，對於音樂自動分析的貢獻能更大。

第五章、 結論與未來展望

能讓電腦自動作曲一直是我們的夢想，而如果能讓電腦自動學習以前的人作曲上的精華之處，進一步利用所學的規則進行自動作曲那更是一套在我們心目中接近完美的自動作曲軟體。

然而音樂本來就不是因為電腦的發達才產生的產物，利用電腦分析音樂這樣跨領域的學問當中，很多人都還在努力摸索該如何更完美的結合這兩種領域。讓電腦自動分析音樂困難之處在於音樂的分析上很多東西的邏輯太複雜，且需要的應變能力和彈性都無法用很精確的數學或邏輯算式所描述。

David Cope 曾經利用他的自動作曲軟體，選出其中較為完美的音樂灌錄唱片，這在音樂自動分析和自動作曲上是很了不起的成就，目前幾乎無人能做得更好。然而 Cope 自己說過，自動產生出的音樂當中只有部份是「能聽的」，而剩餘的有些根本完全無法和音樂劃上等號，因此有人會質疑經過挑選的音樂能稱為自動作曲嗎？或者有些對音樂較為瞭解的人會認為他所產生的音樂只是很多音樂片段的重組而不是創作。

無論如何，我們還是希望朝向這個夢想前進。本篇論文希望能盡可能解決分析音樂上的問題，以利後人利用這個系統所產生的結果來進行自動作曲。而又因為我們不希望我們所產生出來的音樂也只是重組而不是創造，所以我們選則的方式為分析規則而不是記錄，希望先化繁為簡再進行分析。

然而由於時間限制，關於節奏、動機、樂曲架構等分析我們還未加入在這個系統當中，但簡化系統其實對於這幾個項目的分析有大的幫助，未來也希望能進一步利用簡化系統來一一加入樂曲架構上的分析，又或者在我們已實做的項目中以後有出現更好的演算法也能進行更改。

另外，分析上的準確率當然很重要，在研究過程中我們發現許多分析項目都還有可以改進的空間，例如本文中所使用的和聲分析演算法，在目前所存在的和聲分析演算法中算是很好的一種演算法，但如果在這個演算法當中加入和弦前後連接脈絡的判斷，或許可以提高更多分析的準確率，其它項目分析的演算法也一樣，都有我們可以改進及研究的地方。

透過本篇論文所提出的簡化演算法 AMSA 已可增加調性和和聲分析上的準確率。然而我們期望這個系統可以有更多的用處，或許可以把簡化利用在取得音樂特徵上，這樣一來或許對近年來很多人在研究的樂曲搜尋有很大的幫助。而如果利用簡化前和簡化後的音樂做“相減”的動作，或許也可取出樂曲裝飾音的部份，進而用來分析作曲家裝

飾的手法，成為另一項自動分析的音樂參數。如果撇開自動分析不談，在音樂分析的輔助上，使用者也可利用簡化來找出音樂架構，例如使用者可觀察在簡化到什麼程度後答案由錯誤轉為正確，進而找出在被分析的音樂中什麼音是會影響分析正確率的音符，也可以藉此看出哪些音是較為重要的骨幹音，而作者到底又是怎麼利用裝飾音來裝飾音樂等等。也可利用簡化系統在不同閥值的變化中一步步看出音樂架構如何化繁為簡。很多音樂上的應用都希望可以利用到我們所提出的簡化演算法，畢竟音樂本來就是透過由簡到繁的過程來產生，所以簡化對學習或分析音樂是很必要的過程。

然而簡化系統也不是完美，未來希望能透過進一步分析簡化結果和答案正確與否使得簡化系統更完美，或許可增加新的簡化規則，或許可找出更好的找尋閥值的方式使準確率更高，無論什麼方向都是我們努力的目標。

參考文獻

[1] Dirk-Jan Povel. “Melody Generator.” 26 May 2009. <http://www.socsci.kun.nl/~povel/Melody/index.html>.

[2] Maciej Biedrzycki. “cgMusic - Can computers create music?.” 2008. <http://codeminion.com/blogs/maciek/2008/05/cgmusic-computers-create-music/>.

[3] David Cope. The Algorithmic Composer. Middleton: A-R Editions, 2000.

[4] David Temperley. Music and Probability. Massachusetts: The MIT Press, 2007.

[5] David Temperley. The Cognition Musical Structures. Massachusetts: The MIT Press, 2001.

[6] Elaine Chew. “Toward a Mathematical Model of Tonality.” Ph. D. diss., Massachusetts Institute of Technology, 2000.

[7] David Miles Huber. The MIDI Manual. A Division of Macmillan Computer, 1991.

[8] Joseph N. Straus. Introduction to Post-Tonal Theory. New Jersey: Prentice Hall, 2004.

[9] Phil Winsor. Automated Music Composition. Denton: University of North Texas Press, 1992.

[10] Longuet-Higgins and M. J. Steedman. “On Interpreting Bach.” in Machine Intelligence, 6:221. Edinburgh University Press, 1971.

[11] Krumhans. Cognitive Foundations of Musical Pitch. Oxford: Oxford University Press, 1990.

[12] Bryan Pardo and William P. Birmingham. “Algorithms for Chordal Analysis.” Computer Music Journal 26:2 (July 2002): 27-49

[13] Tom Pankhurst. “Tom Pankhurst's Guide to Schenkerian Analysis.” 2009. <http://www.schenkerguide.com/>.

[14] Phillip B. Kirlin and Paul E. Utgoff. “A Framework for Automated Schenkerian Analysis,” in International Conference on Music Information Retrieval, Philadelphia, 2008.

[15] Alan Marsden. “Towards Schenkerian Analysis by Computer: A Reductional Matrix,” in Proceedings of International Computer Music Conference, 2005.

[16] Christy Keele and Todd McCready. “Automated Schenkerian Analysis.” 2006.

[17] 愛德華・奧德偉(Edward Aldwell)、卡爾・夏克(Carl Schachter)合著；張己任譯。和聲與聲部導進。台北市：小雅音樂有限公司，2003 年。

[18] H. C. Longuet-Higgins and C. S. Lee. “The rhythmic interpretation of monophonic Music Perception.” Music Perception 1:4 (Summer 1984): 424–441.

[19] S. Kostka and D. Payne. Tonal Harmony. New York: McGraw-Hill, 1984.

[20] 黃志方等著。「樂曲簡化演算法之研究」。數位內容國際學術研討會，桃園，民國 98 年

附錄一：調性分析實驗資料

Title	Composer	Key
Op.599 – 99	Czerny	B Major
Op.599 – 98	Czerny	C Major
Op.599 – 96	Czerny	D Major
Op.599 – 95	Czerny	A Major
Op.599 – 94	Czerny	E Major
Op.599 – 93	Czerny	Ab Major
Op.599 – 92	Czerny	C Major
Op.599 – 91	Czerny	G Major
Op.599 – 90	Czerny	G Major
Op.599 – 89	Czerny	G Major
Op.599 – 88	Czerny	Bb Major
Op.599 – 87	Czerny	F Major
Op.599 – 85	Czerny	D Major
Op.599 – 83	Czerny	Bb Major
Op.599 – 81	Czerny	C Major
Op.599 – 80	Czerny	C Major
Op.599 – 79	Czerny	Bb Major
Op.599 – 78	Czerny	F Major
Op.599 – 77	Czerny	A Major
Op.599 – 76	Czerny	G Major
Op.599 – 75	Czerny	Eb Major
Op.599 – 74	Czerny	C Major
Op.599 – 71	Czerny	F Major
Op.599 – 69	Czerny	D Major
Op.599 – 68	Czerny	C Major
Op.599 – 67	Czerny	F Major
Op.599 – 66	Czerny	A Major
Op.599 – 65	Czerny	D Major
Op.599 – 64	Czerny	Bb Major
Op.599 – 61	Czerny	C Major
Op.599 – 60	Czerny	F Major
Op.599 – 59	Czerny	G Major
Op.599 – 58	Czerny	C Major

Op.599 – 57	Czerny	C Major
Op.599 – 56	Czerny	F Major
Op.599 – 53	Czerny	Bb Major
Op.599 – 52	Czerny	D Major
Op.599 – 51	Czerny	D Major
Op.599 – 50	Czerny	D Major
Op.599 – 49	Czerny	Bb Major
Op.599 – 48	Czerny	Bb Major
Op.599 – 47	Czerny	F Major
Op.599 – 46	Czerny	C Major
Op.599 – 45	Czerny	G Major
Op.599 – 44	Czerny	C Major
Op.599 – 43	Czerny	C Major
Op.599 – 42	Czerny	F Major
Op.599 – 41	Czerny	G Major
Op.599 – 40	Czerny	F Major
Op.599 – 39	Czerny	G Major
Op.599 – 38	Czerny	C Major
Op.599 – 37	Czerny	C Major
Op.599 – 36	Czerny	C Major
Op.599 – 35	Czerny	C Major
Op.599 – 34	Czerny	C Major
Op.599 – 33	Czerny	C Major
Op.599 – 32	Czerny	C Major
Op.599 – 30	Czerny	C Major
Op.599 – 29	Czerny	C Major
Op.599 – 28	Czerny	C Major
Op.599 – 27	Czerny	C Major
Nocturne No.1 – Part 1	Chopin	Bb Minor
Nocturne No.1 – Part 2	Chopin	Db Major
Nocturne No.1 – Part 3	Chopin	Bb Minor
Nocturne No.2	Chopin	Eb Major
Nocturne No.4 – Part 1	Chopin	F Major
Nocturne No.4 – Part 2	Chopin	F Minor
Nocturne No.4 – Part 3	Chopin	F Major
Nocturne No.5	Chopin	F# Major
Nocturne No.8	Chopin	Db Major

Nocturne No.9	Chopin	B Major
Nocturne No.10 – Part 1	Chopin	Ab Major
Nocturne No.10 – Part 2	Chopin	F# Minor
Nocturne No.10 – Part 3	Chopin	Ab Major
Nocturne No.11 – Part 1	Chopin	G Minor
Nocturne No.11 – Part 2	Chopin	Eb Major
Nocturne No.11 – Part 3	Chopin	G Minor
Variation K. 265 – Part 1	Mozart	C Major
Variation K. 265 – Part 2	Mozart	C Minor
Variation K. 265 – Part 3	Mozart	C Major

附錄二：和聲分析實驗資料

Book	**Title**
Tonal Harmony	EX5-17 Mozart ,Sonata K.284,III
Tonal Harmony	EX7-17 Haydn, Sonata NO35,II
Tonal Harmony	EX7-18 Haydn, Sonata No.35,III
Tonal Harmony	EX8-1 Haydn, Sonata No.33 ,III
Tonal Harmony	EX8-18 Beathoven,Rondo,OP51,NO1
Tonal Harmony	EX11-5 Schumann, Scherzo Op.32
Tonal Harmony	EX12-3 Haydn, Sonata No.35,I
Tonal Harmony	EX13-17 Mozart, Sonata K.309.III
Tonal Harmony	EX7-19 Bach, Als Vierzig Tag nach Ostern
Tonal Harmony	EX13-13 Mozart,Sonata K570 III
Tonal Harmony	EX13-21 Beethoven, Sonata Op13,II
Tonal Harmony	EX 10-10 Haydn, Sonata No.15, II 25-28
Tonal Harmony	Self Test 10-1(2) Mozart, Sonata K. 284, II 16
Tonal Harmony	EX 11-2 P1
Tonal Harmony	EX 11-2 P2
Tonal Harmony	EX 11-3
Tonal Harmony	EX12-10 Schumann, Reaper's Song, OP.68 NO.18
Tonal Harmony	EX13-3(a)
Tonal Harmony	EX13-5 Schubert, Quartet, Op.post., I
Tonal Harmony	EX13-12 Bach, Sinfonia No.9
Tonal Harmony	EX11-1 Schubert, Fruhlingstraum, Op.89, No.11
Harmony & Voice Leading	EX7-12 韓德爾 G 小調雙簧管協奏曲第四樂章
Harmony & Voice Leading	EX9-13 Mozart,鋼琴奏鳴曲 K310 第一樂章
Harmony & Voice Leading	EX9-25 舒曼,大衛盟舞曲,OP6-5
Harmony & Voice Leading	EX12-6 舒伯特,即興曲,D.935