Sviluppo di un generatore di sequenze accordali in ambiente Max/ MSP e Javascript

di Paolo Gatti

Introduzione

Lo scopo del presente lavoro, è stato quello di arrivare ad una versione beta di un generatore di sequenze accordali, in linea con la teoria insiemistica dei "pitch class - sets" di A. Forte. Il tool in questione, è orientato all'esecuzione in tempo reale e può essere incorporato all' interno di configurazioni esecutive che prevedono l' utilizzo di due o più computers (nella fattispecie, ho adeguato l'algoritmo alla presenza di quattro computers sullo stage). Il lavoro è stato sviluppato in ambiente Javascript e MaxMSP, consentendo quest'ultimo, l'integrazione con il primo linguaggio mediante l'oggetto Max "js".

L'algoritmo

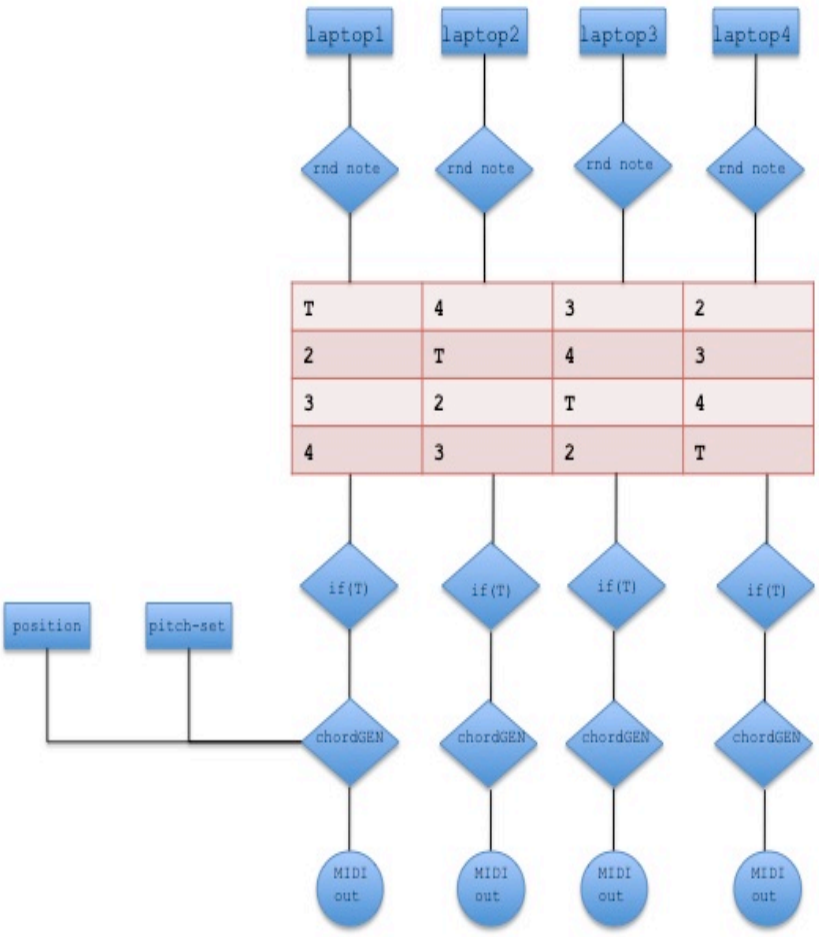

fig 1: diagramma a blocchi dell'algoritmo

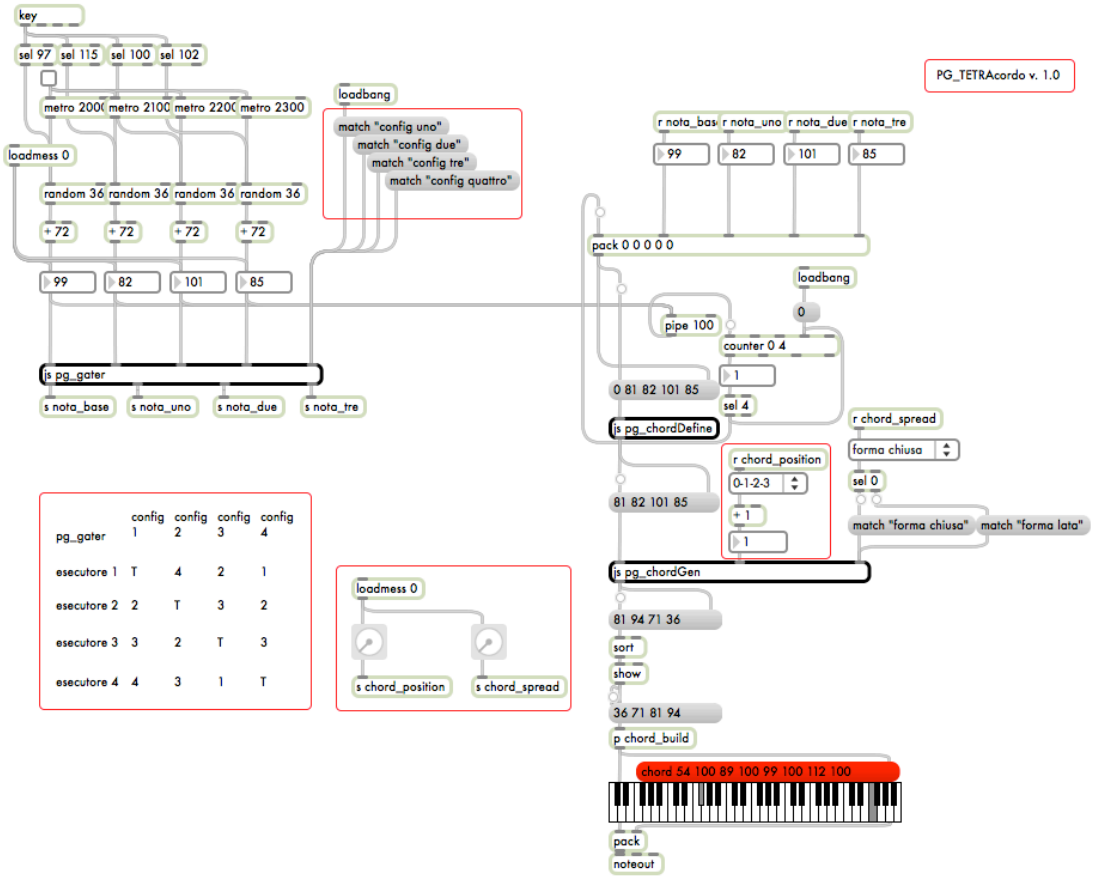

fig 2: patch che implementa l'algoritmo

Passiamo in rassegna i differenti "blocchi" dell' algoritmo. Vi sono quattro inputs corrispondenti ad una serie di quattro note MIDI (triggerate dai vari esecutori e ricevute dall'algoritmo mediante protocollo OSC, o semplicemente via cavo, qualora le connessioni avvengano per mezzo di piastre audio dedicate). L'algoritmo attende che tutti gli inputs siano riempiti e solo allora manda in uscita le note, in modo da avere i quattro valori da cui ottenere successivamente il tetracordo. In realtà, l'unica nota importante a tal proposito è la tonica ("T"), mentre le altre note servono puramente a riempire la lista e a triggerare l'uscita (che avviene proprio ogni quattro valori di input). Le quattro note MIDI, vengono inviate all' oggetto "js pg_gater", il cui listato è il seguente:

```
//pg_gater
//shifta i valori numerici per un array di quattro elementi
```

```
inlets=5;
outlets=4;

// array contenente i valori di nota MIDI in ingresso
var note=new Array();
note[0]=note[1]=note[2]=note[3]=0;

//variabile di controllo per lo shifting
var counter =0;

if (jsarguments.length>1)
        input[1] = jsarguments[1];

/*--------------------string matching--------------------*/
var vpattern = ".*";
var vmodifier = "i";
var vregexp = new RegExp(vpattern,vmodifier);

function pattern(p)
{
        vpattern = p;
        vregexp.compile(vpattern,vmodifier);
}

function modifier(m)
{
        vmodifier = m;
        vregexp.compile(vpattern,vmodifier);
}

function match(v)
{
var r = vregexp.exec(v);
   if (r=="config uno")
    {
      counter= 1;
      return counter;
    }
   if (r=="config due")
    {
      counter= 2;
      return counter;
    }
   if (r=="config tre")
    {
      counter= 3;
      return counter;
    }
   if (r=="config quattro")
    {
      counter= 4;
      return counter;
    }
}
/*-------------------------------------------------------*/

//gestione degli ingressi numerici
function msg_int(i)
{
        note[inlet] = i;
        //var m=0;
        //var s=0;
        if ((0<=inlet<=3))
```

```
   {
        uscita();
        }
}

/*--funzione di assegnamento degli ingressi alle uscite-------*/
function uscita()
{
        switch(counter)
        {
        case 1:
        outlet(0,note[0]);
        outlet(1,note[1]);
        outlet(2,note[2]);
        outlet(3,note[3]);
        break;
        case 2:
        outlet(0,note[3]);
        outlet(1,note[0]);
        outlet(2,note[1]);
        outlet(3,note[2]);
        break;
        case 3:
        outlet(0,note[2]);
        outlet(1,note[3]);
        outlet(2,note[0]);
        outlet(3,note[1]);
        break;
        case 4:
        outlet(0,note[1]);
        outlet(1,note[2]);
        outlet(2,note[3]);
        outlet(3,note[0]);
        break;
   }
}
/*-------------------------------------------------------------*/
```

Questa sequenza di istruzioni, consente a ogni esecutore (in base ad una funzione di matching relativa alle stringhe contenute nei message box), di poter inviare in uscita il tono fondamentale da cui verrà generato il tetracordo. A questo punto la lista di valori viene immagazzinata all'interno degli ingressi a polo freddo di un oggetto pack, che riceve il bang nel primo inlet (polo caldo) proprio quando i quattro valori all' istante i-esimo sono pervenuti. Si è resa necessaria quindi, l'implementazione di un oggetto javascript che troncasse la lista da cinque a quattro elementi (togliendo il primo 0, dovuto all'attivazione del polo caldo di pack).

```
//pg_chordDefine
//rimuove il primo valore di una lista

inlets=1;
```

```
outlets=1;

var alwaysoutput = 0;
var input=new Array();
input[0]=input[1]=input[2]=input[3]=0;

var lista;

if (jsarguments.length>1)
	input[1] = jsarguments[1];
if (jsarguments.length>2)
	alwaysoutput = jsarguments[2];

function list()//definizione dei termini dell'array rispetto all'input
{
	if (arguments.length>0)
		input[0] = arguments[0];
	if (arguments.length>1)
		input[1] = arguments[1];
		if (arguments.length>2)
		input[2] = arguments[2];
		if (arguments.length>3)
		input[3] = arguments[3];
		if (arguments.length>4)
		input[4] = arguments[4];

	 remove_first();
}

function remove_first()
{
lista= input.splice(1,4);//manda in output tutti gli elementi della lista meno il primo
out_result();
}

function out_result()
{
	outlet(0,lista);
}
```

A questo punto la lista di quattro elementi entra in ingresso all' oggetto chiave dell'algoritmo, ovvero pg_chordGen il cui listato porge:

```
//pg_chordGen
//genera accordi deducendo diverse configurazioni di pitch class, al variare
//del tono fondamentale

inlets=3;
outlets=1;

var input=new Array();
input[0]=input[1]=input[2]=input[3]=0;

//variabili di controllo
var counter=0;
var spread_fact=0;

//variabili per l'input matching
var vpattern = ".*";
var vmodifier = "i";
var vregexp = new RegExp(vpattern,vmodifier);
```

```
function pattern(p)
{
	vpattern = p;
	vregexp.compile(vpattern,vmodifier);
}

function modifier(m)
{
	vmodifier = m;
	vregexp.compile(vpattern,vmodifier);
}


if (jsarguments.length>1)
	input[1] = jsarguments[1];
if (jsarguments.length>2)
	input[2] = jsarguments[2];
if (jsarguments.length>3)
	input[3] = jsarguments[3];


function list()//definizione dei termini dell'array rispetto all'input
{
	if (arguments.length>0)
		input[0] = arguments[0];
	if (arguments.length>1)
		input[1] = arguments[1];
	if (arguments.length>2)
		input[2] = arguments[2];
	if (arguments.length>3)
		input[3] = arguments[3];
	if (arguments.length>4)
		input[4] = arguments[4];

		genera_primo_tetracordo(); //funzioni di generazione delle tipologie accordali
		genera_secondo_tetracordo();
		genera_terzo_tetracordo();
}

function msg_int(i)//gestione quadriadi (pitch class forms)
{
	input[inlet] = i;
	if ((inlet=1) )
	 {
	  switch(i)
		{
		 case 1:
		 counter=1;
		 break;
		 case 2:
		 counter=2;
		 break;
		 case 3:
		 counter=3;
		 break;
		}
	 }
}
//posizione accordale
function match(v)
{
var r = vregexp.exec(v);
  if (r=="forma chiusa")
```

```
   {
    spread_fact= 1;
    return spread_fact;
   }
   if (r=="forma lata")
   {
    spread_fact= 2;
    return spread_fact;
   }
}
//corpo delle funzioni di generazione
function genera_primo_tetracordo()
{
  if(counter==1)
          {
                  if(spread_fact==1)  //posizione accordale "stretta"
                   {
                   input[0]=input[0];
                   input[1]=input[0]+1;
                   input[2]=input[1]+1;
                   input[3]=input[2]+1;
                   }
                   if(spread_fact==2) //posizione lata
                   {
                   input[0]=input[0];
                   input[1]=input[0]+13;
                   input[2]=input[1] + 1 - 24;
                   input[3]=input[2]+1 - 36;
                   }
            outlet(0,(input));
          }
}

function genera_secondo_tetracordo()
{
  if(counter==2)
          {
                  if(spread_fact==1)  //posizione accordale "stretta"
                   {
                   input[0]=input[0];
                   input[1]=input[0]+1;
                   input[2]=input[1]+1;
                   input[3]=input[2]+2;
                   }
                   if(spread_fact==2) //posizione lata
                   {
                   input[0]=input[0];
                   input[1]=input[0]+13;
                   input[2]=input[1] + 1 - 24;
                   input[3]=input[2]+2 - 36;
                   }
            outlet(0,(input));
          }
}

function genera_terzo_tetracordo()
{
  if(counter==3)
          {
                  if(spread_fact==1)  //posizione accordale "stretta"
                   {
                   input[0]=input[0];
                   input[1]=input[0]+1;
                   input[2]=input[1]+2;
```

```
                    input[3]=input[2]+2;
                    }
                    if(spread_fact==2) //posizione lata
                    {
                    input[0]=input[0];
                    input[1]=input[0]+13;
                    input[2]=input[1] + 2 - 24;
                    input[3]=input[2]+2 - 36;
                    }
            outlet(0,(input));
          }
}
```

Il listato di cui sopra, consente la generazione di un tetracordo fra quelli proposti nel metodo di analisi insiemistica di Allen Forte, partendo dalla nota fondamentale "T", precedentemente impostata nei passi precedenti dell'algoritmo. Nella versione beta che propongo in questo articolo, è possibile scegliere fra tre sole tipologie di insiemi : 0-1-2-3, 0-1-2-4, 0-1-3-5; in aggiunta è possibile impostare due diverse "posizioni accordali": una posizione "stretta" dell'accordo, o una posizione "lata".

Sviluppi futuri

Questo generatore di sequenze accordali è uno strumento sicuramente utile nella creazione di movimenti armonici in tempo reale. Implementando tutte le principali combinazioni accordali in varie posizioni e migliorando l'automatizzazione dei parametri di controllo, sarà possibile ottenere uno strumento versatile, utile sia in momenti compositivi che in contesti esecutivo-improvvisativi.