# Minimum Concurrency for Assembling Computer Music

Carlos Eduardo Marciano
Presenter

Abilio Lucena

Felipe M. G. França

Luidi G. Simonetti

Systems and Computing Engineering Program
Federal University of Rio de Janeiro (UFRJ)

INOC 2019

cemarciano@poli.ufrj.br

# Roteiro

# Motivation

- The *Dining Philosophers:* proposed by *Edsger Dijkstra* in 1965 to illustrate *deadlocks, starvation* and race condition.
- Variant with two states: "eating" (consuming resources) or "hungry" (ready to eat).

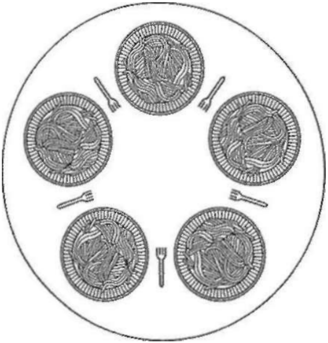


Figure 1:
The *Dining Philosophers* [1].

# Resource Graph

- Nodes represent processes to be scheduled.
- Edges represent shared resources between two nodes.
- <u>How to schedule nodes</u> in order to attain justice and prevent classic scheduling problems?

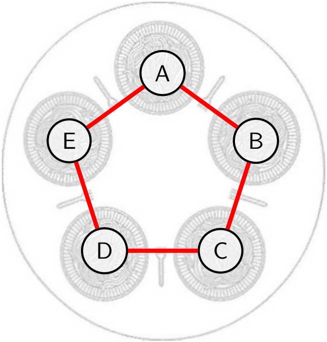



Figure 2: Resource Graph for the *Dining Philosophers*.

# Scheduling by Edge Reversal *(SER)*

- Distributed solution for heavily loaded neighborhood-constrained systems.
- Acyclic orientation: *sinks* operate simultaneously and revert their edges, forming new *sinks*.
- Justice: all nodes operate the same number of times within a period.

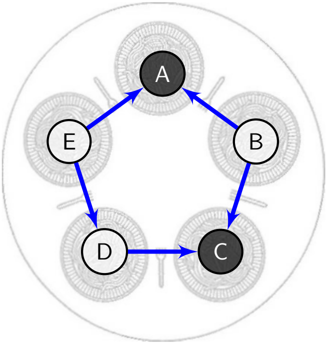



Figure 3: *DAG* representing the *Dining Philosophers*.

# *SER* Example

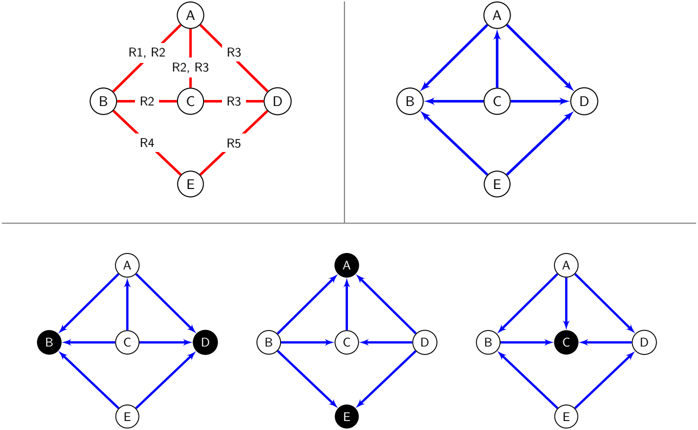
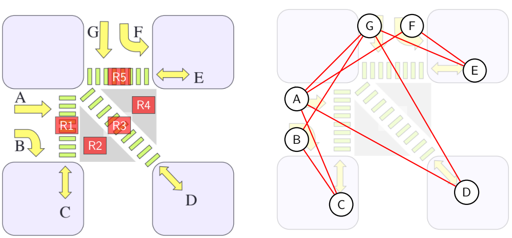
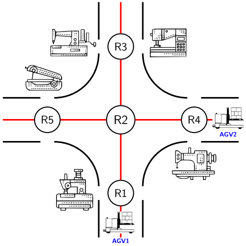



Figure 4: Oriented resource graph *(sup.)* and period induced by the algorithm *(inf.)*.

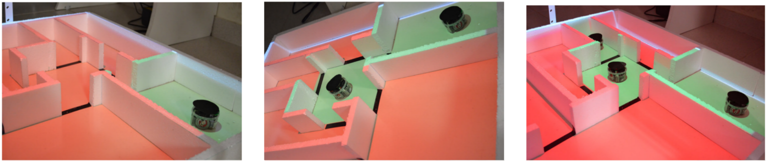
# Applications




(a) Road junctions [2].




(b) AGV Routing [3].



(c) Firefighting by autonomous robots [4].

Figure 5: *SER* applications.

# Definitions

## Definition: Simple Cycle

For $G = (V, E)$, a simple cycle $\kappa \subseteq V$ is <u>a subset of vertices</u> that form a sequence $i_0, i_1, ..., i_{|\kappa|-1}, i_0$. We define $K$ as the set of all simple cycles of $G$.

# Definitions

## Definition: Simple Cycle

For $G = (V, E)$, a simple cycle $\kappa \subseteq V$ is a subset of vertices that form a sequence $i_0, i_1, ..., i_{|\kappa|-1}, i_0$. We define $K$ as the set of all simple cycles of $G$.

## Definition: Acyclic Orientation

An acyclic orientation of $G$ is a function $\omega : E \rightarrow V$ such that no cycle $\kappa$ of the form $i_0, i_1, ..., i_{|\kappa|-1}, i_0$ exists for which $\omega(i_0, i_1) = i_1$, $\omega(i_1, i_2) = i_2$, ..., $\omega(i_{|\kappa|-1}, i_0) = i_0$. Let $\Omega$ be the set of all acyclic orientations of $G$.

# Definitions

**Definition: Simple Cycle**

For $G = (V, E)$, a simple cycle $\kappa \subseteq V$ is a subset of vertices that form a sequence $i_0, i_1, ..., i_{|\kappa|-1}, i_0$. We define $K$ as the set of all simple cycles of $G$.

**Definition: Acyclic Orientation**

An acyclic orientation of $G$ is a function $\omega : E \rightarrow V$ such that no cycle $\kappa$ of the form $i_0, i_1, ..., i_{|\kappa|-1}, i_0$ exists for which $\omega(i_0, i_1) = i_1$, $\omega(i_1, i_2) = i_2$, ..., $\omega(i_{|\kappa|-1}, i_0) = i_0$. Let $\Omega$ be the set of all acyclic orientations of $G$.

**Definition: Direction of Orientation**

We define $n_{cw}(\kappa, \omega)$ as the number of edges in the cycle $\kappa$ oriented by $\omega$ in the clockwise direction, and $n_{ccw}(\kappa, \omega)$ as the ones oriented counterclockwise.

# Concorrência

## Definição: Concorrência (1)

Seja $m$ o número de vezes que cada nó opera em um período do algoritmo *SER*. Seja $p$ o comprimento de um período, medido em orientações. Para $G = (V, E)$, definimos *concorrência* como uma função $\gamma : \Omega \rightarrow \mathbb{R}$ tal que:

$$\gamma(\omega) = \frac{m}{p} \tag{1}$$

# Concorrência

## Definição: Concorrência (1)

Seja $m$ o número de vezes que cada nó opera em um período do algoritmo *SER*. Seja $p$ o comprimento de um período, medido em orientações. Para $G = (V, E)$, definimos *concorrência* como uma função $\gamma : \Omega \to \mathbb{R}$ tal que:

$$\gamma(\omega) = \frac{m}{p} \tag{1}$$

## Definição: Concorrência (2)

Alternativamente, para $G = (V, E)$, definimos *concorrência* como:

$$\gamma(\omega) = \min_{\kappa \in K} \left\{ \frac{\min\{n_{cw}(\kappa, \omega), n_{ccw}(\kappa, \omega)\}}{|\kappa|} \right\} \tag{2}$$

# Exemplo *SER* (reprise)

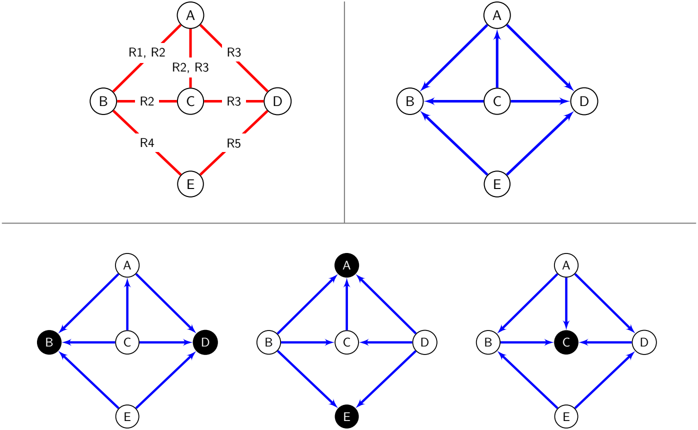



Figure 6: Concorrência: $\gamma(\omega) = m/p$; ou $\gamma(\omega) = \min_{\kappa \in K} \left\{ \frac{\min\{n_{cw}(\kappa,\omega), n_{ccw}(\kappa,\omega)\}}{|\kappa|} \right\}$.

# Roteiro

# Concorrência Mínima via Ciclos Máximos (1)

- NP-Completo [5]: Minimizar $\gamma(\omega)$ sobre todo o conjunto $\Omega$:

$$\gamma^* = \min_{\omega \in \Omega} \left\{ \min_{\kappa \in K} \left\{ \frac{\min\left\{n_{cw}(\kappa,\omega), n_{ccw}(\kappa,\omega)\right\}}{|\kappa|} \right\} \right\} \quad (3)$$

# Concorrência Mínima via Ciclos Máximos (1)

- NP-Completo [5]: Minimizar $\gamma(\omega)$ sobre todo o conjunto $\Omega$:

$$\gamma^* = \min_{\omega \in \Omega} \left\{ \min_{\kappa \in K} \left\{ \frac{\min \{ n_{cw}(\kappa, \omega), n_{ccw}(\kappa, \omega) \}}{|\kappa|} \right\} \right\} \tag{3}$$

**Lema 1**

$$\gamma^* = \min_{\kappa \in K} \left\{ \frac{1}{|\kappa|} \right\}$$

# Concorrência Mínima via Ciclos Máximos (1)

- NP-Completo [5]: Minimizar $\gamma(\omega)$ sobre todo o conjunto $\Omega$:

$$\gamma^* = \min_{\omega \in \Omega} \left\{ \min_{\kappa \in K} \left\{ \frac{\min\{n_{cw}(\kappa,\omega), n_{ccw}(\kappa,\omega)\}}{|\kappa|} \right\} \right\} \tag{3}$$

## Lema 1

$$\gamma^* = \min_{\kappa \in K} \left\{ \frac{1}{|\kappa|} \right\}$$

## Demonstração

Relembre a definição de concorrência: $\gamma(\omega) = \min_{\kappa \in K} \left\{ \frac{\min\{n_{cw}(\kappa,\omega), n_{ccw}(\kappa,\omega)\}}{|\kappa|} \right\}$. Para um dado $\omega'$, seja $\kappa'$ o ciclo escolhido pelo minimizador da definição de concorrência. Seja $x = \min\{n_{cw}(\kappa',\omega'), n_{ccw}(\kappa',\omega')\}$. Logo, temos $\gamma(\omega) = x/|\kappa'|$.

1/3

# Concorrência Mínima via Ciclos Máximos (1)

- NP-Completo [5]: Minimizar $\gamma(\omega)$ sobre todo o conjunto $\Omega$:

$$\gamma^* = \min_{\omega \in \Omega} \left\{ \min_{\kappa \in K} \left\{ \frac{\min \{n_{cw}(\kappa,\omega), n_{ccw}(\kappa,\omega)\}}{|\kappa|} \right\} \right\} \tag{3}$$

**Lema 1**

$$\gamma^* = \min_{\kappa \in K} \left\{ \frac{1}{|\kappa|} \right\}$$

**Demonstração**

Porém, para qualquer ciclo $\kappa \in K$, é possível orientar $\kappa$ com algum $\omega \in \Omega$ de forma que $n_{cw}(\kappa,\omega) = 1$ e $n_{ccw}(\kappa,\omega) = |\kappa| - 1$, ou vice-versa. Logo, se $\omega'$, aplicado a $\kappa'$, não produziu o valor $x = 1$, haverá outra orientação $\omega \in \Omega$ que produzirá $\gamma(\omega) = 1/|\kappa'|$. 2/3

# Concorrência Mínima via Ciclos Máximos (1)

- NP-Completo [5]: Minimizar $\gamma(\omega)$ sobre todo o conjunto $\Omega$:

$$\gamma^* = \min_{\omega \in \Omega} \left\{ \min_{\kappa \in K} \left\{ \frac{\min\left\{n_{cw}(\kappa, \omega), n_{ccw}(\kappa, \omega)\right\}}{|\kappa|} \right\} \right\} \quad (3)$$

**Lema 1**

$$\gamma^* = \min_{\kappa \in K} \left\{ \frac{1}{|\kappa|} \right\}$$

**Demonstração**

Suponha que $\gamma*$, a concorrência mínima de $G$, seja menor que $1/|\kappa'|$. Se isto for verdade, deverá existir um ciclo $\kappa^*$ que, sob alguma orientação $\omega*$, produzirá $1/|\kappa^*| < 1/|\kappa'|$. Logo, encontrar $\gamma^*$ tornou-se um problema de minimização sobre todo $\kappa \in K$. $\square$

# Concorrência Mínima via Ciclos Máximos (2)

- Resta encontrar $\omega^*$ tal que $\gamma^* = \gamma(\omega^*)$.

# Concorrência Mínima via Ciclos Máximos (2)

- Resta encontrar $\omega^*$ tal que $\gamma^* = \gamma(\omega^*)$.

**Teorema 1**

Dado qualquer ciclo máximo $\kappa^* \in K$ como entrada, existe um algoritmo de complexidade linear para encontrar uma orientação $\omega^* \in \Omega$ tal que $\gamma(\omega^*)$ é mínimo para todo $\omega \in \Omega$.

# Concorrência Mínima via Ciclos Máximos (2)

- Resta encontrar $\omega^*$ tal que $\gamma^* = \gamma(\omega^*)$.

**Teorema 1**

Dado qualquer ciclo máximo $\kappa^* \in K$ como entrada, existe um algoritmo de complexidade linear para encontrar uma orientação $\omega^* \in \Omega$ tal que $\gamma(\omega^*)$ é mínimo para todo $\omega \in \Omega$.

**Demonstração**

Pela prova do Lema 1, para atingir $\gamma^*$, deve-se orientar $\kappa^*$ tal que $n_{cw}(\kappa^*, \omega^*) = 1$ e $n_{ccw}(\kappa^*, \omega^*) = |\kappa^*| - 1$ (ou vice-versa). Isto pode ser realizado em tempo linear ao percorrermos o ciclo $\kappa^*$ e atribuirmos um número de identificação crescente $1, ..., |\kappa^*|$ para cada vértice visitado, resultando em uma ordenação topológica do ciclo. Por fim, orienta-se as arestas no sentido dos vértices de maior identificador, cumprindo o requisito. 1/2

# Concorrência Mínima via Ciclos Máximos (2)

- Resta encontrar $\omega^*$ tal que $\gamma^* = \gamma(\omega^*)$.

**Teorema 1**

Dado qualquer ciclo máximo $\kappa^* \in K$ como entrada, existe um algoritmo de complexidade linear para encontrar uma orientação $\omega^* \in \Omega$ tal que $\gamma(\omega^*)$ é mínimo para todo $\omega \in \Omega$.

**Demonstração**

Resta orientar os demais vértices de $G$ tal que $\omega^*$ sempre será de fato acíclica. Seja $S = V - \kappa^*$ o conjunto dos vértices restantes de $G$. Atribui-se um número de identificação crescente $|\kappa^*| + 1, ..., |V|$ para cada vértice em $S$, e então orienta-se todas as arestas de $G$ na direção dos vértices com maior identificador. Por absurdo, se $\omega^*$ possuir ciclos, existirá um caminho direcionado $i_0, i_1, ..., i_0$. No entanto, como $id[i_0] > id[i_1]$, é impossível retornar a $i_0$ após a partida, para qualquer $i_0 \in V$. Portanto, nenhum ciclo será formado. $\square$

# Experimental Results

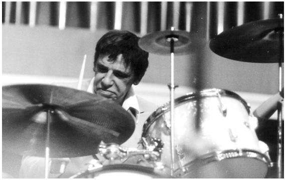
- *Simple Cycle Problem* model from Lucena et al. [6]:

| Nodes | p | Avg. Edges | Solved | Avg. Min. Conc. | CPU Time (s) |
|---|---|---|---|---|---|
| 200 | 0.01 | 391 | 10 | 1/178 | 0.6 (± 0.9) |
| 200 | 0.1 | 3 780 | 10 | 1/200 | 6.5 (± 7.3) |
| 1000 | 0.002 | 2 062 | 10 | 1/905 | 73.2 (± 51.4) |
| 1000 | 0.02 | 19 695 | 10 | 1/1000 | 797.0 (± 547.3) |
| 1000 | 0.2 | 179 806 | 3 | 1/1000 | 2 619.9 (± 1 015.0) |
| 2000 | 0.001 | 4 091 | 10 | 1/1805 | 425.9 (± 371.3) |
| 2000 | 0.01 | 39 807 | 3 | 1/2000 | 2 107.9 (± 1 561.5) |
| 2000 | 0.1 | 380 199 | 0 | – | – |

Table 1: Experiments for finding minimum concurrency of random graphs G(n,p).

# Roteiro

# Musical Context



(a) Buddy Rich, jazz.

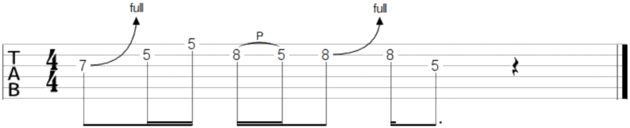
(b) Joe Bonamassa, blues.

Figure 7: Virtuosos *(Creative Commons)*.

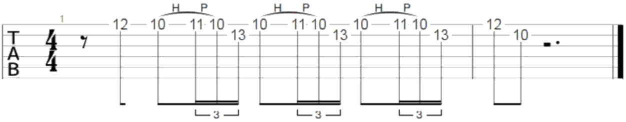
- Computer generation of melody has been studied since the early 1950's [7].
- Two approaches: explicit (in which composition rules are specified by humans) and implicit [8].
- Western music: features *counterpoint* (or *polyphony*), with multiple melodic voices [9].

# Musical Phrases

- In *blues*, *jazz* and *rock* music, it's common to exist a "question/answer" dynamic with musical phrases:




(a) Antecedent phrase.




(b) Consequent phrase.

Figure 8: Examples of music tablature [10].

# Assembling Maximum-length Tracks

- We'd like our model to capture the following restrictions:
  - A *consequent* phrase may only be played after an *antecedent* phrase, forming a *lick*;
  - Only phrases of the same type (*antecedent* or *consequent*) may be played simultaneously;
  - Phrases of different intensities (*e.g.* note counts) may not go well together;
  - The final composition must be a *loop*, include all phrases and be of maximum length.

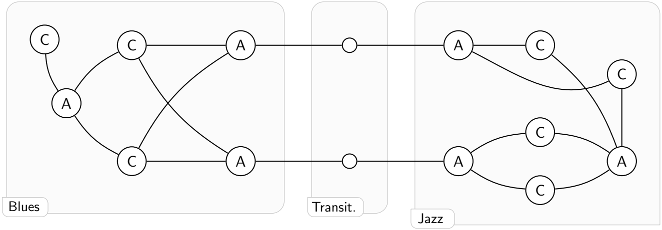
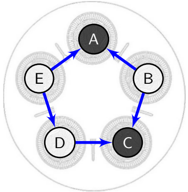


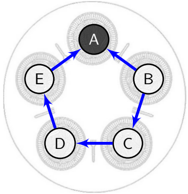

Figure 9: Modelling example.

# Conclusion

- Contributions: computational strategy for obtaining minimum concurrency and new approach for creating musical tracks.
- The *MIDI* standard: hour-long tracks and potential source of inspiration for artists.
- Future work: computational model for maximum concurrency under *SER*; investigate octave information for better-quality polyphony.




(a) Maximum concurrency.




(b) Minimum concurrency.

Figure 10: Extreme concurrencies.

# Agradecimentos

Obrigado!

Perguntas & Respostas

# Bibliografia I


[1] TANENBAUM, A. S., *Modern Operating Systems*.
3rd ed., pp. 143–165.
Upper Saddle River, NJ, USA: Pearson Prentice Hall, 2007.

[2] CARVALHO, D., PROTTI, F., DE GREGORIO, M., et al., "A Novel Distributed Scheduling Algorithm for Resource Sharing Under Near-Heavy Load", *Lecture Notes in Computer Science*, v. 3544, pp. 431–442, 2004.

[3] LENGERKE, O., ACUÑA, H. G., DUTRA, M. S., et al., "Distributed control of job-shop systems via edge reversal dynamics for automated guided vehicles", *1st International Conference on Intelligent Systems and Applications*, pp. 25–30, 2012.

[4] ALVES, D. S. F., SOARES, E. E., STRACHAN, G. C., et al., *A Swarm Robotics Approach to Decontamination. In: Mobile Ad Hoc Robots and Wireless Robotic Systems: Design and Implementation*.
1st ed., pp. 107–122.
Hershey, PA, USA: IGI Publishing Hershey, 2012.

[5] ARANTES JR, G. M., *Trilhas, Otimização de Concorrência e Inicialização Probabilística em Sistemas sob Reversão de Arestas*, Ph.D. Thesis, Prog. de Eng. de Sist. e Comp., Univ. Fed. do Rio de Janeiro, 2006.

[6] LUCENA, A., DA CUNHA, A. S., SIMONETTI, L., "A New Formulation and Computational Results for the Simple Cycle Problem", *Electronic Notes in Discrete Mathematics*, v. 44, no. 5, pp. 83–88, 2013.

# Bibliografia II

[7] NIERHAUS, G., *Algorithmic Composition: Paradigms of Automated Music Generation*. Springer-Verlag: Vienna, Austria, 2009.

[8] SHAN, M.-K., CHIU, S.-C., "Algorithmic compositions based on discovered musical patterns", *Multimedia Tools and Applications*, v. 46, n. 1, pp. 1–23, Jan. 2010.

[9] SCHMIDT-JONES, C., *Understanding Basic Music Theory*. OpenStax CNX: Houston, TX, USA, 2007.

[10] BELL, J., *144 Blues Guitar Licks*. JamString: East Midlands, UK, 2015, mobile application. Version 15.41942290.