Minimum Concurrency for Assembling Computer Music

Carlos E. 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

Roadmap

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).

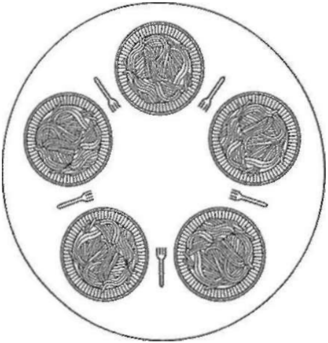


Figure 1:
The *Dining Philosophers* [1].

Resource Graph

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

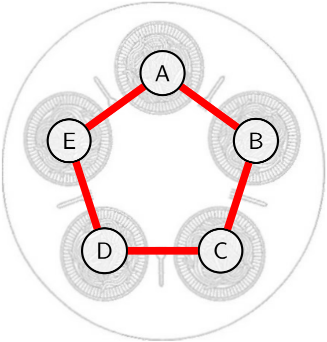


Figure 2: Resource Graph for the *Dining Philosophers*.

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.

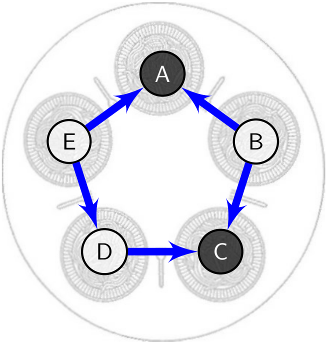
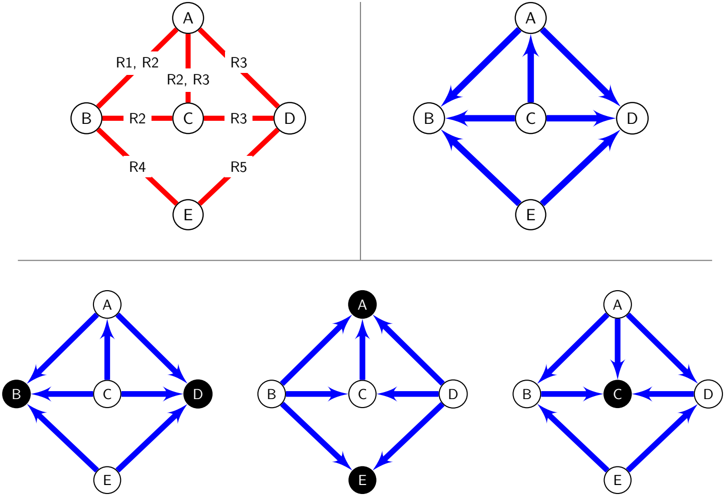
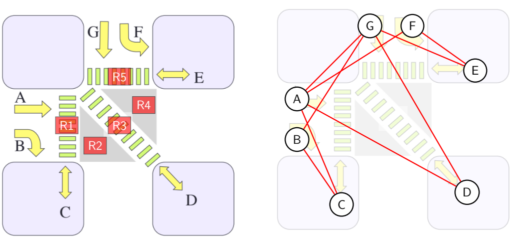


Figure 3: *DAG* representing the *Dining Philosophers*.

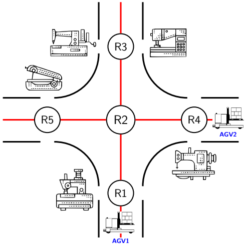
SER Example



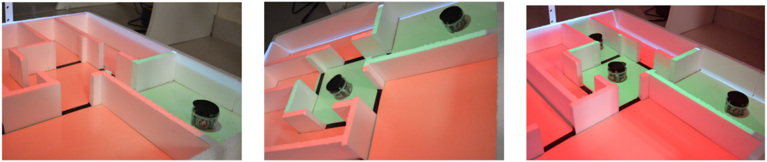
Applications



(d) Road junctions [2].



(e) AGV Routing [3].



(f) Firefighting by autonomous robots [4].

Figure 4: *SER* applications.

Definitions

Simple Cycle

…

$\kappa_3 = \{i_0, \ldots, i_{|\kappa_3 - 1|}, i_0\}$

$\kappa_1 = \{i_0, \ldots, i_{|\kappa_1 - 1|}, i_0\}$

$\kappa_2 = \{i_0, \ldots, i_{|\kappa_2 - 1|}, i_0\}$

…

K

Acyclic Orientation

…

$\omega_3 : E \to V$

$\omega_1 : E \to V$

$\omega_2 : E \to V$

…

Ω

Direction of Orientation

$n_{cw}(\kappa, \omega) = 3$

$n_{ccw}(\kappa, \omega) = 1$

Definitions

Simple Cycle

$\dots$

$\kappa_3 = \{i_0, \dots, i_{|\kappa_3 - 1|}, i_0\}$

$\kappa_1 = \{i_0, \dots, i_{|\kappa_1 - 1|}, i_0\}$

$\kappa_2 = \{i_0, \dots, i_{|\kappa_2 - 1|}, i_0\}$

$\dots$

K

Acyclic Orientation

$\dots$

$\omega_3 : E \rightarrow V$

$\omega_1 : E \rightarrow V$

$\omega_2 : E \rightarrow V$

$\dots$

Ω

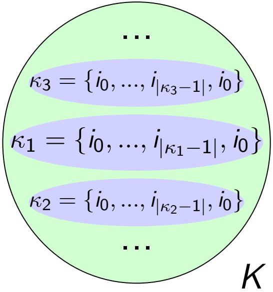
Direction of Orientation

$n_{cw}(\kappa, \omega) = 3$

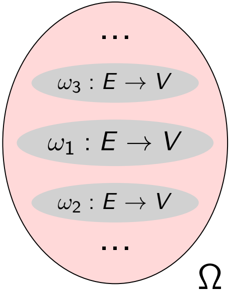
$n_{ccw}(\kappa, \omega) = 1$

Definitions

Simple Cycle



Acyclic Orientation

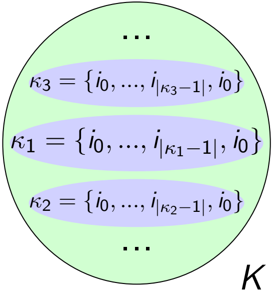


Direction of Orientation

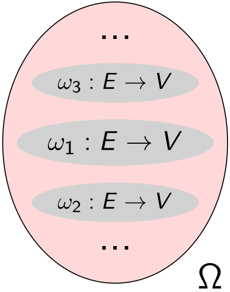
$$n_{cw}(\kappa, \omega) = 3$$
$$n_{ccw}(\kappa, \omega) = 1$$

Definitions

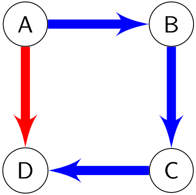
Simple Cycle



Acyclic Orientation



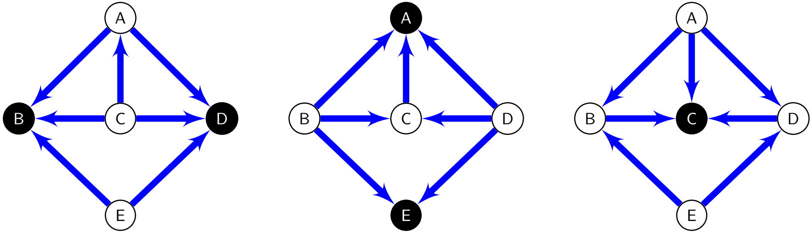
Direction of Orientation



$$n_{cw}(\kappa, \omega) = 3$$
$$n_{ccw}(\kappa, \omega) = 1$$

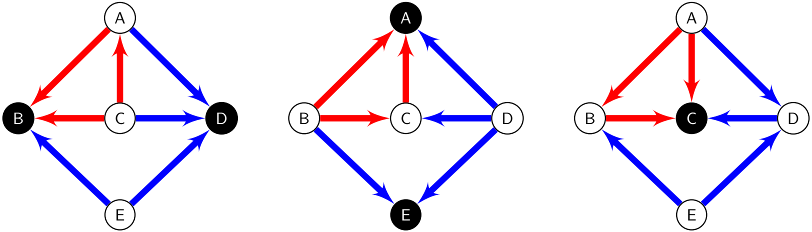
SER Concurrency ($\gamma : \Omega \rightarrow \mathbb{R}$), dynamic definition

$$\gamma(\omega) = \frac{\#\ of\ times\ each\ node\ operates}{period\ length}$$



SER Concurrency ($\gamma : \Omega \to \mathbb{R}$), static definition

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



Roadmap

Minimum Concurrency via Maximum Cycles

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

Minimum Concurrency via Maximum Cycles

- NP-Complete [5]: Minimize $\gamma(\omega)$ over all $\omega \in \Omega$:

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

Minimum Concurrency via Maximum Cycles

- NP-Complete [5]: Minimize $\gamma(\omega)$ over all $\omega \in \Omega$:

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

Minimum Concurrency via Maximum Cycles

- NP-Complete [5]: Minimize $\gamma(\omega)$ over all $\omega \in \Omega$:

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

Minimum Concurrency via Maximum Cycles

- NP-Complete [5]: Minimize $\gamma(\omega)$ over all $\omega \in \Omega$:

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

Lemma 1

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

Concorrência Mínima via Ciclos Máximos (2)

- We still need to find ω^* such that $\gamma(\omega^*) = \gamma^*$:

Algorithm 1: Obtaining an orientation in linear time that leads to minimum concurrency.

Input : Undirected graph $G = (V, E)$ and longest cycle $\kappa^* \subseteq V$

- Assign increasing ids to each vertex of κ^*
- Assign increasing ids (strictly greater than the ones in κ^*) to remaining vertices
- Create an “empty" orientation ω^*
- Orient edges towards the smaller (or larger) ids

return ω^*

Experimental Results

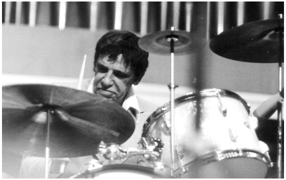
- *Simple Cycle Problem* model (Lucena 2013 [6]) with $G(n, p)$ graphs:

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	–	–

Using the *XPRESS Mixed Integer Programming* package v8.5.3 with all other features off (pre-processing, primal heuristics, etc). Intel Core i9-8950HK, 16 Gbytes of RAM, Linux Ubuntu 18.04.1, one thread.

Roadmap

Musical Context



(j) Buddy Rich, jazz.

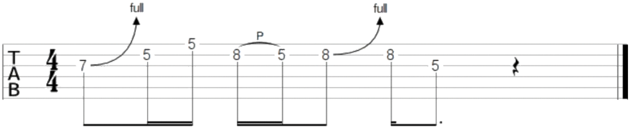
(k) Joe Bonamassa, blues.

Figure 5: Virtuosos *(Creative Commons)*.

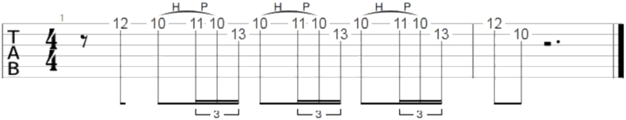
- Computer generation of melody has been studied since the early 1950's [7].
- Two approaches: explicit (in which composition rules are specified by humans) and implicit [8].
- Western music: features *counterpoint* (or *polyphony*), with multiple melodic voices [9].

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 6: 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.

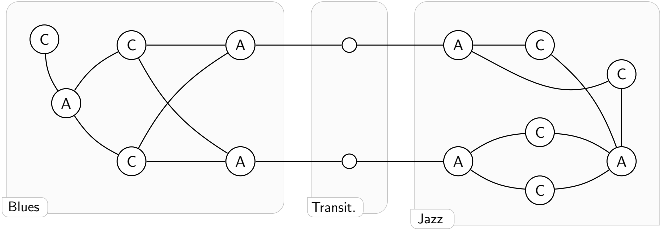
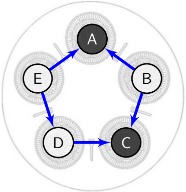


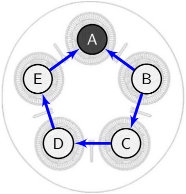
Figure 7: Modelling example.

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 8: Extreme concurrencies.

Closure

Thank you!

Questions & Answers

This presentation is available in PDF format at:
https://tinyurl.com/inoc2019-32

Bibliography 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.

Bibliography 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.

Appendix A: *Simple Cycle Problem* model [6]

$$\max \{\sum_{e\in E}(x_e + z_e) : (\mathbf{x}, \mathbf{y}, \mathbf{z}) \in \mathcal{R} \cap (\mathbb{B}^{|E|}, \mathbb{B}^{|V|}, \mathbb{R}_+^{|E|})\} \tag{1}$$

$$\sum_{e\in E(S)} x_e \leq \sum_{i\in S\setminus\{j\}} y_i, \quad \forall j \in S, \quad S \subset V \tag{2}$$

$$\sum_{e\in E} x_e \geq 2 \tag{3}$$

$$\sum_{e\in E} z_e = 1 \tag{4}$$

$$x_e + z_e \leq y_k, \quad \forall e = \{i, j\} \in E, \quad k = i \vee k = j \tag{5}$$

$$\sum_{e\in\delta(i)} (x_e + z_e) = 2y_i, \quad \forall i \in V \tag{6}$$

$$x_e, z_e \geq 0, \quad \forall e \in E \tag{7}$$

$$0 \leq y_i \leq 1, \forall i \in V. \tag{8}$$