O Problema do Milênio sobre Intratabilidade Computacional

Celina Miraglia Herrera de Figueiredo

COPPE
UFRJ

CNMAC 2019

Teoria da Computação

Ciência fundamental, assim como Biologia e Física

Por que alguns problemas são fáceis e outros difíceis?

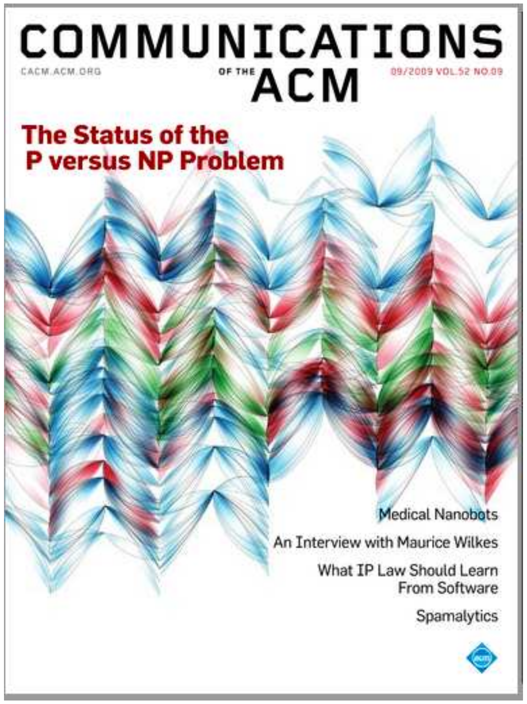
Não estuda quão rápido os computadores são
- Astronomia não é o estudo dos telescópios

Estuda a estrutura matemática dos problemas
- como a estrutura ajuda a resolver ou impede a tentativa de resolver

O Problema do Milênio

Problema central em Teoria da Computação: P versus NP

Existe pergunta cuja resposta pode ser verificada rapidamente, mas cuja resposta requer muito tempo para ser encontrada?



setembro 2009

RESOLVER ou VERIFICAR?

UMA PERGUNTA QUE VALE UM MILHÃO DE DÓLARES

O problema P versus NP: Resolver ou Verificar?

Em 1903, o matemático americano Frank Cole provou que o número

$$2^{67} - 1 = 147573952589676412927$$

não é primo exibindo a fatoração $193707721 \times 761838257287$.

É simples (embora tedioso se feito manualmente) calcular $2^{67} - 1$, calcular o produto $193707721 \times 761838257287$ e verificar que dão o mesmo número. Já encontrar essa fatoração é difícil. Cole disse que ele levou três anos trabalhando aos domingos.

Resolver ou Verificar? é uma pergunta que vale um milhão de dólares

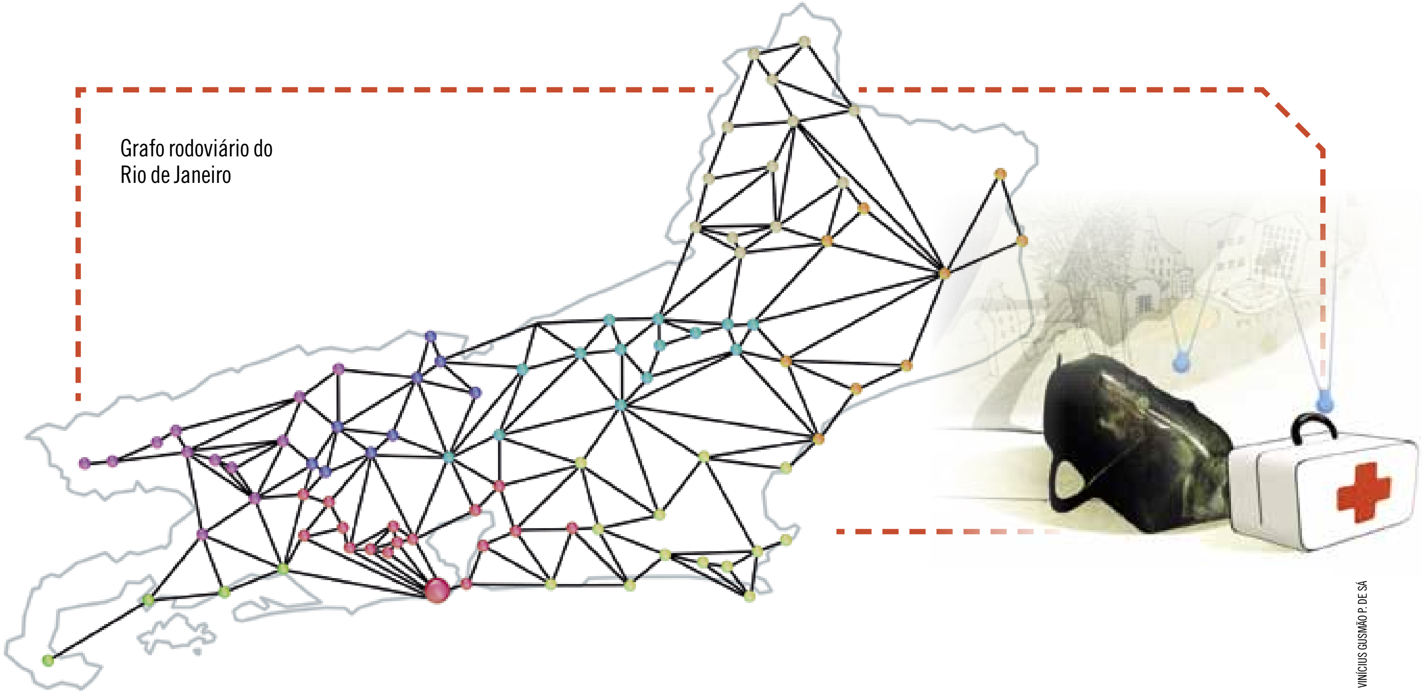
Clay Mathematics Institute Millennium problems

The Millennium Prize Problems

In order to celebrate mathematics in the new millennium, The Clay Mathematics Institute of Cambridge, Massachusetts (CMI) established seven *Prize Problems*. The Prizes were conceived to record some of the most difficult problems with which mathematicians were grappling at the turn of the second millennium; to elevate in the consciousness of the general public the fact that in mathematics, the frontier is still open and abounds in important unsolved problems; to emphasize the importance of working towards a solution of the deepest, most difficult problems; and to recognize achievement in mathematics of historical magnitude.

P vs NP Problem

If it is easy to check that a solution to a problem is correct, is it also easy to solve the problem? This is the essence of the P vs NP question. Typical of the NP problems is that of the Hamiltonian Path Problem: given N cities to visit (by car), how can one do this without visiting a city twice? If you give me a solution, I can easily check that it is correct. But I cannot so easily (given the methods I know) find a solution.


Grafo rodoviário do
Rio de Janeiro

Teoria dos Grafos

A matemática da conectividade, um dos ramos da matemática discreta

O artigo de Euler de 1736 é o nascimento da Teoria dos Grafos

- O circuito de Euler corresponde ao percurso do Pavimentador
- O ciclo de Hamilton corresponde ao percurso do Vacinador

Euler em seu artigo sobre as pontes de Konigsberg estuda como a dificuldade do problema cresce ou escala em função do número de pontes

Euler apresenta uma caracterização

- prova de que a cidade admite o circuito de Euler
- prova de que a cidade não admite o circuito de Euler

Complexidade computacional

A maioria dos problemas computacionais pertence à classe NP,
admitem um certificado polinomial

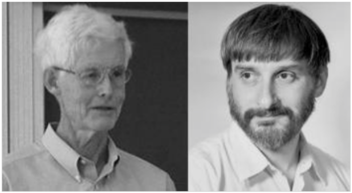
Em várias e diferentes áreas, procuramos objetos matemáticos:
percurso de um caixeiro viajante, atribuição de verdade,
emparelhamento máximo, coloração mínima de um grafo

O objeto matemático procurado é o certificado,
a prova de que o problema está em NP

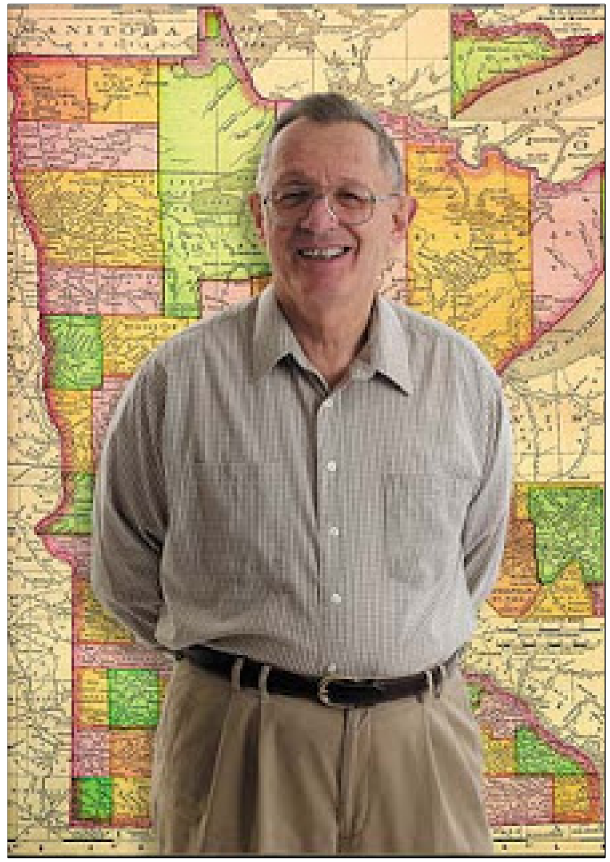
O estudo da complexidade computacional de problemas considera principalmente problemas em NP e tenta distinguir os solúveis em tempo polinomial dos não através da classe dos problemas NP-completos

C. Papadimitriou – *Computational Complexity* 1994

P vs NP Problem



Suppose that you are organizing housing accommodations for a group of four hundred university students. Space is limited and only one hundred of the students will receive places in the dormitory. To complicate matters, the Dean has provided you with a list of pairs of incompatible students, and requested that no pair from this list appear in your final choice. This is an example of what computer scientists call an NP-problem, since it is easy to check if a given choice of one hundred students proposed by a coworker is satisfactory (i.e., no pair taken from your coworker's list also appears on the list from the Dean's office), however the task of generating such a list from scratch seems to be so hard as to be completely impractical. Indeed, the total number of ways of choosing one hundred students from the four hundred applicants is greater than the number of atoms in the known universe! Thus no future civilization could ever hope to build a supercomputer capable of solving the problem by brute force; that is, by checking every possible combination of 100 students. However, this apparent difficulty may only reflect the lack of ingenuity of your programmer. In fact, one of the outstanding problems in computer science is determining whether questions exist whose answer can be quickly checked, but which require an impossibly long time to solve by any direct procedure. Problems like the one listed above certainly seem to be of this kind, but so far no one has managed to prove that any of them really are so hard as they appear, i.e., that there really is no feasible way to generate an answer with the help of a computer. Stephen Cook and Leonid Levin formulated the P (i.e., easy to find) versus NP (i.e., easy to check) problem independently in 1971.



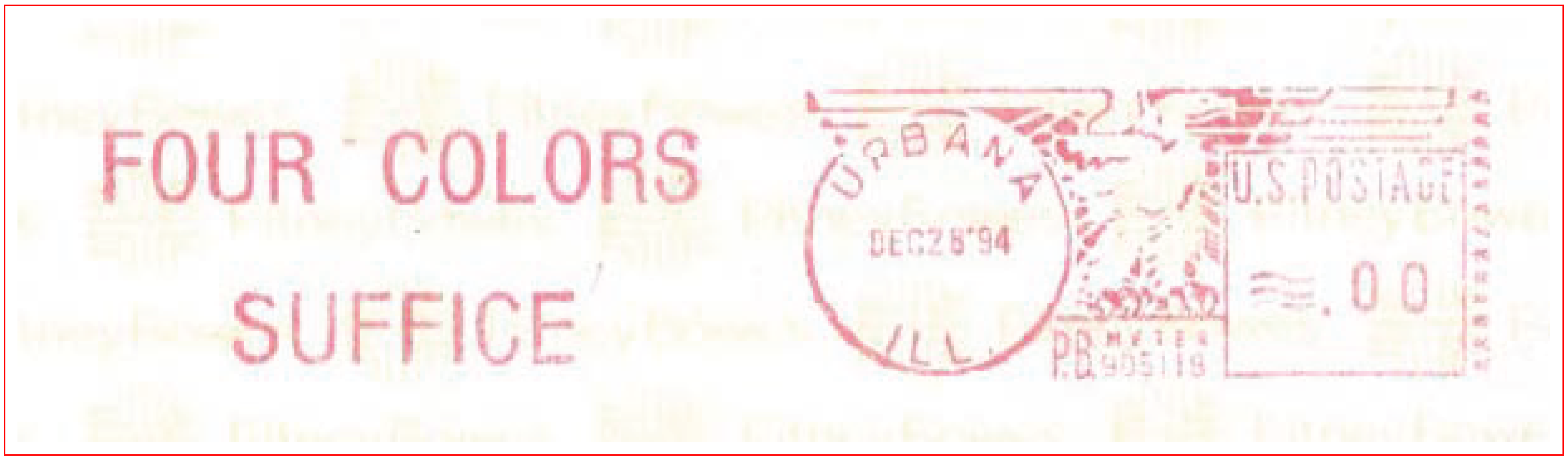
Kenneth Appel

1932 – 2013

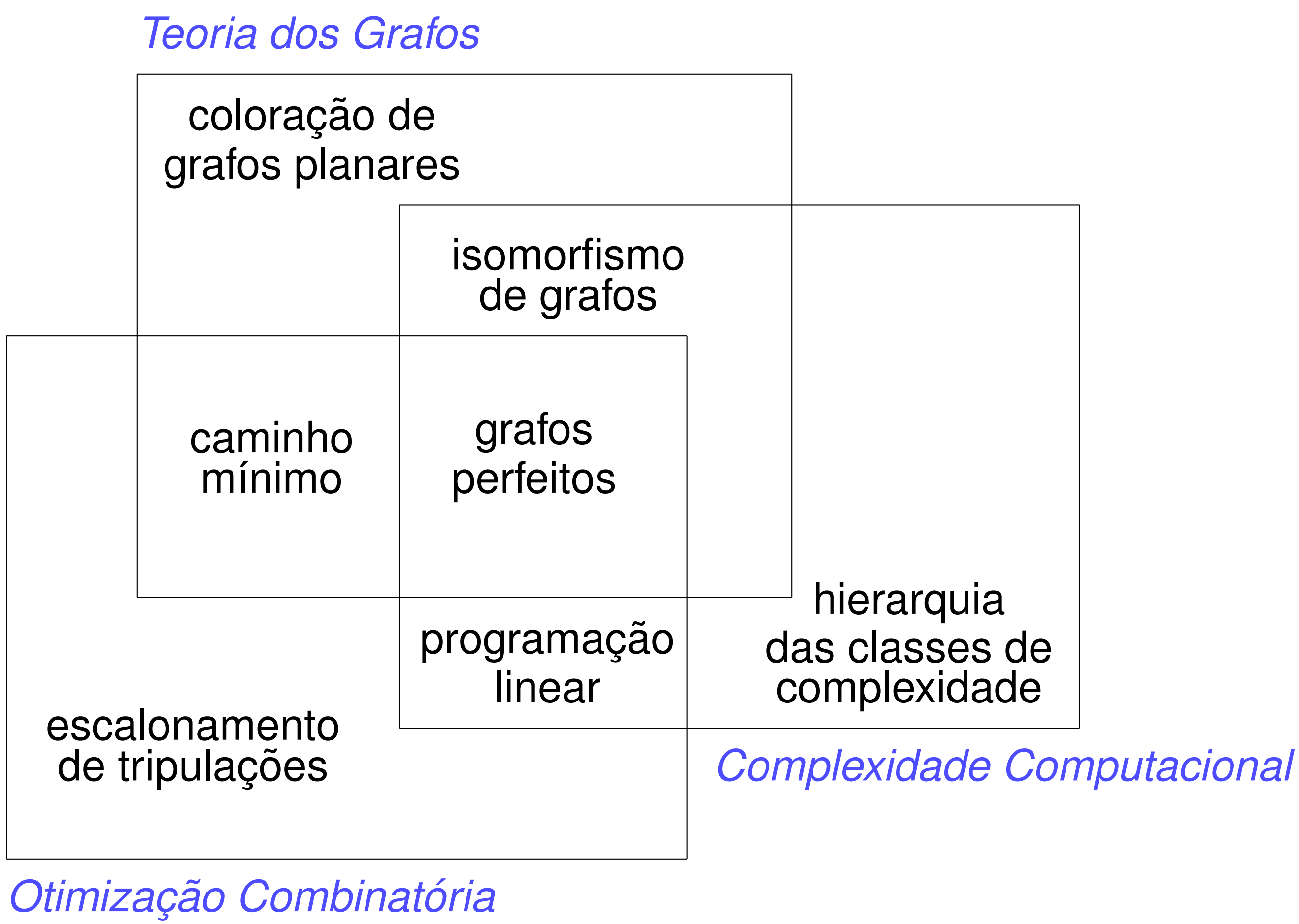
O Teorema das Quatro Cores

Appel e Haken (1977):

- Sim! (prova usando computador)
- conjunto inevitável de 1476 configurações redutíveis
- algoritmo colore grafo planar com quatro cores em tempo $O(n^4)$



Origem e desenvolvimento da área de pesquisa



Guia de NP-completude

Identificação de problema interessante, de classe de grafos interessante

Classificação da complexidade de um problema: P ou NP-completo

Problema que separa classes de grafos

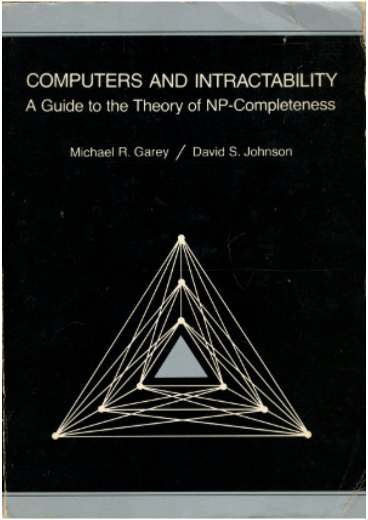
Classe de grafos que separa problemas

D. Johnson – *J. Algorithms* 1985, *ACM Trans. Algorithms* 2005

M. Golumbic, H. Kaplan, R. Shamir – *J. Algorithms* 1995

J. Spinrad – *Efficient Graph Representations* 2003

The Guide – Computers and Intractability



"Despite that 23 years have passed since its publication, I consider Garey and Johnson the single most important book on my office bookshelf. Every computer scientist should have this book on their shelves as well. NP-completeness is the single most important concept to come out of theoretical computer science and no book covers it as well as Garey and Johnson."

Lance Fortnow, "Great Books: Computers and Intractability: A Guide to the Theory of NP-Completeness"

Ongoing Guide – graph restrictions and their effect

Graph Class	Member	IndSet	Clique	CliPar	ChrNum	ChrInd	HamCir	DomSet	MaxCut	StTree	GraIso
Trees/Forests	P [T]	P [GJ]	P [T]	P [GJ]	P [T]	P [GJ]	P [T]	P [GJ]	P [GJ]	P [T]	P [GJ]
Almost Trees (k)	P	P [24]	P [T]	P?	P?	P?	P?	P [45]	P?	P?	P?
Partial k-Trees	P [2]	P [1]	P [T]	P?	P [1]	O?	P [3]	P [3]	P?	P?	O?
Bandwidth-k	P [68]	P [64]	P [T]	P?	P [64]	P?	P?	P [64]	P [64]	P?	P [58]
Degree-k	P [T]	N [GJ]	P [T]	N [GJ]	N [GJ]	N [49]	N [GJ]	N [GJ]	N [GJ]	N [GJ]	P [58]
Planar	P [GJ]	N [GJ]	P [T]	N [10]	N [GJ]	O	N [GJ]	N [GJ]	P [GJ]	N [35]	P [GJ]
Series Parallel	P [79]	P [75]	P [T]	P?	P [74]	P [74]	P [74]	P [54]	P [GJ]	P [82]	P [GJ]
Outerplanar	P	P [6]	P [T]	P [6]	P [67]	P [67]	P [T]	P [6]	P [GJ]	P [81]	P [GJ]
Halin	P	P [6]	P [T]	P [6]	P [74]	P [74]	P [T]	P [6]	P [GJ]	P?	P [GJ]
k-Outerplanar	P	P [6]	P [T]	P [6]	P [6]	O?	P [6]	P [6]	P [GJ]	P?	P [GJ]
Grid	P	P [GJ]	P [T]	P [GJ]	P [T]	P [GJ]	N [51]	N [55]	P [T]	N [35]	P [GJ]
$K_{3,3}$-Free	P [4]	N [GJ]	P [T]	N [10]	N [GJ]	O?	N [GJ]	N [GJ]	P [5]	N [GJ]	O?
Thickness-k	N [60]	N [GJ]	P [T]	N [10]	N [GJ]	N [49]	N [GJ]	N [GJ]	N [7]	N [GJ]	O?
Genus-k	P [34]	N [GJ]	P [T]	N [10]	N [GJ]	O?	N [GJ]	N [GJ]	O?	N [GJ]	P [61]
Perfect	O!	P [42]	P [42]	P [42]	P [42]	O?	N [1]	N [14]	O?	N [GJ]	I [GJ]
Chordal	P [76]	P [40]	P [40]	P [40]	P [40]	O?	N [22]	N [14]	O?	N [83]	I [GJ]
Split	P [40]	P [40]	P [40]	P [40]	P [40]	O?	N [22]	N [19]	O?	N [83]	I [15]
Strongly Chordal	P [31]	P [40]	P [40]	P [40]	P [40]	O?	O?	P [32]	O?	P [83]	O?
Comparability	P [40]	P [40]	P [40]	P [40]	P [40]	O?	N [1]	N [28]	O?	N [GJ]	I [GJ]
Bipartite	P [T]	P [GJ]	P [T]	P [GJ]	P [T]	P [GJ]	N [1]	N [28]	P [T]	N [GJ]	I [GJ]
Permutation	P [40]	P [40]	P [40]	P [40]	P [40]	O?	O	P [33]	O?	P [23]	P [21]
Cographs	P [T]	P [40]	P [40]	P [40]	P [40]	O?	P [25]	P [33]	O?	P [23]	P [25]
Undirected Path	P [39]	P [40]	P [40]	P [40]	P [40]	O?	O?	N [16]	O?	O?	I [GJ]
Directed Path	P [38]	P [40]	P [40]	P [40]	P [40]	O?	O?	P [16]	O?	P [83]	O?
Interval	P [17]	P [44]	P [44]	P [44]	P [44]	O?	P [53]	P [16]	O?	P [83]	P [57]
Circular Arc	P [78]	P [44]	P [50]	P [44]	N [36]	O?	O?	P [13]	O?	P [83]	O?
Circle	P [71]	P [GJ]	P [50]	O?	N [36]	O?	P [12]	O?	O?	P [70]	O?
Proper Circ. Arc	P [77]	P [44]	P [50]	P [44]	P [66]	O?	P [12]	P [13]	O?	P [83]	O?
Edge (or Line)	P [47]	P [GJ]	P [T]	N [GJ]	N [49]	O?	N [11]	N [GJ]	O?	N [70]	I [15]
Claw-Free	P [T]	P [63]	O?	N [GJ]	N [49]	O?	N [11]	N [GJ]	O?	N [70]	I [15]

THE COMPLEXITY OF HARD GRAPH PROBLEMS THIRTY YEARS LATER

CELINA MIRAGLIA HERRERA DE FIGUEIREDO

COPPE UFRJ

Graph Class	Member		IndSet		Clique		CliPar		ChrNum		ChrInd		HamCir		DomSet		MaxCut		StTree		GraphIso	
Trees/Forests	P	[T]	P	[GJ]	P	[T]	P	[GJ]	P	[T]	P	[GJ]	P	[T]	P	[GJ]	P	[GJ]	P	[T]	P	[GJ]
Almost Trees (k)	P		P	[OG]	P	[T]	P?		P?		P?		P?		P	[OG]	P?		P?		P?	
Partial k-trees	P	[OG]	P	[OG]	P	[T]	**P**	**[S]**	P	[OG]	**P**	**[S]**	P	[OG]	P	[OG]	**P**	**[S]**	**P**	**[S]**	**P**	**[S]**
Bandwidth-k	P	[OG]	P	[OG]	P	[T]	P?		P	[OG]	P?		P?		P	[OG]	P	[OG]	P?		P	[OG]
Degree-k	P	[T]	N	[GJ]	P	[T]	N	[GJ]	N	[GJ]	N	[OG]	N	[GJ]	N	[GJ]	N	[GJ]	N	[GJ]	P	[OG]
Planar	P	[GJ]	N	[GJ]	P	[T]	N	[OG]	N	[GJ]	O		N	[GJ]	N	[GJ]	P	[GJ]	N	[OG]	P	[GJ]
Series Parallel	P	[OG]	P	[OG]	P	[T]	**P**	**[S]**	P	[OG]	P	[OG]	P	[OG]	P	[OG]	P	[GJ]	P	[OG]	P	[GJ]
Outerplanar	P		P	[OG]	P	[T]	P	[OG]	P	[OG]	P	[OG]	P	[T]	P	[OG]	P	[GJ]	P	[OG]	P	[GJ]
Halin	P		P	[OG]	P	[T]	P	[OG]	P	[OG]	P	[OG]	P	[T]	P	[OG]	P	[GJ]	**P**	**[S]**	P	[GJ]
k-Outerplanar	P		P	[OG]	P	[T]	P	[OG]	P	[OG]	O?		P	[OG]	P	[OG]	P	[GJ]	P?		P	[GJ]
Grid	P		P	[GJ]	P	[T]	P	[T]	P	[T]	P	[GJ]	N	[OG]	N	[OG]	P	[T]	N	[OG]	P	[GJ]
$K_{3,3}$-Free	P	[OG]	N	[GJ]	P	[T]	N	[GJ]	N	[GJ]	O?		N	[GJ]	N	[GJ]	P	[OG]	N	[GJ]	**I**	**[S]**
Thickness-k	N	[OG]	P	[GJ]	P	[T]	N	[GJ]	N	[GJ]	N	[OG]	N	[GJ]	N	[GJ]	N	[OG]	N	[GJ]	O?	
Genus-k	P	[OG]	P	[GJ]	P	[T]	N	[GJ]	N	[GJ]	O?		N	[GJ]	N	[GJ]	O?		N	[GJ]	P	[OG]
Perfect	**P**	**[S]**	P	[OG]	P	[OG]	P	[OG]	P	[OG]	**N**	**[S]**	N	[OG]	N	[OG]	**N**	**[S]**	N	[GJ]	I	[GJ]
Chordal	P	[OG]	P	[OG]	P	[OG]	P	[OG]	P	[OG]	O?		N	[OG]	N	[OG]	**N**	**[S]**	N	[OG]	I	[GJ]
Split	P	[OG]	P	[OG]	P	[OG]	P	[OG]	P	[OG]	O?		N	[OG]	N	[OG]	**N**	**[S]**	N	[OG]	I	[OG]
Strongly Chordal	P	[OG]	P	[OG]	P	[OG]	P	[OG]	P	[OG]	O?		**N**	**[S]**	P	[OG]	**N**	**[S]**	P	[OG]	**I**	**[S]**
Comparability	P	[OG]	P	[OG]	P	[OG]	P	[OG]	P	[OG]	**N**	**[S]**	N	[OG]	N	[OG]	**N**	**[S]**	N	[GJ]	I	[GJ]
Bipartite	P	[T]	P	[GJ]	P	[T]	P	[GJ]	P	[T]	P	[GJ]	N	[OG]	N	[OG]	P	[T]	N	[GJ]	I	[GJ]
Permutation	P	[OG]	P	[OG]	P	[OG]	P	[OG]	P	[OG]	O?		**P**	**[S]**	P	[OG]	O?		P	[OG]	P	[OG]
Cographs	P	[T]	P	[OG]	P	[OG]	P	[OG]	P	[OG]	O?		P	[OG]	P	[OG]	**P**	**[S]**	P	[OG]	P	[OG]
Undirected Path	P	[OG]	P	[OG]	P	[OG]	P	[OG]	P	[OG]	O?		**N**	**[S]**	N	[OG]	**N**	**[S]**	O?		I	[GJ]
Directed Path	P	[OG]	P	[OG]	P	[OG]	P	[OG]	P	[OG]	O?		**N**	**[S]**	P	[OG]	O?		P	[OG]	O?	
Interval	P	[OG]	P	[OG]	P	[OG]	P	[OG]	P	[OG]	O?		P	[OG]	P	[OG]	O?		P	[OG]	P	[OG]
Circular Arc	P	[OG]	P	[OG]	P	[OG]	P	[OG]	N	[OG]	O?		**P**	**[S]**	P	[OG]	O?		P	[OG]	O?	
Circle	P	[OG]	P	[GJ]	P	[OG]	**N**	**[S]**	N	[OG]	O?		P	[OG]	**N**	**[S]**	**N**	**[S]**	P	[OG]	O?	
Proper Circ. Arc	P	[OG]	P	[OG]	P	[OG]	P	[OG]	P	[OG]	O?		P	[OG]	P	[OG]	O?		P	[OG]	**P**	**[S]**
Edge (or Line)	P	[OG]	P	[GJ]	P	[T]	N	[GJ]	N	[OG]	**N**	**[S]**	N	[OG]	N	[GJ]	**P**	**[S]**	N	[OG]	I	[OG]
Claw-Free	P	[T]	P	[OG]	**N**	**[S]**	N	[GJ]	N	[OG]	**N**	**[S]**	N	[OG]	N	[GJ]	**N**	**[S]**	N	[OG]	I	[OG]

The updated table from 1985 to 2018, there are 23 **new references**, classifying 33 former open problems. We keep the abbreviations used in [GJ]: P = Polynomial-time solvable, P? = Appears to be polynomial-time by standard techniques, N = NP-complete, I = Open, but equivalent in complexity to general GRAPH ISOMORPHISM, O?= Apparently open, but possibly easy to resolve, O = Open, and may well be hard, T (as a reference) = restriction trivializes the problem, and GJ (as a reference) = the Guide [GJ]. We use abbreviation OG (as a reference) = the Ongoing guide [OG], please refer to this reference for the entry. There are 33 new entries, all in bold, please refer to the survey paper [S].

[GJ] M.R. Garey, D.S. Johnson, Computers and Intractability, A Guide to the Theory of NP-completeness, WH Freeman, 1979.

[OG] D.S. Johnson, Graph restrictions and their effect, J. Algorithms 6 (1985) 434–451.

[S] C.M.H. de Figueiredo, Complexity-separating graph classes for vertex, edge and total-colouring, Discrete Applied Math. (2019).

Dicotomias cheias

Classificar todo membro de uma família de problemas como
Polinomial (fácil) ou NP-completo (difícil)

Quais grafos tornam o problema fácil ou difícil?

Teoremas de dicotomia fornecem bons projetos de pesquisa:
simples de formular, mas trabalhoso para concluir

Requerem domínio de técnicas para desenvolvimento de algoritmos e para provas de dificuldade computacional

Conclusão

Todo grafo é fácil ou difícil

O problema do milênio sobre intratabilidade investiga limitações computacionais fundamentais

Apresentamos dicotomias onde dizemos precisamente quais grafos tornam o problema fácil ou difícil

Exploramos a fronteira de complexidade através da classificação em polinomial ou NP-completo