Choosing Resources

Damien Woods

University of Seville

$\Rightarrow \quad \Rightarrow$

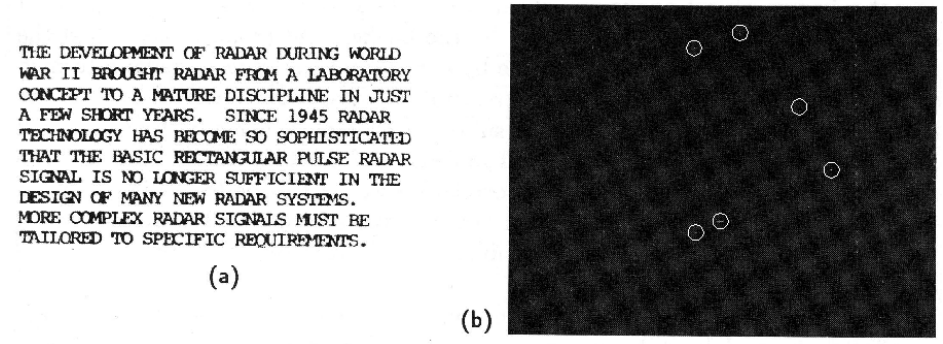
Caltech

Complexity Resources in Physical Computation, 2009

Acknowledgement: Joint work with Thomas J. Naughton & Niall Murphy

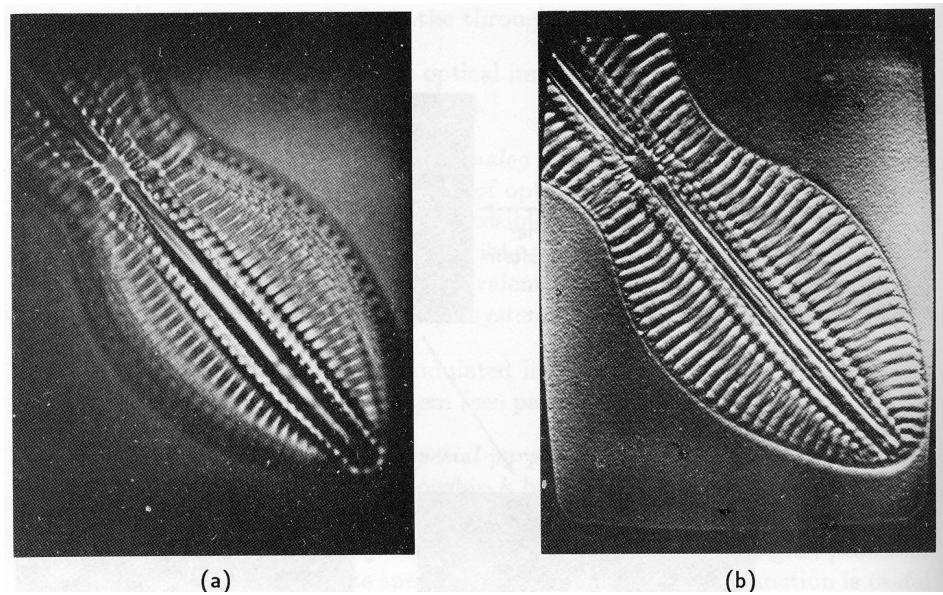
Optical computing: motivations

Object detection (via correlation)



D. Casasent, Proc. IEEE 67, 813 (1979)

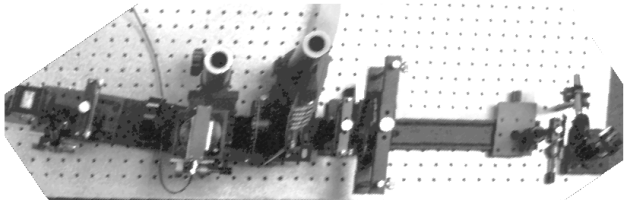
Image processing (convolution: e.g. deblurring, edge enhancement, noise removal)



G. Häusler, Optica Acta 24, 965 (1977)

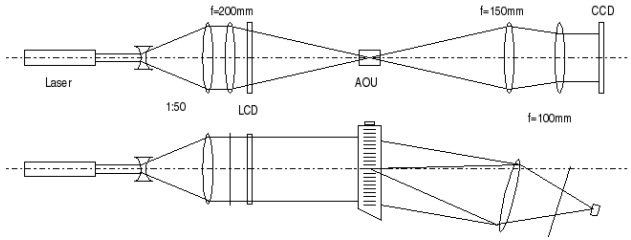
Optical computing: motivations

Previously: relatively little theory for optical computers compared to implementations and designs

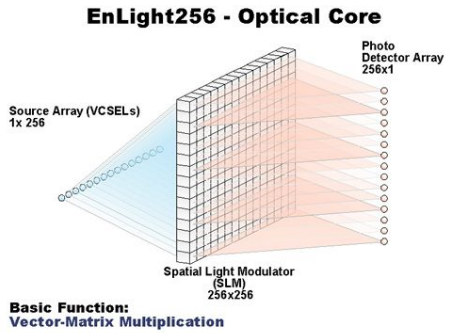


T. Naughton, et al., Opt. Eng. 38, 1170, 1999

Lenslet Enlight (2001)

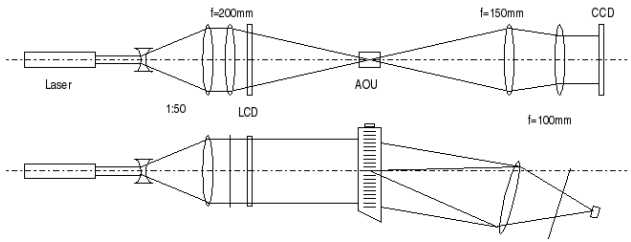


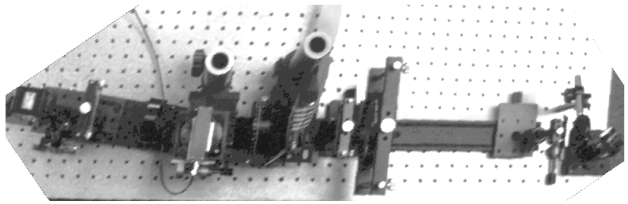
Opposite to many other models! Quantum, DNA, membrane, ...



Optical computing: motivations

Previously: relatively little theory for optical computers compared to implementations and designs

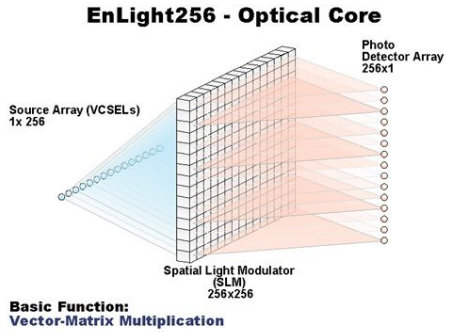




T. Naughton, et al., Opt. Eng. 38, 1170, 1999

Opposite to many other models! Quantum, DNA, membrane, ...

Lenslet Enlight (2001)



Our work: development of a general purpose optical model
Joint work with Thomas J. Naughton

Motivations

- Optical computers been around for some time
- Speed: parallelism via 2D complex-valued functions
- No inherent noise during transmission
- Optical pathways can be placed arbitrarily close together
- Photons do not need a conductor to be transmitted (free space propagation)
- High interconnection densities are possible
- Optical pathways can be switched at arbitrary data rates
- Energy efficient (no heat and no additional energy costs for cooling down processors)

Continuous space machine definition

- Images are the basic data units
- A *(complex-valued) image* is a function

$$f : [0, 1) \times [0, 1) \to \mathbb{C}$$

where $[0, 1)$ is the half-open real unit interval

Continuous space machine definition

An address is a pair $(\xi, \eta) \in \mathbb{N}^+ \times \mathbb{N}^+$

continuous space machine

A CSM is a quintuple $M = (\mathfrak{E}, L, I, P, O)$, where

- $\mathfrak{E} : \mathbb{N} \to \mathcal{N}$ is the address encoding function
- $L = ((s_\xi, s_\eta), (a_\xi, a_\eta), (b_\xi, b_\eta))$ are the addresses: *sta*, *a*, *b*; $a \neq b$
- I and O are finite sets of input and output addresses, respectively
- $P = \{(\zeta_1, p_{1_\xi}, p_{1_\eta}), \ldots, (\zeta_r, p_{r_\xi}, p_{r_\eta})\}$ are the r programming symbols ζ_j and their addresses where $\zeta_j \in (\{h, v, \ldots, hlt\} \cup \mathcal{N}) \subset \mathcal{I}$

configuration

A *configuration* of M is a pair $\langle c, e \rangle$ where

- c is an address called the control
- e is a list of M's images

CSM operations

h	: horizontal 1D Fourier transform on in *a*
v	: vertical 1D Fourier transform on image in *a*
$*$	: complex conjugate of image in *a*
$\cdot$	: pointwise multiplication of *a* and *b*
$+$	: pointwise complex addition of *a* and *b*
ρ z_l z_u	: $z_\mathrm{l}, z_\mathrm{u} \in \mathcal{I}$; filter *a* by amplitude using z_l and z_u as lower and upper amplitude threshold images
st ξ_1 ξ_2 η_1 η_2	: $\xi_1, \xi_2, \eta_1, \eta_2 \in \mathbb{N}$; copy the image in *a* into the rectangle of images whose bottom left-hand corner address is (ξ_1, η_1) and whose top right-hand corner address is (ξ_2, η_2)
ld ξ_1 ξ_2 η_1 η_2	: $\xi_1, \xi_2, \eta_1, \eta_2 \in \mathbb{N}$; copy into *a* the rectangle of images whose bottom left-hand corner address is (ξ_1, η_1) and whose top right-hand corner address is (ξ_2, η_2)
br ξ η	: $\xi, \eta \in \mathbb{N}$; branch to the image at address (ξ, η)
hlt	: halt

CSM programming language

$\mathrm{h}(i_1;i_2)$: replace image at i_2 with horizontal 1D FT of i_1

$\mathrm{v}(i_1;i_2)$: replace image at i_2 with vertical 1D FT of i_1

$*(i_1;i_2)$: replace image at i_2 with complex conjugate of i_1

$\cdot(i_1,i_2;i_3)$: pointwise multiplication of i_1 and i_2, result in i_3

$+(i_1,i_2;i_3)$: pointwise addition of i_1 and i_2, result at i_3

$\rho(i_1,z_\mathrm{l},z_\mathrm{u};i_2)$: filter i_1 by amplitude using z_l, z_u as lower & upper amplitude threshold images

$[\xi_1',\xi_2',\eta_1',\eta_2'] \leftarrow [\xi_1,\xi_2,\eta_1,\eta_2]$: copy the rectangle of images with

- bottom-left address (ξ_1,η_1) & top-right address (ξ_2,η_2)

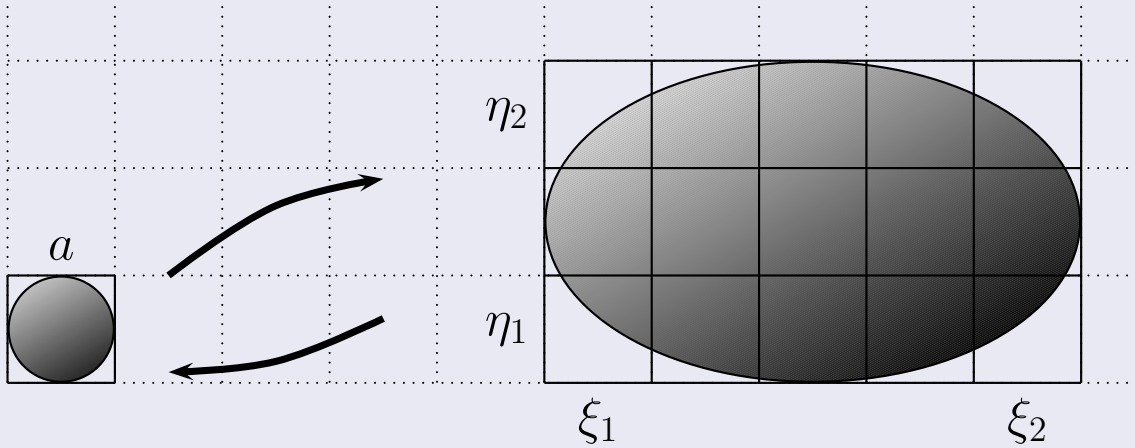
to the rectangle with

- bottom-left address (ξ_1',η_1') top-right address (ξ_2',η_2')

There are also **if/else** and **while** control flow instructions with binary symbol image conditions.

Example

Copying images



$[\xi_1, \xi_2, \eta_1, \eta_2] \leftarrow a$

$a \leftarrow [\xi_1, \xi_2, \eta_1, \eta_2]$

CSM complexity measures

TIME

The number of configurations in the computation sequence of M, beginning with the initial configuration and ending with the first final configuration.

GRID

The minimum number of images, arranged in a rectangular grid, for M to compute correctly on all inputs.

Let $S : \mathcal{I} \times (\mathbb{N} \times \mathbb{N}) \to \mathcal{I}$, where $S(f(x, y), (\Phi, \Psi))$ is a raster image, with $\Phi\Psi$ pixels arranged in Φ columns and Ψ rows, that approximates $f(x, y)$.

SPATIALRES

The minimum $\Phi\Psi$ such that if each image $f(x, y)$ in the computation of M is replaced with $S(f(x, y), (\Phi, \Psi))$ then M computes correctly on all inputs.

CSM complexity measures

TIME

The number of configurations in the computation sequence of M, beginning with the initial configuration and ending with the first final configuration.

GRID

The minimum number of images, arranged in a rectangular grid, for M to compute correctly on all inputs.

Let $S : \mathcal{I} \times (\mathbb{N} \times \mathbb{N}) \to \mathcal{I}$, where $S(f(x, y), (\Phi, \Psi))$ is a raster image, with $\Phi\Psi$ pixels arranged in Φ columns and Ψ rows, that approximates $f(x, y)$.

SPATIALRES

The minimum $\Phi\Psi$ such that if each image $f(x, y)$ in the computation of M is replaced with $S(f(x, y), (\Phi, \Psi))$ then M computes correctly on all inputs.

CSM complexity measures

TIME

The number of configurations in the computation sequence of M, beginning with the initial configuration and ending with the first final configuration.

GRID

The minimum number of images, arranged in a rectangular grid, for M to compute correctly on all inputs.

Let $S : \mathcal{I} \times (\mathbb{N} \times \mathbb{N}) \rightarrow \mathcal{I}$, where $S(f(x, y), (\Phi, \Psi))$ is a raster image, with $\Phi\Psi$ pixels arranged in Φ columns and Ψ rows, that approximates $f(x, y)$.

SPATIALRES

The minimum $\Phi\Psi$ such that if each image $f(x, y)$ in the computation of M is replaced with $S(f(x, y), (\Phi, \Psi))$ then M computes correctly on all inputs.

CSM complexity measures

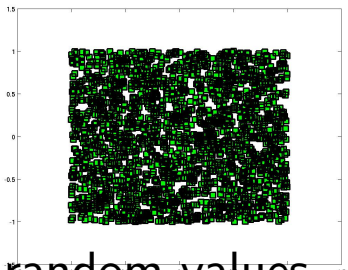
Let $f(x,y) = |f(x,y)|\mathrm{e}^{\mathrm{i}\arg f(x,y)}$. Let $A : \mathcal{I} \times \mathbb{N}^+ \to \mathcal{I}$,

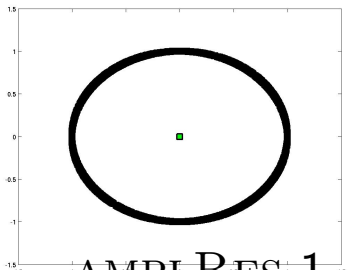
$$A(f(x,y),\mu) = \left\lfloor |f(x,y)|\mu + \frac{1}{2} \right\rfloor \frac{1}{\mu} \mathrm{e}^{\mathrm{i}\arg f(x,y)}$$

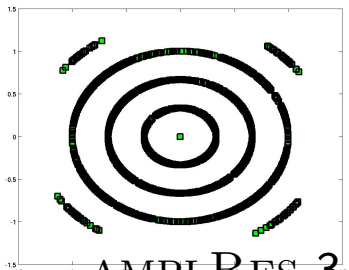
The value μ is the cardinality of the set of discrete nonzero amplitude values that each complex value in $A(f,\mu)$ can take, per half-open unit interval of amplitude.

amplRes

The minimum μ such that if each image $f(x,y)$ in the computation of M is replaced by $A(f(x,y),\mu)$ then M computes correctly on all inputs.


random values


amplRes 1


amplRes 3

CSM complexity measures

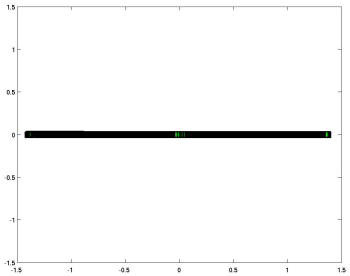
Let $P : \mathcal{I} \times \mathbb{N}^+ \to \mathcal{I}$,

$$P(f(x,y),\mu) = |f(x,y)|e^{\mathrm{i}\left\lfloor \arg(f(x,y))\frac{\mu}{2\pi}+\frac{1}{2}\right\rfloor \frac{2\pi}{\mu}}$$

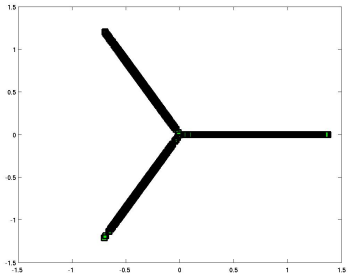
The value μ is the cardinality of the set of discrete phase values that each complex value in $P(f,\mu)$ can take.

phaseRes

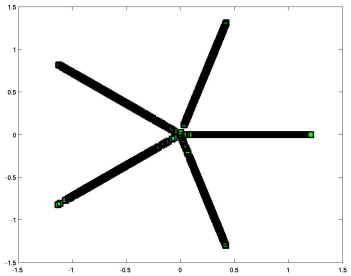
The minimum μ such that if each image $f(x,y)$ in the computation of M is replaced by $P(f(x,y),\mu)$ then M computes correctly on all inputs.



phaseRes 2



phaseRes 3



phaseRes 5

CSM complexity measures

dyRange

The ceiling of the maximum of all the amplitude values stored in all of M's images during M's computation.

FREQ

The minimum optical frequency such that M computes correctly on all inputs.

SPACE

The product of all of M's complexity measures except TIME.

We have defined complexity of computations, we extend this to complexity of configurations and images in a straightforward way.

An upper bound on "Energy complexity" is given by a product of the above measures (except phaseRes)

CSM complexity measures

DYRANGE

The ceiling of the maximum of all the amplitude values stored in all of M's images during M's computation.

FREQ

The minimum optical frequency such that M computes correctly on all inputs.

SPACE

The product of all of M's complexity measures except TIME.

We have defined complexity of computations, we extend this to complexity of configurations and images in a straightforward way.

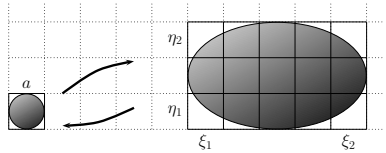
An upper bound on "Energy complexity" is given by a product of the above measures (except PHASERES)

Optical computing: resources

TIME: Number of steps

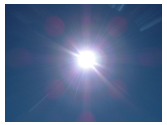
1,2,3,...

GRID: # images

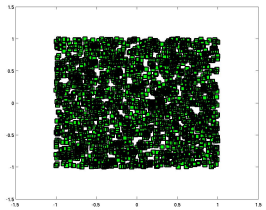


SPATIALRES: # pixels

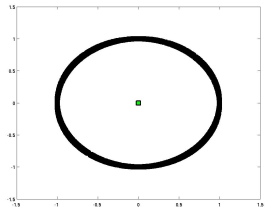
DYRANGE: Max amplitude



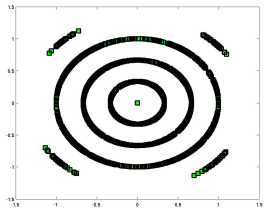
AMPLRES: Amplitude levels



AMPLRES ∞

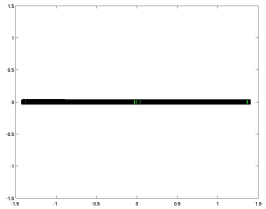


AMPLRES 1

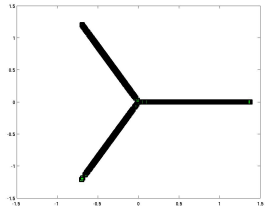


AMPLRES 3

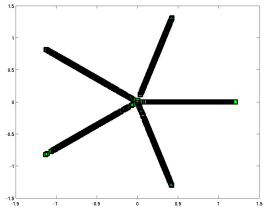
PHASERES: Phase of images



PHASERES 2

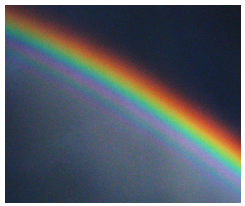


PHASERES 3



PHASERES 5

FREQ: Frequency



Resource usage after one timestep

- In sequential models of computation the increase in space-like resources after a single step is quite obvious
- Parallel models may exhibit large growth
- Here we have many operations and measures; the growth in space-like resources is not immediately obvious

Resource usage after one timestep

	GRID	SPATIALRES	AMPLRES	DYRANGE	PHASERES	FREQ
h	G_T	∞	∞	∞	∞	∞
v	G_T	∞	∞	∞	∞	∞
$*$	G_T	$R_{\mathrm{S},T}$	$R_{\mathrm{A},T}$	$R_{\mathrm{D},T}$	$R_{\mathrm{P},T}$	ν_T
$\cdot$	G_T	$R_{\mathrm{S},T}$	$(R_{\mathrm{A},T})^2$	$(R_{\mathrm{D},T})^2$	$R_{\mathrm{P},T}$	ν_T
$+$	G_T	$R_{\mathrm{S},T}$	∞	$2R_{\mathrm{D},T}$	∞	ν_T
ρ	unbnd	$R_{\mathrm{S},T}$	$R_{\mathrm{A},T}$	$R_{\mathrm{D},T}$	$R_{\mathrm{P},T}$	ν_T
st	unbnd	$R_{\mathrm{S},T}$	$R_{\mathrm{A},T}$	$R_{\mathrm{D},T}$	$R_{\mathrm{P},T}$	ν_T
ld	unbnd	unbnd	$R_{\mathrm{A},T}$	$R_{\mathrm{D},T}$	$R_{\mathrm{P},T}$	unbnd
br	G_T	$R_{\mathrm{S},T}$	$R_{\mathrm{A},T}$	$R_{\mathrm{D},T}$	$R_{\mathrm{P},T}$	ν_T
hlt	G_T	$R_{\mathrm{S},T}$	$R_{\mathrm{A},T}$	$R_{\mathrm{D},T}$	$R_{\mathrm{P},T}$	ν_T

Table: Worst case (lub) resource usage at TIME $T+1$, in terms of resource use at T: GRID $= G_T$, SPATIALRES $= R_{\mathrm{S},T}$, AMPLRES $= R_{\mathrm{A},T}$, DYRANGE $= R_{\mathrm{D},T}$, PHASERES $= R_{\mathrm{P},T}$, FREQ $= \nu_T$.

CSM decides any language $L \in 2^{\{0,1\}^*}$.

Resource usage after one timestep

	GRID	SPATIALRES	AMPLRES	DYRANGE	PHASERES	FREQ
h	G_T	∞	∞	∞	∞	∞
v	G_T	∞	∞	∞	∞	∞
$*$	G_T	$R_{\mathrm{S},T}$	$R_{\mathrm{A},T}$	$R_{\mathrm{D},T}$	$R_{\mathrm{P},T}$	ν_T
$\cdot$	G_T	$R_{\mathrm{S},T}$	$(R_{\mathrm{A},T})^2$	$(R_{\mathrm{D},T})^2$	$R_{\mathrm{P},T}$	ν_T
$+$	G_T	$R_{\mathrm{S},T}$	∞	$2R_{\mathrm{D},T}$	∞	ν_T
ρ	unbnd	$R_{\mathrm{S},T}$	$R_{\mathrm{A},T}$	$R_{\mathrm{D},T}$	$R_{\mathrm{P},T}$	ν_T
st	unbnd	$R_{\mathrm{S},T}$	$R_{\mathrm{A},T}$	$R_{\mathrm{D},T}$	$R_{\mathrm{P},T}$	ν_T
ld	unbnd	unbnd	$R_{\mathrm{A},T}$	$R_{\mathrm{D},T}$	$R_{\mathrm{P},T}$	unbnd
br	G_T	$R_{\mathrm{S},T}$	$R_{\mathrm{A},T}$	$R_{\mathrm{D},T}$	$R_{\mathrm{P},T}$	ν_T
hlt	G_T	$R_{\mathrm{S},T}$	$R_{\mathrm{A},T}$	$R_{\mathrm{D},T}$	$R_{\mathrm{P},T}$	ν_T

Table: Worst case (lub) resource usage at TIME $T+1$, in terms of resource use at T: GRID $= G_T$, SPATIALRES $= R_{\mathrm{S},T}$, AMPLRES $= R_{\mathrm{A},T}$, DYRANGE $= R_{\mathrm{D},T}$, PHASERES $= R_{\mathrm{P},T}$, FREQ $= \nu_T$.

CSM decides any language $L \in 2^{\{0,1\}^*}$.

$\mathcal{C}_2$-CSM

Motivated by a desire to apply standard complexity theory tools to the model, we define a restricted class of CSM.

$\mathcal{C}_2$-CSM

- AMPLRES and PHASERES have constant value of 2
- at TIME t each of GRID, SPATIALRES, DYRANGE is $O(2^t)$
- replace FT with DFT
- address encoding is computable in logspace

2005: lower and upper bounds on $\mathcal{C}_2$-CSM power

The $\mathcal{C}_2$-CSM verifies the parallel computation thesis

$\Leftrightarrow$ $\mathcal{C}_2$-CSM TIME is (polynomially) equivalent to sequential space

$\Leftrightarrow \mathcal{C}_2\text{-CSM–TIME}(S^{O(1)}(n)) = \text{NSPACE}(S^{O(1)}(n))$

For example, $\mathcal{C}_2\text{-CSM–TIME}(n^{O(1)}) = \text{PSPACE}$
For example, $\mathcal{C}_2$-CSMs solve NP-complete problems polynomial time, but use exponential SPACE (of course!)

Poly SPACE, polylog TIME $\mathcal{C}_2$-CSMs accept exactly NC
i.e. $\mathcal{C}_2\text{-CSM–SPACE, TIME}(n^{O(1)}, \log^{O(1)} n) = \text{NC}$

These characterisations are robust wrt variations in the $\mathcal{C}_2$-CSM definition

2005: lower and upper bounds on $\mathcal{C}_2$-CSM power

The $\mathcal{C}_2$-CSM verifies the parallel computation thesis

$\Leftrightarrow$ $\mathcal{C}_2$-CSM TIME is (polynomially) equivalent to sequential space

$\Leftrightarrow$ $\mathcal{C}_2$-CSM–$\mathrm{TIME}(S^{O(1)}(n)) = \mathrm{NSPACE}(S^{O(1)}(n))$

For example, $\mathcal{C}_2$-CSM–$\mathrm{TIME}(n^{O(1)}) = \mathrm{PSPACE}$
For example, $\mathcal{C}_2$-CSMs solve NP-complete problems polynomial time, but use exponential SPACE (of course!)

Poly SPACE, polylog TIME $\mathcal{C}_2$-CSMs accept exactly NC
i.e. $\mathcal{C}_2$-CSM–$\mathrm{SPACE}, \mathrm{TIME}(n^{O(1)}, \log^{O(1)} n) = \mathrm{NC}$

These characterisations are robust wrt variations in the $\mathcal{C}_2$-CSM definition

Pixels, pixels, pixels,...

- We already know that:
- $\mathrm{PSPACE} = \mathcal{C}_2\text{-CSM}$ poly-TIME, exp-SPATIALRES
- Result holds for constant $O(1)$ usuage of the other resources: DYRANGE, GRID, AMPLRES, PHASERES

parallelism $\approx$? pixels

- Backed up by existing intuition through many examples of optical algorithms

Pixels, pixels, pixels,...

- We already know that:
- $\mathrm{PSPACE} = \mathcal{C}_2\text{-CSM}$ poly-TIME, exp-SPATIALRES
- Result holds for constant $O(1)$ usuage of the other resources: DYRANGE, GRID, AMPLRES, PHASERES

parallelism $\approx$? pixels

- Backed up by existing intuition through many examples of optical algorithms

Pixels, pixels, pixels,...

- We already know that:
- $\mathrm{PSPACE} = \mathcal{C}_2\text{-CSM}$ poly-TIME, exp-SPATIALRES
- Result holds for constant $O(1)$ usuage of the other resources: DYRANGE, GRID, AMPLRES, PHASERES

parallelism $\approx$? pixels

- Backed up by existing intuition through many examples of optical algorithms

Pixels, pixels, pixels,...

- We already know that:
- $\mathrm{PSPACE} = \mathcal{C}_2\text{-CSM}$ poly-TIME, exp-SPATIALRES
- Result holds for constant $O(1)$ usuage of the other resources: DYRANGE, GRID, AMPLRES, PHASERES

parallelism $\approx$? pixels

- Backed up by existing intuition through many examples of optical algorithms

Parallelism without pixels?

- What if we restrict (fix!) the number of pixels?

i.e. $O(1)$ SPATIALRES

- Our previous highly parallel algorithms don't work
- Have we crippled the system?

Parallelism without pixels?

- What if we restrict (fix!) the number of pixels?

i.e. $O(1)$ SPATIALRES

- Our previous highly parallel algorithms don't work
- Have we crippled the system?
- No!

Constant number of pixels

Theorem

PSPACE *is characterised by* $\mathcal{C}_2$-CSM*s that are restricted to use polynomial* TIME $T = O(n^k)$, SPATIALRES $O(1)$, GRID $O(1)$, *and generalised to use* AMPLRES $O(2^{2^T})$, DYRANGE $O(2^{2^T})$.

- Proof idea (upperbound). Extend previous upperbound, swapping the roles of SPATIALRES and the other resources.
- Proof idea (lowerbound). Via simulation of RAM$(\times, +, \leftarrow)$. Such RAMs are known to characterise PSPACE in polynomial time.

Constant number of pixels

Theorem

PSPACE *is characterised by* $\mathcal{C}_2$-CSM*s that are restricted to use polynomial* TIME $T = O(n^k)$, SPATIALRES $O(1)$, GRID $O(1)$, *and generalised to use* AMPLRES $O(2^{2^T})$, DYRANGE $O(2^{2^T})$.

- We can get "high parallelism" with a fixed number of pixels!
- Intuition — there are at least two ways to compute quickly in optics: use pixels or generate large numbers
- However, not a realistic way to do optical computing: using large AMPLRES and DYRANGE is more expensive and unrealistic than large SPATIALRES and/or GRID
- From the proofs: we are using multiplication, rather than pixels
- So what happens if we dissallow (such unrealistic) multiplication?

What happens if we remove the multiplication operation?

Theorem

$\mathcal{C}_2$-CSMs without multiplication, that compute in polynomial TIME, *polynomial* GRID $O(n^k)$, *and constant* SPATIALRES $O(1)$, *characterise P.*

Theorem

$\mathcal{C}_2$-CSMs without multiplication, that compute in polynomial TIME, *constant* GRID $O(1)$, *polynomial* SPATIALRES $O(n^k)$, *characterise P.*

- Significant reduction in power
- These results are general in the sense that the other resources are arbitrary (i.e. unrestricted GRID, DYRANGE, PHASERES, AMPLRES)
- More evidence for parallelism $\approx$ pixels in optics?

Conclusions

- Two ways in which we get huge parallelism from optics:
- Characterise PSPACE in poly TIME, but exp SPATIALRES
- Char. PSPACE in poly TIME, but exp AMPLRES & DYRANGE
- Remove multiplication $\Rightarrow$ characterise P
- Corollaries: characterisations of NC via optical machines that run in polylog TIME & polynomial SPACE
- So we can take existing, fast parallel algorithms and compile to fast parallel optical algorithms
- log time searching algorithm & implementation design

Conclusions

- NC: problems solved in polylog time, and poly space
- NC $\subseteq$ P
- Rather than focus on (presumed) hard problems, perhaps the optical computing community can get more out of optical computers by finding NC problems that are well-suited to optical architectures

Membrane Computing

- Joint work with Niall Murphy
- Complexity below P
- Adleman's choice matters!

Uniformity: Introduction

- Families of finite computing devices: Boolean circuits, DNA computers, membrane systems, cellular automata, tile assembly systems, etc.
- Extremely powerful in the absence of an restrictions on the structure of the family
- Uniform: Family C is uniform if $\exists M : n \longrightarrow c_n$, where $n \in \mathbb{N}$ is the length of the input w.
- However, there are actually two distinct kinds of uniformity conditions in above list!
- Semi-uniform: Family C is semi-uniform if $\exists M : w \longrightarrow c_w$.

Uniformity

E.g. a graph problem: Hamiltonian path

"Uniform": Input length $n \longrightarrow$ Circuit c_n
(works for all graphs of size n)

Adleman's Hamiltonian path experiment:

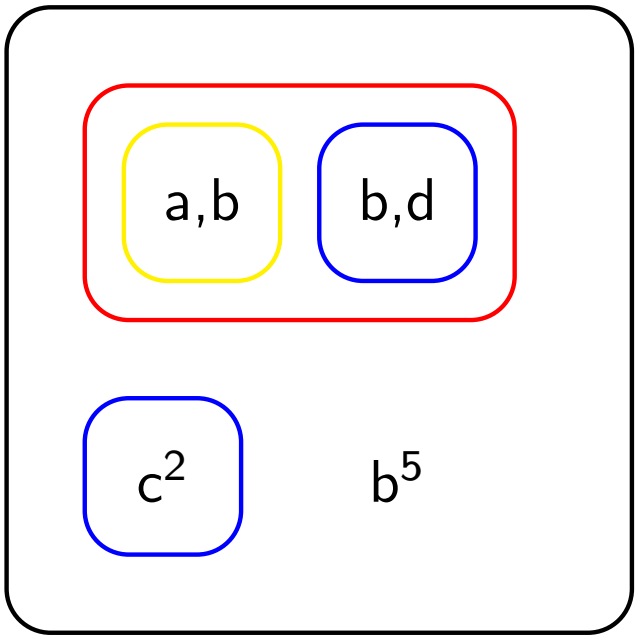
"Semi-uniform": Graph $G \longrightarrow$ DNA-device m_G

Many, many "nature inspired" algorithms (collections of devices) are semi-uniform rather than uniform!

Open problem: Are these notions really the same or not?

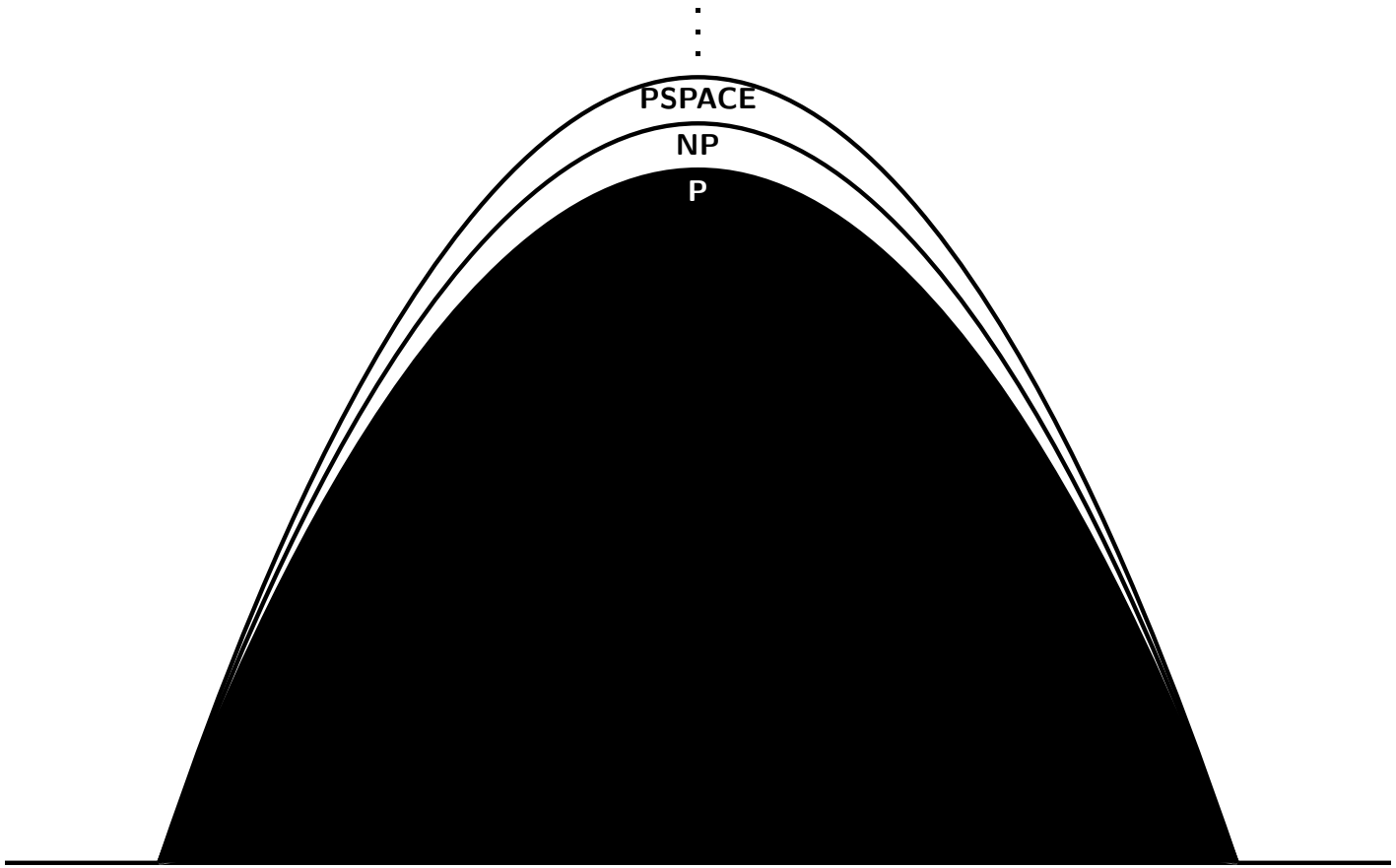
[Paŭn, 2005]

Membranes

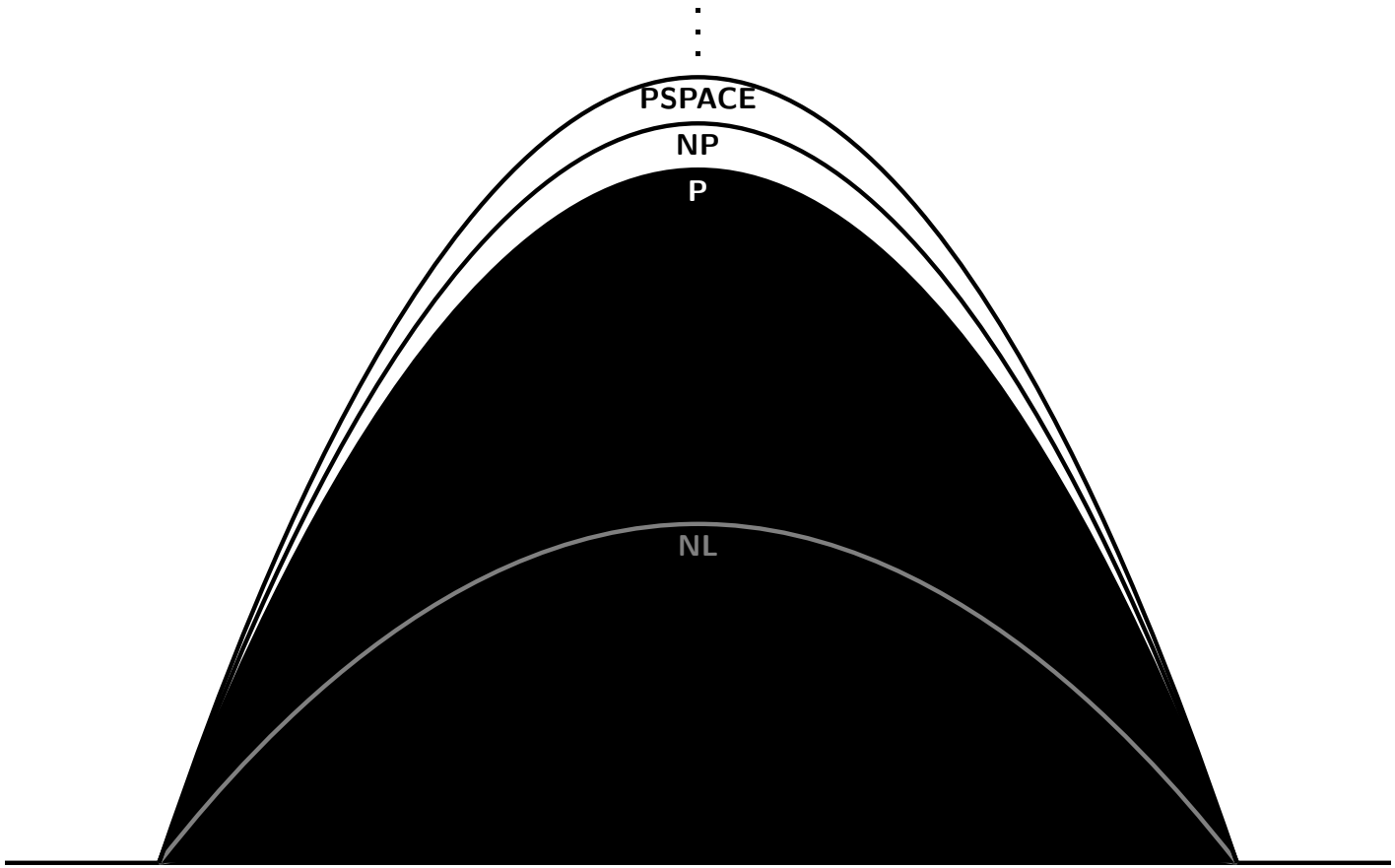


$$[\ a\ ] \longrightarrow cccd$$
$$[\ a\ ] \longrightarrow [\ d\ ][\ d\ ]$$
$$[\ c\ ] \longrightarrow \lambda$$
$$[\ a\ ] \longrightarrow [\ ]c$$
$$[\ ]d \longrightarrow [\ a\ ]$$

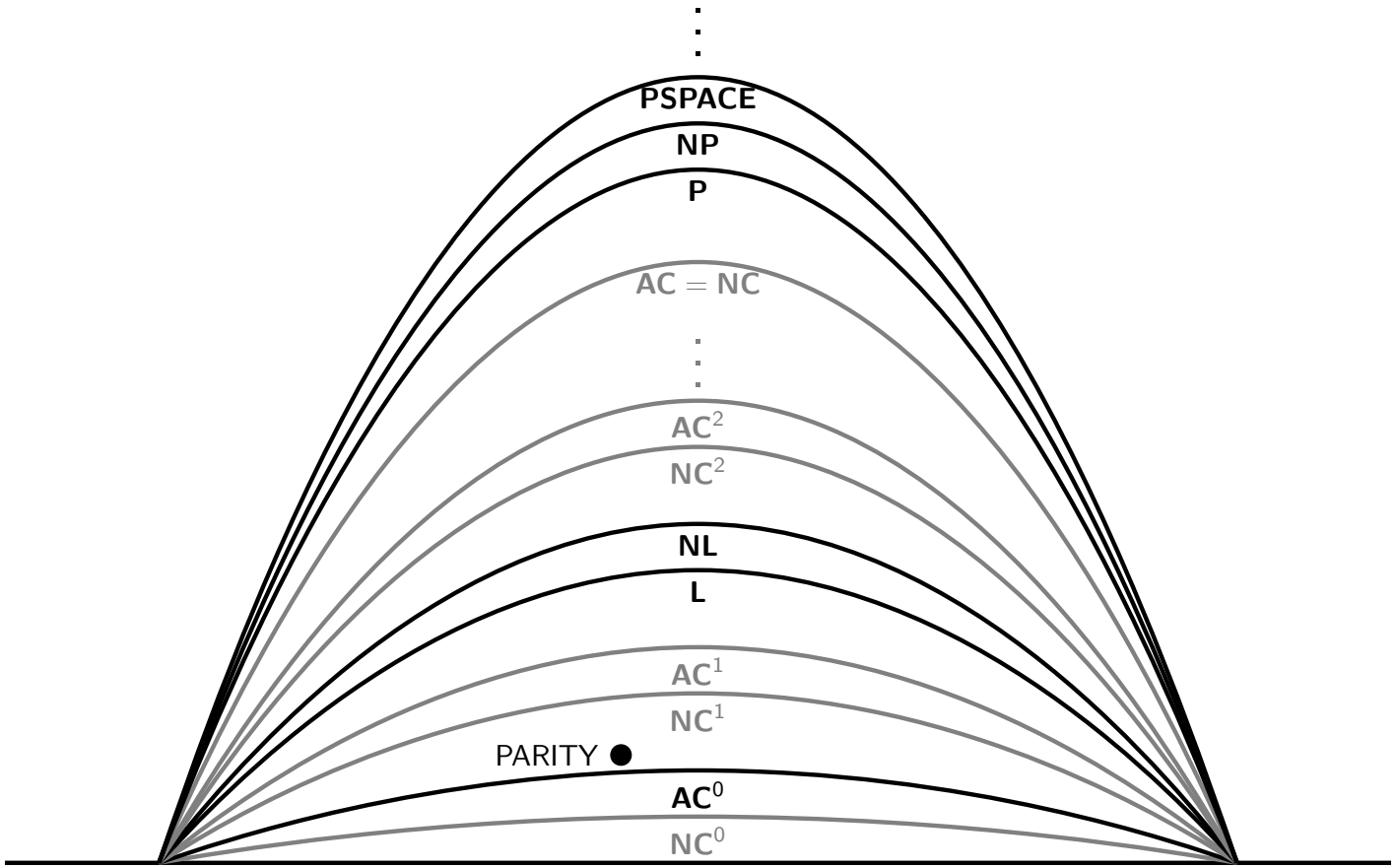
The world according to **P**-uniformity



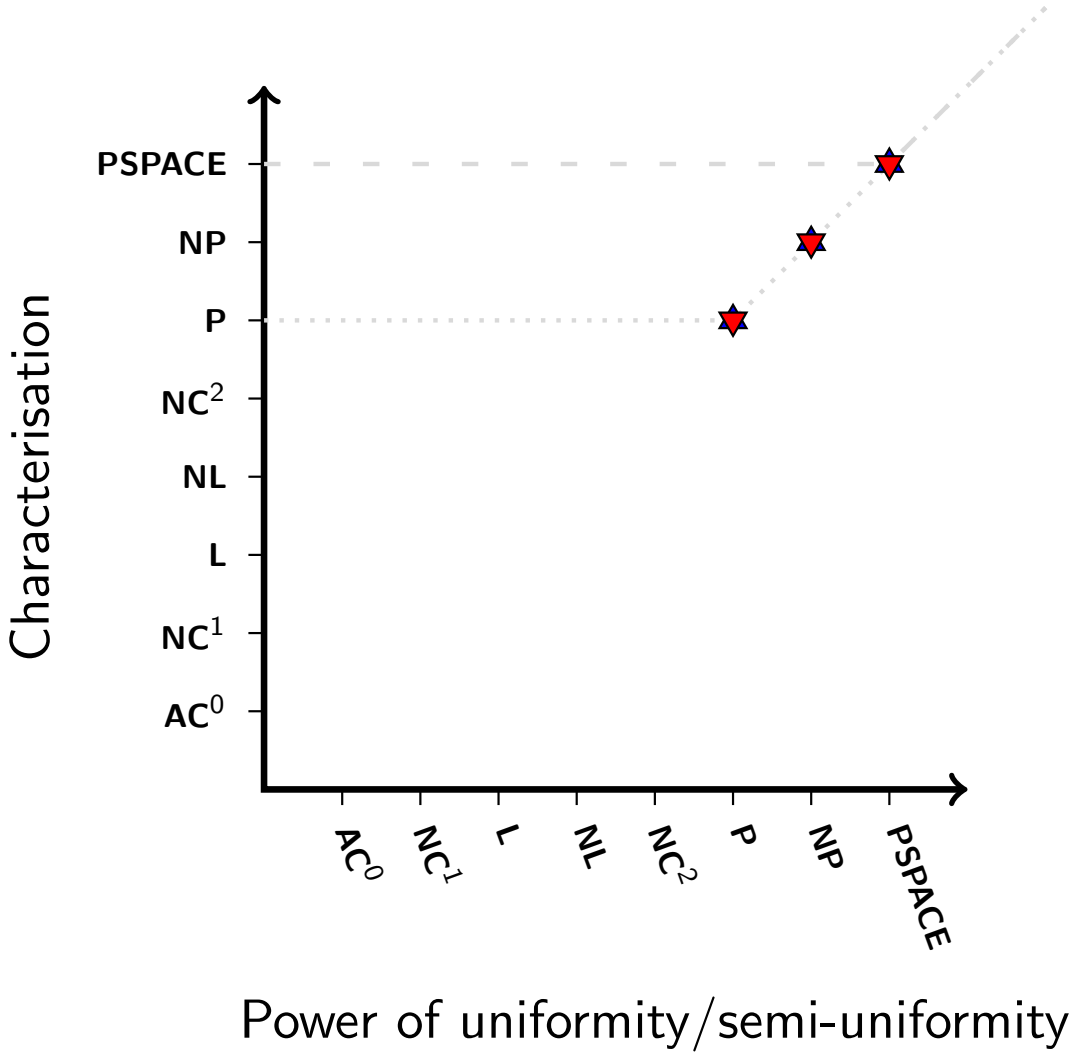
The world according to **P**-uniformity



The world according to sub-**P**-uniformity



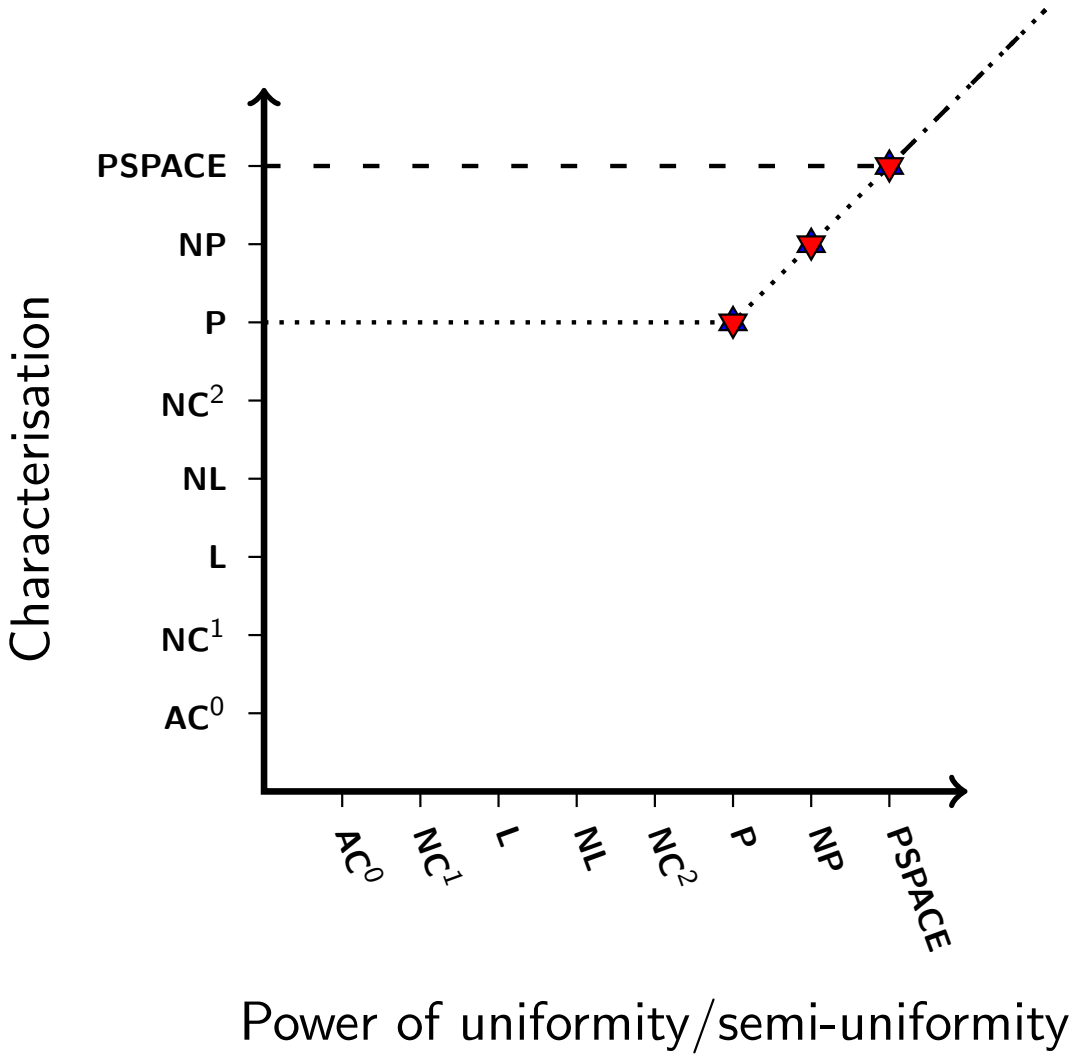
Semi-uniformity vs uniformity



Uniform: ▲

Semi-uniform: ▼

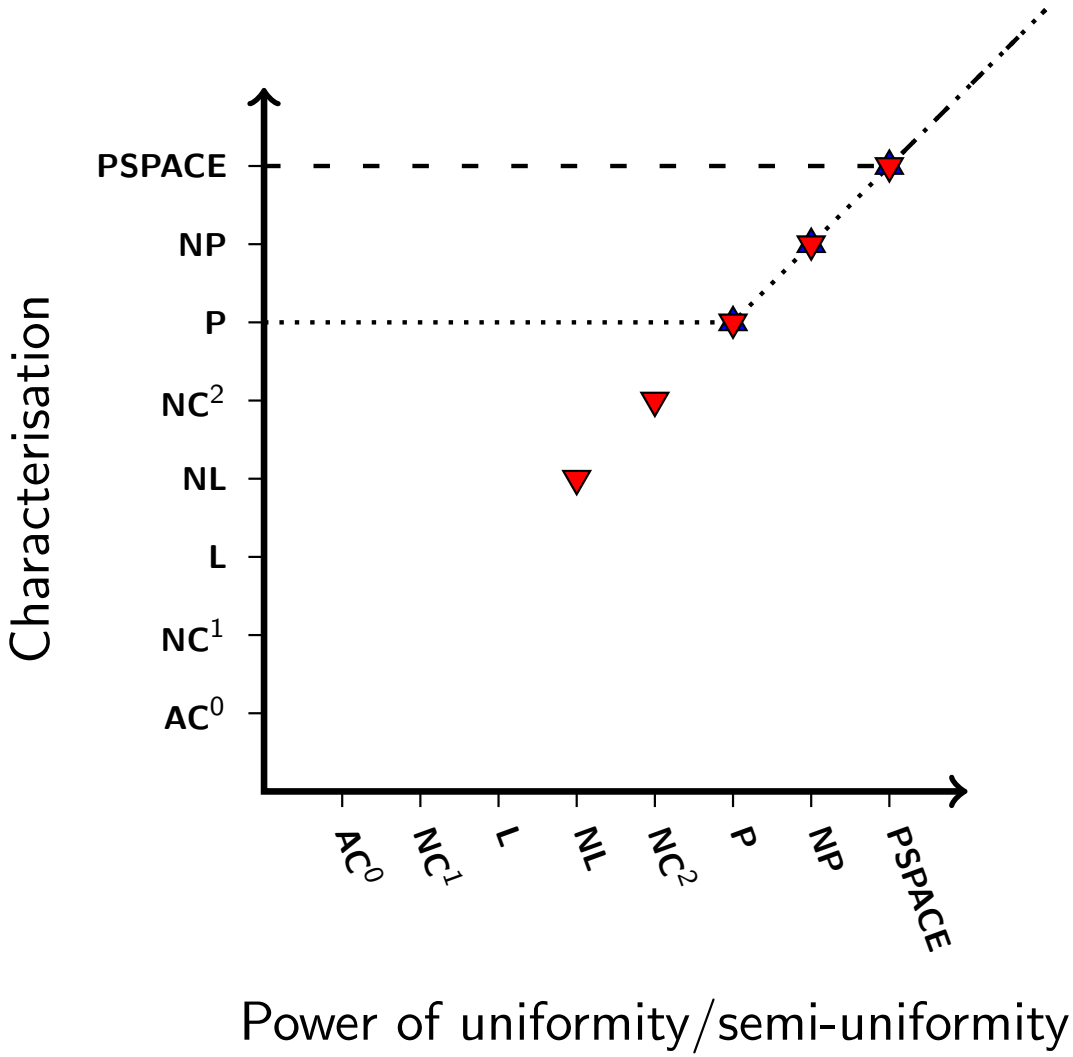
Semi-uniformity vs uniformity



Uniform: ▲

Semi-uniform: ▼

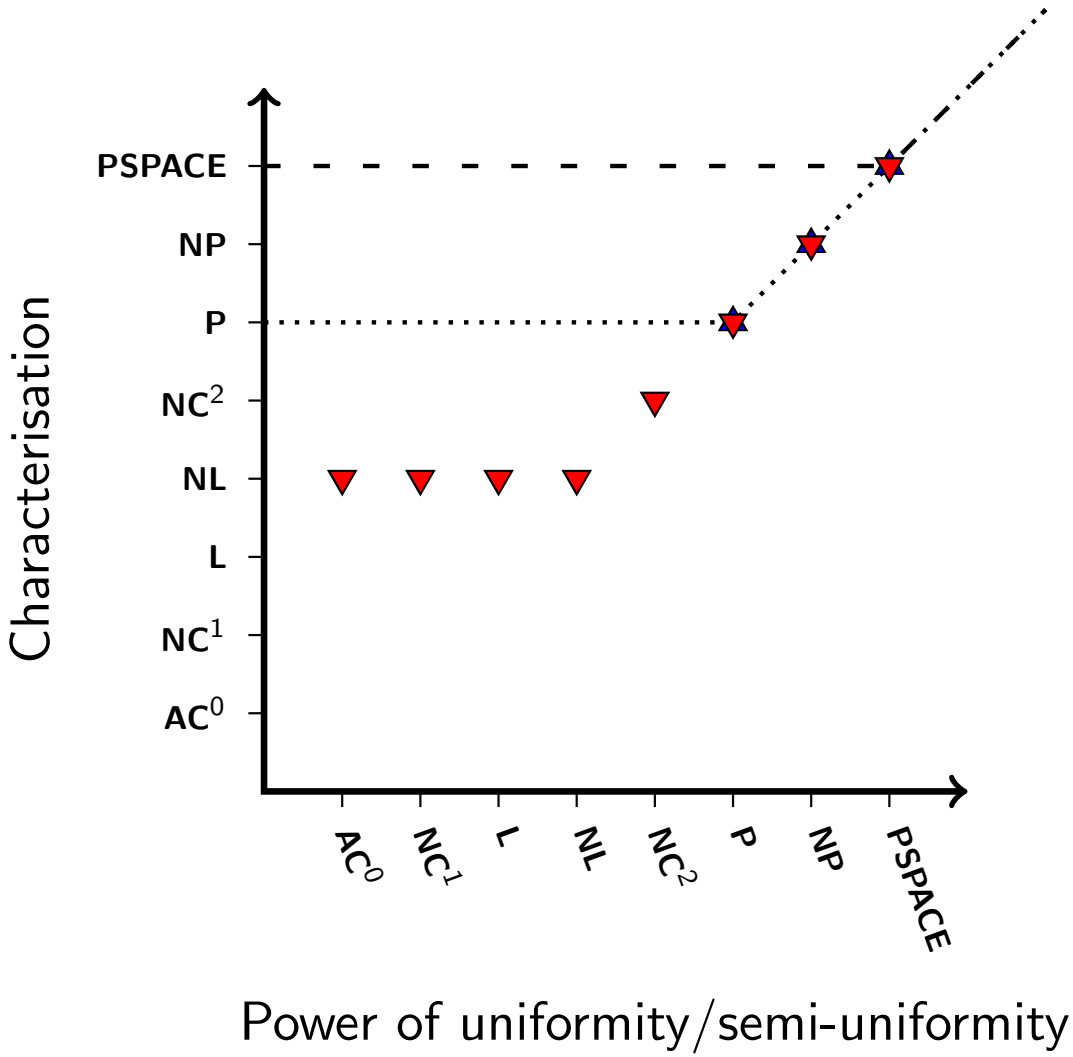
Semi-uniformity vs uniformity



Uniform: ▲

Semi-uniform: ▼

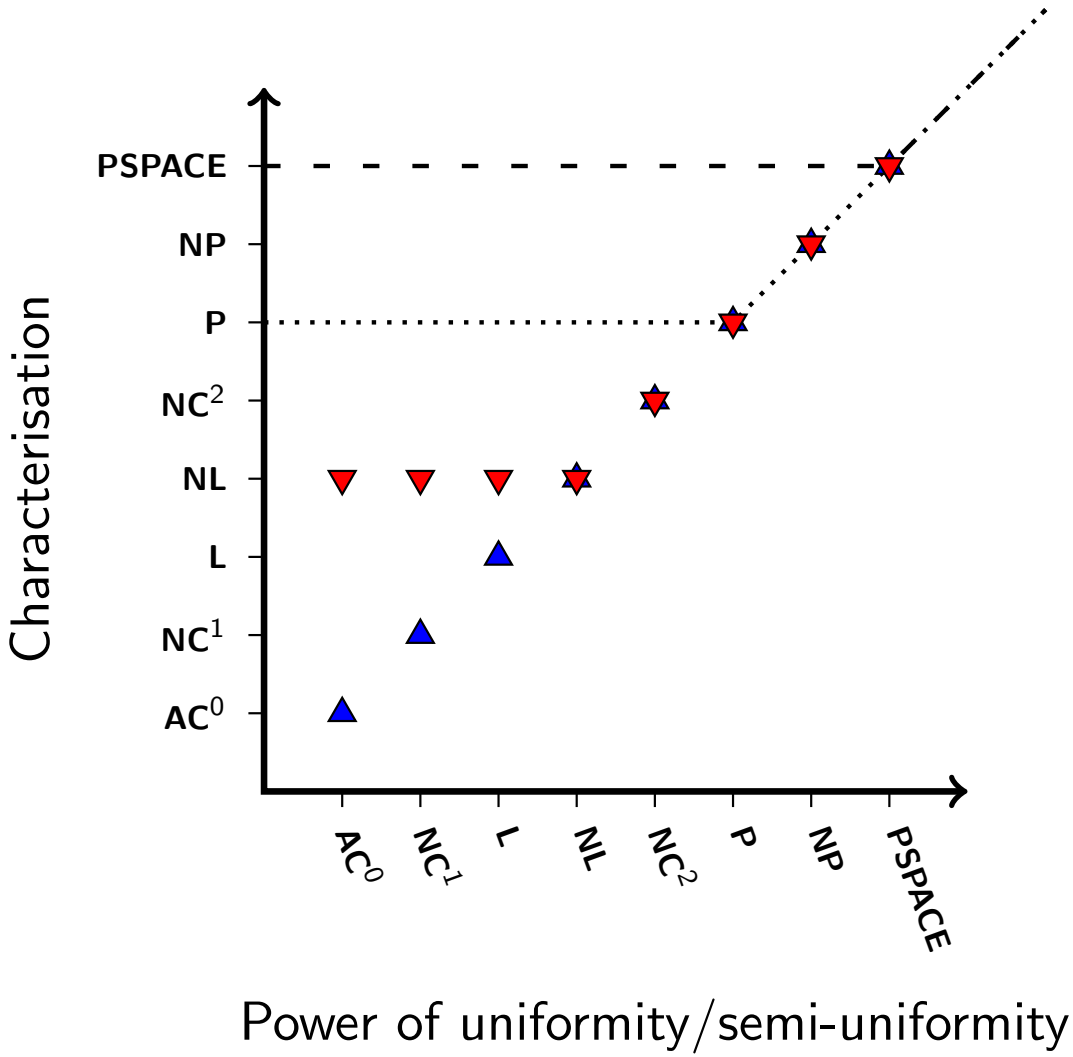
Semi-uniformity vs uniformity



Uniform: ▲

Semi-uniform: ▼

Semi-uniformity vs uniformity



Uniform: ▲

Semi-uniform: ▼

What did we do?

- Usually: **P** uniformity
- Using tighter uniformity conditions we explore the "Black Hole"
- Found a bunch of results that really clarify what is going on with some of these models (exp. wasteful)
- Uniform and Semi-Uniform families are different
- Validates our original intuition, and solves Paŭn's open problem

Conclusions

- Nature-inspired computing resources come in a number of guises
- However, these are amenable to analysis using ideas from older, more well-known, resources
- Nevertheless, we can find new questions and new directions
- One measure of success is what we can give back to older computing models