# Complexity of Machine Learning and Landscapes

Jim Halverson
Northeastern University

ICTP - Machine Learning Landscape,
December 2018

Based on 1809.08279
with Fabian Ruehle

**see also:** 2006 work of [Douglas, Denef], 2010 work of [Cvetic, Garcia-Etxebarria, JH]

# Question 1:

## Why should string theorists care about computational complexity?

# Punchline:

# Punchline:

Difficulties that we run into in landscapes are not only due to exponentially large sizes, which take exponential time to process by nature of their size.

There are also due to the existence of hard problems, which take exponential time to solve because of their complexity.

Progress requires dealing with both.

# Complexity: why consider?

**proxy for now:** “hardness” of computation can be made precise.

# Complexity: why consider?

**proxy for now:** “hardness” of computation can be made precise.

- Practical issues: we have goals, and run into bottlenecks!
  Is it because we’re not that sophisicated, or
  is there a fundamental complexity obstruction?

# Complexity: why consider?

**proxy for now:** “hardness” of computation can be made precise.

- Practical issues: we have goals, and run into bottlenecks!
  Is it because we’re not that sophisicated, or
  is there a fundamental complexity obstruction?

- Critical observables could be computationally hard.
  e.g. Bousso-Polchinski and ADK CCs.

# Complexity: why consider?

**proxy for now:** "hardness" of computation can be made precise.

- Practical issues: we have goals, and run into bottlenecks!
  Is it because we're not that sophisicated, or
  is there a fundamental complexity obstruction?

- Critical observables could be computationally hard.
  e.g. Bousso-Polchinski and ADK CCs. **[Denef, Douglas]**

- Critical observables could correlate with hard problems.

# Complexity: why consider?

**proxy for now:** "hardness" of computation can be made precise.

- Practical issues: we have goals, and run into bottlenecks!
  Is it because we're not that sophisicated, or
  is there a fundamental complexity obstruction?

- Critical observables could be computationally hard.
  e.g. Bousso-Polchinski and ADK CCs. **[Denef, Douglas]**

- Critical observables could correlate with hard problems.

- If physical system implicitly solves a problem, then hardness results can affects its dynamics. **e.g. protein folding: [Wolynes]**
  **e.g. strings: [Denef, Douglas, Greene, Zukowski], also [J.H., Ruehle]**

# Complexity: why consider?

**proxy for now:** “hardness” of computation can be made precise.

- Practical issues: we have goals, and run into bottlenecks!
  Is it because we’re not that sophisicated, or
  is there a fundamental complexity obstruction?

- Critical observables could be computationally hard.
  e.g. Bousso-Polchinski and ADK CCs. **[Denef, Douglas]**

- Critical observables could correlate with hard problems.

- If physical system implicitly solves a problem, then hardness results can affects its dynamics. **e.g. protein folding: [Wolynes]**
  **e.g. strings: [Denef, Douglas, Greene, Zukowski], also [J.H., Ruehle]**

- Undecidability: decision prob —> diophantine
  —> landscape algorithmically patchy. **[Cvetic, Garcia-Etxebarria, J.H.]**

# Outline: 5 Questions

# Outline: 5 Questions

- Why should string theorists care about complexity?

# Outline: 5 Questions

- Why should string theorists care about complexity?
- What is computational complexity?

# Outline: 5 Questions

- Why should string theorists care about complexity?
- What is computational complexity?
- What is the complexity of vacua in landscapes?

# Outline: 5 Questions

- Why should string theorists care about complexity?
- What is computational complexity?
- What is the complexity of vacua in landscapes?
- What is the complexity of vacua in the string landscape?

# Outline: 5 Questions

- Why should string theorists care about complexity?
- What is computational complexity?
- What is the complexity of vacua in landscapes?
- What is the complexity of vacua in the string landscape?
- What are potential complexity loopholes and what does it mean for applying ML / AI to landscapes?

# Question 2:
# What is computational complexity?

**Flow:** **Problems —> P vs. NP —> Polytime Reduction —> Hardest NP Probs —> Optimization vs. Decision —> Example**

# Problems

- a PROBLEM    F: I —> B    maps instances to outputs.

- a DECISION PROBLEM has B = {yes, no}.

- Example: a clique of an undirected graph G is a set of vertices that are all connected to one another.

  **CLIQUE:** Given an undirected graph $G$ and $n \in \mathbb{Z}$, does $G$ have an $n$-clique?

  where I = S x Z, and S the set of undirected graphs.

- an algorithm that **computes** F always returns an output.

- **polytime** algorithms return an output in time bounded by polynomial in the input size. otherwise, will say **exponential time**.

# P vs. NP

# P vs. NP

- are some classes of problems harder than others?

# P vs. NP

- are some classes of problems harder than others?

- **P:** class of problems with polytime solution algorithms.
  trivial example: multiplication
  non-trivial example: PRIMES   (see “PRIMES is in P”)

# P vs. NP

- are some classes of problems harder than others?

- **P:** class of problems with polytime solution algorithms.
  trivial example: multiplication
  non-trivial example: PRIMES (see “PRIMES is in P”)

- **NP:** class of problems with polytime verifiers. **NP contains P.**
  example: sudoku (can check proposed solutions quickly)

# P vs. NP

- are some classes of problems harder than others?

- **P:** class of problems with polytime solution algorithms.
  trivial example: multiplication
  non-trivial example: PRIMES (see “PRIMES is in P”)

- **NP:** class of problems with polytime verifiers. **NP contains P.**
  example: sudoku (can check proposed solutions quickly)

- **P vs. NP:**
  is solving problems as hard as verifying proposed solutions?

  i.e., is P = NP? **million dollar problem** (Clay Math)

# P vs. NP

- are some classes of problems harder than others?

- **P:** class of problems with polytime solution algorithms.
  trivial example: multiplication
  non-trivial example: PRIMES (see “PRIMES is in P”)

- **NP:** class of problems with polytime verifiers. **NP contains P.**
  example: sudoku (can check proposed solutions quickly)

- **P vs. NP:**
  is solving problems as hard as verifying proposed solutions?

  i.e., is P = NP? **million dollar problem** (Clay Math)

- problem is open, but consensus is P != NP.

# P vs. NP

- are some classes of problems harder than others?

- **P:** class of problems with polytime solution algorithms.
  trivial example: multiplication
  non-trivial example: PRIMES (see "PRIMES is in P")

- **NP:** class of problems with polytime verifiers. **NP contains P.**
  example: sudoku (can check proposed solutions quickly)

- **P vs. NP:**
  is solving problems as hard as verifying proposed solutions?

  i.e., is P = NP? **million dollar problem** (Clay Math)

- problem is open, but consensus is P != NP.

- is there a notion of the hardest problems in NP?

# Polytime Reduction

# Polytime Reduction

- Consider two problems:

  F: I —> {yes, no}                    G: I’ —> {yes, no}

# Polytime Reduction

- Consider two problems:

  F: I —> {yes, no}    G: I' —> {yes, no}

- We say that there is a **polytime reduction** from F to G if there is a polytime algorithm f: I —> I' such that

  F(x) = yes <—> G(f(x)) = yes.

# Polytime Reduction

- Consider two problems:

  F: I —> {yes, no} G: I' —> {yes, no}

- We say that there is a **polytime reduction** from F to G if there is a polytime algorithm f: I —> I' such that

  F(x) = yes <—> G(f(x)) = yes.

- Colloquially, can use solutions of G to solve F.

# Polytime Reduction

- Consider two problems:

  F: I —> {yes, no}     G: I’ —> {yes, no}

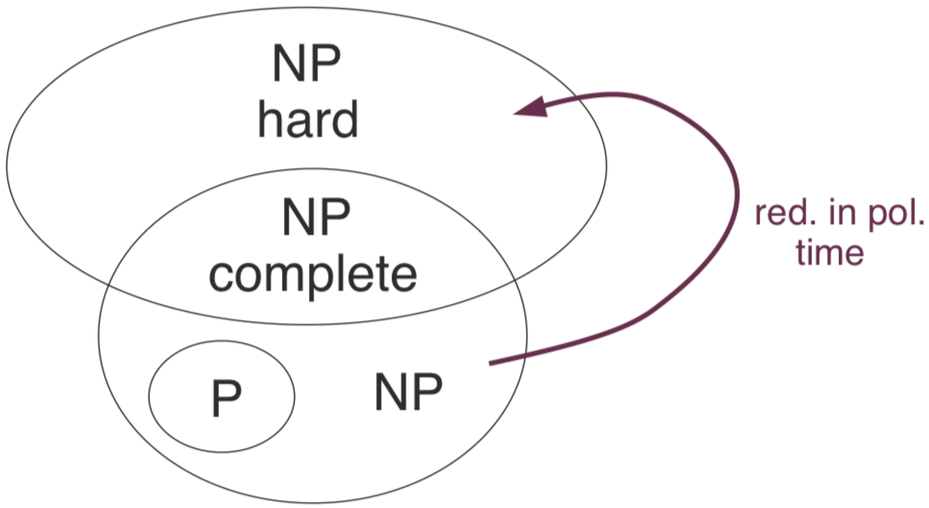
- We say that there is a **polytime reduction** from F to G if there is a polytime algorithm f: I —> I’ such that

  F(x) = yes <—> G(f(x)) = yes.

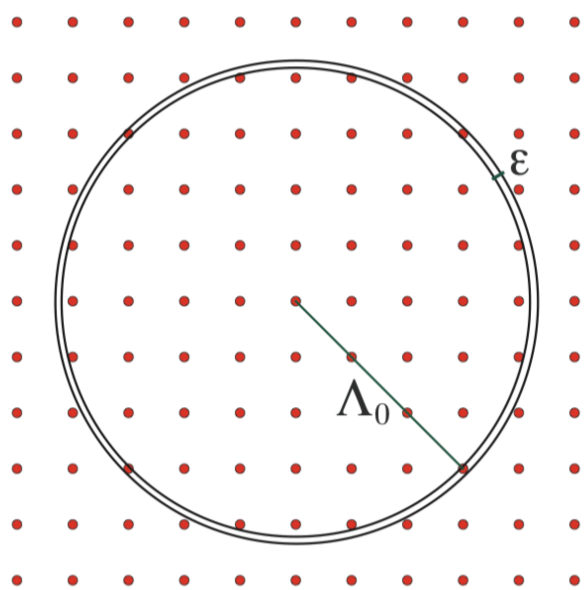
- Colloquially, can use solutions of G to solve F.

- Specifically, if polytime alg for G, then also for F.

# The Hardest NP Problems

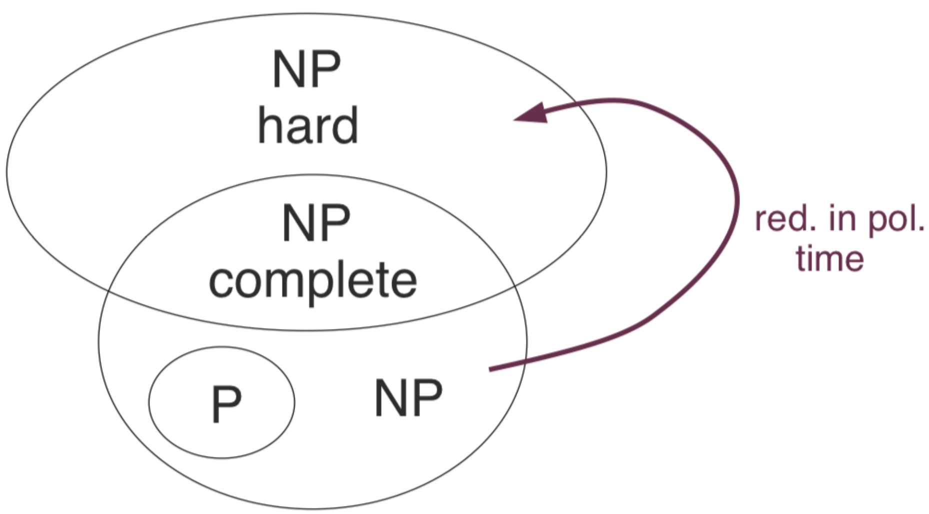



images from:
[Denef, Douglas]

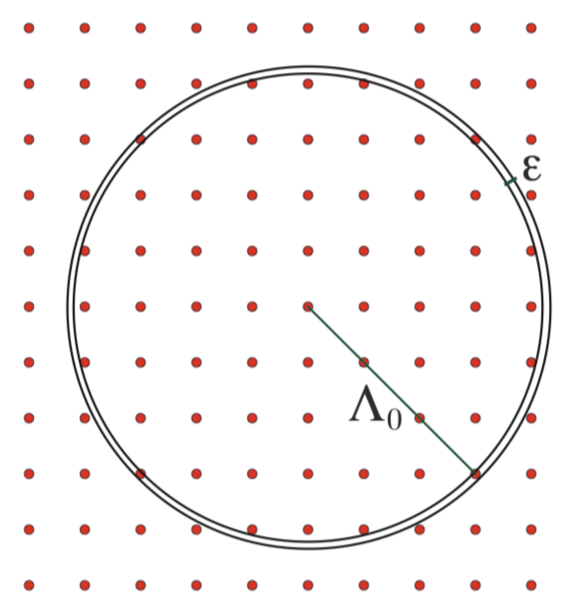


# The Hardest NP Problems

- problem G is **NP-hard** if there exists a polytime reduction to G for every problem in NP.



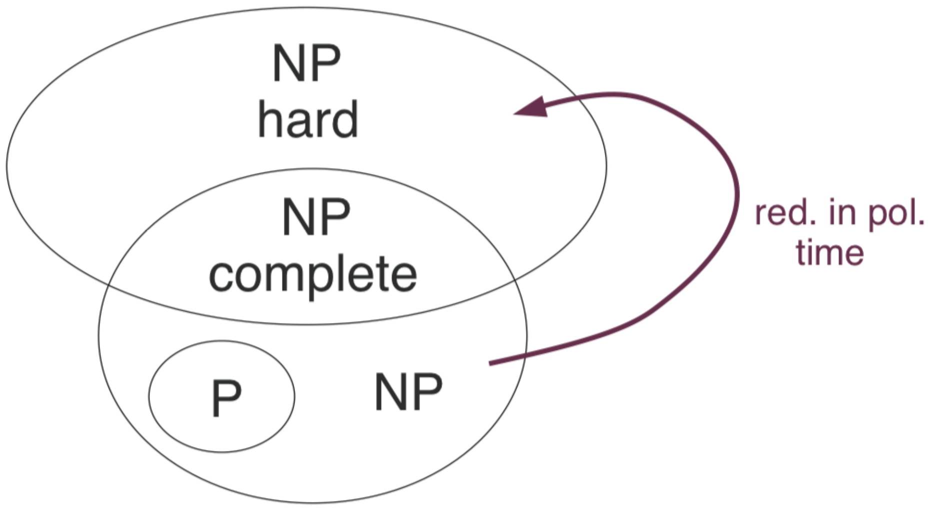

images from:
[Denef, Douglas]



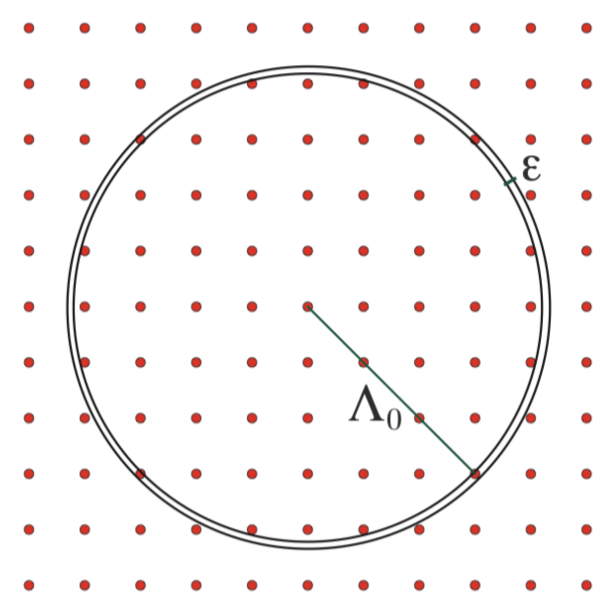
# The Hardest NP Problems

- problem G is **NP-hard** if there exists a polytime reduction to G for every problem in NP.

- practically: solve G, solve every problem in NP.

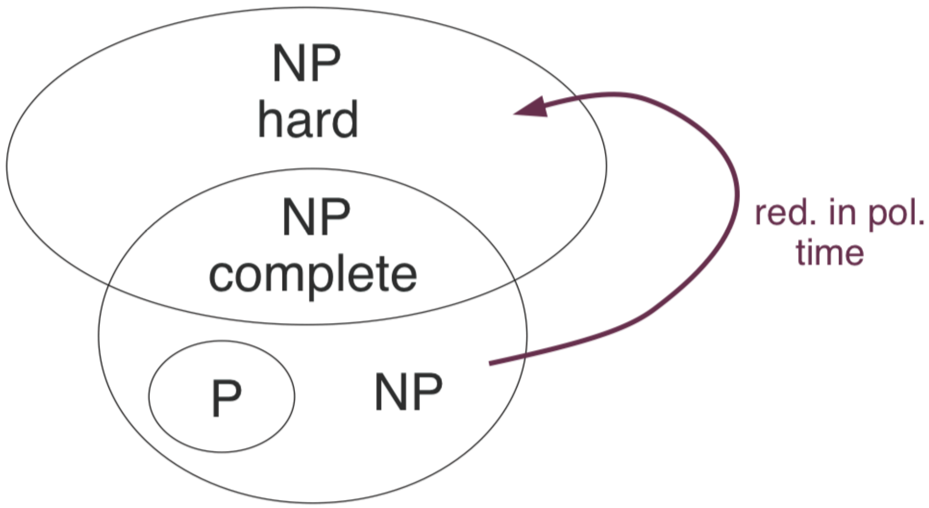



**images from:**
**[Denef, Douglas]**

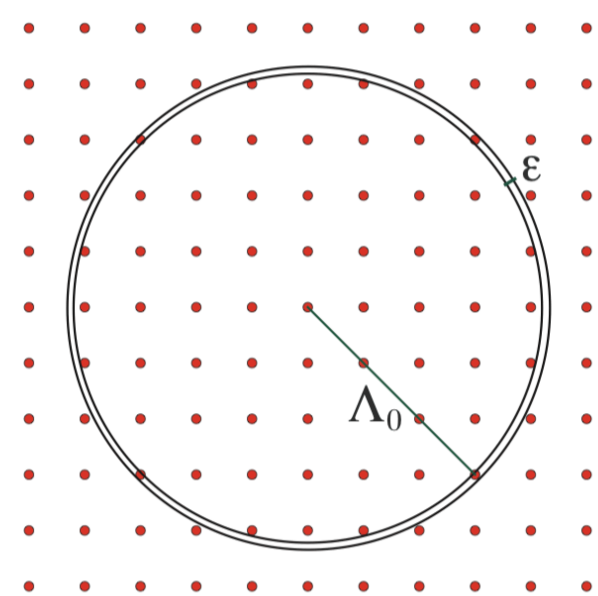


# The Hardest NP Problems

- problem G is **NP-hard** if there exists a polytime reduction to G for every problem in NP.
- practically: solve G, solve every problem in NP.
- find polytime alg. for NP-hard problem? proves P = NP.

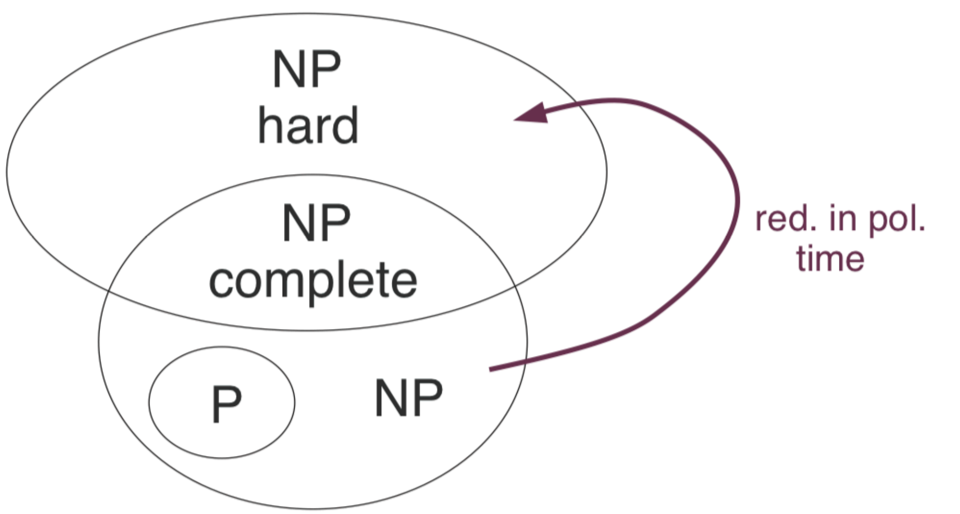



**images from: [Denef, Douglas]**

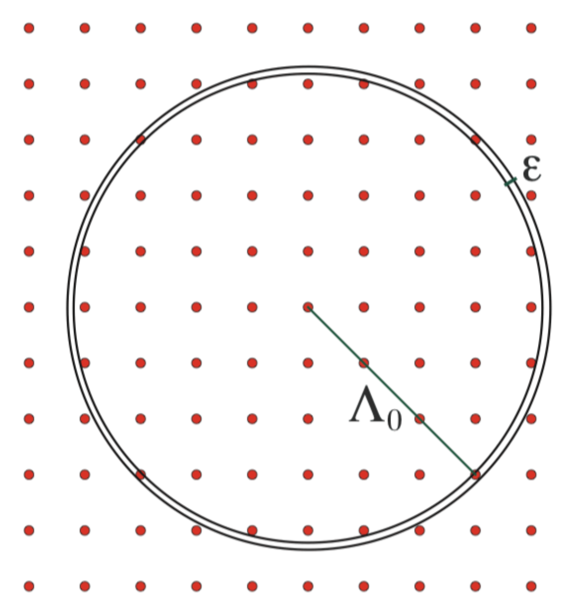


# The Hardest NP Problems

- problem G is **NP-hard** if there exists a polytime reduction to G for every problem in NP.
- practically: solve G, solve every problem in NP.
- find polytime alg. for NP-hard problem? proves P = NP.
- therefore if P != NP, no polytime algorithm! problem takes exponential time, call **hard**.

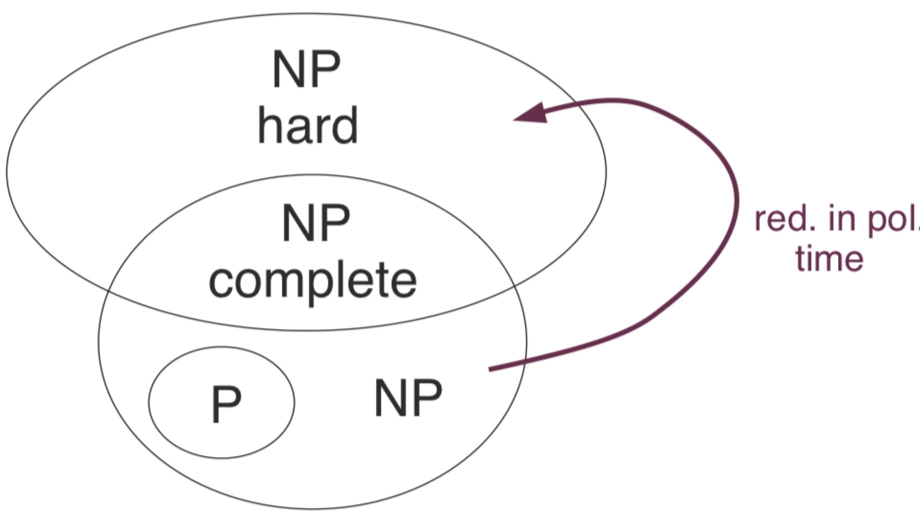



**images from: [Denef, Douglas]**

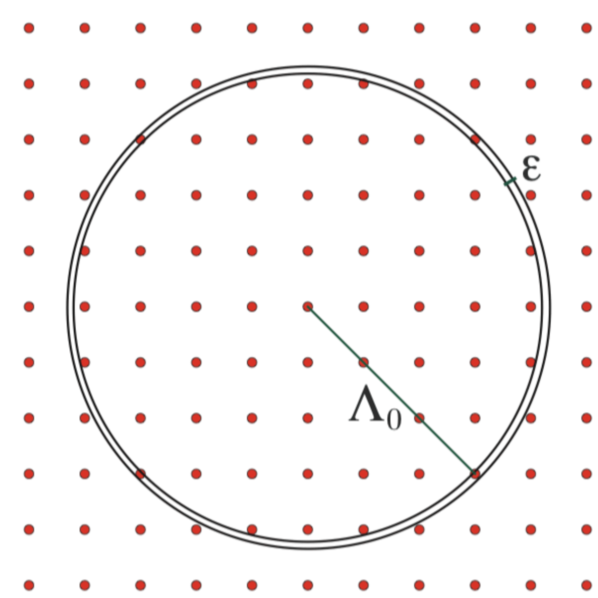


# The Hardest NP Problems

- problem G is **NP-hard** if there exists a polytime reduction to G for every problem in NP.
- practically: solve G, solve every problem in NP.
- find polytime alg. for NP-hard problem? proves P = NP.
- therefore if P != NP, no polytime algorithm! problem takes exponential time, call **hard**.
- an **NP-complete** problem is NP and NP-hard. Examples: SUBSET SUM and KNAPSACK




**images from: [Denef, Douglas]**



# The Hardest NP Problems

- problem G is **NP-hard** if there exists a polytime reduction to G for every problem in NP.

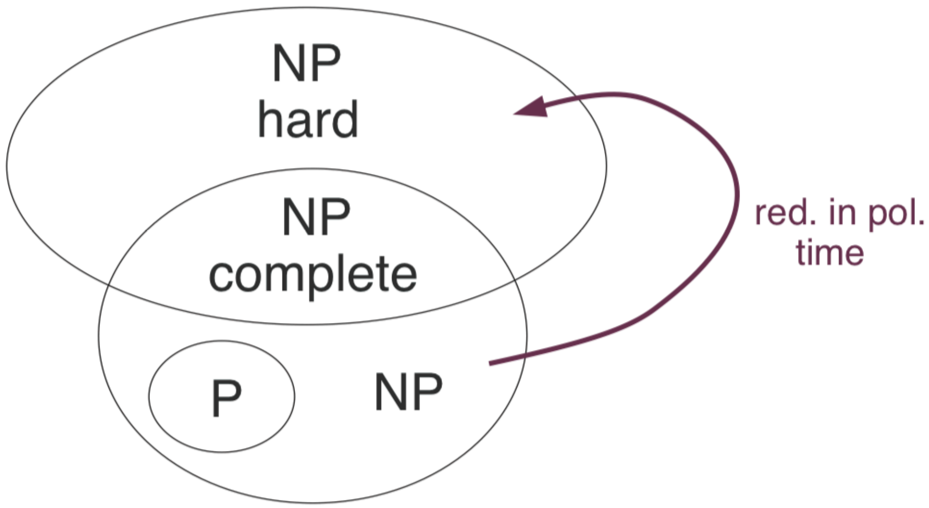
- practically: solve G, solve every problem in NP.

- find polytime alg. for NP-hard problem? proves P = NP.

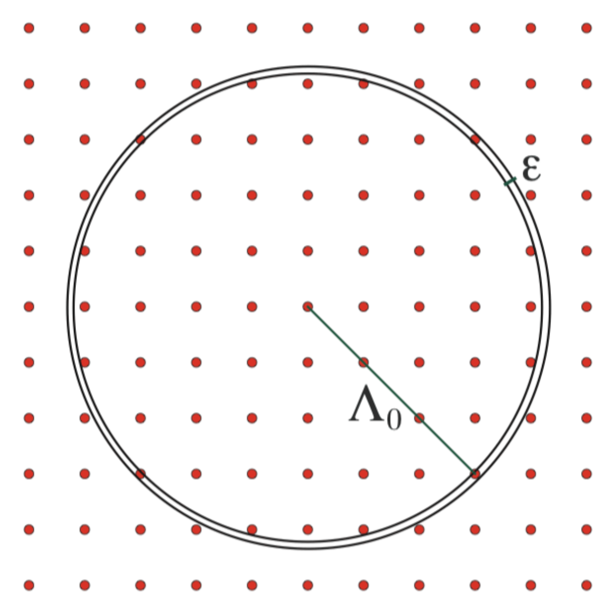
- therefore if P != NP, no polytime algorithm! problem takes exponential time, call **hard**.

- an **NP-complete** problem is NP and NP-hard. Examples: SUBSET SUM and KNAPSACK

- Note: NP-complete problem can have instances in P.




**images from: [Denef, Douglas]**

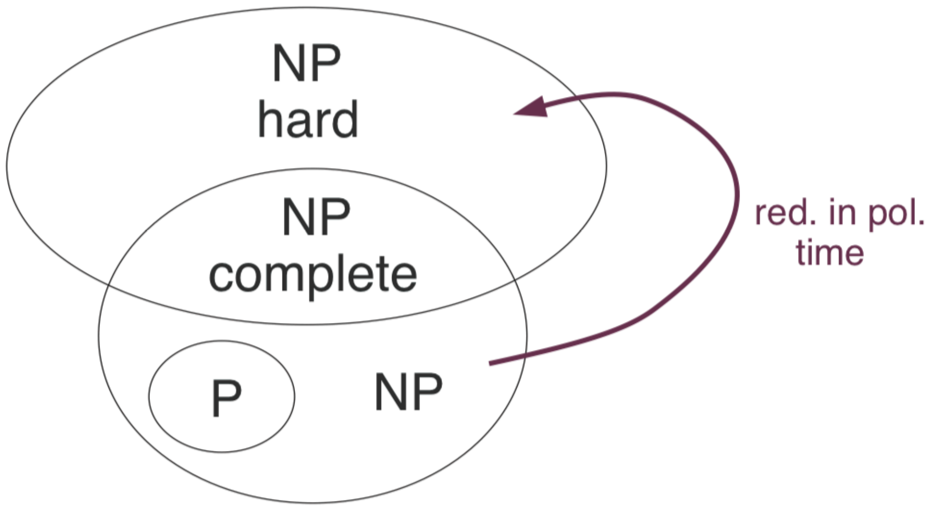


# The Hardest NP Problems

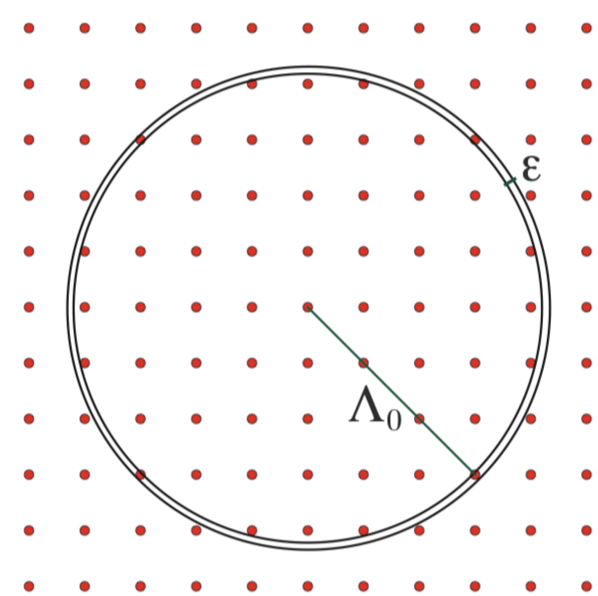
- problem G is **NP-hard** if there exists a polytime reduction to G for every problem in NP.
- practically: solve G, solve every problem in NP.
- find polytime alg. for NP-hard problem? proves P = NP.
- therefore if P != NP, no polytime algorithm! problem takes exponential time, call **hard**.
- an **NP-complete** problem is NP and NP-hard. Examples: SUBSET SUM and KNAPSACK
- Note: NP-complete problem can have instances in P.
- e.g. Bousso-Polchinski and ADK CCs are NP-complete.

  **complexity result: [Denef, Douglas]**

  **tackle with reinforcement learning: [JH, Long, Ruehle]**




**images from: [Denef, Douglas]**



# Optimization vs. Decision

# Optimization vs. Decision

- technically, complexity classes defined with respect to decision problems, i.e. problems with yes / no answers.

# Optimization vs. Decision

- technically, complexity classes defined with respect to decision problems, i.e. problems with yes / no answers.
- optimization: find local or global optimum of h(x).

# Optimization vs. Decision

- technically, complexity classes defined with respect to decision problems, i.e. problems with yes / no answers.
- optimization: find local or global optimum of h(x).
- associated decision problem: is a given point x* a local or global optimum of h(x)?

# Optimization vs. Decision

- technically, complexity classes defined with respect to decision problems, i.e. problems with yes / no answers.
- optimization: find local or global optimum of $h(x)$.
- associated decision problem: is a given point $x^*$ a local or global optimum of $h(x)$?
- optimization problems O are at least as hard as associated decision problems D: solve O, implicitly solve D.

# Optimization: Protein Folding

complexity result: [Unger, Moult] 1993 Early Review: chem-ph/9411008

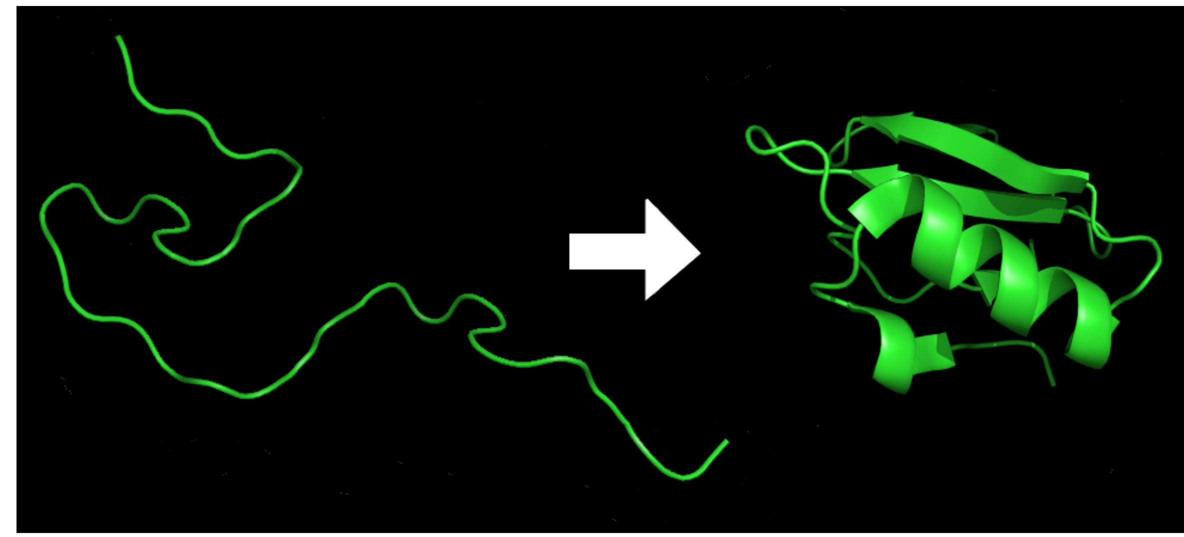


Image: Wikipedia

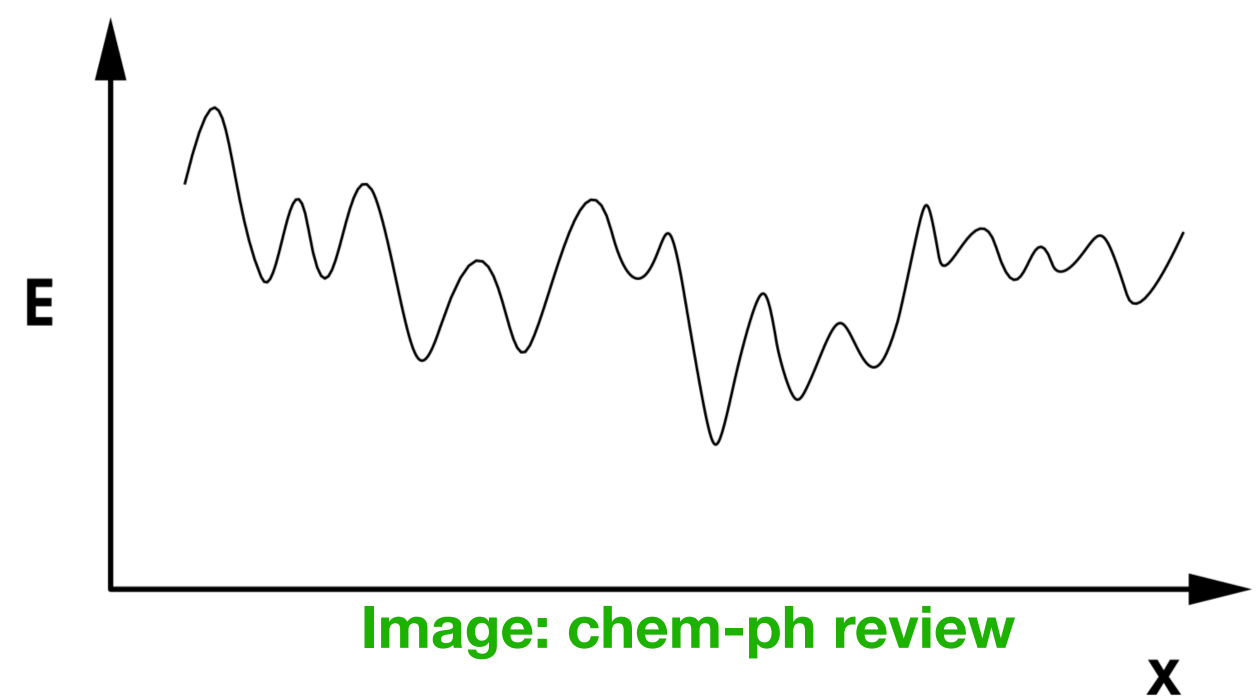



Image: chem-ph review

thanks to P. Wolynes for many references I am still diving into, including his works.

# Optimization: Protein Folding

complexity result: [Unger, Moult] 1993 Early Review: chem-ph/9411008

- Complex system analogous to string landscape.

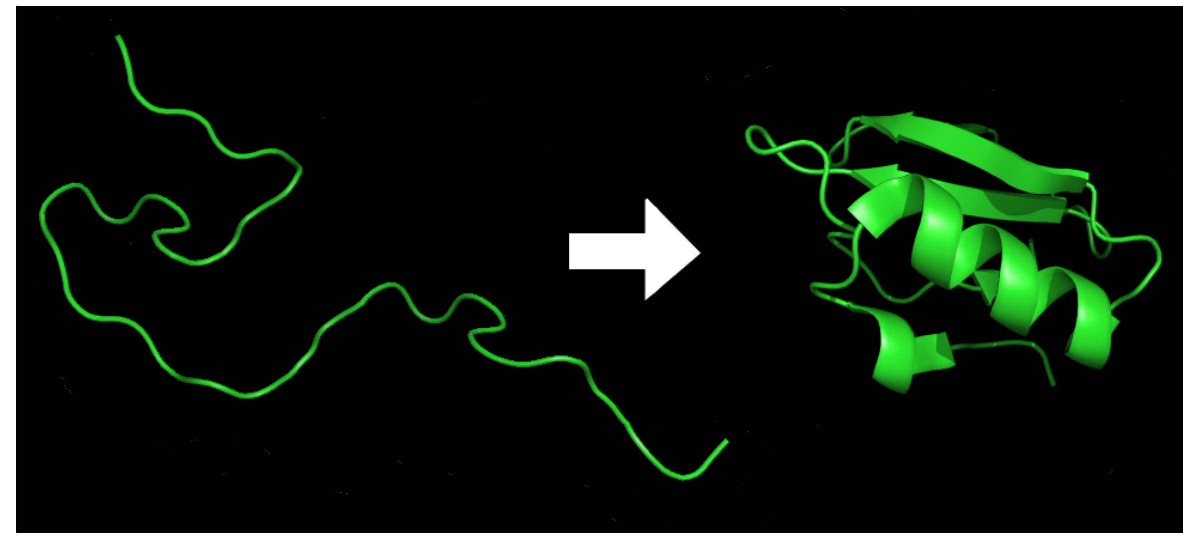


Image: Wikipedia

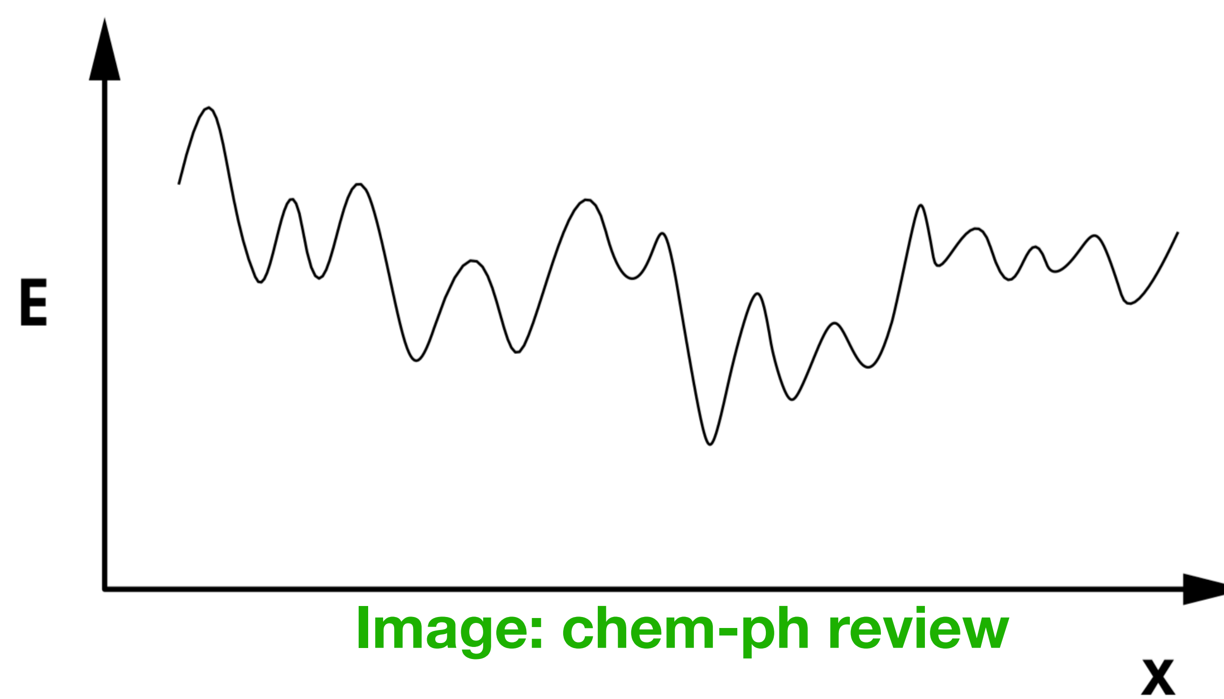



Image: chem-ph review

thanks to P. Wolynes for many references I am still diving into, including his works.

# Optimization: Protein Folding

complexity result: [Unger, Moult] 1993 Early Review: chem-ph/9411008

- Complex system analogous to string landscape.
- Protein folding (find global energy minimum) is NP-complete.

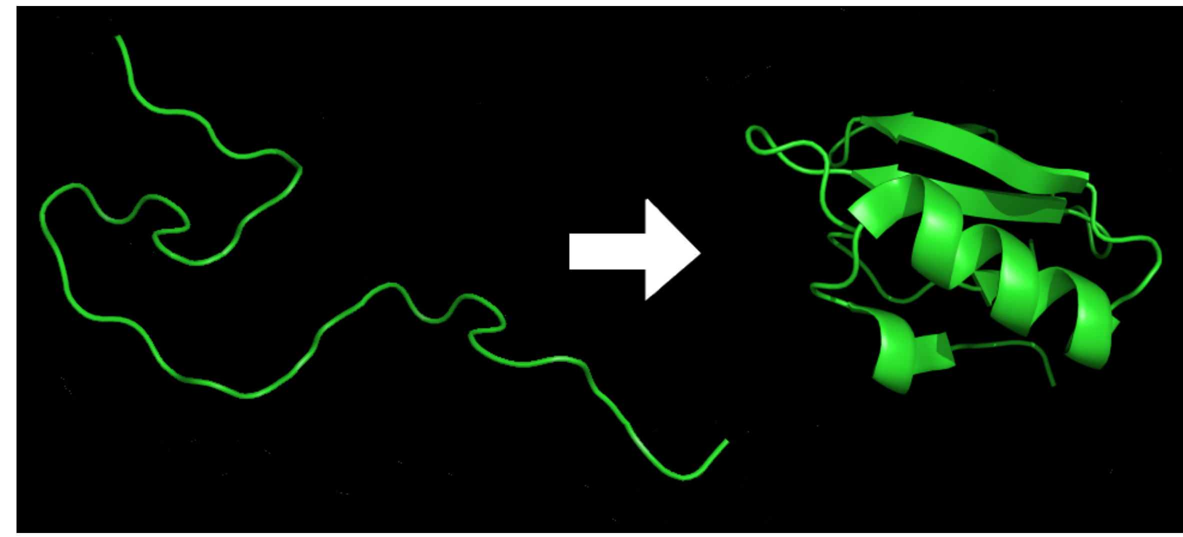


Image: Wikipedia

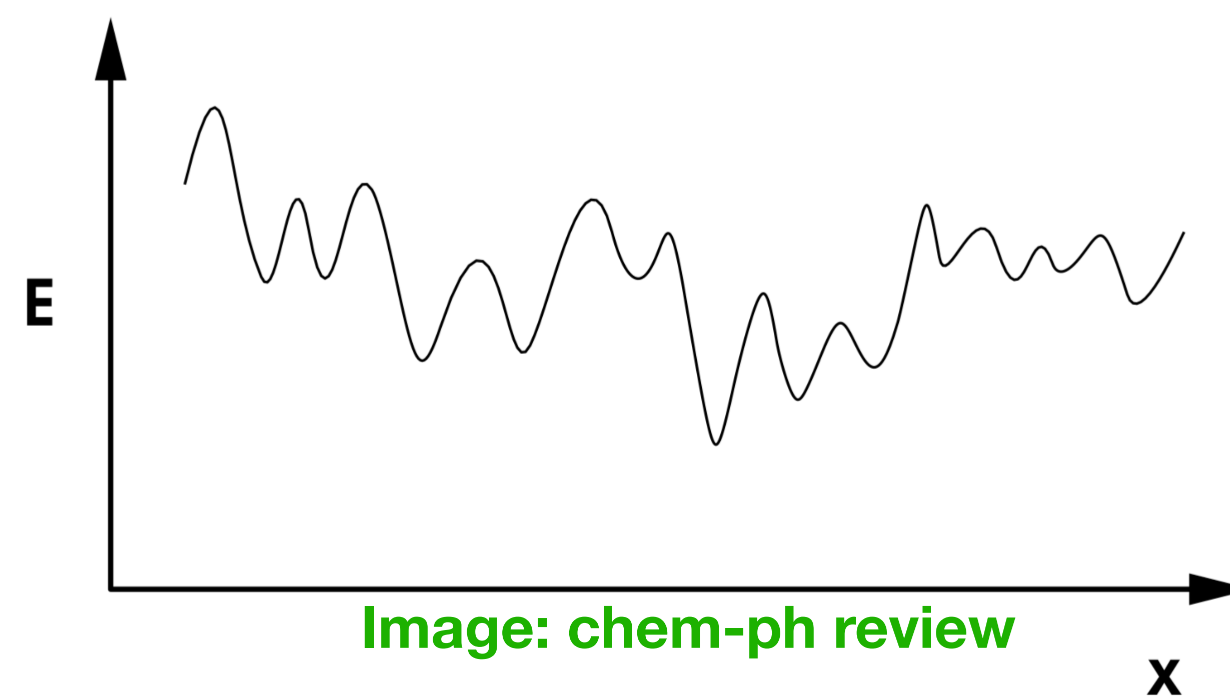



Image: chem-ph review

thanks to P. Wolynes for many references I am still diving into, including his works.

# Optimization: Protein Folding

complexity result: [Unger, Moult] 1993 Early Review: chem-ph/9411008

- Complex system analogous to string landscape.
- Protein folding (find global energy minimum) is NP-complete.
- Affects dynamics: create random stretched protein in lab, see exponential folding time.

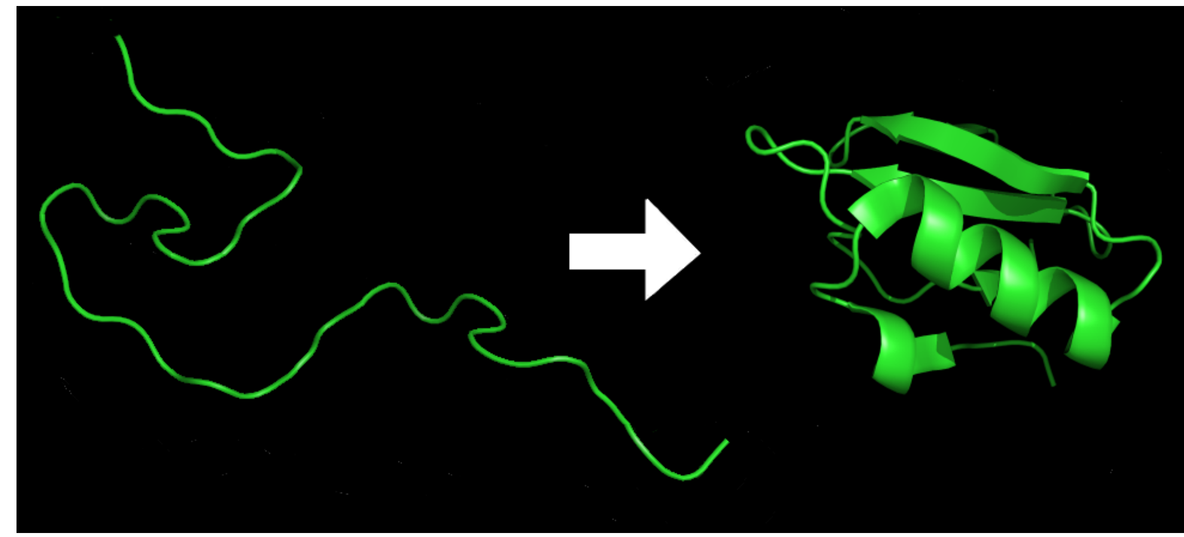


Image: Wikipedia

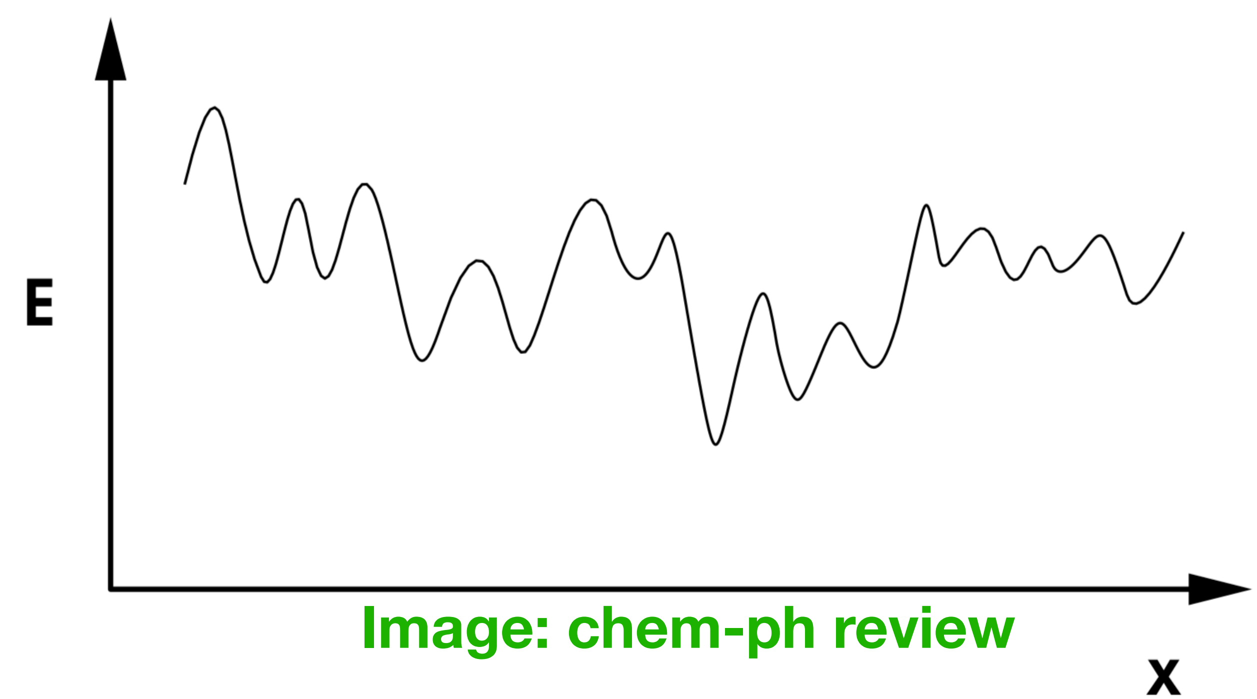



Image: chem-ph review

thanks to P. Wolynes for many references I am still diving into, including his works.

# Optimization: Protein Folding

complexity result: [Unger, Moult] 1993    Early Review: chem-ph/9411008

- Complex system analogous to string landscape.
- Protein folding (find global energy minimum) is NP-complete.
- Affects dynamics: create random stretched protein in lab, see exponential folding time.
- On the other hand: our proteins fold quickly.

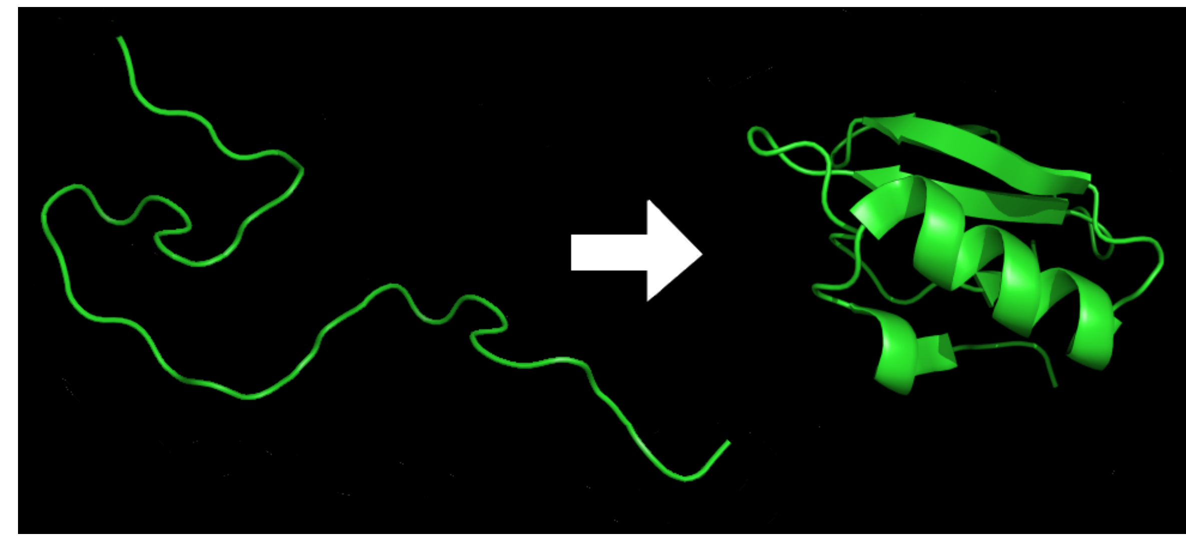


Image: Wikipedia

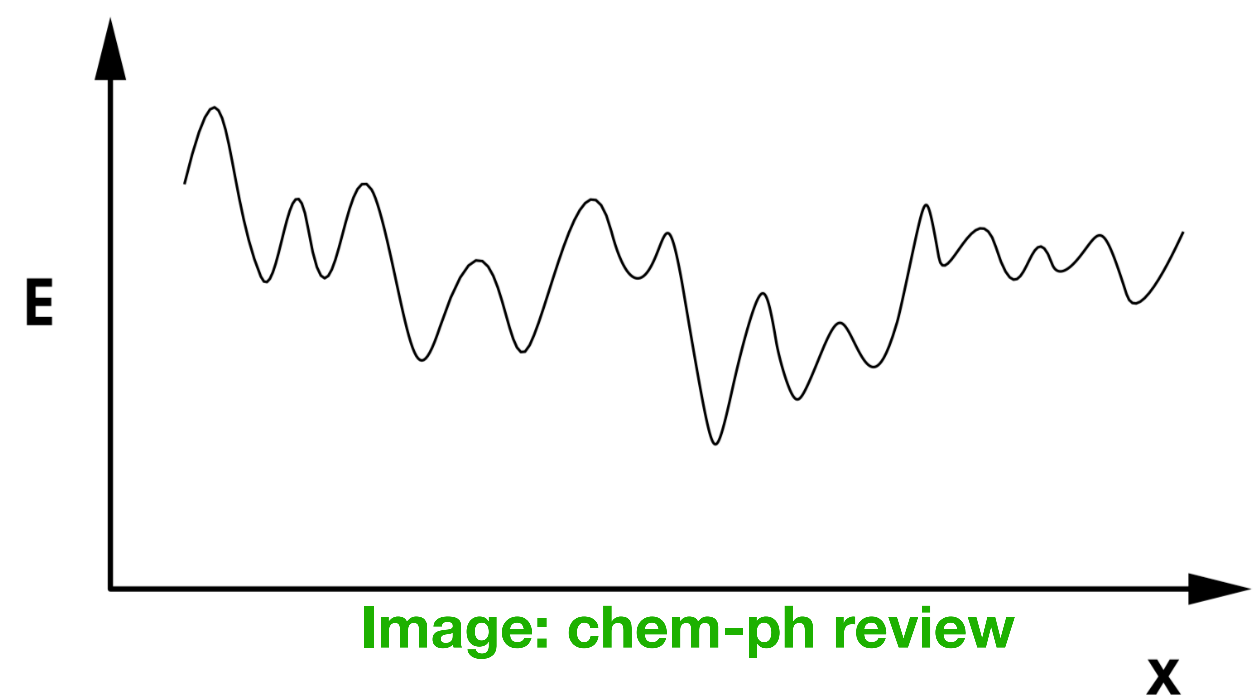



Image: chem-ph review

thanks to P. Wolynes for many references I am still diving into, including his works.

# Optimization: Protein Folding

complexity result: [Unger, Moult] 1993 Early Review: chem-ph/9411008

- Complex system analogous to string landscape.
- Protein folding (find global energy minimum) is NP-complete.
- Affects dynamics: create random stretched protein in lab, see exponential folding time.
- On the other hand:
  our proteins fold quickly.
- Upshot: worst case instances are hard, but evolutionary pressure gives rise to better instances.

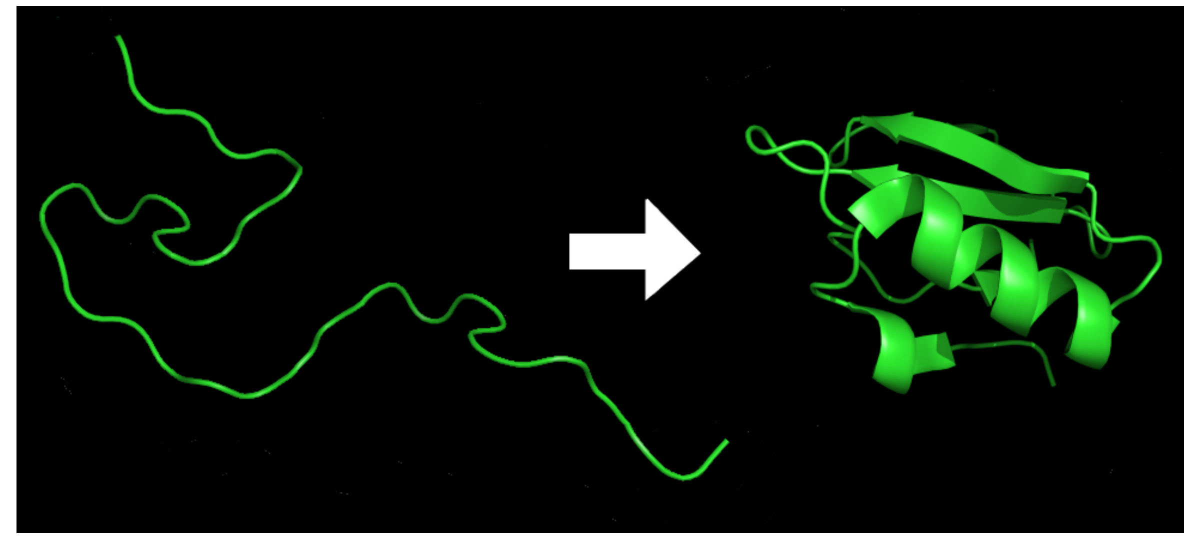


Image: Wikipedia

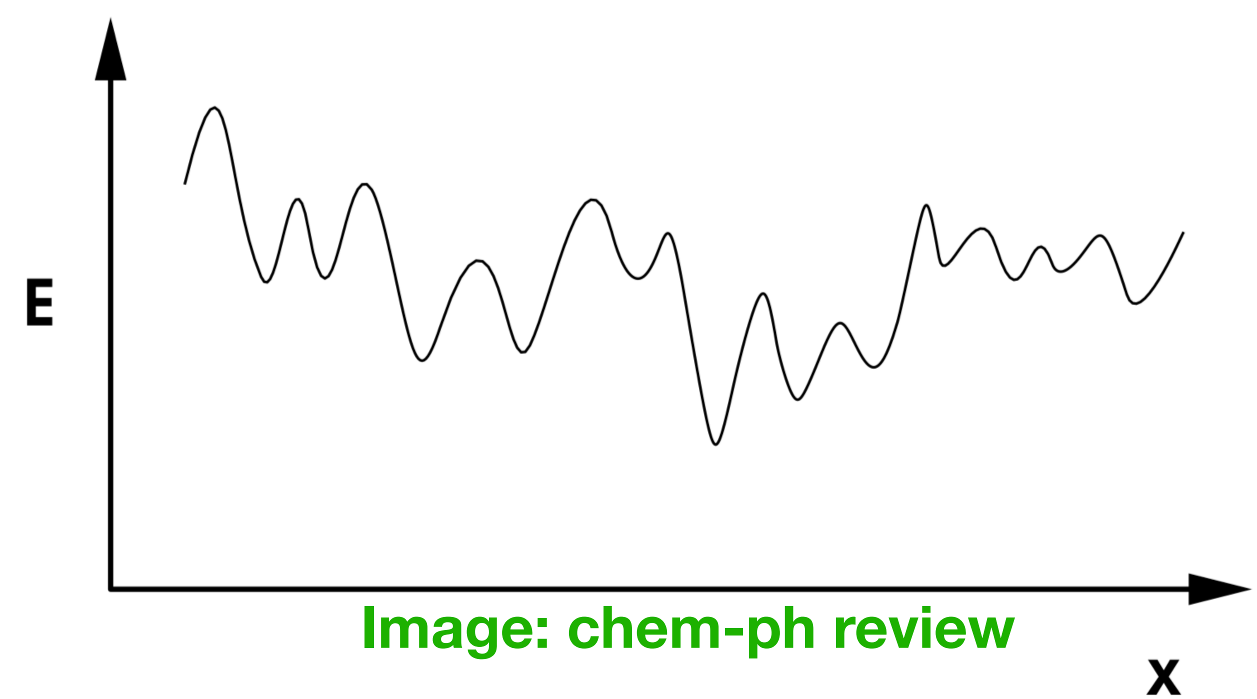



Image: chem-ph review

thanks to P. Wolynes for many references I am still diving into, including his works.

# Question 3:

# What is the complexity of vacua in landscapes?

**Goal:** given $V(\phi)$, is it hard to find stable vacua? metastable vacua? near-vacua?

**Note: training neural nets is effectively the same problem! Complexity carries over.**

# Framing the Problem

# Framing the Problem

- Finding vacua = finding critical point + det. it is a local min.

  Is it hard to find a critical point?
  Is it hard to determine whether it is a local min? Global min?

# Framing the Problem

- Finding vacua = finding critical point + det. it is a local min.

  Is it hard to find a critical point?
  Is it hard to determine whether it is a local min? Global min?

- Maybe we tunnel to the side of a hill that is near a vacuum and inflate from there.

  Is it hard to find a near-vacuum?

# Are critical points hard?

# Are critical points hard?

- Take polynomial $V(\phi)$ (of course, could be worse).

  CRITPOINTS is problem of finding critical points of $V(\phi)$
  requires finding roots of non-trivial system of polynomials. Call POLYROOTS.
  **Claim:** POLYROOTS is NP-hard.

# Are critical points hard?

- Take polynomial V(ϕ)        (of course, could be worse).

  CRITPOINTS is problem of finding critical points of V(ϕ)
  requires finding roots of non-trivial system of polynomials. Call POLYROOTS.
  **Claim:** POLYROOTS is NP-hard.

- Concrete demonstration, as least once. Need SAT.

# Are critical points hard?

- Take polynomial V(ϕ)      (of course, could be worse).

  CRITPOINTS is problem of finding critical points of V(ϕ)
  requires finding roots of non-trivial system of polynomials. Call POLYROOTS.
  **Claim:** POLYROOTS is NP-hard.

- Concrete demonstration, as least once. Need SAT.

- **SAT:** given a CNF-formula ρ, is ρ satisfiable?

  - literal of boolean variable is the variable (x) or its negative (not x).

  - clause: an or of literals.      e.g.,  $x_1 \vee x_3 \vee \neg x_{10}$

  - CNF-formula: “and” of clauses.   e.g.,  $x_1 \wedge (x_2 \vee \neg x_1) \wedge (\neg x_1 \vee \neg x_3)$

  - CNF-formula ρ is satisfiable iff there is an assignment of values to the boolean variables such that ρ evaluates to yes.

# Are critical points hard?

- Take polynomial V(ϕ)        (of course, could be worse).

  CRITPOINTS is problem of finding critical points of V(ϕ)
  requires finding roots of non-trivial system of polynomials. Call POLYROOTS.
  **Claim:** POLYROOTS is NP-hard.

- Concrete demonstration, as least once. Need SAT.

- **SAT:** given a CNF-formula ρ, is ρ satisfiable?

  - literal of boolean variable is the variable (x) or its negative (not x).

  - clause: an or of literals.     e.g.,  $x_1 \vee x_3 \vee \neg x_{10}$

  - CNF-formula: “and” of clauses.  e.g.,  $x_1 \wedge (x_2 \vee \neg x_1) \wedge (\neg x_1 \vee \neg x_3)$

  - CNF-formula ρ is satisfiable iff there is an assignment of values to the boolean variables such that ρ evaluates to yes.

- **Cook-Levin theorem:** SAT is NP-complete. (see any complexity textbook).

# Are critical points hard?

# Are critical points hard?

- **POLYROOTS:** given a system of polynomial equations, is there a non-trivial root?

# Are critical points hard?

- **POLYROOTS:** given a system of polynomial equations, is there a non-trivial root?

- wish to obtain polytime reduction SAT —> POLYROOTS.

  for each instance of SAT, requires constructing instance of POLYROOTS such that non-trivial roots exist iff satisfiable.

# Are critical points hard?

- **POLYROOTS:** given a system of polynomial equations, is there a non-trivial root?

- wish to obtain polytime reduction SAT —> POLYROOTS.

  for each instance of SAT, requires constructing instance of POLYROOTS such that non-trivial roots exist iff satisfiable.

- Form system S of polynomial equations

  - for each boolean $x_i$, add $x_i(1-x_i)$ to S.

  - associate polynomial p(l) to each literal l via:

    $$p(l = x) = (1 - x). \qquad p(l = \neg x) = x$$

  - to a clause $C = l_1 \vee \cdots \vee l_n$ associate $\tilde{p}(C) = \prod_{l \in C} p(l)$

  - for each clause C in the CNF-formula, add $\tilde{p}(C)$ to S

# Are critical points hard?

- **POLYROOTS:** given a system of polynomial equations, is there a non-trivial root?
- wish to obtain polytime reduction SAT —> POLYROOTS.

  for each instance of SAT, requires constructing instance of POLYROOTS such that non-trivial roots exist iff satisfiable.

- Form system S of polynomial equations
  - for each boolean $x_i$, add $x_i(1-x_i)$ to S.
  - associate polynomial p(l) to each literal l via:

    $$p(l = x) = (1 - x) \qquad p(l = \neg x) = x$$

  - to a clause $C = l_1 \vee \cdots \vee l_n$ associate $\tilde{p}(C) = \prod_{l \in C} p(l)$
  - for each clause C in the CNF-formula, add $\tilde{p}(C)$ to S
- **Note:** S has a non-trivial root iff the CNF-formula is satisfiable. **POLYROOTS is NP-hard.**

# Critical Points are Hard

# Critical Points are Hard

- Reduce hard POLYROOTS instance with $\{f_i(\phi)=0\}$ set to CRITPOINTS instance with $V(\chi,\phi) = \chi_i\ f_i^2$

# Critical Points are Hard

- Reduce hard POLYROOTS instance with $\{f_i(\phi)=0\}$ set to CRITPOINTS instance with $V(\chi,\phi) = \chi_i\ f_i^2$

- h has critical points iff POLYROOTS instance has solution.

# Critical Points are Hard

- Reduce hard POLYROOTS instance with $\{f_i(\phi)=0\}$ set to CRITPOINTS instance with $V(\chi,\phi) = \chi_i f_i^2$
- h has critical points iff POLYROOTS instance has solution.
- Result: via reduction SAT —> POLYROOTS —> CRITPOINTS,

**CRITPOINTS is NP-hard.**

# Metastable Vacua

**MSVAC:** Find a local minimum of $V(\phi)$, such that $l_i \leq \phi_i \leq u_i$,

where $\phi = (\phi_1, \ldots, \phi_n) \in \mathbb{R}^n$ and $l_i, u_i \in \mathbb{R}$ are used to give so-called box constraints. Here the box constraints encode the fact that the EFT has a regime of validity that bounds the scalar fields values.

# Metastable Vacua

**MSVAC:** Find a local minimum of $V(\phi)$, such that $l_i \leq \phi_i \leq u_i$,

where $\phi = (\phi_1, \ldots, \phi_n) \in \mathbb{R}^n$ and $l_i, u_i \in \mathbb{R}$ are used to give so-called box constraints. Here the box constraints encode the fact that the EFT has a regime of validity that bounds the scalar fields values.

- **decision version:** (is crit point Φ* a local minimum?) **Result: co-NP-hard.**

  - required modification of local quadratic programming to quartic case, to put difficulty in interior of box for EFT. only difficult for positive semi-definite Hessian.

  - one proof critically utilizes reduction from complement of MAX-CLIQUE.
  See appendix / extra slides for proof sketch.

# Metastable Vacua

**MSVAC:** Find a local minimum of $V(\phi)$, such that $l_i \leq \phi_i \leq u_i$,

where $\phi = (\phi_1, \ldots, \phi_n) \in \mathbb{R}^n$ and $l_i, u_i \in \mathbb{R}$ are used to give so-called box constraints. Here the box constraints encode the fact that the EFT has a regime of validity that bounds the scalar fields values.

- **decision version:** (is crit point $\Phi^*$ a local minimum?) **Result: co-NP-hard.**

  - required modification of local quadratic programming to quartic case, to put difficulty in interior of box for EFT. only difficult for positive semi-definite Hessian.

  - one proof critically utilizes reduction from complement of MAX-CLIQUE. See appendix / extra slides for proof sketch.

- **optimization version:** (find a local minimum)
  must find critical point, which is NP-hard, then solve decision problem reg. loc min.

# Metastable Vacua

**MSVAC:** Find a local minimum of $V(\phi)$, such that $l_i \leq \phi_i \leq u_i$,

where $\phi = (\phi_1, \ldots, \phi_n) \in \mathbb{R}^n$ and $l_i, u_i \in \mathbb{R}$ are used to give so-called box constraints. Here the box constraints encode the fact that the EFT has a regime of validity that bounds the scalar fields values.

- **decision version:** (is crit point $\Phi^*$ a local minimum?) **Result: co-NP-hard.**

  - required modification of local quadratic programming to quartic case, to put difficulty in interior of box for EFT. only difficult for positive semi-definite Hessian.

  - one proof critically utilizes reduction from complement of MAX-CLIQUE. See appendix / extra slides for proof sketch.

- **optimization version:** (find a local minimum)
  must find critical point, which is NP-hard, then solve decision problem reg. loc min.

- **special case:** only strict saddles, SGD (as in ML) finds minima in P.

# Stable Vacua

**SVAC:** Find a global minimum of $V(\phi)$, such that $l_i \leq \phi_i \leq u_i$,

- global minimum is hard because local minimum is already hard!
- difficulty of global minimization is well-known,
  e.g. global quadratic programming or protein folding.
- it was the fact that local minima is hard that we found very surprising.

# Near-Vacua

- **Definition:** $x^*$ is an **ε-approximate local minimum** of a continuous function f: U —> R if there is an open set N in U containing $x^*$ such that $f(x^*) <= f(x) + \varepsilon\, |x-x^*|$ for all x in N.

- Idea: this is a *near*-vacuum. Define associated problem:

**NEAR-VAC:** Given a scalar potential $V(\phi)$ and $\epsilon > 0$, find a point $\phi^*$ that is an $\epsilon$-vacuum.

- Fast algorithm of Vavasis: $|f(x) - f(y)| \leq M|x - y|$

**Theorem:** *Let $f : U \to \mathbb{R}$ be a $C^1$ function whose gradient satisfies a Lipschitz condition with bound $M$. Then an $\epsilon$-minimum can be found with at most $4n(M/\epsilon)^2$ function and gradient evaluations.*

- NEAR-VAC is in P.

# Question 4:

## What is the complexity of vacua in the string landscape?

**Goal:** is it hard to determine $V(\Phi)$ in string theory?

# Framing the Problem

- Hard to find both stable and metastable vacua, given V(ϕ).
- Computing V(ϕ) subject of much string research.
    - IIB: KKLT and LVS.
      **[Kachru, Kallosh, Linde, Trivedi], [Balasubranian, Berglund, Conlon, Quevedo]**
    - e.g. infinite # of M2-instantons on certain G2-manifolds.
      **[Braun, Del Zotto, JH, Larfors, Morrison, Schafer-Nameki]**
- **Q:** is it also hard to compute V(ϕ)?
- **goal:** show string V(ϕ) contributions req. solving instances of NP-complete probs.      (open up Garey and Johnson!)

# Rural Postman

Given a graph $G$, lengths $l(e) \in \mathbb{Z}^+$ for all edges $e \in E$, a subset $E' \subset E$, and a bound $B \in \mathbb{Z}^+$, we have the problem:

**RURAL POSTMAN:** Is there a circuit (closed loop) in $G$ that includes each edge in $E'$ and that has total length no more than $B$?

This problem is NP-complete in general, and also if edge lengths are set to one or if the graph is directed.

# Rural Postman

Given a graph $G$, lengths $l(e) \in \mathbb{Z}^+$ for all edges $e \in E$, a subset $E' \subset E$, and a bound $B \in \mathbb{Z}^+$, we have the problem:

| | |
|---|---|
| **RURAL POSTMAN:** | Is there a circuit (closed loop) in $G$ that includes each edge in $E'$ and that has total length no more than $B$? |

This problem is NP-complete in general, and also if edge lengths are set to one or if the graph is directed.

**Physical Realization:** given a quiver gauge theory, does there exist a scalar GIO O that couples a fixed subset E' of fields to one another, such that dim(O) <= B?

# Integer Programming

Now we turn to INTEGER PROGRAMMING (INT-PROG). Given $(A \in \mathbb{Z}^{m\times n}, b \in \mathbb{Z}^n, c \in \mathbb{Z}^n, B \in \mathbb{Z})$,

**INT-PROG:** Is there a $y \in \mathbb{Z}^n$ such that $Ay \leq b$ and $c \cdot y \leq B$?

This is asking whether or not there is an integral point that satisfies systems of hyperplane constraints, where the latter may cut out cones or polyhedra.

# Integer Programming

Now we turn to INTEGER PROGRAMMING (INT-PROG). Given $(A \in \mathbb{Z}^{m\times n}, b \in \mathbb{Z}^n, c \in \mathbb{Z}^n, B \in \mathbb{Z})$,

**INT-PROG:** Is there a $y \in \mathbb{Z}^n$ such that $Ay \leq b$ and $c \cdot y \leq B$?

This is asking whether or not there is an integral point that satisfies systems of hyperplane constraints, where the latter may cut out cones or polyhedra.

**Physical Realization:** relevant for counting lattice points that satisfy hyperplane constraints, which is relevant for cohomology calculations that arise when computing matter spectra or instanton zero modes.

**Super concrete:** line bundle cohomology on toric varieties.

# Quadratic Diophantine

Finally, consider the problem QUADRATIC DIOPHANTINE EQUATION (QDE):

**QDE:** Given a quadratic diophantine equation, does it have a solution?

Though generic diophantine equations are undecidable [47] (see [11] for a study of diophantine undecidability in string theory), any single QDE is decidable [48]. However, already in the case where the QDE has the form $ax_1^2 + bx_2 = c$, it is NP-complete [49].

# Quadratic Diophantine

Finally, consider the problem QUADRATIC DIOPHANTINE EQUATION (QDE):

**QDE:** Given a quadratic diophantine equation, does it have a solution?

Though generic diophantine equations are undecidable [47] (see [11] for a study of diophantine undecidability in string theory), any single QDE is decidable [48]. However, already in the case where the QDE has the form $ax_1^2 + bx_2 = c$, it is NP-complete [49].

**Physical Realization:** e.g., certain 3-7 instanton zero mode calculations.

**Interesting caveat:** generic diophantines are *undecidable*, due to Matiyasevich's theorem that solved Hilbert's tenth problem. (**see [Cvetic, Garcia-Etxebarria, JH]**).

# Question 5:

What are potential complexity loopholes and what does it mean for applying ML / AI to landscapes?

# Loopholes: Break Assumptions

# Loopholes: Break Assumptions

- Classical complexity theory is about algorithms on a classical computer that “computes” the problem.

# Loopholes: Break Assumptions

- Classical complexity theory is about algorithms on a classical computer that "computes" the problem.

- Don't go classical:

  - quantum: e.g. Shor's algorithm for factorization.
    but quantum speedup isn't automatic.

  - stochastic: only strict saddles, can escape find loc min in P.
    [Ge, Huang, Jin, Yuan] 2016 (relevant for ML)

# Loopholes: Break Assumptions

- Classical complexity theory is about algorithms on a classical computer that "computes" the problem.

- Don't go classical:

  - quantum: e.g. Shor's algorithm for factorization.
    but quantum speedup isn't automatic.

  - stochastic: only strict saddles, can escape find loc min in P.
    **[Ge, Huang, Jin, Yuan] 2016** **(relevant for ML)**

- Don't "compute": 99% accuracy breaks the assumption, but may be good enough for some purposes, could have P-alg.

# Loopholes: Break Assumptions

- Classical complexity theory is about algorithms on a classical computer that “computes” the problem.

- Don’t go classical:

  - quantum: e.g. Shor’s algorithm for factorization.
    but quantum speedup isn’t automatic.

  - stochastic: only strict saddles, can escape find loc min in P.
    **[Ge, Huang, Jin, Yuan] 2016** **(relevant for ML)**

- Don’t “compute”: 99% accuracy breaks the assumption, but may be good enough for some purposes, could have P-alg.

- Accordingly: are extra classes, BPP and BQP that allow error, and also probabilistic and quantum algorithms, respectively.

# Loopholes: Special Instances and Reasonable N

- Special instances: there can be instances that are in P (nature sometimes utilizes them, e.g., ``minimal frustration” in folding).

- People solve NP-complete problems every day.

  In real-world problems (including theoretical physics) we often don’t care about asymptotic N.

  Google Brain KNAPSACK200: this is an ADK cosmological constant problem in disguise, and they use RL to solve it quickly. But 200 is a perfectly fine # moduli!

  Amazon: solves traveling salesman in warehouses. But your shopping cart only ever have O(10) items! Not O(1,000,000).

# Some Implications

- Each of these loopholes gives potentials ways forward for computationally complex problems that we care about.
- As far as I can tell, there are no hard and fast rules (as we're used to with ML), one should try different possibilities and look for best results.
- Some techniques (e.g. RL, with stochasticity, ε-greedy) can immediately have some of the loopholes bult in.

# Summary

# Summary

- **Why should I care about computational complexity?**
  - not rare: arises quite readily in many systems that we care about.
  - practical implication: one of two obstacle to large N landscapes.
  - physical implication: dynamics can be understood by complexity.

# Summary

- **Why should I care about computational complexity?**
  - not rare: arises quite readily in many systems that we care about.
  - practical implication: one of two obstacle to large N landscapes.
  - physical implication: dynamics can be understood by complexity.

- **What is computational complexity?**
  - a field that formalizes relative difficulty of problems
  - “hard” problems have exponential time instances if P != NP.

# Summary

- **Why should I care about computational complexity?**
  - not rare: arises quite readily in many systems that we care about.
  - practical implication: one of two obstacle to large N landscapes.
  - physical implication: dynamics can be understood by complexity.

- **What is computational complexity?**
  - a field that formalizes relative difficulty of problems
  - “hard” problems have exponential time instances if P != NP.

- **What is the complexity of vacua in landscapes?**
  - finding critical points is hard.
  - pos semi-def Hessian: det. whether crit. pt is loc min is hard.
  - near vacua is in P.

# Summary

- **Why should I care about computational complexity?**
  - not rare: arises quite readily in many systems that we care about.
  - practical implication: one of two obstacle to large N landscapes.
  - physical implication: dynamics can be understood by complexity.

- **What is computational complexity?**
  - a field that formalizes relative difficulty of problems
  - “hard” problems have exponential time instances if P != NP.

- **What is the complexity of vacua in landscapes?**
  - finding critical points is hard.
  - pos semi-def Hessian: det. whether crit. pt is loc min is hard.
  - near vacua is in P.

- **What is the complexity of vacua in the string landscape?**
  - determining the scalar potential involves many hard problems.

# Summary

- **Why should I care about computational complexity?**
  - not rare: arises quite readily in many systems that we care about.
  - practical implication: one of two obstacle to large N landscapes.
  - physical implication: dynamics can be understood by complexity.

- **What is computational complexity?**
  - a field that formalizes relative difficulty of problems
  - “hard” problems have exponential time instances if P != NP.

- **What is the complexity of vacua in landscapes?**
  - finding critical points is hard.
  - pos semi-def Hessian: det. whether crit. pt is loc min is hard.
  - near vacua is in P.

- **What is the complexity of vacua in the string landscape?**
  - determining the scalar potential involves many hard problems.

- **What are potential complexity loopholes and what does it mean for applying ML / AI to landscapes?**
  - break assumptions. e.g., classical, exact computation.
  - nice instances exist, or real-world N. **punchline:** complexity != give up!

# Final Thought:

Most of the string landscape lives at large N, but complexity limits our ability to work in that regime, e.g., our ability to make statistical predictions.

# Final Thought:

Most of the string landscape lives at large N, but complexity limits our ability to work in that regime, e.g., our ability to make statistical predictions.

**This motivates a concrete ML program:**
at various moderate N, learn distributions for generating random EFTs that match string observables, study whether they can be scaled to large N, and (if so) make predictions.

See Cody's talk.

# Thanks!

# Practical Implications

**Question:** **what are the practical takeaways?**

**does this mean anything for dS swampland?**

# Practical Implications

# Practical Implications

- **Recap 1:** given $V(\phi)$, finding either stable or metastable vacua is co-NP-hard.

  finding critical points $\phi^*$ is NP-hard.

  determining whether critical point $\phi^*$ is min is co-NP-hard only if Hessian is positive semi-def at $\phi^*$.

- **Recap 2:** determining $V(\phi)$ in string theory requires solving instances of NP-complete problems.

- **Nested hard problems.** If P != NP, difficulty of finding string vacua is exponential in # of scalar fields.

- explains absence of concrete vacua at # scalars >= 20.
  makes dS swampland not directly verifiable. but it is falsifiable.

# MSVAC Proof

- MAX-CLIQUE: does G have a clique of size >= n?
- QP: is $x^*$ a global minimum of $x^T H x + c^T x$.
- QPLOC: is $x^*$ a local minimum of same.
- Vavasis: QPLOC is co-NP-hard by red. from MAX-CLIQUE.
  But it is a boundary point that is hard.
- [JH, Ruehle]: to Vavasis' QPLOC instance, map it to BOX-QUARTLOC, which makes Vavasis' boundary point and interior point and makes that problem quartic.
- [JH, Ruehle]: MSVAC is co-NP-hard via inclusion from BOX-QUARTLOC.
- remember: co-NP-hard decision problem occurs at points with PSD Hessian.

# Dynamical Implications

**Question:** could complexity affect dynamics?

(this is more speculative, based primarily
on two papers of Douglas, Denef et al and considering
our results in light of their ideas.)

# Complexity Measure

**landscape measure: [Douglas, Denef, Greene,  Zukowski] 2017**
**see also: [Douglas, Denef] 2006, [Aaronson] 2005**

# Complexity Measure

**landscape measure: [Douglas, Denef, Greene,  Zukowski] 2017**
**see also: [Douglas, Denef] 2006, [Aaronson] 2005**

- **rough question:** does Nature solve hard problems?

# Complexity Measure

**landscape measure: [Douglas, Denef, Greene,  Zukowski] 2017**
**see also: [Douglas, Denef] 2006, [Aaronson] 2005**

- **rough question:** does Nature solve hard problems?

- **rough measure idea:**

  We ended up in the universe we observe not necessarily because it is ubiquitous in the landscape, but because it is easy to find.

# Complexity Measure

**landscape measure: [Douglas, Denef, Greene, Zukowski] 2017**
**see also: [Douglas, Denef] 2006, [Aaronson] 2005**

- **rough question:** does Nature solve hard problems?

- **rough measure idea:**

  We ended up in the universe we observe not necessarily because it is ubiquitous in the landscape, but because it is easy to find.

- **what might this mean?**

  **1)** hard problems can have simple instances (i.e. instances in P), in which case we might expect to see the simple instances in Nature.

  **2)** if there is an alternative to solving the hard problem, might expect that.

# Complexity Measure

**landscape measure: [Douglas, Denef, Greene, Zukowski] 2017**
**see also: [Douglas, Denef] 2006, [Aaronson] 2005**

- **rough question:** does Nature solve hard problems?

- **rough measure idea:**

  We ended up in the universe we observe not necessarily because it is ubiquitous in the landscape, but because it is easy to find.

- **what might this mean?**

  **1)** hard problems can have simple instances (i.e. instances in P), in which case we might expect to see the simple instances in Nature.

  **2)** if there is an alternative to solving the hard problem, might expect that.

- what we don’t expect is to see solutions to hard instances of the hard problem.

# Implications for Proteins

- **what might this mean?**

  **1)** see simple instances

  (e.g. bioproteins evolved for simple fast folding)

  **2)** see alternative to solving hard problem.

  (e.g. synthetic proteins in genei are hard instances, don't reach ground state.)

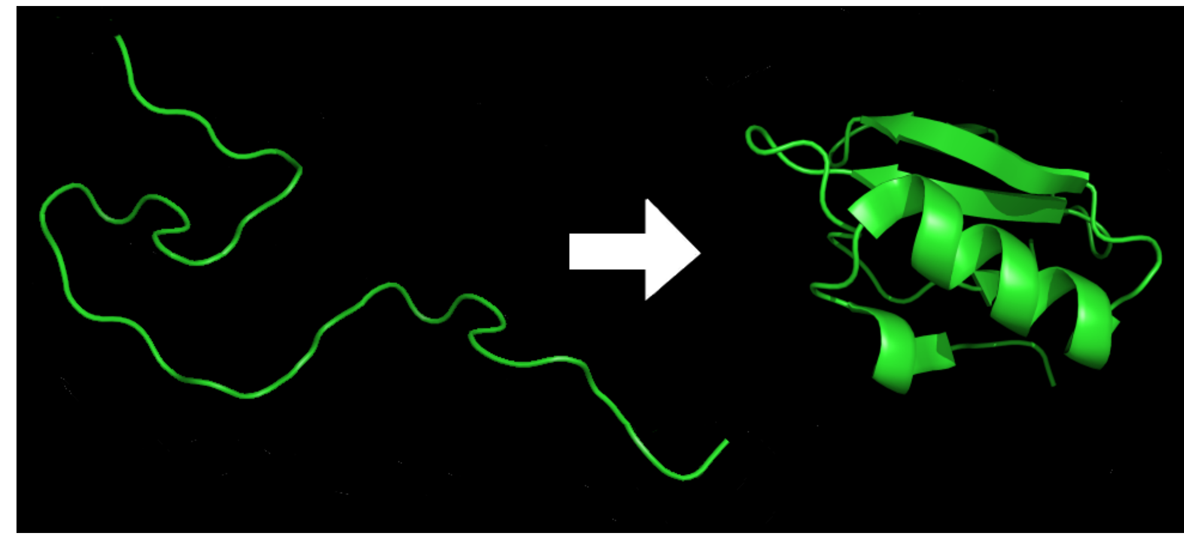


Image: Wikipedia

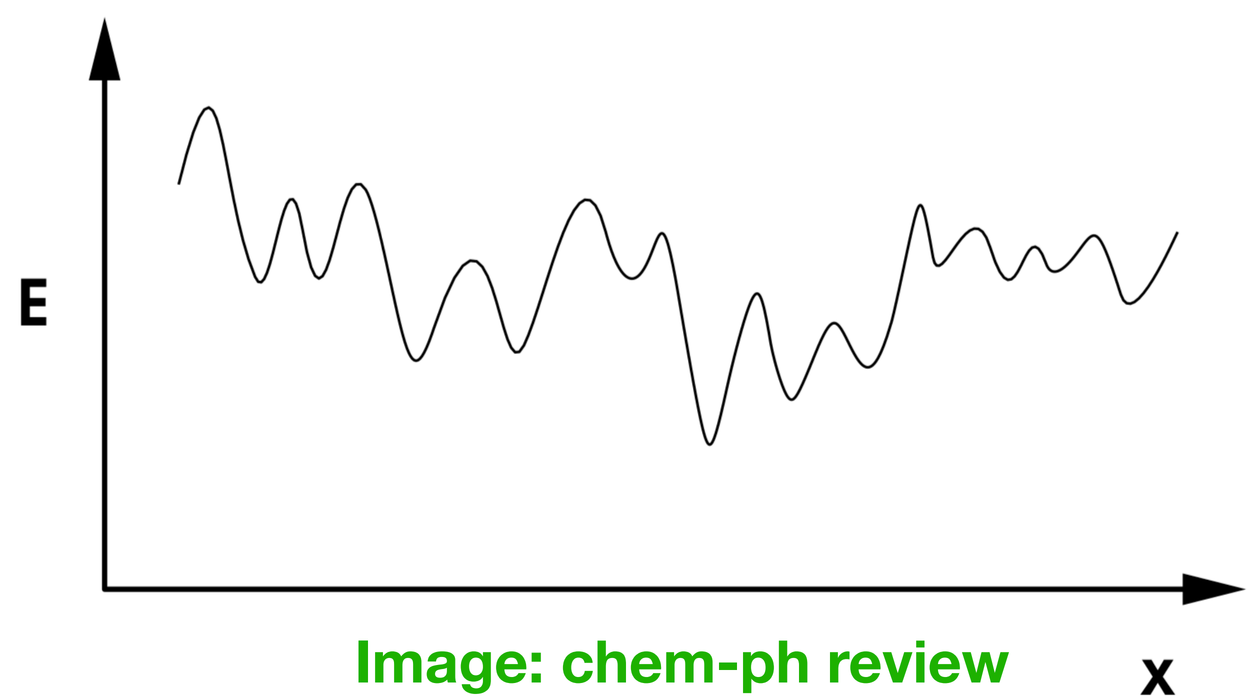



Image: chem-ph review

# Implications for MSVAC

**MSVAC:** Find a local minimum of $V(\phi)$, such that $l_i \leq \phi_i \leq u_i$,

where $\phi = (\phi_1, \ldots, \phi_n) \in \mathbb{R}^n$ and $l_i, u_i \in \mathbb{R}$ are used to give so-called box constraints. Here the box constraints encode the fact that the EFT has a regime of validity that bounds the scalar fields values.

# Implications for MSVAC

**MSVAC:** Find a local minimum of $V(\phi)$, such that $l_i \leq \phi_i \leq u_i$,

where $\phi = (\phi_1, \ldots, \phi_n) \in \mathbb{R}^n$ and $l_i, u_i \in \mathbb{R}$ are used to give so-called box constraints. Here the box constraints encode the fact that the EFT has a regime of validity that bounds the scalar fields values.

- **what might this mean?**

  **1)** see simple instances.
  (e.g. find field or string theory vacua that are in P.)

  **2)** see alternative to solving hard problem.
  (in rolling solution, don't reach local minimum.)

  **3)** long lifetimes? need study of string vacuum decay distribs.