Computational Complexity / Decision Making (at Chess)

Kenneth W. Regan[1]
University at Buffalo (SUNY)

6 September, 2016

[1] Recent Students: Robert Surówka, Tamal Biswas, Michael Wehar, James Clay

Computational Complexity

- The study of the time *needed* to solve computational problems, and how much memory and other resources computers require.

Computational Complexity

- The study of the time *needed* to solve computational problems, and how much memory and other resources computers require.
- Largely independent of the computer model,

Computational Complexity

- The study of the time *needed* to solve computational problems, and how much memory and other resources computers require.
- Largely independent of the computer model, beyond a fundamental divide into **serial**,

Computational Complexity

- The study of the time *needed* to solve computational problems, and how much memory and other resources computers require.
- Largely independent of the computer model, beyond a fundamental divide into **serial**, **parallel**,

Computational Complexity

- The study of the time *needed* to solve computational problems, and how much memory and other resources computers require.
- Largely independent of the computer model, beyond a fundamental divide into **serial**, **parallel**, and **quantum**.

Computational Complexity

- The study of the time *needed* to solve computational problems, and how much memory and other resources computers require.
- Largely independent of the computer model, beyond a fundamental divide into **serial**, **parallel**, and **quantum**.
- Main technical achievement: the relation of computational problems by **reducibility**.

Computational Complexity

- The study of the time *needed* to solve computational problems, and how much memory and other resources computers require.
- Largely independent of the computer model, beyond a fundamental divide into **serial**, **parallel**, and **quantum**.
- Main technical achievement: the relation of computational problems by **reducibility**.
- Main scientific surprise:

Computational Complexity

- The study of the time *needed* to solve computational problems, and how much memory and other resources computers require.
- Largely independent of the computer model, beyond a fundamental divide into **serial**, **parallel**, and **quantum**.
- Main technical achievement: the relation of computational problems by **reducibility**.
- Main scientific surprise:

> The **many thousands** of computational problems that have been studied in many disciplines, some for centuries, cluster into **barely over a dozen** equivalence classes under reducibility.

Computational Complexity

- The study of the time *needed* to solve computational problems, and how much memory and other resources computers require.
- Largely independent of the computer model, beyond a fundamental divide into **serial**, **parallel**, and **quantum**.
- Main technical achievement: the relation of computational problems by **reducibility**.
- Main scientific surprise:

> The **many thousands** of computational problems that have been studied in many disciplines, some for centuries, cluster into **barely over a dozen** equivalence classes under reducibility.

- The biggest cluster is the class of **NP-complete** problems.

P=NP and Worse

- **P**: problems with algorithms that **solve** them in **polynomial time**:

  As the size of the data doubles, the time needed goes up by at most a **linear** factor

P=NP and Worse

- **P**: problems with algorithms that **solve** them in **polynomial time**:

  As the size of the data doubles, the time needed goes up by at most a **linear** factor: $t(n) = n^k \implies t(2n) \leq Kt(n)$,

P=NP and Worse

- **P**: problems with algorithms that **solve** them in **polynomial time**:

As the size of the data doubles, the time needed goes up by at most a **linear** factor: $t(n) = n^k \implies t(2n) \leq Kt(n),\ K = 2^k$.

P=NP and Worse

- **P**: problems with algorithms that **solve** them in **polynomial time**:

As the size of the data doubles, the time needed goes up by at most a **linear** factor: $t(n) = n^k \implies t(2n) \leq Kt(n)$, $K = 2^k$.

- **NP**: "Nondeterministic" Polynomial Time

P=NP and Worse

- **P**: problems with algorithms that **solve** them in **polynomial time**:

> As the size of the data doubles, the time needed goes up by at most a **linear** factor: $t(n) = n^k \implies t(2n) \leq Kt(n),\ K = 2^k$.

- **NP**: "Nondeterministic" Polynomial Time: If you know a secret fact or guess a good answer, you can verify and **teach** it to someone in polynomial time.

P=NP and Worse

- **P**: problems with algorithms that **solve** them in **polynomial time**:

> As the size of the data doubles, the time needed goes up by at most a **linear** factor: $t(n) = n^k \implies t(2n) \leq Kt(n)$, $K = 2^k$.

- **NP**: "Nondeterministic" Polynomial Time: If you know a secret fact or guess a good answer, you can verify and **teach** it to someone in polynomial time.
- Example: Given a Boolean formula f like

$$f = (x_1 \vee (\neg x_2)) \wedge ((\neg x_1) \vee x_2 \vee x_3) \wedge ((\neg x_2) \vee (\neg x_3)),$$

is there a way to make f true?

P=NP and Worse

- **P**: problems with algorithms that **solve** them in **polynomial time**:

As the size of the data doubles, the time needed goes up by at most a **linear** factor: $t(n) = n^k \implies t(2n) \leq Kt(n)$, $K = 2^k$.

- **NP**: "Nondeterministic" Polynomial Time: If you know a secret fact or guess a good answer, you can verify and **teach** it to someone in polynomial time.
- Example: Given a Boolean formula f like

$$f = (x_1 \vee (\neg x_2)) \wedge ((\neg x_1) \vee x_2 \vee x_3) \wedge ((\neg x_2) \vee (\neg x_3)),$$

is there a way to make f true? Called *Satisfiability* (SAT).

P=NP and Worse

- **P**: problems with algorithms that **solve** them in **polynomial time**:

> As the size of the data doubles, the time needed goes up by at most a **linear** factor: $t(n) = n^k \implies t(2n) \leq Kt(n)$, $K = 2^k$.

- **NP**: "Nondeterministic" Polynomial Time: If you know a secret fact or guess a good answer, you can verify and **teach** it to someone in polynomial time.
- Example: Given a Boolean formula f like

$$f = (x_1 \vee (\neg x_2)) \wedge ((\neg x_1) \vee x_2 \vee x_3) \wedge ((\neg x_2) \vee (\neg x_3)),$$

  is there a way to make f true? Called *Satisfiability* (SAT).
- Equivalent to $\neg f$ *not* being a **tautology**.

P=NP and Worse

- **P**: problems with algorithms that **solve** them in **polynomial time**:

> As the size of the data doubles, the time needed goes up by at most a **linear** factor: $t(n) = n^k \implies t(2n) \le Kt(n)$, $K = 2^k$.

- **NP**: "Nondeterministic" Polynomial Time: If you know a secret fact or guess a good answer, you can verify and **teach** it to someone in polynomial time.
- Example: Given a Boolean formula f like

$$f = (x_1 \vee (\neg x_2)) \wedge ((\neg x_1) \vee x_2 \vee x_3) \wedge ((\neg x_2) \vee (\neg x_3)),$$

  is there a way to make f true? Called *Satisfiability* (SAT).
- Equivalent to $\neg f$ *not* being a **tautology**.
- Is NP-complete,

P=NP and Worse

- **P**: problems with algorithms that **solve** them in **polynomial time**:

> As the size of the data doubles, the time needed goes up by at most a **linear** factor: $t(n) = n^k \implies t(2n) \leq Kt(n)$, $K = 2^k$.

- **NP**: "Nondeterministic" Polynomial Time: If you know a secret fact or guess a good answer, you can verify and **teach** it to someone in polynomial time.
- Example: Given a Boolean formula f like

$$f = (x_1 \vee (\neg x_2)) \wedge ((\neg x_1) \vee x_2 \vee x_3) \wedge ((\neg x_2) \vee (\neg x_3)),$$

  is there a way to make f true? Called *Satisfiability* (SAT).
- Equivalent to $\neg f$ *not* being a **tautology**.
- Is NP-complete, so NP = P $\iff$ SAT belongs to P.

P=NP and Worse

- **P**: problems with algorithms that **solve** them in **polynomial time**:

As the size of the data doubles, the time needed goes up by at most a linear factor: $t(n) = n^k \implies t(2n) \leq Kt(n)$, $K = 2^k$.

- **NP**: "Nondeterministic" Polynomial Time: If you know a secret fact or guess a good answer, you can verify and **teach** it to someone in polynomial time.
- Example: Given a Boolean formula f like

$$f = (x_1 \vee (\neg x_2)) \wedge ((\neg x_1) \vee x_2 \vee x_3) \wedge ((\neg x_2) \vee (\neg x_3)),$$

is there a way to make f true? Called *Satisfiability* (SAT).

- Equivalent to $\neg f$ *not* being a **tautology**.
- Is NP-complete, so NP $=$ P $\iff$ SAT belongs to P.
- We don't even know whether SAT can be solved in **linear** time!

Other Problems and Models

Other Problems and Models

- **Factoring** is among a handful of problems in NP not known to be complete or in P.

Other Problems and Models

- **Factoring** is among a handful of problems in NP not known to be complete or in P.
- Among few problems we *want* to be hard, since RSA security depends on it.

Other Problems and Models

- **Factoring** is among a handful of problems in NP not known to be complete or in P.
- Among few problems we *want* to be hard, since RSA security depends on it.
- But solvable in polynomial time by a **quantum computer**.

Other Problems and Models

- **Factoring** is among a handful of problems in NP not known to be complete or in P.
- Among few problems we *want* to be hard, since RSA security depends on it.
- But solvable in polynomial time by a **quantum computer**.
- Textbook on quantum algorithms;

Other Problems and Models

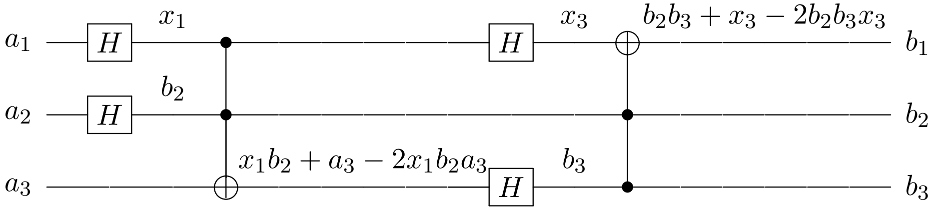
- **Factoring** is among a handful of problems in NP not known to be complete or in P.
- Among few problems we *want* to be hard, since RSA security depends on it.
- But solvable in polynomial time by a **quantum computer**.
- Textbook on quantum algorithms; blog series: Can QCs be Built?

Other Problems and Models

- **Factoring** is among a handful of problems in NP not known to be complete or in P.
- Among few problems we *want* to be hard, since RSA security depends on it.
- But solvable in polynomial time by a **quantum computer**.
- Textbook on quantum algorithms; blog series: Can QCs be Built?
- Research on simulating **quantum circuits** by algebra,

Other Problems and Models

- **Factoring** is among a handful of problems in NP not known to be complete or in P.
- Among few problems we *want* to be hard, since RSA security depends on it.
- But solvable in polynomial time by a **quantum computer**.
- Textbook on quantum algorithms; blog series: Can QCs be Built?
- Research on simulating **quantum circuits** by algebra, for example:



Decision Making in Chess...

Decision Making in Chess...

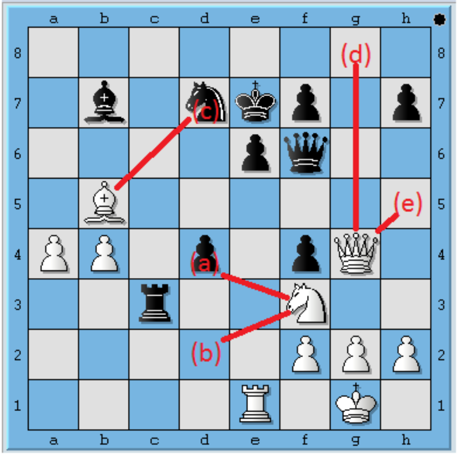
Decision Making in Chess... and Tests

The ____ of drug-resistant strains of bacteria and viruses has ____ researchers' hopes that permanent victories against many diseases have been achieved.

- (a) vigor . . corroborated
- (b) feebleness . . dashed
- (c) proliferation . . blighted
- (d) destruction . . disputed
- (e) disappearance . . frustrated

(source: itunes.apple.com)

=



Advantages of Chess Model

Advantages of Chess Model

1. **Large data**: tens of millions of moves in the public record of games.

Advantages of Chess Model

1. **Large data**: tens of millions of moves in the public record of games.
2. **Known and Stable Standards**: Quality in chess measured by **Elo rating scale**.

Advantages of Chess Model

1. **Large data**: tens of millions of moves in the public record of games.
2. **Known and Stable Standards**: Quality in chess measured by **Elo rating scale**.
3. **Depth** and **level** of thinking natural from structure of game.

Advantages of Chess Model

1. **Large data**: tens of millions of moves in the public record of games.
2. **Known and Stable Standards**: Quality in chess measured by **Elo rating scale**.
3. **Depth** and **level** of thinking natural from structure of game.
4. **Intrinsic** formulation of **difficulty**.

Advantages of Chess Model

1. **Large data**: tens of millions of moves in the public record of games.
2. **Known and Stable Standards**: Quality in chess measured by **Elo rating scale**.
3. **Depth** and **level** of thinking natural from structure of game.
4. **Intrinsic** formulation of **difficulty**.
5. **Tight correspondence** to **item-response theory** and other *psychometric* and decision-making models.

Advantages of Chess Model

1. **Large data**: tens of millions of moves in the public record of games.
2. **Known and Stable Standards**: Quality in chess measured by **Elo rating scale**.
3. **Depth** and **level** of thinking natural from structure of game.
4. **Intrinsic** formulation of **difficulty**.
5. **Tight correspondence** to **item-response theory** and other *psychometric* and decision-making models.
6. **Predictive Analytics**: can do risk evaluation, fraud detection...

Advantages of Chess Model

1. **Large data**: tens of millions of moves in the public record of games.
2. **Known and Stable Standards**: Quality in chess measured by **Elo rating scale**.
3. **Depth** and **level** of thinking natural from structure of game.
4. **Intrinsic** formulation of **difficulty**.
5. **Tight correspondence** to **item-response theory** and other *psychometric* and decision-making models.
6. **Predictive Analytics**: can do risk evaluation, fraud detection...
7. Within chess: **intrinsic ratings** and **cheating testing**.

Advantages of Chess Model

1. **Large data**: tens of millions of moves in the public record of games.
2. **Known and Stable Standards**: Quality in chess measured by **Elo rating scale**.
3. **Depth** and **level** of thinking natural from structure of game.
4. **Intrinsic** formulation of **difficulty**.
5. **Tight correspondence** to **item-response theory** and other *psychometric* and decision-making models.
6. **Predictive Analytics**: can do risk evaluation, fraud detection...
7. Within chess: **intrinsic ratings** and **cheating testing**.
8. **Discover new scientific regularities of human thought processes.**