Computations with Incomplete or Imperfect Information

Gregory Igusa

University of Notre Dame

10 June, 2014

In complexity theory, it has been observed that problems can be difficult in theory while being quite easy to solve in practice.

1986: Levin introduces “average-case complexity.”

2003: Kapovich, Miasnikov, Schupp and Shpilrain introduce “generic-case complexity.”

Background

In complexity theory, it has been observed that problems can be difficult in theory while being quite easy to solve in practice.

1986: Levin introduces “average-case complexity.”

2003: Kapovich, Miasnikov, Schupp and Shpilrain introduce “generic-case complexity.”

Background

In complexity theory, it has been observed that problems can be difficult in theory while being quite easy to solve in practice.

1986: Levin introduces “average-case complexity.”

2003: Kapovich, Miasnikov, Schupp and Shpilrain introduce “generic-case complexity.”

In 2012, Jockusch, and Schupp introduce and analyze the notion of generic computability. Informally, real is genericaly computable if there is a computation of that real that is usually correct.

We formalize our notion of "usually" using asymptotic density:

Definition

The density of real A is the limit of the densities of its initial segments, $\lim_{n\to\infty} \frac{|A \cap n|}{n}$.

Density

In 2012, Jockusch, and Schupp introduce and analyze the notion of generic computability. Informally, real is genericaly computable if there is a computation of that real that is usually correct.

We formalize our notion of “usually” using asymptotic density:

Definition

The density of real A is the limit of the densities of its initial segments, $\lim_{n\to\infty} \frac{|A\cap n|}{n}$.

Generic and coarse computablity

Definition

A real A is generically computable if there exists a partial computable function φ whose domain has density 1 such that $\varphi(n) = A(n)$ for all $n \in \mathrm{dom}(\varphi)$.

This is distinct from the following related notion.

Definition

A real A is coarsely computable if there exists a total computable function φ such that $\{n : \varphi(n) = B(n)\}$ has density 1.

So a generic computation is a computation that usually halts, always correctly, while a coarse computation is a computation that always halts, usually correctly.

Generic and coarse computablity

Definition

A real A is generically computable if there exists a partial computable function φ whose domain has density 1 such that $\varphi(n) = A(n)$ for all $n \in \mathrm{dom}(\varphi)$.

This is distinct from the following related notion.

Definition

A real A is coarsely computable if there exists a total computable function φ such that $\{n : \varphi(n) = B(n)\}$ has density 1.

So a generic computation is a computation that usually halts, always correctly, while a coarse computation is a computation that always halts, usually correctly.

Generic and coarse computablity

Definition

A real A is generically computable if there exists a partial computable function φ whose domain has density 1 such that $\varphi(n) = A(n)$ for all $n \in \mathrm{dom}(\varphi)$.

This is distinct from the following related notion.

Definition

A real A is coarsely computable if there exists a total computable function φ such that $\{n : \varphi(n) = B(n)\}$ has density 1.

So a generic computation is a computation that usually halts, always correctly, while a coarse computation is a computation that always halts, usually correctly.

Examples

Theorem (Jockusch, Schupp, 2012)

Neither generic computability, nor coarse computability implies the other.

Objection (Moral Grounds)

It is better to be incomplete than to be inaccurate!

Metatheorem

Generic computability is closer to coarse computability than coarse computability is to generic computability.

Examples

Theorem (Jockusch, Schupp, 2012)

Neither generic computability, nor coarse computability implies the other.

Objection (Moral Grounds)

It is better to be incomplete than to be inaccurate!

Metatheorem

Generic computability is closer to coarse computability than coarse computability is to generic computability.

Theorem (Jockusch, Schupp, 2012)

Neither generic computability, nor coarse computability implies the other.

Objection (Moral Grounds)

It is better to be incomplete than to be inaccurate!

Metatheorem

Generic computability is closer to coarse computability than coarse computability is to generic computability.

Coarse but not Generic

How does one produce a coarsely computable real that is not generically computable?

Any real that has density 1 is coarsely computable – just make sure each potential generic computation is wrong at least once.

Theorem (I, 2013)

Every nonzero Turing degree (Turing) computes a real that is coarsely computable but not generically computable.

Coarse but not Generic

How does one produce a coarsely computable real that is not generically computable?

Any real that has density 1 is coarsely computable – just make sure each potential generic computation is wrong at least once.

Theorem (I, 2013)

Every nonzero Turing degree (Turing) computes a real that is coarsely computable but not generically computable.

So, how does one produce something generically computable but not coarsely computable? (Proof sketch)

Theorem (Downey, Jockusch, Schupp, 2013)

Every nonzero c.e. degree (Turing) computes a real that is generically computable but not coarsely computable.

Theorem (Hirschfeld, Jockusch, McNicholl, Schupp)

If A is 1-generic, or weakly 2-random, then A does not compute any sets that are generically computable but not coarsely computable.

Generic but not Coarse

So, how does one produce something generically computable but not coarsely computable? (Proof sketch)

Theorem (Downey, Jockusch, Schupp, 2013)

Every nonzero c.e. degree (Turing) computes a real that is generically computable but not coarsely computable.

Theorem (Hirschfeld, Jockusch, McNicholl, Schupp)

If A is 1-generic, or weakly 2-random, then A does not compute any sets that are generically computable but not coarsely computable.

This addresses the question of how difficult it is to witness a nonimplication, but now we ask how far the nonimplications can be pushed.

Let $r \in [0, 1]$.

Definition

A real A is generically computable at density r if there exists a partial computable function φ whose domain has lower density $\geq \alpha$ such that $\varphi(n) = A(n)$ for all $n \in \mathrm{dom}(\varphi)$.

Definition

A real A is coarsely computable at density r if there exists a total computable function φ such that $\{n : \varphi(n) = B(n)\}$ has lower density $\geq \alpha$.

This addresses the question of how difficult it is to witness a nonimplication, but now we ask how far the nonimplications can be pushed.

Let $r \in [0, 1]$.

Definition

A real A is generically computable at density r if there exists a partial computable function φ whose domain has lower density $\geq \alpha$ such that $\varphi(n) = A(n)$ for all $n \in \mathrm{dom}(\varphi)$.

Definition

A real A is coarsely computable at density r if there exists a total computable function φ such that $\{n : \varphi(n) = B(n)\}$ has lower density $\geq \alpha$.

This addresses the question of how difficult it is to witness a nonimplication, but now we ask how far the nonimplications can be pushed.

Let $r \in [0, 1]$.

Definition

A real A is generically computable at density r if there exists a partial computable function φ whose domain has lower density $\geq \alpha$ such that $\varphi(n) = A(n)$ for all $n \in \mathrm{dom}(\varphi)$.

Definition

A real A is coarsely computable at density r if there exists a total computable function φ such that $\{n : \varphi(n) = B(n)\}$ has lower density $\geq \alpha$.

This addresses the question of how difficult it is to witness a nonimplication, but now we ask how far the nonimplications can be pushed.

Let $r \in [0, 1]$.

Definition

A real A is generically computable at density r if there exists a partial computable function φ whose domain has lower density $\geq \alpha$ such that $\varphi(n) = A(n)$ for all $n \in \mathrm{dom}(\varphi)$.

Definition

A real A is coarsely computable at density r if there exists a total computable function φ such that $\{n : \varphi(n) = B(n)\}$ has lower density $\geq \alpha$.

Theorem (Downey, Jockusch, McNicholl, Schupp)

If A is generically computable at density r, then for every $\epsilon > 0$, A is coarsely computable at density $r - \epsilon$.

Proof: Nonuniformly give yourself the point at which the density of the domain of the generic computation never again drops below $r - \frac{\epsilon}{2}$

Observation

There exist reals that are coarsely computable, but not generically computable at any positive density. (I.e. coarsely computable, and absolutely undecidable.)

Theorem (Downey, Jockusch, McNicholl, Schupp)

If A is generically computable at density r, then for every $\epsilon > 0$, A is coarsely computable at density $r - \epsilon$.

Proof: Nonuniformly give yourself the point at which the density of the domain of the generic computation never again drops below $r - \frac{\epsilon}{2}$

Observation

There exist reals that are coarsely computable, but not generically computable at any positive density. (I.e. coarsely computable, and absolutely undecidable.)

Theorem (Downey, Jockusch, McNicholl, Schupp)

If A is generically computable at density r, then for every $\epsilon > 0$, A is coarsely computable at density $r - \epsilon$.

Proof: Nonuniformly give yourself the point at which the density of the domain of the generic computation never again drops below $r - \frac{\epsilon}{2}$

Observation

There exist reals that are coarsely computable, but not generically computable at any positive density. (I.e. coarsely computable, and absolutely undecidable.)

For a recursion theorist, probably the most natural way of asking how close something is to being computable is by asking about its Turing degree.

If we wish to know how close a real is to being generically or coarsely computable, we should ask the question within a degree structure for that computability.

We now introduce generic reducibility, and coarse reducibility.

For a recursion theorist, probably the most natural way of asking how close something is to being computable is by asking about its Turing degree.

If we wish to know how close a real is to being generically or coarsely computable, we should ask the question within a degree structure for that computability.

We now introduce generic reducibility, and coarse reducibility.

For a recursion theorist, probably the most natural way of asking how close something is to being computable is by asking about its Turing degree.

If we wish to know how close a real is to being generically or coarsely computable, we should ask the question within a degree structure for that computability.

We now introduce generic reducibility, and coarse reducibility.

Partial Oracles

Definition

Let A be a real. Then a (time-dependent) partial oracle, (A), for A is a set of ordered triples $\langle n, x, s\rangle$ such that:
$\exists s(\langle n, 0, s\rangle \in (A)) \implies n \notin A$,
$\exists s(\langle n, 1, s\rangle \in (A)) \implies n \in A$.

We think of (A) as a partial function, sending n to x. We think of s as the number of steps it takes (A) to converge.

The domain of (A) is the set of n for which there exists such an x, s.

Definition

Let A be a real. Then a (time-dependent) partial oracle, (A), for A is a set of ordered triples $\langle n, x, s\rangle$ such that:
$\exists s(\langle n, 0, s\rangle \in (A)) \implies n \notin A$,
$\exists s(\langle n, 1, s\rangle \in (A)) \implies n \in A$.

We think of (A) as a partial function, sending n to x. We think of s as the number of steps it takes (A) to converge.

The domain of (A) is the set of n for which there exists such an x, s.

Partial Oracles

Definition

Let A be a real. Then a (time-dependent) partial oracle, (A), for A is a set of ordered triples $\langle n, x, s\rangle$ such that:
$\exists s(\langle n, 0, s\rangle \in (A)) \implies n \notin A$,
$\exists s(\langle n, 1, s\rangle \in (A)) \implies n \in A$.

We think of (A) as a partial function, sending n to x. We think of s as the number of steps it takes (A) to converge.

The domain of (A) is the set of n for which there exists such an x, s.

Generic reductions

Definition

Let A be a real. Then a generic oracle for A is a partial oracle whose domain is density-1.

Note that generically computing A is equivalent to computing a generic oracle for A.

Definition

Let A, B be reals. We say A is generically reducible to B (or $A \leq_g B$) if there is a Turing functional φ such that for every generic oracle (B), for B, $\varphi^{(B)}$ is a generic computation of A.

Definition

Let A be a real. Then a generic oracle for A is a partial oracle whose domain is density-1.

Note that generically computing A is equivalent to computing a generic oracle for A.

Definition

Let A, B be reals. We say A is generically reducible to B (or $A \leq_g B$) if there is a Turing functional φ such that for every generic oracle (B), for B, $\varphi^{(B)}$ is a generic computation of A.

Generic reductions

Definition

Let A be a real. Then a generic oracle for A is a partial oracle whose domain is density-1.

Note that generically computing A is equivalent to computing a generic oracle for A.

Definition

Let A, B be reals. We say A is generically reducible to B (or $A \leq_g B$) if there is a Turing functional φ such that for every generic oracle (B), for B, $\varphi^{(B)}$ is a generic computation of A.

Coarse reductions

Definition

Let A be a real. Then a coarse oracle for A is an (ordinary Turing) oracle for a set that agrees with A on density-1.

Definition

Let A, B be reals. We say A is coarsely reducible to B (or $A \leq_g B$) if there is a Turing functional φ such that for every coarse oracle (B), for B, $\varphi^{(B)}$ is a coarse computation of A.

Embedding the Turing degrees in the generic degrees

There is a natural embedding of the Turing degrees into the generic degrees:

Definition

For any real X, let $\mathcal{R}(X)$ be defined as follows.
$\mathcal{R}(X) = \{2^n(2k+1) : n \in X\}$.

So we have "stretched" every bit of X into a positive density "column" of $\mathcal{R}(X)$.

Since every generic computation of $\mathcal{R}(X)$ must include at least one bit from every column, it must be able to compute X.

As a result, generically computing $\mathcal{R}(X)$ is the same as computing X, and working with $\mathcal{R}(X)$ as a generic oracle is the same as working with X as an oracle in the usual sense.

Embedding the Turing degrees in the generic degrees

There is a natural embedding of the Turing degrees into the generic degrees:

Definition

For any real X, let $\mathcal{R}(X)$ be defined as follows.
$\mathcal{R}(X) = \{2^n(2k+1) : n \in X\}$.

So we have "stretched" every bit of X into a positive density "column" of $\mathcal{R}(X)$.

Since every generic computation of $\mathcal{R}(X)$ must include at least one bit from every column, it must be able to compute X.

As a result, generically computing $\mathcal{R}(X)$ is the same as computing X, and working with $\mathcal{R}(X)$ as a generic oracle is the same as working with X as an oracle in the usual sense.

There is a natural embedding of the Turing degrees into the generic degrees:

Definition

For any real X, let $\mathcal{R}(X)$ be defined as follows.
$\mathcal{R}(X) = \{2^n(2k+1) : n \in X\}$.

So we have "stretched" every bit of X into a positive density "column" of $\mathcal{R}(X)$.

Since every generic computation of $\mathcal{R}(X)$ must include at least one bit from every column, it must be able to compute X.

As a result, generically computing $\mathcal{R}(X)$ is the same as computing X, and working with $\mathcal{R}(X)$ as a generic oracle is the same as working with X as an oracle in the usual sense.

There is a natural embedding of the Turing degrees into the generic degrees:

Definition

For any real X, let $\mathcal{R}(X)$ be defined as follows.
$\mathcal{R}(X) = \{2^n(2k+1) : n \in X\}$.

So we have "stretched" every bit of X into a positive density "column" of $\mathcal{R}(X)$.

Since every generic computation of $\mathcal{R}(X)$ must include at least one bit from every column, it must be able to compute X.

As a result, generically computing $\mathcal{R}(X)$ is the same as computing X, and working with $\mathcal{R}(X)$ as a generic oracle is the same as working with X as an oracle in the usual sense.

Embedding the Turing degrees in the generic degrees

There is a natural embedding of the Turing degrees into the generic degrees:

Definition

For any real X, let $\mathcal{R}(X)$ be defined as follows.
$\mathcal{R}(X) = \{2^n(2k+1) : n \in X\}$.

So we have "stretched" every bit of X into a positive density "column" of $\mathcal{R}(X)$.

Since every generic computation of $\mathcal{R}(X)$ must include at least one bit from every column, it must be able to compute X.

As a result, generically computing $\mathcal{R}(X)$ is the same as computing X, and working with $\mathcal{R}(X)$ as a generic oracle is the same as working with X as an oracle in the usual sense.

Note that this embedding fails quite badly for the coarse degrees.

Observation

If A is Δ^0_2, then $\mathcal{R}(A)$ is coarsely computable.

Theorem (Hirschfeldt, Jockusch, Kuyper, Schupp), (Dzhafarov, I.)

There exists an embedding of the Turing degrees into the coarse degrees.

Note that this embedding fails quite badly for the coarse degrees.

Observation

If A is Δ_2^0, then $\mathcal{R}(A)$ is coarsely computable.

Theorem (Hirschfeldt, Jockusch, Kuyper, Schupp), (Dzhafarov, I.)

There exists an embedding of the Turing degrees into the coarse degrees.

Density-1 Generic Degrees

We say that a generic degree is density-1 if it is the generic degree of a density-1 real.

Lemma

The density-1 generic degrees are precisely the generic degrees of the coarsely computable reals.

Proof: Consider the set of n on which the coarse computation is correct.

Density-1 Generic Degrees

We say that a generic degree is density-1 if it is the generic degree of a density-1 real.

Lemma

The density-1 generic degrees are precisely the generic degrees of the coarsely computable reals.

Proof: Consider the set of n on which the coarse computation is correct.

Theorem (I.)

Let A be a real. Then A is hyperarithmetic if and only if there is a density-1 real B, such that $B \geq_g \mathcal{R}(A)$.

This uses Solovay's characterization of the hyperarithmetic reals in terms of moduli of computation.

Theorem (Solovay)

Let A be a real. Then A is hyperarithmetic if and only if there is a function f, and a Turing functional φ such that for every function g majorizing f, φ^g is a computation of A. In this case, we say that f is a modulus of computation for A.

Theorem (I.)

Let A be a real. Then A is hyperarithmetic if and only if there is a density-1 real B, such that $B \geq_g \mathcal{R}(A)$.

This uses Solovay's characterization of the hyperarithmetic reals in terms of moduli of computation.

Theorem (Solovay)

Let A be a real. Then A is hyperarithmetic if and only if there is a function f, and a Turing functional φ such that for every function g majorizing f, φ^g is a computation of A. In this case, we say that f is a modulus of computation for A.

Idea: Let B be a density-1 real. Then the rate at which the density of B goes to 1 is a slow growing function, and any generic oracle for B computes a slower growing function.

This gives us one direction immediately: any modulus of computation can be emulated by the generic degree of a coarsely computable real.

Objection

Just because you know how quickly the density goes to 1 doesn't mean you know exactly which elements are missing!

Idea: Let B be a density-1 real. Then the rate at which the density of B goes to 1 is a slow growing function, and any generic oracle for B computes a slower growing function.

This gives us one direction immediately: any modulus of computation can be emulated by the generic degree of a coarsely computable real.

Objection

Just because you know how quickly the density goes to 1 doesn't mean you know exactly which elements are missing!

Idea: Let B be a density-1 real. Then the rate at which the density of B goes to 1 is a slow growing function, and any generic oracle for B computes a slower growing function.

This gives us one direction immediately: any modulus of computation can be emulated by the generic degree of a coarsely computable real.

Objection

Just because you know how quickly the density goes to 1 doesn't mean you know exactly which elements are missing!

Theorem (I.)

Let A be a real. Then A is hyperarithmetic if and only if there is a density-1 real B, such that $B \geq_g \mathcal{R}(A)$.

($\Rightarrow$) This direction is easy:

Make the density of B approach 1 very slowly. Then any generic oracle will have density that also approaches 1 at least as slowly.

Theorem (I.)

Let A be a real. Then A is hyperarithmetic if and only if there is a density-1 real B, such that $B \geq_g \mathcal{R}(A)$.

($\Rightarrow$) This direction is easy:
Make the density of B approach 1 very slowly. Then any generic oracle will have density that also approaches 1 at least as slowly.

That's totally correct!

Theorem (I.)

There exists a density-1 real, B, such that for every $f : \mathbb{N} \to \mathbb{N}$, and every φ, there is a $g \geq f$ such that φ^g is not a generic computation of B.

However, the rate of growth of B can be used to compute any *Turing degree* that embeds below B.

- Start with $B \geq_g \mathcal{R}(A)$
- Choose f so that for any $g \gg f$, g can generate a tree of density-1 oracles that includes B.
- Those oracles then repeatedly attempt to elect a “leader” who can cause them to vote unanimously.
- B is such a leader, so eventually they will find one.
- B always votes correctly, so when they find a leader, the vote will be correct.

Note that intersecting B with a density-1 real provides a generic oracle for B.

- Start with $B \geq_g \mathcal{R}(A)$
- Choose f so that for any $g \gg f$, g can generate a tree of density-1 oracles that includes B.
- Those oracles then repeatedly attempt to elect a "leader" who can cause them to vote unanimously.
- B is such a leader, so eventually they will find one.
- B always votes correctly, so when they find a leader, the vote will be correct.

Note that intersecting B with a density-1 real provides a generic oracle for B.

- Start with $B \geq_g \mathcal{R}(A)$
- Choose f so that for any $g \gg f$, g can generate a tree of density-1 oracles that includes B.
- Those oracles then repeatedly attempt to elect a "leader" who can cause them to vote unanimously.
- B is such a leader, so eventually they will find one.
- B always votes correctly, so when they find a leader, the vote will be correct.

Note that intersecting B with a density-1 real provides a generic oracle for B.

- Start with $B \geq_g \mathcal{R}(A)$
- Choose f so that for any $g \gg f$, g can generate a tree of density-1 oracles that includes B.
- Those oracles then repeatedly attempt to elect a “leader” who can cause them to vote unanimously.
- B is such a leader, so eventually they will find one.
- B always votes correctly, so when they find a leader, the vote will be correct.

Note that intersecting B with a density-1 real provides a generic oracle for B.

- Start with $B \geq_g \mathcal{R}(A)$
- Choose f so that for any $g \gg f$, g can generate a tree of density-1 oracles that includes B.
- Those oracles then repeatedly attempt to elect a “leader” who can cause them to vote unanimously.
- B is such a leader, so eventually they will find one.
- B always votes correctly, so when they find a leader, the vote will be correct.

Note that intersecting B with a density-1 real provides a generic oracle for B.

Question

What about the coarse degrees of generically computable reals? Is it possible to code any Turing information into such a degree?

Minimal degrees and pairs

We ask one last question

Question

Given a nonzero generic degree $\mathbf{a}$*, is there always a density-1 degree* $\mathbf{b}$ *such that is* $\mathbf{a} \geq_g \mathbf{b}$*?*

If the answer to the question is "yes," then there cannot be any minimal generic degrees, because the density-1 degrees are dense.

If the answer to the question is "no," then the counterexample is half of a minimal pair for generic reduction.

Minimal degrees and pairs

We ask one last question

Question

Given a nonzero generic degree $\mathbf{a}$*, is there always a density-1 degree* $\mathbf{b}$ *such that is* $\mathbf{a} \geq_g \mathbf{b}$*?*

If the answer to the question is “yes,” then there cannot be any minimal generic degrees, because the density-1 degrees are dense.

If the answer to the question is “no,” then the counterexample is half of a minimal pair for generic reduction.

End

Thank you for your attention.

References

R. Downey, C. Jockusch, and P. Schupp, Asymptotic density and computably enumerable sets, *Journal of Mathematical Logic* 13 (2) (2013), 135005 1–43.

D. Hirschfeldt, C. Jockusch, T. McNicholl, and P. Schupp, Asymptotic density and the coarse computability bound, in preparation.

G. Igusa, Nonexistence of minimal pairs for generic computability, *Journal of Symbolic Logic* 78 (2) (2013), 511–522.

C. Jockusch and P. Schupp, Generic computability, Turing degrees and asymptotic density, *Journal of the London Mathematical Society* 85 (2) (2012), 472–490.