Critical Hits and Calculus

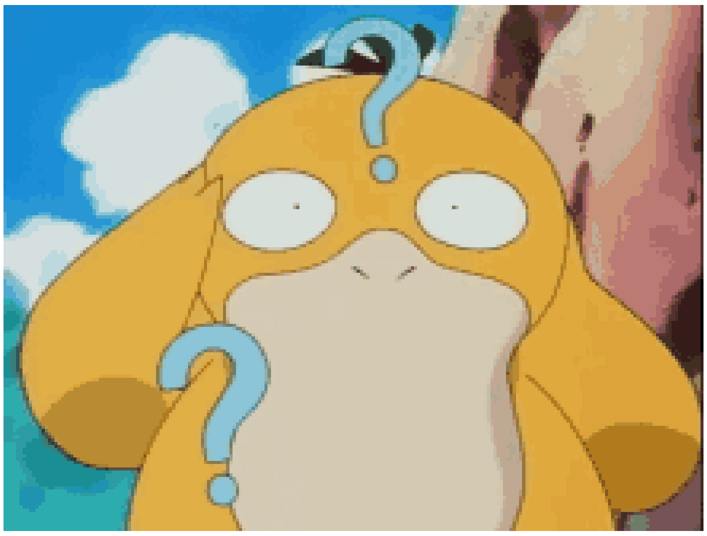
Game Theory, Spectral Graphs, and Probability in Competitive Pokémon

Dev Goyal

Tom Rocks Maths Essay Competition 2025

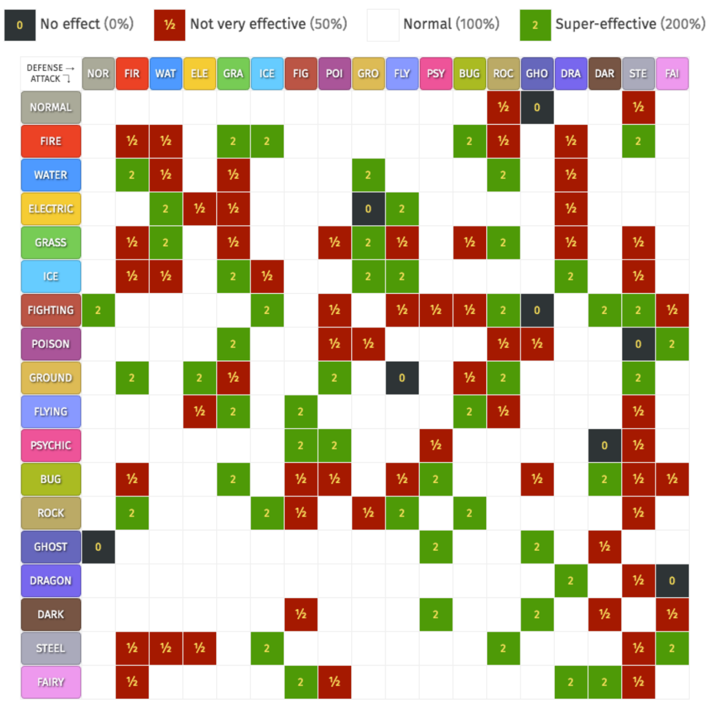
Introduction: A Game of Chess and Poker with Pokémon

Even though Pokémon is one of the world's most valuable franchises, many people remain unaware of the competitive game. I was astonished when I discovered the depth of strategy in the official Video Game Championships (VGC)**. I started playing 5 years ago and still love it. I have been consistently ranked among the Top 500 players on the planet and along with that have been #1 for a while during Reg H of 2024.** Competitive Pokémon VGC blends chess-like turn-based positioning with poker-like hidden information and probability. Players bring six Pokémon, then secretly choose four to battle in double battles (two on the field per side). Each turn, both players select moves simultaneously, and success hinges on careful calculations under uncertainty, managing damage variance, statuses, and synergy. Below, we explore the mathematical frameworks—from type charts and EV optimization to game theory and Monte Carlo methods—that underlie this complex domain.



Types and Damage Multipliers

One of the first pillars of competitive play is the type system. Each Pokémon and each move has one or two types among the 18 elemental types (Fire, Water, Grass, etc.).



0 No effect (0%) ½ Not very effective (50%) Normal (100%) 2 Super-effective (200%)

DEFENSE → ATTACK ⤵	NOR	FIR	WAT	ELE	GRA	ICE	FIG	POI	GRO	FLY	PSY	BUG	ROC	GHO	DRA	DAR	STE	FAI
NORMAL													½	0			½	
FIRE		½	½		2	2						2	½		½		2	
WATER		2	½		½				2				2		½			
ELECTRIC			2	½	½				0	2					½			
GRASS		½	2		½			½	2	½		½	2		½		½	
ICE		½	½		2	½			2	2					2		½	
FIGHTING	2					2		½		½	½	½	2	0		2	2	½
POISON					2			½	½				½	½			0	2
GROUND		2		2	½			2		0		½	2				2	
FLYING				½	2		2					2	½				½	
PSYCHIC							2	2			½					0	½	
BUG		½			2		½	½		½	2			½		2	½	½
ROCK		2				2	½		½	2		2					½	
GHOST	0										2			2		½		
DRAGON															2		½	0
DARK							½				2			2		½		½
STEEL		½	½	½		2							2				½	2
FAIRY		½					2	½							2	2	½	

Attack–defense matchups produce multipliers of 2× (super-effective), 1× (neutral), 0.5× (resisted), or 0× (immune). If a target is dual-typed, these factors multiply (e.g., 2× × 2× = 4× if both types are weak).

Additionally, if the move's type matches the user's type, a Same Type Attack Bonus (STAB) of 1.5× is applied.

This network of multiplicative modifiers fosters strategic coverage planning: Grass is strong against Water/Ground (Phanpy, for instance), but vulnerable to Fire. Over time, new types like Fairy were introduced to balance previously-dominant Dragon types. Thus, the type chart acts like a graph with weighted edges, inviting deeper spectral analysis.

Base Stats, IVs, EVs, and Natures

Each Pokémon's species is defined by base stats: HP, Attack, Defense, Special Attack, Special Defense, and Speed. For example**, Mewtwo's high base Special Attack far exceeds that of an unevolved Pokémon like Eevee.** On top of base stats, two hidden parameters shape final stats:

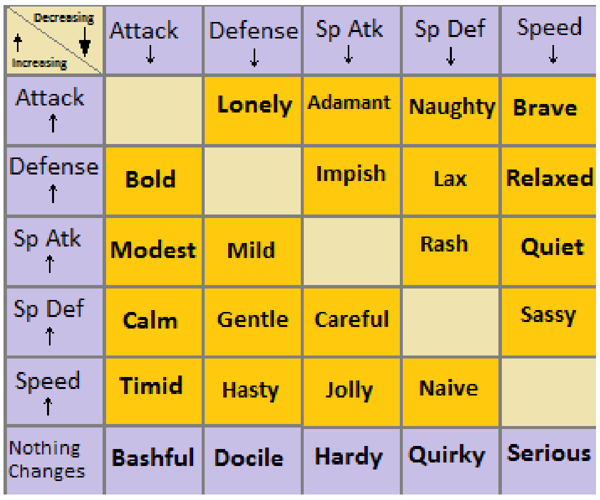
1. **Individual Values (IVs) (0–31 in each stat):** A genetically "seeded" bonus. At Level 50, each ~2 IV points is ~+1 final stat. Competitive players usually aim for perfect IVs (31) in crucial stats (Attack, Speed, etc.).

2. **Effort Values (EVs):** Up to 510 total can be distributed, max 252 per stat. Roughly every 8 EV at Level 50 adds +1 in that stat. This allows customizing Pokémon—e.g.**, a Pancham can be made extremely offensive by giving 252 Attack EVs (doubling its Attack from ~78 to ~147 at Level 50).**

A Pokémon's Nature modifies a chosen stat by +10% and another by –10%. For example, an Adamant nature (+Atk, –SpA) suits a physically-oriented Pokémon like Pancham. Combining these factors, the final stat formula for (non-HP) at Level 50 is often represented as:

$$\text{Stat} = \lfloor(\frac{(2 \times \text{Base} + \text{IV} + \lfloor \text{EV}/4 \rfloor) \times 50}{100} + 5)\rfloor \times \text{NatureMod.}$$

3. Turn Order: Speed and Priority



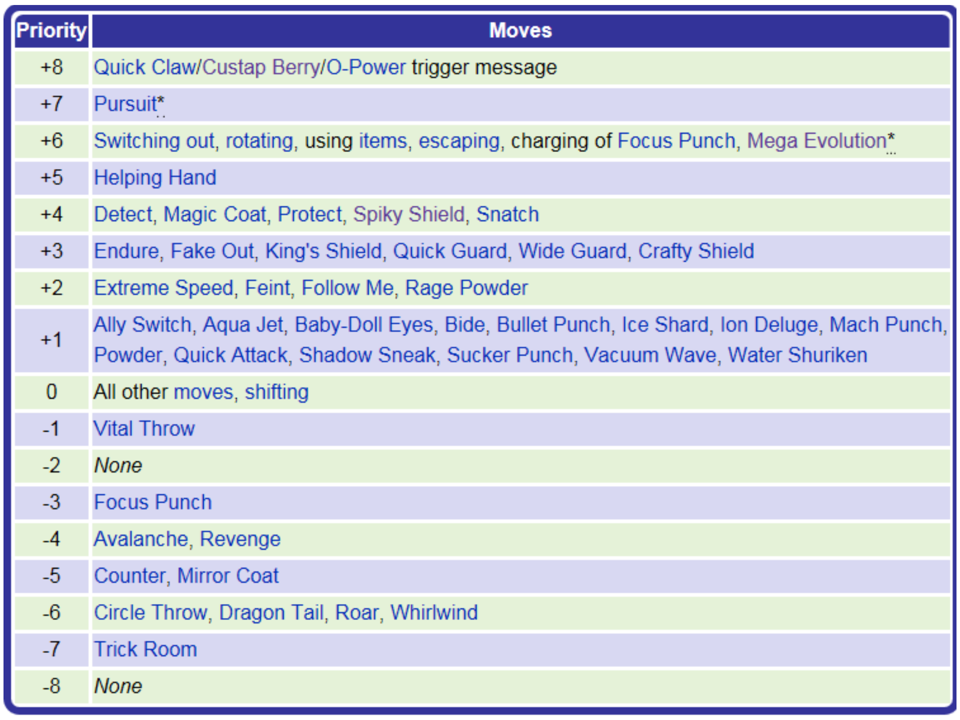
Decreasing ↓ / ↑ Increasing	Attack ↓	Defense ↓	Sp Atk ↓	Sp Def ↓	Speed ↓
Attack ↑		Lonely	Adamant	Naughty	Brave
Defense ↑	Bold		Impish	Lax	Relaxed
Sp Atk ↑	Modest	Mild		Rash	Quiet
Sp Def ↑	Calm	Gentle	Careful		Sassy
Speed ↑	Timid	Hasty	Jolly	Naive	
Nothing Changes	Bashful	Docile	Hardy	Quirky	Serious

After stats are set, battles revolve around simultaneous moves. The core sequence is:

1. **Check Priority:** Moves have inherent priority brackets (+3, 0, −7, etc.). A move at priority +4 (e.g., Protect) always goes before a +0 move.

2. **Within each Priority bracket, compare Speed:** The faster Pokémon acts first. If two speeds match, it's a coin flip.

Hence, a slow Pokémon with a +1 priority move can outspeed a faster opponent at 0 priority. Speed control via moves like Tailwind (double Speed for four turns) or Trick Room (reverse Speed order for five turns) is vital in doubles. Trick Room has priority −7, so it typically resolves last in that turn.



Priority	Moves
+8	Quick Claw/Custap Berry/O-Power trigger message
+7	Pursuit*
+6	Switching out, rotating, using items, escaping, charging of Focus Punch, Mega Evolution*
+5	Helping Hand
+4	Detect, Magic Coat, Protect, Spiky Shield, Snatch
+3	Endure, Fake Out, King's Shield, Quick Guard, Wide Guard, Crafty Shield
+2	Extreme Speed, Feint, Follow Me, Rage Powder
+1	Ally Switch, Aqua Jet, Baby-Doll Eyes, Bide, Bullet Punch, Ice Shard, Ion Deluge, Mach Punch, Powder, Quick Attack, Shadow Sneak, Sucker Punch, Vacuum Wave, Water Shuriken
0	All other moves, shifting
-1	Vital Throw
-2	*None*
-3	Focus Punch
-4	Avalanche, Revenge
-5	Counter, Mirror Coat
-6	Circle Throw, Dragon Tail, Roar, Whirlwind
-7	Trick Room
-8	*None*

Status Conditions and Probability

Status ailments—Burn, Paralysis, Sleep, Poison, Freeze—add random and/or deterministic penalties:

- **Burn:** Halves physical Attack, plus minor chip damage each turn.
- **Paralysis:** Halves Speed and imposes a 25% chance of full immobility each turn.

- **Sleep**: 1–3 turns of no action, random duration.
- **Poison:** Gradual HP loss (1/8 each turn for regular Poison, or increasing “Toxic” damage).
- **Freeze:** 20% chance to thaw each turn, otherwise no action indefinitely.

These probabilities and multipliers shape optimization: e.g., Will-O-Wisp’s guaranteed Burn vs. Scald’s 30% Burn chance. The math can be seen as discrete-time Markov processes with absorbing states, but the essential takeaway is each status forces players to weigh risk.

The Damage Formula

Pokémon’s damage formula is a classic:

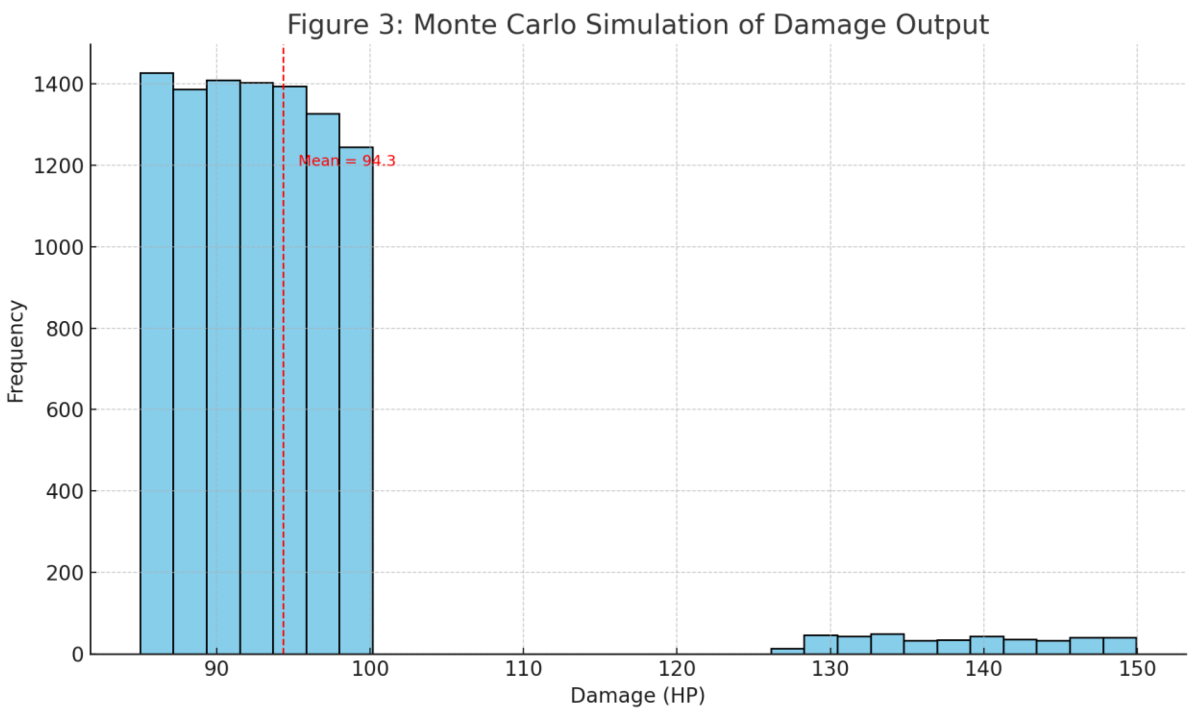
$$\text{Damage} = \left(\left(\frac{2 \times L}{5} + 2\right) \times \frac{A \times P}{D} \middle/ 50 + 2\right) \times \text{Modifier.}$$

- **L**: Level (50 in VGC)
- **A:** Attacker’s relevant Attack or SpA
- **P:** Move’s base power
- **D:** Defender’s relevant Defense or SpD
- **Modifier:** STAB × type effectiveness × critical × random(0.85–1.00) × item × ability × weather, etc.

A typical scenario: *“Does my 252 Attack, Adamant Pancham’s Close Combat OHKO a 252 HP Mewtwo?”* is answered by plugging in each factor: e.g., is it super-effective (×2)? Does user have an Attack boost? The random multiplier alone can produce anywhere from 85% to 100% of the “expected” damage. In a tight match, that difference could decide victory or defeat.

Move Mechanics, Field Effects, and Spreads

- **Spread Moves (Rock Slide, Earthquake, Heat Wave):** Hit multiple foes but each hit is scaled by 0.75×. If one foe is immune/protected, the move still counts as a spread for the other (0.75×).
- **Reflect/Light Screen/Aurora Veil:** Halve (or 0.67× in doubles) physical/special damage for five turns.
- **Weathers (Rain, Sun, Sandstorm, Snow) and Terrains (Electric, Grassy, Misty, Psychic)** impose further multipliers, e.g., **Rain** boosting **Water 1.5×** and **halving Fire.**
- **Priority:** Certain moves (e.g. Fake Out +3) overshadow Speed. Psychic Terrain blocks all priority for grounded Pokémon.



Dracovish Case Study: A famous destructive set

Dracovish @ Choice Band

Ability: Strong Jaw

EVs: 252 Atk / 4 SpD / 252 Spe

Adamant Nature

- Fishious Rend

- Psychic Fangs

- Crunch

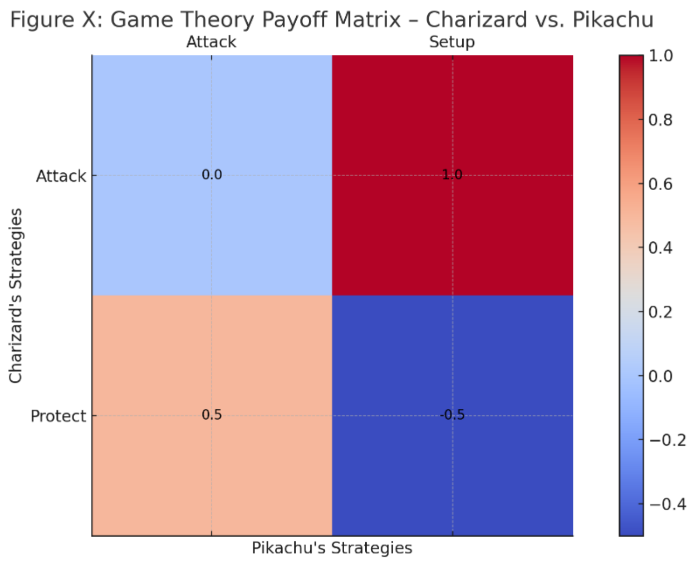
- Outrage

- Strong Jaw boosts biting moves by 1.5×.
- Fishious Rend doubles in base power if Dracovish moves first.
- Choice Band multiplies Attack by 1.5× but locks the user into its first chosen move.

Aside from being my favourite Pokemon, if **Dracovish** is faster, **Fishious Rend** effectively gets **2**× from "going first," then **1.5× from Strong Jaw,** plus **1.5× from Band**, plus **1.5× from STAB (Water move used by Water type).** In best-case scenarios, **that's**

2×1.5×1.5×1.5 = 6.75× normal damage. Many defensive walls can't withstand that. This synergy demonstrates how stacked multipliers can break standard calculations. The flipside is that if Dracovish is slower or Intimidated, it loses the unstoppable advantage, so speed control or synergy is key.

Game Theory and Mixed-Strategy Nash Equilibria in Move Selection



Pokémon battles are often likened to a combination of chess and poker–there is deep strategy but also hidden information and probabilistic outcomes. Each turn in a Pokémon VGC match, both players choose their moves simultaneously. This simultaneous decision-making can be modeled with **game theory** as a series of one-turn games. Each player has a set of possible actions (attacks, status moves, switching Pokémon, protecting, etc.), and the outcome of a turn depends on the *pair* of actions chosen.

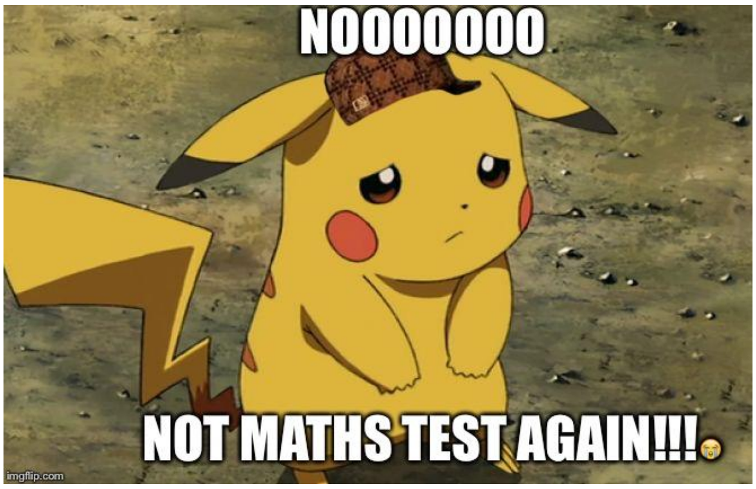
We can represent a simplified scenario with a **payoff matrix**: one player's options as rows and the opponent's as columns, with each cell describing the result (or a numerical payoff like chance to win). These simultaneous-move interactions often resemble classic examples like rock–paper–scissors in that there is **no single "best" move unilaterally** – if you become predictable, your opponent can counter-exploit you.

Nash equilibrium is the central game-theoretic concept that captures this idea of optimal mixed play. Formally, a Nash equilibrium is a strategy combination where neither player can improve their expected outcome by deviating alone. In other words, each player's strategy is the best response to the other's. In Pokémon terms, a Nash equilibrium in move selection means **each player has optimized their move choices (including some randomness) such that the opponent cannot gain an edge by guessing or countering differently**.

Often, this equilibrium is realized by **mixed strategies**, where a player randomizes over multiple moves with certain probabilities. By mixing moves, you become less predictable – the opponent cannot reliably exploit any single pattern. As one competitive analysis puts it, *"you can adapt a mixed strategy"*, e.g. mostly using a strong attack but occasionally surprising the opponent with a status or setup move. Likewise, the opponent might mostly make the safe play (say, switching out a threatened Pokémon or using Protect) but sometimes do the unexpected (stay in and attack when you assumed they would switch).

When both sides mix optimally, they reach a **balance where each is indifferent to the other's choices** – this is the Nash equilibrium. No player can unilaterally change their probability mix to increase their chance of winning.

To make this concrete, consider a one-on-one scenario: **Charizard vs. Pikachu**, late in a game. Charizard (Fire/Flying) is low on health and **fears Pikachu's Thunderbolt** (an Electric attack that would be super-effective). Pikachu, meanwhile, fears a potential knockout from Charizard's powerful Fire attack if it doesn't KO Charizard first. Charizard has two main options: **Attack** (try to KO Pikachu before it moves) or **Protect** (shield itself for one turn, expecting Pikachu's Thunderbolt).



Pikachu's options could be **Attack** (use Thunderbolt) or **Status/Setup** (for example, use *Agility* to boost speed, or even switch to a different Pokémon anticipating a Protect). Now, how do they choose? If Charizard always attacks, Pikachu will quickly learn to *always* use Thunderbolt – Charizard would get knocked out first in that race. If Charizard always protects on the obvious Thunderbolt-turn, Pikachu could take advantage by using Agility on that turn (wasting Charizard's Protect and gaining a huge advantage for the following turn). There is **no single deterministic strategy** that works best for Charizard – a predictable pattern will be exploited by a skilled opponent. The same holds for Pikachu. This is exactly the kind of situation game theory resolves: **randomize the choices**.

Charizard might decide, for example, to Protect *say 40% of the time* and Attack 60% of the time in this scenario. Pikachu might, in response, Attack 60% and go for a setup or switch

40%. These probabilities would be chosen such that each player is indifferent to the opponent's mix. If Charizard Protects too often, Pikachu will start capitalizing by not attacking (since Protect only blocks damage and confers no advantage if the opponent wasn't attacking).

If Charizard attacks too often, Pikachu will revert to the safe Thunderbolt most of the time. There is a balance point – the **mixed-strategy Nash equilibrium** – where, perhaps Charizard protecting 50% and attacking 50%, and Pikachu attacking 50% and using a setup 50%, neither can improve their *overall* survival or win chances by shifting those percentages. At that equilibrium, Charizard's expected outcome from attacking or protecting is the same (so it's indifferent, and thus mixing is optimal), and similarly for Pikachu's options. In short, each choice is made just often enough to keep the opponent honest.

We can analyze such situations with actual payoff values. Suppose if both **Charizard and Pikachu attack**, Charizard might faint first (Pikachu's Thunderbolt lands before Charizard's attack if Pikachu is faster or if Charizard is severely low HP), resulting in a loss for Charizard – call this outcome value 0 for Charizard (bad). If **Charizard attacks while Pikachu tries to set up** (expecting a Protect), Charizard scores a KO – a great outcome (value 1 for Charizard).

If **Charizard protects while Pikachu attacks**, Charizard successfully blocks damage (gaining a small advantage, say 0.5, since now Pikachu's threat is revealed and Charizard gets another chance next turn). If **both Charizard and Pikachu choose non-damaging moves** (Charizard protects and Pikachu sets up), nothing immediately happens except Pikachu gets a boost – an unfavorable outcome for Charizard (say value -0.5). We could construct a payoff matrix with these hypothetical values. Solving for the equilibrium (by equalizing the expected payoff of Charizard's Attack vs Protect and of Pikachu's Attack vs Setup) would yield the optimal mix probabilities (in this rough example, it might be around "attack 70%" for one and "protect 30%" etc.).

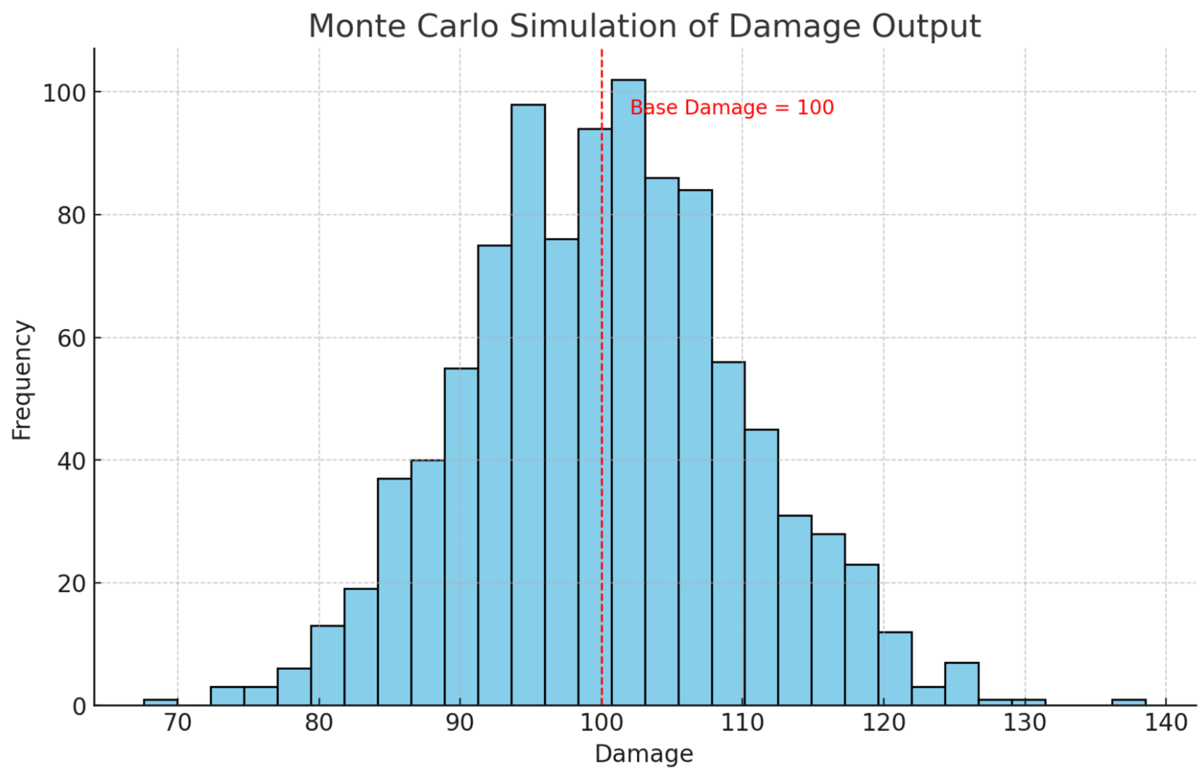
The **key result** is that in equilibrium each player is choosing a probabilistic strategy that *"if the opponent knows the strategy, they still can't exploit it by changing their own"*. This mixture prevents exploitation. Indeed, high-level Pokémon players often talk about "mixing it up" or making plays with a certain frequency – this is an intuitive use of mixed strategies. For instance, a player might note that their **Mewtwo** (a famously fast and strong Pokémon) wins a direct confrontation against the opponent’s Mewtwo only 50% of the time due to a **speed tie** (a random coin flip who moves first). Rather than always take that coin flip by just attacking, the player may occasionally use a different tactic – perhaps have Mewtwo use *Protect* or *Disable* on the first turn to disrupt an expected attack, or switch to a sturdier Pokémon. By doing so unpredictably, the player avoids a pure coin-flip each time and forces the opponent to respond to a broader strategy. In summary, **game theory** in Pokémon teaches players to *randomize their choices in crucial guess situations*, achieving a **mixed-strategy Nash equilibrium** that makes them **unexploitable on average**. This elevates play from simple tactics to a probabilistic mind-game – much like a poker player mixing bluffs and honest plays – and is a big part of what makes competitive Pokémon intellectually rich.

Monte Carlo Simulation of Damage Ranges

One of the iconic aspects of Pokémon battles is the **damage formula** – a somewhat complex calculation that determines how much HP a move removes from the target. This formula includes many variables: the attacking Pokémon’s level and relevant Attack stat, the defending Pokémon’s corresponding Defense stat, the base power of the move, and a host of modifiers.

In simplified form, **Damage** is computed by first calculating a base amount from level, stats, and power, then multiplying by modifiers for things like **Same-Type Attack Bonus (STAB)** (typically 1.5× if the move’s type matches the user’s type), **type effectiveness** (e.g. 0 for no effect, 0.5 for “not very effective”, 2 for “super effective”, etc.), **critical hits**, weather or item boosts, and finally a uniform **random factor** in the range [0.85, 1.00]. The presence of that random 15% range means that even with identical conditions, damage rolls can slightly vary from hit to hit.

For example, a move might do *85 damage on one use and 100 on the next*, purely due to luck. Critical hits (typically ~4% chance) add another layer of variance: a critical hit multiplies damage by 1.5× (in recent generations) and ignores certain defensive boosts. Secondary effects (like a 10% chance to burn the target, which over time causes additional HP loss) introduce further randomness in how much “effective damage” a move contributes in the long run. All these random elements make Pokémon damage an ideal candidate for **Monte Carlo simulation** to understand the distribution of possible outcomes.

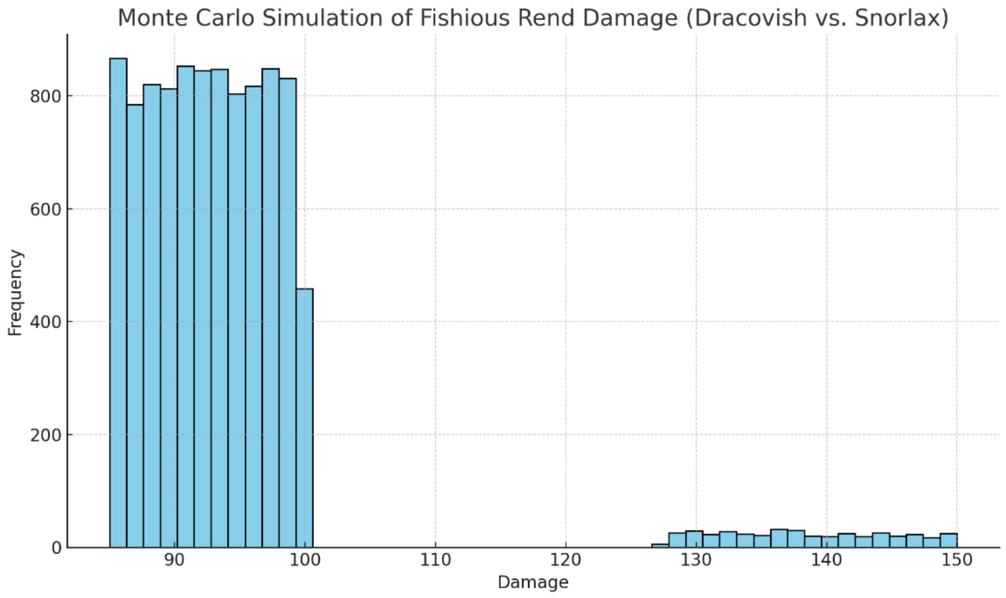


Monte Carlo methods are computational algorithms that *"rely on repeated random sampling to obtain numerical results"*, using randomness to solve problems that might be deterministic in principle. In our context, the problem is deterministic given specific inputs (if we fixed the random roll at its average, we'd get a single damage value), but the built-in game randomness makes the actual outcome uncertain. We can simulate this by sampling the random components many times and observing the results. Essentially, we **repeat the damage calculation thousands of times**, each time with random values for the damage modifier (and random events like critical hits or status conditions occurring or not according to their probabilities). By doing this, we empirically build up the **probability distribution** of damage outcomes. The steps are straightforward:

1. **Define the scenario inputs**: Choose the attacking Pokémon, defending Pokémon, and move. For example, consider the move **Fishious Rend used by Dracovish**. Dracovish (a Water/Dragon type from Generation VIII) is infamous for its ability to deal massive damage, especially when moving first (Fishious Rend doubles in power if the user strikes before the target). Let's pit Dracovish against a **bulky defensive Pokémon** like Snorlax – Snorlax has a high HP stat and

decent defenses, making it a sturdy target. We fix all the known values: Dracovish's Attack stat (say at level 50 with a strong Attack investment), Snorlax's Defense stat, Fishious Rend's base power (85, doubled to 170 if Dracovish goes first), plus modifiers like STAB (1.5× since Dracovish is Water-type using a Water move), possibly a held item boost (Choice Band for +50% perhaps), and type effectiveness (Water vs Normal is neutral 1×). These give us the deterministic parts of the damage formula.

2. **Identify the random variables**: In damage calculation, the randomness comes from the damage **random factor (Uniform[0.85,1.00])** and the **critical hit chance (~4.17%)**. (For completeness, if we were simulating a move like *Thunderbolt*, we'd also include the 10% chance to paralyze, but Fishious Rend has no secondary effect, so we focus on damage and crits.)
3. **Simulate many trials**: We then programmatically or conceptually simulate, say, **10,000 uses of Fishious Rend** under these conditions. Each trial, we sample a fresh random damage multiplier between 0.85 and 1.0 and decide whether a critical hit occurs (with 4.17% probability). We plug these into the damage formula to get a damage outcome (an integer number of HP damage).
4. **Aggregate the results**: After thousands of runs, we can calculate statistics and visualize the distribution. We might get, for example, an **average damage** of around, say, 85 HP (out of Snorlax's perhaps ~150 HP at level 50). We can also determine the



spread: perhaps a **minimum** of about 75 damage (if the lowest roll occurs with no crit) and a **maximum** of around 130 damage (if a critical hit occurs on a max roll).

5. We can build a histogram of damage frequencies which might show a cluster of outcomes in the 80–100 range and a small separate cluster out past 120 representing the critical hits. From this distribution, we can answer strategic questions: **What is the probability that Dracovish's attack one-shots Snorlax?** In this simulated example, virtually 0% – even the highest typical non-crit damage (around 100) doesn't reach Snorlax's full HP. Only a lucky critical hit (around 130 damage) could KO, and that combined chance (crit *and* high roll) might be on the order of 1–2%. We could thus say there is a ~98–99% chance Fishious Rend **fails to OHKO** Snorlax. However, the simulation might show it **always 2HKO**s: even the lowest roll (~75) is about half of Snorlax's HP, so two hits will definitely KO. We can also compute a **confidence interval** for the *mean* damage: with 10,000 trials, the standard error is small, so we might report something like "the average damage is 85 ± 1 HP at 95% confidence" – but more practically, we care about the range of outcomes. We find that **95% of the time, the damage will fall between say 78 and 100 HP** (the 2.5th and 97.5th percentiles of our simulation), with the remaining ~5% of outcomes being those critical-hit extremes above 100 HP. This gives a very concrete picture of what to expect from the move.

The Monte Carlo approach is extremely useful because the Pokémon damage formula, while deterministic in structure, yields a **distribution of possible results** due to in-game randomness. Rather than crunching the formula algebraically, simulation lets us **approximate the probability of various outcomes**. This has direct strategic implications. For instance, if our simulation showed, say, a 30% chance to OHKO the target, a player might decide whether to gamble on that 30% for an immediate win or to play more conservatively. In our Dracovish vs. Snorlax case, since the OHKO chance is essentially 0, the player knows **with high confidence** that they will need two Fishious Rends to secure the KO. They might plan accordingly: perhaps use Fishious Rend once, expect Snorlax to survive, and have a follow-up plan (like a second Pokémon to finish Snorlax, or be ready to endure Snorlax's counterattack). The simulation also highlights the impact of **critical hits** – those few outlier trials where the damage spiked to ~130. If the battle situation is desperate, a player might "play to their outs" by hoping for a critical hit to score an unexpected OHKO. Knowing the roughly ~4% crit chance, they can weigh that slim possibility. Furthermore, Monte Carlo simulations can incorporate *additional randomness* beyond a single move's damage. For example, one could simulate a **full battle scenario** turn by turn, or evaluate a complex tactic .

In a more academic sense, Monte Carlo simulation provides a numerical confirmation of the theoretical expectations. The law of large numbers guarantees that as we simulate more trials, the **empirical average damage will converge** to the true expected value, and we can make the confidence interval for the mean as tight as we want by increasing the number of trials. In a Pokémon context, this means our estimates of a move's performance become very reliable with enough simulation. For competitive players, such simulations (often implicitly done via **damage calculators** or tools) are routine – they speak of **"damage rolls"** and percentages when discussing whether a certain attack will KO a certain Pokémon. Our example demonstrates how those percentages can be obtained. By simulating Fishious Rend 10,000 times, we found perhaps that it does, say, *55% – 65% of Snorlax's HP 95% of the time*. So a player might say *"Fishious Rend is a guaranteed 2HKO on Snorlax, and never a 1HKO barring a lucky crit."* This kind of language shows a probabilistic understanding of damage, exactly what Monte Carlo analysis provides.

CONCLUSON

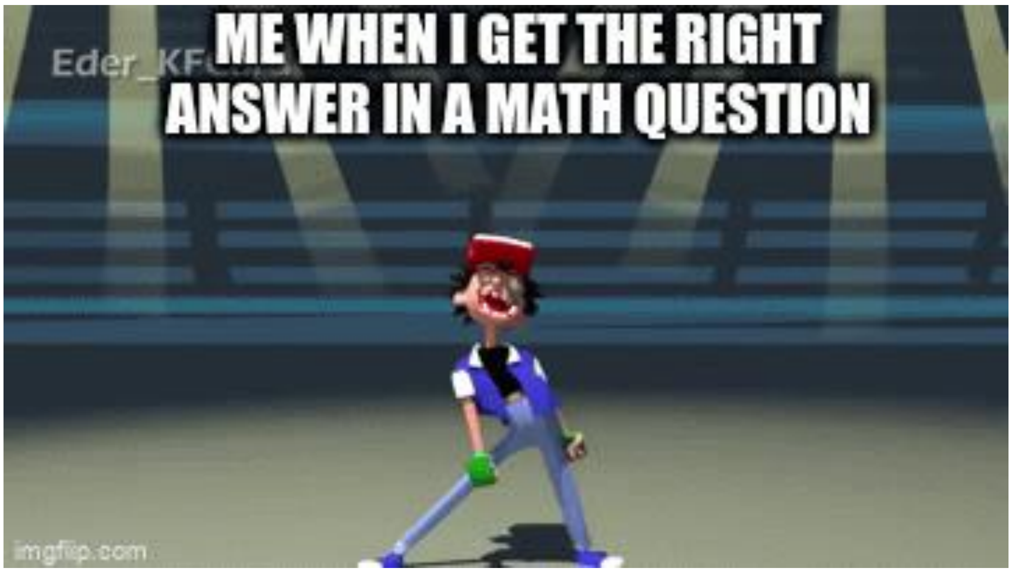
Competitive Pokémon VGC is more than a game—it is a dynamic and deeply strategic arena where mathematics becomes a language of battle, prediction, and innovation. From the deceptively simple type matchups to the nuanced layers of EV training, IV spreads, move priorities, damage variance, and probability management, every decision is grounded in quantifiable structures. This essay has explored the game not merely as entertainment, but as a simulation of real-world strategy problems, solvable through tools like game theory, probability distributions, Monte Carlo methods, and spectral graph theory.

Yet, what makes Pokémon truly fascinating is how this mathematical elegance intertwines with human intuition. You don't have to be a top-tier player to appreciate the brilliance of a low-tier Pokémon like Pachirisu winning Worlds—not because it defied logic, but because it embraced a different logic. The game celebrates creativity within constraint, rewarding both the optimizer and the outlier.

In building competitive teams and calculating outcomes, players become experimental mathematicians, navigating a web of decisions under uncertainty. Concepts like mixed-strategy Nash equilibria and type network adjacency matrices aren't just theoretical constructs—they're part of the lived reality of top players who must balance consistency with surprise, synergy with singularity, and math with mind games.

For me, a long-time enthusiast of both math and Pokémon, this convergence has offered a playground where abstract formulas find real meaning. Pokémon VGC has not only honed my strategic thinking but shown me how mathematical thought can elevate play into art—and art into science.

In the end, whether you're optimizing Dracovish's Fishious Rend damage or calculating the best nature for a speedy Eevee, you're engaging in an elegant dance of logic and unpredictability. A dance that proves, perhaps more than any other game, that Pokémon truly is the most mathematical game ever made.



References

1. Bulbapedia. 2024. *Damage Calculation (Generation VIII)*
2. Bulbapedia. 2024. *Effort Values (EVs)*
3. Bulbapedia. 2024. *Individual Values (IVs)*
4. Glick, Wolfe. 2023
5. Pokémon Company International. 2024. *Official VGC Ruleset and Regulations*
6. Skyrms, Brian. 2004. *The Stag Hunt and the Evolution of Social Structure*. Cambridge: Cambridge University Press.
7. Nash, John. 1950. "Equilibrium Points in n-Person Games." *Proceedings of the National Academy of Sciences* 36 (1): 48–49.
8. Motwani, Rajeev, and Prabhakar Raghavan. 1995. *Randomized Algorithms*. Cambridge: Cambridge University Press.
9. Horn, Roger A., and Charles R. Johnson. 2012. *Matrix Analysis*. 2nd ed. Cambridge: Cambridge University Press.
10. Chung, Fan R. K. 1997. *Spectral Graph Theory*. CBMS Regional Conference Series in Mathematics, No. 92. American Mathematical Society.
11. Serebii.net. 2025. *Competitive Pokémon Data and Tier Lists*. Accessed April 6, 2025.
12. Psacake. 2024. *Pokémon Showdown Damage Calculator (Updated)*. Accessed April 6, 2025. https://calc.pokemonshowdown.com.