Inteligencja roju: cząsteczki

Co to jest optymalizacja roju cząstek?

Optymalizacja roju cząstek to kolejny algorytm roju. Inteligencja roju polega na wyłaniającym się zachowaniu wielu jednostek, aby wspólnie rozwiązywać trudne problemy. W rozdziale 6 widzieliśmy, jak mrówki mogą znaleźć najkrótsze ścieżki między miejscami docelowymi dzięki wykorzystaniu feromonów. Stada ptaków są kolejnym idealnym przykładem inteligencji roju w przyrodzie. Kiedy pojedynczy ptak leci, może wykonać kilka manewrów i technik w celu zachowania energii, takich jak skakanie i szybowanie w powietrzu lub wykorzystywanie prądów wiatru do unoszenia go w kierunku, w którym zamierza podróżować. To zachowanie wskazuje na pewien prymitywny poziom inteligencji u jednej osoby. Ale ptaki również muszą migrować w różnych porach roku. Zimą jest mniej owadów i innego pokarmu. Brakuje również odpowiednich miejsc lęgowych. Ptaki mają tendencję do gromadzenia się w cieplejszych obszarach, aby skorzystać z lepszych warunków pogodowych, co poprawia szanse przeżycia. Migracja zwykle nie jest krótką podróżą. Potrzeba tysięcy kilometrów ruchu, aby dotrzeć do obszaru o odpowiednich warunkach. Kiedy ptaki pokonują tak duże odległości, mają tendencję do gromadzenia się. Ptaki gromadzą się, ponieważ w walce z drapieżnikami jest siła; dodatkowo oszczędza energię. Formacja, którą obserwujemy w stadach ptaków, ma kilka zalet. Duży, silny ptak przejmie inicjatywę, a gdy zacznie trzepotać skrzydłami, spowoduje uniesienie ptaków za nim. Te ptaki potrafią latać, zużywając znacznie mniej energii. Stada mogą zmienić przywódców, jeśli zmieni się kierunek lub jeśli przywódca stanie się zmęczony. Kiedy określony ptak wychodzi z formacji, ma większe trudności w lataniu z powodu oporu powietrza i koryguje swój ruch, aby powrócić do formacji.

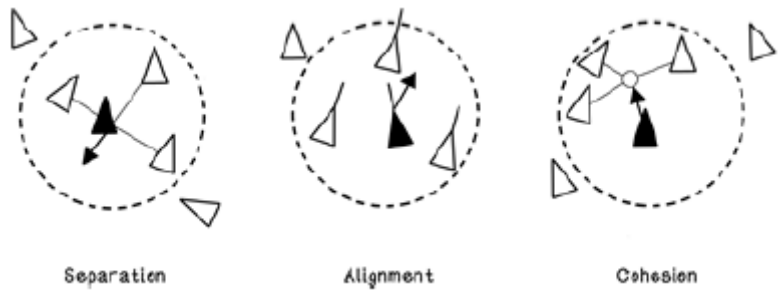
Craig Reynolds opracował program symulatora w 1987 roku, aby zrozumieć atrybuty pojawiającego się zachowania w stadach ptaków i wykorzystał następujące zasady, aby pokierować grupą. Zasady te pochodzą z obserwacji stad ptaków:

• Wyrównanie - jednostka powinna kierować się w średnim kierunku do swoich sąsiadów, aby upewnić się, że grupa podróżuje w podobnym kierunku.

• Spójność - jednostka powinna zbliżyć się do przeciętnej pozycji swoich sąsiadów, aby utrzymać formację grupy.

• Separacja - jednostka powinna unikać tłoczenia się lub kolizji z sąsiadami, aby upewnić się, że osoby nie zderzają się, co przeszkadza grupie.

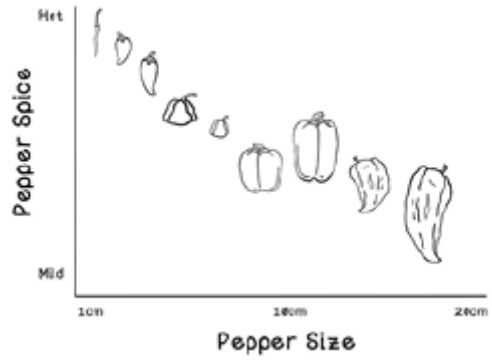
Dodatkowe reguły są używane w różnych wariantach prób symulowania zachowania roju. Rysunek ilustruje zachowanie jednostki w różnych scenariuszach, a także kierunek, w którym wpływa na nią, aby postępować zgodnie z odpowiednią regułą. Regulacja ruchu jest równowagą tych trzech zasad przedstawionych na rysunku.



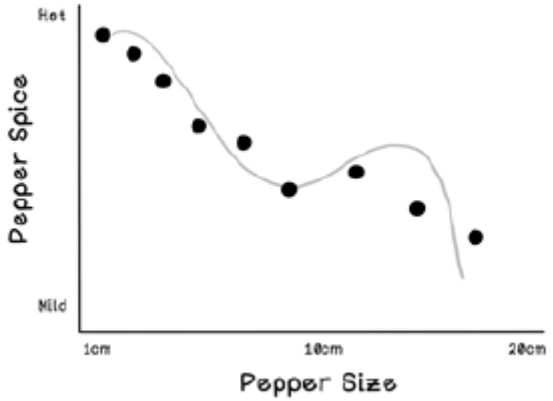
Optymalizacja roju cząstek obejmuje grupę osób w różnych punktach przestrzeni rozwiązania, wszystkie wykorzystujące rzeczywiste koncepcje roju, aby znaleźć optymalne rozwiązanie w przestrzeni. Ten rozdział zagłębia się w działanie algorytmu optymalizacji roju cząstek i pokazuje, jak można go wykorzystać do rozwiązywania problemów. Wyobraź sobie rój pszczół, który rozprzestrzenia się w poszukiwaniu kwiatów i stopniowo zbiera się na obszarze o największej gęstości kwiatów. Im więcej pszczół znajduje kwiaty, tym więcej kwiatów przyciągają. Podstawą tego przykładu jest optymalizacja roju cząstek. Problemy z optymalizacją zostały wspomniane w kilku rozdziałach. Znalezienie optymalnej ścieżki w labiryncie, ustalenie optymalnych przedmiotów dla plecaka i znalezienie optymalnej ścieżki między atrakcjami w karnawale to przykłady problemów optymalizacyjnych. Przerabialiśmy je bez zagłębiania się w szczegóły za nimi. Jednak od tego rozdziału ważne jest głębsze zrozumienie problemów optymalizacji. Rozdział 7.2 opiera się na intuicji, aby móc wykryć problemy optymalizacji, gdy się pojawią.

Problemy z optymalizacją: nieco bardziej techniczna perspektywa

Załóżmy, że mamy kilka papryczek różnej wielkości. Zwykle mała papryka jest ostrzejsza niż duża. Jeśli wykreślimy wszystkie papryki na wykresie opartym na wielkości i ostrości, może to wyglądać jak rysunek



Rysunek przedstawia wielkość każdej papryki i jej ostrego smaku. Teraz, usuwając obrazy papryki, wykreślając punkty danych i rysując możliwą krzywą między nimi, pozostaje rysunek. Gdybyśmy mieli więcej papryki, mielibyśmy więcej punktów danych, a krzywa byłaby dokładniejsza.

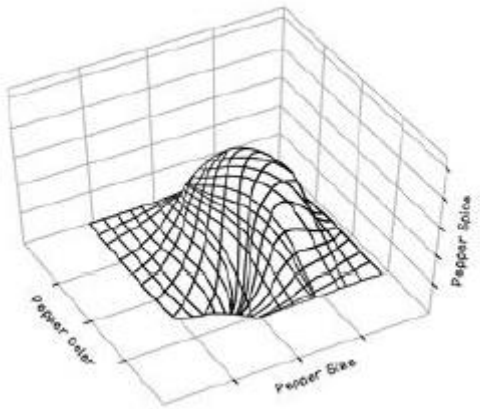


Ten przykład może potencjalnie stanowić problem optymalizacji. Gdybyśmy szukali minimum od lewej do prawej, natrafilibyśmy o kilka punktów mniej niż poprzednie, ale w środku natrafilibyśmy na wyższy. Powinniśmy przestać? Gdybyśmy to zrobili, stracilibyśmy rzeczywiste minimum, które jest ostatnim

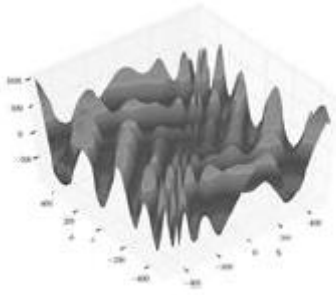
punktem danych, znanym jako minimum globalne. Przybliżoną linię / krzywą trendu można przedstawić za pomocą funkcji, takiej jak ta pokazana na rysunku. Tę funkcję można interpretować jako równą ostrość papryki do wyniku tej funkcji, gdzie rozmiar pieprzu jest reprezentowany przez x.

$$f(x) = -(x-4)(x-0.2)(x-2)(x-3)+5$$

Rzeczywiste problemy zwykle obejmują tysiące punktów danych, a minimalny wynik funkcji nie jest tak jasny, jak w tym przykładzie. Przestrzenie poszukiwań są ogromne i trudne do rozwiązania ręcznie. Zauważ, że użyliśmy tylko 2 właściwości pieprzu do stworzenia punktów danych, co dało prostą krzywą. Jeśli weźmiemy pod uwagę inną właściwość pieprzu, taką jak kolor, reprezentacja danych znacznie się zmienia. Teraz wykres musi być przedstawiony w 3D, a trend staje się powierzchnią zamiast krzywej. Powierzchnia jest jak wypaczony koc w trzech wymiarach. Ta powierzchnia jest również reprezentowana jako funkcja, ale jest bardziej złożona.



Ponadto przestrzeń wyszukiwania 3D może wyglądać na dość prostą, lub być tak złożona, że próba jej wizualnego sprawdzenia w celu znalezienia minimum jest prawie niemożliwa.



Oto funkcja, która reprezentuje tę płaszczyznę:

$$f(x,y) = -(y+47)\sin\sqrt{\left|\frac{x}{2}+(y+47)\right|} - x\sin\sqrt{|x-(y+47)|}$$

Robi się ciekawiej! Przyjrzeliśmy się trzem cechom papryki: jej wielkości, kolorowi i ostrości. W rezultacie szukamy w 3 wymiarach. A co jeśli chcemy uwzględnić lokalizację wzrostu? Ten atrybut jeszcze bardziej utrudniłby wizualizację i zrozumienie danych, ponieważ szukamy w 4 wymiarach. Jeśli dodamy wiek papryki i ilość nawozu użytego podczas jej uprawy, otrzymamy ogromną przestrzeń

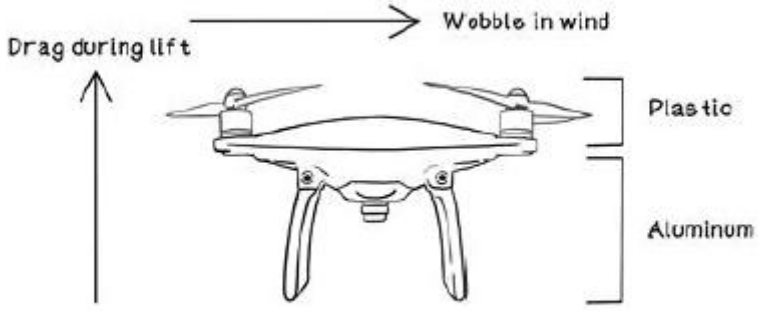
poszukiwań w 6 wymiarach i nie możemy sobie wyobrazić, jak może wyglądać to wyszukiwanie. To poszukiwanie również jest reprezentowane przez funkcję, ale znowu jest zbyt złożone i trudne do rozwiązania dla osoby. Algorytmy optymalizacji roju cząstek są szczególnie dobre w rozwiązywaniu trudnych problemów optymalizacyjnych. Cząsteczki są rozproszone w wielowymiarowej przestrzeni poszukiwań i współpracują, aby znaleźć dobre maksima lub minimum. Algorytmy optymalizacji roju cząstek są szczególnie przydatne w następujących scenariuszach:

• Duże przestrzenie wyszukiwania - istnieje wiele punktów danych i możliwości kombinacji.

• Przestrzenie wyszukiwania o dużych wymiarach - w dużych wymiarach występuje złożoność. Znalezienie dobrego rozwiązania wymaga wielu wymiarów problemu.

Problemy dotyczące optymalizacji roju cząstek

Wyobraź sobie, że opracowujemy drona, a jego korpus i skrzydła śmigła (łopatki, które sprawiają, że latają) są wykorzystywane do wykonania jego korpusu. Dzięki wielu próbom badawczym odkryliśmy, że różne ilości dwóch określonych materiałów dają różne wyniki pod względem optymalnej wydajności podnoszenia drona i oporu przy silnym wietrze. Te dwa materiały to aluminium na obudowę i plastik na łopatki. Za dużo lub za mało któregokolwiek materiału spowoduje słabe działanie drona. Ale kilka kombinacji daje dobry dron, a tylko jedna kombinacja daje wyjątkowo dobry dron. Rysunek 7.10 ilustruje komponenty wykonane z tworzywa sztucznego i komponenty wykonane z aluminium. Strzałki ilustrują siły, które wpływają na wydajność drona. Mówiąc prościej, chcemy znaleźć dobry stosunek plastiku do aluminium dla wersji drona, która zmniejsza opór podczas podnoszenia i zmniejsza kołysanie się na wietrze. Tak więc plastik i aluminium są wejściami, a wyjściem jest wynikająca z tego stabilność drona. Opiszmy idealną stabilność jako zmniejszenie oporu podczas startu i kołysania się na wietrze.



Good performance = Low drag & Less wobble in the wind

Ważna jest precyzja w stosunku aluminium do plastiku, a zakres możliwości jest duży. W tym scenariuszu naukowcy odkryli funkcję stosunku aluminium do plastiku. Będziemy używać tej funkcji w symulowanym środowisku wirtualnym, które testuje opór i kołysanie, aby znaleźć najlepsze wartości dla każdego materiału, zanim wyprodukujemy kolejny prototypowy dron. Wiemy również, że maksymalne i minimalne stosunki dla materiałów wynoszą odpowiednio 10 i -10. Ta funkcja dopasowania jest podobna do heurystyki. Rysunek przedstawia funkcję dopasowania dla stosunku aluminium (x) do tworzywa sztucznego (y). Rezultatem jest ocena wydajności oparta na oporze i kołysaniu, biorąc pod uwagę wartości wejściowe dla x i y.

$$f(x,y) = (x + 2y - 7)^2 + (2x + y - 5)^2$$

Jak możemy znaleźć ilość aluminium i ilość plastiku potrzebną do stworzenia dobrego drona? Jedną z możliwości jest wypróbowanie każdej kombinacji wartości dla aluminium i plastiku, aż znajdziemy najlepszy stosunek materiałów do naszego drona. Cofnij się o krok i wyobraź sobie ilość obliczeń potrzebnych do znalezienia tego współczynnika. Moglibyśmy przeprowadzić prawie nieskończoną liczbę obliczeń przed znalezieniem rozwiązania, gdybyśmy wypróbowali każdą możliwą liczbę. Musimy obliczyć wynik dla pozycji w tabeli. Zwróć uwagę, że liczby ujemne dla aluminium i plastiku są w rzeczywistości dziwaczne, jednak używamy ich w tym przykładzie, aby zademonstrować funkcję dopasowania używaną do optymalizacji tych wartości.

Ile części aluminiowych? (x): Ile części plastikowych? (y)

-0,1: 1,34

-0,134: 0,575

-1,1: 0,24

-1,1645: 1,432

-2,034: -0,65

-2,12: -0,874

0,743: -1,1645

0,3623: -1,87

1,75: -2,7756

…:…

-10 ≥ Aluminium ≥ 10: -10 ≥ Plastik ≥ 10

To obliczenie będzie kontynuowane dla każdej możliwej liczby między ograniczeniami i jest kosztowne obliczeniowo, więc realistycznie niemożliwe jest brutalne wymuszenie tego problemu. Potrzebne jest lepsze podejście. Optymalizacja roju cząstek umożliwia przeszukiwanie dużej przestrzeni wyszukiwania bez sprawdzania każdej wartości w każdym wymiarze. W przypadku drona aluminium jest jednym wymiarem problemu, plastik jest drugim wymiarem, a wynikająca z tego wydajność drona jest trzecim wymiarem. W kolejnej sekcji określamy struktury danych wymagane do reprezentacji cząstki, w tym dane dotyczące problemu, który będzie zawierała.

Stan reprezentujący: jak wyglądają cząsteczki?

Ponieważ cząstki poruszają się w przestrzeni poszukiwań, należy zdefiniować pojęcie cząstki. Przedstawiające pojęcie cząstki:

• Pozycja - pozycja cząstki we wszystkich wymiarach

• Najlepsza pozycja - najlepsza pozycja znaleziona za pomocą funkcji fitness

• Prędkość - aktualna prędkość ruchu cząstki

Pseudo kod

Aby spełnić trzy atrybuty cząstki, w tym pozycję, najlepszą pozycję i prędkość, w konstruktorze cząstki wymagane są następujące właściwości dla różnych operacji algorytmu optymalizacji roju cząstek. Nie martw się teraz o bezwładność, komponent poznawczy i komponent społeczny; zostaną one wyjaśnione w kolejnych sekcjach.

```
Particle(x, y, inertia, cognitive_constant, social_constant):
  let particle.x equal to x
  let particle.y equal to y
  let particle.fitness equal to infinity
  let particle.velocity equal to 0
  let particle.best_x equal to x
  let particle.best_y equal to y
  let particle.best_fitness equal to infinity
  let particle.inertia equal to inertia
  let particle.cognitive_constant equal to cognitive_constant
  let particle.social_constant equal to social_constant
```

Cykl życia optymalizacji roju cząstek

Podejście do projektowania algorytmu optymalizacji roju cząstek opiera się na rozwiązaniu problemu. Każdy problem ma unikalny kontekst i inną domenę, w której reprezentowane są dane. Rozwiązania różnych problemów są również mierzone inaczej. Przyjrzyjmy się, jak można zaprojektować optymalizację roju cząstek, aby rozwiązać problem konstrukcji dronów. Ogólny cykl życia algorytmu optymalizacji roju cząstek jest następujący:

1. Zainicjuj populację cząstek. Określ liczbę cząstek, które mają być użyte, i zainicjuj każdą z nich w losowej pozycji w przestrzeni wyszukiwania.

2. Oblicz sprawność każdej cząstki. Biorąc pod uwagę położenie każdej cząstki, określ przydatność tej cząstki w tej pozycji.

3. Zaktualizuj pozycję każdej cząstki. Powtarzaj aktualizację pozycji wszystkich cząstek, korzystając z zasad inteligencji roju. Cząstki zbadają przestrzeń poszukiwań, a następnie zbiegną się w dobre rozwiązania.

4. Określ kryteria zatrzymania. Określ, kiedy cząsteczki przestaną się aktualizować, a algorytm się zatrzyma.

Algorytm optymalizacji roju cząstek jest dość prosty, ale szczegóły kroku 3 są szczególnie skomplikowane. W kolejnych sekcjach przyjrzymy się każdemu krokowi z osobna i odkryjemy szczegóły, dzięki którym algorytm działa.

Inicjalizacja populacji cząstek

Algorytm rozpoczyna się od utworzenia określonej liczby cząstek, która pozostanie taka sama przez cały okres istnienia algorytmu. Trzy czynniki, które są ważne w inicjalizacji cząstek, to

• Liczba cząstek - liczba cząstek wpływa na obliczenia. Im więcej cząstek istnieje, tym więcej wymaga obliczeń. Ponadto, więcej cząstek prawdopodobnie będzie oznaczać, że zbieżność w celu znalezienia najlepszego globalnego rozwiązania potrwa dłużej, ponieważ jest ich więcej , przyciągają ich najlepsze lokalne rozwiązania. Ograniczenia problemu wpływają również na liczbę cząstek. Większa przestrzeń wyszukiwania może wymagać większej liczby cząstek, aby ją zbadać. Cząstek może być nawet 1000 lub

zaledwie 4. Zwykle od 50 do 100 cząstek daje dobre rozwiązania bez zbytniego nakładu obliczeniowego.

• Pozycja początkowa dla każdej cząstki - Pozycja początkowa każdej cząstki powinna być przypadkowa we wszystkich odpowiednich wymiarach. Ważne jest, aby cząstki były równomiernie rozmieszczone w przestrzeni wyszukiwania. Jeśli większość cząstek znajduje się w określonym regionie przestrzeni poszukiwań, będą miały trudności ze znalezieniem rozwiązań poza tym obszarem.

• Prędkość początkowa dla każdej cząstki - Prędkość cząstek jest inicjalizowana na 0, ponieważ nie wpłynęła jeszcze na cząstki. Dobrą analogią jest to, że ptaki startują do lotu z pozycji stacjonarnej.

Tabela opisuje dane zawarte w każdej cząstce na etapie inicjalizacji algorytmu. Zauważ, że prędkość wynosi 0; aktualne i najlepsze wartości kondycji wynoszą 0, ponieważ nie zostały jeszcze obliczone.

Cząstka: Prędkość: Prąd aluminium (x): Aktualny plastik (y): Aktualna sprawność : Najlepsze aluminium (x): Najlepszye plastik (y): Najlepsza sprawność

1 : 0 : 7 : 1 : 0 : 7 : 1 : 0

2 : 0 : -1 : 9 : 0 : -1 : 9 : 0

3 : 0 : -10 : 1 : 0 :-10 : 1 : 0

4 : 0 : -2 : -5 : 0 : -2 : -5 : 0

Metoda generowania roju polega na utworzeniu pustej listy i dołączeniu do niej nowych cząstek. Kluczowymi czynnikami są

• Zapewnienie konfigurowalnej liczby cząstek.

• Zapewnienie, że generowana liczba losowa jest jednolita; liczby są rozmieszczone w przestrzeni wyszukiwania w ramach ograniczeń. Ta implementacja zależy od funkcji używanego generatora liczb losowych.

• Zapewnienie, że ograniczenia przestrzeni poszukiwań są określone, w tym przypadku -10 i 10 zarówno dla x, jak i y cząstki.

```
generate_swarm(number_of_particles):
  let particles equal an empty list
  for particle in range(number_of_particles):
    append Particle(random(-10, 10), random(-10, 10), INERTIA,
                    COGNITIVE_CONSTANT, SOCIAL_CONSTANT) to particles
  return particles
```

Oblicz zdatność każdej cząstki

Następnym krokiem jest obliczenie sprawności każdej cząstki w jej aktualnej pozycji. Sprawność cząstek jest obliczana za każdym razem, gdy cały rój zmienia pozycję. W scenariuszu z dronem naukowcy zapewnili funkcję, w której wynikiem jest wielkość oporu i kołysania przy określonej liczbie elementów aluminiowych i plastikowych. Ta funkcja jest używana jako funkcja przystosowania w algorytmie optymalizacji roju cząstek w tym przykładzie.

$$f(x,y) = (x + 2y - 7)^2 + (2x + y - 5)^2$$

Jeśli x jest aluminium, a y jest tworzywem sztucznym, można wykonać następujące obliczenia dla każdej cząstki, aby określić jej przydatność, zastępując x i y wartości aluminium i tworzywa sztucznego.

$$f(7,1) = (7 + 2(1) - 7)^2 + (2(7) + 1 - 5)^2 = 104$$
$$f(-1,9) = (-1 + 2(9) - 7)^2 + (2(-1) + 9 - 5)^2 = 104$$
$$f(-10,1) = (-10 + 2(1) - 7)^2 + (2(-10) + 1 - 5)^2 = 801$$
$$f(-2,-5) = (-2 + 2(-5) - 7)^2 + (2(-2) - 5 - 5)^2 = 557$$

Teraz tabela cząstek przedstawia obliczoną przydatność dla każdej cząstki. Jest również ustawiona jako najlepsza przystosowanie dla każdej cząstki, ponieważ jest to jedyna znana przydatność w pierwszej iteracji. Po pierwszej iteracji najlepszym dopasowaniem dla każdej cząstki jest najwyższe dopasowanie w historii każdej konkretnej cząstki.

Cząstka: Prędkość: Prąd aluminium (x): Aktualny plastik (y): Aktualna sprawność: Najlepsze aluminium (x): Najlepszy plastik (y): Najlepsza sprawność

1: 0: 7: 1: 296: 7: 1: 296

2: 0: -1: 9: 104: -1: 9: 104

3: 0: -10: 1: 80: -10: 1: 80

4: 0: -2: -5: 365: -2: -5: 365

Pseudo kod

Funkcja sprawości reprezentuje funkcję matematyczną w kodzie. Każda biblioteka matematyczna będzie zawierała wymagane operacje, takie jak funkcja potęgi i funkcja pierwiastka kwadratowego.

```
calculate_fitness(x, y):
  return power(x + 2 * y - 7, 2) + power(2 * x + y - 5, 2)
```

Funkcja aktualizowania sprawności cząstki jest również trywialna, ponieważ określa, czy nowa sprawność lepsza od poprzedniej najlepszej, a następnie przechowuje tę informację.

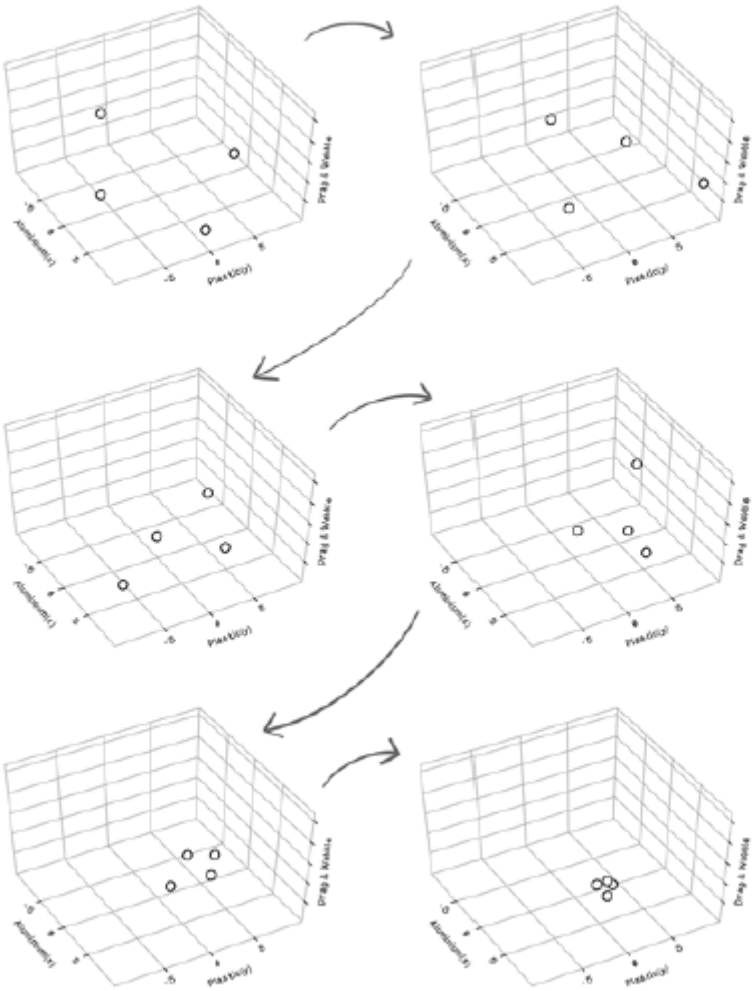
```
update_fitness(x, y):
  let particle.fitness equal the result of calculate_fitness(x, y)
  if particle.fitness is less than particle.best_fitness:
    let particle.best_fitness equal particle.fitness
    let particle.best_x equal x
    let particle.best_y equal y
```

Funkcja określania najlepszej cząstki w roju iteruje przez wszystkie cząstki, aktualizuje ich sprawność w oparciu o ich nowe pozycje i znajduje cząstkę, która daje najmniejszą wartość funkcji przystosowania. W tym przypadku minimalizujemy, więc mniejsza wartość jest lepsza.

```
get_best(swarm):
  let best_fitness equal infinity
  let best_particle equal nothing
  for particle in swarm:
    update fitness of particle
    if particle.fitness is less than best_fitness:
      let best_fitness equal particle.fitness
      let best_particle equal particle
  return best_particle
```

Zaktualizuj położenie każdej cząstki

Etap aktualizacji algorytmu jest najbardziej skomplikowany, ponieważ to tam dzieje się magia. Etap aktualizacji obejmuje właściwości inteligencji roju w przyrodzie w model matematyczny, który umożliwia eksplorację przestrzeni poszukiwań przy jednoczesnym doskonaleniu dobrych rozwiązań. Cząsteczki w roju aktualizują swoją pozycję, biorąc pod uwagę zdolności poznawcze i czynniki w środowisku wokół nich, takie jak bezwładność i to, co rój robi. Czynniki te wpływają na prędkość i położenie każdej cząstki. Pierwszym krokiem jest zrozumienie, w jaki sposób aktualizowana jest prędkość. Prędkość określa kierunek i prędkość ruchu cząstki. Cząsteczki w roju przemieszczają się do różnych punktów w przestrzeni poszukiwań, aby znaleźć lepsze rozwiązania. Każda cząstka polega na swojej pamięci dobrego rozwiązania i znajomości najlepszego rozwiązania roju. Rysunek ilustruje ruch cząstek w roju podczas aktualizacji ich pozycji.



SKŁADNIKI AKTUALIZACJI PRĘDKOŚCI

Do obliczenia nowej prędkości każdej cząstki używane są trzy komponenty: bezwładność, poznawcza i społeczna. Każdy składnik wpływa na ruch cząstki. Przyjrzymy się każdemu z elementów oddzielnie, zanim zagłębimy się w sposób ich łączenia, aby zaktualizować prędkość i ostatecznie położenie cząstki:

• Bezwładność - składnik bezwładności reprezentuje opór ruchu lub zmianę kierunku dla określonej cząstki, który wpływa na jej prędkość. Składowa bezwładności składa się z dwóch wartości: wielkości bezwładności i aktualnej prędkości cząstki. Wartość bezwładności to liczba od 0 do 1.

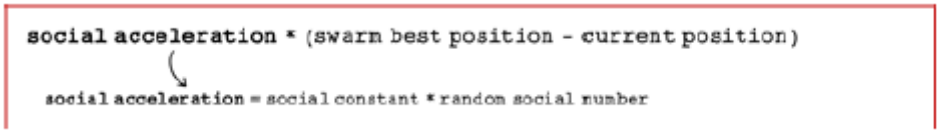
```
inertia * current velocity
```

o Wartość bliższa 0 przekłada się na eksplorację, potencjalnie wymagającą większej liczby iteracji.

o Wartość bliższa 1 oznacza większą eksplorację cząstek w mniejszej liczbie iteracji.

• Poznawczy - komponent poznawczy reprezentuje wewnętrzną zdolność poznawczą określonej cząstki. Zdolność poznawcza to poczucie cząstki znającej swoją najlepszą pozycję i używającej tej pozycji do wpływania na jej ruch. Stała poznawcza to liczba większa niż 0 i mniejsza niż 2. Większa stała poznawcza oznacza większe wykorzystanie przez cząstki.

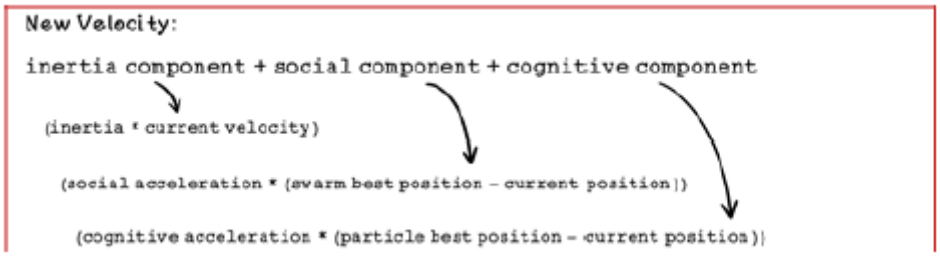
• Społeczny - komponent społeczny reprezentuje zdolność cząstki do interakcji z rojem. Cząstka zna najlepszą pozycję w roju i wykorzystuje tę informację do wpływania na jej ruch. Przyspieszenie społeczne jest określane za pomocą stałej i skalowanie jej liczbą losową. Stała społeczna pozostaje taka sama przez cały czas istnienia algorytmu, a czynnik losowy zachęca do różnorodności, sprzyjając czynnikowi społecznemu.



Im większa stała społeczna, tym większa będzie eksploracja, ponieważ cząstka bardziej faworyzuje swój składnik społeczny. Stała społeczna to liczba od 0 do 2. Większa stała społeczna oznacza więcej eksploracji.

AKTUALIZACJA PRĘDKOŚCI

Teraz, gdy rozumiemy składnik bezwładności, składnik poznawczy i składnik społeczny, przyjrzyjmy się, jak można je połączyć, aby zaktualizować nową prędkość cząstek.



Patrząc na matematykę, możemy mieć trudności ze zrozumieniem, jak różne składowe funkcji wpływają na prędkość cząstek. Rysunek 7.22 pokazuje, jak różne czynniki wpływają na cząstkę.

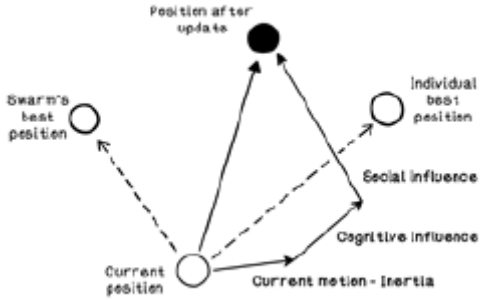


Tabela przedstawia atrybuty każdej cząstki po obliczeniu sprawności każdej z nich.

Cząstka: Prędkość: Aktualne aluminium: Aktualny plastik: Aktualna sprawność: Najlepsze aluminium: Najlepszy plastik: Najlepsza sprawność

1: 0: 7: 1: 296: 2: 4: 2: 4

3: 0: -10: 1: 80: -10: 1: 80

4: 0: -2: -5: 365: -2: -5: 365

Następnie zajmiemy się obliczeniami aktualizacji prędkości dla cząstki, biorąc pod uwagę wzory, które przepracowaliśmy. Oto stałe konfiguracje, które zostały ustawione dla tego scenariusza:

• Bezwładność jest ustawiona na 0,2. To ustawienie sprzyja wolniejszej eksploracji.

• Stała poznawcza wynosi 0,35. Ponieważ ta stała jest mniejsza niż stała społeczna, składnik społeczny jest faworyzowany nad poznawczym komponentem pojedynczej cząstki.

• Stała społeczna jest ustawiona na 0,45. Ponieważ ta stała jest czymś więcej niż stałą poznawczą, preferowany jest składnik społeczny. Cząsteczki zwiększają wagę najlepszych wartości znalezionych przez rój.

Rysunek przedstawia obliczenia komponentu bezwładności, komponentu poznawczego i komponentu społecznego dla wzoru na aktualizację prędkości. Przejście do obliczeń opisano na rysunku dla cząstki 1 na liście.

Inertia Component:

inertia * current velocity
= 0.2 * 0
= 0

Cognitive Component:

cognitive acceleration = cognitive constant * random cognitive number
= 0.35 * 0.2
= 0.07

cognitive acceleration * (particle best position - current position)
= 0.07 * ([7,1] - [7,1])
= 0.07 * 0
= 0

Social Component:

social acceleration = social constant * random social number
= 0.45 * 0.3
= 0.135

social acceleration * (swarm best position - current position)
= 0.135 * ([-10,1] - [7,1])
= 0.135 * sqrt((-10 - 7)^2+ (1 - 1)^2) Distance Formula: sqrt((x1 - x2)^2 + (y1 - y2)^2)
= 0.135 * 17
= 2.295

New Velocity:

inertia component + cognitive component + social component
= 0 + 0 + 2.295
= 2.295

Po wykonaniu tych obliczeń dla wszystkich cząstek, prędkość każdej cząstki jest aktualizowana, jak przedstawiono w tabeli

Cząstka: Prędkość: Aktualne aluminium: Aktualny plastik: Aktualna sprawność : Najlepsze aluminium: Najlepszy plastik : Najlepsza przydatność

1: 2,295: 7: 1: 296: 7: 1: 296

2: 1,626: -1: 9: 104: -1: 9: 104

3: 2,043: -10: 1: 80: -10: 1: 80

4: 1,35: -2: -5: 365: -2: -5: 365

AKTUALIZACJA POZYCJI

Teraz, gdy już wiemy, jak aktualizowana jest prędkość, możemy zaktualizować bieżące położenie każdej cząstki, używając nowej prędkości.

```
current position + new velocity
```

```
New Position:
current position + new velocity
= ([7,1]) + 2.295
= [9.295, 3.295]
```

Dodając aktualną pozycję i nową prędkość, możemy określić nowe położenie każdej cząstki i zaktualizować tabelę atrybutów cząstek o nowe prędkości. Następnie ponownie obliczana jest sprawność każdej cząstki, biorąc pod uwagę jej nowe położenie, i zapamiętywana jest jej najlepsza pozycja.

Cząstka: Prędkość: Aktualne aluminium: Aktualny plastik : Aktualna sprawność : Najlepsze aluminium: Najlepszy plastik : Najlepsza sprawność

1 : 2,295 : 9,925 : 3,325 : 721,286 : 1 : 296

2 : 1,626 : 0,626 : 10 : 73,538 : 0,626 : 10 : 73,538

3 : 2,043 : 7,043 : 1,043 : 302,214 : -10 : 1: 80

4 : 1,35 : -0,65 : -3,65 : 179,105 : -0,65 : -3,65 : 179,105

Obliczenie początkowej prędkości dla każdej cząstki w pierwszej iteracji jest dość proste, ponieważ nie było poprzedniej najlepszej pozycji dla każdej cząstki - tylko najlepsza pozycja roju, która miała wpływ tylko na składnik społeczny. Przyjrzyjmy się, jak będzie wyglądać obliczenie aktualizacji prędkości z nowymi informacjami o najlepszej pozycji każdej cząstki i nowej najlepszej pozycji roju. Przejście do obliczeń opisano na rysunku 7.25 dla cząstki 1 na liście.

Inertia Component:

inertia * current velocity

= 0.2 * 2.295

= 0.59

Cognitive Component:

cognitive acceleration = cognitive constant * random cognitive number

= 0.35 * 0.2 Note: We're adjusting the random numbers for ease of understanding only

= 0.07

cognitive acceleration * (particle best position - current position)

= 0.07 * ([7,1] - [9.925,3.325])

= 0.07 * $\text{sqrt}((7 - 9.925)^2 + (1 - 3.325)^2)$

= 0.07 * 3.736

= 0.266

Social Component:

social acceleration = social constant * random social number

= 0.45 * 0.3

= 0.135

social acceleration * (swarm best position – current position)

= 0.135 * ([0.626,10] – [9.925,3.325])

= 0.135 * $\text{sqrt}((0.626 - 9.925)^2 + (10 - 3.325)^2)$

= 0.135 * 11.447

= 1.545

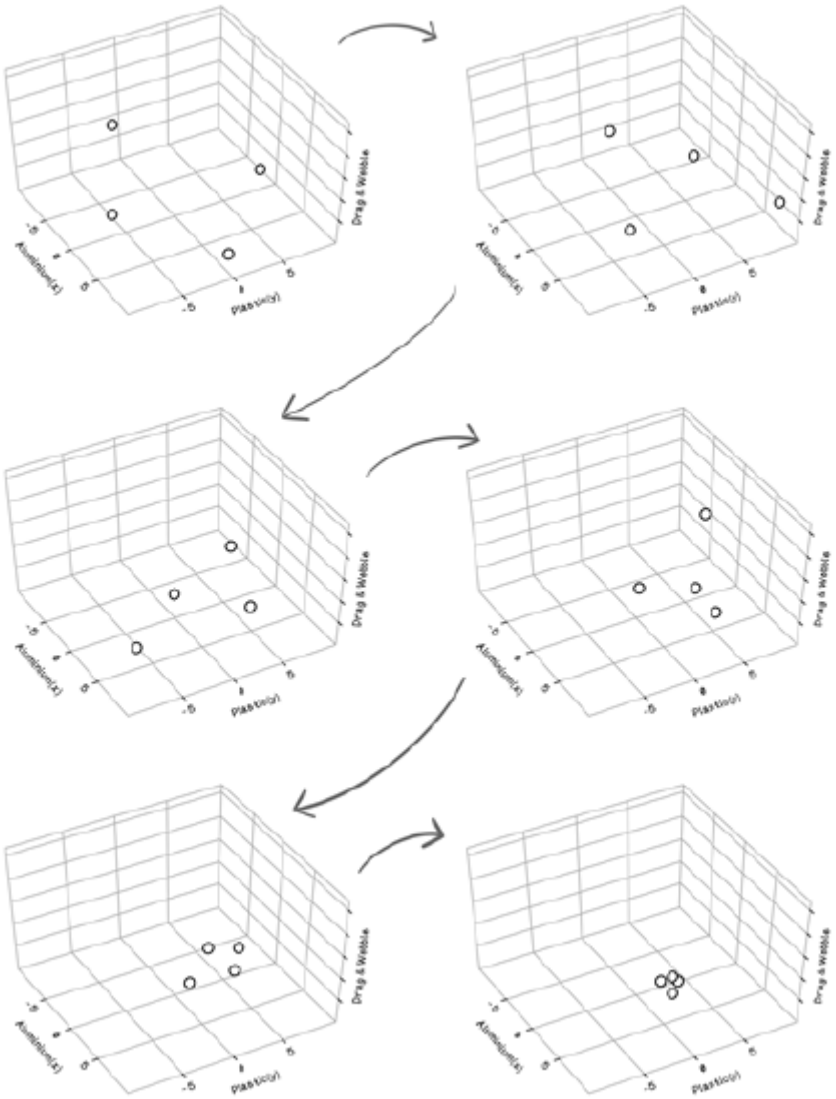
New Velocity:

inertia component + cognitive component + social component

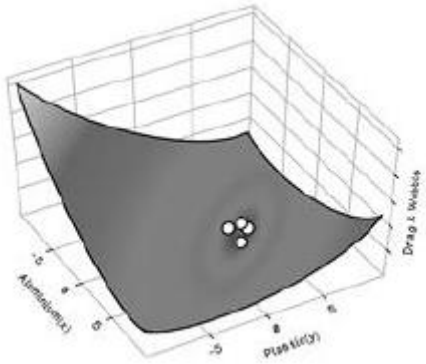
= 0.59 + 0.266 + 1.545

= 2.401

W tym scenariuszu komponent poznawczy i komponent społeczny odgrywają rolę w aktualizacji prędkości, podczas gdy na scenariusz oddziaływał komponent społeczny, ponieważ jest to pierwsza iteracja. Cząsteczki przemieszczają się do różnych pozycji w ciągu kilku iteracji. Rysunek przedstawia ruch cząstek i ich zbieżność w roztworze.



W ostatniej klatce na rysunku wszystkie cząstki zbiegły się w określonym regionie w przestrzeni poszukiwań. Najlepsze rozwiązanie z roju zostanie użyte jako ostateczne rozwiązanie. W rzeczywistych problemach optymalizacyjnych nie jest możliwa wizualizacja całej przestrzeni wyszukiwania (co spowodowałoby, że algorytmy optymalizacyjne byłyby niepotrzebne). Ale funkcja, której użyliśmy w przykładzie drona, to znana funkcja, zwana funkcją Bootha. Odwzorowując ją na trójwymiarową płaszczyznę kartezjańską, możemy zobaczyć, że cząstki rzeczywiście zbiegają się w minimalnym punkcie przestrzeni poszukiwań.



Po zastosowaniu algorytmu optymalizacji roju cząstek dla przykładu drona, okazuje się, że optymalny stosunek aluminium i plastiku do zminimalizowania oporu i kołysania wynosi 1: 3 - to znaczy 1 część aluminium i 3 części plastiku. Kiedy wprowadzamy te wartości do funkcji dopasowania, wynikiem jest 0, co jest minimalną wartością funkcji.

Pseudo kod

Krok aktualizacji może wydawać się zniechęcający, ale jeśli komponenty zostaną rozbite na proste, skoncentrowane funkcje, kod staje się prostszy i łatwiejszy do napisania, używania i zrozumienia. Pierwsze funkcje to funkcja obliczania bezwładności, funkcja przyspieszenia poznawczego i funkcja przyspieszenia społecznego. Potrzebujemy również funkcji do pomiaru odległości między dwoma punktami, która jest reprezentowana przez podniesienie do kwadratu sumy kwadratów różnicy wartości x zsumowanych z kwadratem różnicy wartości y.

```
calculate_inertia(inertia_constant, velocity):
  return inertia_constant * current_velocity

calculate_cognitive_acceleration(cognitive_constant):
  return cognitive_constant * random number between 0 and 1

calculate_social_acceleration(social_constant):
  return social_constant * random number between 0 and 1

calculate_distance(best_x, best_y, current_x, current_y):
  return square_root(
                power(best_x - current_x), 2) + power(best_y - current_y), 2)
                    )
```

Komponent poznawczy oblicza się, znajdując przyspieszenie poznawcze, używając funkcji, którą zdefiniowaliśmy wcześniej w sekcji 7.5.3, oraz odległości między najlepszą pozycją cząstki a jej aktualną pozycją.

```
calculate_cognitive(cognitive_constant,
                    particle_best_x,particle_best_y
                    particle_current_x,particle_current_y):
  let acceleration equal cognative_acceleration(cognitive_constant)
  let distance equal calculate_distance(particle_best_x,
                                        particle_best_y
                                        particle_current_x,
                                        particle_current_y)
  return acceleration * distance
```

Składnik społeczny jest obliczany poprzez znalezienie przyspieszenia społecznego przy użyciu funkcji, którą zdefiniowaliśmy wcześniej w sekcji 7.5.3, oraz odległości między najlepszą pozycją roju a aktualną pozycją cząstki.

```
calculate_social(social_constant,
                 swarm_best_x,swarm_best_y
                 particle_current_x,particle_current_y):
  let acceleration equal social_acceleration(social_constant)
  let distance equal calculate_distance(swarm_best_x,
                                        swarm_best_y
                                        particle_current_x,
                                        particle_current_y)
  return acceleration * distance
```

Funkcja aktualizacji obejmuje wszystko, co zdefiniowaliśmy, aby przeprowadzić faktyczną aktualizację prędkości i położenia cząstki. Prędkość jest obliczana przy użyciu komponentu bezwładności, komponentu poznawczego i komponentu społecznego. Pozycja jest obliczana przez dodanie nowej prędkości do aktualnej pozycji cząstki.

```
update_particle(cognitive_constant,social_constant,particle_velocity,
                particle_best_x,particle_best_y,
                swarm_best_x,swarm_best_y,
                particle_current_x,particle_current_y)
  let inertia equal calculate_inertia(inertia_constant,
                                      particle_constant)
  let cognitive equal calculate_cognitive(cognitive_constant,
                                      particle_best_x,particle_best_y
                                      particle_current_x,particle_current_y)
  let social equal calculate_social(social_constant,
                                      swarm_best_x,swarm_best_y
                                      particle_current_x,particle_current_y)
  let particle.velocity equal inertia + cognitive + social
  let particle.x equal particle.x + velocity
  let particle.y equal particle.y + velocity
```

Określ kryteria zatrzymania

Cząsteczki w roju nie mogą aktualizować się i wyszukiwać w nieskończoność. Należy określić kryteria zatrzymania, aby umożliwić działanie algorytmu przez rozsądną liczbę iteracji w celu znalezienia odpowiedniego rozwiązania. Liczba iteracji wpływa na kilka aspektów znajdowania rozwiązań, w tym

• Eksploracja-Cząsteczki wymagają czasu na zbadanie przestrzeni poszukiwań, aby znaleźć obszary z lepszymi rozwiązaniami. Na eksplorację mają również wpływ stałe zdefiniowane w funkcji prędkości aktualizacji.

• Eksploatacja - cząstki powinny zbiegać się w dobrym rozwiązaniu po przeprowadzeniu rozsądnej eksploracji.

Strategia zatrzymania algorytmu polega na zbadaniu najlepszego rozwiązania w roju i ustaleniu, czy jest w stagnacji. Stagnacja występuje, gdy wartość najlepszego rozwiązania nie zmienia się lub nie zmienia się w znaczący sposób. Uruchomienie większej liczby iteracji w tym scenariuszu nie pomoże znaleźć lepszych rozwiązań. W przypadku stagnacji najlepszego rozwiązania parametry funkcji aktualizacji można dostosować, aby sprzyjać większej eksploracji. Jeśli pożądana jest większa eksploracja, zwykle ta korekta oznacza więcej iteracji. Stagnacja może oznaczać, że znaleziono dobre rozwiązanie lub że rój utknął na najlepszym lokalnym rozwiązaniu. Jeśli na początku doszło do wystarczającej eksploracji, a rój stopniowo ulega stagnacji, rój znalazł dobre rozwiązanie.

Przypadki użycia algorytmów optymalizacji roju cząstek

Algorytmy optymalizacji roju cząstek są interesujące, ponieważ symulują zjawisko naturalne, co ułatwia ich zrozumienie, ale można je zastosować do szeregu problemów na różnych poziomach abstrakcji. W tym rozdziale przyjrzano się problemowi optymalizacji produkcji dronów, ale algorytmy optymalizacji roju cząstek mogą być używane w połączeniu z innymi algorytmami, takimi jak sztuczne sieci neuronowe, odgrywając niewielką, ale krytyczną rolę w znajdowaniu dobrych rozwiązań. Jednym z interesujących zastosowań algorytmu optymalizacji roju cząstek jest głęboka stymulacja mózgu. Koncepcja polega na zainstalowaniu sond z elektrodami w ludzkim mózgu, aby stymulować go do leczenia schorzeń, takich jak choroba Parkinsona. Każda sonda zawiera elektrody, które można skonfigurować w różnych kierunkach, aby prawidłowo leczyć stan u każdego pacjenta. Naukowcy z University of Minnesota opracowali algorytm optymalizacji roju cząstek, aby zoptymalizować kierunek każdej elektrody, aby zmaksymalizować obszar zainteresowania, zminimalizować obszar unikania i zminimalizować zużycie energii. Ponieważ cząstki skutecznie przeszukują te wielowymiarowe przestrzenie problemowe, algorytm optymalizacji roju cząstek jest skuteczny w znajdowaniu optymalnych konfiguracji elektrod w sondach. Oto kilka innych rzeczywistych zastosowań algorytmów optymalizacji roju cząstek:

• Optymalizacja wag w sztucznej sieci neuronowej - Sztuczne sieci neuronowe są wzorowane na idei działania ludzkiego mózgu. Neurony przekazują sygnały do innych neuronów, a każdy neuron dostosowuje sygnał przed przekazaniem go dalej. Sztuczna sieć neuronowa wykorzystuje wagi do dostosowania każdego sygnału. Siła sieci polega na znalezieniu właściwej równowagi wag w celu utworzenia wzorców w relacjach danych. Dostosowywanie wag jest kosztowne obliczeniowo, ponieważ przestrzeń wyszukiwania jest ogromna. Wyobraź sobie, że musisz brutalnie wymusić każdą możliwą kombinację liczb dziesiętnych dla 10 wag. Ten proces trwałby latami. Nie panikuj, jeśli ta koncepcja brzmi myląco. Jak działają sztuczne sieci neuronowe, zajmiemy się w rozdziale 9. Optymalizacja roju cząstek może być wykorzystana do szybszego dostosowania wag sieci neuronowych, ponieważ szuka ona optymalnych wartości w przestrzeni poszukiwań bez wyczerpująco próbując każdego z nich.

• Śledzenie ruchu w filmach - Śledzenie ruchu ludzi jest trudnym zadaniem w wizji komputerowej. Celem jest rozpoznanie pozycji ludzi i zasugerowanie ruchu przy użyciu samych informacji z obrazów na wideo. Ludzie poruszają się inaczej, chociaż ich stawy poruszają się podobnie. Ponieważ obrazy zawierają wiele aspektów, przestrzeń poszukiwań staje się duża, a wiele wymiarów umożliwia

przewidywanie ruchu osoby. Optymalizacja roju cząstek działa dobrze w dużych przestrzeniach wyszukiwania i może być używana do poprawy wydajności śledzenia i przewidywania ruchu.

• Poprawa mowy w nagraniach audio-audio jest dopracowana. W tle zawsze słychać szum, który może zakłócać to, co ktoś mówi w nagraniu. Rozwiązaniem jest usunięcie szumu z nagranych klipów dźwiękowych mowy. Technika stosowana w tym celu polega na filtrowaniu klipu audio za pomocą szumu i porównywaniu podobnych dźwięków w celu usunięcia szumu w klipie audio. To rozwiązanie jest nadal skomplikowane, ponieważ redukcja pewnych częstotliwości może być dobra dla części klipu audio, ale może pogorszyć inne jego części. Aby zapewnić dobre usuwanie szumów, należy przeprowadzić dokładne wyszukiwanie i dopasowywanie. Tradycyjne metody są powolne, ponieważ przestrzeń wyszukiwania jest duża. Optymalizacja roju cząstek działa dobrze w dużych przestrzeniach wyszukiwania i może służyć do przyspieszenia procesu usuwania szumu z klipów audio.