AG – sprawozdanie do seminarium

Prowadzący: dr hab. H. Kwaśnicka

„Niszowanie i koewolucja”

Autorzy:
Konrad Falkowski (cz. 1)
Marcin Żybura (cz. 2)
Paweł Wnuk-Lipiński (cz. 3)

1 Wstęp

1.1 Niszowanie

1.1.1 Terminologia.

Słowo „niszowanie” jak nie trudno się domyśleć pochodzi od słowa „nisza”.
Czym jest nisza? Słowo pochodzi z języka francuskiego, oznacza wnękę.
Możemy spotkać się z następującymi określeniami:

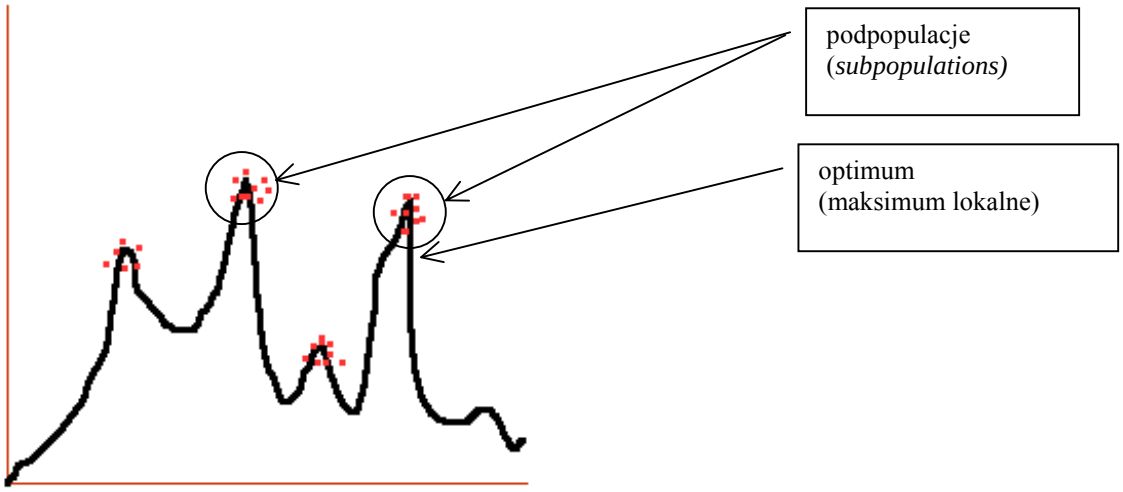
- **Nisza architektoniczna** – pionowe wgłębienie w murze, filarze itp.
- **Nisza ekologiczna** – zespół czynników ekologicznych i przestrzeń fizyczna zajmowana przez dany organizm niezbędna dla jego egzystencji
- **Nisza abrazyjna** – wklęsła forma u podstaw klifu, tworząca się w wyniku działalności fal morskich.
- **Nisza rynkowa** – segment rynku lub jego wydzielona część, w której niewielka gruba nabywców, posiadająca konkretne potrzeby, gotowa jest zapłacić wyższą cenę za dobro lub usługę
- **Nisza źródłowa** – wgłębienie w terenie powstałe wokół wypływu wody.
- **Nisza matematyczna** - obszar podejrzany o optimum.

Ogólnie: wnęka, wydzielony obszar.

Niszowanie jest właśnie mechanizmem wyszukującym nisze.

1.1.2 Czym jest niszowanie (ang. niching)?

Niszowanie jest techniką, która promuje formowanie i utrzymywanie stabilnych podpopulacji (*subpopulations*) w AG.



1.1.3 Do czego służy niszowanie?

Niszowanie służy do znajdowania większej liczby rozwiązań danego problemu. Inaczej mówiąc, korzystając z niszowania, znajdziemy wszystkie optima, nie tylko globalne.

W niektórych przypadkach znalezienie wszystkich optimów funkcji jest podstawą prawidłowego rozwiązania problemu. Przede wszystkim w tym celu powstał mechanizm niszowania, który pozwala zostać osobnikom na optimach lokalnych.

Niszowanie jest także pomocne dla znalezienia lepszego, pojedynczego rozwiązania trudnych problemów.

1.1.4 Dlaczego nie można lokalizować wielokrotnych rozwiązań za pomocą iterowanego AG?

Można, ale dają one bardzo słabe rezultaty i niepożądane efekty uboczne, takie jak:

- przedwczesne, wielokrotne zbieganie się osobników populacji w tym samym optimum;
- dyskryminowanie słabszych rozwiązań

1.1.5 Trochę historii.

Niszowane AG (AG z wykorzystaniem technik niszowania) zostały zauważone w świecie dzięki swojej zdolności do znajdowania wielu dobrych zróżnicowanych rozwiązań.

Techniki niszowania stosuje się od ponad 10 lat.

W zwykłym AG, ciśnienie selekcyjne powodowały zbyt szybkie zbieganie do jednolitej populacji, zawierającej kopie najlepszego rozwiązania. Dzięki różnorakim mechanizmom niszowania (o których powie Marcin) pojedyncze osobniki populacji, stały się „samowystarczalnymi" dobrymi rozwiązaniami, co najważniejsze: różnymi. Dzięki temu, jakby załatano dziury rozwiązań optymalnych przestrzeni poszukiwań.

1.2 Koewolucja.

1.2.1 Terminologia.

Koewolucja – inaczej ewolucja kooperacyjna. Proces, najczęściej długoterminowy polegający na zmianie w kompozycji genetycznej jednego gatunku (bądź grupy) w odpowiedzi na zmianę genetyczną innego gatunku (grupy).
Bardziej ogólnie: obustronna zmiana ewolucyjna w środowisku odziaływującym na siebie.

1.2.2 Kilka cytatów, czyli: Gdzie spotkamy się z koewolucją?

- w biologii:

(...) Zasadą doboru naturalnego jest utrwalanie się tych właściwości organizmu, które zwiększają szansę jego przeżycia i rozrodu. Dotyczy to również takich właściwości, które objawiają się skłonnością do współdziałania z innymi osobnikami, pod warunkiem jednak, że są one nosicielami tych samych genów. Wówczas ten sam "gen kooperacji" rozprzestrzeni się poprzez potomstwo każdego z partnerów (piszę "gen kooperacji" w cudzysłowie, bo na pewno nie chodzi tu o jeden gen, lecz o zestaw wielu genów). Na tym polega działanie doboru krewniaczego, bo u bliskich krewnych z dużym prawdopodobieństwem występują takie same geny. Również współpraca osobników o zupełnie odmiennych genotypach może doprowadzić do utrwalenia się ich "genów współpracy" (różnych u obu partnerów), osobno u potomstwa każdego z nich, o ile efekty tej współpracy są opłacalne dla obu stron. Taka obustronna korzyść może zachodzić w sytuacji, gdy partnerzy korzystają z rozmaitych zasobów, wzajemnie ułatwiając sobie ich zdobycie, ale nie konkurują o te same dobra. Na tej zasadzie opiera się **koewolucja** różnych gatunków, prowadząca do częstej w przyrodzie symbiozy mutualistycznej, na przykład kwiatów z zapylającymi je owadami (...)

- w biologii (a konkretniej w lesie):

(...) w naturalnym lesie następuje ciągła **koewolucja** patogenów i drzew. Patogeny ewoluują bardzo szybko – często maja wiele pokoleń rocznie. Nowe siewki już na początku swego życia przechodzą przez sito selekcyjne - te które są podatne na patogenny które przystosowały się do żerowania na ich rodzicach - po prostu nie przeżywają. A wiec drzewa są różne, jedne maja więcej metabolitów wtórnych, inny skład itd. Jest to mechanizm który zapobiega tym sławnym gradacjom szkodników. (...)

- w literaturze (esej S. Lema):

(...)Wirusy tak długo z małpami współistniały, ze obecnie już nie powodują im żadnych przypadłości. Takim to sposobem **koewolucja**, trwająca wiele milionów lat, powoduje "pokojowe współżycie" prapasożyta i pragospodarza. (...)

1.3 Koewolucyjny model ECT (Evolutionary Computation Toolkit)*

- Jej pomysłodawcą i twórcą jest Dr Mitchell A. Potter

- ECT jest kolekcją klas JAVA służącą do konstruowania systemów opartych o zasady ewolucji Darwina.

  W systemach konstruowanych w oparciu o ECT, problemy są rozwiązywane poprzez rozwijające się populacje rywalizujących ze sobą osobników - rozwiązań. Rozwiązania są poddawane klasycznym operacjom genetycznym, jak mutacje i krzyżowania. Lepsze osobniki mają większą szansę na przeżycie i przekazanie części swoich cech następnym pokoleniom.

- ECT posiada w sobie cechy, wyróżniające ją spośród innych bibliotek ewolucyjnych.

  \+ jest oparta o koewolucyjny model wielogatunkowy, znacznie wydajniejszy od tradycyjnego jednogatunkowego modelu ewolucyjnego.
  Umożliwia on stosowanie algorytmów ewolucyjnych do bardzo złożonych problemów, poprzez rozwijanie rozwiązań będących ze sobą w interakcji → komponenty (rozwiązania, osobniki) współdziałają ze sobą.

  \+ ECT jest biblioteką otwartą (rozszerzalność jest przecież podstawową cechą języka JAVA) nie ograniczoną do żadnych szczególnych reprezentacji EA. Więcej, każdy komponent – gatunek, może posiadać własną reprezentację (mówimy wówczas o hybrydyzacji).

  \+ ECT kodowana jest w Javie (inne biblioteki ewolucyjne w C++). Java posiada znacznie więcej zalet niż wysłużony C. Należą do nich m.in. niezależność platformowa (przenośność), łatwość implementacji, bezpieczeństwo, wielowątkowość.

- Koewolucyjny model ECT

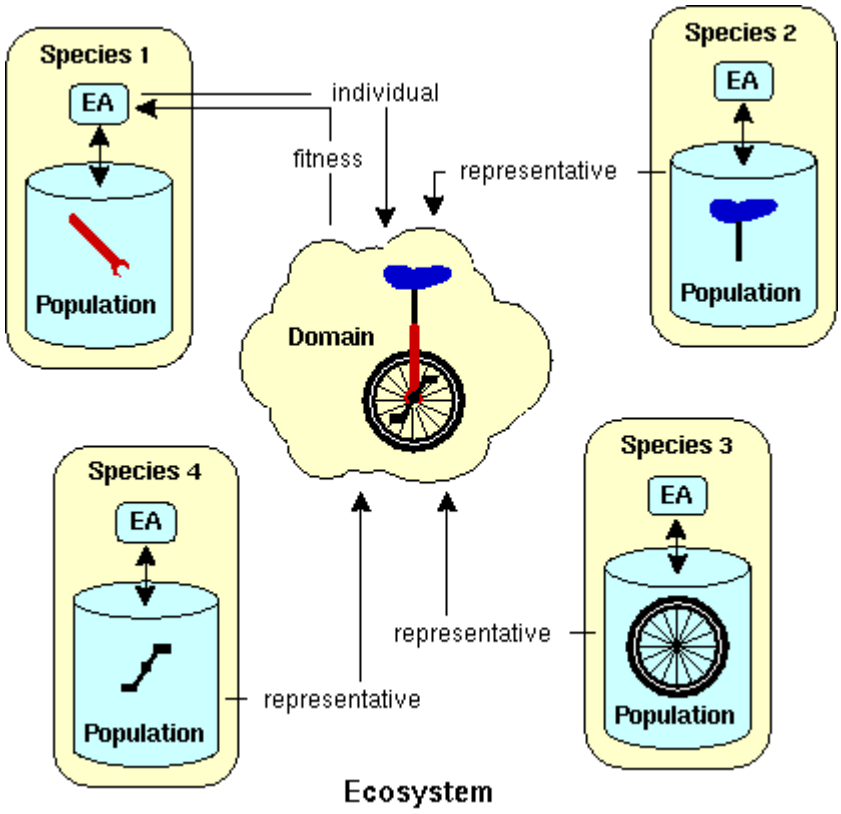
  Dobrym przykładem złożonego problemu (rozwiązywanego przez ECT) jest uczenie robotów zadań, gdzie złożone zachowania są dekomponowane na małe „zachowania składowe" (subbehaviors).

  Istnieją dwa główne powody, dla których tradycyjny jednogatunkowy model ewolucyjny nie jest odpowiedni do rozwiązywania tego typu problemów. Po pierwsze: populacje ewoluujące w oparciu o systemy EC mają silną tendencję do zbiegającego się rozwiązania. Przez to wyklucza on długoterminowe utrzymywanie różnorodnych komponentów składowych, ponieważ wszystkie słabsze będą eliminowane.
  Po drugie: pojedyncze gatunki reprezentują kompletne rozwiązania i ewoluują w izolacji (tzn. nie wnoszą nic nowego, tylko się doskonalą).

---

*) model ECT nazywany jest też w literaturze modelem EC*Kit*

Koewolucyjny model na którym opiera się ECT, reprezentowany jest przez **ekosystem** składający się z dwóch lub większej liczby gatunków. Podobnie jak w naturze, każdy gatunek jest genetycznie izolowany → oznacza to tyle, że osobniki „współżyją ” z osobnikami tego samego gatunku (mrówka i słoń nie będą miały potomstwa). Jest to egzekwowane głównie dzięki temu, że osobniki danego gatunku są odseparowane od innych. Jednak pomimo tego, że gatunki są izolowane genetycznie, wchodzą w interakcje z innymi gatunkami w tzw. „dzielonej dziedzinie modelu” (SDM – Sharing Domain Model).



Przykładowy ekosystem, składający się z 4 gatunków. Każdy z nich rozwija się w swojej populacji i przystosowuje się do środowiska poprzez powtarzający się algorytm EA.
W przykładzie, zadanie polegające na wyewoluowanie „monocyklu” zostało dekomponowane na cztery podzadania (subtasks). W kolejnym kroku gatunki prezentują wyniki w SDM. Zauważamy, że fitness zostaje zwrócony do jednego gatunku w celu dalszej ewolucji.

- Reprezentacja ECT – klasy i interfejsy.

ECT stosuje prostą translację docelowego modelu na klasy Javy. Klasy te są powiązane ze sobą w pakiet **eckit**. ECT posiada też zasób klas implementujących strategie ewolucyjne **eckit.strategies**.

Wyróżniamy sześć podstawowych klas ECT:

1) Ecosystem – główna, zawiera wszystkie gatunki i dziedzinę.
2) Species – zawiera populację dorosłych osobników i strategie do ewolucji
3) Group – implementuje kolekcję organizmów, zawiera metody: mutacji, krzyżowania, selekcji, etc...
4) Strategy – interfejs, zawiera pojedynczą metodę dla ewolucji gatunku w jednym pokoleniu.
5) Organism – jest klasą abstrakcyjną, reprezentującą ogranizm.
6) Domain – jest klasą abstrakcyjną. Zawiera strukturą aplikacji. Jest jedyną klasą w ECT, która musi być dostarczona przez użytkownika systemu.

2 Metody niszowania

2.1 Preselection

Podstawowym mechanizmem działania AG jest promowanie najlepiej przystosowanych osobników podczas operacji reprodukcji. W prostym AG ten pęd ku najlepszemu przystosowaniu nie jest niczym ograniczony, co powoduje szybkie upodobnienie się wszystkich osobników do siebie. W naturze można zaobserwować, że zróżnicowanie osobników żyjących na jednej przestrzeni umożliwia czerpanie zasobów na wiele sposobów. Okazuje się, że w danej sytuacji dobre jest wiele rozwiązań.

Cavicchio (1970) był jednym z pierwszych, którzy podjeli próbę wywołania „niszopodobnego" zachowania algorytmu genetycznego. Wprowadził on mechanizm nazywany preselekcją. Powszechnie metodę tą określa się mianem „niszopodobnej" ponieważ umożliwia ona utrzymanie różnorodności nie tak jak dzieje się to w przyrodzie poprzez dzielenie zasobów między osobnikami, ale poprzez wpływanie na sposób zastępowania starych pokoleń nowymi. Covacchio słusznie zauważył, że zróżnicowanie osobników możemy zachować, jeżeli nowe osobniki będą zastępowały podobne do siebie stare osobniki. Aby nie zwiększać kosztów algorytmu stwierdził, że wystarczającą miarą podobieństwa jest bliskie pokrewieństwo i tak powstał algorytm, który ogólnie polegał na zastępowaniu gorszego z rodziców przez ich potomka, o ile tylko potomek wykazywał lepsze od niego przystosowanie. Cavacchio twierdził, że udało mu się w ten sposób utrzymać bardziej różnorodne populacje w szeregu przebiegów symulacyjnych, przy stosunkowo niedużych rozmiarach populacji (n=20). W [1] efekty, które uzyskał Cavicchio zostały zakwestionowane gdyż zastosował on wiele innych czynników wpływających na działanie algorytmu. Metoda ze względu na swoją małą skuteczność praktycznie nigdy nie była stosowana.

2.2 Crowding

De Jong (1975) uogólnił technikę preselekcji w postaci modelu ze ściskiem. Podobnie jak Covacchio w celu zachowania różnorodności w populacji skupił się na sposobie wprowadzania nowego pokolenia. W modelu tym używa się populacji mieszanych (wielopokoleniowych) tzn. takich, w których tylko część populacji może reprodukować i umiera w każdym pokoleniu. Każdy nowo wygenerowany osobnik zastępuje najbardziej podobnego do siebie osobnika ze starej populacji. Aby uzyskać przybliżenie takiego „najbardziej podobnego" zastępowania, brana jest mała próbka z istniejącej populacji i nowo wygenerowane osobniki zastępują najbardziej podobnych osobników z tej próbki.

Model ze ściskiem działa w następujący sposób. Odsetek populacji, określony przez parametr GG [generetrion gap], jest wybrany przez selekcje (promującą najlepiej przystosowane osobniki) i poddany krzyżowaniu oraz mutacji. Po krzyżowaniu i mutacji, GG*n osobników z populacji jest wybierana do zastąpienia przez nowe potomstwo. Dla każdego potomka losowo wypierana jest z podpopulacja o wielkości CF [crowding factor] – współczynnik ścisku. Potomek zastępuje najbardziej podobnego do siebie osobnika z tej podpopulacji. Podobieństwo jest definiowane jako odległość Hamminga między binarną reprezentacją dwóch genotypów. De Jung odniósł sukces, stosując model ze ściskiem do funkcji wielomodalnej przy czynniku ścisku CF równym 2 lub 3.

Metoda ta jest inspirowana na zjawisku ekologicznym, w którym osobniki w naturalnej populacji, często na tym samym terenie, rywalizują z sobą o ograniczone zasoby. Zróżnicowanie osobników prowadzi z czasem do zajęcia różnych nisz tak, że w zasadzie nie zachodzi rywalizacja. Ostatecznym rezultatem jest to, że w zrównoważonej populacji o stałej wielkości, nowe osobniki z poszczególnych gatunków zastępują starych. W idealnej sytuacji, całkowita ilość członków danego gatunku nie zmienia się.

Oczywiście metoda ze ściskiem nie zakłada, że wejściowa populacja jest stabilną mieszanką gatunków, lecz dąży do utrzymania zróżnicowania w początkowej populacji. Można by zadać pytanie, dlaczego nie zostawić populacji początkowej, przecież jest zróżnicowana. Niestety ten sposób nie promuje dobrego zróżnicowania. Dobre zróżnicowanie definiujemy jako istnienie reprezentantów wszystkich gatunków, włączając optymalne osobniki z każdego gatunku. Gatunki różnią się tak samo jak lokalne optima w przestrzeni poszukiwań. Każde optimum rozważane jest jako nisza.

W [1] przeprowadzone zostały badania z 6 odmianami algorytmu. Dowiedziono, że najbardziej efektywny jest algorytm znany pod nazwą dereminisic crowding (DC), który działa w następujący sposób. Najpierw każdy element populacji grupowany jest w pary (jest ich n/2). Następnie osobniki z każdej pary są krzyżowane a potomstwo ulega mutacji. Każdy potomek rywalizuje z rodzicem bardziej podobnym do siebie. Poniżej przedstawiam pseudokod prezentowany w [2] .

Deterministic Crowding

(REPEAT for g generations)

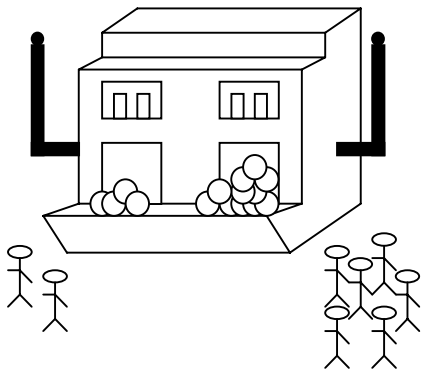
DO $n/2$ times:

1. Select 2 parents, p_1 and p_2, randomly, no replacement
2. Cross them, yielding c_1 and c_2
3. Apply mutation / other operators, yielding c_1' and c_2'
4. IF $[d(p_1,c_1') + d(p_2,c_2')] \leq [d(p_1,c_2') + d(p_2,c_1')]$
   - IF $f(c_1') > f(p_1)$ replace p_1 with c_1'
   - IF $f(c_2') > f(p_2)$ replace p_2 with c_2'

   ELSE
   - IF $f(c_2') > f(p_1)$ replace p_1 with c_2'
   - IF $f(c_1') > f(p_2)$ replace p_2 with c_1'

2.3 Sharing

Pierwszemu, któremu udało się przenieść idee nisz ekologicznych i specjacji na grunt algorytmów genetycznych był Holladnd (1975). Aby zilustrować mechanizm nisz i specjacji, posłużył się zagadnieniem dwuramiennego bandyty z podziałem wypłat.



Wyobraźmy sobie dwuramiennego bandytę (powyższy rysunek). Mamy tu dwa ramiona i z każdym z nich związana jest inna średnia wypłata. Przypuśćmy, że dla prawego ramienia wynosi ona 75 dolarów, a dla lewego 25 dolarów. Załóżmy następnie, że mamy populację złożoną ze 100 graczy i że każdy z nich otrzymuje pełną kwotę wygranej związaną z ramieniem, które wybrał w danej próbie. Jeśli poprzestaniemy na tym i pozwolimy graczom wybierać dowolnie ramię, to jeśli gracze „reprodukują się" proporcjonalnie do przystosowania, to coraz większa liczba członków populacji powinna skupić się w kolejce do prawego ramienia.

Wprowadzimy teraz istotną modyfikację do zasad gry, która spowoduje tworzenie się stabilnych podpopulacji wokół każdego z ramion. Zamiast wypłacać

pełną wygraną każdemu osobnikowi, będziemy ją dzielić pomiędzy graczy z danej kolejki. W zmodyfikowanej grze każdy gracz otrzymuje wypłatę zależną od ramienia, które wybrał oraz od liczby osobników stojących w tej samej co on kolejce. Jeżeli w naszym przykładzie osobnik wybierze ramię prawe – w przypadku gdy wszyscy pozostali gracze stoją w tej samej kolejce to otrzyma on wypłatę w wysokości \$75/100=\$0,75. Z drugiej strony, jeżeli wszyscy osobnicy wybraliby lewe ramię to otrzymaliby wypłatę w kwocie \$25/100=\$0,25. W obu przypadkach pewna liczba graczy ma motywację do zmiany kolejki. W pierwszym przypadku gracz zmieniając kolejkę zarabia \$25,00-\$0,75=\$24,25. W drugim przypadku motywacja do zmiany kolejki jest jeszcze większa. Możemy się więc spodziewać, że gdzieś pośrodku leży punkt, w którym nikomu nie będzie się opłacało zmieniać kolejkę.

Praktyczna metoda kreowania nisz i gatunków, oparta bezpośrednio na przedstawionej idei podziału zasobów, została opisana przez Goldberga i Richardsona (1987). Wprowadza się tam funkcję współudziału [sharing function], która określa sąsiedztwo i stopień współudziału dla każdego ciągu kodowego w populacji. Liczbę współudziałów dla danego osobnika oblicza się, sumując wszystkie wartości funkcji współudziałów wnoszone przez inne ciągi w populacji. Ciągi znajdujące się w bliskim sąsiedztwie danego osobnika mają duże współudziały (bliskie jedności), natomiast ciągi odległe – bardzo małe współudziały (bliskie zeru). Po zsumowianiu wszystkich obliczonych w ten sposób współudziałów, zdeprecjonowany wskaźnik przystosowania oblicza się dzieląc jego potencjalny wskaźnik przystosowania przez łączną liczbę wsółudziałów:

$$f_s(x_i) = \frac{f(x_i)}{\sum_{j=1}^{n} s(d(x_i, x_j))}$$

gdzie :

$d(x_i,x_j)$- funkcja podobieństwa między osobnikiem x_i ,a x_j, może być rozpatrywana na poziomie genotypu (np. odległość Haminga między ciągami), lub na poziomie fenotypu.

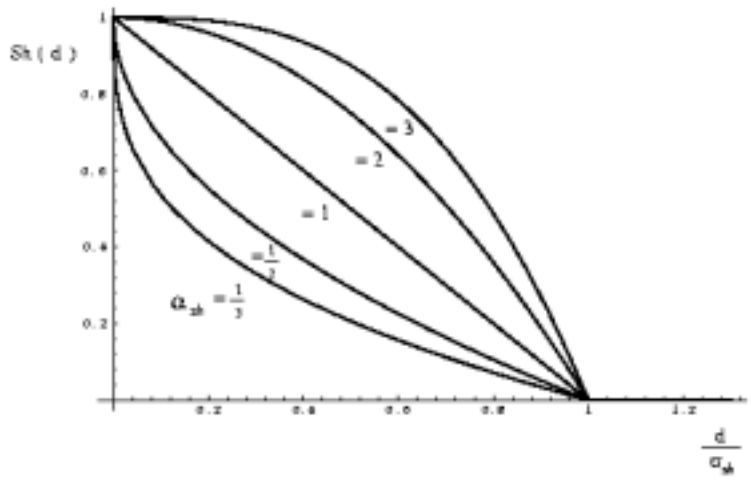
S()- funkcja współudziału określona w następujący sposób:

$$Sh(d) = \begin{cases} 1 - (\frac{d}{\sigma_{share}})^{\alpha} & dla \quad d < \sigma_{share} \\ 0 & w\_przeciwnym\_przypadku \\ . \end{cases}$$

gdzie:

σ_{share} - stały parametrem określającym wielkość niszy.

α – stały parametr od którego zależy kształt funkcji współudziału (poniższy rysunek). Do tej pory nie zbadano jaki wpływ ma ten parametr na działanie algorytmu dlatego zazwyczaj jest on równy 1.



postać funcji Sh(d) dla różnych wartości parametru α

2.4 Sequential niching

Niszowanie sekwencyjne (SN) po raz pierwszy zaprezentowane zostało przez Beasley`a, Bull`a, i Martin`a (1993). Metoda ta polega na wielokrotnym uruchomianiu prostego AG i utrzymywanie najlepszego rozwiązania z każdej iteracji. Aby uniknąć wielokrotnego zbiegania się populacji do tych samych obszarów przestrzeni rozwiązań za każdym razem, gdy SN znajdzie rozwiązanie, zmniejsza wartość funkcji w pewnym otoczeniu rozwiązania (podobnie jak parametr σ_{share} w metodzie sharing). Autorzy tej metody zauważyli trzy potencjalne korzyści wynikające z jej stosowania. Pierwsza to prostota: SN łatwo dodać do istniejącej metody optymalizacji. Druga to możliwość pracy z mniejszymi populacjami, ponieważ za każdą iteracją algorytm ma za zadanie zlokalizować tylko jeden szczyt. Trzecia to szybkość, i ta zaleta jest częściowo produktem ubocznym drugiej zalety. Jak zauważyli autorzy pracy [2] co najmniej dwie z tych korzyści nigdy nie występuje, a tak naprawdę bardzo szybko pojawia się wiele wad: wielokrotne przeszukiwanie obniżonych obszarów, wielokrotne zbieganie się do tego samego rozwiązania, powolność działania.

2.5 Clearing procedure

Procedura clearing jest metodą niszowania inspirowaną na zasadzie prezentowanej przez J.H. Hollanda w 1975: dzielenie skończonych zasobów między osobniki z tej samej subpopulacji (części populacji złożonej z podobnych osobników). Jednakże zamiast równomiernie dzielić zasoby między członków subpopulacji, procedura clearing oddaje całość zasobów tylko najlepszym osobnikom z każdej subpopulacji.

Procedura clearing stosowana jest po obliczeniu przystosowania każdego osobnika, a przed zastosowaniem operacji selekcji. Podobnie jak sharing method procedura clearing używa odległości Haminga do określenia czy dwa osobniki należą do tej samej subpopulacji czy nie.

Każda subpopulacja zawiera osobnika dominującego: tego, który ma najlepsze przystosowanie. Osobnik należy do danej subpopulacji, jeżeli różni się on od osobnika dominującego mniej niż parametr σ [clearing radius]. Podstawowy algorytm clearing zachowuje przystosowanie osobnika dominującego, a przystosowanie innych osobników tej samej subpopulacji zeruje. W ten sposób procedura clearing przypisuje cały zasób niszy jednemu osobnikowi. Osobnik dominujący zabiera cały zasób zamiast dzielić się nim z innymi osobnikami z tej samej niszy, jak to było w metodzie shering.

Działanie takiego mechanizmu nie pozwala nam dowiedzieć się do jakich nisz należa poszczególne osobniki populacji. W zasadzie dany osobnik może być zdominowany przez wiele innych. Ale z drugiej strony, dla danej populacji, zbiór osobników dominujących jest unikatowy. To twierdzenie jest dowiedzione przez indukcje: osobnik, który jest lepiej przystosowany w populacji staje się z konieczności osobnikiem dominującym. Osobnik ten wraz z wszystkimi, które zdominował zostaje fikcyjnie usunięty z populacji. Tak samo postępujemy z pozostałą częścią populacji. W ten sposób lista dominantów tworzy się po pewnej liczbie kroków.

Możliwe jest także uogólnienie algorytmu clearing poprzez akceptowanie kilku osobników dominujących wybranych spośród najlepiej przystosowanych osobników w danej niszy. Pojemność niszy jest definiowana jako maksymalna ilość osobników dominujących, których może zaakceptować nisza. Zauważmy, że jeżeli pojemność niszy jest większa niż jeden wtedy zbiór osobników dominujących nie jest w ogólności unikatowy. Jeżeli pojemność ustalimy taką, jaka jest liczba osobników to otrzymamy prosty AG.

Poniżej przedstawiona jest w pseudo kodzie prosta wersja procedury clearing. P i n są zmiennymi globalnymi. „Sigma" to omawiany wcześniej parametr σ(clearing radius), „Kappa" jest pojemnością każdej niszy. „nbWinners" wskazuje ilość osobników dominujących w subpopulacji związanej z daną niszą. Populacja P może być rozważana jako tablica n elementowa.

```
function Clearing(Sigma, Kappa)

begin
  SortFitness(P)
  for i = 0 to n - 1
    if Fitness(P[i]) > 0
      nbWinners := 1
      for j = i + 1 to n - 1
        if Fitness(P[j]) > 0 and
            Distance(P[i], P[j]) < Sigma
          if nbWinners < Kappa
            nbWinners := nbWinners + 1
          else
            Fitness(P[j]) := 0.0
          endif
        endif
      endfor
    endif
  endfor

end
```

Algorytm używa trzech funkcji: SortFitnes()- funkcja sortująca populację P według rosnącego przystosowania, Fitness(P[i]) – funkcja zwracająca przystosowanie i-tego osobnika populacji P, Distance(P[i],P[j]) – funkcja zwraca różnicę (np. odległość Haminga) między osobnikiem i-tym a j-tym populacji P.

2.6 Overspecification

W pracy [5] pod określeniem overspecification rozumie się ogólną idee wykorzystania nadmiarowości w genotypie do poszukiwania nisz. Dokładniejsze informacje o działaniu diploidalnego aparatu genetycznego można znaleźć w [4]. W rozdziale tym chciałem przedstawić kilka praktycznych zastosowań tej metody.

Dotychczas rozważaliśmy jedynie najprostszy rodzaj genotypów spotykanych w przyrodzie – o haploidalnej liczbie chromosomów. W modelu tym pojedynczy ciąg kodowy zawiera całą informację istotną dla rozpatrywanego zagadnienia. Mino, że przyroda zna wiele organizmów haploidalnych, większość z nich reprezentuje raczej nieskomplikowane formy życia. Wydaje się, że ilekroć przyroda chciała wytworzyć wyższe postaci życia roślinnego lub zwierzęcego, musiała polegać na bardziej skomplikowanej strukturze genetycznej – genotypach o diploidalnej liczbie chromosomów. Genotyp w postaci diploidalnej składa się z jednej lub więcej par chromosomów (zwanymi chromosomami homologicznymi), każdy z nich przenosi informację służącą tym samym funkcjom.

Diploidalność od dawna były przedmiotem studiów genetycznych, przedstawiono liczne teorie mające wyjaśnić jej rolę. Z naszego punktu widzenia najbardziej interesujące są hipotezy, według których podwójny zestaw chromosomów stanowi mechanizm do zapamiętywania tych alleli oraz ich kombinacji, które w przeszłości okazały się pożyteczne.

Badania przeprowadzone przez Smitha (1988) pokazały brak korzyści z zastosowania diploidalnego genotypu w optymalizacji statycznych funkcji.

Goldberg i Smith (1987) zaproponowali, że skoro diploidalny genotyp nie przynosi korzyści dla statycznego środowiska to może przyniesie dla zmieniającego się w czasie środowiska. Diploidalny chromosom powinien być zdolny do przechowywania informacji o rozwiązaniach, które okazały się dobre w przeszłości (mogą być przydatne w przyszłości). Skutecznie zastosowali oni diploidalność do śledzenia globalnego optimum, które zmieniało się regularnie pomiędzy dwoma wartościami. Populacja utrzymywała w chromosomie obie lokalizacje optimum. Chociaż tylko jedna lokalizacja była, w danym czasie, wyrażona.

O tym, który allel będzie dominujący decyduje mapa dominacji. Każda pozycja w diploidalnym chromosomie zawiera dwa allele, ale w danym czasie tylko jeden z nich jest wyrażany (jest dominujący), jest używany do obliczenia fenotypu i później wartości funkcji. Kiedy środowisko jest stale zmieniane, wcześniej użyteczne rozwiązania zapisane są w nie wyrażonej części genotypu całej populacji. Jeżeli środowisko później zmieni się na podobne do tego, co wcześniej, zachowane rozwiązanie może zostać ponownie wykorzystane. Zauważmy, że populacja jako całość przechowuje całe rozwiązanie. To rozwiązanie musi zostać złożone z kawałków zawartych w poszczególnych osobnikach. Aktualnie nie wyrażone rozwiązanie ulega ewolucji przez mapę dominacji.

Goldberg i Smith ograniczyli swoją uwagę do funkcji, która oscylowała pomiędzy dwoma optimami. Nie jest jasne czy diploidalny chromosom jest wystarczający do śledzenia więcej niż dwóch optimów, czy potrzebny jest wtedy poliploidalny chromosom. Nie jest także jasne czy diploidalny chromosom będzie przydatny do śledzenia środowiska, które przechodzi w całkiem nowe stany. Może istnieją pewne stopnie podobieństwa pomiędzy nowym środowiskiem a którymś ze starych. Wstępne aplikacje do nowych środowisk wydają się obiecujące, jak pokazał Hillis (1990).

Wiadomym jest, że nadmiarowość czy w dziedzinie komputerów czy w żywych organizmach chroni przed stratą informacji. Przeanalizowaliśmy już korzyści z nadmiarowości w diploidalnych chromosomach. Alternatywą może być trzymanie nadmiarowości (być może konfliktowych informacji) w dodatkowych segmentach haploidalnego chromosomu. Goldberg (1989c) zauważył możliwość użycia różnej długości łańcuchów w kombinacji z operatorami duplikacji (duplication) i usuwania (deletion).

AG, który wykorzystuje overspecification w haploidalnym chromosomie nazywany jest messy GA. Messy GA zawiera różnej długości łańcuchy i radzi sobie z overspecification używając pozycjonalnej procedury. Po zastosowaniu opretorów genetycznych wcześniej ukryty gen może stać się z powrotem dominującym.

2.7 Porównanie metod crowding, sharing, sequential niching i parallel hillclimbing

W rozdziale tym pragnę przedstawić porównanie metod niszowania przedstawione w [2]. W artykule tym zdefiniowana jest metoda parallel hillclimbing, stanowi ona ważne odniesienie w porównywaniu metod niszowania.

Metoda parallel hillclimbing rozpoczyna się od wygenerowania losowej populacji i wymusza na każdym z jej elementów dążenie do najbliższego optimum. Operator sąsiedztwa zdefiniowany zarówno na poziomie fenotypowym jak genotypowym .

Poniżej przedstawiam pseudokod :

Parallel Hillclimbing (Phenotypic)

1. Initialize *Step Size*
2. WHILE *Step Size* $\geq \epsilon$
   (a) FOR each population element
   - Randomly pick a starting variable
   - *Change* = TRUE
   - WHILE *Change*
     - *Change* = FALSE
     - FOR each variable
       - IF adding *Step Size* to current variable yields improved fitness
         - Perform the addition
         - *Change* = TRUE
       - ELSE IF subtracting *Step Size* from current variable yields improved fitness
         - Perform the subtraction
         - *Change* = TRUE

   (b) *Step Size* = *Step Size* / 2

Metoda jest podobna do prostej operacji szukania binarnego. Jeżeli operujemy na fenotypowej przestrzeni to algorytm rozpoczyna się od dużych kroków. Każdy osobnik populacji wspina się o określoną długość kroku tak długo, jak długo przynosi to pozytywne rezultaty. Jeżeli wykonanie kroku o pierwotnej długości nie powoduję zwiększenia przystosowania, wtedy długość kroku jest zmniejszana o połowę i ponownie zaczyna się wspinanie wszystkich osobników. Jeżeli operator sąsiedztwa operuje na przestrzeni fenotypowej to sprawdzenie czy dana długość kroku jest jeszcze przydatna następuje poprzez wykonanie „testowego" kroku we wszystkich możliwych kierunkach przestrzeni poszukiwań. Algorytm powtarzany jest tak długo aż długość kroku osiągnie minimalną wartość ε.. Problem jest z dobraniem początkowej długości kroku. Jeżeli krok będzie zbyt mały to będzie to prowadziło do bardzo długiego działania. Jeżeli krok będzie zbyt duży osobniki będą przeskakiwały z jednego optimum do innego. W przypadku, gdy sąsiedztwo obliczamy na podstawie genotypu wtedy długość kroku niezmiennie wynosi jeden bit.

Parallel hillclimbing jest ważną podstawą do porównania, ponieważ wykorzystane w niej mechanizmy uniemożliwiają zbiegania się całej populacji na globalnego optimum.

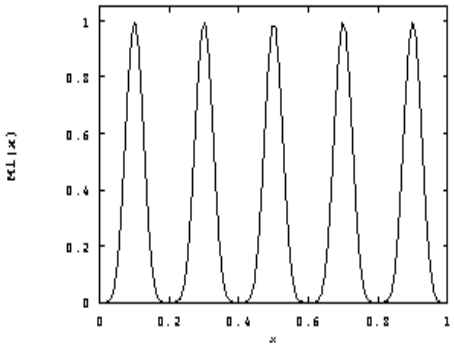
Do testów wykorzystano 11 problemów, różniących się trudnością M1-M9 to funkcje multimodalne:

$$
\begin{aligned}
M1(x) &= \sin^6(5\pi x) \\
M2(x) &= e^{-2(\ln 2)\left(\frac{x-0.1}{0.8}\right)^2} \sin^6(5\pi x) \\
M3(x) &= \sin^6(5\pi[x^{0.75} - .05]) \\
M4(x) &= e^{-2(\ln 2)\left(\frac{x-0.08}{0.854}\right)^2} \sin^6(5\pi[x^{0.75} - .05])
\end{aligned}
$$

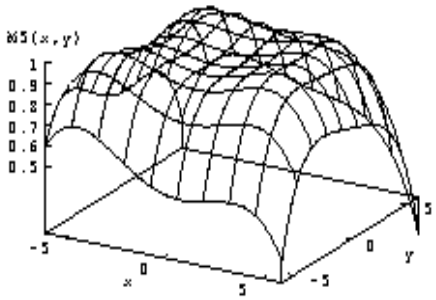
$$
M5(x,y) = \frac{2186 - (x^2 + y - 11)^2 - (x + y^2 - 7)^2}{2186}
$$

$$
M6(x,y) = 500 - \frac{1}{.002 + \sum_{i=0}^{24} \frac{1}{1+i+(x-a(i))^6+(y-b(i))^6}}
$$

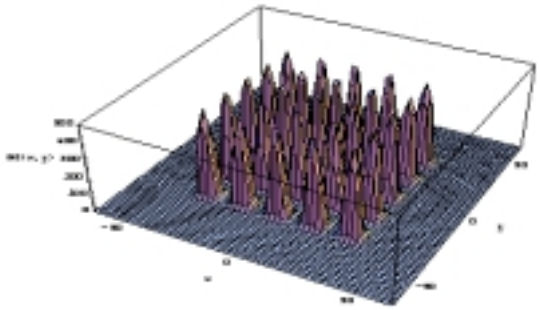
$$
M7(x_0, \ldots, x_{29}) = \sum_{i=0}^{4} u\left(\sum_{j=0}^{5} x_{6i+j}\right) \qquad M8 = 5\left(\frac{M7}{5}\right)^{1.5}
$$



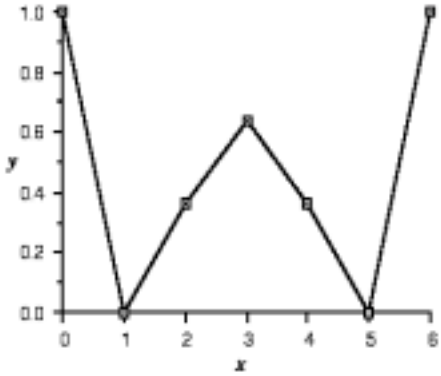
M1(x)



M5(x)



M6(x)



y=u(x)

MUX-6i PAR-8 są problemami klasyfikacji.

Każda próba (testowanie danego algorytmu na danej funkcji) rozpoczynała się od minimalnej wielkości populacji tzn. takiej która była potęgą dwójki i umożliwiała znalezienie wszystkich rozwiązań (n=2 dla SN). Następnie podwajano n i powtarzano całą sekwencję aż do znalezienia wszystkich rozwiązań lub wyczerpania się cierpliwości badaczy. Badacze tracili cierpliwość, gdy ilość obliczeń funkcji w jednym pokoleniu (dla AG)wynosiła 1.5 miliona lub ogólna ilość obliczeń funkcji wynosiła 2.0 miliony (dla AG i HC). Dokładniejsze informacje o sposobie przeprowadzonego porównania można uzyskać z [2].

A oto otrzymane wyniki:

Method	$\bar{n}$	$\bar{g}$	*GA: μ*	*Combo: μ*
		M1		
HC	2.72			**1017**
SN	3.68	46.40	738	4112
SH	5.76	8.00	**264**	2431
DC	2.40	28.00	380	1246
		M2		
HC	2.72			**1021**
SN	4.64	75.60	1770	8632
SH	8.96	8.70	442	3827
DC	2.40	27.40	**372**	1264
		M3		
HC	3.04			1150
SN	5.92	26.80	719	4375
SH	6.08	8.70	294	2579
DC	2.08	20.30	**262**	**1013**
		M4		
HC	3.04			1140
SN	5.12	72.40	2445	10231
SH	6.72	9.20	352	2892
DC	2.08	17.00	**210**	**975**
		M5		
HC	2.50			**901**
SN	1.30	32.30	180	1456
SH	2.80	8.00	**103**	1111
DC	5.60	25.60	603	2459
		MUX-6		
HC	10.40			**1257**
SN	6.40	140.70	4423	9439
SH	13.60	8.90	**534**	1931

Method	$\bar{n}$	$\bar{g}$	*GA: μ*	*Combo: μ*
		M6		
HC	12.29			29,017
SN	3.58	146.30	12,202	46,657
SH	5.12	11.80	**1,638**	**12,910**
DC			$> 1.5 \times 10^6$	
		PAR-8		
HC	48.08			202,387
SN	19.20	36.40	100,557	263,666
SH	9.60	12.60	**17,203**	**54,402**
DC	11.20	87.40	125,850	149,022

Method	$\bar{n}$	$\bar{g}$	*GA: μ*	*Combo: μ*
		M7		
HC				> 2000K
SN			> 1500K	
SH			> 1500K	
DC	20.80	119.80	**81K**	**101K**
		M8		
HC				> 2000K
SN			> 1500K	
SH	19.20	19.20	**13K**	**38K**
DC	22.40	134.40	98K	119K
		M9		
HC				> 2000K
SN			> 1500K	
SH				> 2000K
DC	136.53	337.80	**1253K**	**1342K**

Oznaczenia:
(HC) – parallel hilclimbing
(SN)- sequential niching
(SH) – fintess sharing
(DC) – deterministic crowding

$\bar{n}$ - średni rozmiar subpopolacji

$\bar{g}$ - średnia liczba pokoleń

GA:μ – ilość obliczeń funkcji dla każdego pokolenia
Combo: μ – całkowita ilość obliczeń funkcji
K - tysiące
nalepsze wyniki zostały pogrubione

Analiza rezultatu porównania:

- Metoda parallel hillclimbing jest najlepsza dla łatwych problemów. Może także pracować w rozsądnym czasie dla problemów o średniej złożoności. Jednakże metoda ta zawodzi dla bardziej złożonych problemów.

- Metoda sequential niching słaba dla prostych problemów i nie jest w stanie rozwiązywać trudniejszych problemów. Generalnie parallel hillclimbing jest lepsza.
- Metoda sharing może pracować dla problemów o różnym stopniu złożoności. Jednakże nie radzi sobie dobrze z problemami, w których istnieje wiele ubocznych szczytów o podobnej wysokości (patrz wynik dla funkcji M-6).
- Metoda deterministic crowding jest w ogólności dobra dla problemów z wszystkich poziomów złożoności. Jednakże może ostatecznie zgubić niższe optima leżące na drodze do wyższego optimum.

3 Koewolucja

Zasada koewolucji polega na tym, że globalne oddziaływania na poziomie selekcji i krzyżowania, stają się w pewnym stopniu lokalne. Zasadę tą realizuje się podstawowymi algorytmami.

3.1 Podstawowe algorytmy

3.1.1 Algorytm wyspowy

W algorytmie tym mamy wiele populacji, które ewoluują niezależnie od siebie, sporadycznie wymieniając między sobą informację. Koewoluujące populacje mogą się składać z osobników tego samego typu bądź różnych typów. Może się zdarzyć że w różnych populacjach obowiązywać będą różne funkcje przystosowania, wówczas występuje często zależność między funkcjami przystosowania poszczególnych populacji.

W algorytmie wyspowym mamy do czynienia z konkurencją w ramach każdej populacji o ograniczone zasoby zaś migracja i krzyżowanie osobników z różnych populacji zdarza się dość rzadko.

Algorytm wewnątrz każdej populacji:

T:=0

Inicjacja P_t

Ocena P_t

While (not warunek stopu) do

Begin

T_t := reprodukcja P_t

O_t := operacje genetyczne T_t

Ocena Ot

P_{t+1} := sukcesja (P_t ,O_t)

Wymiana osobników z innymi populacjami

T:=t+1

End

Algorytm ten dopuszcza dopływ osobników z zewnątrz jak i wysłanie osobnika na zewnątrz. Przy czym słowo z zewnątrz oznacza z innej populacji i na zewnątrz - do innej populacji.

Spotyka się dwa modele tej wymiany:

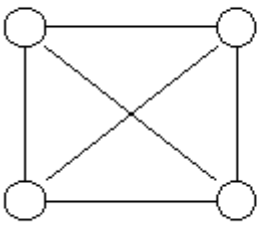
Emigracja - osobnik jest usuwany z jego populacji bazowej

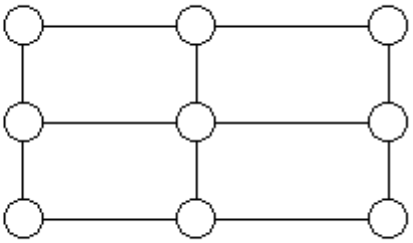
Imigracja - wysyłana jest kopia osobnika, natomiast on sam pozostaje w populacji bazowej.

Przyjęcie osobnika do populacji odbywa się na zasadzie podobnej do selekcji. (Nowy osobnik konkuruje ze starym o miejsce w populacji)

Wymiana informacji między populacjami może dokonywać się różnymi sposobami. Zbiór populacji może być nieuporządkowany albo może występować pewna relacja sąsiedztwa. W pierwszym przypadku wymiana osobników (lub innych informacji) następuje między dwoma dowolnymi populacjami, w drugim przypadku wymiana jest ograniczona do populacji sąsiadujących ze sobą.

Przykładowe topologie:





3.1.2 Algorytm komórkowy

W populacji osobników występuje pewna arbitralnie wybrana topologia. Dla każdego osobnika X_t jest zdefiniowane jego sąsiedztwo $N_r(X_t)$ (r- promień sąsiedztwa, osobniki często należą do wielu populacji tworzonych przez relacje sąsiedztwa). Każdy osobnik podlega niezależnej ewolucji, która polega na tym, że wchodzi on w pewną interakcję z osobnikami z otoczenia, co prowadzi do wygenerowania nowego osobnika który konkuruje z poprzednikiem na etapie sukcesji.

Algorytm ten nazywany jest niekiedy dyfuzyjnym ze względu na sposób rozprzestrzeniania się rozwiązania (osobnik dobrze przystosowany będzie miał duży wpływ na nowo powstające osobniki w jego sąsiedztwie). W kolejnych pokoleniach dobre rozwiązanie rozprzestrzenia się na sąsiadów dobrze przystosowanego osobnika, później na ich sąsiadów itd...

Algorytm komórkowy:

T:=0

Inicjacja X_t

Ocena X_t

While (not warunek stopu) do

Begin

T_t := reprodukcja (X_t +$N_r(X_t)$)

Y_t := operacje genetyczne (T_t)

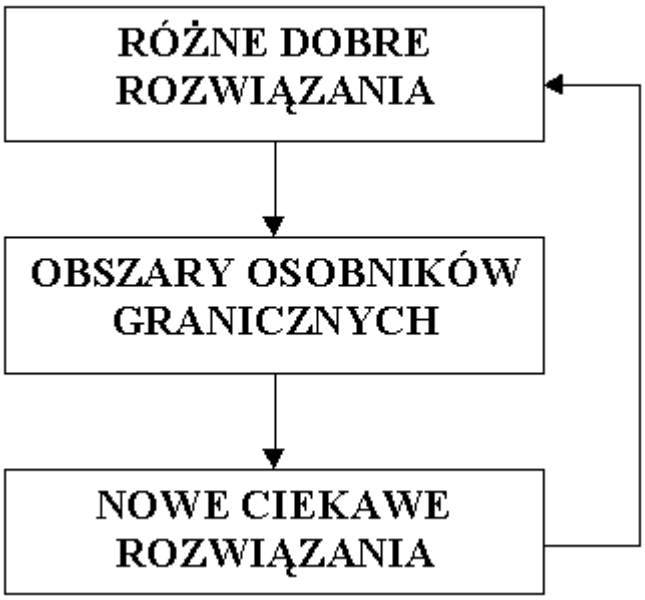
Ocena Y_t

X_{t+1}:= sukcesja ({X_t ,Y_t})

T:=t+1

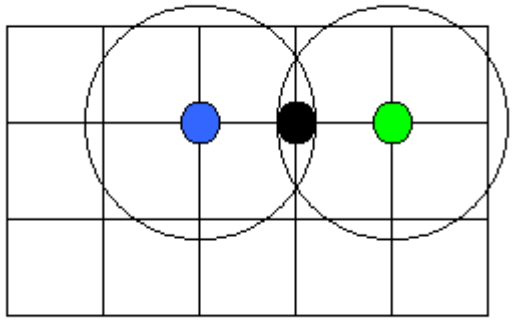
End

W momencie powstania rozwiązań równie dobrych powstają między nimi obszary graniczne (oddzielające strefy dominacji równie dobrych rozwiązania). W obszarach tych często powstają nowe ciekawe osobniki, o jeszcze większym przystosowaniu które znowu się rozprzestrzeniają tworząc nowe obszary graniczne:



Kluczową rolę w tym algorytmie odgrywa relacja sąsiedztwa, wykorzystywana do określania konkurujących ze sobą osobników. Im więcej osobników w sąsiedztwie tym szybsza zbieżność ale za to mniejsza odporność na maksima lokalne.

Przykładowa topologia:



3.2 Eksperymenty

3.2.1 Zastosowanie koewolucji w sortowaniu łańcuchów

Zadanie: Znalezienie algorytmu sortującego dowolną 16 elementową listę w jak najmniejszej liczbie porównań.

Problem ten najpierw rozwiązali ludzie, otrzymując algorytmy odpowiednio:

ROK	ILOŚĆ PORÓWNAŃ
1962	65
1964	63
1969	60

W 1980 roku problemem zainteresował się Hillis. Postanowił on zastosować do rozwiązania algorytm genetyczny. Początkowo był to algorytm nie stosujący koewolucji. Hillis podpowiedział mu pierwsze 32 kroki takie jakie były w algorytmach wymyślonych przez ludzi, oraz ustanowił populację na ilość pomiędzy 512 a 1 mln. Mimo takich parametrów algorytm genetyczny znalazł rozwiązanie sortujące w 65 porównaniach. Tak więc algorytm ten znalazł pewne rozwiązanie problemu, ale ponieważ ludziom udało się znaleźć lepsze rozwiązania (i to o aż 5 kroków !) można wnioskować, że algorytm zatrzymał się na minimum lokalnym.

W związku z tym Hillis zastosował algorytm koewolucyjny, bazujący na "wyścigu zbrojeń".

W algorytmie tym można wyróżnić populację żywicieli i pasożytów (podobne do algorytmu wyspowego). Populację żywicieli stanowiły testowane algorytmy, natomiast populację pasożytów stanowiły zbiory ciągów testowych.

Funkcje przystosowania były zdefiniowane dla poszczególnych populacji następująco:

- żywicieli: procent dobrze posortowanych ciągów testowych
- pasożytów: procent źle posortowanych ciągów testowych

Po zastosowaniu takiego algorytmu otrzymano algorytm sortujący w 61 porównaniach czyli o 4 lepiej niż w przypadku zastosowania normalnego algorytmu genetycznego.

3.2.2 Zastosowanie koewolucji w robotyce

Zadanie

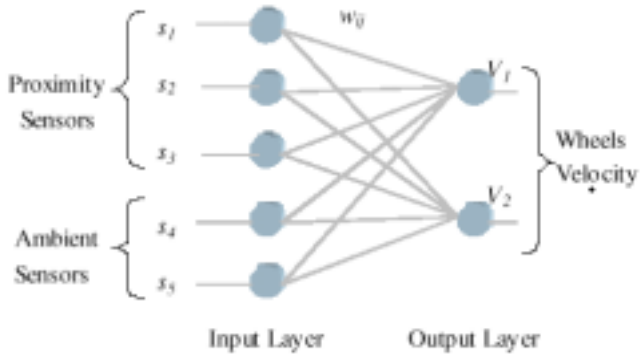
zapewnienie robotowi bezpiecznego, bezkolizyjnego przemieszczania w dowolnym środowisku

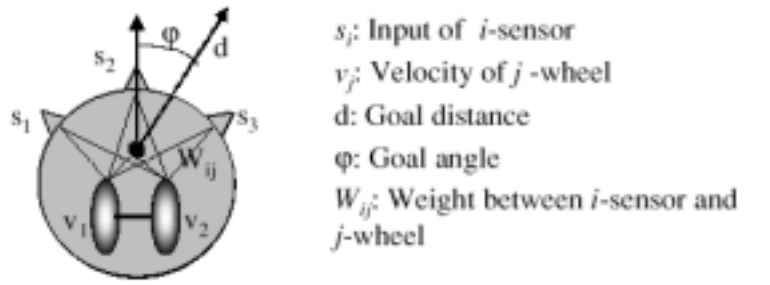
Eksperyment przeprowadzony i opisany przez: A.Berlanga, A.Sanchis, P.Isasi, J.M.Molina. Eksperyment ten polegał próbie nauczenia robotów bezkolizyjnego przemieszczania się w różnych środowiskach. Przeprowadzono eksperyment z wykorzystaniem koewolucji oraz bez wykorzystywania metod koewolucyjnych. Do symulacji użyto mini-robota Khepera. Rezultatem eksperymentu było pokazanie iż algorytm koewolucujny znalazł lepsze ogólne rozwiązania umożliwiające mu poruszanie się w różnych środowiskach, natomiast zwykły algorytm ewolucyjny znalazł rozwiązanie tylko dla środowiska w którym się poruszał.

Zastosowana matoda koewolucyjna bazowała na algorytmie "wyścigu zbrojeń".

ROBOTY

W eksperymencie użyto robotów mających dwa silniczki (każdy połączony z gąsienicą) stanowiących napęd robota, oraz 5 sensorów sterujących pracą tych silników. Wyróżniono 3 sensory badające bliskie otoczenie (Proximity sensors) oraz 2 sensorów badających dalekie otoczenie (ambient sensors).





Zadaniem sensorów badających dalekie otoczenie było mierzenie odległości do punktu do którego robot chciał dotrzeć oraz kąta między skierowaniem robota a tym punktem. Ponadto każdy sensor dawał każdemu silnikowi jego aktualną wartość (np. przeszkoda, droga wolna). Ponadto każdy sensor miał pewną wagę w która mówiła jak ważne są informacje od tego sensora dla danego silnika. Akcję silnika wyliczano ze wzoru:

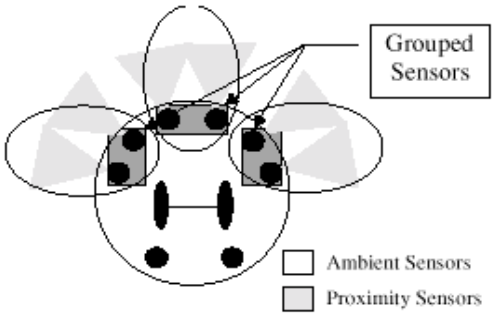
$$v_j = f\left(\sum_{i=1}^{5} w_{ij} \times s_i\right) \qquad (1)$$

Gdzie f jest funkcją rozstrzygającą o akcji robota w zależności od parametru wywołania. Argumentem jest suma iloczynów wag i sygnałów konkretnych sensorów dla danego silnika. Wynikiem jest v czyli prędkość jaką miał dany silnik nadać robotowi.

W eksperymencie poszukiwano zatem tylko wektora wag który rozstrzygał o zachowaniu robota w zależności od sygnałów ze środowiska.

Charakterystyka robota Khepera:

- 5,5 cm średnicy
- 3 cm wysokości
- 70 gr wagi
- 8 sensorów czułych na podczerwień (sensor składał się z emitera i odbiornika, były one niezależne co pozwalało aby odbiornik mierzył światło odbijane, gdy był włączony emiter lub światło otoczenia gdy emiter był wyłączony - pozwalało to ustalić rozmieszczenie przeszkód na drodze robota)
- sensory były pogrupowane - sensor mierzący bliskie otoczenie składał się z dwóch sensorów czułych na podczerwień



Środowisko życia robotów było prostokątem z pewnym stałym układem przeszkód.

Algorytm koewolucyjny:

Jak już wspomniano algorytm bazuje na wyścigu zbrojeń. Wyróżniono dwie populacje:

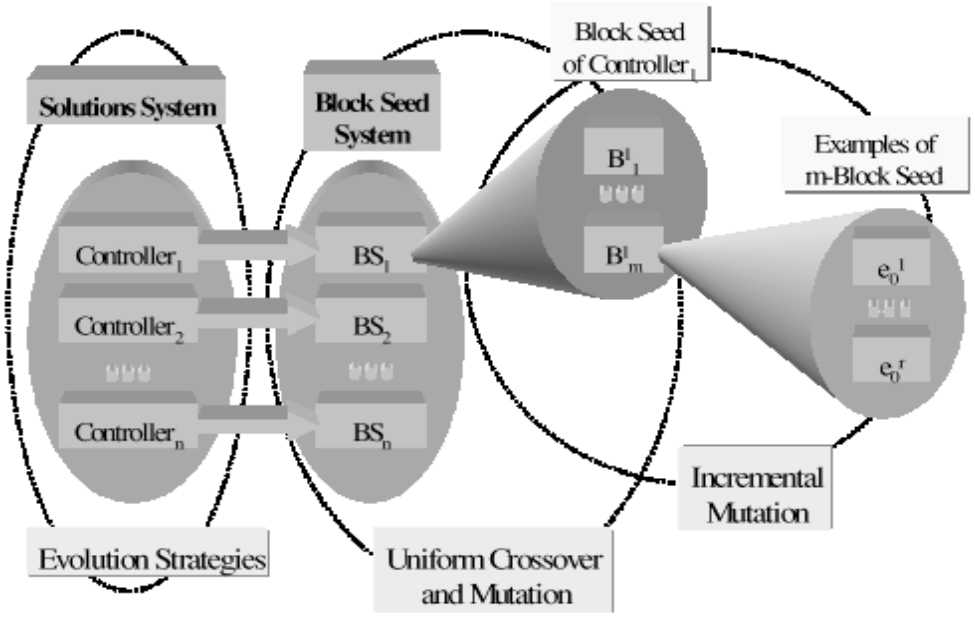
- populację rozwiązań (robotów, wektorów wag sensorów)
- zbiorów "ziaren" generujących pozycję początkową robota.

Jak już wspomniano robot ewoluował w środowisku w którym układ przeszkód był stały, stały był również punkt docelowy robota. Każdy robot był

związany z pewnym zbiorem ziaren. Każde ziarno za pomocą metody nazwanej "inkrementowaną mutacją" generowało zbiór punktów początkowych robota oraz jego skierowanie dla każdego punku startowego. Na podstawie zachowania robota dla wszystkich punktów startowych wygenerowanych z danego ziarna obliczano przystosowanie ziarna - im szybciej robot dochodził do punktu końcowego tym przystosowanie ziarna było mniejsze.

Działanie operatora "inkrementowanej mutacji" polegało na tym, że im ziarno było mniej przystosowane (robot dobrze sobie w nim radził) tym bardziej rozrzucone punkty startowe były generowane przez ten operator.

Przystosowaniem robota było jego średnie przystosowanie w ziarnach.



Schemat populacji występujących w eksperymencie (każdy robot związany ze zbiorem ziaren z których każde generuje przykładowe środowiska)

Cykl operacji do wyliczenia przystosowania robota:

- obliczyć przystosowanie robota w każdym przykładowym środowisku
- obliczyć przystosowanie każdego ziarna ze zbioru ziaren związanych z robotem(2,3,4,5,6)
- obliczyć przystosowanie robota wykorzystywane przy krzyżowaniu(7)

$$a^i = 1 - \frac{\sum_{j=1}^{N_{examples}} f_j^i}{N_{examples} F_{max}} \qquad (2)$$

gdzie F_{max} oznacza najgorszą wartość jaką może mieć przystosowanie w środowisku. Tak wiec a oznacza jak dobry jest robot w danym ziarnie im lepszy tym wyższe a.

$$\begin{cases} f_{max} = f_{min} \quad , \quad \beta_j^i = 0 \\ f_{max} \neq f_{min} \quad , \quad \beta_j^i = \dfrac{f_j^i - f_{min}}{f_{max} - f_{min}} \end{cases} \qquad (3)$$

Ten współczynnik mówi nam jak bardzo zróżnicowane rozwiązania znalazł robot, dla wartości bliskich najgorszym znalezionym wartość parametru będzie duża.

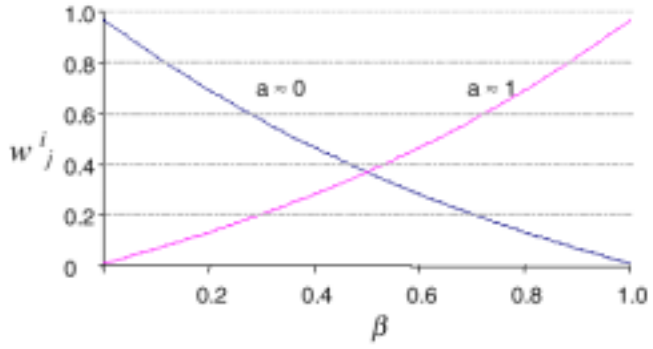
f_{max} oznacza najgorsze znalezione rozwiązanie

f_{min} oznacza najlepsze znalezione rozwiązanie

$$w_j^i = \frac{\left(e^{\alpha^i} - 1\right)\left(e^{\beta_j^i} - 1\right) + \left(e^{1-\alpha^i} - 1\right)\left(e^{1-\beta_j^i} - 1\right)}{(e-1)^2} \qquad (4)$$

Parametr ten zależy od parametrów a i β gdzie zarówno dobre zachowanie jak i złe zachowanie robota w danym przykładowym środowisku powoduje dużą wartość tego parametru. Parametr ten rozstrzyga jak duży udział będzie miało dane przykładowe środowisko w obliczaniu przystosowania całego ziarna.

Wykres zależności parametru w od a i β.



$$\alpha_j^i = \frac{w_j^i}{\sum_{k=1}^{N_{examples}} w_k^i} \qquad (5)$$

$$Bf^i = \sum_{j=1}^{N_{examples}} \alpha_j^i f_j^i \qquad (6)$$

$$f_{SOL} = \sum_{i=1}^{N_{blocks}} \frac{Bf^i}{N_{blocks}} - K\sigma_{fB} \qquad (7)$$

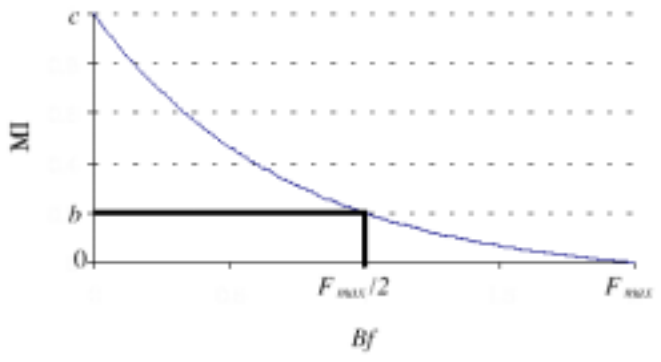
Przystosowanie ziarna Bf jest obliczane jako suma iloczynów przystosowań robotów w przykładowych środowiskach wygenerowanych z ziarna i wagi tego środowiska (udział parametru w przykładowego środowiska w sumie parametrów w wszystkich przykładowych środowisk danego ziarna)

Przystosowanie robota jest wyliczane jako średnia wartość przystosowań ziaren związanych z tym robotem (zmniejszona o odchylenie standardowe Bf)

Pozostaje jeszcze tylko określić jak jest obliczany operator "inkrementowanej mutacji". Przypomnijmy, że operator ten różnicuje przykłady generowane z ziaren. Wzór z którego jest on obliczany jest dość skomplikowany:

$$\mathrm{MI} = \frac{b^2\left(e^{\frac{2Bf}{F_{max}}\ln\left(\frac{c-b}{b}\right)} - 1\right)}{c-2b} \qquad (10)$$

Ale dużo prostszy do zrozumienia jest wykres go przedstawiający :



Rezultaty eksperymentu:

Przeprowadzono dwa doświadczenia w których wykorzystano odpowiednio:
- zwykły algorytm genetyczny
- przedstawiony algorytm koewolucyjny

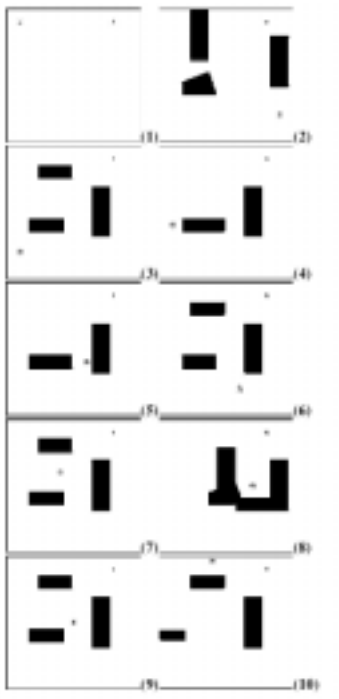
W pierwszym eksperymencie wszystko było stałe (pozycja startowa, skierowanie robota, układ przeszkód, punkt docelowy).
W drugim eksperymencie stałe były: układ przeszkód i pozycja docelowa natomiast zmieniała się pozycja startowa i skierowanie robota.

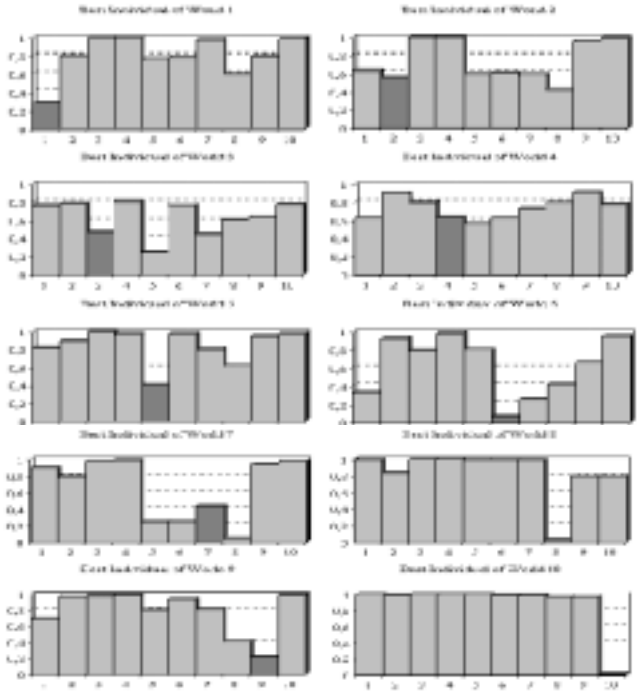
Do obliczania przystosowania robota w danym środowisku użyto wzoru:

$$f_j^i = 20T - 1.5L + 10C + 10S + 10D_m - 1.5D_o \quad (11)$$

Gdzie: T - liczba cykli potrzebna do osiągnięcia celu
L - długość drogi
C - liczba kolizji
S - liczba cykli w których robot nie zmienił pozycji
D_m - dystans pomiędzy pozycją początkową a końcową
D_o - dystans pomiędzy pozycją początkową a zadanym celem

W pierwszym eksperymencie roboty rozwijały się równolegle w 10 środowiskach, a następnie każdy został przetestowany w pozostałych środowiskach. Na wykresach poniżej przedstawiono zachowanie najlepszego robota ewoluującego w danym środowisku we wszystkich środowiskach oraz na rysunkach wyglądy poszczególnych środowisk.

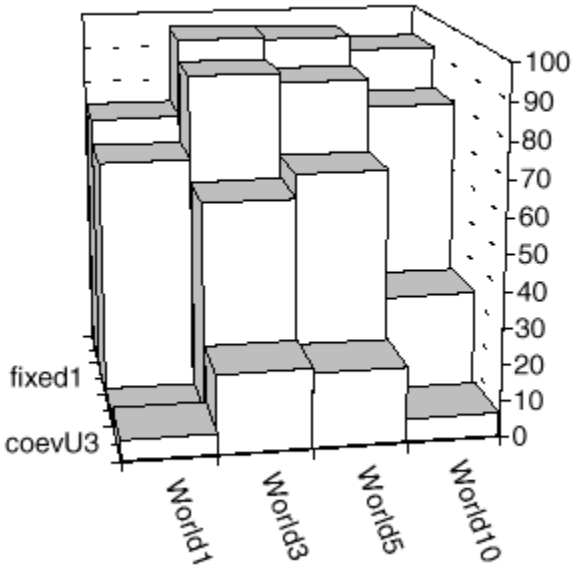




Jak widać zastosowanie zwykłego algorytmu genetycznego sprawiło, iż roboty dobrze wykształciły się w środowiskach w których się rozwijały (szybko dochodziły do celu), natomiast były nieprzystosowane do innych środowisk.

W eksperymencie drugim ograniczono się do środowisk 1 i 3 i testowania uzyskanych rozwiązań w środowiskach 1,3,5 i 10 (ponieważ zmienia się pozycja początkowa więc nie ma sensu ewolucja w środowiskach o tym samym układzie przeszkód). Eksperyment trwał 1000 pokoleń. Uzyskane wyniki (CoevU) porównano z uzyskanymi w wyniku działania zwykłego algorytmu (fixed) co obrazuje tabela i wykres:

	W_1	W_3	W_5	W_{10}
$Fixed_1$	67 ± 4	90 ± 6	87 ± 6	79 ± 6
$Fixed_3$	73 ± 5	96 ± 3	95 ± 3	91 ± 4
$CoevU_1$	2 ± 1	61 ± 10	67 ± 9	32 ± 11
$CoevU_3$	6 ± 2	22 ± 4	22 ± 7	6 ± 3



Jak widać zastosowanie algorytmu koewolucyjnego przyniosło dużo lepsze rozwiązanie a przy tym i ogólniejsze dające się zastosować w różnych środowiskach. Szczególnie jest to widoczne u robota wykształconego w środowisku trzecim.

4 Literatura

Wstęp
www.cs.gmu.edu/~mpotter

Niszowanie

[1] Samir W. Mahfoud (1992) „Crowding and Preselection Revisited”
[2] Samir W. Mahfoud „A Comparison of Parallel and Sequential Niching Methods”
[3] A. Petrowski „A Clearing Procedure as a Niching Method for Genetic Algorithms”
[4] D.E.Goldberg “Algorytmy genetyczne i ich zastosowania”
[5] Samir W. Mahfoud (1995) “Niching Methods for Genetic Algorithms”
[6] Jeffrey Horn “The Nature of Niching: Genetic Algorithms and The Evolution of Optimal, Cooperative Population”

Koewolucja

[1] J.Arabas "Wykłady z algorytmów ewolucyjnych"
[2] www.generation5 "An Introduction to coevolution"
[3] A.Berlanga ,A.Sanchis,P.Isasi,J.M.Molina "A General Learning Co-Evolution Method to Generalize Autonomous Robot Navigation Bahavior"