Algorytmy Ewolucyjne – sprawozdanie

1. Wyniki działania EA

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| instancja | Opt. | Alg. losowy [10k] | | | | Alg. zachłanny [N] | | | | Alg. ewolucyjny [10x] | | | |
|  |  | Best | Worst | Avg | Std | Best | Worst | Avg | Std | Best | Worst | Avg | Std |
| berlin52 | 7 542 | 22 919 | 34 673 | 29 886 | 1 575 | 8 182 | 10 299 | 9 373 | 471 | 7 544 | 8 403 | 8 101 | 249 |
| kroA100 | 21 282 | 135 121 | 199 624 | 171 092 | 8 201 | 24 698 | 28 695 | 27 045 | 812 | 21 921 | 23 931 | 22 869 | 638 |
| kroA150 | 26 524 | 221 533 | 292 867 | 257 645 | 10 131 | 31 482 | 35 935 | 33 640 | 862 | 28 442 | 29 704 | 29 142 | 381 |
| kroA200 | 29 368 | 293 275 | 384 126 | 340 217 | 11 808 | 34 548 | 42 045 | 37 469 | 1 405 | 31 204 | 33 363 | 32 261 | 708 |
| fl417 | 11 861 | 441 242 | 548 808 | 496 005 | 13 752 | 14 759 | 17 097 | 15 789 | 417 | 12 653 | 14 255 | 13 222 | 421 |

Opis parametrów

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Pop\_size | Gen | Krzyżowanie | Px | Mutacja | Pm | Selekcja | (Tour) |
| Berlin52 | 200 | 300 | OX | 0.7 | Inversion | 0.25 | Tournament | 4 |
| kroA100 | 400 | 500 | OX | 0.7 | Inversion | 0.4 | Tournament | 8 |
| kroA150 | 1000 | 650 | OX | 0.7 | Inversion | 0.4 | Tournament | 16 |
| kroA200 | 1000 | 750 | OX | 0.7 | Inversion | 0.4 | Tournament | 64 |
| Fl417 | 1250 | 1400 | OX | 0.7 | Inversion | 0.4 | Tournament | 128 |

Dodatkową rzeczą, która miała wpływ na działanie EA, było to, że do każdej nowej populacji był dodawany najlepszy z poprzedniej.

Przykładowe wykresy dla EA

* berlin52
* KroA100
* kroA150
* kroA200
* fl417

1. Badanie parametrów
   1. PopSize – Gdy parametr jest zbyt mały, populacja staje się bardzo ograniczona – jest bardzo słabo rozsiana po przestrzeni możliwych wyników, łatwo może wpaść w lokalne optima. Zbyt duża populacja powoduje jednak duży wzrost kosztów w postaci pamięci lub czasu wykonywania. Trzeba próbować znaleźć odpowiednią wartość, aby populacja była jak najmniejsza, ale wystarczająco bogata w różnorodne osobniki, aby kolejne generacje nie wpadły w wyżej wymienione optima lokalne.
   2. Gen – za mała wartość sprawia, że algorytm nie zdąży jeszcze dobrze przeskanować otoczenia, a już jest przerywany. Na wykresie poniżej widać, że gdyby wartość gen została w miejscu pierwszej pionowej linii to wynik wiele by stracił – w następnych generacjach nadal wystąpiłby duży spadek odległości. Zbyt duża wartość gen sprawia, że algorytm zbiega do optimum lokalnego i już z niego się nie wydostaje – generacje w okolicy drugiej zielonej linii i późniejsze są bezsensowne, bo nie wprowadzają żadnych lepszych osobników.
2. Badanie metod selekcji
   1. Turniej – wielkość tour bardzo wpływa na skuteczność EA – gdy jest zbyt mała selekcja staje się zbyt miękka – wraz z zmniejszaniem się tour zachowywane są coraz słabsze osobniki. Dla tour = 1 algorytm staje się losowy – nie ma żadnego wyławiania coraz lepszych osobników. Dla tour = 2 już widać, że z generacji na generację wyniki się polepszają, chociaż bardzo wolno – nadal duża ilość słabszych osobników zostaje w populacji. Dla dużych wartości tour selekcja staje się coraz twardsza – wybierani są prawie tylko najlepsi. Wyniki szybko zbiegają, ale tylko do optimum lokalnego, przestrzeń wokół nie jest dobrze przeszukiwana.
   2. Ruletka – skuteczność tej metody zależy w większości od wyboru algorytmu obliczenia fitness. Dla pierwszego sposobu jaki przychodzi do głowy – 1/score okazuje się, że daje zbyt bliskie wartości prawdopodobieństwa dla dobrych jak i słabych osobników – EA zachowuje się bardzo podobnie do losowego. Próbując jakoś uwypuklić lepszych osobników spróbowałem funkcji Fitness = Max – score + 10, która od razu dała lepsze wyniki, ale nadal po niewielkim czasie EA zaczął zachowywać się tak samo jak z poprzednim fitness. Po wyskalowaniu funkcji do fit = (max – score + 10) ^ e od razu widać, że EA działa o wiele lepiej.
3. Badanie operatorów
   1. Krzyżowanie (Ordered Crossover) – Parametr Px odpowiada za to jak często są krzyżowane osobniki, więc gdy jest za mały nie występuje wymiana genów między osobnikami, przez co nie ma generowanych nowych osobników – zmiana populacji występuje w większości przez mutację. Gdy wartość jest za wysoka osobniki zostają cały czas zmieniane – jest możliwość utraty dobrego osobnika. Porównując wykresy Px = 0.01 i Px 0.9 od razu widać różnicę – Px = 0.01 znacznie wolniej dąży do celu, jest bardziej płaski.
   2. Mutacja (Inversion) – Parametr Pm wpływa jak bardzo osobniki są mutowane – gdy jest mały to przestrzeń wyników jest bardzo słabo przeszukiwana, przez co łatwo wpada w optimum lokalne i nie odnajduje dalej znajdujących się rozwiązań. Gdy wartość jest za duża, mutacja ciągle zmienia osobniki, w tym te dobre, co sprawia, że EA zbiega do algorytmu losowego.