Kordowski Mateusz 293127
Przybysz Filip 293137

SK.ALHE.7

1. Treść Zadania

Masz 10 kart ponumerowanych od 1 do 10. Znajdź przy użyciu Algorytmu Ewolucyjnego sposób na podział kart na dwie kupki w taki sposób, że suma kart na pierwszej kupce jest jak najbliższa wartości A, a suma kart na drugiej kupce jest jak najbliższa wartości B. Należy zastosować dodatkowo inny wybrany algorytm i porównać wyniki.

Dobroć rozwiązania zależy liniowo od sumy wartości bezwzględnych różnic pomiędzy osiągniętymi a zadanymi wartościami sum kart każdej z kupek. Rozwiązanie jest tym lepsze im ta suma jest mniejsza.

2. Wykorzystywane technologie

Język C++.
Scons - narzędzie do automatyzacji procesu budowy oprogramowania.
Doxygen – generator dokumentacji.

3. Metody rozwiązania

3.1 Metoda ewolucyjna

Wykorzystujemy algorytm **genetyczny**, czyli rodzaj algorytmu ewolucyjnego z dyskretną przestrzenią rozwiązań.
Osobniki reprezentujemy za pomocą ciągu 10 symboli, gdzie i-ty symbol oznacza przynależność i-tej karty do odpowiadającej mu kupki.
Minimalizowaną **funkcją celu** jest zdefiniowana wyżej dobroć rozwiązania.
Inicjalizacja polega na stworzeniu losowej populacji początkowej.
W każdym pokoleniu algorytm koła ruletki odpowiada za **selekcję** osobników zarówno do puli rozrodczej jak i do następnego pokolenia (spośród aktualnego pokolenia i nowo wygenerowanych dzieci). Nie

wyklucza się wielokrotnego wyboru tego samego osobnika.
Osobniki z puli rozrodczej poddawane są **krzyżowaniu** (jednopunktowemu, wielopunktowemu lub równomiernemu). Wybór osobników z puli rozrodczej jest losowaniem z rozkładem jednostajnym dyskretnym.
Nowe osobniki poddawane są **mutacji**, w której każdy z ich genów ma szansę na zanegowanie swojej wartości.
Warunkiem stopu jest znalezienie rozwiązania optymalnego lub utworzenie ustalonej maksymalnej liczby pokoleń.
Stosujemy dodatkowo **strategię elitarną** zapewniającą pojawienie się w kolejnym pokoleniu kopii najlepszego osobnika z poprzedniego. Sam osobnik nie jest wykluczany z krzyżowania ani mutacji.

3.2 Metoda zachłanna

Algorytm buduje rozwiązania cząstkowe przypisując pojedynczo karty do kupek. Przechodzi po kartach w kolejności ich malejących wartości. W każdym kroku przypisuje właśnie rozważaną kartę tej kupce, której różnica wartości zadanej i wartości już przypisanych jej kart jest większa, przy czym dopuszcza się ujemne wartości różnicy.

4. Opis Funkcjonalny

Aplikacja pozwala na znalezienie rozwiązania optymalnego problemu z treści zadania, przy pomocy algorytmu genetycznego lub zachłannego, z pomiarem czasu wykonania lub bez.
Dokładny opis elementów programu znaleźć można w załączonej dokumentacji w formacie html.

5. Opis interfejsu użytkownika

Aby zbudować program należy wykonać polecenie *scons*. Aby cofnąć proces należy skorzystać z polecenia *scons -c*.

Aby skorzystać z programu należy wywołać plik wykonywalny *cards* ze ścieżką do przygotowanego pliku tekstowego z komendami.

Format komend:
<tryb działania programu> -A <docelowa wartość dla stosu A> -B <docelowa wartość dla stosu B> -mrand <licznik prawdopodobieństwa mutacji> -mmax <mianownik prawdopodobieństwa mutacji> -cnumb <ilość osobników poddawanych krzyżowaniu w każdym pokoleniu> -popsize <wielkość populacji> -popsizetourn <wielkość populacji po selekcji> -crosspoints <ilość punktów przecięcia dla krzyżowania wielopunktowego> -crosstype <typ krzyżowania> -path <ścieżka dla pliku wynikowego>
Niezbędne jest podanie wartości flag *A* i *B.* Pozostałym, jeżeli nie zostaną podane, przypisane będą wartości domyślne.

tryb działania programu przyjmuje wartości:

- **testNormal –** test algorytmu genetycznego
- **testNormalSilent –** test algorytmu genetycznego bez dodatkowych napisów
- **testNormalHeuristic –** test algorytmu zachłannego
- **testNormalHeuristicSilent –** test algorytmu zachłannego bez dodatkowych napisów
- **testTime –** test algorytmu genetycznego z pomiarem czasu
- **testTimeSilent –** test algorytmu genetycznego z pomiarem czasu bez dodatkowych napisów
- **testTimeHeuristic –** test algorytmu zachłannego z pomiarem czasu
- **testTimeHeuristicSilent –** test algorytmu zachłannego z pomiarem czasu bez dodatkowych napisów

typ krzyżowania przyjmuje wartości:

- **singlePoint -** jednopunktowe
- **multiplePoint -** wielopunktowe
- **uniform -** równomierne

6. Metodyka i wyniki testów

Przygotowano 110 przykładowych kombinacji wartości docelowych A i B. 55 osiągalnych (z powtórzeniami) i 55 nieosiągalnych (po 5 na różnicę od wartości osiągalnej od 1 do 11).
Algorytm ewolucyjny zawsze znajduje rozwiązanie optymalne w mniej niż

200 iteracjach.
Algorytm zachłanny także nie ma problemów w znajdowaniu rozwiązań dla przygotowanych kombinacji.
Czas wykonania przetestowano dla różnych wielkości populacji (10, 20, 50), prawdopodobieństw mutacji (20‰, 40‰, 70‰) oraz typów krzyżowania (jednopunktowe, wielopunktowe, równomierne) i porównano z algorytmem zachłannym.
Czas działania obliczano z pomocą mechanizmów std::chrono.
Średni czas po 5-krotnym wykonaniu przygotowanych zestawów kombinacji przedstawia tabela:

		Krzyżowanie	Jednopunktowe			Wielopunktowe			Równomierne			Rozwiązanie zachłanne
		Populacja	10	20	50	10	20	50	10	20	50	
Zestaw danych	P. mutacji [‰]											
Osiągalne	20		116405	156717	376104	122909	126417	277165	107110	149291	276511	418
	40		107231	122297	243051	88514	145402	229075	86721	142615	302950	
	70		96985	153550	312030	93633	144039	197461	86719	116506	261946	
Nieosiągalne	20		2412061	2658875	3778939	2626693	2946882	3813166	2416771	2727960	3643281	416
	40		2579080	2824364	3619443	2559964	2918795	3790197	2425499	2804200	3740939	
	70		2478340	3041888	3866258	2619731	2936333	3764853	2465878	2732118	3694881	

Pomiarów dokonano na komputerze osobistym pod kontrolą systemu operacyjnego Windows 10 i procesorem Intel Core i7-3770k CPU @ 3.50 GHz.

Jak widać, dla tego prostego problemu, w którym rozwiązanie zachłanne jest niemal na pewno zawsze poprawne, algorytm genetyczny nie sprawdza się najlepiej. Jest znacznie bardziej skomplikowany i działa znacznie dłużej od prostszej alternatywy działającej w stałym czasie niezależnie od osiągalności rozwiązania.

Bez dodatkowego mechanizmu zatrzymania związanego z długotrwałym brakiem poprawy, dla nieosiągalnych wartości docelowych algorytm genetyczny zawsze wykonuje maksymalną dopuszczalną ilość pokoleń przez co czasy wykonania nie wykazują poprawy wraz ze wzrostem wielkości populacji czy prawdopodobieństwa mutacji.

Zwiększanie wielkości populacji nie jest efektywne. W przeciwieństwie do zwiększania prawdopodobieństwa mutacji, które właściwie nie wnosi dodatkowych narzutów a pomaga dywersyfikować osobniki populacji.

Najodpowiedniejszym rodzajem krzyżowania zdaje się być krzyżowanie równomierne.