Akademia Górniczo-Hutnicza

Wydział Fizyki i Informatyki Stosowanej

Systemy równoległe i rozproszone

Monte Carlo - Przydział pokoi metodą simulated annealing

Autorzy:
Maciej Kubicki
Tomasz Chronowski

Prowadzący:
dr inż. Antoni Dydejczyk

13 maja 2017

Spis treści

1 Wstęp

1.1 Temat projektu

Temat projektu to przydział pokoi metodą simulated annealing. Zadanie polega na przydzieleniu parzystej liczby n studentów do $\frac{n}{2}$ pokoi w taki sposób, aby ich niezadowolenie było najniższe. W tym celu posługujemy się tabelą "nielubienia".

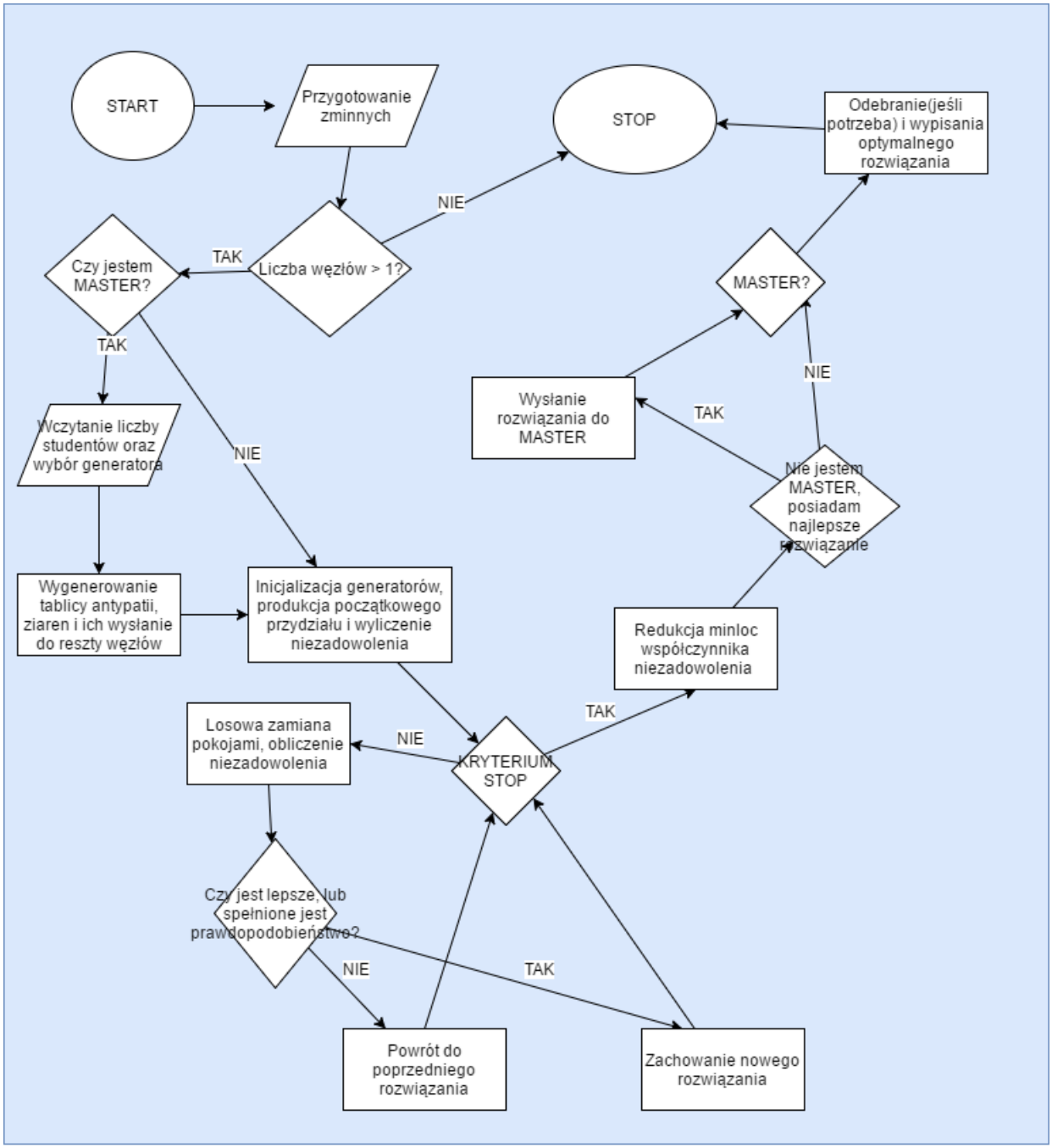
1.2 Simulated annealing

Po polsku - symulowane wyżarzenie - algorytm heurystyczny, którego celem jest znalezienie najlepszego rozwiązanie problemu spośród alternatywnych rozwiązań. Kroki algorytmu:

1. Losowe rozwiązanie i ustawienie temperatury $T = T_{max}$ - w przypadku przydziału do pokoi - losowe rozmieszczenie studentów w pokojach,
2. Obliczenie funkcji kosztu $F(w)$ - w przypadku przydziału do pokoi - współczynnik niezadowolenie studentów,
3. Wyznaczenie rozwiązania $w' = w + \Delta w$, Δw - realizacja zmiennej losowej o rozkładzie normalnym - w przypadku przydziału do pokoi - zamiana pokojami dwóch losowych studentów,
4. Obliczenie $F(w')$ i w przypadku gdy $F(w')$ jest lepsze od $F(w)$ lub $u <= e^{\left(\frac{F(w) - F(w')}{T}\right)}$ to do w przypisujemy w' - w przypadku przydziału do pokoi - gdy współczynnik niezadowolenia zmaleje zachowujemy zmianę pokoi, w innym wypadku wracamy do poprzedniego rozmieszczenia,
5. Zmniejszenie wartości temperatury $T = T * n$, gdzie $n \in (0, 1)$,
6. powtarzamy kroki 2-6, aż osiągniemy warunek stopu. Kończymy obliczenia, jeśli nie zmieni się rozwiązanie przez 1000 iteracji lub przez 1000 iteracji nie zostanie spełniony warunek $u <= e^{\left(\frac{F(w) - F(w')}{T}\right)}$.

2 Rozwiązanie

Podstawowym założeniem projektu było napisanie równoległego algorytmu. W związku z tym, że metoda simulated annealing nie zawsze zwraca optymalne wyniki i jej działanie opiera się w dużej mierze na generatorze losowym, tak więc najrozsądniejszą metodą zrównoleglenia aplikacji będzie praca na kilku węzłach posiadających różne ziarna, a następnie wybranie najbardziej optymalnego rozwiązania. W programie równoległym użyliśmy biblioteki z generatorami przystosowanymi do pracy w równoległych aplikacjach - SPRNG(*ang.* The Scalable Parallel Random Number Generators Library). Schemat blokowy algorytmu:



Rysunek 1: Schemat blokowy algorytmu równoległego

Przygotowany przez nas kod zakłada, że liczba studentów powinna być parzysta i podaję ją użytkownik. W przypadku, gdy użytkownik poda wartość nieparzystą, to brana będzie pod uwagę wartość o jeden większa. Tablica antypatii jest generowana każdorazowo dla wpisanej liczby studentów na węźle MASTER. Następnie jest rozesłana do wszystkich węzłów. Każdy węzeł posiada generator z innym ziarnem i wyznacza rozwiązanie problemu wykorzystując metodę simulated annealing. Następnie wybierane jest najbardziej optymalne i wypisane w węźle MASTER. Wygenerowana tablica antypatii znajduję się w pliku **antipation.dat**, a przydział do pokoi w pliku **result.dat**.

Oto przykładowa tablica antypatii dla 10 studentów oraz wynik algorytmu - najbardziej optymalny przydział do pokoi:

```
0 9.31603 8.40598 2.37182 4.84797 6.68578 3.79952 8.75237 8.1588 4.96943
9.31603 0 1.54768 1.84665 8.1972 2.63701 4.86133 4.90291 2.10593 5.18719
8.40598 1.54768 0 9.19981 1.65579 4.40644 7.44535 6.82542 0.495018 4.4941
2.37182 1.84665 9.19981 0 5.46968 7.45958 5.56683 7.51093 0.692054 4.54776
4.84797 8.1972 1.65579 5.46968 0 5.34855 8.40859 7.24183 9.40744 3.35554
6.68578 2.63701 4.40644 7.45958 5.34855 0 2.97114 4.86717 8.93797 5.16437
3.79952 4.86133 7.44535 5.56683 8.40859 2.97114 0 7.41753 1.54068 0.198034
8.75237 4.90291 6.82542 7.51093 7.24183 4.86717 7.41753 0 7.69279 6.1119
8.1588 2.10593 0.495018 0.692054 9.40744 8.93797 1.54068 7.69279 0 0.180783
4.96943 5.18719 4.4941 4.54776 3.35554 5.16437 0.198034 6.1119 0.180783 0

```

Rysunek 2: Przykładowa tablica "nielubienia" dla 10 studentów

```
Student Room
0       4
1       1
2       3
3       0
4       1
5       2
6       0
7       3
8       4
9       2

```

Rysunek 3: Otrzymany przydział do pokoi na podstawie powyższej tabeli i algorytmu simulated annealing

3 Wyniki i uwagi

W ramach projektu stworzyliśmy program działający równolegle i drugi sekwencyjny. Postanowiliśmy porównać ich działanie. Oba programy testujemy dla 200 studentów i czterokrotnie wykonując algorytm w celu znalezienia najbardziej optymalnego rozwiązania. Szybkość wykonania programów była badana na procesorze posiadającym cztery wątki oraz na Taurusie(połączenie SSH - nie z poziomu pracowni). W pierwszym przypadku w programie równoległym był użyty generator rand(), a w drugim jeden z generatorów w bibliotece SPRNG.

Tabela 1: Tabela wyników na moim komputerze

	Program równoległy z rand()	Program sekwencyjny
Czas działania[s]	6.42	18.67

Tabela 2: Tabela wyników na Taurusie

	Program równoległy z SPRNG(lcg)	Program sekwencyjny
Czas działania[s]	3.35	16.02

Widzimy, że program działa wyraźne przyśpieszenia działania programu równoległego.

3.1 makefile

Wraz z projektem załączony jest plik makefile.

- make - kompilacja rozwiązania równoległego,
- make run - uruchomienie rozwiązanie równoległego - w katalogu musi znajdować się plik nodes z dostępnymi węzłami,
- make seq - kompilacja rozwiązania sekwencyjnego,
- make runseq - uruchomienie rozwiązania sekwencyjnego,
- make mpe - kompilacja rozwiązania z MPE,
- make runmpe - uruchomienie rozwiązanie z MPE,
- make node - przygotowanie węzłów do obliczeń(skrypt station_name_list.sh musi mieć odpowiednie uprawnienia),
- make conv - konwersja logu MPE - clog2 do slog2,
- make jump - uruchomienie logu slog2 programem jumpshot,
- make clean - sprzątanie katalogu.

W razie potrzeby uruchomienia programu równoległego na większej liczby węzłów, niż 4 w pliku makefile przy komendzie run(runmpe) należy zmienić liczbę po "-np".