Parallel Simulated Annealing for the Set-partitioning Problem

KONSPEKT REFERATU

Na podstawie pracy autorstwa Zbigniewa J. Czecha

1 Opis problemu

DP – Delivery Problem (*Problem Dostaw*).

- centralny magazyn z pewnymi zasobami, oraz n klientów, rozmieszczonych w różnych odległościach od tego magazynu,
- zasoby muszą być dostarczone do każdego klienta przy użyciu samochodów dostawczych,
- mamy zbiór tras pozwalających dojechać do każdego odbiorcy,
- liczba samochodów jest dowolna, jednakże podczas jednego wyjazdu z bazy jeden samochód może odwiedzić co najwyżej k klientów (a k jest zazwyczaj małą stałą),
- rozważamy problem, w którym $k = 3$,
- Celem jest znalezienie rozwiązania, które minimalizuje całkowitą długość przebytej trasy podczas dostarczania towarów,
- DP redukuje się do NP-zupełnego „Problemu Podziału na Podzbiory" [ang. *SPP – Set Partitioning Problem*],

Omówimy sekwencyjną i równoległą wersję algorytmu rozwiązującego problem DP. Algorytm ten bazuje na technice tzw. symulowanego wyżarzania. Jego zrównoleglenie ma na celu poprawę jakości znajdywanych rozwiązań.

2 Przestrzeń rozwiązań

- w celu znalezienia rozwiązania instancji problemu DP musimy rozważyć zbiory wszystkich możliwych tras dotarcia do n klientów,
- dla rozważanego $k = 3$ mamy:

$$n! \cdot \sum_{i=0}^{\lfloor n/3 \rfloor} \left(\frac{1}{6^i i!} \cdot \sum_{j=0}^{\lfloor (n-3i)/2 \rfloor} \frac{1}{(n-3i-2j)!2^j j!} \right)$$

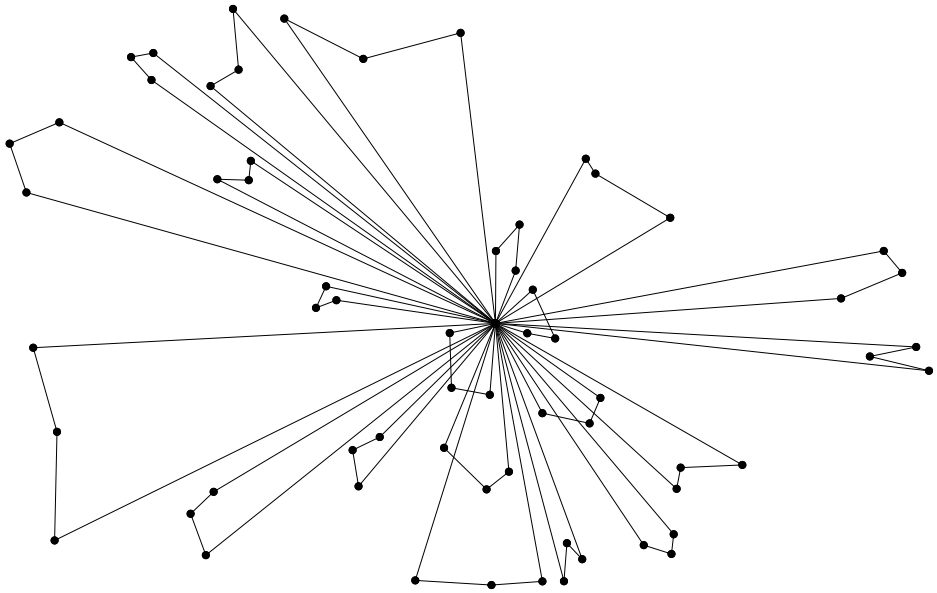
co oznacza, że dla $n = 40$ musimy odrzucić $\sim 8 \cdot 10^{21}$ nieoptymalnych rozwiązań, natomiast dla $n = 100$ aż $\sim 1.15 \cdot 10^{103}$ (czyli 1150 googooli!!),

3 Agorytm sekwencyjny

Definicja. Symulowane wyżarzanie [ang. *Simulated Annealing*] – rodzaj algorytmu heurystycznego przeszukującego przestrzeń alternatywnych rozwiązań problemu w celu wyszukania rozwiązań najlepszych. Jest wariantem metody przeszukiwania lokalnego [ang. *Local Search*].

Ogólny szkic działania algorytmu symulowanego wyżarzania:

1. losowy wybór punktu startowego,
2. losowy wybór sąsiada,
3. sąsiedzi o niższym koszcie są akceptowani bezwarunkowo, natomiast sąsiedzi o koszcie wyższym są akceptowani z prawdopodobieństwem zależnym od aktualnej temperatury,
4. po każdej iteracji temperatura zostaje zaktualizowana ($T = n \cdot T \ \wedge \ n \in (0, 1)$),
5. algorytm zatrzymuje się gdy w ciągu ustalonej liczby iteracji nie uda się osiągnąć lepszego wyniku,



Przykładowe optymalne rozwiązanie problemu DP z magazynem w centralnym punkcie.

```
1  create an initial old_solution as the set of random routes of size 3;
2  best_solution := old_solution;
3  equilibrium_counter := 0;                  # set the equilibrium counter
4  T := cost (best_solution) / 1000;          # initial temperature of annealing
5  repeat
6      for iteration_counter := 1 to n^2 do
7          annealing_step (old_solution, best_solution);
8      end for;
9      T := T · α;                            # temperature reduction
10     equilibrium_counter := equilibrium_counter + 1;
11 until equilibrium_counter > 20;
```

Sekwencyjny algorytm symulowanego wyżarzania dla DP.

Algorytm sekwencyjny postępuje zgodnie z ogólnym schematem wyżarzania.

- w liniach 6 – 8 wykonywane jest n^2 kroków wyżarzania, po których modyfikowana jest temperatura,
- krok wyżarzania (procedura `annealing_step`) polega na losowym wybraniu klienta oraz trasy i stworzeniu nowej trasy. Nowa trasa powstaje z dodania wybranego klienta do istniejącej, lub zamienieniu miejscami wybranego klienta i jednego z klientów wybranej trasy,
- tak powstałe rozwiązanie jest oceniane ze względu na zmianę długości całkowitej trasy w porównaniu z wcześniejszym rozwiązaniem i (uwzględniając aktualną temperaturę) jest akceptowane lub odrzucane.

```
1  procedure annealing_step (old_solution, best_solution);
2      select randomly a customer;
3      select randomly a route (distinct from the customer's route selected above)
           from the set containing the routes of the old_solution and an empty
           route;
4      if the route size is less than 3 then
5          create the new_solution by moving the customer into the chosen route;
6      else
7          select randomly a customer in the route;
8          create the new_solution by exchanging the selected customers between
               their routes;
9      end if;
10     δ := cost (new_solution) − cost (old_solution);
11     generate random x uniformly in the range (0, 1);
12     if (δ < 0) or (x < T/(T + δ)) then
13         old_solution := new_solution;
14         if cost (new_solution) < cost (best_solution) then
15             best_solution := new_solution;
16             equilibrium_counter := 0;
17         end if;
18     end if;
19 end annealing_step;
```

Procedura implementująca jeden krok wyżarzania.

4 Algorytm równoległy

Załóżmy, że mamy do dyspozycji p procesorów. Możemy więc każdemu z nich przydzielić zadanie poszukiwania optymalnego rozwiązania. Można tu zastosować dwie strategie:

1. każdy procesor szuka optymalnego rozwiązania niezależnie od pozostałych procesorów (dopiero po odnalezieniu tego rozwiązania – w ustalonym przedziale temperaturowym – uzgadniane jest które rozwiązanie jest najlepsze),
2. procesory wymieniają się pewnymi informacjami w trakcie poszukiwań optymalnego rozwiązania.

4.1 Równoległe, niezależne poszukiwanie.

Polega na uruchomieniu p procesów wyszukiwania i wybraniu najlepszego spośród zwróconych wyników. Pozwala polepszyć jakość rozwiązania, gdyż proporcjonalnie do zwiększania ilości procesów wzrasta szansa znalezienia optymalnego rozwiązania.

```
1  parfor P_j, j = 1,2,...,p do
2      if j = 1 then
3          create an initial best_solution as the set of random routes of size 3;
4          equilibrium_counter := 0;            # set the equilibrium counter
5          T := cost (best_solution) / 1000;    # initial temperature of annealing
6      end if;
7      old_solution_j := best_solution;
8      best_local_solution_j := best_solution;
9      repeat
10         for iteration_counter_j := 1 to n^2 do
11             annealing_step (old_solution_j, best_local_solution_j);
12         end for;
13         lock (best_solution); lock (equilibrium_counter);
14         if cost (best_local_solution_j) < cost (best_solution) then
15             best_solution := best_local_solution_j;
16             equilibrium_counter := 0;
17         end if;
18         unlock (equilibrium_counter); unlock (best_solution);
19         if j = 1 then
20             T := T · α;                      # temperature reduction
21             equilibrium_counter := equilibrium_counter + 1;
22         end if;
23     until equilibrium_counter > 20;
24 end parfor;
```

Algorytm równoległego, niezależnego poszukiwania dla maszyny CREW PRAM.

- procesy wykonują niezależne poszukiwania rozpoczynając z tego samego punktu startowego, oraz z takim samym schematem zmian temperatury (tak jak w algorytmie sekwencyjnym),
- dla danej temperatury, proces wykonuje niezależnie od innych n^2 kroków wyżarzania a następnie rozpatruje czy zastąpić globalne najlepsze rozwiązanie,
- temperatura jest obniżona dla wszystkich procesów i ponownie rozpoczynają się niezależne poszukiwania,
- gdy ilość niepowodzeń w znalezieniu lepszego rozwiązania będzie większa niż ustalona stała proces kończy działanie,
- gdy wszystkie procesy zakończą działanie wynik znajduje się w globalnej zmiennej.

4.2 Równoległe, okresowo współdziałające poszukiwanie.

W tym przypadku, procesy szukające rozwiązań współpracują ze sobą, dzieląc się ze sobą znalezionym lokalnie najlepszym rozwiązaniem co s kroków symulowanego wyżarzania.

```
1  parfor P_j, j = 1, 2, ..., p do
2      if j = 1 then
3          create an initial best_solution as the set of random routes of size 3;
4          equilibrium_counter := 0;                # set the equilibrium counter
5          T := cost (best_solution) / 1000;        # initial temperature of annealing
6      end if;
7      old_solution_j := best_solution;
8      best_local_solution_j := best_solution;
9      repeat
10         for iteration_counter_j := 1 to n^2 do
11             annealing_step (old_solution_j, best_local_solution_j);
12             propagate results in chain topology;
13         end for;
14         lock (best_solution); lock (equilibrium_counter);
15         if cost (best_local_solution_j) < cost (best_solution) then
16             best_solution := best_local_solution_j;
17             equilibrium_counter := 0;
18         end if;
19         unlock (equilibrium_counter); unlock (best_solution);
20         if j = 1 then
21             T := T · α;                          # temperature reduction
22             equilibrium_counter := equilibrium_counter + 1;
23         end if;
24     until equilibrium_counter > 20;
25 end parfor;
```

Algorytm równoległego, okresowo współdziałającego poszukiwania dla maszyny CREW PRAM.

- procesy rozpoczynają z różnych punktów startowych,
- procesy po s krokach dzielą się swoim – chwilowo najlepszym – rozwiązaniem:
  - proces o numerze 1 przekazuje swoje rozwiązanie procesowi o numerze 2,
  - proces numer 2 wybiera między swoim i otrzymanym rozwiązaniem. Lepsze z nich przekazuje do procesu 3, itd.
  - n-ty proces wybiera między swoim a najlepszym rozwiązaniem z rozwiązań wszystkich procesów o niższych numerach,
- po zakończeniu kroków symulowanego wyżarzania procesy zgłaszają swoje najlepsze rozwiązania,
- temperatura jest obniżana globalnie dla wszystkich procesów i ponownie rozpoczynają się kroki symulowanego wyżarzania,
- gdy wszystkie procesy zakończą się wynik znajduje się w globalnej zmennej.

Spis treści