Parallel Simulated Annealing for the Set-partitioning Problem

Mateusz Styrczula Michał Kraszewski

Instytut Informatyki Uniwersytetu Wrocławskiego

4 grudnia 2008

Delivery Problem

DP – Delivery Problem (*Problem Dostaw*).

Delivery Problem

DP – Delivery Problem (*Problem Dostaw*).

- centralny magazyn z pewnymi zasobami, oraz n klientów, rozmieszczonych w różnych odległościach od tego magazynu,

Delivery Problem

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,

Delivery Problem

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,

Delivery Problem

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,

Delivery Problem

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,
- podczas jednego wyjazdu z bazy samochód może odwiedzić co najwyżej k klientów,

Delivery Problem

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,
- podczas jednego wyjazdu z bazy samochód może odwiedzić co najwyżej k klientów,
- rozważamy problem, w którym $k = 3$,

Delivery Problem

- Celem jest znalezienie rozwiązania, które minimalizuje całkowitą długość przebytej trasy podczas dostarczania towarów,

Delivery Problem

- 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*],

Delivery Problem

- w celu znalezienia rozwiązania instancji problemu DP musimy rozważyć zbiory wszystkich możliwych tras dotarcia do n klientów,

Delivery Problem

- 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 $\sim 1.15 \cdot 10^{103}$,

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*].

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:

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,

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,

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. odpowiednia akceptacja sąsiada,

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. odpowiednia akceptacja sąsiada,
4. po każdej iteracji temperatura zostaje zaktualizowana:

$$T = n \cdot T \ \wedge \ n \in (0, 1)$$

Definicja

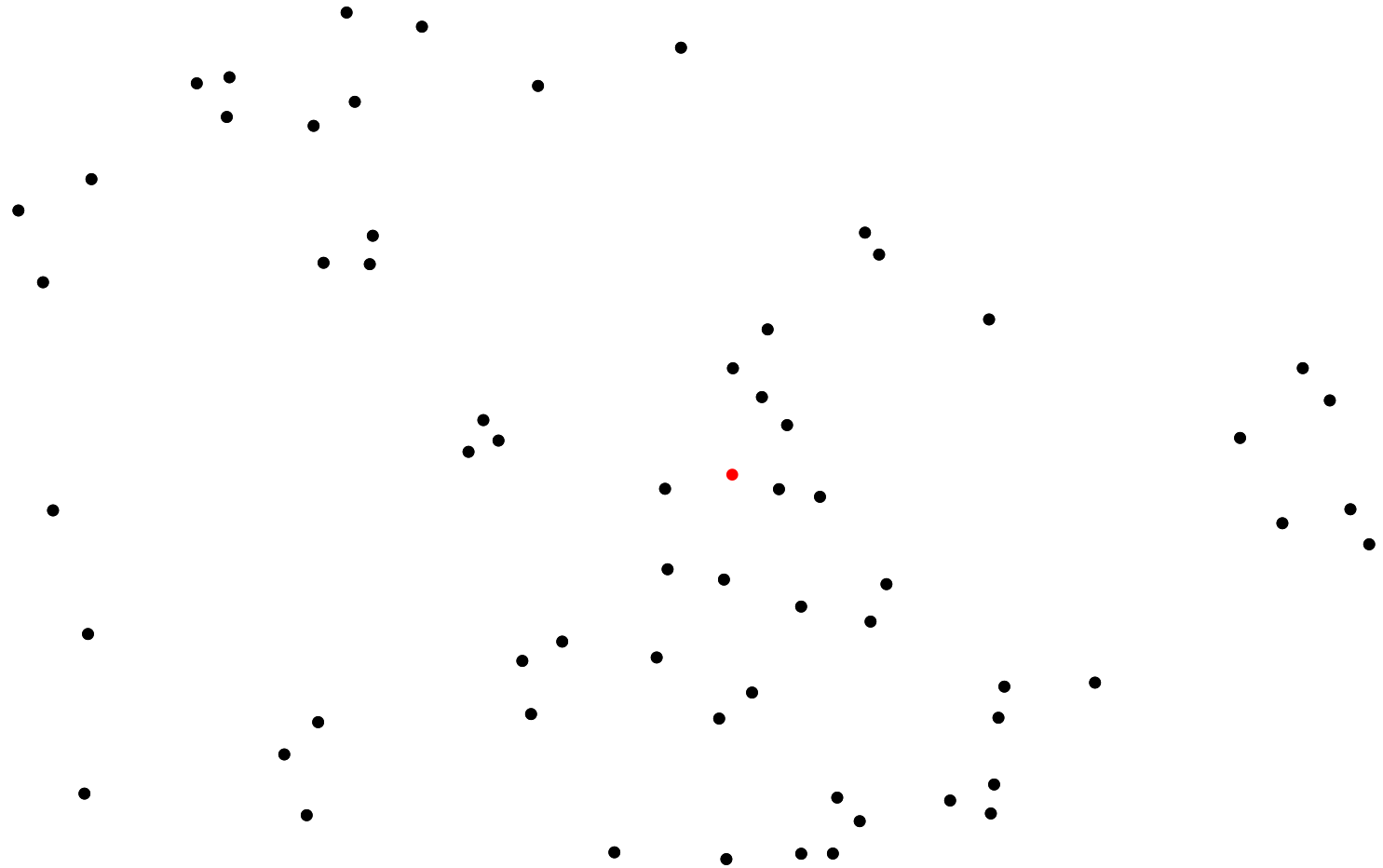
Symulowane wyżarzanie [ang. *Simulated Annealing*] – rodzaj algorytmu heurystycznego przeszukującego przestrzeń alternatywnych rozwiązań problemu w celu wyszukania rozwiązań najlepszych. Jest wariantem metody przeszukiwania lokalnego [ang. *Local Search*].

Ogólny szkic działania algorytmu symulowanego wyżarzania:

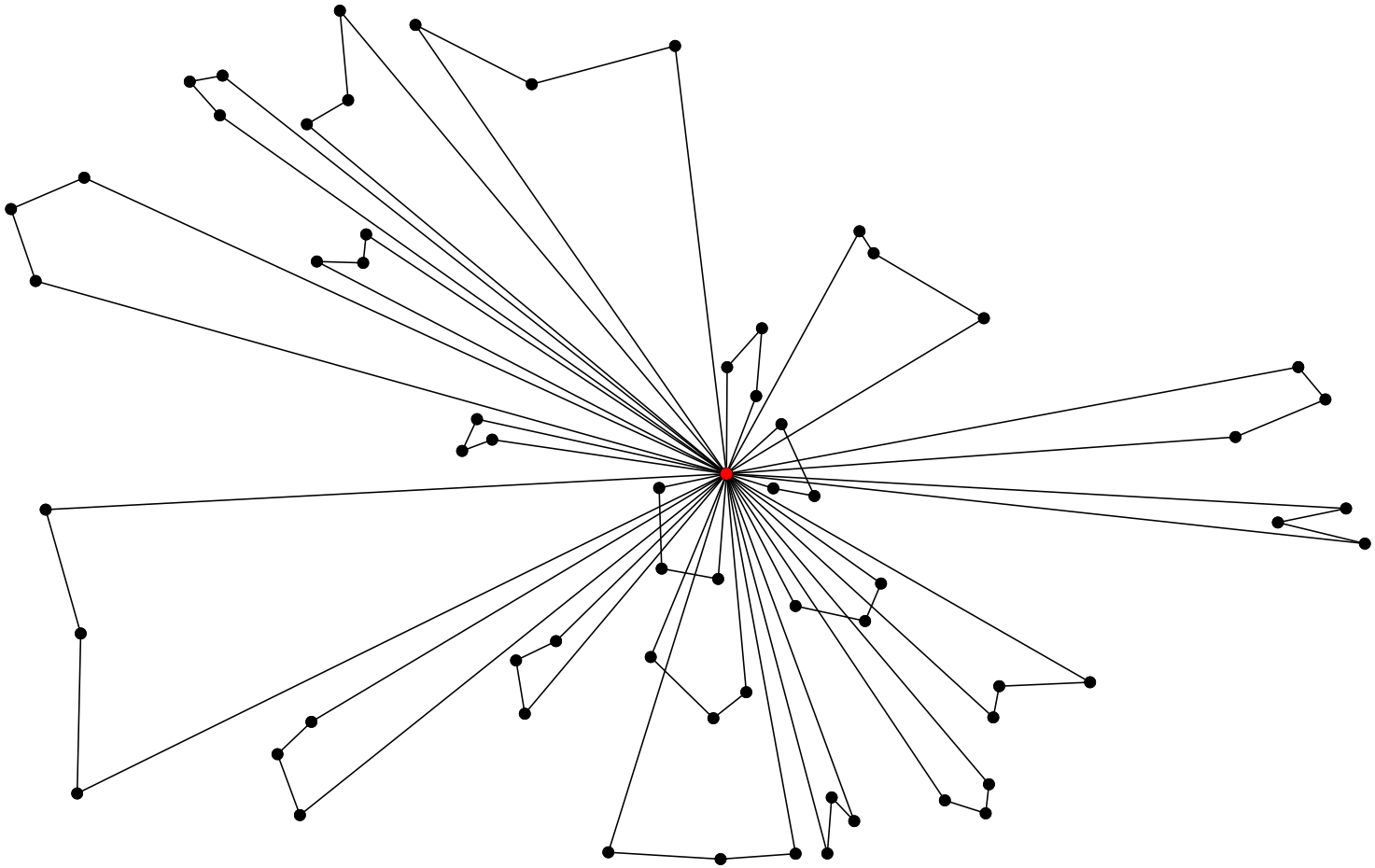
1. losowy wybór punktu startowego,
2. losowy wybór sąsiada,
3. odpowiednia akceptacja sąsiada,
4. po każdej iteracji temperatura zostaje zaktualizowana:

$$T = n \cdot T \ \wedge \ n \in (0, 1)$$

5. algorytm zatrzymuje się gdy w ciągu ustalonej liczby iteracji nie uda się osiągnąć lepszego wyniku,



Przykładowy DP z magazynem w centralnym punkcie.



Przykładowe optymalne rozwiązanie 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.

```
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.

Dwie strategie:

Dwie strategie:

1. każdy procesor szuka optymalnego rozwiązania niezależnie od pozostałych procesorów,

Dwie strategie:

1. każdy procesor szuka optymalnego rozwiązania niezależnie od pozostałych procesorów,
2. procesory wymieniają się pewnymi informacjami w trakcie poszukiwań 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;
```

```
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;
```

KONIEC