Complexity Analysis

Donglai Wei

2010.10.03

0.Notation

Restaurant-Level:

L-c:Local customer move, find the best table for a customer given others fixed
L-t:Local table move, find the best dish for a table given others fixed
M-t:Merge table move, merge two tables
DR:Decompose Restaurant
Dish-Level:

L-d: Local word move, reallocate the words distribution in a dish given others fixed
M-d:Merge dish move, merge two dishes
DD:Decompose Dish

$m_{.k}$ is the number of tables in dish k
$m_{j.}$ the number of tables in Restaurant j
$n_{jt.}$ the number of customers in Restaurant j table t
$n_{j..}$ the number of tables in Restaurant j
$n_{..k}$ the number of customers in dish k
$n^{w}_{..k}$ the number of tables in dish k with word w.

Method	Complexity
L-c	$\mathcal{O}(n_{j..}m_{j.})$
L-t	$\mathcal{O}(Km_{j.})$
M-t	$\mathcal{O}(m_{j.}^2)$
DR	$\mathcal{O}((m_{j.}+K^2)n_{j..}+(K+m_{j.})m_{j.})$
L-d	$\mathcal{O}(n_{..k}^w+m_{..})$
M-d	$\mathcal{O}(K^2)$
DD	$K*[\mathcal{O}(\text{L-d})+\mathcal{O}(\text{M-d})]+m_{..}*\mathcal{O}(DR)$

J restaurants, K dishes, L:log-probability(K-term+T-term)

1) Restaurant-level

1.1) local-merge move

Local-Customer: $\mathcal{O}(m_{j.}^2\frac{n_{j..}}{m_{j.}})=\mathcal{O}(n_{j..}m_{j.})$

Local-Table: $\mathcal{O}(m_{j.}K)$
Merge-Table: $\mathcal{O}(m_{j.}^2)$

```
While L doesn't increase any more:
    %a)Local-Customer:
      For t_1=Randperm(m_j.):
          For t_2=Randperm(m_j.\ t_1):
             While local move can be made:
                   Greedily move one customer at a time from t_1 to t_2 if the move increases L'(T)
             End
          End
      End

    %b)Local-Table:
      For t_1=Randperm(m_j.):
          Assign t_1 to the dish which increase L most(allow it to have new dish)
      End

    %c)Merge Table:
      For t_1=Randperm(m_j.):
          Merge table t_1 to the table in j with the best dish k, which increase L'(T) most
          %if cannot increase L'(T),then leave it alone
      End
End
```

1.2) Decompose restaurant

Initialization: $\mathcal{O}(n_{j..}K^2)$ (#proposed table$\propto$K)
Local-Merge Moves in (1.1): $\mathcal{O}(m_{j.}n_{j..})+\mathcal{O}(m_{j.}K)+\mathcal{O}(m_{j.}^2)$

In all, **Decompose-Restaurant**: $\mathcal{O}((m_{j.}+K^2)n_{j..})+(m_{j.}(K+(m_{j.}))$

```
%a) Initialization
Make Restaurant j into one table t_0 where customers following uniform distribution
%(P(t_ji = t_0) = 1/W)
Possible Dish={Nonempty dishes} \ k_0
While Possible Dish is not empty:
      For each dish k ∈Possible Dish:
          For each customer i in t_0:
              sample t_ji ∈ {t_0, t_k} ~ {1/W, (n_..k^w + φ)/(n_..k + Wφ)}
          End
      Propose to form table t_k with customers whose t_ji = t_k
      Calculate the change d_k for k-term and w-term:
      End
      %Sample a proposal t_k* according to the weight and make the new table:
      Sample a proposal {t_k1, ..., t_kK} ~ e^{r_proposal {d_k1, ..., d_kK}}
      %r_proposal > 0, the more decrease of d_k, the less propable to form table t_k
      Possible Dish=Possible Dish\ k_*
      t_0 = t_0 \ t_k*
End
If there are still customers left in t_0:
   make it a new table with a new dish K+1
End

%b)Refinement for Restaurant j(Local-Search and Merge Move for tables in Restaurant j)
LM-Restaurant(j,T)

%c)Decision
Calculate the change of L between present Restaurant j config and its previous config:
Δ L' = Δ k-term + TΔ t-term + Δ w-term
If(Δ L' <0):
    Accept the new configuration
Else:
    Restore Previous Config
End
```

2) Dish-level

2.1) Local-Dish

Find promising dishes: $\mathcal{O}(m_{..})$
Find the best approximated Reallocation of word w: $\mathcal{O}(n^{w}_{..k})$

In all: $\mathcal{O}(n^{w}_{..k} + m_{..})$

```
Dish List={Nonempty dishes} \ k
For w=Randperm(Words Occur in Dish k):

    %i) Find promising dishes(appear most often with the tables having word w in dish k)
    For kk=Dish List
        Restaurant_index(kk)=index of restaurants in dish kk, that have tables serving dish k
```

```
        Restaurant_count(kk)=length(Restaurant_index(kk))
    End
    Promising_Dishes=find(Restaurant_count==max(Restaurant_count))

    %ii) Find the best approximated Reallocation of word w:
    Δ L=0;
    For kkk=Promising_Dishes:

         %a)Naive Reallocation:
        For j=Restaurant_index(kkk)
            Assign all of words w from the table serving dish k to the table serving dish kkk
        End
        Calculate the change of L between present config and previous config:l_0

        %b)Gready Search:
        Δ l=1
        While Δ l >0
              Δ l_k=change of L by assigning one word w from dish kkk back to dish k
              Δ l_t=max change of L by assigning one word w from dish kkk back to dish k in Restaurant j
              Δ l = Δ l_k + Δ l_t
        End
        Δ L = max(Δ L, l_0 + Δ l)
    End

    %iii)Decision:
    If Δ L >0
        Accept new config
    Else
        Restore previous config
    End
End
```

2.2) Merge-Dish

Merge-Dish: $\mathcal{O}(K^2)$

```
Dish List={Nonempty  dishes} \ k

While Dish List is not empty:
    Randomly pick a dish k∈ Dish List
    Dish List=Dish List\k
    Merge dish k to the dish∈ Dish List which increase  L'(T) mostly
    %if cannot increase L'(T), then leave it alone
End
```

2.3) Decompose Dish

Initialization: $m_{.k}\mathcal{O}(DR)$

Local-Merge Move: $\mathcal{O}(n^w_{..k} + m_{..}) + \mathcal{O}(K^2)$

In all, **Decompose Dish:** $K * [\mathcal{O}(\text{L-d}) + \mathcal{O}(\text{M-d})] + m_{..} * \mathcal{O}(DR)$

```
For k=Randperm(K)

   %a)Initialization: Reconfig without Dish k
      For j=restaurants which have tables serving dish k
          Decompose Restaurant(j,Temperature,k);
      End

   %b)Merge new proposed dishes + Local − Dish
      For k=new proposed dishes
          Merge Dish(Temperature,k);
      End
      For w=words which appear in dish k
          Local-Dish(w,Temperature,k);
      End

   %d)Decision
      If(Δ k − term + Δ w − term + Temperature * Δ t − term <0):
          Accept new config
      End
End
```