# Finaally...Let there be bars

**Donglai Wei**

2010.5.28

## 0) Key Words

1. Light-version: of Split-Restaurant and Split-Dish for more runs
2. Full-version: of Split-Dish and Split-Dish for robustness
3. Split-Dish-All/Further: Bigger moves for Split-Dish
4. Mixing Process: Better mixing Iterations between Split-Restaurant and Split-Dish for faster "convergence"
5. Small Tricks: some little modifications for better split and merge moves.

## 1) Pseudocode

For Initialization:
**Table Config** is the key.
**Aim**: Since dish config is vague, it is wise to just split Restaurant into smaller tables for later reconstruction.

For Search:
**Dish Config** is the key.
**Aim**: Since table config is local,it is wise to focus on better Dish Config across restaurants.

(I) **SCHEME**

(a) Initialization:
1 table and 1 dish per restaurant
Repeat several times:
(1) Run Split-Restaurant(Full-version)[accept]
(2) Run Split-Dish(Light-version)[accept](random Dish k)

(b) Search:
Repeat several times:
(1) Run Split-Restaurant(Light-Version)[accept/reject](random Restaurant j)
(2) Run Split-Dish(Full-Version)[accept/reject](random Dish k)
(3) (Occasionally)Run Split-Dish-All[accept/reject]

(4) (Occasionally)Run Split-Dish-Further[accept/reject](random Dish k)

(II) **Split-Restaurant**

(a) Light-Version[Decision](j)

(1) In random order,Split all tables in j using 2-means++(producing $2m_{j\cdot}$ initial tables)

(2) Run TKM search(Local Search table/dish+Merge table)

(3) Decision for the new config

(b) Full-Version[Decision]

(1) For j=randperm(J)

(2) In random order,Split all tables in j using 2-means++(producing $2m_{j\cdot}$ initial tables)

(3) Run TKM search(Local Search table/dish)

(4) Decision for the new config

(5) End

(III) **Split-Dish**

(a) Light-Version[Decision](k)

(1) In random order,Split all tables in dish k using 2-means++(producing $2m_{\cdot k}$ initial tables)

(2) Split dish k using 2-means++

(3) Run TKM search(Local Search table/dish)

(4) Decision for the new config

(b) Full-Version[Decision](k)

(1) In random order,Split all tables in dish k using 2-means++(producing $2m_{\cdot k}$ initial tables)

(2) (Multiple Runs)Split dish k using 2-means++

(3) (Multiple Runs)Run TKM search(Local Search table/dish+merge dish)(?merge table)

(4) Decision for the new config

(c) All-Version[Decision]

(1) For k=randperm(K)

(2) if(k$leq$length(dishes))

(3) Light-Version Split-Dishes[accept](k)

(4) End

(5) End

(6) (Multiple Runs)Run TKM search(Local Search table/dish+merge dish)

(7) Decision for the new configuration

(d) Further-Version[Decision](k)

(1) Full-Version Split-Dishes[accept](k)
d1,d2 are the dishes that the splitted dishes are assigned to

(2) If(d1$\neq$d2)

(3) Further-Version Split-Dishes[accept](d1)

(4) Further-Version Split-Dishes[accept](d2)

(5) End

(6) (Multiple Runs)Run TKM search(Local Search table/dish+merge dish)

(7) Decision for the new configuration

Figure 1: 40 Restaurants

(IV) **TKM** While log probability does not increase any more:

(1) Local Search Tables(For chosen restaurants)

(2) Local Search Dishes(For chosen dishes)

(3) Merge Tables(For chosen restaurants)

(4) Merge Dishes(For chosen dishes)

(V) **Small Tricks**

(1) Merge Dish:

(i) Previously: Simply Merge two dishes without changing table config

(ii) Better:Given Dish config$\vec{k}_{jt}$(k-term fixed), it is better to merge tables(belong to dish k) in the same restaurant to further decrease -log Probability.

(2) Split table:

(i) Previously: 2-means++ allows new tables to have new/previous dishes

(ii) Better:During Search, in order to split "sticky" table/dish, it is better not to allow them assigned to new/previous dishes.

(3) Split dish:(similar to Split table)

# 2) Experiment

## i)Settings

40 Restaurants,10 bars

## ii) Initialization

Run 5 times:Split-Restaurant(Full-Version) + Split-Dish(Light-Version)

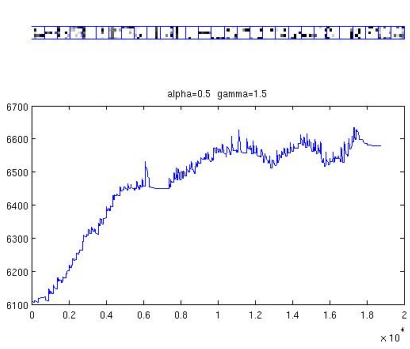


Figure 2: 5 runs:Dish Config and -log Probability

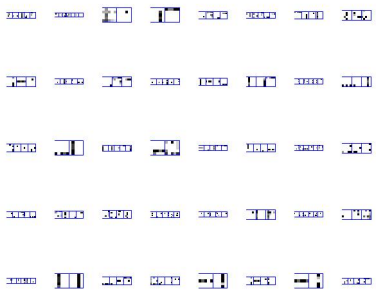

Figure 3: 5 runs:Table Config for 40 restaurants

## iii) Search

**Outline:**

1. **Scheme 1:Split-Restaurant:Split-Dish=1:1**
2. **Scheme 2:Split-Restaurant:Split-Dish=5:1**
3. **Bigger Moves:Split-Dish(All-Version)+(Further-Version)**

1) For j=randperm(J)

1. Split-Restaurant(Light-Version)
2. Split-Dish(Full-Version)

End

Two variants:
a)Split-Dish(Full-Version) with merge tables in TKM
b)Split-Dish(Full-Version) with merge tables in TKM

**Conclusion:**
I tried several runs and found a) in the beginning tends to create single table for the restaurant since the dish config is still vague(t-term dominates).
Below are the typical examples.

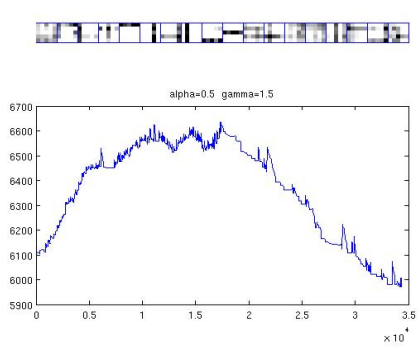


Figure 4: a) with merge table:Dish Config and -log Probability

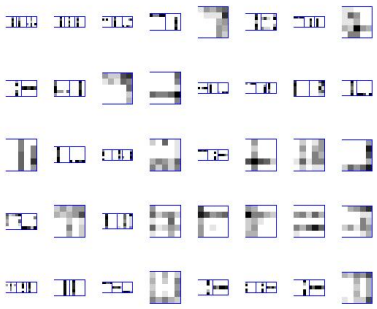

Figure 5: a) with merge table:Table Config for 40 restaurants

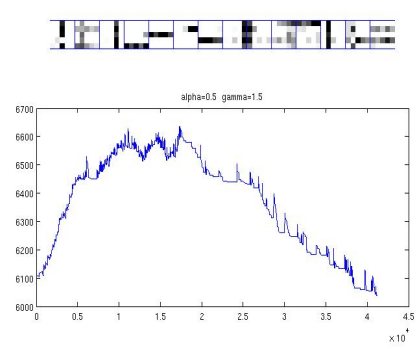


Figure 6: b) without merge table:Dish Config and -log Probability

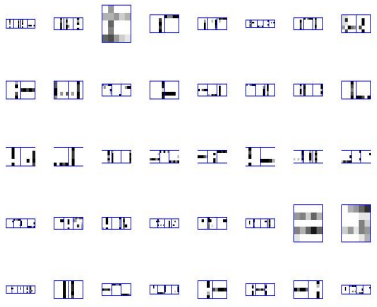

Figure 7: b) without merge table:Table Config for 40 restaurants

2)Since Split-Dish(Full-Version) is expensive and may not benefit much from one Split-Restaurant(Light-Version), we may try the mixing process below: (No merge table during TKM in Split-Dish(Full-Version))

For Iter=1:5
count=1;
For j=randperm(J)

1. Split-Restaurant(Light-Version)
2. If(mode(count,5)==0)
   Split-Dish(Full-Version)
   End
3. count=count+1;

End
End

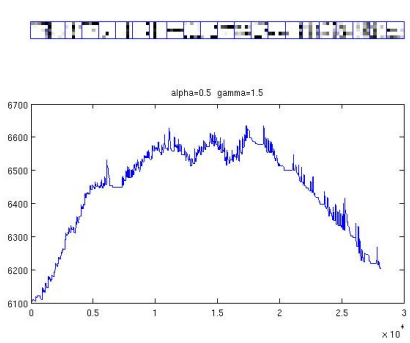


Figure 8: Iter 1:Dish Config and -log Probability

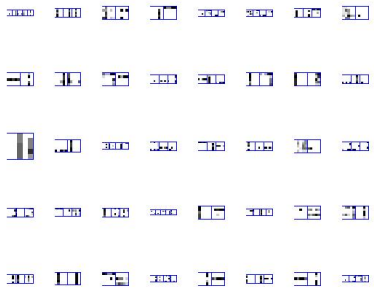

Figure 9: Iter 1:Table Config for 40 restaurants

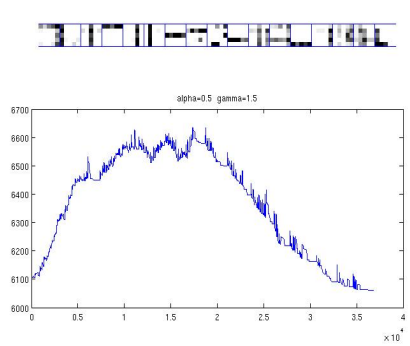


Figure 10: Iter 2:Dish Config and -log Probability

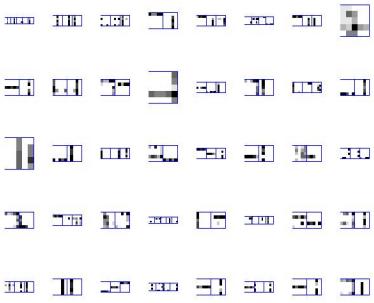

Figure 11: Iter 2:Table Config for 40 restaurants

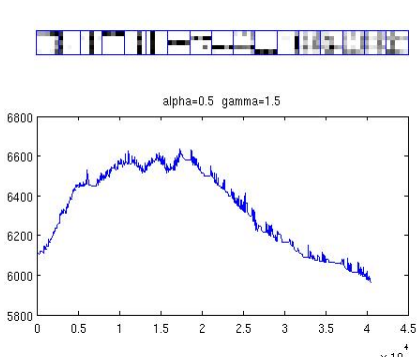


Figure 12: Iter 3:Dish Config and -log Probability

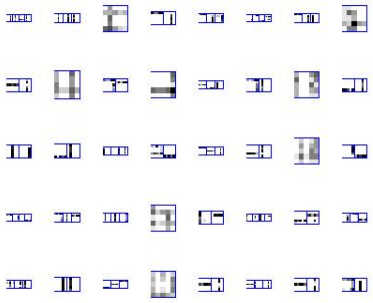

Figure 13: Iter 3:Table Config for 40 restaurants

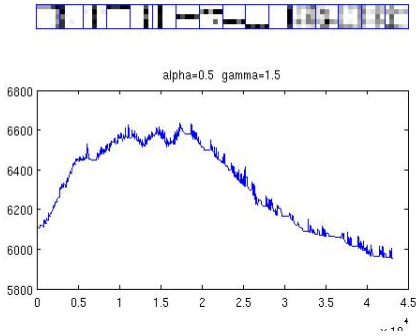


Figure 14: Iter 4:Dish Config and -log Probability

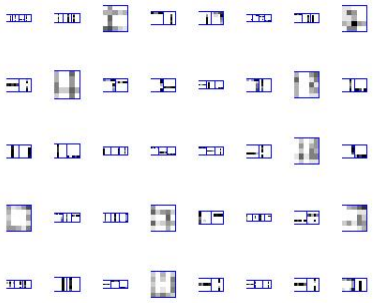

Figure 15: Iter 4:Table Config for 40 restaurants

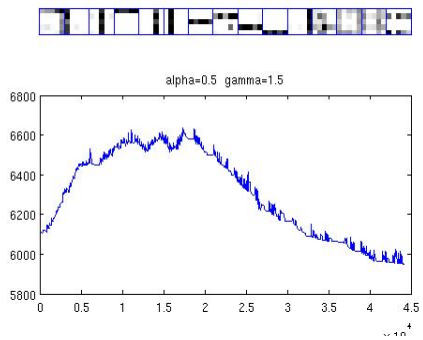


Figure 16: Iter 5:Dish Config and -log Probability

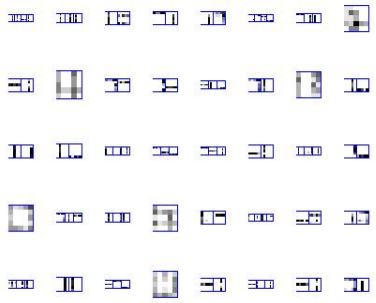

Figure 17: Iter 5:Table Config for 40 restaurants

3)From Above, though we roughly figure out 10 true bars, the noisy dishes refuse to be "explained" by them.
The reason is manyfold:

1. Some of the true bars are not "strong enough":though splitting one noisy dish decreases k-term, the t-term increases more.
2. The noisy dish is composed of more than two bars, thus split it into two components may not be better than leave it alone.

Possible Solutions:

1. Split-Dish-All: though split one noisy dish may be weak, we can wait and accumulate them. In the end, it may be profitable.
2. Split-Dish-Further:Split-Dish(Full version)(k)+Split-Dish(Further version)(d1,d2)
   (p.s. we expect true bars won't be bothered by it:true bars may not be well explained by two other dishes, thus 2-means++ will assign )

Thus, we have following Scheme:

While -log probability cannot decrease any more

count=1;
For j=randperm(J)

1. Split-Restaurant(Light-Version)[accept/reject](j)
2. If(mode(count,5)==0)
   Split-Dish(Full-Version)[accept/reject]
   End
3. count=count+1;

End
Split-Dish(All-Version)[accept/reject]
Split-Dish(Further-Version)[accept/reject](random k)
End

Figure 18: Iterations with Split-Dish(All-Version) only

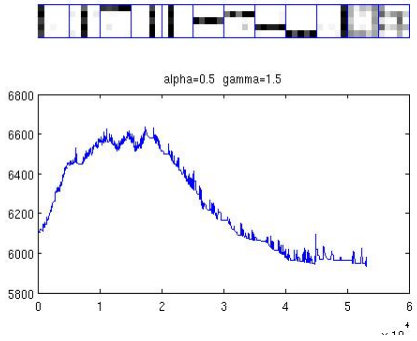


Figure 19: dish config for Fig 18:The last 2 dishes are problematic

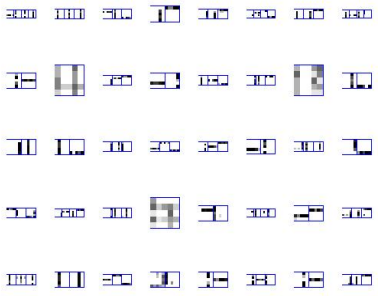

Figure 20: +Split-Dish(Further-Version):Get out of stuck while introducing new noise

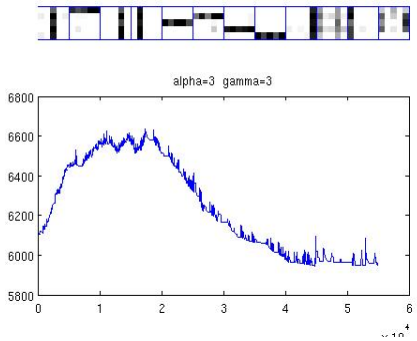


Figure 21: dish config for Fig 20:alleviate the problem by introducing new dishes

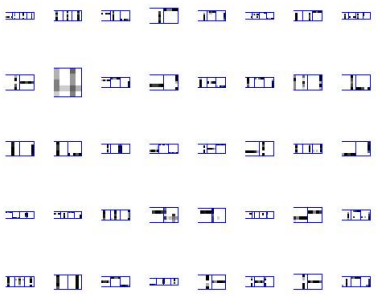

Figure 22: +Split-Dish(Further-Version): get rid of the last dish

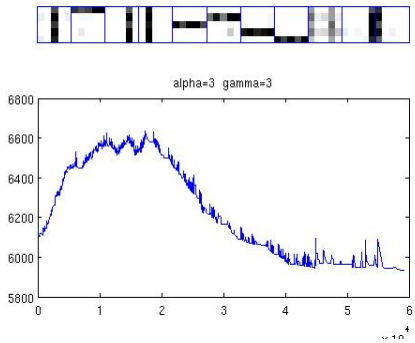


Figure 23: dish config for Fig 20:ALMOST THERE

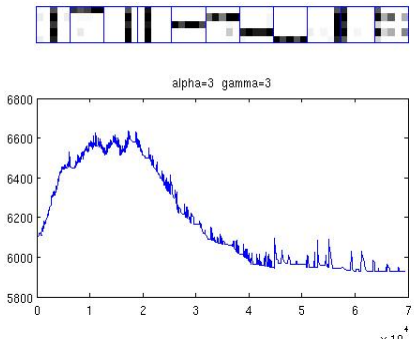


Figure 24: +Split-Dish(All-Version): One more try

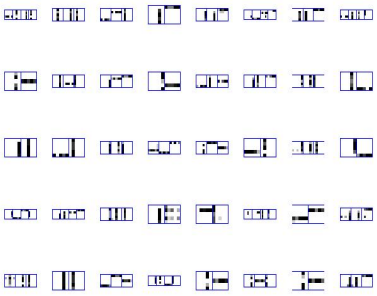

Figure 25: dish config for Fig 22:Closer

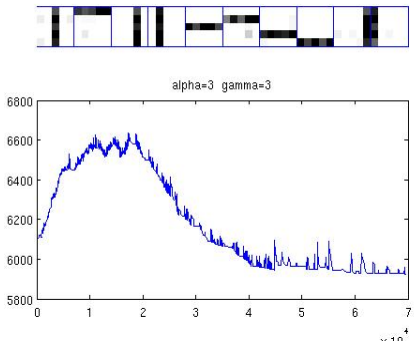


Figure 26: +Split-Restaurant

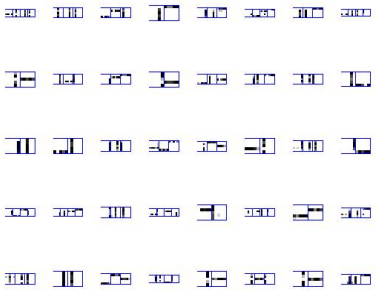

Figure 27: dish config for Fig 24:Finally...