

A* on Hypercube for Multiple Sequence Alignment

OpenMP Implementation

Jian Chen
11/21/2019

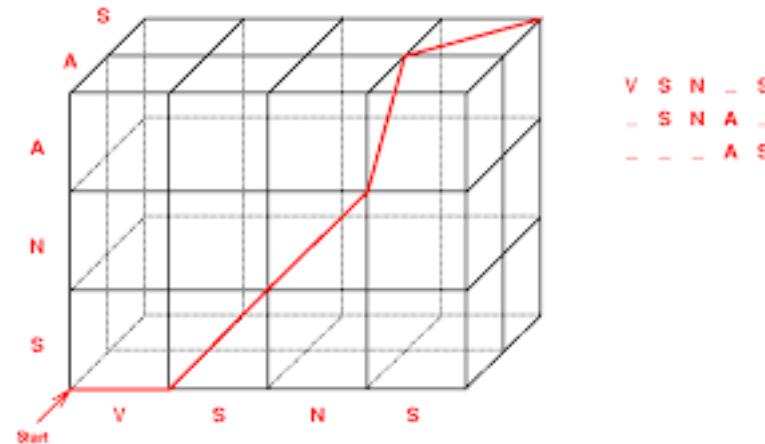
Protein Sequence Multiple Alignment

		*	.	:	*	:	:	:	ruler		
Q5E940_BOVIN	-----	M P R E D R A T W K S N Y F L K I I Q L L D D Y P K C F I V G A D N V G S K Q M Q Q I R M S L R G K - A V V L M G K N T M M R K A I R G H L E N N -- P A L E								76	
RLAO_HUMAN	-----	M P R E D R A T W K S N Y F L K I I Q L L D D Y P K C F I V G A D N V G S K Q M Q Q I R M S L R G K - A V V L M G K N T M M R K A I R G H L E N N -- P A L E								76	
RLAO_MOUSE	-----	M P R E D R A T W K S N Y F L K I I Q L L D D Y P K C F I V G A D N V G S K Q M Q Q I R M S L R G K - A V V L M G K N T M M R K A I R G H L E N N -- P A L E								76	
RLAO_RAT	-----	M P R E D R A T W K S N Y F L K I I Q L L D D Y P K C F I V G A D N V G S K Q M Q Q I R M S L R G K - A V V L M G K N T M M R K A I R G H L E N N -- P A L E								76	
RLAO_CHICK	-----	M P R E D R A T W K S N Y F M K I I Q L L D D Y P K C F V V G A D N V G S K Q M Q Q I R M S L R G K - A V V L M G K N T M M R K A I R G H L E N N -- P A L E								76	
RLAO_RANSY	-----	M P R E D R A T W K S N Y F L K I I Q L L D D Y P K C F I V G A D N V G S K Q M Q Q I R M S L R G K - A V V L M G K N T M M R K A I R G H L E N N -- S A L E								76	
Q7ZUG3_BRARE	-----	M P R E D R A T W K S N Y F L K I I Q L L D D Y P K C F I V G A D N V G S K Q M Q T I R L S L R G K - A V V L M G K N T M M R K A I R G H L E N N -- P A L E								76	
RLAO_ICTPU	-----	M P R E D R A T W K S N Y F L K I I Q L L D D Y P K C F I V G A D N V G S K Q M Q T I R L S L R G K - A I V L M G K N T M M R K A I R G H L E N N -- P A L E								76	
RLAO_DROME	-----	M V R E N K A A W K A Q Y F I K V V E L F D E F P K C F I V G A D N V G S K Q M Q N I R T S L R G L - A V V L M G K N T M M R K A I R G H L E N N -- P Q L E								76	
RLAO_DICDI	-----	M S G A G - S K R K K L F I E K A T K L F T T Y D K M I V A E A D F V G S S Q L Q K I R K S I R G I - G A V L M G K K T M I R K V I R D L A D S K -- P E L D								75	
Q54LP0_DICDI	-----	M S G A G - S K R K N Y F I E K A T K L F T T Y D K M I V A E A D F V G S S Q L Q K I R K S I R G I - G A V L M G K K T M I R K V I R D L A D S K -- P E L D								75	
RLAO_PLAF8	-----	M A K L S K Q Q K K Q M Y I E K L L S L I Q Q Y S K I L I V H V D N V G S N Q M A S V R K S L R G K - A T I L M G K N T R I R T A L K K N L Q A V -- P Q I E								76	
RLAO_SULAC	-----	M I G L A V T T T K K I A K W K V D E V A E L T E K L K T H K T I I I A N I E G F P A D K L H E I R K K L R G K - A D I K V T K N N L F N I A L K N A G -- Y D T K								79	
RLAO_SULTO	-----	M R I M A V I T Q E R K I A K W K I E E V K E L E O Q L R E Y H T I I I A N I E G F P A D K L H D I R K K L M R G M - A E I K V T K N T L F G I A A K N A G -- L D V S								80	
RLAO_SULSO	-----	M K R L A L A L K Q R K V A S W K L E E V K E L T E L I K N S T I L I G N L E G F P A D K L H E I R K K L R G K - A T I K V T K N T L F K I A A K N A G -- I D I E								80	
RLAO_AERPE	M S V V S L V G Q M Y K R E K P I P E W K T L M L R E L E E L F S K H R V V L F A D L T G T P I F V V O R V R K K L W K K - Y P M M V A K K R I I L R A M K A A G L -- L D D N									86	
RLAO_PYRAE	M M L A I G K R R Y V R T R Q Y P A R K V K I V S E A T E L L Q K Y P V F L F D L H G L S S R I L H E Y R Y R L R R Y - G V I K I I K P T L F K I A F T K V Y G G -- I P A E									85	
RLAO_METAC	-----	M A E E R H H T E H I P Q W K K D E I E N I K E L I Q S H K V F G M V G I E G I L A T K M Q I R R D L K D V - A V L K V S R N T L T E R A L N Q L G -- E T I P								78	
RLAO_METMA	-----	M A E E R H H T E H I P Q W K K D E I E N I K E L I Q S H K V F G M V R I E G I L A T K I Q K I R R D L K D V - A V L K V S R N T L T E R A L N Q L G -- E S I P								78	
RLAO_ARCFU	-----	M A A V R G S - P P E Y K V R A V E E I K R M I S S K P V V A I V S F R N V P A G Q M O K I R R E F R G K - A E I K V V K N T L L E R A L D A L G -- G D Y L								75	
RLAO_METKA	M A V K A K G Q P P S G Y E P K V A E W K R R E V K E L K E L M D E Y E N V G L V D L E G I P A P Q L Q E I R A K L R E R D T I I R M S R N T L M R I A L E E K L D E R -- P E L E									88	
RLAO_METTH	-----	M A H Y A E W K K K E V Q E L H D L I K G Y E V V G I A N L A D I P A R Q L Q K M R Q T L R D S - A L I R M S K K T L I S L A L E K A G R E L -- E N V D								74	
RLAO_METTL	-----	M I T A E S E H K I A P W K I E E V N K L K E L L K N G Q I V A L V D M M E V P A R Q L Q E I R D K I R - G T M T L K M S R N T L I E R A I K E V A E E T G N P E F A								82	
RLAO_METVA	-----	M I D A K S E H K I A P W K I E E V N A L K E L L K S A N V I A L I D M M E V P A V Q L Q E I R D K I R - D Q M T L K M S R N T L I K R A V E E V A E E T G N P E F A								82	
RLAO_METJA	-----	M E T K V K A H V A P W K I E E V K T L K G L I K S K P V V A I V D M M D V P A P Q L Q E I R D K I R - D K V K L R M S R N T L I I R A L K E A A E E L N N P K L A								81	
RLAO_PYRAB	-----	M A H Y A E W K K K E V E E L A N L I K S Y P V I A L V D V S S M P A Y P L S Q M R R L I R E N G G L L R V S R N T L I E L A I K K A A Q E L G K P E L E								77	
RLAO_PYRHO	-----	M A H Y A E W K K K E V E E L A K L I K S Y P V I A L V D V S S M P A Y P L S Q M R R L I R E N G G L L R V S R N T L I E L A I K K A A K E L G K P E L E								77	
RLAO_PYRFU	-----	M A H Y A E W K K K E V E E L A N L I K S Y P V V A L V D V S S M P A Y P L S Q M R R L I R E N G G L L R V S R N T L I E L A I K K V A A Q E L G K P E L E								77	
RLAO_PYRKO	-----	M A H Y A E W K K K E V E E L A N I I K S Y P V I A L V D V A G V P A Y P L S K M R D K L R - G K A L L R V S R N T L I E L A I K R A A Q E L G Q P E L E								76	
RLAO_HALMA	-----	M S A E S E R K T E T I P E W K Q E E V D A I V E M I E S Y E S V G V V N I A G I P S R Q L Q D M R R D L H G T - A E L R V S R N T L L E R A L D D V D -- D G L E								79	
RLAO_HALVO	-----	M S E S E V R Q T E V I P Q W K R E E V D E L V D F I E S Y E S V G V V V G V A G I P S R Q L Q D M R R D L H G S - A A V R M S R N T L V N R A L D E V N -- D G F E								79	
RLAO_HALSA	-----	M S A E E Q R T T E E V P E W K R Q E V A E L V D L L E T Y D S V G V V V N V T G I P S K Q L Q D M R R G L H G Q - A A L R M S R N T L L V R A L E E A G -- D G L D								79	
RLAO_THE_AC	-----	M K E V S Q Q K K E L V N E I T O R I K A S R S V A I V D T A G I R T R Q I Q D I R G K N R G K - I N L K V I K K T L L F K A L E N L G D -- E K L S								72	
RLAO_THE_VO	-----	M R K I N P K K K E I V S E L A Q D I T K S K A V A I V D I K G V R T R Q M Q D I R A K N R D K - V K I K V V K K T L L F K A L D S I N D -- E K L T								72	
RLAO_PICTO	-----	M T E P A Q W K I D F V K N L E N E I N S R K V A A I V S I K G L R N N E F Q K I R N S I R D K - A R I K V S R A R L L R A I E N T G K -- N N I V								72	
	ruler	1.....	10.....	20.....	30.....	40.....	50.....	60.....	70.....	80.....	90

Motivation:

The main problem in multiple sequence alignment makes it so much harder than pairwise alignment is the curse of dimensionality of the cost hypercube. The memory cost to save the value in a hypercube increases exponentially.

```
RLAO_MET VA --MIDAKSEHKIA[PW]KIEEVNAL[KELL]KSANVIALIDMME[VPAV]CLOEIRDK
RLAO_MET JA --METKVKAH[VAPW]KIEEV[KTL]KG[IKSKP]VVAVDMD[VPAPO]QLEIRDK
RLAO_PYRAB -----MAHVAEWKKEVEELANL[IKSYP]VIALVDVSSMPAYPLSQMRRL
RLAO_PYRHO -----MAHVAEWKKEVEELAKL[IKSYP]VIALVDVSSMPAYPLSQMRRL
RLAO_PYRFU -----MAHVAEWKKEVEELANL[IKSYP]VVALVDVSSMPAYPLSQMRRL
RLAO_PYRKO -----MAHVAEWKKEVEELANI[IKSYP]VIALVDVAG[VPA]YPLSKMRDK
RLAO_HALMA [MSA]ESERKTETIP[E]PWKQEEVDAIVE[MIESYESVGVVN]IAGIPS[RQLQDMRRD
RLAO_HALVO [MSSE]EVQRQTEV[P]QWKREEVDELVDFIESYESVGVVN[GAGIPS[RQLQDSMRRE
RLAO_HALSA [MSA]EEQRRTTEEV[P]EWKRQEVAELVDLLETYDSVGVVNVTGIPS[KQLOQDMRRG
RLAO_THE AC -----MKEVSQQKKELVNEIT[ORIKASRSVAIVDTAGIRTRQI[D]IRGK
RLAO_THE VO -----MRKINPKKKEIVSELAD[IT]KS[KA]VAVIDIKGVVR[ROMOD]IRAK
RLAO_PICTO -----MTEPAQWKIDFVKNLENEINSRKVAAIVSIKGLRNNEFOKIRNS
```



Method:

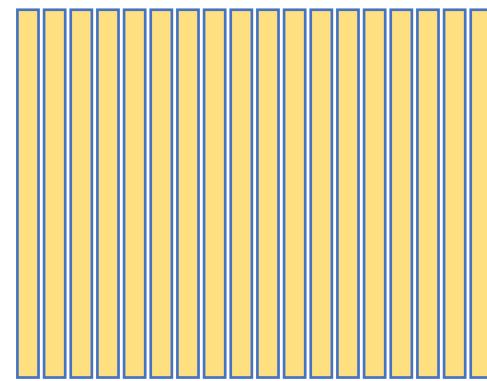
We can substitute the Matching score with an embedding vector cosine similarity. We find this embeddings have the ability to measure the dissimilarity between aligned patches.

**Characters
sequence**

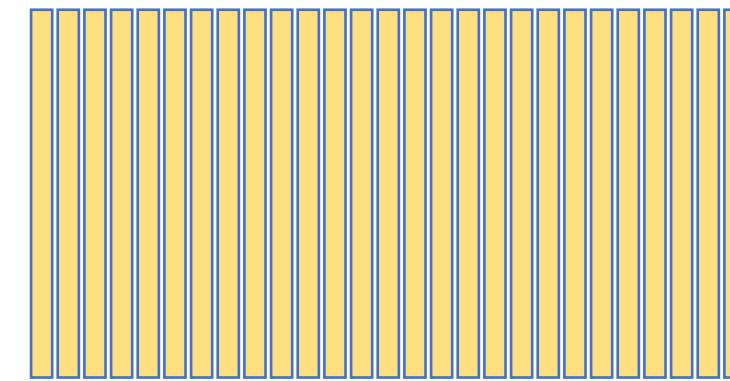
MSSARFDSSD ... KIGDQEFDHLPAIEPAPRLVTL



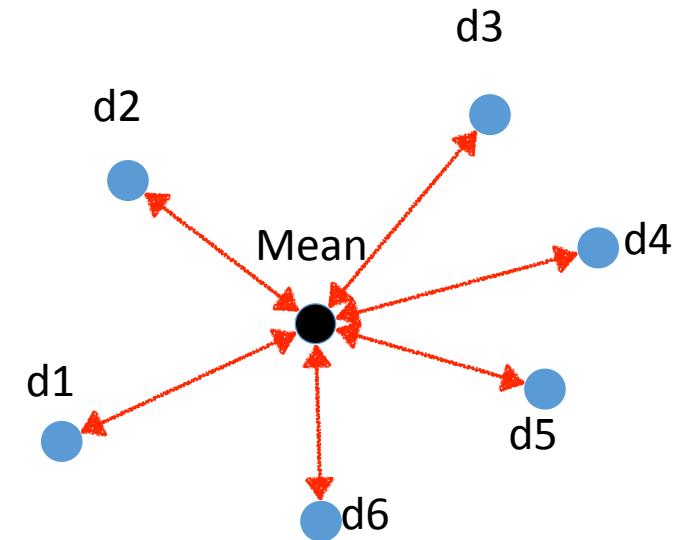
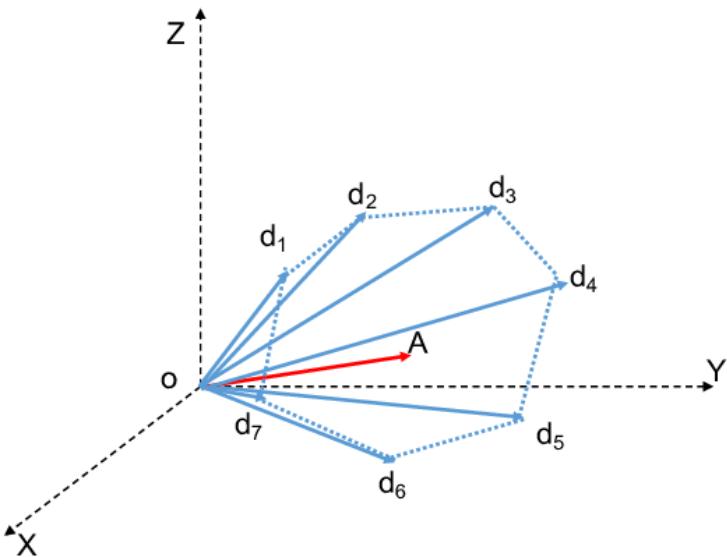
**Vector
sequence**



...



The dissimilarity between multiple patches can be captured by the convex cone formed by the N embedding vectors, since they preserve the relationship in their cosine similarity.



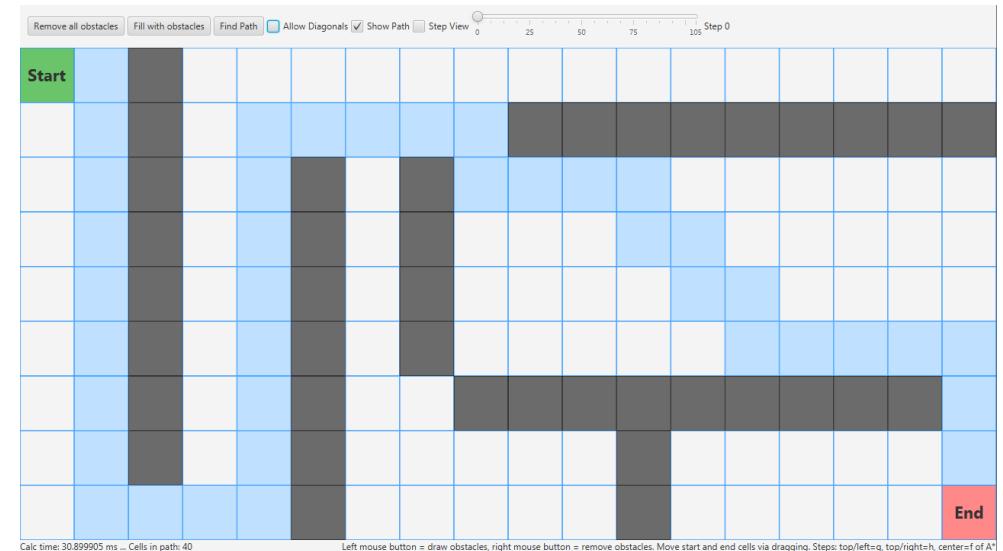
A* algorithm

A* is a classical path search algorithm. At each iteration of its main loop, A* needs to determine which of its paths to extend. It does so based on the cost of the path and an estimate of the cost required to extend the path all the way to the goal. Specifically, A* selects the path that minimizes

$$f(n) = g(n) + h(n)$$

where n is the next node on the path, $g(n)$ is the cost of the path from the start node to n , and $h(n)$ is a **heuristic** function that estimates the cost of the cheapest path from n to the goal. Which encourage the agent go directly to the goal.

In this way, we only need to save the values on the path. It is a way to reduce the computational complexity to O(NL).



The max over an exponential number of choices is still a barrier in the path search on hypercube.
e.g. find the max/min value in the bellman recursion step.

$$V(I_0) = \text{cosy}(I_0) + \beta \max_{a:I_0 \rightarrow I_1} V(I_0, a)$$

As shown in the figure, the possible move for even a single step increases exponentially w.r.t. the number of sequences to work on (note as N).

It is easy to see the single step chooses correspond to all vertex on the hypercube except for the origin.

$(1, 0)$

$(1, 1)$

$(0, 1)$

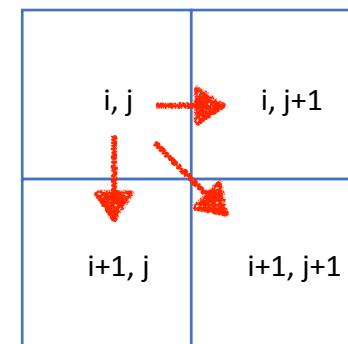
$(1, 0, 0)(0, 1, 1)$

$(0, 1, 0)(1, 0, 1)$

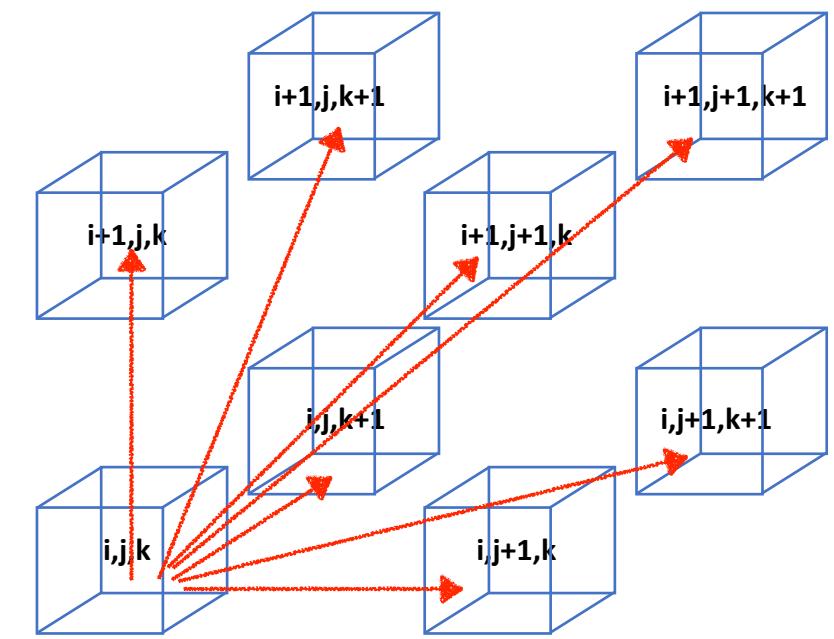
$(0, 0, 1)(1, 1, 0)$

$(1, 1, 1)$

Choices on square



Choices on 3D cube



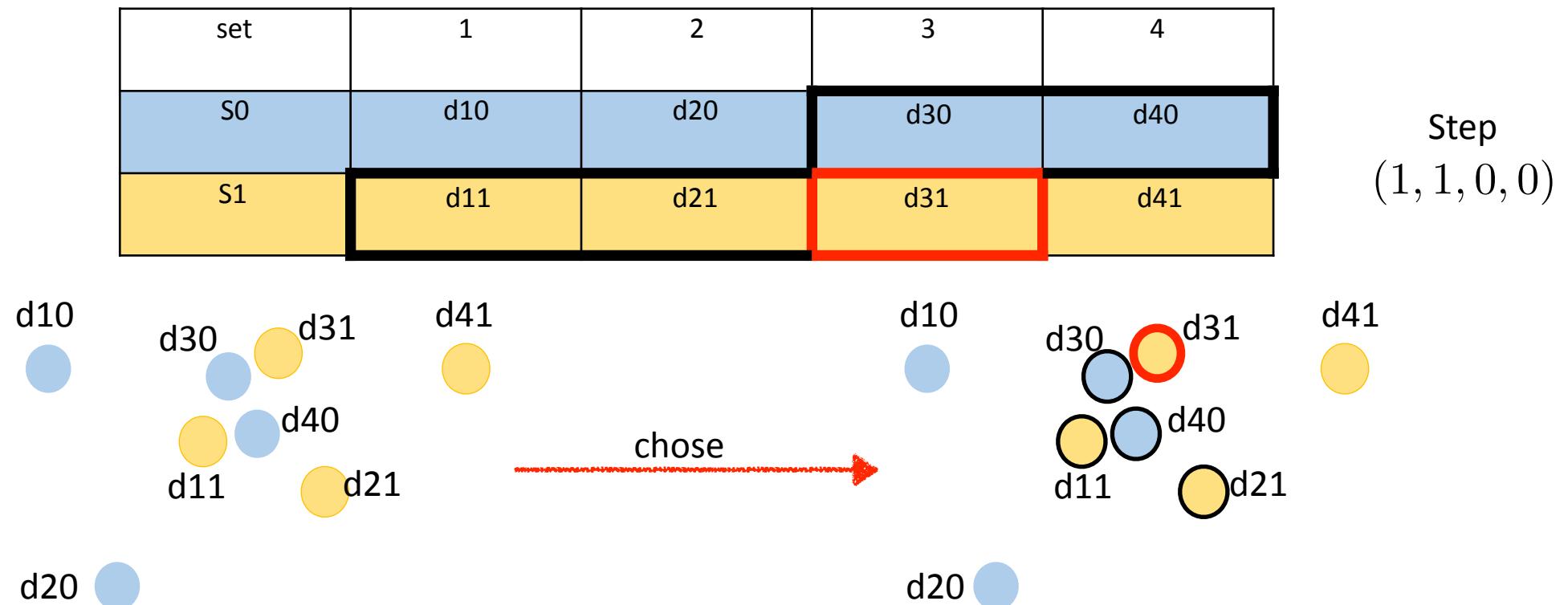
N sequences, ND,

2^N Neighbors.

So the min choice is equal to:

Given N pair of data, each pair has one data from set S0 and one from S1. Try to choose one element from each pair of data, such that the chosen element form the most compact cluster.

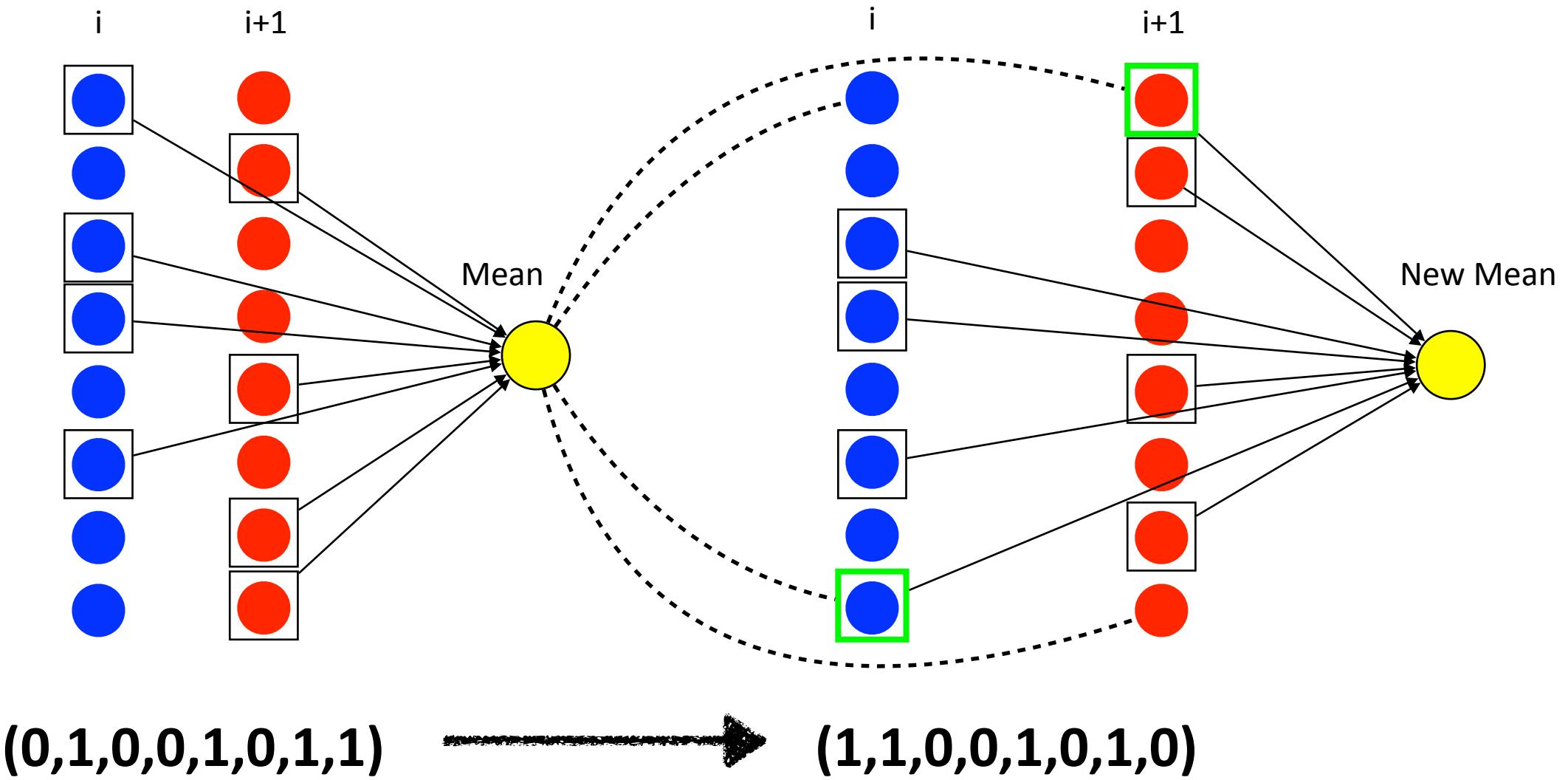
A solution example on 4D hypercube, we chose d11, d21, d30, d40 to form the most compact cluster. We give up on d31, because we can only chose 1 element in each pair (each column). And the choice we made give use the step (1,1,0,0)



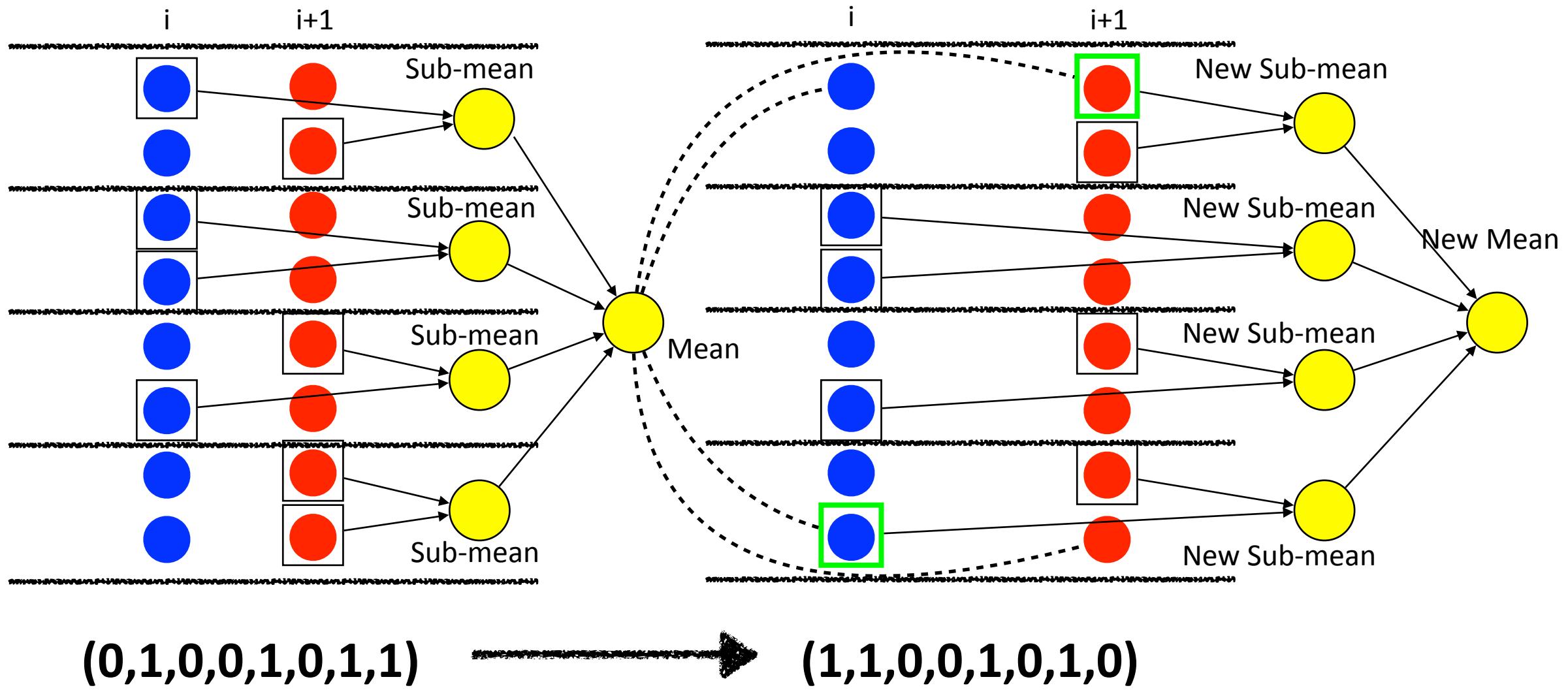
We can find a local optimal solution for this min choice problem in a iterative way similar to k-means algorithms. The convergence to a local optimal is guaranteed.

```
chose initial choice to be step = (1, 1, 1, 1, ..., 1).
compute the initial center C_0, of by all embeddings at position 1.
C_old = C_0
while C_new != C_old:
    C_old = C_new
    for i = 1 to N:
        current_choice = step[i]  (0 or 1)
        alternate_choice = 1 - step[i]  (1 or 0)
        distance_current = D(C_old, embedding[i][current_choice])
        distance_alternate = D(C_old, embedding[i][alternate_choice])
        if distance_current > distance_alternate:
            step[i] = 1 - step[i]
    C_new = sum(embedding[i][step[i]])
```

Sequential



Parallel



Code Structure

```
#include <stdio.h>
#include <omp.h>
#include <sstream>
#include <iostream>
#include <string>
#include <vector>
```

omp_set_num_threads(N);

Initialize vectors for sequences

Initialize empty mean vectors

check # thread avail
print, define variables

while(not over)

#pragma omp parallel private(myid){

Parallel Code

myid = omp_get_thread_num();

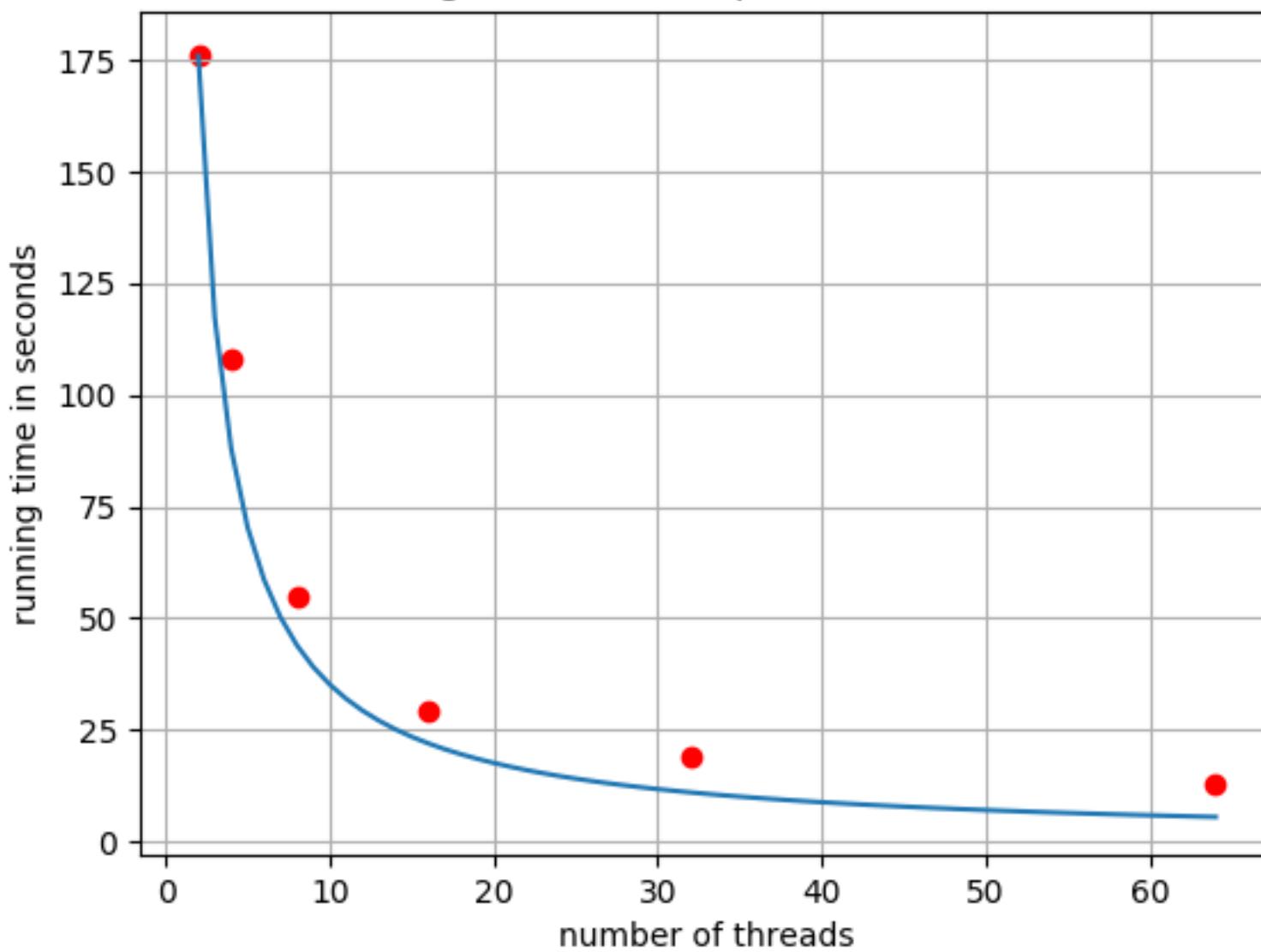
Parallel Code

}

Sequential Code

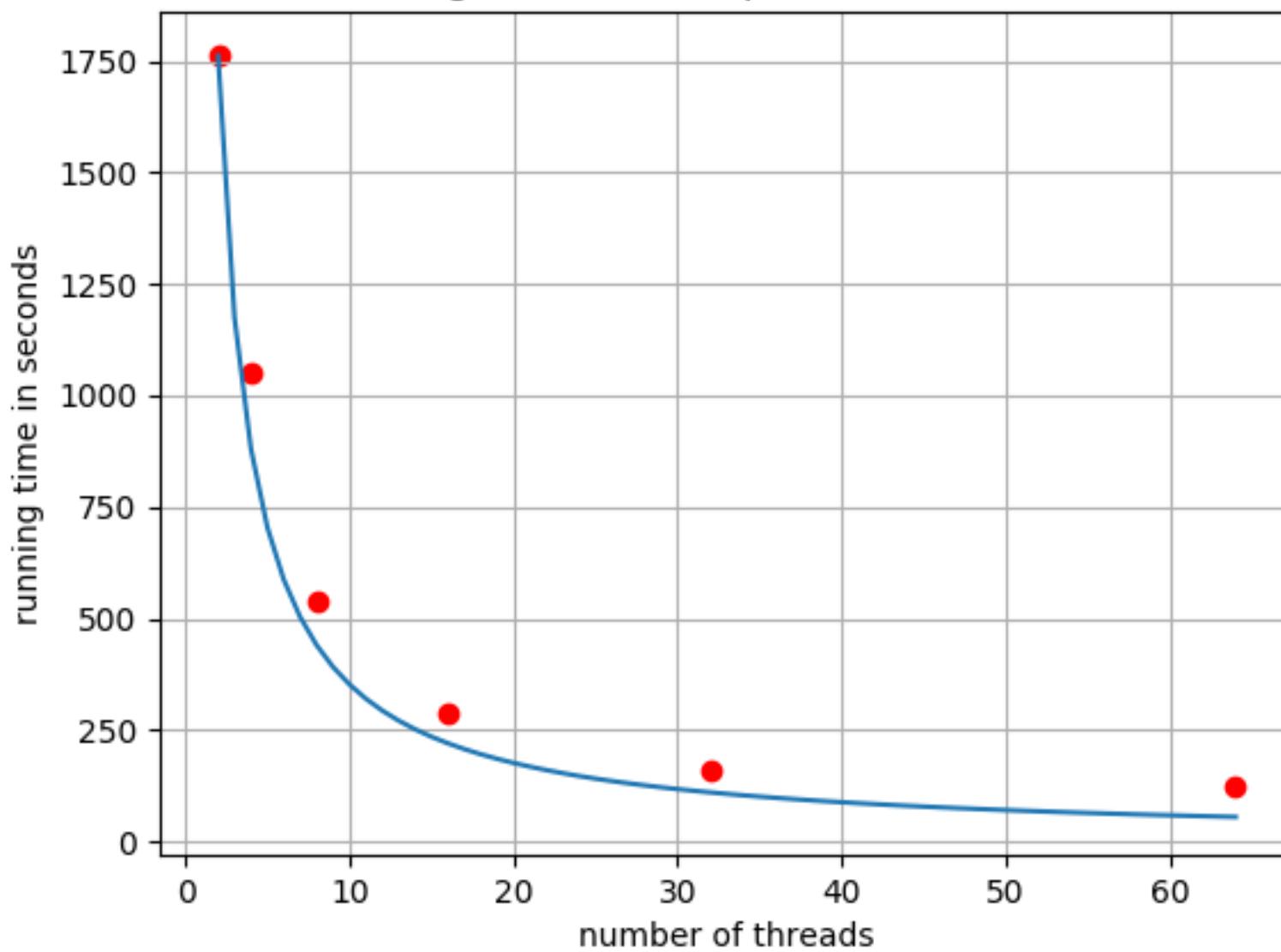
Result fix total

length 1000, # seq 204800, d 100



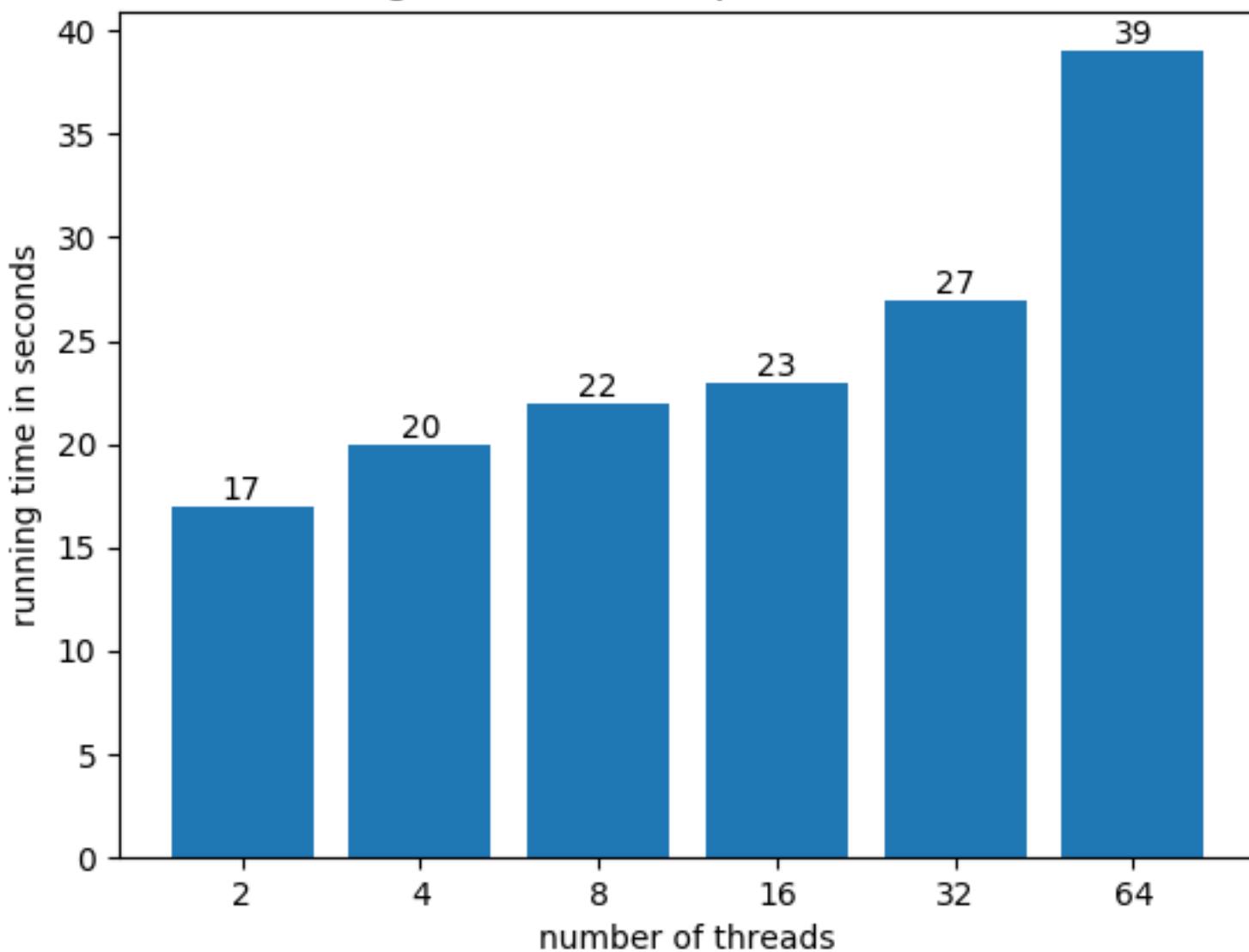
Result fix total

length 1000, # seq 2048000, d 100



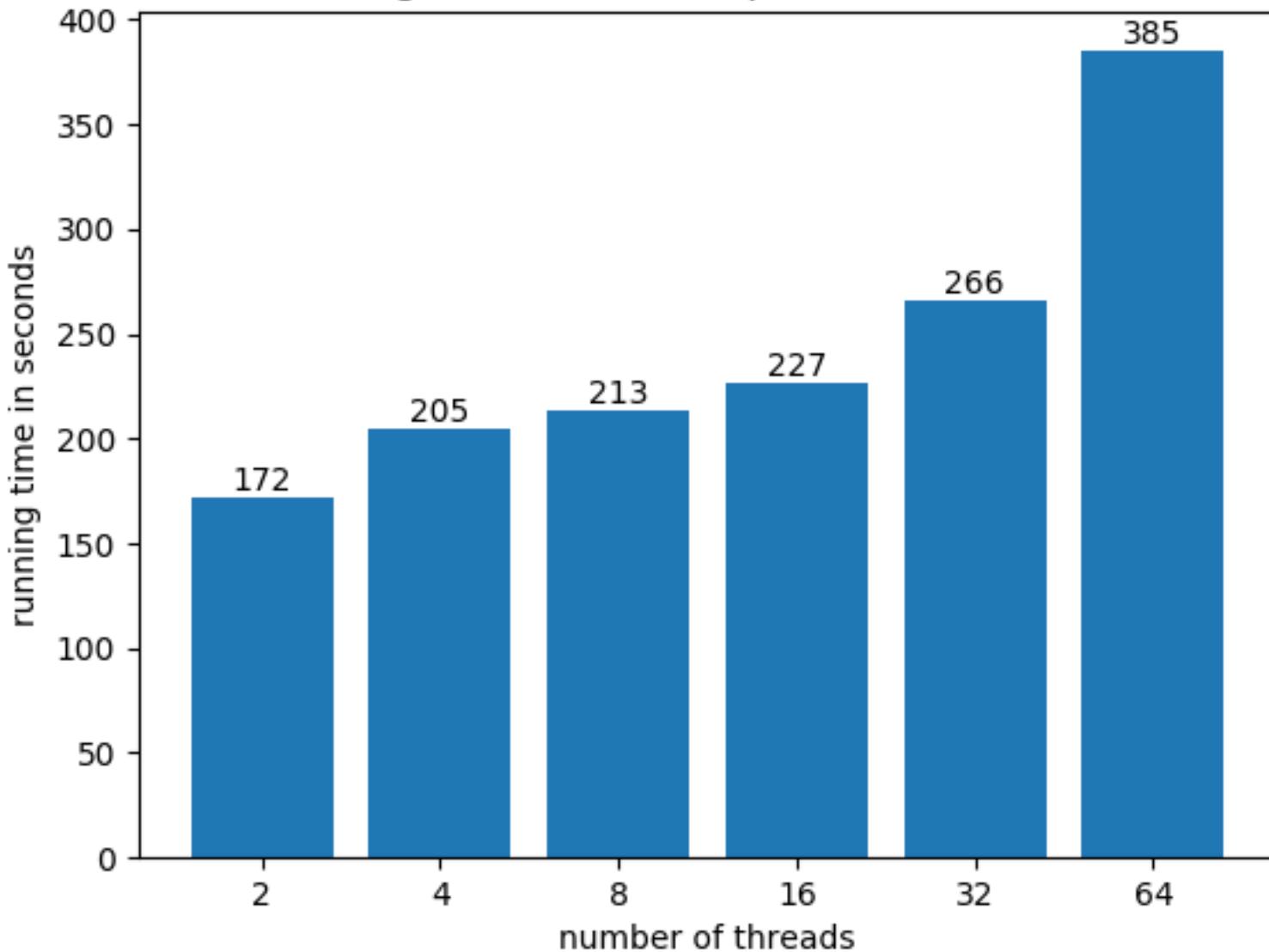
Result fix thread load

length 1000, 10000 per thread, d 100



Result fix thread load

length 1000, 100000 per thread, d 100



Thanks