From Eliminating Irrelevant Factors to Determining the Meeting Venue—A Computational Approach

—"I'm feeling tired."

Abstract

International meetings are increasingly common in business and academic communities due to globalisation and the demand of cooperation in all kinds of industries.

Therefore, a problem of paramount importance is to decide the host city for such meetings. The organiser of these international meetings may first receive the list of all attendees and where they are from and then choose the optimal host city in consideration of the productivity in the meeting.

To study the effect of different factors to the productivity of attendees from different countries, we can first refer to the result of the International Olympiad in Informatics (IOI) as its nature is similar to the meeting mentioned above, and each contestant has already been assigned a score. We calculate the difference of a country's relative ranking and its average relative ranking in the three most recent years to isolate the effect of factors that varies each year from those that have a permanent effect.

We then forcibly do linear regression on this measure of performance against different factors, including time zone difference, temperature difference, sunshine duration difference, flight distance, and elevation difference between a contestant's home city and the host city. Student's t-test then show that we have no evidence to reject the null hypothesis that the regressions of productivity against temperature difference, sunshine duration difference and elevation difference each has zero slope.

Then we focus on minimising the total time difference between contestants' home cities and the host city, and also minimising the flight distance in order to lower the cost with the premise that we do not violate the former constraint. We develop several algorithms to fulfill our purpose. The first one assumes that the productivity drops linearly as the time zone difference increases, which can perform extremely well in determining the longitude in $O(n)$ time. The second one which does not require that assumption can produce an answer accurately yet not so efficiently. Afterwards we choose the optimal city near that longitude. We also take in account the fact that jet lag due to flying eastbound and westbound has different severity.

Hence we can input the test cases given in the problem statement to decide where we should hold the meeting. Our conclusion is that, for the smaller test case, Perth, Australia is the best choice, and for the larger test case, our recommendation is that the meeting be held in Cape Town, South Africa.

Content

1 Introduction……3
1.1 Problem Interpretation……3
1.2 Assumptions……3
1.3 Factors Considered……3
2 Calculation of Flight Distance……4
3 Exclusion of Other Factors……5
3.1 Choice of Quantitised Data Describing Productivity……5
3.2 Data Processing……6
3.2.1 The Average Relative Ranking……6
3.2.2 Performance Index as the Difference of Yearly Performance……6
3.3 Implementation of Statistical Computation……7
3.4 Findings……7
3.4.1 Temperature Difference……7
3.4.2 Sunshine Duration Difference……8
3.4.3 Elevation Difference……8
3.4.4 Latitude Difference……9
3.4.5 Time Zone Difference……9
3.4.6 Flight Distance……10
4 Modelling the Effect of Jet Lag……10
5 Determining the City Where the Meeting Should Be Held……11
5.1 Naïve Algorithm……11
5.2 Target of the Algorithm……12
5.3 The First Step……12
5.4 The Specific Linear Case……12
5.4.1 Case Specification……12
5.4.2 The Critical Point……13
5.4.3 Monotonicity Leads to Further Simplification……13
5.4.4 Fast State Transition……14
5.4.5 Analysing Time Complexity……14
5.5 The General Non-linear Case……14
5.5.1 Precision in the Approximation……15
5.5.2 Choosing the Optimal Longitude……15
5.5.3 Analysing Time Complexity……15
5.6 The Optimal Latitude……15
5.6.1 Dividing the Meridean……15
5.6.2 Possible Improvement……16
6 Case Studies……17
6.1 The Small Test Case……17
6.2 The Large Test Case……18
7 Further Discussion……19
7.1 Advantages of the Algorithm……19
7.2 Disadvantages of the Algorithm……19
8 Appendix……19
8.1 Source Code in C++ as Implementation of Section 5……19
8.2 Reference……22

1 Introduction

In this modern society, scholars and entrepreneurs more often than not have to attend international meetings with their counterparts all over the world. As different people live in different areas, they have to locate somewhere to hold the conference, thus choosing an optimal place for them to congregate is an inevitable task in order to maintain a quality conference and boost their productivity.

1.1 Problem Interpretation

Our goal is to develop an algorithm that can produce a destination given the distribution of the attendants and the conference period maximising the productivity of the whole meeting.

1.2 Assumptions

- Each attendant has exactly the same habitual sleep schedule in their hometown. we also neglect short term productivity boost such as caffeine.

Justification: We eliminating the personal differences in lifestyle for simplicity.

- The organiser always has enough budget to host the meeting provided that it does not pay for the extra cost incurred if some participants arrive early.

Justification: As given in the problem statement, the organiser does not need to pay for early arrivals. We assume that it has enough budget to host the meeting at any city in the world, not limited to a few specific cities.

- The meeting is held in a city.

Justification: Rural areas may not have suitable conference facilities for formal business or academic meetings

- The Earth's surface is a perfect sphere.

Justification: This is used in calculating the distance travelling by a flight. The variation in the radius of the Earth at different direction is negligible compared with this distance.

1.3 Factors Considered

Due to the distribution of participants in the world, it is important to consider all the elements which may affect the attendees' performances. We first list out the factors as follows, and then examine their correlation to the attendees' productivity and see if they really impact on the productivity level.

- The flight time

Justification: Staying in an enclosed and cramped space in the cabin for several hours will somewhat affect their rests and hence reduce their productivity.

- Jet lag

Justification: In our own experience, the fatigue due to jet lag makes us uncomfortable and thus reduces our productivity.

- The difference in weather and climate

Justification: For example, the larger the temperature difference between the destination and the participant's home town, the longer the time it may take for the participants to acclimatise. This process may also affect their productivity. Other factors in this category includes sunshine duration difference and elevation difference.

2 Calculation of Flight Distance

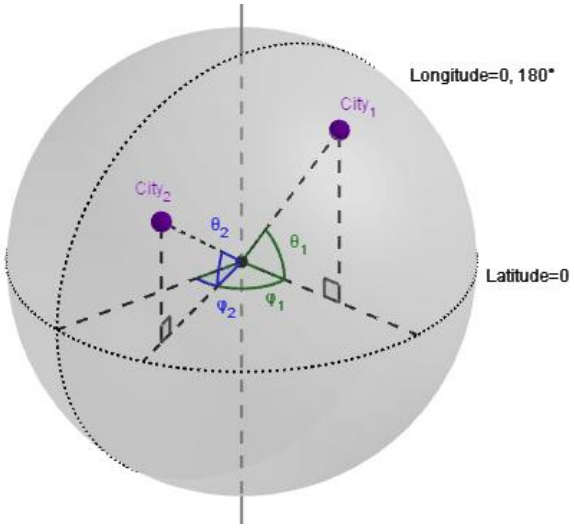
Though this may not be of highest priority, we need to minimise the flight distance because the participant may feel exhausted after a longer journey and it is also better if we can reduce the budget on transport.

Given the coordinates of a city and the host city, we want to find their distance along the surface of the Earth, namely the arc length between the two points of the great circle passing through both points on the sphere.

If city A has latitude θ_1 and longitude ϕ_1, and city B has latitude θ_2, then their coordinates in the 3-D Cartesian space are

$$A = (R\cos\theta_1\cos\phi_1, R\cos\theta_1\sin\phi_1, R\sin\theta_1),$$
$$B = (R\cos\theta_2\cos\phi_2, R\cos\theta_2\sin\phi_2, R\sin\theta_2),$$

respectively, assuming that R is the radius of the Earth, the xOy-plane is the equatorial plane, and the xOz-plane passes through Prime meridian. Here latitudes are positive towards north and longitudes towards east.



As the length of the great arc only depends on the angle between $\overrightarrow{OA}$ and $\overrightarrow{OB}$, we can first find the angle α between $\overrightarrow{OA}$ and $\overrightarrow{OB}$.

$$\cos\alpha = \frac{\overrightarrow{OA}\cdot\overrightarrow{OB}}{R^2}$$
$$= \cos\theta_1\cos\phi_1\cos\theta_2\cos\phi_2 + \cos\theta_1\sin\phi_1\cos\theta_2\sin\phi_2 + \sin\theta_1\sin\theta_2$$
$$= \cos\theta_1\cos\theta_2\cos(\phi_1-\phi_2) + \sin\theta_1\sin\theta_2$$
$$\therefore \alpha = \arccos(\cos\theta_1\cos\theta_2\cos(\phi_1-\phi_2) + \sin\theta_1\sin\theta_2)$$

With this formula, we can compute the distance between any pair of cities when we need.

3 Exclusion of Other Factors

3.1 Choice of Quantitised Data Describing Productivity

It is difficult to find any data that measure the contribution or productivity in terms of intellectual labour of each individual in an international meeting in a quantitised way.

However, some international competitions have a similar setting that can compare to the conference as stated in the problem statement, and each contestant's intellectual productivity can be measured by their score in the competition. This gives us some information to work on and enable us to see which proposed factors above have really affected the productivity level.

The International Olympiad in Informatics (IOI) is one of such competitions. A benefit of choosing this competition is that it offers a comprehensive historical information about each participant's score, absolute ranking and relative ranking online[1]. Furthermore, the country that the contestant represented is also shown so we can track where they were from.

3.2 Data Processing

As we assume that many factors would affect the contestants' performances, we have to gather all sorts of information about the contestants' hometown including the geographic coordinates, time zone, temperature, sunshine duration from websites like ClimaTemps.com[2], Wikipedia[3][4]. Besides, we only know which country the contestants were from, but not exactly their hometown, so the best we can do is to assume that their hometown is one of the major cities[5] (e.g. most populated, capital, largest area) of their country, whose data usually represents the typical climate in the country.

3.2.1 The Average Relative Ranking

Now we have to measure the contestants' productivity. As the national-wide education level on science subjects may vary over years, and we would like to isolate the effect of jet lag, etc. in each year, we propose the following performance index to describe a country's yearly performance in the IOI, which is considered the productivity outcome affected by the factors we are to examine.

The team for each country has at most 4 contestants. Their relative ranking in the competition (i.e. what proportion of all contestants a contestant is better than or equal to) is averaged.

3.2.2 Performance Index as the Difference of Yearly Performance

Then we calculate the difference of this yearly average performance and the average performance over the three most recent years (i.e. that year, the year before that year, and the year before the year before that year and if the country is absent in any of the year mentioned above, its data is substituted by its performance in the still previous year). In short,

$$\Delta perf = year_n - \frac{year_n + \mathrm{year}_{\mathrm{n-1}} + \mathrm{year}_{\mathrm{n-2}}}{3}.$$

This would provide a good estimate of the countries performance as other permanent factors e.g. the level of education development are cancelled out when we calculate the difference.

3.3 Implementation of Statistical Computation

After gathering the data (please refer to the screenshot of the Excel file in appendix), we can use the R programming language to perform Student's t-test

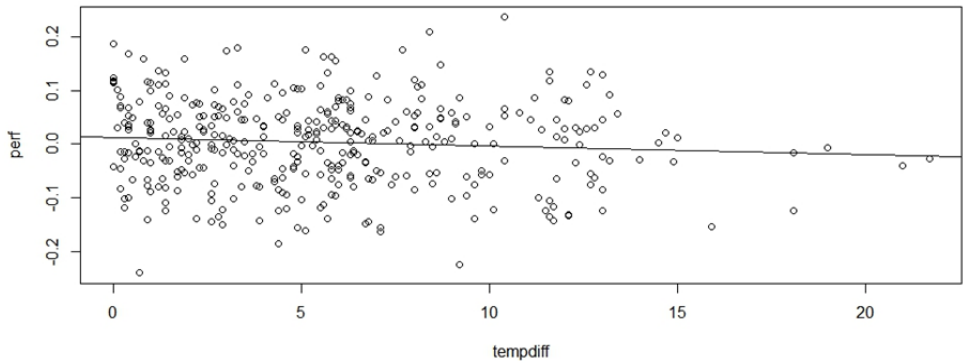
and determine if we can reject the null hypothesis that the regression line of $\Delta perf$ against some of the factors has slope 0, which is equivalent to saying that we do not have any evidence that the supposed factor actually contributes to the productivity of contestants.

For example, if we store the performance index as in Section 3.2 in an array `perf`, and the absolute difference of temperature in July (when the IOI is held) in another array `tempdiff`, we can use the command `plot(perf~tempdiff)` to show the scatter diagram and `abline(lm(perf~tempdiff))` to forcibly add the best fit line. Then we retrospectively check the detail of our modal with `summary(lm(perf~tempdiff))` to see the P-value for Student's t-test.

The use of linear model is justified below. Judging from each of the scattering diagrams, although no obvious linear correlation is seen and the residuals have too large a variance, the performance index is roughly normally distributed for each specific value of the independent variables.

3.4 Findings

3.4.1 Temperature Difference

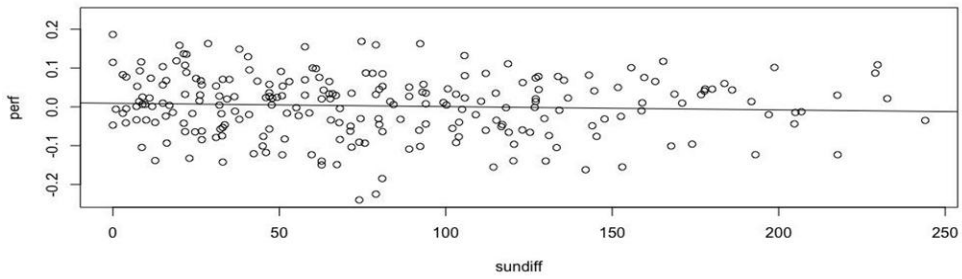


	Estimate	Std. Error	t value	Pr(>|t|)
(Intercept)	0.0118609	0.0068496	1.732	0.0842
tempdiff	-0.0015460	0.0009912	-1.560	0.1196
Residual standard error: 0.07913 on 376 degrees of freedom				
Multiple R-squared: 0.006429, Adjusted R-squared: 0.003787				
F-statistic: 2.433 on 1 and 376 DF, p-value: 0.1196				

By common sense, the conference is held in an air-conditioned room or hall. Also, the hotel where they stay should also provide them a comfortable environment which implies that the temperature is well-managed and the air

temperature outdoors should not matter, which agrees with our statistical results that the slope of the regression line does not significantly differ from zero.

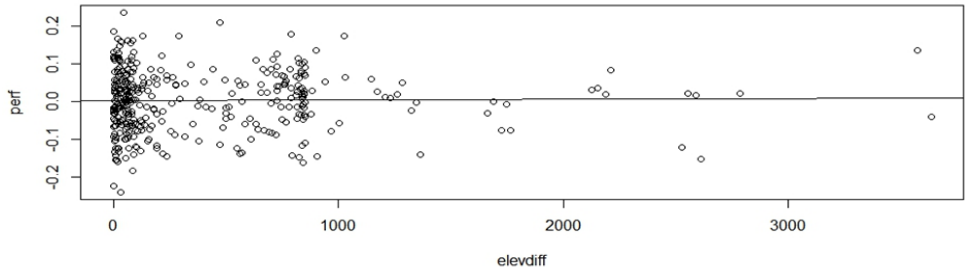
3.4.2 Sunshine Duration Difference



	Estimate	Std. Error	t value	Pr(>|t|)
(Intercept)	9.135e-03	8.832e-03	1.034	0.302
sundiff	-8.811e-05	8.945e-05	0.985	0.326
Residual standard error: 0.07866 on 231 degrees of freedom				
Multiple R-squared: 0.004183, Adjusted R-squared: -0.0001278				
F-statistic: 0.9704 on 1 and 231 DF, p-value: 0.3256				

By common sense, conference is held in an enclosed illuminated area, so the lighting they enjoy has nothing to do with the sunlight outside.

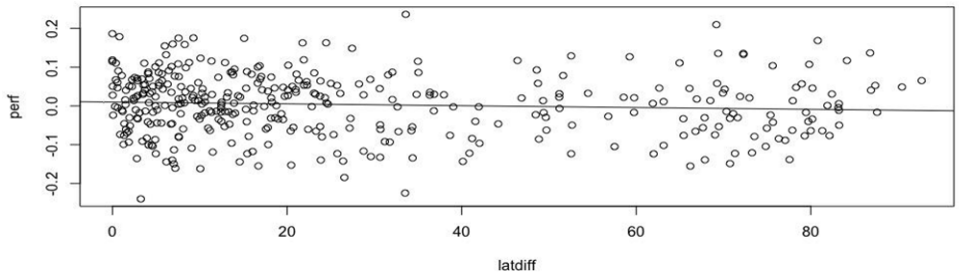
3.4.3 Elevation Difference



	Estimate	Std. Error	t value	Pr(>|t|)
(Intercept)	2.603e-03	5.011e-03	0.520	0.604
elevdiff	1.698e-06	7.427e-06	0.229	0.819
Residual standard error: 0.07938 on 376 degrees of freedom				
(680 observations deleted due to missingness)				
Multiple R-squared: 0.000139, Adjusted R-squared: -0.00252				
F-statistic: 0.05228 on 1 and 376 DF, p-value: 0.8193				

In general few cities are built on high mountains, so the elevation difference is at most 3000m, which does not have great impact on our health.

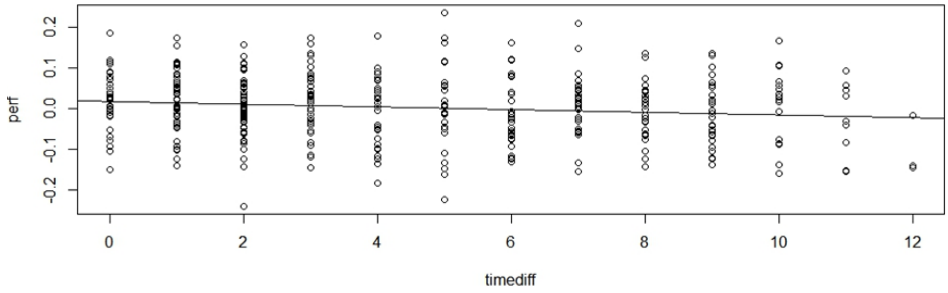
3.4.4 Latitude Difference



	Estimate	Std. Error	t value	Pr(>|t|)
(Intercept)	0.0099262	0.0059154	1.678	0.0942
latdiff	0.0002404	0.0001550	-1.551	0.1217
Residual standard error: 0.07913 on 376 degrees of freedom				
Multiple R-squared: 0.006359, Adjusted R-squared: 0.003716				
F-statistic: 2.406 on 1 and 376 DF, p-value: 0.1217				

Latitude is highly correlated to temperature difference and sunshine duration difference, but the regression line of performance against each of them does not have a slope that is significantly non-zero, so it is reasonable to expect that it is also the case for latitude.

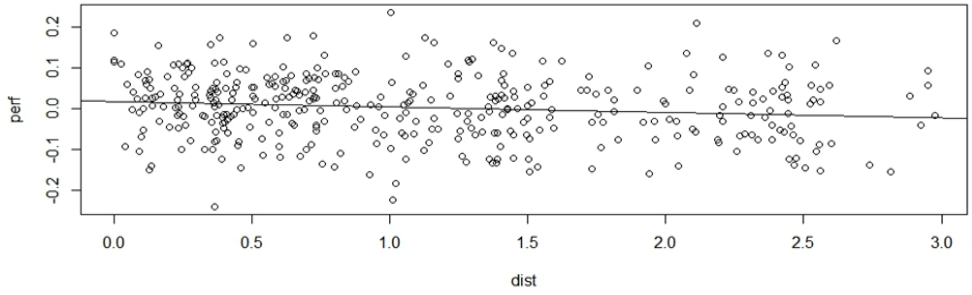
3.4.5 Time Zone Difference



	Estimate	Std. Error	t value	Pr(>|t|)
(Intercept)	0.018839	0.007003	2.690	0.00746 **
timediff	0.003426	0.001258	2.724	0.00676 **
Residual standard error: 0.07861 on 376 degrees of freedom				
Multiple R-squared: 0.01935, Adjusted R-squared: 0.01674				
F-statistic: 7.418 on 1 and 376 DF, p-value: 0.006757				

Although the variance in time zone difference cannot explain the variance of performance in the scattering plot, it is evident from the t-test that we should reject the null hypothesis that the slope of the regression line is zero.

3.4.6 Flight Distance

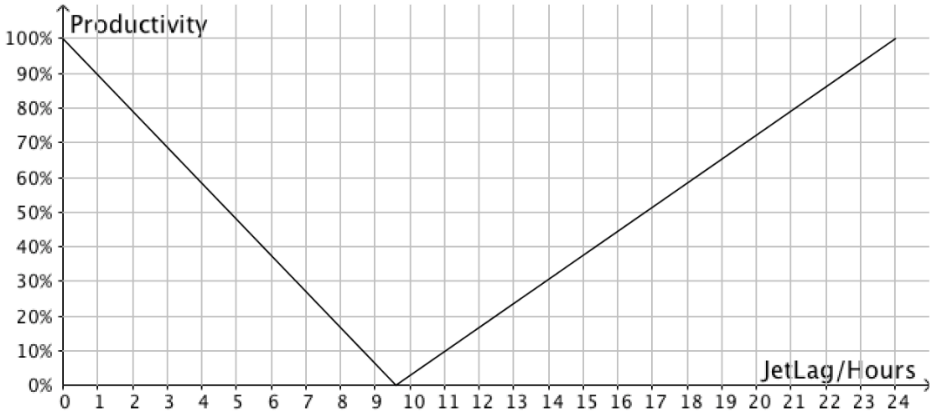


	Estimate	Std. Error	t value	Pr(>|t|)
(Intercept)	0.018382	0.007062	2.603	0.00961 **
dist	0.013556	0.005191	2.612	0.00938 **
Residual standard error: 0.07868 on 376 degrees of freedom				
Multiple R-squared: 0.01782, Adjusted R-squared: 0.0152				
F-statistic: 6.82 on 1 and 376 DF, p-value: 0.009375				

In our diagram, the relationship between flight distance and performances of the participants is mainly due to the strong relationship between flight distance and time zone difference between the host and the cities; therefore, we do not have to consider the flight distance independently.

4 Modelling the Effect of Jet Lag

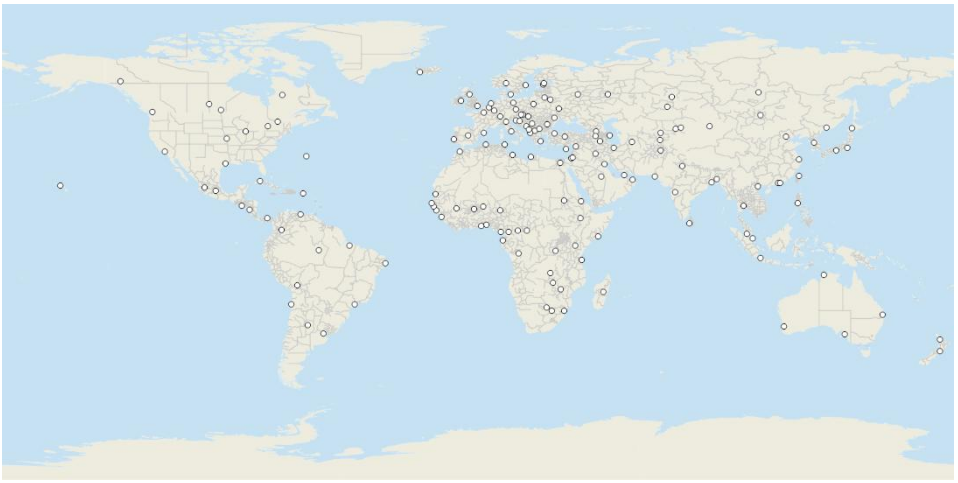
In our common sense, it is more difficult for one to revive from jet lag after an eastbound flight; therefore, productivity of scientists or businessmen from the west may be lower than that from the east. One can adopt different therapies, such as light therapy and using melatonin, to adjust their circadian rhythm in advance or after arrival. Though not an academic source, an article on *USA Today* states that the maximum shift in circadian rhythm per day is about 1.5 hours per day for westward travel and 1 hour per day for eastward travel[6], which is relatively small compared to the time zone difference. For simplicity, we would directly adopt this figure for the comparison between eastbound flight and westbound flight. Hence it is reasonable to ignore the choice of prevention used before the flight. Based on these data, we may assume the decrease in productivity due to jet lag is a function of the time zone difference.



5 Determining the City where the Meeting Should Be Held

5.1 Naïve Algorithm

To choose the most suitable city, we can simply enumerate all the cities on Earth and calculate the total productivity level once for each of them. Then we can pick the one with the maximal sum of productivity to host the meeting. Therefore, we list out around 150 major cities in the world with an international airport as candidates.



[Map] Generated with qGIS.

This is a very simple yet naïve method that requires a comprehensive list of major cities. However, we may not have such a global list. We must solve it in another mathematical way to locate the coordinate of the optimal point and then find a city near it because it is easier for a local city list to be complete.

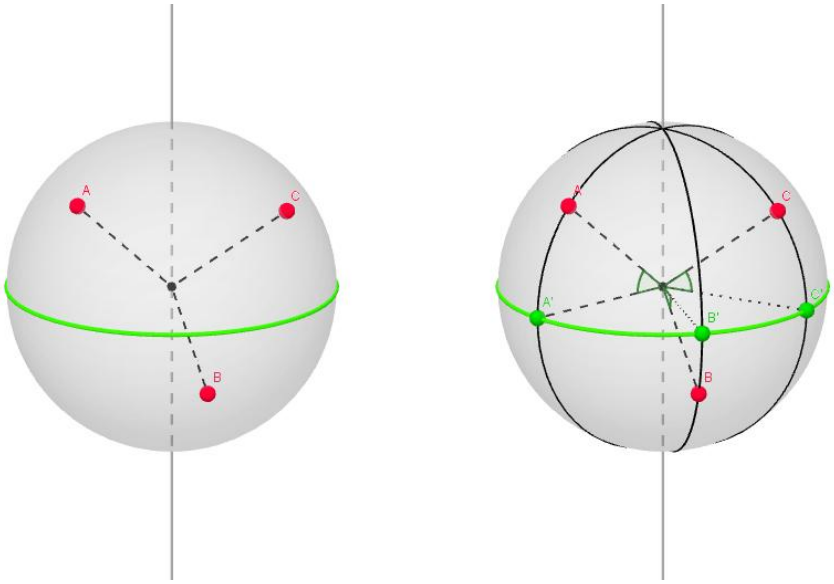
5.2 Target of the Algorithm

At the moment, only the time zone difference concerns us. In order to simplify the question, we may only consider the equator of the Earth and place every

participant on the equator according to the longitude of their home town because the time zone difference only depends on the longitude of where the participants are from and the conference is held. Hence, we can find an optimal point on the circle minimising the total productivity loss (i.e. the sum of the loss of productivity for each person). Detailed algorithm is discussed below.

5.3 The First Step

We can project all the home cities of the participants on the Equator according to their longitude. We then obtain a diagram like the one below.



[Diagram] Red points represent the cities, while green points are the projections of the cities. Generated with GeoGebra.

After that, we have two different strategies to pick the optimal host city in two different cases.

5.4 The Specific Linear Case

5.4.1 Case Specification

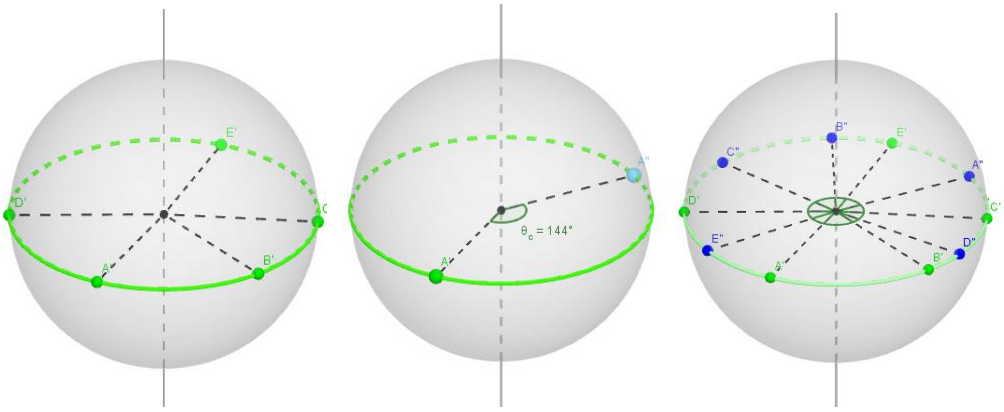
As mentioned above in Section 4, the difficulty of recovery from an eastbound flight (with positive time difference) is harder than that from a westbound flight (with negative time difference) and the recovering efficiency is 1.5 times higher for a westbound flight. For simplicity, we assume that the graph of productivity against time zone difference is a V-shaped line and resembles the graph of recovering time needed against time zone difference. Hence, the productivity zero point should be at time difference $+24 \times 2/(3+2) = +9.6$ hours, or equivalently, $-24 \times 3/(3+$

$2) = -14.4$ hours, with regards to the fact that the common habitual sleeping schedule is periodical every 24 hours. (N.B. Here the productivity zero point does not necessarily mean a state when an attendee cannot make any contribution in a meeting. It is just the hypothetical minimum of productivity in our scale, from 0 to 1)

5.4.2 The Critical Point

We define the critical point of a longitude as the other longitude at which the productivity will become zero if we travel from the original longitude, i.e. the point on the equator that is $144°$ eastward with respect to the original point on the equator.

After the step in Section 5.2, we can create the corresponding critical point of every point on the equator that already exists.



[Diagram] Blue points are the corresponding critical points of different cities' projections. Generated with GeoGebra.

We claim that the minimum must appear at either a critical point or an original point. After using fast state transition to calculate the total productivity at every point (cities and their corresponding points), the optimal point can be easily found in $O(n)$ time.

5.4.3 Monotonicity Leads to Further Simplification

Claim

The minimum must appear at either a critical point or an original point.

Proof

Let A be the set of original points from which the attendees must travel

eastbound (no more than $144°$ longitude) to reach the host city. Likewise, let B denote the set of original points from which the attendees have to travel westbound (no more than $216°$ longitude) to reach the host city.

For any two consecutive points H, K (critical or original) on the equator, if we choose any point between them to be the host, A and B will both remain the same set as we move the host along the arc segment $\widehat{HK}$ because we do not pass through any critical point of a home city; therefore, the change in productivity will be linear in terms of x, the angular displacement between the host and one of the two consecutive points, so obviously the local extremum can only be reached when $x = 0$ or $x = d$, where d is the angular displacement between H and K.

5.4.4 Fast State Transition

If $|A| = a$ and $|B| = n - a$, where n denotes the total number of participants, are known, the change when we move the host from one city to another can be calculated in $O(1)$ time. Every time we cross no matter an original point or a critical point, either $|A|$ increases by 1 and $|B|$ decreases by 1 or $|B|$ increases by 1 and $|A|$ decreases by 1. If we move the host $d°$ eastward without crossing any point, the change in total productivity is simply $-a \cdot \frac{d}{144} + (n - a) \cdot \frac{d}{216}$

5.4.5 Analysing Time Complexity:

When we initialise the process, we need to do an $O(n)$ total productivity calculation. After initialising, each state transition when we move the host from one point to another point can be done in $O(1)$ time. As there are a total of $2n$ points on the equator, the total time complexity will be $O(n) + O(n) = O(n)$.

5.5 The General Non-linear Case

In this second case, we no longer consider the situation when the graph of productivity against time zone difference is a V-shaped line as in Section 4; therefore, we can no longer solve the problem as above anymore because the claim that any extremum must be at the discretised longitudes of home cities or their corresponding critical points does not generally hold since the formula for the decrease in productivity may not be a linear function. Thus, we have to develop another algorithm to achieve the task.

5.5.1 Precision in the Approximation

Even if we can locate a very precise location to be our destination, for example, a distinctive tree at the north-west edge of a terrace, it is completely meaningless, so we can set the precision to $0.01°$, which suffices for our use.

5.5.2 Choosing the Optimal Longitude

We divide longitude into 36000 parts. For the end of each interval of longitude, we calculate the total productivity once and then we can get the point where we can obtain the maximum productivity.

5.5.3 Analysing Time Complexity

For each end of each parts, we have to do $O(n)$ times of calculation, once for each home city. As there are 36000 parts in total, the total time complexity would be $36000O(n) = O(n)$. Although this is still $O(n)$, we must be aware that this is with a very large constant, so it has a substantially poorer runtime performance than the algorithm in Section 5.3.

5.6 The Optimal Latitude

Now that we have already determined the optimal longitude where the host should be, the remaining problem becomes finding the latitude of the host such that the sum of flight distance from the participants to the host is minimised in order to reduce cost.

We have the following latitude-based algorithm.

5.6.1 Dividing the Meridian

Using the same technique as in Section 5.4.2, we can divide the meridian arc into 18000 parts, and hence find a rather precise position to hold the conference. Similarly, the total time complexity will thus be $18000O(n) = O(n)$.

5.6.2 Possible Improvement

We wish to find out a close form for the latitude so that the sum of flight distance is minimised; however, we do not have a satisfying answer.

As in Section 2, $\cos\alpha_i = \cos\theta\cos\theta_i\cos(\phi - \phi_i) + \sin\theta\sin\theta_i$, where θ_i is the latitude of the i-th city, and ϕ_i the longitude thereof.

Let $A_i = \cos\theta_i \cos(\phi - \phi_i)$ and $B_i = \sin\theta_i$,

Then $\cos\alpha_i = \sqrt{A_i^2 + B_i^2}\left(\frac{A_i}{\sqrt{A_i^2+B_i^2}} \cdot \cos\theta + \frac{B_i}{\sqrt{A_i^2+B_i^2}} \cdot \sin\theta\right)$.

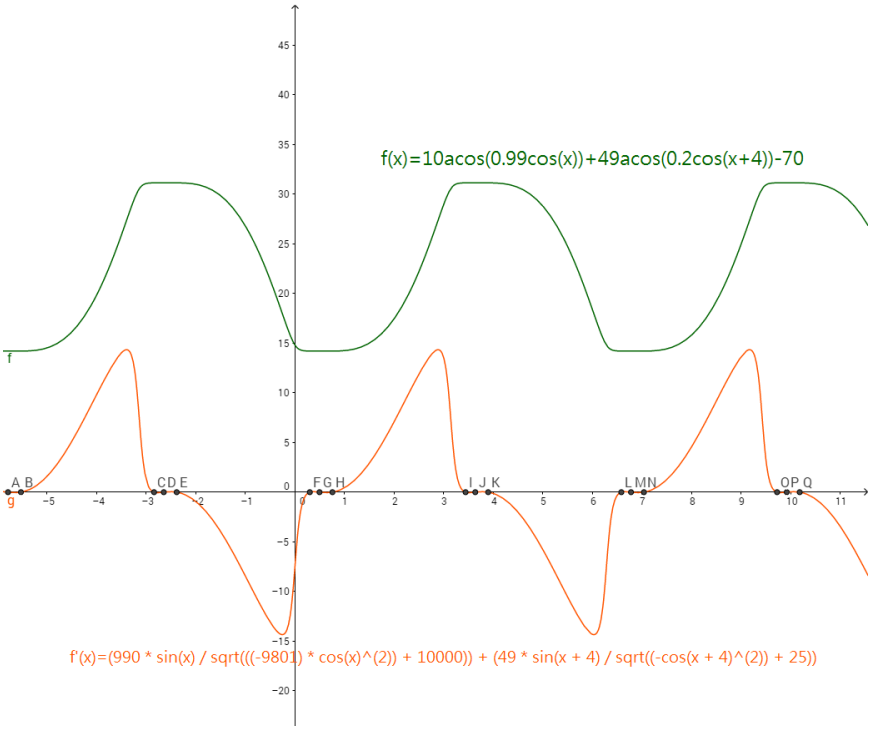
Let $\beta_i = \arcsin\left(\frac{\sin\theta_i}{R_i}\right)$ where $R_i = \sqrt{A_i^2 + B_i^2}$. Hence, $\cos\alpha_i = R_i \cdot \cos(\theta - \beta_i)$ $\alpha_i = \arccos(R_i \cdot \cos(\theta - \beta_i))$.

$$\therefore f(\theta) = \sum_i \alpha_i = \sum_i \arccos(R_i \cdot \cos(\theta - \beta_i))$$

We tried finding the extremum values of $f(\theta)$.

We found that the shape of the graph of some $f(\theta)$ against θ seems to be sinusoidal, especially when $R_i (i = 1,2,\dots,n)$ is small, which has only a peak per period. We tried to prove it as we can utilise this feature to binary search the answer if this observation is true.

Evidently, when $R = 1$, $\arccos(R \cdot \cos\theta) = \arccos\cos\theta$ is a "zig-zag" function, so when $R \to 1$, the shape of the graph will no longer be sinusoidal, and hence the shape of the graph of $f(\theta)$ will become strange when some of the Rs in $f(\theta)$ tend to 1. We tried to construct different $f(\theta)$ to see if the observation is false, and we succeeded.



[Diagram] Generated with GeoGebra.

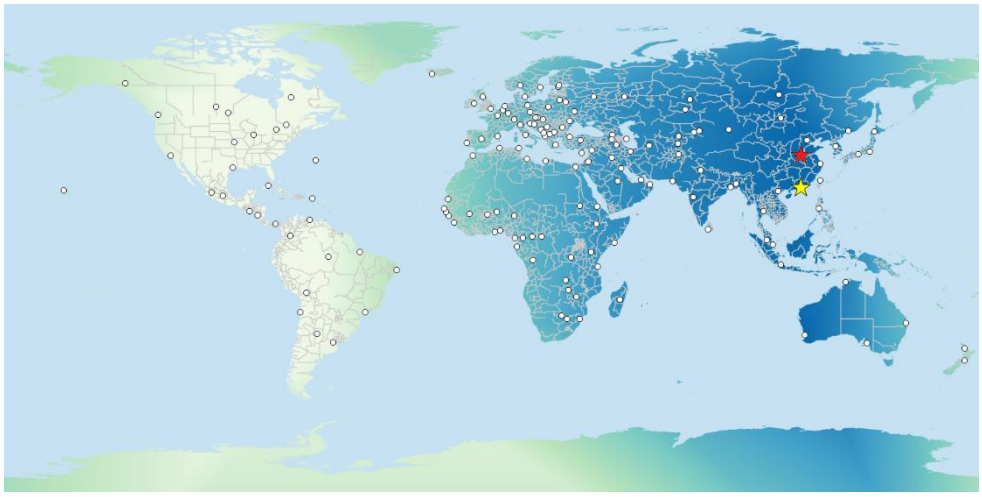
Obviously, this graph has 3 peaks in one period, so we can only perform the naïve yet very simple method to find the optimal latitude.

6 Case Studies

The following test cases are the ones given in the problem statement. The linear case assumption as in Section 5.4 applies throughout the following Section.

After thorough consideration, we would like to make an additional assumption that the host city should have an international airport or else the traffic time from the nearest international airport to the host city will significantly increase the transport time for every attendee, which is undesirable.

6.1 The Small Test Case



[Diagram] The colour on the map is the productivity level if we host the meeting in each candidate city. Blue means a higher productivity. Pale yellow means lower. Generated with qGIS.

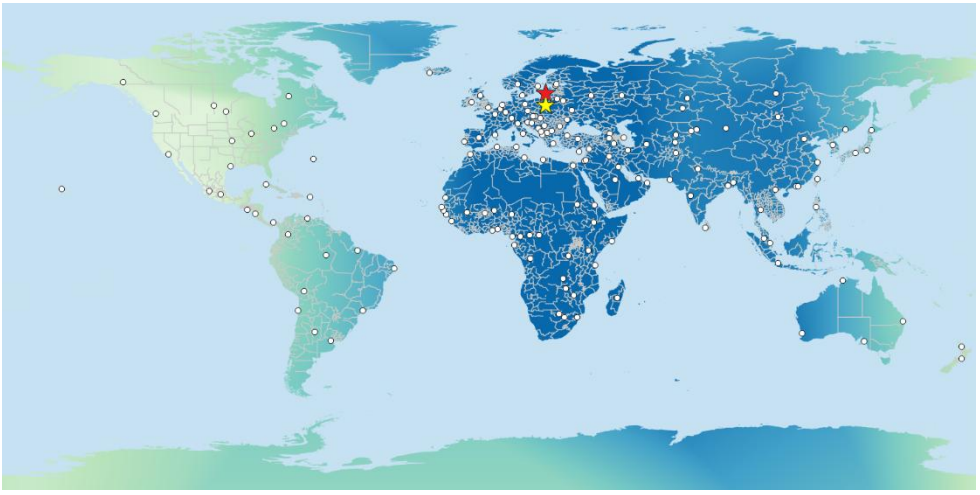
After gathering the longitude and latitude information of the cities where the participants are from, we input all the data into our program as shown in the appendix Section 8.1 and we get the optimal longitude and latitude where the host city should be.

The best time zone for the optimal productivity level is that of Hong Kong, at a longitude of 114.17°E.

To minimize the cost, the optimal latitude is 33.81°N according to the output of our program.

Then we input the longitude and latitude into Wolfram Alpha[7], we obtain that the nearest city with an international airport to that longitude and latitude is Zhengzhou China.

6.2 The Large Test Case



After collecting the longitude and latitude information of the cities where the attendees are from, we use the same technique as shown above and we find that the optimal latitude and longitude are 57.82°N and 21.01°E (the longitude of Warsaw), respectively. We find that the nearest city with an international airport should be Riga, the capital of Latvia by using Wolfram Alpha[8], again.

7 Further Discussion

7.1 Advantages of the Algorithm

- $O(n)$ time complexity leads to high efficiency.
- Statistical analysis eliminates factors such as temperature and sunshine duration.
- The algorithm always gives feasible outcomes as the host city we choose is limited to those that has an international airport for accessibility.

7.2 Disadvantages of the Algorithm

- We are unable to effectively combine different factors together.
- Our list of all candidate cities may not be comprehensive, so even if we can find the best longitude and latitude, we may not find the city nearest to that coordinate.

8 Appendix

8.1 Source Code in C++ as Implementation of Section 5

```
#include <iostream>
#include <map>
#include <vector>
#include <cmath>
#include <string>

#define INF 1632567029
#define eps 1e-8
using namespace std;
typedef double dbl;
#define pi (acos(0.0)*2)
struct City{
    string name;
    double lo, la;
    int state;
};

bool cmp(City a,City b) {
    if (a.lo<b.lo)
        return 1;
    else
        return 0;
}

dbl pt(dbl a,dbl b){
    dbl diff = b - a;
    if(abs(diff) < eps) return 1.0;
    if(diff >= 0 && diff <= 144.0) return 1.0 - diff/144.0;
    else if(diff >= 0 && diff > 144.0) return 1.0 - (360-diff)/(360-144);
    else if(diff < 0 && diff > (144 - 360)) return 1.0 + diff/(360-144);
    else return 1.0 - (360 + diff)/144;
}

dbl dist(dbl lat1, dbl lon1, dbl lat2, dbl lon2){
    dbl ans = 0.0;
    ans += cos(lat1 / 180.0 * pi)*cos(lat2 / 180.0 * pi)*cos((lon1 - lon2) / 180.0
* pi);
    ans += sin(lat1 / 180.0 * pi) * sin(lat2 / 180.0 * pi);
    return acos(ans);
}

int east(dbl a,dbl b){
    dbl diff = b - a;
    if(abs(diff) < eps) return 1;
    if(diff >= 0 && diff <= 144.0) return 1;
    else if(diff >= 0 && diff > 144.0) return 0;
    else if(diff < 0 && diff > (144 - 360)) return 0;
    else return 1;
}

dbl MOD(dbl a,dbl mod){
    while(a < 0){a += mod;}
    while(a > mod){a -= mod;}
    return a;
}

int main() {
    string s;
    dbl a,b,c,d;
    char x,y;
    vector<City> vc;

    //input,    name lat   '   N/S  lon   '   E/W
    while(cin >> s >> a >> b >> x >> c >> d >> y){
```

```
    dbl lat, lon;
    lat = a + b / 60.0;
    if(x == 'S') lat = -lat;
    lon = c + d / 60.0;
    if(y == 'W') lon = -lon;
    City bit;
    bit.name = s;
    bit.lo = MOD(lon + 180.0, 360.0);
    bit.la = lat;
    bit.state = 0;
    vc.push_back(bit);
}

//create critical point
int n = vc.size();
for(int i = 0;i < n;i++){
    City cy = vc[i];
    City bit;
    bit.name = "critical point of " + cy.name;
    bit.lo = MOD(cy.lo + 144.0, 360.0);
    bit.la = cy.la; //not in use
    bit.state = 1;
    vc.push_back(bit);
}

//sorting
sort(vc.begin(),vc.end(),cmp);

//O(n) cal productivity
dbl des_lo = vc[0].lo;
dbl ttpt = 0.0;
int set_a,set_b;
set_a = set_b = 0;
for(int i = 0;i < vc.size();i++){
    if(vc[i].state == 0){
        ttpt += pt(vc[i].lo,des_lo);
        if(east(vc[i].lo,des_lo)) set_a++;
        else set_b++;
    }
}

//O(1) transition & O(n) point
dbl maxpt = ttpt;
City hostcity = vc[0];
if(vc[0].state == 1) {set_b++;set_a--;}
for(int i = 1;i < vc.size();i++){
    City old = vc[i-1],now = vc[i];
    dbl diff = now.lo - old.lo;
    ttpt = ttpt - set_a * diff / 144.0 + set_b * diff / (360-144.0);
    if(maxpt < ttpt){
        maxpt = max(maxpt,ttpt);
        hostcity = now;
    }
    if(now.state == 0) {set_a++;set_b--;}
    else {set_b++;set_a--;}
}

//output
cout << "O(n) algorithm: " << endl << endl;
cout << "host city = " << hostcity.name << endl;
cout << "longitude = " << hostcity.lo - 180 << endl;
cout << "maximum productivity = " << maxpt << endl;

//O(n^2) compare
maxpt = -INF;
for(int j = 0;j < vc.size();j++){
    des_lo = vc[j].lo;
```

```
        ttpt = 0.0;
        for(int i = 0;i < vc.size();i++){
            if(vc[i].state == 0){
                ttpt += pt(vc[i].lo,des_lo);
            }
        }
        if(maxpt < ttpt){
            maxpt = ttpt;
            hostcity = vc[j];
        }
    }

    //output
    cout << endl;
    cout << "O(n^2) algorithm: " << endl << endl;
    cout << "host city = " << hostcity.name << endl;
    cout << "longitude = " << hostcity.lo - 180 << endl;
    cout << "maximum productivity = " << maxpt << endl;

    //determine the optimal lat
    dbl mindist = INF;
    const dbl lon = hostcity.lo;
    dbl bestlat = -INF ;
    for(dbl lat = -90.00 ; lat <= 90.00 ; lat += 0.01){
        dbl ttdist = 0.0;
        for(int i = 0;i < vc.size();i++){
            if(vc[i].state == 0){
                ttdist += dist(vc[i].la, vc[i].lo, lat, lon);
            }
        }
        if(mindist > ttdist){
            mindist = ttdist;
            bestlat = lat;
        }
    }
    if(abs(bestlat) < eps) bestlat = 0;
    cout << "latitude = " << bestlat << endl;

    //pause
    system("PAUSE");
    return 0;
}
```

8.2 Reference

[1] International Olympiad in Informatics database, http://stats.ioinformatics.org/

[2] Various pages on ClimaTemps.com, http://www.climatemps.com/

[3] Wikipedia, “List of cities by temperature”, https://en.wikipedia.org/wiki/List_of_cities_by_temperature

[4] Wikipedia, “List of cities by sunshine duration”, https://en.wikipedia.org/wiki/List_of_cities_by_sunshine_duration

[5] Wikipedia, “List of largest cities”, https://en.wikipedia.org/wiki/List_of_largest_cities

[6] USA Today, “How Long Does it Take to Get Over Jet Lag?”, http://traveltips.usatoday.com/long-over-jet-lag-63114.html

[7] Wolfram Alpha, Query “longitude 114.17 latitude 33.81”, https://www.wolframalpha.com/input/?i=longitude+114.17+latitude+33.81

[8] Wolfram Alpha, Query “longitude 21 latitude 57.82”, https://www.wolframalpha.com/input/?i=longitude+21+latitude+57.82