Algorithms Seminar
Airline Seat Swap

Notes / Edited by: S Munson

August 31, 2012 - Sept 07, 2012

Introduction / Motivation

M Bender brought in an idea from his summer journeys by plane. Travelling alone, he'd been asked by the flight crew to swap seats with another passenger so that a couple could sit together. He got to wondering if this process could be optimized ...

So in our conjecture, Stony Brook Air, like many airlines, prides itself in high-quality customer service. In order to improve the flying experience, they have begun a new service, allowing their flight staff to help arrange people once boarded on a flight. The flight crew will then help to arrange persons who are in couples or families or groups to sit together, allowing them to be 'happier' during the flight.

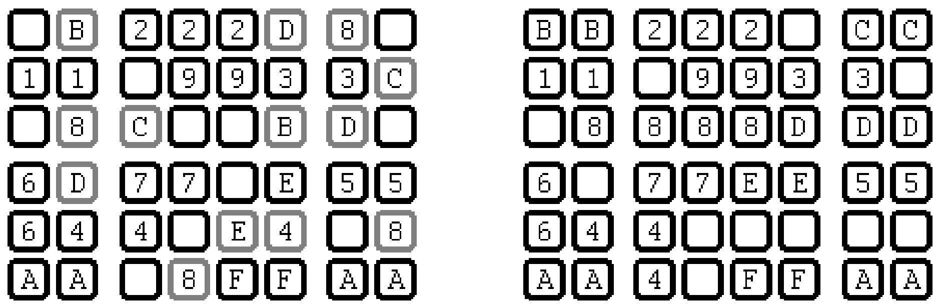

Figure 1: an initial arrangement, and a happier one (gray units are unhappy)

Q: By what process can the flight crew achieve the maximum level of happiness in the section while still disturbing the minimum number of seated people?

In the General case we also will want to preserve certain seat qualities (aisle, window, etc). This may make the problem significantly harder to solve.

Assumptions: We assume that the initial distribution of fliers is arbitrary. We also limit our problem by stating that the solution must be solved in-place once fliers are boarded. Singletons people are always 'happy' sitting next to anyone. Members of a group are Happier when surrounded by their group members.

Initial week Analysis

Some proposals for initial analysis modes were put forward ; S Skiena suggested colouring groups and then seeking maximum color adjacency, or a graph structure with edges between members of the same group. J Mitchell proposed simplifying the problem first to a one-dimensional model of the problem, with only couples on board (maybe a plane full of honeymooners headed to a post-nuptial resort?).

We also restrict ourselves to a simple two-person swap. Now we have a clearly defined problem with strict limits on choices of actions. It also creates a sorting / Matching class of problem which we can attack with well-known ideas.

E Arkin asked for clarification whether we were seeking to minimize the number of swaps or the total swap distance. J Mitchell suggested that either could be sought as this was an optimal path type of problem. S Skiena commented that he felt we should think about disturbing the fewest number of persons.

J Zuber mentioned that this initially seemed like with complete unhappiness (a worst-case Scenario), this could be solved in $\frac{n}{2}$ operations.

M Biro commented that this bound is tight.

After some discussion, E Arkin proposed the following sweeping-line algorithm as an initial attempt : algorithmic

```
while $i \leq n-2$ do Read a pair at a time $(i, i+1)$
    if $group(i) \neq group(i+1)$ then
get the $jth$ element from the right with $group(j) = group(i)$
swap the $jth$ and $i+1th$ element
    end if
end while
```

For example :

1 3 1 2 3 2 Initial condition
1 1 3 2 3 2 swap position 2 with position 3
1 1 3 3 2 2 swap position 4 with position 5

J Zuber believes that this is optimal, and offers the following thumbnail proof:

Proof: Any swap that is more optimal will find two pairs to satisfy at once. Since the left-right sweep will find them at the same time, no other search can be more optimal. Additionally, if the pairs are odd-aligned, no single swap can satisfy both pairs at once.

J Yan points out the that last swap will satisfy both remaining unsatisfied pairs simultaneously, and therefore a slightly tighter bound is $\frac{n}{2} - 1$ is a better supper bound.

Discussion then started using the terms 0-swap to describe a swap that satisfies no pairs, a 1-swap to be a swap that satisfies one pair, and a 2-swap that satisfies two pairs at once. S Skiena then threw out the question of whether we could have a sitauation where two 0-swaps could then create the scenario where 3 2-swaps could happen, giving a more efficient algorithm.

L Walsh pointed out that each 0-swap could never make any more than 2 2-swaps possible.

Pairs with singletons we then moved on to the 1-dimensional model with the addition of possible single persons flying alone. These people are considered always happy whomever they sit next to.

E Arkin challenged seminar members to find a counterexample demonstrating that the previous algorithm would be weaker, and Paul Fodor brought forth the following example:
1 2 3 3 4 2

It was pointed out that since 1 is unmatched, the algorithm would break as written. If it were amended to 'skip' singletons, then the algorithm would work out as :
1 2 3 3 4 2 Initial conditions
1 2 2 3 4 3 swap positions 3 and 6
1 2 2 3 3 4 swap positions 5 and 6

and clearly a more optimal solution is to simply swap positions 1 and 6.

S Skiena pointed out that the key here would be to detect 2-swaps consistently.

A Ban pointed out that the center element in a triplet of singletons would never be moved.

S Skiena Pointed out that there would never be a circumstance where both members of a pair were moved.

There was general discussion after this point but no more significant progress was made on the problem, and the meeting was closed with encouragement of the students to work more on this project, as it seemed to M Bender that this could result in a publishable work.

Post-session material

Hardness commentary for large k-families. Professor S Skiena introduced the following material between sessions suggesting that the problem is hard for k-sized 'families' when the value of k is 'large'

The problem we discussed in reading group yesterday is hard for large families (i.e. not just limited to singletons and couples).

The reduction is from 3-Partition: given a set of $3m$ integers partition them into m groups of three such that the sum of each group is equal. This is strongly NP-complete, so it is still hard if the numbers can be written in unary.

Say the input instance has $n = 3m$ integers whose total sum is $m \times k$, so each group has to add up to exactly k.

We will reduce the 3-partition instance into an airline reseating instance with $3m + m + 1$ families (call them colors). Each input integer will be represented by a distinct color, with a number of elements equal to its size. The total is $m \times k$ "integer" elements. We will also have m distinct "spacer" colors (a different shade of black), each with a large family ($X > m \times k$ elements each). Finally, we have exactly $m \times k$ "red" elements, the same as the total of the input integers.

These are initially arranged as:
k reds, X black(1)s, k reds, X black(2)s ... k reds, X black(m)s, highly scrambled mix of the $m\ timesk$ elements.
To reseat the elements, the reds have to be brought all together and the integer elements sorted. The only place to reseat the reds without moving blacks (which is costly) is to put them in the $m * k$ slots on the plane, where the "integer" elements sit. The only way everyone can be made happy in $n \times k$ moves is if the integer elements can be relocated where the reds are with a 3-Partition solution.

Ed Note: In the interest of the pedagogical mission of the seminar, we present the definition of 3-Partition [SP15] as in Pg. 224, "Computers and Intractability" by Garey / Johnson, which is presently call number QA.76.6.G35 c.2 in the Stony Brook library, or ISBN # 0-7167-1044-7.

Instance: set A of $3m$ Elements, a bound of $B \in Z^+$, and a size $s(a) \in Z^+$ for each $a \in A$ such that $B/4 < s(a) < B/2$ and such that $\sum_{a \in A} s(a) = mB$.

Question: Can A be partitioned into m disjoint sets $A_1, A_2, A_3, ...A_m$ such that for $1 \leq i \leq m$, $\sum_{a \in A_i} s(a) = B$ (note that each A_i must therefore contain exactly three elements from A) ?

Chao Xu's formalization

Undergraduate C Xu submitted this formalization between weeks, and it was examined more closely in session.

Formally : Consider the sequence $a_0, \ldots, a_{n-1}$. $a_i = a_j$, $i \neq j$ implies people seating on ith and jth seats are a couple.

Let $v_i = \{a_{2i}, a_{2i+1}, \{i\}\}$. Let $V = \{v_0, \ldots, v_{n/2-1}\}$. We build a graph $G = (V, E)$, such that $\{v_i, v_j\} \in E$ iff $v_i \cap v_j \neq \emptyset$.

Intuitively, v_i represents the $2i$th and $2i+1$th seat. The edge represent there exist a couple shared in between the seats.

Claim 1 : The resulting graph is a disjoint set of cycles, K_2 and isolated vertices.

First show each vertex has at most degree 2, then show a vertex have degree 1 only if it is in a K_2, and a vertex have degree 0 iff it's an isolated vertex. The remaining vertices must form disjoint set of cycles.

Claim 2: Every swap operation can change the number of components of G by at most 1.

Notice each swap operation on the array can be reflected as removal of 2 edges, then adding 2 edges back to the graph. Except the special cases where we have K_2 and isolated vertices, but if we consider the problem as a multigraph instead, then the K_2 become two vertices sharing 2 edges, and isolated vertices are vertices with two self loops.

By checking a few case work, one see this operation together with the restriction in Claim 1 implies Claim 2.

Claim 3: If G has k components, the minimum number of swaps is $n/2 - k$.
This is because every couple sits together iff G has $n/2$ components. Using Claim 3, we can show any algorithm that increase the number of components in G by one is an optimal algorithm.

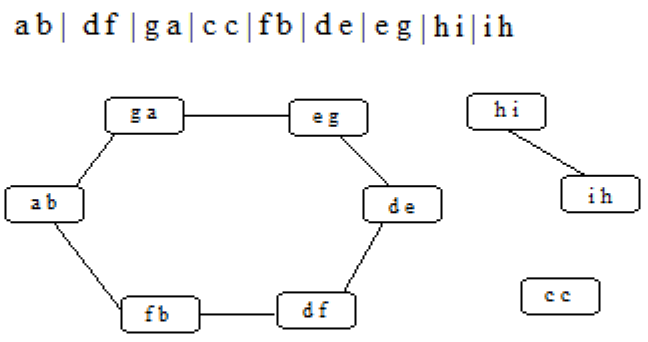

Figure 2: a seat arrangement, and its graph

Week two notes

There was a discussion of C Xu's formalization process, which led to several notes:

It was noted that elements that seem to be K2 are situations where the edge is constructed twice. In a setting like a multi-graph, there would be two edges between the two nodes, and it would be a special case of a cycle of size two.

There was a Presentation by R Chowdhury of S Skiena's Hardness proof. There was then a discussion of other work to be followed up on this discussion, and What constituted 'interesting' questions. The Following topics were suggested as avenues to explore:

Singletons + Families If we allow a number of people flying alone, how does this change the problem?

Mixed size How does this problem change with families of mixed sizes?

Fixed / variable size if the families are of the same size, does this problem get easier or harder?

Bounded size What happens to the complexity of the problem when families are of a bounded size (1,2,3,... b-1, b)?

Minimum Distance travelled can we find the smallest amount of total motion needed?

Parallel Swaps If we can allow multiple pairs of passengers to swap at the same time (assuming that they don't need to cross paths) will this effect the optimal algorithm? how?

Minimize the Maximum How do we minimize the maximum amount of movement for any one person. Can we guarantee that each person will have to move no more than one time?

Seat Flavours If seats are not interchangeable, but have a 'flavour' (aisle, window, bulkhead, exit row, near bathroom, 1st class) what is the effect? If we allow some fractional 'unhappiness' by people who are in their group, but in the wrong flavour of seat, does this change the problem substantially? How?

Number of algorithms Are there a fixed number of optimal algorithms?

Aversion imagine some persons have preferences that they sit a certain number of seats from a person who is a a particular type (like a crying baby) ?

Post-session material

Michael Biro's comments on 1-swap limit

M Biro submitted the following proof of a modification of the existing sweeping algorithm for families of size 2 (no singletons).

Claim: The optimal algorithm for families all of size 2 may be modified to move each person at most once.

Proof. Take the graph defined in the proof of correctness (vertices correspond to the $\frac{n}{2}$ paired spaces, and edges between two vertices if they share a number). Singleton vertices are already done, and length 1 paths can be done with one swap, so without loss of generality assume the graph is made of cycles of length at least 3. We will never swap between cycles, and so we just show a protocol to handle each cycle independently, while moving each person at most once. The protocol is as follows:

1. If this is the first swap in the cycle, no member has been moved. Pick any member and swap so that that member is satisfied. This increases the number of connected components in the graph, so it is part of an optimal swapping schedule, now there is exactly one moved member in the cycle, and the vertices in the cycle have been reduced by one.

2. If this is not the first swap in the cycle, there is exactly one moved member. Find its unmoved pair and swap so that both are satisfied. We move a new member, but the old member that was moved is now satisfied and out of the cycle, so there remains exactly one moved member. This move is part of an optimal swapping schedule and the cycle is now smaller by one vertex. Repeat until all members of the cycle are satisfied.

Michael Biro's comments on k-family problem

Claim: The sweep algorithm described last week satisfies everyone in a k-family problem with n seats in at most $(k-1)*(n/k-1)$ swaps.

Proof: Sweep over the n/k groups of k spots from left to right. Take the first number in the group and swap at most $(k-1)$ of its partners into that group to satisfy them all. Do this for each of the n/k groups except the last, which is satisfied by default if the preceding groups are. This comes out to at most $(k-1)(n/k-1)$ swaps.

Claim: The k-family problem with n seats sometimes requires at least $\frac{n}{k}-1$ swaps to satisfy everyone (for arbitrary n and k).

Proof: Take the 1223344...nn1 example with $2n$ numbers that shows $n-1$ required swaps for $k=2$, and split into pairs 12|23|34|...|n1. Then insert $k-2$ copies of the first number in each pair after the second in each pair, so it becomes 1211..1—2322..2—...—n1nn..n. This is now a k-family problem, with $2n+(k-2)n=nk$ numbers. Here, the optimal solution is (I think, this requires some extra arguments) to swap what is essentially the original $k=2$ solution, for $n-1$ swaps. This works out to $\frac{nk}{k}-1$, as claimed.

Claim: There are k = 3 family situations that require greater than $\frac{n}{3} - 1$ swaps but fewer than $\frac{2n}{3} - 2$ swaps, meaning the above bounds aren't yet tight.

For small n and $k = 3$, 123 123 123 requires 3 swaps, which is $\frac{n}{3}$, not $\frac{n}{3} - 1$.

I believe 123 145 167 246 257 347 356 requires at least 9 swaps for $n = 21$ numbers, which is worse than $\frac{n}{3} - 1 = 6$, but better than $\frac{2n}{3} - 2 = 12$. No proof yet.

Claim: Any instance of the 3-family problem on n seats that requires more than $\lfloor \frac{n}{2} \rfloor - 1$ swaps has $n \geq 21$.

Proof: Split into triples of seats, and build the graph G with vertices representing triples, with an edge between two vertices if they share a number. If any triple has a number majority (2 family members in the same triple), swap the last remaining number, and charge the swap to that triple (1 swap to satisfy 3 people, gives at most $\frac{n}{3} - 1$). Otherwise, if two triples share at least two numbers between them, with one swap we can create two triples with a number majority in each, then satisfy both triples with a total of at most 3 swaps to satisfy 6 people, giving a max of $\lfloor \frac{n}{2} \rfloor - 1$ swaps. So, assume that no triple has a number majority, and no pair of triples shares two numbers. Then, the vertex corresponding to each unsatisfied triple has degree exactly 6, and so the number of triples must be at least 7. Therefore $n \geq 21$. The above, 123 145 167 246 257 347 356, is 21 seats, gives $\lfloor \frac{21}{2} \rfloor - 1 = 9$ swaps (I think).