A Greedy Algorithm for Team Formation that is Fair over Time

Maarten Fokkinga

Database Group, CTIT, University of Twente, The Netherlands

Version of April 1, 2010, 9:03

ABSTRACT

In terms of a concrete example we derive a greedy algorithm for a hard problem. The problem is to compute a value which minimizes a given expression, and has exponential time complexity. The greedy algorithm computes a sub-optimal solution but has polynomial time complexity, and is fast enough for practical applications. The concrete problem is: the formation of teams from a given set of players such that, when repeated many times, each player is equally often teammate of each other player. The algorithm is easily coded in about thirty lines in a functional programming language. A few simple experiments give an indication of the quality of the greedy algorithm, and of some variants which trade quality for speed. We also abstract from the particulars of the concrete problem, and formalize our greedy algorithm in a general and abstract setting.

1. PROBLEM STATEMENT

Consider the following scenario. A fixed set of players is known: *Player*. Several times in history —for example "on a weekly basis"— so-called *teams* have been formed, each team being a group of players. (Teams are nonempty and mutually disjoint.) We want to have an algorithm A that produces yet another team formation — for example "for the coming week". The input of algorithm A is the "current" historical data about past team formations (preferably in some aggregated form such as *counts*), the desired team size, and the subset of players that want to play. Algorithm A must have the following so-called *fairness* property:

> When repeated over and over again, then in the end each player is, *as fair as possible*, equally often teammate of each other player.

When the fairness property is not fulfilled, players might in the long run complain that they have been placed too often together in one team, or that the team formation is biassed.

> **Footnote.** This problem is actual in the recreational golf sport. During a season of about forty weeks, at each week the participating players, about fifty, are grouped into teams of size three, and at some weeks the team size is four. (The teams simultaneously play one game, with scores per team or per individual player.) In the team formation at my club there is no automated support for fairness, and, indeed, there have been complaints by the players as expressed above.

We shall now first formalize the requirement in §2, then design the algorithm in §4 with several variants in §5 for a trade-off between efficiency of the computation and quality of the result (as measured by the formal requirement), code the algorithm in a functional programming style in §6, and conclude with a modest experimental comparison between the quality of the algorithm and its variants in §7. The *structure of the algorithm* is for a large part independent of the specific requirement, and has a wider applicability. It is known as a form of *greedy* algorithm:

> A greedy algorithm is any algorithm that follows the problem solving metaheuristic of making the locally optimal choice at each stage with the hope of finding the global optimum. *Wikipedia*, http://en.wikipedia.org/w/index.php?title=Greedy_algorithm&oldid=338460107.

In §8 we give *our* greedy algorithm in an abstract form.

2. REQUIREMENT FORMALIZATION

Let us try and formalize the fairness requirement. Suppose we have two algorithms, A_1 and A_2 that, after a sufficiently large number of invocations, transform the current state into future states; of course, the algorithms run for the same number of invocations and, for each invocation, they get as input the same desired team size and the same subset of players. What property makes us designate A_1 as "better" than A_2? Clearly, we must look at the final states and find that the state resulting from A_1 is "better" than the one resulting from A_2. We have two alternatives:

- To measure in a state how "equally often" a player p has been teammate of the other players, we take the standard deviation of the function $\lambda\, q{:}\, Player' \bullet count(p, q)$, where $count(p, q)$ is the number of times player p, q have been teammates in the history so far. Player p has to be excluded from the function's domain since "$count(p, p)$" makes no sense; so we put $Player' = Player \setminus \{p\}$. For example, let $Player = \{0, 1, \ldots, 8\}$ and $p = 0$ and consider the following counts:

q =	1	2	3	4	5	6	7	8	std dev
$count_1(0, q)$ =	2	2	3	2	3	3	2	2	0.4 . . .
$count_2(0, q)$ =	9	0	0	2	0	0	1	7	3.5 . . .

  In the histories of both $count_1$ and $count_2$, player 0 has 19 times been teammate of some other player, but the standard deviation of $count_1(0, \_)$ is smaller than that of $count_2(0, \_)$. If all other things are equal, we declare $count_1$ better than $count_2$.

- However, equality in absolute counts might be called 'not *as fair as possible*'. It may be more fair to look at the frequency: the number of times p, q have been teammates

divided by the number of times that p, q could have been teammates (i.e., were present in the same team formation in possibly different teams). We call this number $\mathit{freq}(p, q)$. For example, let $\mathit{Player} = \{0, 1, \ldots, 8\}$ and $p = 0$ and consider the following frequencies:

q =	1	2	3	4	5	6	7	8	std dev
$\mathit{freq}_1(0, q)$ =	$\frac{4}{5}$	$\frac{4}{5}$	$\frac{4}{5}$	$\frac{4}{5}$	$\frac{2}{3}$	$\frac{2}{3}$	$\frac{2}{3}$	$\frac{2}{3}$	0.06...
$\mathit{freq}_2(0, q)$ =	$\frac{3}{5}$	$\frac{3}{5}$	$\frac{3}{5}$	$\frac{3}{5}$	$\frac{3}{3}$	$\frac{3}{3}$	$\frac{3}{3}$	$\frac{3}{3}$	0.2

The notation 4/5 in column q means that player 0 and q have been teammates 4 times while they have been participating in the same team formation 5 times, possibly in different teams; so $\mathit{freq}(0, q) = 4/5 = 0.8 = 80\%$. In the histories of both freq_1 and freq_2, player 0 has 24 times been teammate of some other player, but the standard deviation of $\mathit{freq}_1(0, \_)$ is smaller than that of $\mathit{freq}_2(0, \_)$. If all other things are equal, we declare freq_1 better than freq_2. (Note that regarding counts the second one is better than the first one since, in that case, all counts are 3 and so the standard deviation is 0.) Thus we take as measure the standard deviation of the function: $\lambda\, q\colon \mathit{Player}' \bullet \mathit{freq}(p, q)$. If some player q has not shown up at all, in the entire history, then it doesn't make sense to consider player q; indeed, $\mathit{freq}(p, q)$ will be 0/0, and player q has to be excluded from consideration. So we put $\mathit{Player}' = \{q\colon \mathit{Player} \mid q \neq p \wedge \mathit{freq}(p, q) = \text{`.../0'}\}$.

In order to know which player to exclude and to avoid division by zero, we choose to let $\mathit{freq}(p, q)$ be a pair (a, b), and transform this to the fraction a/b tacitly when the need arises.

Of the two alternatives above, we will adopt the second one in the sequel. The reader may notice, however, that the sequel is almost *completely* independent from this particular choice. The only reason to make the choice explicit is to be less abstract and more concrete (easier to understand).

We have still to solve the following question:

How should the standard deviations for *individual* players $p, p', p'', \ldots$ contribute to the property that *entire* freq_1 is "better than" *entire* freq_2?

There are two options:

- One option is that *for all* players p, without exception, it is required that 'its standard deviation in freq_1' is smaller than 'that in freq_2'. However, then the relation "better than" is not total but partial (that is, not defined for all frequency functions), and very partial indeed. This formalization makes the problem unsolvable and the algorithm non-existent: too often it would be impossible to achieve the situation that "each player is, as fair as possible, equally often teammate of each other player".

- The other option is to somehow *aggregate*, over all p, the standard deviations of p in freq_1, and take that outcome as freq_1's measure for "better than". This makes "better than" a total pre-order (not an order since different frequencies may be "equally good"), and thus the problem solvable. Aggregation in the form of summation seems most reasonable (and taking the maximum seems second best), and in order not to depend on the number of players, we divide the sum by that number, thus effectively specializing aggregation to averaging.

Thus we arrive at the following definitions:

$$\begin{array}{lcl} \mathit{sdev}(p, \mathit{freq}) & = & \text{std dev of } \lambda\, q\colon \mathit{Player}' \bullet \mathit{freq}(p, q) \\ & & \text{where} \\ & & \mathit{Player}' = \{q\colon \mathit{Player} \mid q \neq p \wedge \mathit{freq}(p, q) = \text{`.../0'}\} \\ \mathit{avg\_sdev}(\mathit{freq}) & = & \frac{1}{\#\mathit{Player}} \sum_{p:\mathit{Player}} \mathit{sdev}(p, \mathit{freq}) \\ \mathit{freq}_1 \preceq \mathit{freq}_2 & \equiv & \mathit{avg\_sdev}(\mathit{freq}_1) \leq \mathit{avg\_sdev}(\mathit{freq}_2) \end{array}$$

Now the sub-phrase "each player is, *as fair as possible*, equally often teammate of each other player" is interpreted as "the resulting frequency is minimal with respect to $\preceq$".

There is one other sub-phrase in the informal requirement that needs attention, namely "when repeated over and over again, then in the end. . . ". Since in practice it is not known in advance which players will participate in which weeks and, maybe, how many weeks the season will take, we propose to replace the sub-phrase "in the end" by "after *each* invocation of the algorithm". (This change is a *weakening* of the original requirement, since an algorithm satisfying the original requirement —assumed it makes sense somehow— also satisfies the new one.) Thus, with as input a frequency info freq, a desired team size n, and a player set P, the required algorithm A has to produce the following outcome $A(\mathit{freq}, n, P)$:

$A(\mathit{freq}, n, P) =$
some team formation T for size n of P such that
for each other team formation T' for size n of P:
"freq updated with T" $\preceq$ "freq updated with T'"

The formalization of the concepts in this requirement is straightforward:

- T is a *team formation* for size n of P, denoted $T \in \mathit{TF}(n, P)$, if: T is a partitioning of P such that all parts have size in $1\,..\,n$ and at most one part has size smaller than n (their sizes together sum up to $\#P$ which need not be a multiple of n).

- freq updated with T, denoted $\mathit{freq} \oplus T$, is: the function that maps (p, q) to
$(a + \langle\!\langle p, q \text{ are teammates in } T \rangle\!\rangle,\ b + \langle\!\langle p, q \text{ are in } T \rangle\!\rangle)$
where 'a/b' $= \mathit{freq}(p, q)$

Here, the notation $\langle\!\langle \mathit{condition} \rangle\!\rangle$ denotes 1 if *condition* holds, and 0 otherwise.

Thanks to the way $\preceq$ has been defined in terms of the $\leq$-relation on the $\mathit{avg\_sdev}$ values, algorithm A can now be specified in one line:

$$\boxed{A(\mathit{freq}, n, P) \ \in\ \arg\min_{T:\,\mathit{TF}(n,P)}\ \mathit{avg\_sdev}(\mathit{freq} \oplus T)}$$

Indeed, $A(\mathit{freq}, n, P)$ has to be some T from $\mathit{TF}(n, P)$ for which $\mathit{avg\_sdev}(\mathit{freq} \oplus T)$ is minimal. Operation arg min is a well-known mathematical concept, and may be defined thus:

$$\arg\min_{x:X}\ f\, x \ \ = \ \ \{x\colon X \mid (\forall\, y\colon X \bullet f\, x \leq f\, y)\}$$

Since all ingredients of the specification are computable, the above specification is executable as well. However, $\mathit{TF}(n, P)$ has more than $\frac{\#P}{n}!$ members, so that, as an algorithm, the specification is far too inefficient for all practical purposes.

3. BACKTRACKING

A computation of $A(\mathit{freq}, n, P)$ might be done by a systematic search over all totally and partially completed team formations from $TF(n, P)$, meanwhile computing or approximating $\mathit{avg\_sdev}(\mathit{freq} \oplus T)$, and comparing these with other such values. This is *backtracking*: simple to formulate but still time-consuming to execute. Too inefficient for the practical applications that we have in mind (about fifty players, team size of three or four). So we propose a greedy algorithm in the next section.

4. GREEDY ALGORITHM

We adopt a *greedy* approach: we are willing to *weaken* the requirement and we strive for an algorithm A' in which the team formation is built gradually in the following way:

- There is *no backtracking*: at each stage in the formation, each two players that are teammates remain teammates in all subsequent stages.
- Each decision to let two players be teammates, is *locally optimal*.

More precisely, we propose and accept that algorithm A' with input freq, n, P has the following *structure*; it successively computes team formations $T_0, T_1, \ldots, T_N$ of P, and then delivers T_N, where the following holds:

- In each T_i, the team sizes are at most n (and may differ from each other).
- T_0 is the team formation in which each team consists of exactly one player.
- Each T_i *contracts* to T_{i+1}, meaning that T_{i+1} results from T_i by uniting two or more teams and not touching all other teams.
- T_N has at most one team whose size is smaller than n.
- Each T_i is *finalizable*, which is explained, motivated, and defined below.

The execution of this greedy algorithm A' costs at most $\frac{\#P}{n}$ contraction steps, which compares favorably with the more than $\frac{\#P}{n}!$ comparison steps in the brute-force realization of A. The downside, however, is that the formal specification of A will not hold of A'. Even worse, we cannot predict precisely what property will hold for the outcome of A'. The only "claim" we can make is that we shall use the specification of A as a guide in the contraction steps, hoping that then the outcome T of A' will yield a reasonably small value for $\mathit{avg\_sdev}(\mathit{freq} \oplus T)$.

Before elaborating the contraction step, there is one more concern: proper termination or, in other words, the possibility to do a contraction step as long as the final formation has not been achieved. We call this property *finalizability* of a team formation. The following example shows a formation that is not finalizable:

> Take $n = 3$ and let all teams in T have size 2; then T is not final but nevertheless each contraction of T has a team whose size exceeds 3 and thus violates the team size constraint ("all team sizes at most n") and consequently cannot be taken as the successor of T in the algorithm. If this T is one of the T_i in the algorithm, then a successor T_{i+1} does not exist and the algorithm cannot deliver a result.

We shall now first elaborate the finalizability condition and then the contraction step; having done so, the greedy algorithm has been defined completely.

Finalizability

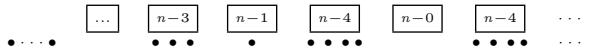
Let us first introduce some terminology:

t is *complete*	$\equiv$	$\mathit{size}\, t = n$
T is *final*	$\equiv$	at most one team in T is incomplete

Now, consider a team formation. What property of the formation guarantees that we can contract it eventually to a final team formation? The weakest such property seems very hard to formulate and, worse, costly to check: somehow all possible future contractions have to be considered in order to see whether progress can still be made; exponential time complexity is lurking again [I've not elaborated and analyzed this in detail]. So we take a crude approach and are satisfied with a sufficient condition (stronger than necessary):

> *Finalizability*:
> There are enough singleton teams to complete all-but-one non-singleton teams.

Here is an illustration of a typical finalizable team formation:



The top line lists each non-singleton team as a box with its size made explicit; the bottom line lists, in the form of bullets, the singleton teams that complete the teams above them. In the top line the first box represents the team that is possibly not completed by the singleton teams. In the bottom line the first group of singleton teams are those that are *more* than sufficient to complete all-but-one non-singleton teams; this group is empty iff "there are *precisely just enough* ...".

For example:

> Take $n = 5$ and let the teams in T have sizes $3, 2, 1$. Then T *can* be further contracted to a final formation with teams of size 3+2 and 1, but the condition is not fulfilled: the single singleton team is not enough to complete all-but-one non-singleton teams. If the teams in T have sizes $3, 2, 1, 1$, then the two singleton teams are enough contract T to a final formation with teams of sizes 3+1+1 and 2.

Because the finalizability condition is so strong, it happens to be sufficient as well:

> Theorem. Each contraction of two teams (with result size at most n) preserves finalizability except when there are *precisely just enough* singleton teams to complete all-but-one non-singleton teams, and the non-completed team is uniquely determined and is one of the two contraction teams, and the other contraction team is a singleton.
>
> > **Proof sketch.** Consider the typical finalizable team formation illustrated above. By case distinction it is easy to *visually see* the truth of the claims of the theorem; the cases are: uniting two non-first teams from the top line, one non-first team from the top line and one singleton below it, etc. The exceptional case is: the first group of singletons is empty and the teams to be united are the first team of the top line and another singleton team.

So, finalizability guarantees that a contraction *can* be done to a finalizable formation, and so, by induction, that a final formation *can* be reached. Finalizability is easy to formalize:

$$\mathit{finalizable}(T) \quad \equiv \quad \#T_1 \geq \textstyle\sum_{t:\, T_3}(n - \mathit{size}\, t)$$

where

$$\begin{array}{lcl} T_1 & = & \{t{:}\ T \mid 1 = \mathit{size}\, t\} \quad \text{— the singleton teams} \\ T_2 & = & \{t{:}\ T \mid 1 < \mathit{size}\, t\} \quad \text{— the non-singleton teams} \\ T_3 & = & T_2 \quad \text{if } T_2 = \varnothing \text{ else } \quad T_2 \setminus \{t_0\} \\ t_0 & \in & \arg\min_{t:\ T_2}\ \mathit{size}\, t \end{array}$$

Team t_0 stands for the exception that may be left incomplete; it is well-defined iff $T_2 \neq \varnothing$.

The contraction step

First we define $T[t, \ldots, t']$ to mean: the contraction of T by uniting just $t, \ldots, t'$ into one team, and leaving all other teams untouched.

Consider a non-final stage in the greedy algorithm, that is, a team formation T_i which is non-final, finalizable, and has only teams of size at most n. Our task is to define a contraction of T_i to some T_{i+1} that, again, has teams of size at most n and is finalizable. Moreover, hoping for a good result in the end, we must use the specification of A as a guide. So, a suitable definition is:

> T_{i+1} = some T of S that minimizes $\mathit{avg\_sdev}(\mathit{freq} \oplus T)$
>
> where
>
> S is the set that consists of formations $T_i[t, \ldots, t']$ for each $t, \ldots, t'$ such that (i) $\mathit{size}\, t + \cdots + \mathit{size}\, t' \leq n$, (ii) $T_i[t, \ldots, t']$ is finalizable.

How large is set S? Too large; exponential time complexity is lurking again. [There are $\sum_{k:\, 2..\#T_i-1}\binom{\#T_i}{k}$ contractions of T_i.] So we severely restrict S to a much smaller set by restricting $t, \ldots, t'$ to just two teams:

> S consists of formations $T_i[t, t']$ for each t, t' such that (i) $\mathit{size}\, t + \mathit{size}\, t' \leq n$, (ii) $T_i[t, t']$ is finalizable.

How large is set S? The number of choices for each of t and t' is bounded by $\#T_i$, which is at most $\#P$, and, in the greedy algorithm, this contraction step has to be done at most $\frac{\#P}{n}$ times. Although the number of steps has an acceptable order of magnitude, $(\#P)^3/n$, the question arises: can we do more efficiently while still achieving an acceptably small value of $\mathit{avg\_sdev}(\mathit{freq} \oplus T_i[t, t'])$? We eliminate one factor $\#P$ in the time complexity by severely restricting the choices for t, leaving an extensive search for t' only:

> S consists of formations $T_i[t, t']$ for each t, t' such that (i) t is promising, (ii) $\mathit{size}\, t + \mathit{size}\, t' \leq n$, (iii) $T_i[t, t']$ is finalizable.
>
> Team t is *promising* if: (a) t is incomplete, (b) t is the team of some player p for which $\mathit{sdev}(p, \mathit{freq} \oplus T_i)$ is maximal.

Condition (a) is necessary for (ii). Condition (b) leaves hardly any choice for t (it almost uniquely determines t) and is intended to make possible a small value of $\mathit{avg\_sdev}$ $(\mathit{freq} \oplus T_i[t, t'])$, by a suitable choice for t': in order to decrease the *average* standard deviation, it seems promising to start with the player that has a *maximal* standard deviation, and find for this one a team or teammate that decreases the deviation as much as possible.

This leads us to define $T_{i+1} = T_i[t, t']$ where:

- t = the team in T_i of some player p whose team is incomplete and which has maximal $\mathit{sdev}(p, \mathit{freq} \oplus T_i)$.
- t' = a team in T_i of size at most $n - \mathit{size}\, t$ such that $T_i[t, t']$ is finalizable and $\mathit{avg\_sdev}(\mathit{freq} \oplus T_i[t, t'])$ is minimal.

This completes the elaboration of the contraction step and thus of the greedy algorithm. The complete code is presented in §6. Figure 1 in §9 shows the algorithm in action.

5. FURTHER TRADE-OFFS

Trade-off 1. We can improve the time complexity of the *contract* step by this alternative definition for t':

> t' = a team in T_i such that either t or t' is a singleton team, its size is at most $n - \mathit{size}\, t$, and $T_i[t, t']$ is finalizable and, in addition, $\mathit{sdev}(p, \mathit{freq})$ is minimal for the/some p in t'.

There are two major differences with the previous definition. First, now a contraction step always adds a singleton team to another, which decreases the search for t' considerably. Second, we are not aiming at a minimal $\mathit{avg\_sdev}(\mathit{freq} \oplus T_i[t, t'])$ (which is an expensive computation) but instead hope that the least frequent player (much cheaper to find) together with the already chosen team with a maximal frequent player, is a reasonable attempt to achieve a small value for $\mathit{avg\_sdev}(\mathit{freq} \oplus T_i[t, t'])$. Experiments in §7 show that this trade-off decreases the computation time by a factor of about five, but also decreases the quality of the outcome T: after 25 iterations the $\mathit{avg\_sdev}$ value of the final frequency is doubled.

Trade-off 2. A second trade-off is not to search for a t with maximal $\mathit{sdev}(p, \mathit{freq} \oplus T_i)$ computed over the entire set *Player* (minus what is explicitly excluded in *sdev* itself) but only over the set P of players for which a team formation is being constructed; this is easy to realize by giving *sdev* and $\mathit{avg\_sdev}$ an extra parameter for P. And similarly for t'. Experiments in §7 show that this trade-off halves the computation time (in our Miranda implementation) and, remarkably, does not affect the quality of the outcome T at all: the new final $\mathit{avg\_sdev}$ value is about the same as the original one, and sometimes even smaller!

6. THE CODE

We express the algorithm in a functional language like Miranda or Haskell, but take the liberties of using math notations: capital letters for normal values (P, T), symbols for infix operations $(\oplus, \cdot)$, logical operations $(\wedge, \vee, \neg)$, and non-strict indentation. We do not present the coding of the trade-offs; they consist of some simple small changes.

We represent sets by unordered lists without duplicates. So in the context of some player set P, the list $[[p] \mid p \leftarrow P]$ represents the initial team formation T_0 consisting of all singleton sets $\{p\}$ for $p \in P$.

The code for the greedy algorithm is just one line:

$$\begin{array}{lcl} \mathit{greedyA}(\mathit{freq},\ n,\ P) & = & \mathit{until}\ \mathit{final}\ (\mathit{contract}\ \mathit{freq}\ P)\ T_0 \\ & & \text{where } T_0 \ = \ [[p] \mid p \leftarrow P] \end{array}$$

Function *until* is defined in the standard environment; the invocation $\mathit{until}\ \mathit{cond}\ f\ x$ yields $f^n\ x$ where n is the smallest i

for which *cond* $(f^i\ x)$ is true. Function *final* is really simple:

$$\begin{array}{lcl} \mathit{size}\ t & = & \#t \\ \mathit{complete}\ t & = & \mathit{size}\ t = n \\ \mathit{final}\ T & = & \#[t \mid t \leftarrow T;\ \neg\ \mathit{complete}\ t] \leq 1 \end{array}$$

In order to give the code for *contract*, we shall first give the auxiliary functions. The math expression '$\arg\min_{x:X} f(x)$' is coded as $\mathit{argmin}\,[(x, f\ x) \mid x \leftarrow X]$. A definition of *argmin* with linear complexity is simple (and occurs in my actual code), but would take more lines than the following definition, whose time complexity is quadratic in its arguments size:

$$\begin{array}{l} \mathit{argmin}\ zs = [x \mid (x, fx) \leftarrow zs;\ \ \mathit{and}\,[fx \leq fy \mid (y, fy) \leftarrow zs]\,] \\ \mathit{argmax} \quad = \mathit{argmin} \cdot \mathit{map}\, f \quad \text{where } f\,(x, v) = (x, -v) \end{array}$$

When we have to choose some member of the result list of *argmin*, we simply take the head:

$$\mathit{someof} \quad = \quad \mathit{head}$$

Next the standard deviation. We represent a function f with domain *domf* by the pair (domf, f):

$$\begin{array}{l} \mathit{standarddev}\,([\,], f) \quad = \quad 0 \\ \mathit{standarddev}\,(\mathit{domf}, f) \quad = \\ \quad \mathit{sqrt}\,(\mathit{sum}\,[(f\ x - \mathit{avg})^2 \mid x \leftarrow \mathit{domf}]\ /\ \#\mathit{domf}) \\ \quad \text{where} \quad \mathit{avg} \ = \ \mathit{sum}\,[f\ x \mid x \leftarrow \mathit{domf}]\ /\ \#\mathit{domf} \end{array}$$

The standard deviation is invoked with *freq* by our *avg_sdev* and *sdev*. Recall that *freq* will yield a pair (a, b) which has to be transformed to the value a/b (only if it makes sense); this is done by composing *freq* with *val* (written as *val.freq*):

$$\begin{array}{lcl} \mathit{avg\_sdev}\,(\mathit{freq}) & = & \dfrac{\mathit{sum}\,[\,\mathit{sdev}(p, \mathit{freq}) \mid p \leftarrow \mathit{Player}]}{\#\mathit{Player}} \\ \mathit{sdev}\,(p, \mathit{freq}) & = & \mathit{standarddev}\,(\mathit{dom},\ \mathit{val.freq}) \end{array}$$

where

$$\begin{array}{l} \mathit{dom} \quad = [(p, q) \mid q \leftarrow \mathit{Player};\ q \neq p;\ \mathit{snd}(\mathit{freq}(p, q)) \neq 0] \\ \mathit{val}(a, b) = a/b \end{array}$$

After these preparations, the code for *contract* literally follows the abstract definition:

$$\begin{array}{lcl} \mathit{contract}\ \mathit{freq}\ T & = & T{\cdot}[t, t'] \quad \text{where} \\ t & = & \mathit{someof}\,(\mathit{argmax}\,[\,(t,\ \mathit{sdev}(p,\ \mathit{freq} \oplus T)) \\ & & \qquad \mid t \leftarrow T; \\ & & \qquad\ \neg\ \mathit{complete}\ t; \\ & & \qquad\ p \leftarrow t\ ]\ ) \\ t' & = & \mathit{someof}\,(\mathit{argmin}\,[\,(t',\ \mathit{avg\_sdev}\,(\mathit{freq} \oplus\ T{\cdot}[t, t'])) \\ & & \qquad \mid t' \leftarrow T \text{ -- } [t]; \\ & & \qquad\ \mathit{size}\ t + \mathit{size}\ t' \leq n; \\ & & \qquad\ \mathit{finalizable}\,(T{\cdot}[t, t'])\ ]\ ) \end{array}$$

It only remains to define infix operations $\cdot$ and $\oplus$. First, $T{\cdot}[t, t']$ denotes the team formation that results from T by uniting just t, t':

$$T{\cdot}[t, t'] \quad = \quad [t +\!\!+ t'] +\!\!+ (T \text{ -- } [t, t'])$$

Second, the code for $\mathit{freq} \oplus T$ literally follows the abstract definition:

$$\begin{array}{lcl} (\mathit{freq} \oplus T)(p, q) & = & (a + a',\ b + b') \quad \text{where} \\ (a, b) & = & \mathit{freq}\,(p, q) \\ a' & = & \mathit{toNum}\,(\mathit{or}\,[\mathit{member}\ t\ p \wedge \mathit{member}\ t\ q \mid t \leftarrow T]) \\ b' & = & \mathit{toNum}\,(\quad \mathit{member}\ P\ p \wedge \mathit{member}\ P\ q \quad\ ) \\ P & = & \mathit{concat}\ T \end{array}$$

Here, *toNum* converts a truth value to a number:

$$\mathit{toNum}\ x \quad = \quad 1 \quad \text{if } x \text{ else} \quad 0$$

Finally, for completeness' sake once more the definition of finalizability:

$$\begin{array}{lcl} \mathit{finalizable}\ T & = & \#T_1 \geq \mathit{sum}[n - \mathit{size}\ t \mid t \leftarrow T_3] \quad \text{where} \\ \quad T_1 & = & [t \mid t \leftarrow T;\ 1 = \mathit{size}\ t] \\ \quad T_2 & = & [t \mid t \leftarrow T;\ 1 < \mathit{size}\ t] \\ \quad T_3 & = & T_2 \quad \text{if } T_2 = [\,] \text{ else} \\ & & \quad T_2 \text{ -- } [\mathit{someof}\,(\mathit{argmin}\,[(t,\ \mathit{size}\ t) \mid t \leftarrow T_2])] \end{array}$$

7. QUALITY ASSESMENT

It is almost certain that the outcome $T \in TF(n, P)$ of $\mathit{greedyA}(\mathit{freq}, n, P)$ does not minimize $\mathit{avg\_sdev}(\mathit{freq} \oplus T)$; how small the value of $\mathit{avg\_sdev}(\mathit{freq} \oplus T)$ will be can only be assessed experimentally. For a *thorough* assessment we should vary all parameters and average the outcomes, and even more, we should vary the problem statement so as to get an idea of our *greedy* algorithm in general. Here we are satisfied with a *rough* assessment of the quality of the algorithm and its variants, and do some test runs with the following *fixed* parameter settings:

- Desired team size: $n = 3$
- $\mathit{Player} = [0..13]$
- Number of iterations (= number of "weeks"): 25
- Initial frequency: either all-zero or random; *freqZ* and *freqR*, respectively. These are defined thus: $\mathit{freqZ}(p, q) =$ '0/0' and $\mathit{freqR}(p, q) =$ 'a/b' where a, b are chosen uniform randomly from $0\,..\,19$, and interchanged so as to get $a \leq b$. For completeness' sake, Figure 5 shows *freqR*.
- P: the players that want to play. These have been randomly generated, but in such a way that players $0\,..13\,\mathrm{div}\,2$ will play almost always (90% probability) and the others will play about half of the time (45% probability); see the left column in Figure 1.

We compare the following algorithms:

- *greedyA*: the algorithm A' as described and coded in earlier sections.
- *trade-off* 1: the more efficient choice for t' in *contract*.
- *trade-off* 2: in *contract*, the t and t' are determined by computing *sdev* and *avg_sdev* over P rather than over *Player*.
- *trade-off* 1&2: the combination of the optimizations of *trade-off* 1 and *trade-off* 2.
- *rndm*: the team formation is completely random; no attempt is made to minimize $\mathit{avg\_sdev}(\_)$.
- *bias*: the ith team is the ith segment of size n of *sort* P.

We run the algorithm "for 25 weeks": for each week number w we put $freq_{w+1} = freq_w \oplus greedyA(freq_w, n, P_w)$, taking $freq_0$ to be the all-zero *freqZ* or the random *freqR*, respectively.

Here is the comparison between the four variants of the algorithm, where *avg_sdev*(*freq*) had to be minimized. The *count* columns are listed for completeness' sake, and therefore in small print: the algorithm did not aim at a small value of *avg_sdev*(*count*). The column 'reductions' gives the number of reductions (elementary computation steps) in multiples of 10^7 that our implementation took with the language Miranda; it is a rough indication of the relative run times. Clearly, *greedyA* and *trade-off*2 (abbreviated to 'gA' and 't2') are almost of the same quality, that is, they both have a small *avg_sdev*(*freq*); the winner depends on the initial frequency started with:

	$freq_0 = freqZ$		$freq_0 = freqR$		$\times 10^7$
	avg_sdev ($freq_{25}$)	*avg_sdev* ($count_{25}$)	*avg_sdev* ($freq_{25}$)	*avg_sdev* ($count_{25}$)	red's
gA	**0.069**	1.096	**0.122**	3.427	59 ± 2
t2	**0.072**	1.071	**0.121**	3.407	24
t1&2	**0.145**	1.671	**0.182**	4.554	5
t1	**0.169**	1.920	**0.188**	4.745	9
rndm	**0.197**	**2.178**	**0.200**	**5.065**	
bias	**0.297**	**3.618**	**0.224**	**6.223**	

Notice that the average standard deviation of the final $freq_{25}$ produced by *trade-off* 1&2 and *trade-off* 1 is more than *twice* that value of *greedyA* and *trade-off* 2, when the initial frequency is the all-zero *freqZ*; for the initial *freqR* the ratio is one-and-a-half.

Here is the comparison between the four variants of the algorithm, where *avg_sdev*(*count*) had to be minimized. Now, the *freq* columns are listed for completeness' sake, and therefore in small print: the algorithm did not aim at a small value of *avg_sdev*(*freq*). Again, *greedyA* and *trade-off* 2 are almost of the same quality:

	$freq_0 = freqZ$		$freq_0 = freqR$		$\times 10^7$
	avg_sdev ($freq_{25}$)	*avg_sdev* ($count_{25}$)	*avg_sdev* ($freq_{25}$)	*avg_sdev* ($count_{25}$)	red's
gA	0.090	**1.071**	0.135	**3.275**	27 ± 2
t2	0.086	**1.066**	0.135	**3.288**	11
t1&2	0.135	**1.464**	0.195	**4.742**	2
t1	0.156	**1.744**	0.193	**4.678**	5 ± 2
rndm	**0.197**	**2.178**	**0.200**	**5.065**	
bias	**0.297**	**3.618**	**0.224**	**6.223**	

Comparing the two tables we see that, as expected, each variant produces a better final frequency when it is designed to minimize the *avg_sdev*-value of *freq* rather than that of *count*; and, similarly, better final counts when it is designed to minimize *avg_sdev*(*count*) rather than *avg_sdev*(*freq*).

In the appendix, §9, we give some more figures:

- Figure 1 gives the produced team formations.
- Figure 2 gives an indication of the quality of *greedyA*: its behavior over time; for the other variants the behavior looks similar.
- Figure 3 and 4 give the initial *freqZ* and *countZ* and their resulting $freq_{25}$ and $count_{25}$.
- Figure 5 and 6 give the initial *freqR* and *countR* and their resulting $freq_{25}$ and $count_{25}$.

8. ABSTRACT GREEDY ALGORITHM

To show the wider applicability of our greedy algorithm, we abstract from the particulars of our concrete team formation problem and formulate the algorithm in an abstract setting. First we present the abstract formulation, and afterwards we show how our team formation problem is a particular instantiation of the abstract one.

The abstract problem is to produce for input M and x an output M' such that the value of $f(M \oplus M')$ is minimal. The choice for output M' is not entirely free; it must come from a set $\mathcal{M}$ that may depend on M and x. The types of M, x and the entities $\mathcal{M}$, $\oplus$, f are fixed parameters that characterize the problem. Formally, an algorithm A is required such that:

for all M, x we have $A(M, x) \in \arg\min_{M' : \mathcal{M}} f(M \oplus M')$.

Suppose that the cardinality of $\mathcal{M}$ is exponential in the size of M and x, so that the straightforward brute-force computation is unacceptable: a search over entire $\mathcal{M}$ for an M' that has the least $f(M \oplus M')$ value. Then, the following *greedy* algorithm A' might come to rescue:

Find a superset $\mathcal{M}'$ of $\mathcal{M}$ and a function $next : \mathcal{M}' \to \mathbb{P}\,\mathcal{M}'$, and construct a sequence $M_0, M_1, \ldots, M_n$ such that:

- $M_0 \in \mathcal{M}'$ and M_0 is easy to compute from M and x;
- $M_{i+1} \in \arg\min_{M' : next(M_i)} f(M \oplus M')$ — this is the locally optimal choice;
- M_n is the first one in $M_0, M_1, \ldots$ that is in $\mathcal{M}$; be sure that n is finite.

Deliver M_n as outcome of $A'(M, x)$.

Regarding the quality of A' you must hope that M_n has an acceptably small value $f(M \oplus M_n)$. Regarding the time complexity of A': this is polynomial in the size of M and x if both the length of the sequence $M_0, \ldots, M_n$ and the time complexity of each $next(M_i)$ are polynomial in the size of M and x.

One may think of M and M' as (multidimensional) matrices, and $\oplus$ as matrix addition (or multiplication, or whatever). Function f may be specialized to "average g" over all matrix rows (or "sum g", "max g", or any other aggregation of g) for some suitable g.

Instantiation

In our team formation problem, M is instantiated to *freq*; a frequency is really a two-dimensional matrix, with p and q as indices. Output M' is instantiated to T; indeed, a team formation T can be considered as a matrix T', namely:

$$T'(p, q) \quad = \quad \langle\!\langle \text{“}p \text{ and } q \text{ are teammates in } T\text{”} \rangle\!\rangle$$

Here, $\langle\!\langle \text{cond} \rangle\!\rangle$ denotes 1 if *cond* holds, and 0 otherwise. Operation $\oplus$ is then matrix addition: $(freq \oplus T')(p, q) = freq(p, q) + T'(p, q)$. Function f is instantiated to *avg_sdev*. Notice that $\mathcal{M}$ (the set of all possible outcomes) is the set of matrices that represent *final* team formations, while matrices from $\mathcal{M}'$ may represent non-final team formations.

9. APPENDIX: FIGURES FOR THE EXPERIMENTS

See next pages.

Players that want to play	Team formation for freqZ	Team formation for freqR
0 1 2 3 4 5 7 9 10 11 12 13	[2 1 0] [5 4 3] [10 9 7] [13 12 11]	[9 2 12] [11 10 7] [13 3 0] [4 1 5]
0 1 2 3 4 6 8 9 10 11 12	[8 6] [2 1 0] [11 9 3] [12 10 4]	[9 12] [1 3 2] [8 11 10] [6 4 0]
0 1 2 3 5 6 7 8 9 10 11	[8 6] [0 3 7] [5 2 9] [11 10 1]	[11 2] [8 7 0] [5 1 3] [6 10 9]
0 1 2 4 5 6 7 8 9 10	[2 7 8] [5 10 1] [4 9 0] [6]	[8 9 2] [5 7 10] [6 4 0] [1]
0 1 3 4 5 6 7 8 9	[0 5 3] [4 7 6] [1 9 8]	[8 7 9] [5 1 3] [6 4 0]
0 2 3 4 5 6 7 8	[4 0 8] [5 7] [3 2 6]	[8 0] [4 5 7] [6 2 3]
1 2 3 4 5 6 7	[2 5 6] [1 4 3] [7]	[4 1 5] [6 2 3] [7]
0 1 2 3 4 5 6	[4 2 3] [1 6 0] [5]	[1 3 5] [6 4 0] [2]
0 1 2 3 4 5 13	[3 2 0] [13 4 1] [5]	[4 1 5] [13 3 0] [2]
0 1 2 3 4 6 12	[0 6 4] [12 1 3] [2]	[4 12 3] [6 2 0] [1]
0 1 2 3 5 6 11 13	[1 5] [11 2 6] [13 3 0]	[2 5] [11 1 0] [13 6 3]
0 1 2 4 5 6 10 12	[5 1] [10 6 4] [12 0 2]	[1 0] [4 12 5] [6 10 2]
0 1 3 4 5 6 9 11	[3 9] [4 1 6] [11 0 5]	[9 0] [11 3 5] [6 4 1]
0 2 3 4 5 6 8 10	[3 6] [4 2 5] [10 0 8]	[2 5] [4 8 3] [6 10 0]
1 2 3 4 5 6 7 9	[3 5 7] [6 9] [4 2 1]	[7 3] [9 5 2] [6 4 1]
0 1 2 3 4 5 6 8	[2 0] [3 1 6] [4 5 8]	[1 5] [4 8 0] [6 2 3]
0 1 2 3 4 5 6 7 13	[3 6 0] [7 1 4] [13 2 5]	[2 5 7] [4 1 3] [13 6 0]
0 1 2 3 4 5 6 12 13	[4 0 2] [12 6 5] [13 3 1]	[2 0 1] [12 4 5] [13 6 3]
0 1 2 3 4 5 6 11 12	[2 6 5] [11 1 3] [12 4 0]	[12 2 5] [6 0 3] [11 1 4]
0 1 2 3 4 5 10 11 13	[11 4 1] [10 3 2] [13 0 5]	[4 5 2] [11 10 1] [13 0 3]
0 1 2 3 4 6 9 10 12 13	[3 4 2] [12 9 1] [13 6 10] [0]	[9 10 1] [12 2 0] [13 6 4] [3]
0 1 2 3 5 8 9 11 12 13	[11 0 1] [13 9 2] [8 3 5] [12]	[9 2 5] [12 1 3] [13 8 11] [0]
0 1 2 4 6 7 8 10 11 12	[10 0 4] [11 8 12] [7 1 2] [6]	[4 12 0] [6 10 2] [11 7 8] [1]
0 1 3 5 6 7 9 10 11	[1 0 9] [10 3 5] [11 7 6]	[3 1 5] [6 9 10] [11 7 0]
0 2 4 5 6 8 9 10	[4 0] [9 6 5] [8 2 10]	[2 5] [9 6 10] [4 8 0]

Figure 1: The greedy algorithm in action, during 25 successive weeks. For each week number in $1 \,..\, 25$, the left column lists the players for which a team formation has to be produced (the players have been generated randomly, with a 90% probability for players $1 \,..\, 6$ and a 45% probability for the remaining players). The other columns give the team formations produced by *greedyA* when the aim was to minimize $avg\_sdev(freq)$ — not $avg\_sdev(count)$. The middle column for the case that the initial frequency is *freqZ*; the right column for *freqR*.

The values of $avg\_sdev(freq_w)$ for each week $w \in 1 \,..\, 25$ are shown in Figure 2, while Figures 4 and 6 show the final $freq_{25}$ for week 25.

	greedy algorithm				random team formation			
	initial all-zero frequency		initial random frequency		initial all-zero frequency		initial random frequency	
week w	$freq_w$	$count_w$	$freq_w$	$count_w$	$freq_w$	$count_w$	$freq_w$	$count_w$
	$avg\_sdev(\uparrow)$	$avg\_sdev(\uparrow)$	$avg\_sdev(\uparrow)$	$avg\_sdev(\uparrow)$	$avg\_sdev(\uparrow)$	$avg\_sdev(\uparrow)$	$avg\_sdev(\uparrow)$	$avg\_sdev(\uparrow)$
0	0.000	0.000	0.305	4.345	0.000	0.000	0.000	0.000
1	0.331	0.309	0.273	4.241	0.331	0.309	0.286	4.317
2	0.337	0.468	0.255	4.192	0.344	0.478	0.275	4.337
3	0.271	0.536	0.237	4.118	0.324	0.584	0.266	4.421
4	0.228	0.563	0.224	4.056	0.278	0.647	0.255	4.438
5	0.212	0.591	0.215	4.009	0.274	0.770	0.250	4.468
6	0.192	0.592	0.210	3.977	0.267	0.825	0.246	4.470
7	0.188	0.613	0.206	3.940	0.262	0.853	0.243	4.497
8	0.182	0.645	0.203	3.919	0.252	0.872	0.240	4.478
9	0.183	0.693	0.197	3.871	0.225	0.937	0.236	4.471
10	0.176	0.714	0.195	3.837	0.216	0.997	0.232	4.438
11	0.159	0.739	0.187	3.795	0.215	1.057	0.228	4.467
12	0.153	0.780	0.183	3.765	0.215	1.128	0.225	4.507
13	0.149	0.808	0.181	3.795	0.215	1.220	0.224	4.596
14	0.140	0.808	0.178	3.776	0.212	1.283	0.222	4.631
15	0.138	0.852	0.175	3.769	0.212	1.355	0.221	4.664
16	0.135	0.894	0.173	3.750	0.206	1.397	0.218	4.643
17	0.127	0.946	0.167	3.703	0.210	1.486	0.214	4.684
18	0.111	0.985	0.162	3.673	0.215	1.607	0.213	4.740
19	0.110	1.063	0.158	3.662	0.215	1.727	0.212	4.795
20	0.098	1.072	0.153	3.641	0.219	1.843	0.210	4.837
21	0.086	1.069	0.145	3.611	0.211	1.904	0.205	4.884
22	0.079	1.072	0.138	3.556	0.211	1.988	0.202	4.897
23	0.075	1.080	0.131	3.513	0.202	2.082	0.201	4.956
24	0.072	1.084	0.125	3.448	0.201	2.171	0.201	5.053
25	0.069	1.096	0.122	3.427	0.197	2.178	0.200	5.065

Figure 2: The behavior of *greedyA* over time when the aim is to minimize $avg\_sdev(freq)$; the aim was not to minimize $avg\_sdev(count)$ so these numbers are in small print. The behavior of the other variants is similar. Starting with the all-zero frequency, $avg\_sdev(freq)$ decreases faster (than with the random frequency), because then there are fewer different values in $freq_1$.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	sdevF	sdevC
0:		0/0	0/0	0/0	0/0	0/0	0/0	0/0	0/0	0/0	0/0	0/0	0/0	0/0	0.000	0.000
1:	0/0		0/0	0/0	0/0	0/0	0/0	0/0	0/0	0/0	0/0	0/0	0/0	0/0	0.000	0.000
2:	0/0	0/0		0/0	0/0	0/0	0/0	0/0	0/0	0/0	0/0	0/0	0/0	0/0	0.000	0.000
3:	0/0	0/0	0/0		0/0	0/0	0/0	0/0	0/0	0/0	0/0	0/0	0/0	0/0	0.000	0.000
4:	0/0	0/0	0/0	0/0		0/0	0/0	0/0	0/0	0/0	0/0	0/0	0/0	0/0	0.000	0.000
5:	0/0	0/0	0/0	0/0	0/0		0/0	0/0	0/0	0/0	0/0	0/0	0/0	0/0	0.000	0.000
6:	0/0	0/0	0/0	0/0	0/0	0/0		0/0	0/0	0/0	0/0	0/0	0/0	0/0	0.000	0.000
7:	0/0	0/0	0/0	0/0	0/0	0/0	0/0		0/0	0/0	0/0	0/0	0/0	0/0	0.000	0.000
8:	0/0	0/0	0/0	0/0	0/0	0/0	0/0	0/0		0/0	0/0	0/0	0/0	0/0	0.000	0.000
9:	0/0	0/0	0/0	0/0	0/0	0/0	0/0	0/0	0/0		0/0	0/0	0/0	0/0	0.000	0.000
10:	0/0	0/0	0/0	0/0	0/0	0/0	0/0	0/0	0/0	0/0		0/0	0/0	0/0	0.000	0.000
11:	0/0	0/0	0/0	0/0	0/0	0/0	0/0	0/0	0/0	0/0	0/0		0/0	0/0	0.000	0.000
12:	0/0	0/0	0/0	0/0	0/0	0/0	0/0	0/0	0/0	0/0	0/0	0/0		0/0	0.000	0.000
13:	0/0	0/0	0/0	0/0	0/0	0/0	0/0	0/0	0/0	0/0	0/0	0/0	0/0		0.000	0.000
														averaged	0.000	0.000

Figure 3: The initial all-zero frequency *freqZ*. Taking in each 'a/b' only the a, we have the initial *countZ*. For both *freqZ* and *countZ* the standard deviation in each row, and the average over all rows, is 0.0. (Remember that for the standard deviation of frequencies the entries '.../0' are discarded.)

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	sdevF	sdevC
0:		5/20	6/20	5/19	7/19	3/19	3/19	1/8	2/10	2/10	2/11	2/10	2/9	2/8	0.063	1.804
1:	5/20		4/19	5/19	6/18	3/18	3/18	2/9	1/7	3/10	2/9	4/10	2/9	2/8	0.068	1.423
2:	6/20	4/19		5/18	5/19	5/18	4/18	2/8	2/9	2/8	2/10	1/8	1/9	2/8	0.054	1.657
3:	5/19	5/19	5/18		4/17	5/18	4/17	2/8	1/7	2/9	2/7	2/9	1/7	2/8	0.044	1.542
4:	7/19	6/18	5/19	4/17		3/17	4/18	2/8	2/8	1/8	3/9	1/6	2/8	1/6	0.070	1.875
5:	3/19	3/18	5/18	5/18	3/17		4/17	2/9	2/8	2/9	2/8	1/8	1/5	2/7	0.049	1.264
6:	3/19	3/18	4/18	4/17	4/18	4/17		2/9	2/9	2/9	2/9	2/7	1/7	1/4	0.037	1.077
7:	1/8	2/9	2/8	2/8	2/8	2/9	2/9		1/5	1/6	1/5	1/4	0/2	0/2	0.085	0.722
8:	2/10	1/7	2/9	1/7	2/8	2/8	2/9	1/5		1/6	2/6	1/4	1/3	0/1	0.084	0.625
9:	2/10	3/10	2/8	2/9	1/8	2/9	2/9	1/6	1/6		1/7	1/6	1/4	1/3	0.058	0.634
10:	2/11	2/9	2/10	2/7	3/9	2/8	2/9	1/5	2/6	1/7		1/6	1/5	1/3	0.063	0.606
11:	2/10	4/10	1/8	2/9	1/6	1/8	2/7	1/4	1/4	1/6	1/6		2/5	1/4	0.086	0.843
12:	2/9	2/9	1/9	1/7	2/8	1/5	1/7	0/2	1/3	1/4	1/5	2/5		1/4	0.096	0.576
13:	2/8	2/8	2/8	2/8	1/6	2/7	1/4	0/2	0/1	1/3	1/3	1/4	1/4		0.102	0.697
														averaged	0.069	1.096

Figure 4: The final $freq_{25}$ resulting from $freqZ$. Taking in each 'a/b' only the a, we have the final $count_{25}$. The last columns give per row p the values of $sdev(\lambda\, q \bullet freq_{25}(p,q))$ and $sdev(\lambda\, q \bullet count_{25}(p,q))$.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	sdevF	sdevC
0:		5/9	6/18	2/18	4/16	10/13	2/12	1/5	1/18	16/18	7/14	4/6	5/17	1/5	0.256	4.104
1:	5/9		7/15	0/19	0/9	0/8	14/16	11/19	3/4	9/18	4/19	1/3	3/5	14/15	0.308	4.924
2:	6/18	7/15		1/5	14/17	5/14	3/14	6/9	3/8	2/12	11/19	13/15	3/16	14/16	0.255	4.509
3:	2/18	0/19	1/5		5/9	4/19	9/17	10/17	8/15	15/18	8/13	3/4	4/9	1/5	0.248	4.198
4:	4/16	0/9	14/17	5/9		1/8	0/8	9/11	1/14	4/4	10/10	7/17	0/14	14/17	0.385	4.921
5:	10/13	0/8	5/14	4/19	1/8		13/13	2/9	5/5	2/2	4/17	10/16	4/14	6/6	0.369	3.689
6:	2/12	14/16	3/14	9/17	0/8	13/13		13/17	9/11	7/19	0/19	16/18	13/14	0/10	0.378	5.745
7:	1/5	11/19	6/9	10/17	9/11	2/9	13/17		0/10	6/12	1/4	1/16	2/8	5/17	0.257	4.240
8:	1/18	3/4	3/8	8/15	1/14	5/5	9/11	0/10		3/17	6/14	1/17	7/10	1/17	0.332	2.893
9:	16/18	9/18	2/12	15/18	4/4	2/2	7/19	6/12	3/17		0/6	4/5	8/17	11/14	0.321	4.794
10:	7/14	4/19	11/19	8/13	10/10	4/17	0/19	1/4	6/14	0/6		1/8	7/19	6/8	0.285	3.552
11:	4/6	1/3	13/15	3/4	7/17	10/16	16/18	1/16	1/17	4/5	1/8		8/12	10/15	0.289	4.827
12:	5/17	3/5	3/16	4/9	0/14	4/14	13/14	2/8	7/10	8/17	7/19	8/12		3/9	0.238	3.254
13:	1/5	14/15	14/16	1/5	14/17	6/6	0/10	5/17	1/17	11/14	6/8	10/15	3/9		0.344	5.182
														averaged	0.305	4.345

Figure 5: The initial random frequency $freqR$. Taking in each 'a/b' only the a, we have the initial $countR$.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	sdevF	sdevC
0:		8/29	9/38	6/37	11/35	10/32	10/31	3/13	5/28	17/28	8/25	6/16	7/26	5/13	0.108	3.407
1:	8/29		9/34	7/38	7/27	8/26	16/34	11/28	3/11	10/28	6/28	4/13	4/14	14/23	0.109	3.683
2:	9/38	9/34		5/23	15/36	13/32	9/32	7/17	4/17	6/20	13/29	14/23	6/25	14/24	0.128	3.692
3:	6/37	7/38	5/23		8/26	9/37	15/34	11/25	9/22	15/27	8/20	4/13	6/16	6/13	0.115	3.341
4:	11/35	7/27	15/36	8/26		8/25	7/26	10/19	4/22	4/12	10/19	8/23	4/22	15/23	0.134	3.522
5:	10/32	8/26	13/32	9/37	8/25		13/30	5/18	5/13	4/11	5/25	11/24	7/19	6/13	0.078	2.935
6:	10/31	16/34	9/32	15/34	7/26	13/30		13/26	9/20	10/28	6/28	16/25	13/21	4/14	0.128	3.697
7:	3/13	11/28	7/17	11/25	10/19	5/18	13/26		3/15	7/18	3/9	4/20	2/10	5/19	0.111	3.543
8:	5/28	3/11	4/17	9/22	4/22	5/13	9/20	3/15		5/23	7/20	4/21	7/13	2/18	0.123	2.143
9:	17/28	10/28	6/20	15/27	4/12	4/11	10/28	7/18	5/23		4/13	4/11	10/21	11/17	0.123	4.191
10:	8/25	6/28	13/29	8/20	10/19	5/25	6/28	3/9	7/20	4/13		4/14	7/24	6/11	0.108	2.584
11:	6/16	4/13	14/23	4/13	8/23	11/24	16/25	4/20	4/21	4/11	4/14		8/17	11/19	0.141	4.069
12:	7/26	4/14	6/25	6/16	4/22	7/19	13/21	2/10	7/13	10/21	7/24	8/17		3/13	0.133	2.818
13:	5/13	14/23	14/24	6/13	15/23	6/13	4/14	5/19	2/18	11/17	6/11	11/19	3/13		0.170	4.347
														averaged	0.122	3.427

Figure 6: The final $freq_{25}$ resulting from $freqR$, shown in Figure 5. Taking in each 'a/b' only the a, we have the final $count_{25}$ resulting from $countR$, shown Figure 5.