An Investigation into Guest Movement in the Smart Party

Jason Stoops (jstoops@ucla.edu)
Faculty Advisor: Dr. Peter Reiher

Abstract

The Smart Party [1] is a ubiquitous computing application developed using the Panoply [2] middleware. The core idea behind the Smart Party is that someone could host a gathering which spanned multiple rooms, and attendees who are in the same room at a Smart Party will co-operate to select and provide the media for each room. This paper presents the findings of an investigation into the dynamics of a Smart Party gathering, utilizing a program that simulates parties with the media preferences of Last.FM [3] users. Specifically, the effects of various methods of room selection on fairness and satisfaction within the party are examined.

1.1. Introduction to the Smart Party

At a Smart Party, guests carry with them some kind of device with wireless capability (such as a cell phone or media player), which contains media and associated preference data. Furthermore, the user device can use its wireless capability to suggest songs, transfer media, and determine its location in the party. The music played in a room then represents the collection of musical tastes of its occupants. Each room has speakers which are attached to a controlling device that is in charge of coordinating song selection between user devices, obtaining the selected media from a user device, and playing the media in the room. There is also a master controller for the entire Smart Party which handles party-wide tasks, such as guest authorization and localization map distribution. A full discussion of the mechanisms by which the Smart Party accomplishes these tasks is outside the scope of this document, but can be found in [1].

2. Motivation

The goal of the investigation presented by this paper is to determine if there are certain rules for movement which can lead to greater user satisfaction in terms of the music heard. In other words, we seek to determine if there are there room choices that can be made during the party to improve user satisfaction and fairness. Also, by more faithfully modeling the actual Smart Party in simulation, greater insight into the behavior of the Smart Party can be obtained.

3. Introduction to the Smart Party Simulation

A simulation program was originally developed for the Smart Party for the Smart Party Case Study [1] to determine how well song selection and the amount of messages required by the party would scale as more guests were present. This simulation was then extended to add support for modeling and tracking guest movement between rooms in a party, as well as to collect statistics on the party's behavior over time.

The key to the effectiveness of the simulation lies in the use of real user preference data gathered from Last.FM, which is a web site that allows users to upload their media preferences from a variety of sources that support the Audioscrobbler interface [3], including media programs such as WinAMP [4] and portable media

devices, such as the Apple iPod [5]. The web site records which tracks the user has heard, and play counts for these tracks. From this data, a random subset of users and media is selected, and the interaction of this particular subset of users carrying this particular subset of media is then simulated. Many different random selections are simulated, and statistics on these parties are gathered and calculated by the simulation program.

3.1. Key concepts in the Smart Party

Defining several terms in the context of the Smart Party aids significantly in discussing the party's behavior.

One of the core concepts of the Smart Party is the **room**. As stated before, a party contains several rooms. Guests can only be associated with one room at a time and therefore only vote in one room at a time, but can move from any room to any other room. A room can only play one song at a time. Finally, there is no limit to how many guests can be in a room at any given time; all guests at a party could flock to one room, or a room could be empty (in which case no music plays).

The **history length** is the number of previously played songs that the guest device will track. The songs tracked are the songs that ended up being played in the room the guest is in, and this information is used to help the user evaluate the room. Closely associated with the history length is the **satisfaction threshold**, which is a value that is used as a guide for when a guest should consider switching rooms under most mobility models developed. If the average satisfaction gained from the previous *history-length* songs tracked falls below the satisfaction threshold, guests consider moving. Otherwise, the guest will stay in the current room.

A **round** in the Smart Party consists of one song selection and listening cycle. In the simulation, there are three phases in a round. First, the guests may switch rooms. Then, the voting process is carried out in each room to select the next song to be played. Finally, guests hear the song and gain any satisfaction from doing so. The rationale behind the phase structure in the simulation is discussed in Section 3.4.

An **iteration** or **party** consists of thirty rounds with the same random subset of guests and media.

A **simulation** consists of a number of iterations with different, random subsets of guests and media. All iterations in a simulation have the same population, number of rooms, and other constants such as the history length and satisfaction threshold. Statistics are gathered over the various iterations and reported at the end of the simulation.

3.2. Song selection in the Smart Party

Though several methods of song selection are put forward in the Smart Party Case Study [1], a voting algorithm is currently used in the Smart Party demonstration currently running in the UCLA Laboratory for Advanced Systems Research (LASR), and is therefore also used in all simulations run for this document.

The first step in selecting a song under the voting algorithm occurs when the room controller sends a request for nominations to the user devices. Each device then replies with a song nomination, which is a song with a high preference that has not yet been heard by the user. The room controller then collects these nominations and sends them back out to the user devices in a ballot. Each user device examines the ballot sent out and votes by allocating a certain number of points from a fixed pool to each song on the

ballot based on the user's preferences. Points are tallied by the room controller and the ballot is sorted according to how many points each song accumulated during the vote. The bottom half of the ballot is then dropped. The new, shorter ballot is sent back out to the user devices, and the process loops until one song remains.

3.3. Metrics

Several metrics are used in the context of evaluating the Smart Party. Satisfaction and fairness are key measurements of the effectiveness of the party, and room changes per guest yields insight into how much movement is occurring in the party.

3.3.1 Satisfaction

Satisfaction is related to the idea of rating a song on a scale of zero to five stars. The satisfaction gained from a song with a *k*-star rating would then be 2^k (with the exception of songs with a 0-star rating, which yield a satisfaction of 0). This exponential scale is inspired by the nature of the play count data gathered from Last.FM. To determine a song's rating, songs were separated into 5 buckets by play count. The distribution of songs into buckets ended up following an exponential curve, with the number of songs in the top bucket (the songs most often played) being smallest, and the bottom two buckets (songs that were played rarely) containing the majority of the songs [1]. If a song is not in a user's profile, it is considered to have a 0-star rating, and no satisfaction is gained by hearing the song at the party.

In the party simulation, round-by-round satisfaction is the average satisfaction gained across all the guests for each round. Overall satisfaction is the sum of the satisfaction gained during the party.

3.3.2 Fairness

The fairness of the distribution of satisfaction is quantified by calculating a value called the Gini Coefficient [6]. The Gini Coefficient is generally used as a measurement of the distribution of wealth in a population. It is a ratio between 0 and 1. A Gini Coefficient of 0 expresses perfectly equal distribution of available wealth among a population; a value of 1 expresses perfectly unequal distribution, where all wealth is held by one member of a population and others have no wealth.

Instantaneous fairness is the equality of the distribution of satisfaction gained by guests in a single round. This value is significant due to the fact that, in a real party, guests could leave part way through the party. If a guest's voice is not being heard during the party, he or she may leave due to dissatisfaction before getting his or her chance to be heard. Overall fairness is a measure of the equality in distribution of overall satisfaction among guests.

3.4. Simulation Mechanics

The Smart Party simulation has certain limitations due to the nature of the user data that we have to work with. The Last.FM profile data consists of song title, artist name, and a play count for each song associated with a user profile. There is no genre data associated with a song, so the party cannot track the nature of the playlist produced in each room, be it eclectic or more confined to a single genre.

Also, there is no song length information associated with the Last.FM data. As a result, all songs are assumed to be three minutes long, and the party operates in phases, with guests moving, voting and listening in lockstep. First, guests select a room and move. A guest is not forced to move; he or she may choose to stay in the room they are in for the next round, as well. After movement occurs, the guests in each room vote for a song to be played, using the voting mechanism described in Section 3.2. The winning song for the room is played, and guests gain a certain satisfaction based on their preference for the song played. The actual Smart Party demonstration has no such limitation on the timing of events; rooms may vote and play songs asynchronously, and guests may move between rooms at any time. However, due to the limitations of the data set used, the above simplifications have been imposed on the simulation.

At the start of the party, guests are evenly distributed between rooms; much like an entire deck of cards would be dealt to some number of players. The difference between room populations is therefore at most one guest.

Guests in the Smart Party simulation have certain pieces of data available to them on which their movement decisions may be based. After the song selection phase, all guests receive a list of songs played in all rooms. Therefore, though a guest only gains satisfaction for the song playing in his or her room, the guest is aware of the songs being played in all rooms. Also, guests are aware of the population of each room at a given moment. Guests do not know the preferences of other guests explicitly, though preference information can be inferred to some extent through the songs that end up being played in each room.

As discussed in the Smart Party Case Study [1], passing around entire user preference databases results in a large amount of data being transferred, as well as presenting privacy issues. By nominating and voting for songs, guests need not reveal more about their preferences than is necessary to select a song.

The number of times a song is heard is tracked by each guest. Once a song has been heard, a guest will not nominate or vote for the song in future song selection phases. Furthermore, the satisfaction for the song is divided by 4 for each time the song is heard. Since song satisfaction is calculated by computing 2^k, where k is the bucket the song fell into initially, dividing by 4 is equivalent to decrementing k by 2, which is in turn the equivalent of decreasing a song's rating by two stars or buckets. However, this division has the added advantage that the song's rating will never be negative. Therefore, if a guest has heard a song many times, this will still yield more satisfaction than a song that the guest has never heard before.

4. Mobility Models Tested

To evaluate the effects that guest movement has on the Smart Party experience, several mobility models were developed. These models are, at their core, sets of rules which specify when and to which room a guest should move. These rules only incorporate information that the guest has available to them in the party; they do not require any oracular orchestration or knowledge.

4.1. No Movement

This model is a trivial case of movement. Under this model, guests stay in the room in which they start at the party. Guests are initially distributed equally among rooms, so each room has the same number of guests imprisoned in it.

4.2. Random Movement

The random movement model is the simplest type of movement. It uses no information about the party. In a k-room party, after every round, each guest rolls a k-sided die to determine where to go next. The guest may roll the number of the room they are currently in, and therefore not necessarily move at every round.

4.3. Threshold-based Random Movement

Here, guests begin to use some knowledge about the party experience. Also, we introduce the concept of a threshold-based model. Under this and the other threshold based models below, guests track the last n songs, where n is the history length. Then, after each round, the guest evaluates his or her average satisfaction over the last n songs, and if this average is below the satisfaction threshold, the guest considers moving. If not, the guest stays in the current room. Each guest has the same constant satisfaction threshold value throughout the whole simulation. The choice of this value is discussed in Section 5.1.

After moving between rooms, the guest's history is flushed, since satisfaction from songs heard in previous rooms would be irrelevant to the evaluation of the guest's new room. Also, the guest need not wait for the history to fill up before evaluating the average satisfaction. This condition was tested early on and, when guests were forced to wait for n songs before considering moving, satisfaction and fairness were consistently lower than if guests are allowed to move at any time, regardless of how full the history were.

In threshold-based random movement, once average satisfaction drops below the threshold, the guest decides a new room using the same method in the random-movement model: rolling a k-sided die, where k is the number of rooms in the party. So, even after the guest's satisfaction has dropped below the satisfaction threshold, there remains a $1/k$ probability that the guest will remain in his or her current room.

The motivation for this model is to test the effectiveness of a satisfaction-threshold independently.

4.4. Threshold-based to Least Crowded Room

This model uses the same threshold mechanism described in the threshold-based random movement model. However, if satisfaction falls below the threshold here, guests choose the room with the minimum population. If there are multiple rooms that have the minimum population, the guest chooses between these rooms randomly. The idea behind this model is that perhaps in a less crowded room, a guest's vote could be heard more and therefore the guest would be more satisfied.

4.5. Threshold-based Random (Population Weighted)

The population-weighted random movement model is used to evaluate the idea of population-guided room choice using a slightly less strict room decision. Again, this

model uses the idea of a satisfaction threshold laid out in the threshold-based random movement model. Then, to choose a room, rooms are given a share of a range from 1 to 1000. The less crowded the room is, the larger the room's share of this range will be. Then, the guest chooses a number between 1 and 1000, and moves to whichever room "owns" that number. More specifically, the weight for a room is

$$(\mathit{NumberOfGuests} / \mathit{roomPopulation}[\mathit{room}])^{1.7}$$

This number is then normalized to 1000. The choice of 1.7 for a weight is somewhat arbitrary, and is discussed in the analysis of mobility model performance in Section 7.4.

4.6. Threshold to Room with Highest Satisfaction

In this model, room selection is based on the media that has been played in each room in the past. Recall that guests have the songs played in all rooms thus far available to them. In addition to tracking the satisfaction the guest has gained over the past *n* songs (where *n* is again the history length) in the current room, the guest tracks the satisfaction he or she would have gained in other rooms as well, despite not actually gaining this satisfaction. The initial satisfaction-threshold movement policy still applies. Now, when a guest decides to move, he or she chooses the room with the highest average satisfaction over the last *n* songs played in that room. The motivation behind this model is that, if the guest likes the songs recently played in a room, then they may like the songs that will be played in the future. Furthermore, if several guests are attracted to a room based on its history, then perhaps these new guests share other common song preferences as well.

5. Test Procedure

There were two major rounds of simulations performed. The first round of simulations tested a very broad number of values for history length and satisfaction threshold over the various mobility models, in order to determine some "best" values that could be used in each model. The second round of tests was a more in depth comparison of the various mobility models using the values found in the first round and various numbers of rooms and guests to observe the behavior of the Smart Party.

Throughout the simulations performed for this document, a ratio of six guests per room was maintained in choosing numbers of guests and rooms. This was done in order to ensure that the results would be comparable as parties grew. For example, if there were more guests per room on average, then a guest would have less influence on the songs played in the room, thus lowering satisfaction and fairness per guest. Therefore, in order to examine mobility solely in the context of number of rooms available, the ratio of guests to rooms is kept constant.

5.1. Round One: Seeking Values for History Length, Satisfaction Threshold

After developing the rules behind the mobility models, values for the history length and satisfaction threshold needed to be found for each model. It turns out that different values for history length and satisfaction threshold work best for different models. Values were sought using parties of eighteen guests in three rooms and thirty guests in five rooms, using each of the mobility models developed. Furthermore, twenty-five iterations were run for each set of conditions, in order to get a decent average quickly. Normally, 150 iterations would be used to thoroughly test a set of conditions, but

due to the large number of cases (371 different combinations of conditions) and limited time available, fewer iterations were run for this round of simulations.

History lengths between two and seven songs were tested. Satisfaction thresholds between 0.5 and 3 were evaluated as well. This is reasonable, since if we look at the average satisfaction per round for a party with six guests per room (derived by dividing overall satisfaction by the number of rounds) we see that this value typically hovers between 4 and 6 (on the exponential scale discussed in Section 3.3.1). Therefore, a guest should only move if they are especially unsatisfied with a room.

The values chosen for history length and satisfaction threshold were the ones which yielded the highest average overall satisfaction and most fairness for the two party sizes tested (18 guests in 3 rooms and 30 guests in 5 rooms) and are listed in Figure 1 below.

Model	History Length	Threshold
No Movement	n/a	n/a
Random	n/a	n/a
Threshold Random	4	1
Threshold Least Crowded	4	1
Threshold Highest Satisfaction	2	2.25
Threshold Random, Population Weighted	5	0.5

Figure 1: Values used for satisfaction threshold and history length for the various models. These values were found through the simulations described in Section 5.1.

5.2 Round Two: Thorough Tests of Mobility Models

In the next batch of simulations, we use the values for history length and satisfaction threshold found in round one exclusively. The mobility models are compared with populations of 18, 30, 60, and 90 guests in 3, 5, 10, and 15 rooms respectively. Simulation time goes up exponentially as more guests are added to the party, so for some of the 60 and 90 guest parties, simulations for low performing models were not run due to limited time and computing resources.

Also, during this second round of simulation, certain characteristics of the party were tracked on a round by round basis. Specifically, for each round, we have median values for satisfaction, instantaneous fairness, and room changes per guest. In other words, a median value was calculated for each round one in over all party iterations, each round two, and so on.

6. Results

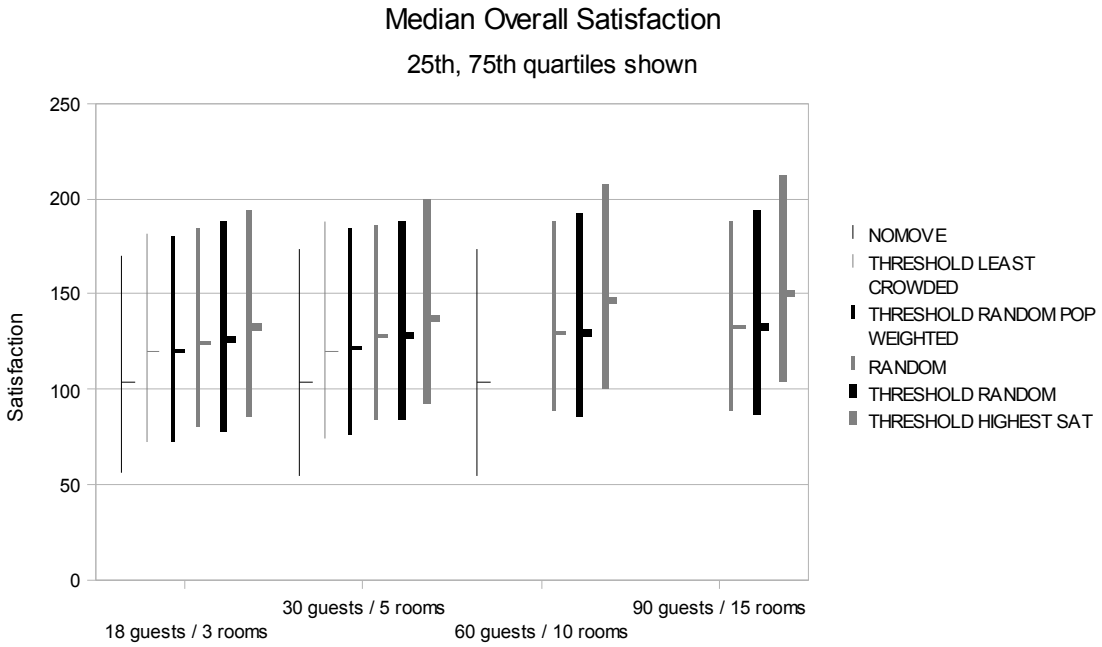

Figure 2: Median overall satisfaction across various mobility models and populations.

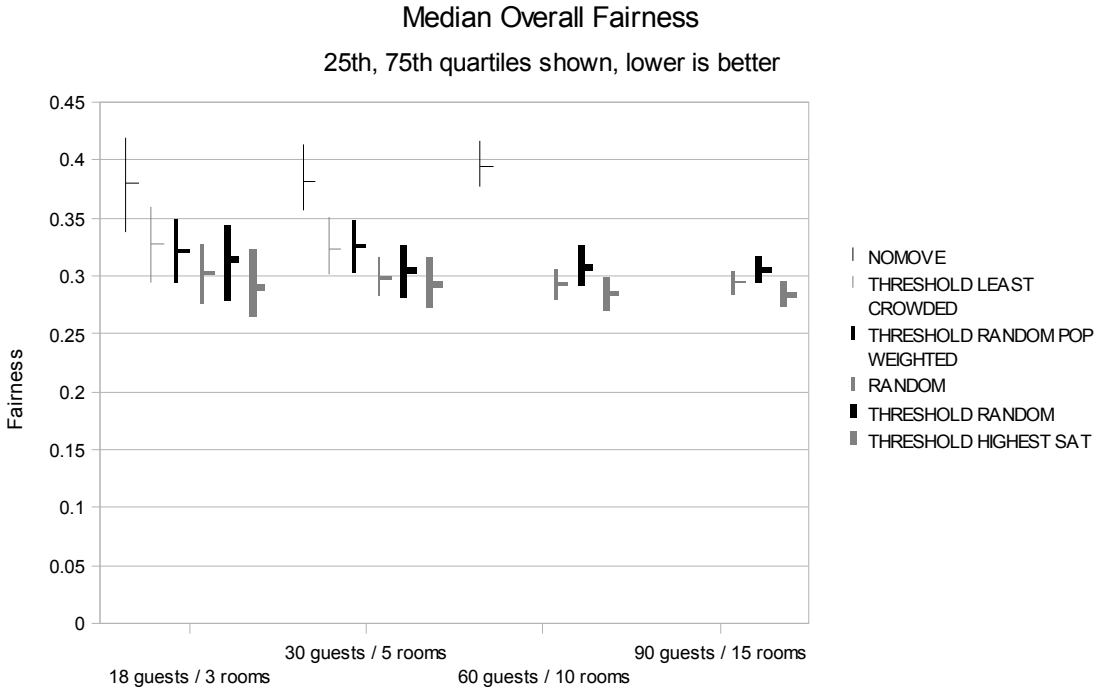

Figure 3: Median overall fairness across various mobility models and populations. Fairness is evaluated using the Gini Coefficient, and a lower value indicates more fairness.

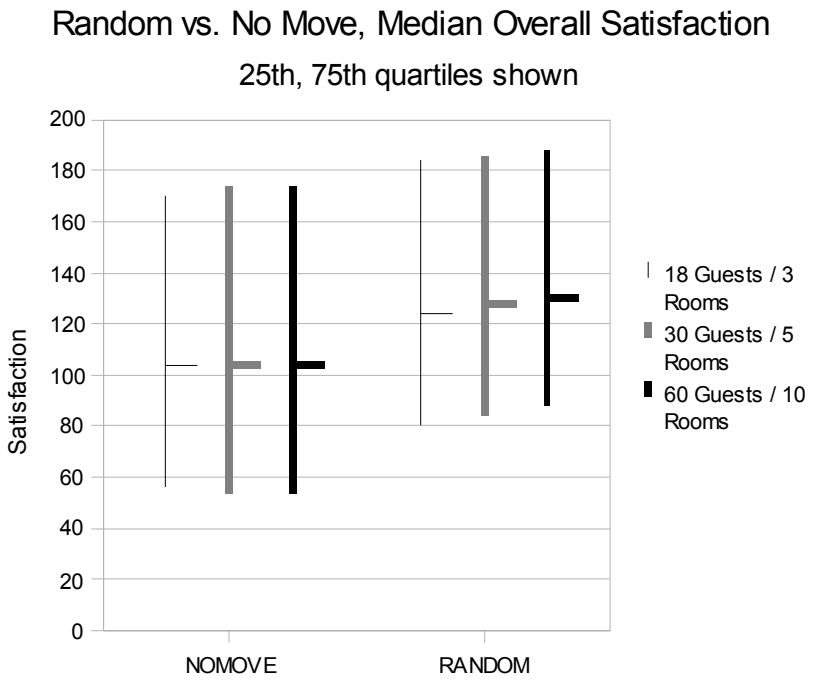

Figure 4: Comparison of satisfaction in random movement and no movement models.

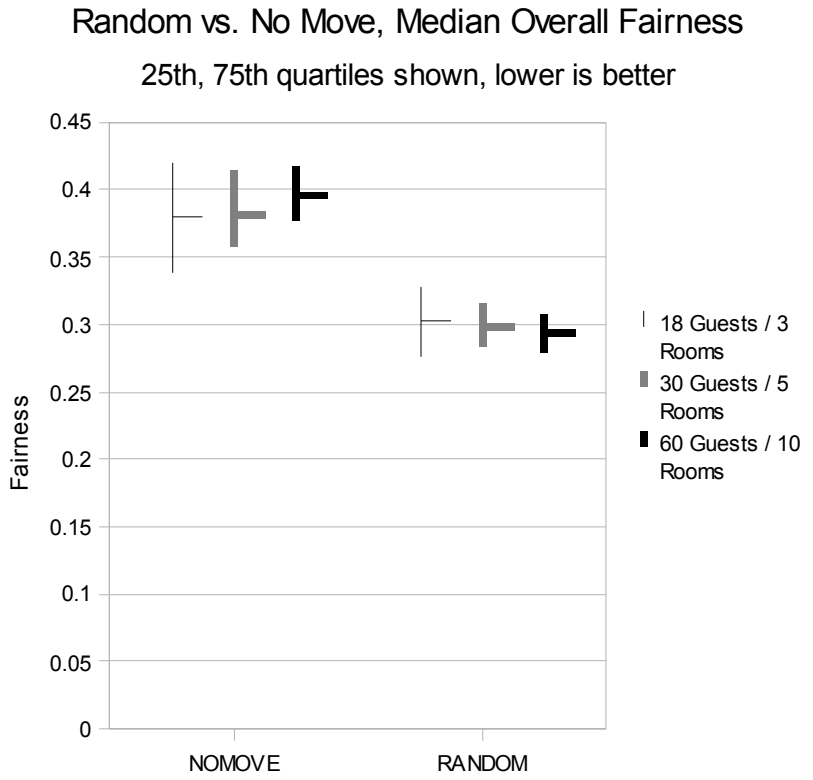

Figure 5: Comparison of fairness in random movement and no movement models.

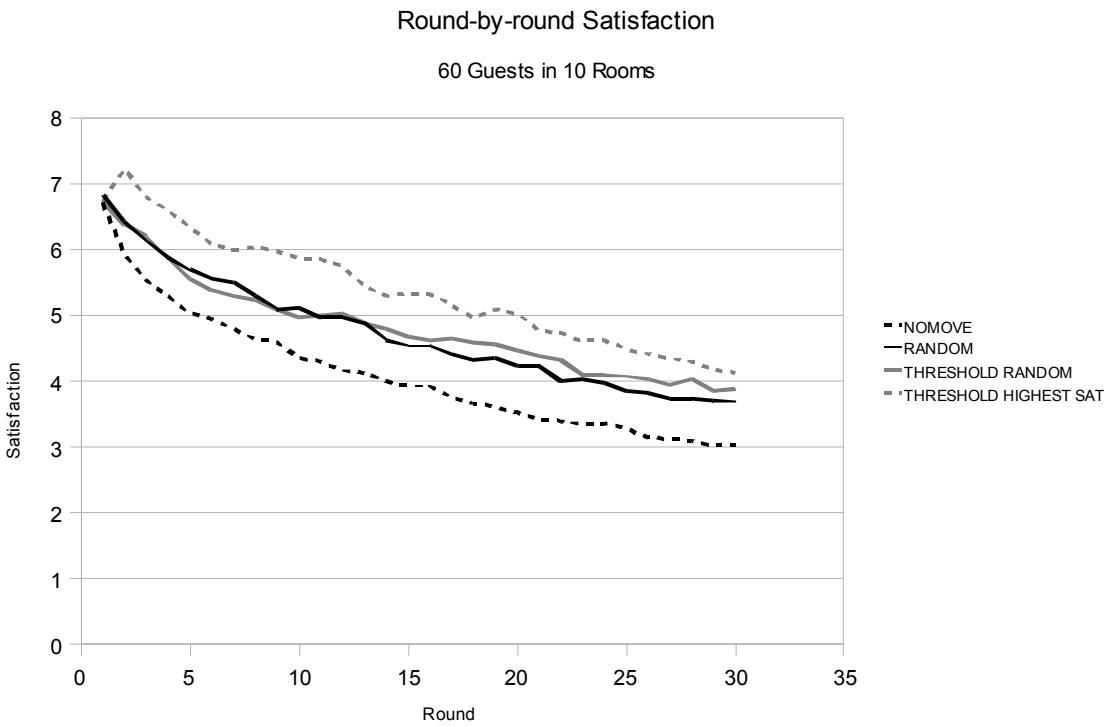

Figure 6: Round-by-round satisfaction for 60 guests in ten rooms. With ten rooms to choose from, the advantage of making good room decisions becomes more apparent.

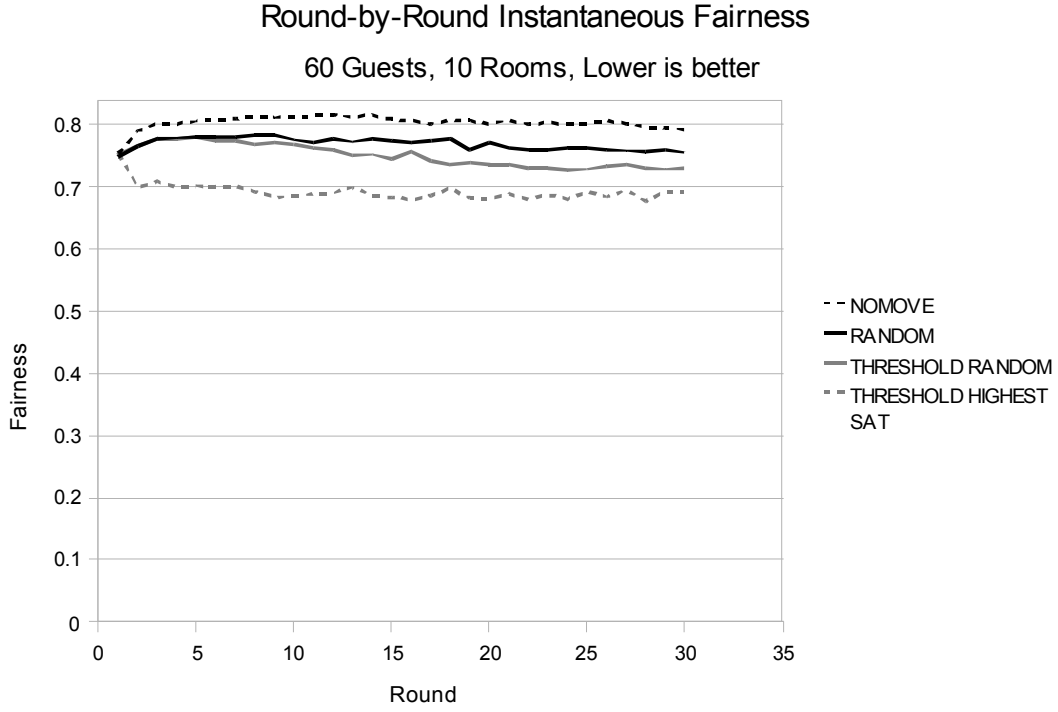

Figure 7: Round-by-round instantaneous fairness for 60 guests in 10 rooms.

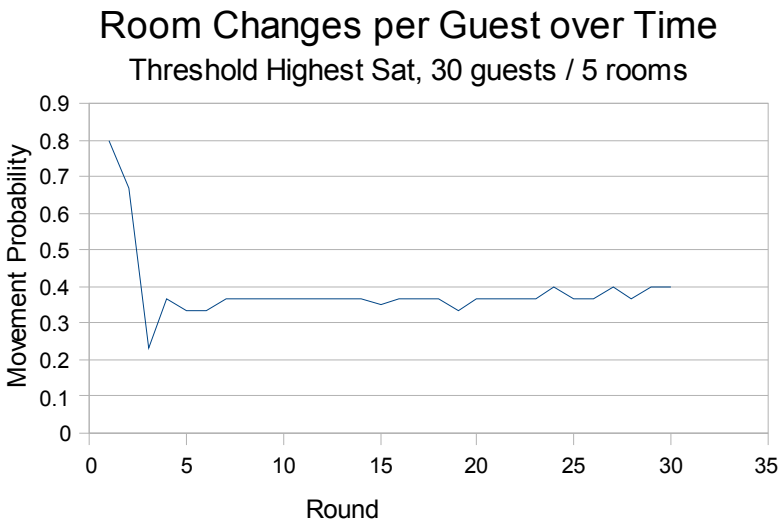

Figure 8: The amount of movement during a 30 guest party. Eventually, movement levels off to roughly 35-40% of guests moving at each round.

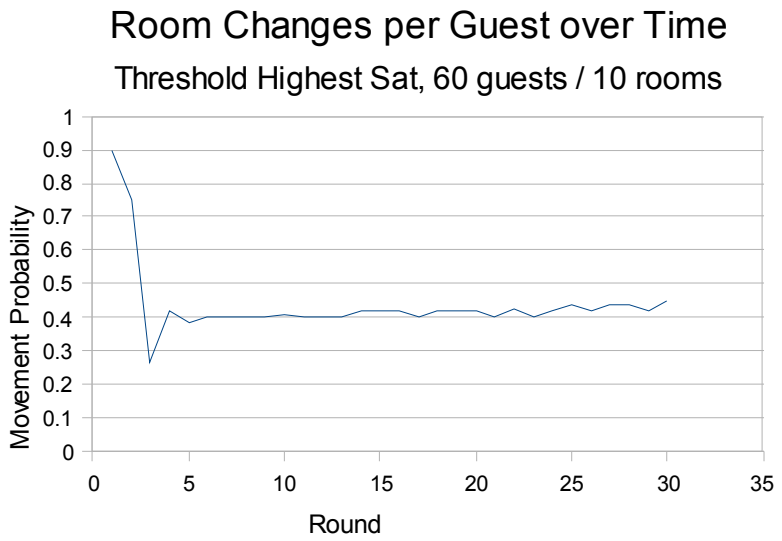

Figure 9: The amount of movement during a 60 guest party fluctuates early on, but then levels off to approximately 40-45% of guests moving at each round.

7. Analysis

After running these simulations, several interesting truths and trends regarding the Smart Party experience emerge.

7.1. Moving is Better Than Not Moving

Any movement in the party is better than no movement at all. This can be seen in Figures 4 and 5. Figure 4 shows a 19-25% increase in satisfaction when a guest moves randomly between rooms rather than staying in the initial room. Figure 5 shows similar improvements in fairness.

One possible explanation for this phenomenon is that any movement "stirs" up the party, making previously unavailable songs more available. For example, consider three users in three different rooms. The three users have the same favorite song, but no other users have heard of this song. As a result, the users may not be able to get the song played in their separate rooms by themselves. However, when users can move between rooms, perhaps at some point these three users find themselves in the same room. Suddenly, the three users now have enough votes to get the song played, thus increasing the satisfaction of these three users.

In expanding this idea to encompass the party, we consider the songs that any given group of users has in common. In other words, we consider the intersection of the preferences of some set of users, since this is the group of songs that will be played if these users are in the same room. If the group of users voting in a room never changes, perhaps they deplete this list of common songs fairly quickly, and find themselves playing songs that few users like and fewer users have even heard of. However, when users are able to move around and change the set of common songs in a room, new common songs crop up. Thus, with more availability of common songs with high preference values, we see consistently higher satisfaction at the party. This trend can be seen in Figure 6; the satisfaction curve for a party with guests moving at random starts off at the same point as the curve for no movement, but stays consistently higher for the remainder of the party.

7.2. Party Stabilization

One original hypothesis regarding the time-behavior of the Smart Party was that eventually, the party would stabilize. Guests would ultimately form groups with common interests and cease to move around the party, since they had found an ideal room. To test this, the threshold-based highest satisfaction model was used, since this reflected guests moving to rooms where they would have been satisfied by previously played content. However, in actuality, this stabilization of the party does not occur. A certain amount of movement occurs throughout the party (35-40% in the case of a 30 guest / 5 room party), as can be seen in Figure 9. Similar trends occurred in other party sizes under the Threshold-based highest satisfaction model.

The primary reason for the lack of stabilization of the party is that the preferences of the guests are constantly changing after each round. When a song is heard, the satisfaction a guest will gain from hearing that song again declines and the guest will no longer nominate or vote for the song. It is conceivable that if the preference vectors were static, then guests would gather in rooms where they liked the song being played and

continue to vote for the same song over and over, gaining some large amount of satisfaction consistently and therefore never leaving the room.

7.3. Initial Room Seeking

Looking at Figure 6, we see that the peak satisfaction for the threshold-based highest satisfaction model occurs in the second round, whereas the peak satisfaction for other models occurs in the first round and then declines over time. The reason for this behavior is that, after one round of listening, guests now have some data on which to base their room movement. Once a song is played in each room, guests will move to the room that played the song they like most. In essence, after the first round, guests with similar interests now have a means by which to find each other. This spike is more pronounced in parties with more rooms; in a three room party, the guest has far fewer choices of music than in a ten or fifteen room party.

Looking at Figure 9, we can see that after this satisfaction spike in round two, there is a dip in movement in round three. This is caused by the history length of two songs in the satisfaction-based model. It takes a round to flush the satisfaction spike from the guest's song history. Until then, a given guest's average satisfaction over the last two songs is likely to be above the threshold and the guest is not going to decide to move.

7.4. Population-Based Models Perform Poorly

Looking at Figures 1 and 2, we see that the two population-based models perform worse than choosing rooms at random. This still supports the notion that any movement is better than no movement, since these models still outperform the no movement model. However, a guest is better off choosing a room at random than choosing the room based on its population.

By examining the movements that would occur in the threshold-based model that moved to the least crowded room every time, we can see why this type of selection creates a problem. At the beginning of the party, all guests are evenly distributed between rooms. Therefore, when a guest's satisfaction drops below the threshold, the guest leaves the room, choosing a room at random since there is a tie between all rooms in the party for the least-populated room. The next guest to have their satisfaction drop below the threshold and desire a room change is then forced into the room that the first guest just left, since it is now the least crowded room. Therefore, only half of the guests moving in the party get a room choice, and the other half are forced into the room just abandoned.

Also, consider the situation in which one room has several guests who like music that is not widely appealing to the rest of the party. The other guests in such a room that make up the minority would decide to leave, and the next guest to decide to move would be forced to move to this room, since it is now the least crowded. However, because the music being played in this room is not widely appealing, this guest is also not content with the room. In this case, the room introduces an extra "step" in finding a room that plays music that some other guest will be satisfied with, since half the time if a guest wants to move, he or she must stop in this unappealing room first.

To further support the idea that population based movement does not yield high satisfaction, we look at the initial development of the threshold based random model with weighting based on population. Recall that the weight of a room is determined by the following equation:

$$(NumberOfGuests / roomPopulation [room])^{1.7}$$

In initial trial and error runs, values between 1 (truly random, no weighting) and 3 were tested for the exponent. As the value for the exponent got closer to 1, the model performed better. This, again, suggests that making a room decision based on population is not ideal.

7.5. Satisfaction-Based Model Performs Well

Figure 2 shows that the satisfaction-based model consistently out-performs other methods of room selection. There is a 4-13% improvement over random movement (depending on party size), and a 27-41% improvement over no movement at all. An informed room choice is better than a random choice, and a guest can improve their experience at the party by knowing what songs are playing in other rooms and using this information to determine where to go next.

This model performs best with a shorter history length (two songs). This is because the amount of movement occurring in the party causes the history data to become stale quickly. Since roughly four in ten guests move to a new room each round under the satisfaction-based model, a song that played in a room five rounds ago is likely a very poor indicator of what song will be played next in that room.

8. Conclusion

The Smart Party User Device Application currently has a GUI which shows the user's current preferences and songs playing in each room. After investigating how these various types of movement affect the Smart Party, it becomes clear that there is a way to implement room recommendations on the Smart Party User Device such that the user could gain more satisfaction in terms of the music heard. Though the user cannot be forced to move according to the device recommendation, it has been demonstrated in simulation that movements made with knowledge of the songs playing in other rooms yield a slightly better party experience than random movements and a vastly better experience than no movement at all.

In keeping with the context of guests and musical taste, it is imaginable that the concept of recommendation based on evolving preferences and other individual's decisions could be expanded to a wider area application that suggested bars or clubs in a city. These suggestions could be based on not only the music being played on a given night, but also measures of how crowded the venue is, gender ratios, length of patron stays, and other observable factors [7].

From a more general standpoint, the lessons learned from the Smart Party scenario presented here can perhaps be generalized to other social interactions. If we consider a more general model of the Smart Party, we see individuals with different preferences trying to form groups that agree on something to maximize their personal reward. Furthermore, the resource providing this reward is finite; in the case of the Smart Party, after hearing a song, the satisfaction gained from hearing the song again declines and the guest will not actively attempt to replay the song.

One such similar interaction could be co-workers at an office selecting groups to carpool to lunch with, and then each car decides on a restaurant destination. Repeatedly

eating at one location becomes tiresome, and thus the co-workers' preferences evolve over lunches, which could cause rearrangement of the carpools. Granted, there are other social interactions in such an environment, but the parallel is interesting nonetheless.

Another scenario could involve individuals attempting to find roommates and a place to live. Though a less direct analogy than the co-worker lunch case, perhaps each individual desires different things in a place to live, such as location, amenities, price, and neighborhood. Furthermore, roommate selection is also based on a preference for the company of various other individuals. To solve this problem, groups must not only have a high "preference" for each other, but similar preferences for a place to live as well.

Yet another possible application of Smart Party lessons and ideas is to consider the problem of congestion and routing on public roads and highways [8]. Suppose that the problem is that of an individual finding a route from point A to point B. The idea of preferences could be applied by expressing the preferred route of this individual, along with the tradeoffs that this individual is willing to make in terms of time versus distance. This determines the weight to give a route that spans a greater distance, but will perhaps yield a shorter travel time. Then, users in an area work together to report traffic hazards and obstructions (such as accidents and road closures) and to select routes cooperatively such that traffic is distributed fairly among available routes. Granted, the complexity of such a task is quite large, but the idea of cooperation, diverse goals and preferences, and rewards (here in the form of time and distance traveled by each individual) are comparable to those in the Smart Party.

9. Acknowledgements

I would like to thank Dr. Peter Reiher, Kevin Eustice, Venkatraman Ramakrishna, and Nam Nguyen for their direction, advice, support, and contributions over the course of this research. Also, I would like to thank Dr. Amit Sahai and Vipul Goyal for their guidance and the opportunity to do this research.

10. References

[1] Eustice, Kevin; Ramakrishna, V.; Nguyen, Nam; Reiher, Peter, "The Smart Party: A Personalized Location-Aware Multimedia Experience," *Consumer Communications and Networking Conference, 2008. CCNC 2008. 5th IEEE* , vol., no., pp.873-877, 10-12 Jan. 2008

[2] Kevin Eustice, Leonard Kleinrock, Shane Markstrum, Gerald Popek, Venkatraman Ramakrishna, Peter Reiher . Enabling Secure Ubiquitous Interactions, In the proceedings of the *1st International Workshop on Middleware for Pervasive and Ad-Hoc Computing (Co-located with Middleware 2003)*, 17 June 2003 in Rio de Janeiro, Brazil.

[3] Audioscrobbler. Web Services described at http://www.audioscrobbler.net/data/webservices/

[4] Winamp. Media playing software by Nullsoft. Available at http://www.winamp.com/

[5] iPod. Portable media player by Apple. http://www.apple.com/itunes/

[6] Gini, Corrado (1912). "Variabilità e mutabilità" Reprinted in Memorie di metodologica statistica (Ed. Pizetti E, Salvemini, T). Rome: Libreria Eredi Virgilio Veschi (1955).

[7] Reiher, Peter. Suggested in discussion of draft of this document. UCLA. March 2008.
[8] Venkatraman Ramakrishna. Suggested in discussion of draft of this document. UCLA. March 2008.