Pair Programming A Novel Weighted Graph Matching Approach for Pair Compatibility

Mohanraj N., Lekshmi R.S., Sankar A and Vidhya R

PSG College of Technology, Department of Applied Mathematics and Computational Sciences, Coimbatore 641 004, India.

ABSTRACT

Pair programming is a popular practice under Extreme Programming. It is increasingly followed in IT industries for effective software development and in many educational institutions for student lab assignments. Traditionally, pairs are formed based on individual preferences or administrative authority’s decision to support organization requirements as there are no standard procedures for forming pairs. A pair’s performance will be highly productive if they are more compatible with each other. In this paper, a novel method is proposed to form student pairs for programming laboratory based on weighted graph matching technique incorporating necessary psychological factors for compatibility between pairs. The experimental results demonstrate that the proposed method yields better performance of the pairs.

© 2014 AENSI Publisher All rights reserved.


INTRODUCTION

Extreme programming (EP) is a kind of agile software development methodology that stresses on achieving customer satisfaction through team work and focuses on bringing high productivity. Pair programming is one among the various principles of EP and it has been followed in many software industries and universities especially for tasks related with software development.

In pair programming, two programmers work together at a single workstation collaborating on a single artifact (code, design or algorithm). One of the two programmers called the Driver uses the keyboard and writes the code and the other called the Navigator, acting as a Reviewer, scrutinizes each line of code as it is typed in and looks for tactical and strategic defects. However, switching roles periodically in between the task is generally followed in pair programming, though not mandatory. The driver and navigator switch their roles often to make sure they contribute equally to their task and they focus on improvising the coding.

Pair programming has been used as a method for teaching programming languages in higher education for student laboratory assignments (Chaparro, E.A., 2005; Hanks, B., et al., 2004; McDowell, C., et al., 2003; Williams, L., 2000). Students and lab instructors feel that the lab sessions are more productive and it provides more satisfaction/less frustration when pair programming is employed as a programming technique (Nagappan, N., et al., 2003). Moreover, other research studies (Hanks, B., 2006; Hanks, B., 2003; McDowell, C., 2003; Nagappan, N., et al., 2003; Williams, L., et al., 2000) which compared the performances of solo programming students with that of students working in pairs, revealed that the latter were more likely to turn in solutions of higher quality for their assignments. According to Cockburn and Williams’ analysis (Cockburn, A., 2000), pair programming improves the quality of software design, reduces the defects of the code, enhances technical skills, and fosters team communication and is considered to be more enjoyable for the participants.

Despite the benefits of pair programming, there are also some negative views about it. Tessem (2003), for example, showed that some students found the experience irritating, inefficient and exhausting. Very similar results were found by Gittings et al. (2005). In their study, participants described the experience with pair programming as demanding and sometimes frustrating. Moreover, VanDeGrift (2004) showed that the students complained about working among people with different personalities and skill levels. Also, Lucas Layman’s study (Layman, L., 2006) on the effects of collaborative work on students cited non participatory partners and
difficulties in scheduling discussion times outside of the classroom as major reasons for students disliking pair programming. In spite of these negative outlooks, new methods are appearing with comparable performances and generally successful outcomes giving credence to the idea of pair programming. Generally, it is always preferable to have a companion who could be supportive in achieving the target, instead of performing the assigned task by working all alone. Programming is not an exception to this conviction.

A challenging task in pair programming is to anticipate and measure the potential compatibility between individuals thereby maximizing the productivity. As there is no principle to evaluate partner compatibility, earlier studies on pair compatibility suggested to form pairs based on various personality factors [8]. The negative views about pair programming can be reduced to a greater extent by formulating an appropriate measure of compatibility.

**Graph Matching:**

An undirected graph G is defined by a set V(G) of elements called vertices, a set E(G) of elements called edges, and a relation of incidence, which associates with each edge an unordered pair of vertices called its end vertices. If a real value is assigned to every edge of G, then G is called a weighted graph. A complete graph \( K_n \) on n vertices is one in which an edge is drawn from each vertex to every other vertex in the graph, resulting in a total of \( n(n-1)/2 \) edges. A bipartite graph is a graph whose vertex set can be partitioned into two disjoint sets X and Y such that every edge connects a vertex in X to a vertex in Y. A complete bipartite graph, denoted by \( K_{m,n} \) is a bipartite graph where the two partitions X and Y are of sizes m and n respectively and every vertex in X is connected to every vertex in Y.

A matching in a simple graph is a collection of mutually nonadjacent edges. The vertices incident to the edges of a matching M are saturated by M; the others are unsaturated. A maximal matching in a graph is a matching that cannot be enlarged by adding an edge. A maximum matching is a matching of maximum number of edges among all matchings in the graph. Every maximum matching is maximal, but not the converse. A perfect matching is a matching in which all the vertices of G are saturated. Maximum Weighted Matching (MWM) of a weighted graph is a matching in which the sum of the weights of the edges is maximum. Mathematically it can be represented as,

\[
MWM = \maximize \sum_{e \in M} w(e)
\]

where \( w(e) \) indicates the weight of the edge e. All possible maximum weighted matchings of a 4-vertex complete weighted graph is shown in Figure1.1.

![Fig. 1.1: Matchings of a weighted complete graph.](image)

The concept of graph matching is used in many industrial applications. In industrial planning, a wide variety of assignments are routinely made. For instance, the assignment of individual workers to tasks, jobs to processors, etc., can be modelled using graph matching. In some scenarios like man-machine assignment, the industry wishes to consider the experience of the person in handling the machine. Weighted bipartite graphs can be used to represent this situation. Various algorithms have already been proposed for finding matchings in general graphs. Variations of Edmonds’ blossom and Hungarian algorithms are generally used for finding maximum weighted matching in complete and complete bipartite graphs respectively. For more concepts on graph theory [15] is a good reference.

**Related Work:**

A good deal of research has already been carried out on pair programming from academic as well as industrial perspective. The recent industrial case study by Bella et al. (2013) that analysed the relationship between pair programming and defect rate under various scenarios for a team of an Italian company shows that pair programming appears to provide a perceivable but small effect on defect reduction. However, their results also indicate that the introduction of new defects tend to decrease when pair programming is practiced.

Salleh et al. (2011) presented a systematic literature review of empirical studies that investigated factors affecting the effectiveness of pair programming for Computer Science/Software Engineering students.
literature review listed 14 compatibility factors (personality types, actual and perceived skill levels, gender, communication skills, learning style, etc.) that affect pair compatibility and/or pair programming’s effectiveness as a pedagogical tool. In their study, skill level was identified to be the most important factor amongst all and their results showed that a pair works well when both students have similar abilities and motivation to succeed in a course.

On the contrary to other studies focusing on the effects of pair programming on software quality and development time, the study by Sillitti et al. (2012) focuses on the effects that pair programming has on developers’ attention and productivity. According to their study, people working in pairs concentrate more on productive activities and they engage themselves in significantly longer and uninterrupted working sessions; they focus more on the assigned tasks and thereby the need for retrieving information from sources other than the partner (for example, using web) is also reduced.

Chaparro et al. (2005) employed various data gathering techniques and analysed the reason for ineffectiveness of pair programming. In their study, students’ skill level and the programming task are identified to play a major role in the effectiveness of pair programming process. They also suggested that students should be matched with a partner who has similar skill level and a novice student should always be paired with a partner with higher skill level. Thus we find that many of the studies on pair programming insist on skill level and compatibility for effective results.

Methodology:

Owing to the ever increasing needs for pair programming, industries and academics use pair programming more frequently, but they often find it hard to understand the underlying principles of forming pairs. The success of pair programming is determined through the pairs’ success which depends on how compatible they are with each other and how effectively they can communicate and understand each other’s thoughts. Incompatible pairs have reduced understanding and improper communication between them. This leads to reduced performance, demotivation / frustration and disengagement from work. The consequences of having incompatible pairs also include delay in meeting deadlines, higher development time even for a smaller module and poor quality work. So pair compatibility is indeed essential for pair programming’s success. A pair is said to be effective if they can produce better performance which arises out of their compatibility. Generally in the case of social and student networks, the most efficient mechanism is to adopt a few psychological factors of human minds and a measure of skill level.

In this paper, we propose a novel method to measure the compatibility between individuals taking into account their skill levels too. Let G be a weighted graph formed by considering the students as vertices and the edge weights represent the measured compatibility between the corresponding pairs of individuals. The problem is to pair the students such that the overall compatibility of pairs is maximized. So graph theoretically, the problem is equivalent to finding maximum weighted matching in G. The concepts of pair programming and graph matching restricts that a student can have only one partner. Also since no student should be left without a partner, a perfect matching is required and to find one, the number of students participating is taken to be even in all our experiments.

The students were prepared for pair programming assignment by conducting a brief session on pair programming and they were instructed that they will be doing their work in pairs on a working day. All students have basic computer skills and we presume there are no scheduling difficulties as they are available during the working time. With the assumptions that there is no gender discrimination; they are all interested in pair programming and participants do not quit the work in between, we proceeded with our experiments.

Experiments And Results:

This work is a case study which is conducted to explore the effectiveness of using graph matching for compatible pairing of students. The success of a pair is determined by the accurate output of the program execution within the allotted time and the usage of effective coding standards. We conducted three experiments on a same batch of first year students from the five years integrated M.Sc Theoretical Computer Science. 26 students volunteered to participate which led to 13 pairs working on a given coding problem in Data Structures using C Programming language for each experiment. The questions given to them on all the three experiments were of similar complexity levels. All the experiments were allotted a maximum fixed time of 2 hours and 20 minutes. The first experiment comprised of pairs on their own choice. The pairs of second and third experiments were formed using graph matching method. In the second experiment we tested the students for compatibility without considering their skill levels, whereas the skill level was also taken into consideration for the third experiment.

Experiment – 1: Self Chosen Pairs:

In this experiment, the students were asked to pick partners of their own choice with no constraints on their options. Therefore students preferred to work with their friends. To evaluate the performance of pairs, we
mainly analysed and compared the lines of code and time taken by each pair. Though lines of code and time consumed do not have a direct impact on performance/outcome, they are in general important factors for determining productivity. The result of the first experiment pairs with respect to these aspects is shown in Figure4.1. Here we have sorted the pairs according to their lines of code and the time taken is plotted respectively.

![Figure 4.1: (a) Lines of code and (b) Time taken by various pairs in the first experiment.](image)

In this experiment four pairs were unsuccessful (indicated in bold bars) as they were unable to complete the programming assignment successfully within the allotted time. Figure 4.1 (b) depicts the time taken by both the successful and unsuccessful pairs. We can also infer that the time taken by unsuccessful pairs is more when compared to that of successful pairs; however it need not be true in general. Though unsuccessful pairs experienced friendliness with their partners, these pairs yielded ineffective outcomes as both partners were found to be less competent.

**Experiment – 2: Pairs chosen using weighted graph matching without considering skill level:**

Here, we assume that individuals participating in pair programming experiment are of same experience levels and they have similar skill levels as we consider peer group only. To obtain correct information about each individual’s personal characteristics which is necessary to assess the compatibility between individuals, every participant was given a questionnaire prior to conducting the first experiment. The questionnaire included various multiple choice questions to test the individual’s skills, behaviours including working nature, decision making skills and temperament. Generally, any number of questions can be asked with n different options. Conventionally four options are preferred. A total of 15 questions were given with each question having 4 different options. The answers to each question were fixed by assigning credits to every option to the answer. The best option was given a credit 4, next best was given 3 and then 2 and finally least possible option was given a credit 1. With a belief that students assess themselves properly, every participant was assigned a score based on his/her answers. The scores were calculated considering only the answers to the questionnaire and no other additional factors were included. We cumulated the candidate’s answer to every question to arrive at his/her final score. If $P_{kj}$ indicates the predefined value to the option j chosen as answer by the kth student for the question i then his/her absolute score $S_k$ is calculated as follows.

\[
S_k = \sum_{i=1}^{15} \left( \frac{\sum_{j=1}^{4} P_{ij} \times \text{credits}}{\text{total credits}} \right)
\]

(1)

From the absolute score, relative score $S_k^r$ to each individual is then calculated using percentiles, to evaluate the relative standing of a student amongst all students. With relative scoring, we try to improvise the batch performance as a whole rather than each individual’s performance.

Having computed the scores of every student, we construct the student network as a weighted complete graph. We assign the weight of vertex k to be the score $S_k$ of kth student. With each vertex having relative score, we computed the edge weight for every edge by taking the average of its vertex weights. Suppose an edge e is incident on vertices $v_1$ and $v_2$, its weight $w_e$ is calculated mathematically as,

\[
w_e = \frac{w_{v1} + w_{v2}}{2}
\]

(2)

where $w_{v1}$ indicates the weight (score) of the vertex(student) $v_1$.

These edge weights measure the level of compatibility between the corresponding individuals, which is a critical element for deciding the effectiveness of pairs. The graph with these calculated edge weights is then fed
to Edmond’s Blossom graph matching algorithm to find a maximum weighted matching. The pairs which were formed as a result of the algorithm are considered for the second experiment.

**Fig. 4.2:** (a) Lines of Code (b) Time taken by the pairs in second experiment.

The result of the second experiment pairs with respect to lines of code and time taken to complete the assignment is shown in Figure 4.2. Notice that there is only one unsuccessful pair (shown in bold), indicating that the number of unsuccessful pairs has reduced when compared to the first experiment.

**Experiment – 3: Pairs chosen using weighted graph matching considering skill level:**

In the previous two experiments, we were not concerned about the skill levels of the pairs. But, a pair is successful in this kind of programming only if they are paired by considering their skill levels too, as suggested by Chaparro et al. (2005). This led us to revamp our methodology in forming pairs and a third experiment was conducted, in which the edge weights incorporate the skill/knowledge level of the pairs. To arrive at the perceived skill level of students, their lab marks, faculty rating and personal rating of the students were included. The faculty rating of each student for C programming language and data structures were obtained from the respective lab faculties. The skill level $SK_k$ of $k^{th}$ student is then derived as,

$$SK_k = L_c + L_d + F_c + F_d + P_c + P_d$$

where $L_c$ and $L_d$ represent the laboratory performance marks of the $k^{th}$ student for C Language and Data structures respectively taken from the semester examinations; $F_c$ and $F_d$ represent the scores given by the corresponding faculties for C Language and Data structures respectively; $P_c$ and $P_d$ represent the self-evaluated scores given by the student for C Language and Data structures respectively. For this experiment, the final score $S_k$ for $k^{th}$ student is calculated by adding the skill level $SK_k$ from equation (3) and $QS_k$ from equation (1).

$$S_k = QS_k + SK_k$$

To ensure that a novice is paired with a partner who has better domain knowledge and with a view to make the overall performance better, we adopt the following strategy in forming pairs. The student scores were sorted in non increasing order and the students were split into two groups A and B according to the sorted scores. Group A included the first 13 students in the sorted list and group B comprised the rest. This means that on average, students in group A have higher skill levels when compared to students in group B. Unlike the student network as the complete graph $K_{26}$ in the second experiment, here the network is a complete bipartite graph $K_{13}$.

As in the second experiment, the weight of vertex $k$ is the score $S_k$ of $k^{th}$ student and the edge weights are calculated by averaging the corresponding vertex weights. The bipartite graph structure restricts that a student from A can be paired only with a student from B, ascertaining that at least one among the pair has better skills in comparison with his/her counterpart. This ensures that all pairs are balanced in terms of skill level/domain knowledge. The bipartite graph with the newly computed edge weights is then fed to Hungarian graph matching algorithm to find a maximum weighted bipartite matching.

The lines of code and time consumed by each of the pairs in the third experiment are shown in Figure 4.3.

Notice that the lines of code and time taken by the new pairs have considerably reduced and for this experiment, all pairs were successful. This method of pairing is best suitable when novices are present since it reduces the possibility of two beginners/proficients getting paired together.
Analysis:

In this section, we perform a comparative study of all the three experiments in terms of lines of code and time consumed. Figure 4.4 plots the graph of the average lines of code and average time consumed by all pairs for the three experiments.

From Figure 4.4(a), we can see that the average lines of code for the second experiment is lesser than that of the first experiment. But the average lines of code for the third experiment is significantly lesser when compared to the other two experiments. Also, Figure 4.4(b) depicts that the average time consumed by the first experiment pairs to complete their programs, is higher than that of the second and third experiment pairs. Of all the three experiments, the third experiment has the least time.

Next we analyzed the completion times of the pairs for the three experiments. The three time intervals (in minutes) considered here are 80-100, 101-120 and 121-140. We can observe that the third experiment pairs have completed faster (within 120 minutes) while many pairs have taken up to 140 minutes during first and second experiments. Figure 4.5 compares the number of pairs for each experiment and the time intervals for their programming.

These results indicate us that the second and third experiments showed better performance than the first experiment. Especially, the third experiment in which the computation of maximum weighted matching
included skill levels too, using a complete bipartite graph representation, is experimentally found to produce fruitful outcome.

After the completion of every experiment, we also obtained feedbacks about pair programming from the participants. A questionnaire was distributed after each assignment was complete, to know how they felt about their new partners and to know if it was productive to them. No reassessment of individuals was done after this questionnaire. Students felt that they were more compatible with their partners of the third experiment than those of the previous experiments; their discussions with their new partners were fruitful and it filled the holes in their subject knowledge. The latter feedback provides strong evidence that skill level plays an important role in pair programming. They opined, pair programming provided them mutual motivation and boosted their confidence in completing the task successfully, as the partners can help each other when required. They were even motivated to work in pairs again.

**Conclusion:**

Pair programming is definitely one of the best mutual teaching-learning methodologies when the pairs are compatible and has the drive to achieve. The feedback survey conducted during our case study revealed that pairs of the third experiment were more comfortable and enthusiastic, which favours the significance of skill level in pairing. It was evident from the third experiment that the pairs were more compatible and they produced promising results. Essentially the success of pair programming depends on both the complexity of programming task and compatibility of the pairs. Only when the pairs are compatible with each other, the working environment will be more interesting for the pairs, which will improve their productivity. It will be a win-win strategy for both the partners where the job is expected to be completed successfully.

Our work on pair programming using graph matching can be extended to distributed programming environment where the pairs will be geographically separated and the communication between the pairs can be through text chatting, voice or video conferencing. Moreover there is scope for enhancing the measure of compatibility by encompassing many other related factors.

**ACKNOWLEDGEMENT**

We would like to thank Sharda Jayaraman, psychotherapist under supervision, for the fruitful discussions and guidance in preparing the questionnaires used in our case study and all students who volunteered their precious time for our research.

**REFERENCES**


Hanks, B., C. McDowell, D. Draper, M. Krmnjac, 2004. *Program Quality with pair programming in CS1*, Proc. 9th Annual SIGCSE conference on Innovation and technology in computer science education (ITiCSE ’04), 176-180.


Appendix:

PAIR PROGRAMMING QUESTIONNAIRE – 1

Do you know the domain of work?  
(  ) Yes  (  ) No  
If yes, then how much do you think is your competency level?  
(  ) High  (  ) Average  (  ) Low  (  ) Don’t Know

1) Assume you are working in pairs, you find a mistake in your partner’s work (code). How will you react to this situation?  
(  ) You will intimate your partner immediately and guide him/her correctly.  
(  ) You will wait until your partner finds the mistake by himself/herself.  
(  ) You will not bother about the mistake as you will rectify it during your turn.  
(  ) You will get your partner’s reason for such a work, before pointing it as a mistake.  

2) What will you do if your partner is unable to find a solution for a problem?  
(  ) Guide your partner to use resources like internet or books to find a solution.  
(  ) Ask your partner to find the solution by himself/herself.  
(  ) You will leave the problem as it is, because of other important activities.  
(  ) You rely on your own experience to find potential solutions to a problem and so you don’t mind much even if your partner is not able to do.  

3) You need to make an immediate decision. What will you do?  
(  ) You will consult your partner, discuss and then take a decision.  
(  ) You feel time will be wasted in unnecessary arguments during decision making and when making a decision, you trust your inner feelings and reactions, so you take the decision on your own.  
(  ) You will maintain silence for a while and avoid the arguments with your partner.  
(  ) You feel that your partner is not capable of deciding, so you will make the decision by yourself.  
(  ) You avoid making important decisions until the pressure is on, so postpone it for a while.  

4) How will you handle a stressful situation?  
(  ) You will seek the help of humour to reduce the tension around.  
(  ) You will be involved in a direct communication with your partner.  
(  ) You will lose patience with the need to get your partner involved in discussion.  
(  ) You will discuss to outsiders about the problems that you face in the team.  

5) What will you do when conflicts arise between you and your partner?  
(  ) You will maintain silence for a while and avoid the arguments with your partner.  
(  ) You will stress for an honest discussion of the differences and the reasons for conflicts.  
(  ) You will explain and provide reasons to prove why one side is correct and the other is incorrect.  
(  ) You will try to break the tension with a supportive or humorous remark.  

6) What will you do when things go wrong on the team?  
(  ) You will emphasize on listening, feedback, and participation.  
(  ) You will arrange for a candid discussion of your problems.  
(  ) You will work hard to provide more and better information.  
(  ) You will suggest revisiting your basic mission and start reworking on it.  

7) Among the following, which do you think is difficult to do and that could put you in a troublesome situation?  
(  ) Questioning some aspect of your partner’s work.  
(  ) Pushing the team to set higher performance standards.  
(  ) Working outside your defined role or job area.  
(  ) Providing your partner with feedback on their behaviour as a team.  

8) Assume you and your partner know the domain of work and both are good at it. Which among the following, you think is required for pair problem solving?  
(  ) Co-ordination and co-operation by both members.
b) High-level listening skills to absorb significant information.
c) A willingness to interrogate your partner with tough questions during work.
d) Acquisition of good, solid data that defines the problem well.
9) Suppose your partner doesn’t have any knowledge in the domain in which you are going to work as a pair, what will you do?
a) You will not accept that assignment.
b) You will try to educate your partner by mutual reading.
c) You will do the entire work by yourself and ask your partner to absorb what you are doing and thereby make him/her learn the area of work.
d) You will give some time for your partner to learn and once he/she finishes, you both will commence the work together.
10) When your partner goes wrong at some point during the work, what will you do?
a) You will criticize your partner so that he/she can learn from it.
b) You will ignore your partner.
c) You will specify the resources for your partner to learn.
d) You will educate him by asking more specific questions and giving suggestions.
11) Suppose you find that your partner is more knowledgeable than you, what will you do?
a) You will let your partner do the work and your involvement will be less.
b) You will frankly admit to your partner and ask him/her to teach you.
c) You will get some time for you to learn on your own.
d) You will not take up such a project.
12) What will you do when pairing up with a new partner for a work?
a) You will try to meet and get to know the person.
b) You will ask direct questions about the goals and methods that you both need to work on.
c) You will talk to your partner to know what is expected of you.
d) You will engage in a discussion with your partner for clarity about your basic mission.
13) What according to you is the basis for the team decision?
a) The team’s mission and goals.
b) A consensus of team members.
c) An open and candid assessment of the issues.
d) The weight of the evidence such as available information, statistics, etc.
14) How will you deal if you feel that your partner is too rigid?
a) You will try to convince him/her with your own ideas.
b) You will just ignore your partner for the time being and concentrate on your work.
c) You will inform your superior for a better replacement.
d) You will tell him frankly that he/she is going in a wrong direction.
15) If your partner keeps on finding faults with you and criticizes your approach, how will you react?
a) You will try to explain and convince him/her about the effectiveness of your logic/approach.
b) You will ignore his/her criticism as you firmly know your approach is right and you will not waste time on proving your point.
c) You will inform your superior about your partner’s attitude.
d) You will tell him/her frankly that he/she is not encouraging.

Answer Credits: (weightage in the order 4 3 2 1)

| 1)   | d a b c | 3)   | a b d c |
| 2)   | a d b c | 4)   | b a d c |
| 5)   | b c d a | 6)   | b a c d |
| 7)   | c b d a | 8)   | a c b d |
| 9)   | b d c a | 10)  | d c a b |
| 11)  | c b a d |
| 12)  | d b a c |
| 13)  | c b a d |
| 14)  | a d b c |
| 15)  | a d b c |

Pair programming questionnaire – ii “**”:
1) Did your partner cooperatively follow the pair programming model (rotating roles of driver and navigator, questioning and making observations as the navigator)?
2) Was your partner’s participation professional and cooperative overall?
3) Did your partner contribute fully, fairly, and actively, to the best of his or her ability, to the completion of the lab assignment?
4) If given another opportunity, would you like to work in pairs again?
5) Did you find social difficulties with your companion?
6) Do you think collaboration with your partner gives you more confidence in solving programming problems?

7) Do you think this collaboration in the forum will be more effective if it has more than two members?

8) Do you think pair programming process enriches your knowledge?

9) Assess the technical competency of your partner relative to yourself
   a) better
   b) about the same
   c) weaker

10) Assess how compatible you and your partner were
    a) Very compatible
    b) OK
    c) Not compatible.

*Questionnaire given after each experiment for feedback survey*