SMP-BASED CLONE DETECTION

Hosam AlHakami, Feng Chen and Helge Janicke

Software Technology Research Laboratory,
De Montfort University, Leicester, UK
hosam.alhakami@myemail.dmu.ac.uk
{fengchen,heljanic}@dmu.ac.uk

ABSTRACT

Code cloning is a severe problem that negatively affects industrial software and threatens intellectual property. This paper presents a novel approach to detecting cloned software by using a bijection matching technique. The proposed approach focuses on increasing the range of similarity measures and thus enhancing the recall and precision of the detection. This is achieved by extending a well-known stable-marriage problem (SMP) and demonstrating how matches between code fragments of different files can be expressed. A prototype of our approach is provided using a proper scenario, which shows a noticeable improvement in several features such as scalability and accuracy.

KEYWORDS

Clone Detection, Stable Marriage Problem

1. INTRODUCTION

The Stable Marriage Problem (SMP) is a well-known problem that has been defined by Gale and Shapley in 1962 [1]. An example of the SMP is allocating the right jobs to their most suitable jobseekers (one-one). Similarly framed problems with differing cardinality are also considered to be instances of the SMP, such as matching graduated medical students to hospitals (one-many) [2]. The SMP grantees the stable match between the candidates.

Clone detection has intensively investigated due to the need of tackling code issues in the maintenance process. Current detection algorithms are more or less search based algorithms that do not consider the preferences of both candidates (code portions) in the process. In this paper, a variant of the stable marriage problem algorithm to clone detection is investigated to find clones of different source files. The extended algorithm introduces the preferences of code segments based on the values of predefined metrics, e.g. the number of calls from or to a method, cyclomatic complexity. The clone detection process should therefore consider the values of both parties.

The remainder of this paper is structured as follows. In Section 2 we introduce the background to the SMP research. In Section 3 we provide the context for the SMP algorithms to be applied to Clone Detection. In Section 4 we present an adapted SMP algorithm that is suitable to generate fair and stable matches between similar code fragments of different source files. In Section 5 we conclude the paper and outline future work in this area.

Natarajan Meghanathan et al. (Eds) : ACSIT, FCST, ITCA, SE, ICITE, SIPM, CMIT - 2014
pp. 89–101, 2014. © CS & IT-CSCP 2014 DOI : 10.5121/csit.2014.4608

2. SMP ALGORITHM

2.1. OVERVIEW

In 1962, David Gale and Lloyd Shapley published their paper College admissions and the stability of marriage [1]. This paper was the first to formally define the Stable Marriage Problem (SMP), and provide an algorithm for its solution. The SMP is a mechanism that is used to match two sets of the same size, considering preference lists in which each element expresses its preference over the participants of the element in the opposite set [1]. Thus, the output has to be stable, which means that the matched pair is satisfied and both candidates have no incentive to disconnect. A matching *M* in the original SMP algorithm is a one-to-one correspondence between the men and women. If man *m* and woman *w* are matched in *M*, then *m* and *w* are called partner in *M*, and written as *m = PM(w)* (which is the M-partner of *w*), *w = PM(m)* (the M-partner of *m*). A man *m* and a woman *w* are said to block a matching *M*, or called a blocking pairs for *M* if *m* and *w* are not partners in *M*, but *m* prefers *w* to *PM(m)* and *w* prefers *m* to *m = PM(w)* [2]. Therefore, a matching *M* is stable when all participants have acceptable partners and there is no possibility of forming blocking pairs. This problem is in interest of a lot of researchers in many different areas from several aspects. Matching problems on bipartite sets where the entities on one side may have different sizes are intimately related to the scheduling problems with processing set restrictions [3].

An instance *I* of SM involves *n* men and *m* women, each of whom ranks all *n* members of the opposite sex in strict order of preference. In *I* we denote the set of men by *m = m1, m2, mn* and the set of women by *w = w1, w2,….wn*. In SM the preference lists are said to be complete, that is each member of *I* ranks every member of the opposite sex as depicted in figure 1.

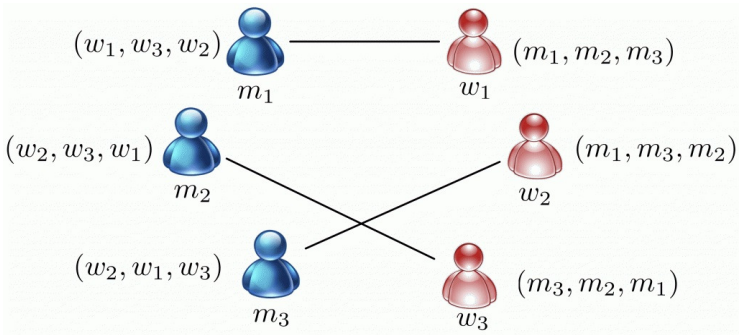

Figure1. General view of SMP. [4]

2.2. GALE SHAPLEY EXTENDED ALGORITHM

The algorithm presented by Gale and Shapley for finding a stable matching uses a simple deferred acceptance strategy, comprising proposals and rejections. There are two possible orientations, depending on who makes the proposals, namely the man-oriented algorithm and the woman-oriented algorithm.

In the man-oriented algorithm, each man *m* proposes in turn to the first woman *w* on his list to whom he has not previously proposed. If *w* is free, then she becomes engaged to *m*. Otherwise, if *w* prefers *m* to her current fiancé *m*, she rejects *m*, who becomes free, and *w* becomes engaged to *m*. Otherwise *w* prefers her current fiancé to *m*, in which case *w* rejects *m*, and *m* remains free. This process is repeated while some man remains free. For the woman-oriented algorithm the process is similar, only here the proposals are made by the women.

The man-oriented and woman-oriented algorithms return the man-optimal and woman-optimal stable matching respectively. The man-optimal stable matching has the property that each man obtains his best possible partner in any stable matching. However, while each man obtains his best possible partner, each woman simultaneously obtains her worst possible partner in any stable matching. Correspondingly, when the woman-oriented algorithm is applied, each woman gets her best possible partner while each man get his worst possible partner in any stable matching.

Theorem 1 All possible execution of the Gale-Shapley algorithm (with the men as proposers) yields the same stable matching, and in this stable matching, each man has the best partner that he can have in any stable matching [2].

According to the previous theorem if each man has given his best stable partner, then the result is a stable matching. The stable matching generated by the man-oriented version of the Gale-Shapely algorithm is called man-optimal. However, in the man-optimal stable matching, each woman has the worst partner that she can have in any stable matching, leading to the terms of man-optimal is also woman-pessimal. This results in the next theorem.

Theorem 2 In the man-optimal stable matching, each woman has the worst partner that she can have in any stable matching [2].

The following example in figure 2 gives the same output for both man-optimal and woman-optimal, the instance formed out of 4 elements.

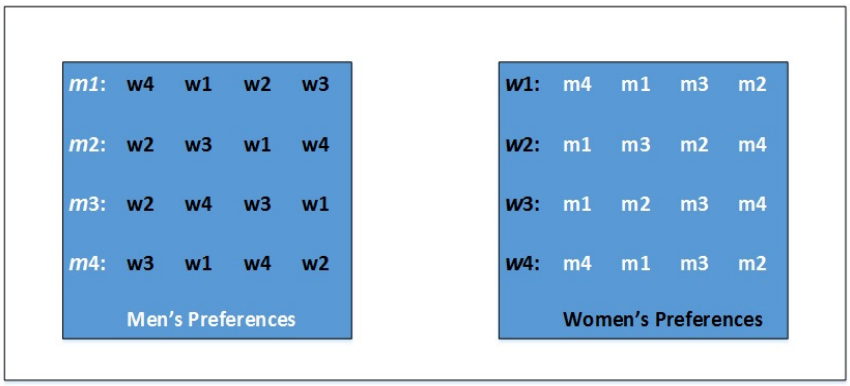

Figure 2. A stable marriage instance of size 4.

The results of different cases differ from man-oriented version to woman-oriented version. The stable matching generated by both man-oriented and women-oriented versions is *M0 = Mz = (1, 4), (2, 3), (3, 2), (4, 1).*

An extended version of Gale-Shapley algorithm has been designed to improve the basic algorithm. The extended version reduces the preference list by eliminating specific pairs that can be clearly identified as unrelated to any stable matching. The deletion process of such pair is performed by deleting each other from the preference lists.

Algorithm 1 Extended Gale-Shapley algorithm [2]

```
1: assign each person to be free
2: while some man m is free do
3: begin
4: w:= first woman on m's list;
5: if some man p is engaged to w then
6: assign p to be free;
7: assign m and w to be engaged to each other;
8: for each successor m' of m on w' list do
9: delete the pair(m',w)
10: end;
```

2.3. HOSPITALS/RESIDENTS PROBLEM

The hospitals/residents problem (also called Colleges/Students problem, and by many other names) reflects a cardinality of many-to-one of the stable marriage problem. This cardinality touches a wide range of large-scale applications that require stable matching such as students/colleges problem. Therefore, it has interested the researchers in different aspects for instance recruitment in which uses schemes to match a group and employers to a group of employees. The National Resident Match Program [5] is a real example in the US which annually matches hospitals to about 30,000 medical residents. An instance of the hospitals/residents (HR) problem consists of a set *R* of *n* residents and a set *H* of *m* hospitals, where each hospital *h* has capacity *ch*, the maximum number of positions available in *h*. Each resident ranks the hospitals in *H* that are acceptable to her in strict order of preference; likewise, each hospital ranks the residents in *R* that are acceptable to it in strict order of preference. A matching *M* for the instance is a set of resident-hospital pairs where in every pair the resident and the hospital are mutually acceptable to each other, every resident appears in at most one pair, and every hospital *h* appears in at most *ch* pairs. A pair forms a $(r, h) \notin M$ blocking pair with respect to *M* if

i) *r* is unmatched and finds *h* acceptable or *r* prefers *h* to the hospital she is assigned to and, simultaneously,

ii) *h* is not filled to capacity and finds *r* acceptable or *h* prefers *r* to one of the residents assigned to it.

Algorithm 2 Hospital-oriented algorithm

```
1: assign each resident to be free
2: assign each hospital to be totally unsubsidised;
3: while (some hospital h is unsubsidised) and (h's list
contains a resident r not provisionally assigned to h) do
4: begin
5: r:= first such resident on h's list;
6: if r is already assigned, say to h', then
7: break the provisional assignment of r to h';
8: provisionally assign r to h;
9: for each successor h' of h on r's list do
10: remove h' and r from each other's lists
11: end;
```

Intuitively, if *(r, h)* forms a blocking pair in *M* then *r* and *h* are likely to break their assignments under *M*, causing the matching to unravel. Thus, the goal of the HR problem is to find a matching

that is stable and has no blocking pairs. In their seminal paper [1], Gale and Shapley first tackled the problem in the simplier stable marriage (SM) setting where residents and hospitals are replaced by men and women. Every participant has a complete preference list (i.e., every man ranks all the women and every woman ranks all the men), and a capacity of one (i.e., every individual can have at most one assigned partner). They introduced the deferred-acceptance algorithm to find a stable matching, and showed that the algorithm can be extended to the more general HR setting. Consequently, they proved that every HR instance has a stable matching which can be computed in *O (nm)* time. In [6] Cheng et al. examined the structure of the set of all stable matchings of an HR instance and introduce the notion of meta-rotations in this setting. Also, they discuss the problem of finding feasible stable matchings.

Theorem 3

(i) The matching specified by the provisional assignments after the execution of the hospital-oriented algorithm is stable.

(ii) In this matching, a hospital h with q available places is assigned either its best q stable partners, or a set of fewer than q residents; in the latter case no other resident is assigned to h in any stable matching.

(iii) Each resident is assigned in this matching to his worst stable partner [2].

We will build on this background in section 4 where we use the extended Gale-Shapley algorithm in our application to the problem of clone detection.

3. CLONE DETECTION

3.1. OVERVIEW

Clone detection is a crucial field that has been intensively conducted by researchers and practitioners for the last two decades to enhance a software systems work and therefore, improves the maintainability for the future lifespan of the software system. Although the clone detection is a wide spread research problem over many years; is considered as a fuzzy terminology, since the researchers have differently defined it according to variants situations and criteria. Thus, it is essential to understand the meaning of clones and its uses to know how to deal with it properly? In this section, we provide different definitions and types of clones.

3.2. CLONE RELATION TERMS

Clone usually detected as a form that terms as one of either clone pair or clone classes. These two terms focus on the similarity relation between two or more pieces of cloned code. Kamiya et al. in [7] describe this relation as an equivalence relation (i.e., a reflexive, transitive, and symmetric relation). It can be said that there is a clone-relation between two fragments of code if (and only if) they have the same sequences (original characters strings, strings without whitespaces, token type etc.) From figure 3 below we can express the meaning of clone pair and clone classes based on the clone relation:

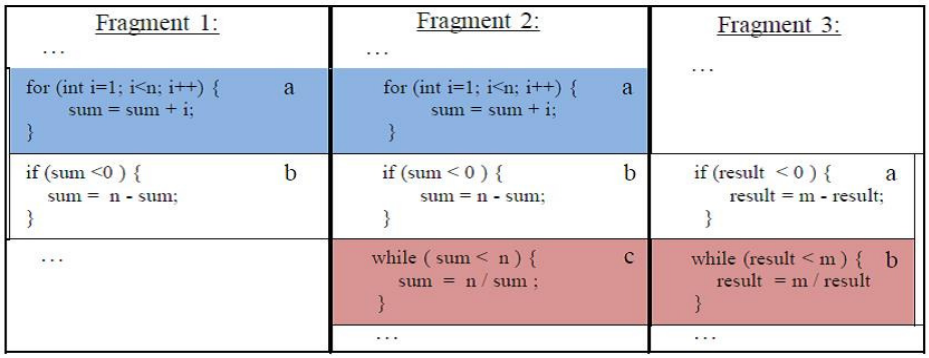

Figure 3. Clone pair and Clone class. [8]

- **Clone Pair:** two fragments of code are considered to form a clone pair when they have a clone-relation between them. That means these two portions are either identical or similar to each other. As seen in the figure 3 for the three code fragments, Fragment 1 *(F1)*, Fragment 2 *(F2)* and Fragment 3 *(F3)*, we can get five clone pairs, *(F1(a), F2(a))*, *(F1(b), F2(b)), (F2(b), F3(a)), (F2(c), F3(b))* and *(F1(b), F3(a))*. If we assume to extend the granularity size of cloned fragments, we get basically two clone pairs, *(F1(a + b), F2(a + b))* and *(F2(b + c), F3(a + b))*. And if we consider the granularity not to be fixed, we get seven clone pairs, *(F1(a), F2(a)), (F1(b), F2(b)), (F2(b), F3(a)), (F2(c), F3(b)), (F1(b), F3(a)), (F1(a+b), F2(a+b))* and *(F2(b + c), F3(a + b))*; each of these fragments is termed as a simple clone [9].

- **Clone Class:** is a maximal set of related portions of code that contains a clone pairs. It can be seen that the three code fragments of Figure 3, we get a clone class of *(F1(b), F2(b), F3(a))* where the three code portions *F1(b), F2(b)* and *F3(a)* form clone pairs with each other *(F1(b), F2(b)), (F2(b), F3(a))* and *(F1(b), F3(a))* result in three clone pairs. Consequently, a clone class is the union of all clone pairs which have portions of code in common [10,11].

- **Clone Communities:** as termed in [12] , it is another name of the Clone classes that reflecting the aggregation of related code fragments which form a clone pairs.

- **Clone Class Family:** researchers in [10] revealed the term of clone class family to group or aggregation of all clone classes that have the same domain.

- **Super Clone:** as have been outlined by [13] multiple clone classes between the same source entities (subsystems or clone classes) are aggregated into one large super clone which is the same as the clone class family.

- **Structural Clones:** it is an aggregation of similar simple clones that spread in different clone classes in the whole system [9] . Therefore, it can be classified as both a class clone (in early stage of clustering similar fragments of code) and super clone.

3.3. DEFINITION OF CODE CLONING

As aforementioned there is no original or specific definition of cloned code and therefore, all anticipated clone detection methods have their own definition for code clone [14,15] (Lakhotia, Li et al. 2003, Kontogiannis 1997). However, a fragments of code that have identical or similar

code fragments in the source code, considered to be a code clones. Regardless the changes that have been applied on a certain code clone, if still in the thresholds of the copied portion, then both the original and the copied fragments term as code clones and they form a clone pair.

Some researchers based their definition of clone code on some definition of Similarity whereas there is no specified definition of detection independent clone similarity. [16] (Baxter, Yahin et al. 1998) defined code clones as the fragments of code that are similar based on definition of similarity and they provide a threshold-based definition of tree similarity for near-miss clones. However, there is a fuzziness of the term similarity; what is meant by similar? , and to what extend are they similar? The definition provided by (Kamiya, Kusumoto et al.2002) zooms in this terminology as they define the clones as the segments of source files that are identical or similar to each other. Another ambiguous definition is proposed by [11,17] (Burd, Bailey 2002, Roy, Cordy 2007) in which fragment of code called clone when there is more existences of that fragment in the source code with or without minor modifications. However, a number of researchers [15,18, 19] (Kontogiannis 1997, Li, Lu et al. 2006, Kapser, Godfrey 2004) tried to control and specify their own detection dependent threshold based definition of the term similarity. Therefore, after several comparisons that run-out by [11,15,20] (Roy, Cordy 2007, Kontogiannis 1997, Bellon, Koschke et al. 2007) they attempt to automatically unify the result sets of multiple detectors, trying to solve the differential detector-based output.

4. EXTENDED SMP ALGORITHM (DUAL-MULTI-ALLOCATION) FOR CLONE DETECTION

4.1. OVERVIEW

SMP has solved several similar optimisation issues in different fields such as matching jobs to the most suitable jobseekers. Since the original SMP algorithm allows only the candidates of the first set (Men) to propose to their first choices, this research devotes to increase the fairness of SMP by allowing the candidates of the second set (Women) to make their own choices i.e. proposes to the best of their choices of the opposite set. The proposed approach considers a dual multi allocation technique that allows the candidates of both first and second set to enter the competition and propose again for a certain times to their preferences. So, each candidate of the first set may have more than one matched participants of the second set and vice versa. This adaption has enhanced the precision of the matching process; it is illustrated in figure 4 below. In the main SMP algorithm the desire is not controlled by the similarity, thus the assigned candidates are not meant that they are similar to each other. However, in clone detection the concept of similarity is essential. Therefore, aforementioned extension of the current state of SMP is necessary to be effectively applied in such applications. A novel matching scheme is needed to achieve smart interaction between the code fragments of the matched source files. This widens the spot to detecting every possible clone.

Practically, this process gives more than one stable matched pairs; respectively Hospital-Oriented-man and Hospital-Oriented-woman. Thus, we enclose a novel way of assigning the related code portions by adding a choosy strategy. This strategy helps to choose the pairs which form similar code clones to a certain threshold.

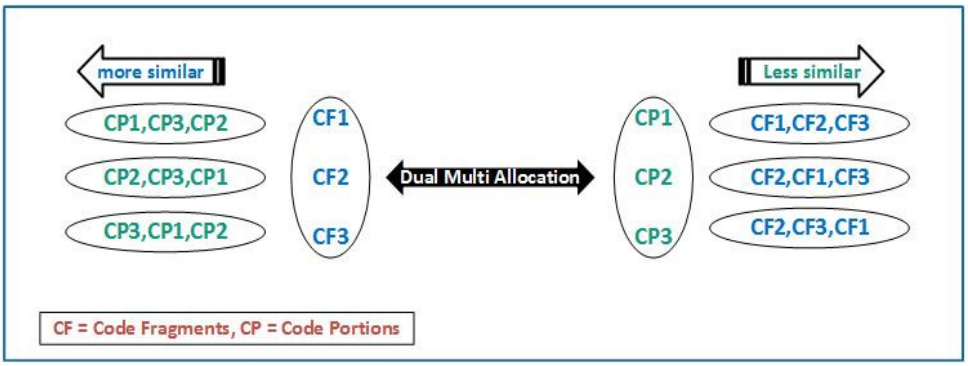

Figure 4. Dual Multi Allocation.

4.2. DUAL MULTI ALLOCATION ALGORITHM

This algorithm results in several stable matching pairs with dissimilar allocated candidates based on love's degree, which can be controlled to reach a certain level of desires. However, the matching process can be fixed as default to retrieve candidates of the highest rank love's degree factor.

The algorithm of Dual Multi Allocation consists out of two phases followed by the Choosy Strategy as following:

- **Phase 1 Hospital-Oriented-Man algorithm.**
- **Phase 2 Hospital-Oriented-Woman algorithm.**
- **Apply Choosy Strategy.**

Algorithm 3 Dual Multi Allocation algorithm

1: assign each person to be free
2: **while** (some man m is unallocated) and (m's list contains a woman w not allocated to m) **do**
3: begin
4: w:= first woman on m's list;
5: **if** w is already allocated, say to m', **then**
6: break the allocation of w to m';
7: assign w to m;
8: **for each** successor m' of m on w's list **do**
9: remove m' and w from each other's lists
10: end;
11: assign $M1$ to the Hospital-oriented-man pair or set of pairs;
12: assign each person to be free
13: **while** (some woman w is unallocated) and (w's list contains a man m not allocated to w) **do**
14: begin
15: m:= first man on w's list;
16: **if** m is already allocated, say to w', **then**
17: break the allocation of m to w';
18: assign m to w;
19: **for each** successor w' of w on m's list **do**
20: remove w' and m from each other's lists
21: end;
22: assign $M2$ to the Hospital-oriented-woman pair or set of pairs;
23: apply Choosy Strategy on $M1$ and $M2$
24: end;

4.3. CHOOSY STRATEGY

In the current state of the SMP algorithms, there are no needs to judge between two pairs to be chosen as an optimal pair. However, if a strategy to choose the optimal pair is raised, a competitive Choosy Strategy to support the newly built extension to help choosing the optimal pair is needed.

The choosy strategy formed out of two main factors, respectively, love's degree and contrast's degree. Love's degree reflects the degree of love from the view of both involved candidates (code fragment). To converge these views, we add the degree of love for both of participated (in the same pair) candidates and divide the result by two. The final result is the love's degree of the pair. The contrast's degree reflects the difference between the actual love's degree of the involved candidates. Thus, the most preferable pair is that with small difference in its contrast's degree. This factor helps when two different pairs has the same love's degree. Also, when more than one candidate has the same love's degree with a certain candidate, then the right candidate will be chosen. Figure 5 depicts the choosy strategy scheme.

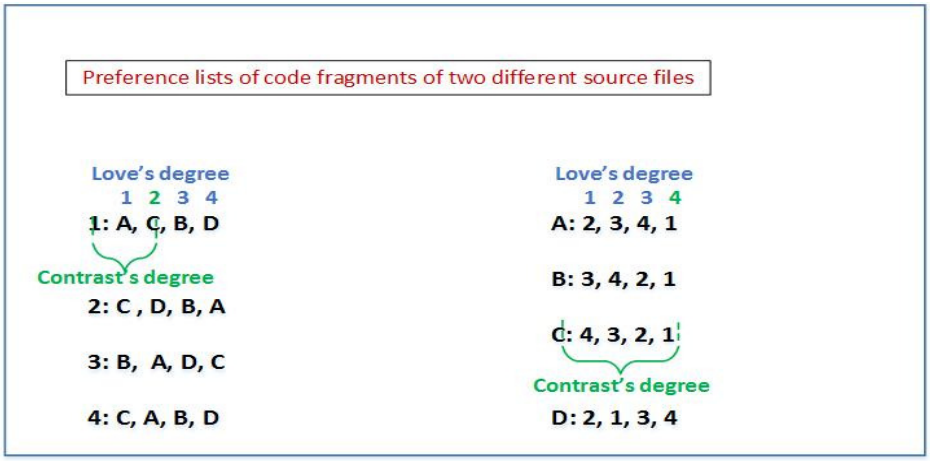

Figure 5. Choosy Strategy Scheme

4.4. SMP-BASED CLONE DETECTION

To apply the SMP algorithm in clone detection, it needs first to build the preference lists of both code fragments. This can be achieved using predefined metrics to specify the most similar related participants (code clone). Each code portion needs to strictly order the code fragments based on the similarity and vice versa. The traditional SMP algorithm performs a single assignment (one-to-one) for the involved candidates, which does not help especially in the case of allocating more than one code portion (method etc.) to the related code fragments of other source file. Multi Dual Allocation algorithm has been proposed to fulfil this requirement which widely needed in such fields. Figure 6 depicts a general prototype of code clones (method-based).

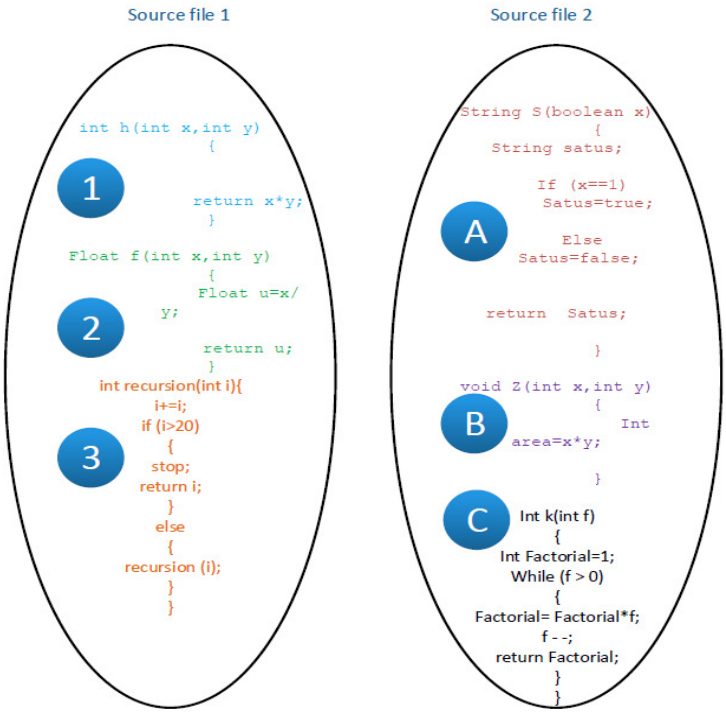

Figure 6. General Example of clones (method-based).

We have considered some metrics for fixed granularity of method based and calculate it using java plug-in with eclipse 1.3.3 (metrics 1.3.6). Figure 7 shows some of these metrics.

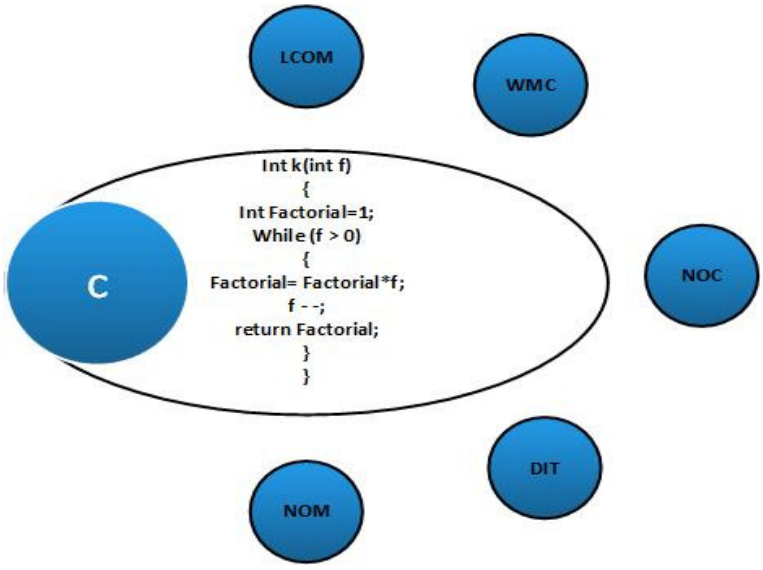

Figure 7. General view of metrics (method-based).

Table 1. Coupling Metrics

Abbreviations	Description
PROM	Number of protected methods
PUBM	Number of public methods
PRIM	Number of private methods
MCIN	Number of calls to a method
MCOUT	Number of calls from a method

Table 2. Method Metrics

Abbreviations	Description
LOC	Number lines of code
Nbp	Number of parameters
Nbv	Number of variables declared in the
Mca	Afferent coupling at method level
Mce	Efferent coupling at method level
CC	McCabe's Cyclomatic Complexity
NBD	Nested Block Depth

To apply the SMP algorithm we consider two main phases **Phase1**, building the preference list of each code fragment of first source file from the second source file's code portions, recording the most desired block and so on, repeats this process from second to first source files. **Phase2**, applying the adapted SMP algorithm based on the given metrics values. Figure 8 shows an assigned code fragment of the first source file to the most suitable (similar) code portions using the adapted SMP algorithm based on the metrics value.

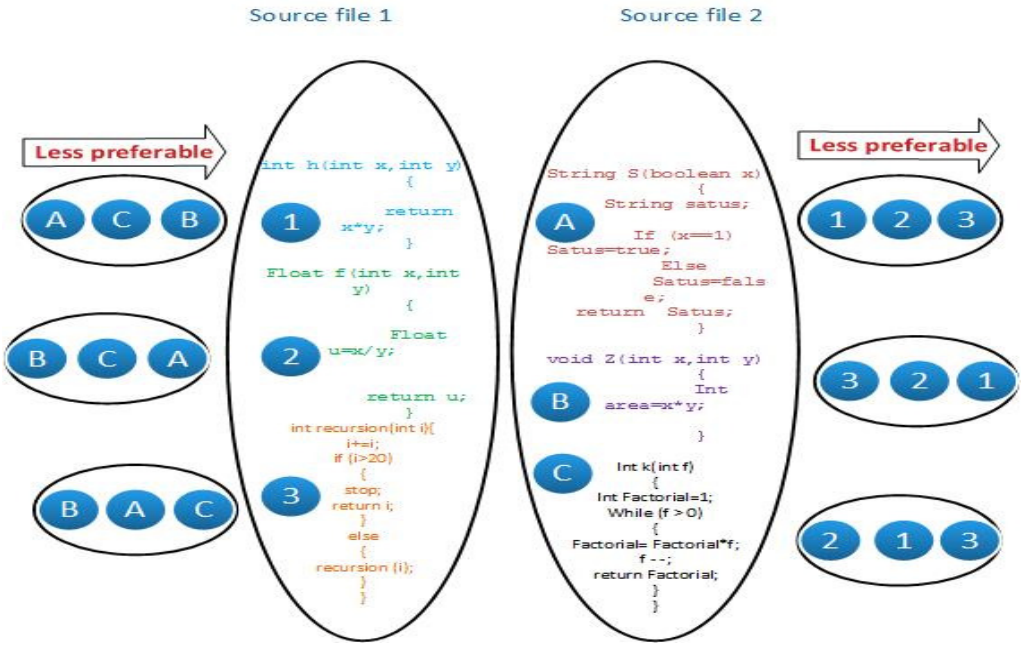

Figure 8. Example of clone detection SMP-based.

Table 3. Source file 1

Method	Metrics						
	LOC	Nbp	Nbv	Mca	Mce	CC	NBD
A	6	1	1	0	0	2	1
B	4	2	1	0	0	1	1
C	6	1	1	0	0	2	2

Table 4. Source file 2

Method	Metrics						
	LOC	Nbp	Nbv	Mca	Mce	CC	NBD
1	1	2	0	0	0	1	1
2	1	2	1	0	0	1	1
3	10	1	0	0	0	2	2

4.5. DISCUSSION

We proofed a remarkable efficiency of our approach by carrying out a case study on two middle size source files, each file with a minimum of 100 of specified blocks. Also, a set of metrics are predefined to determine the specs of each fragment of code, which help each candidate to build up its own preference list in order to apply the SMP algorithm. We observing some appointed features for the extended algorithm (e.g. performance) and the status of the detected clones (e.g. accuracy). This means that we are now able to develop match making code fragments that not only decide on the basis of the candidates' preferences of the first source file, but are actually trying to, within the current set of code fragments of both source files, to optimise the pairings from both perspectives fairly. Also, allowing the relation of many-to-many has increased the range of clones (high recall, high precision) that undetectable with most of previous clone detection approaches. However, the time complexity is challenging in this newly adapted algorithm, which still the same as the original SMP (polynomial time).

5. CONCLUSION

Stable marriage problem are well-known common matching algorithms, helped in several applications in different aspects of live for instance assigning medical schools graduates students to the most suitable hospitals. The paper presented a newly crucial extension which effectively touches a wide range of software engineering fields such as clone detection. Our contribution in this paper is the choosy strategy, which compromises between the preferences of the code fragments of two matched source files in clone detection process. Also, helped to increase the quality of retrieved code clones, through considering the desire of the matched candidates, which results in increased the satisfaction of the candidates in each pair. However, the proposed scheme has some limitations in terms of its complexity and would require longer time to reach the highly required stability. In our future work we would like to look at how dynamically scale the levels of metrics to consider different abstraction levels (e.g. package, class etc.).

REFERENCES

[1] Gale, David & Lloyd S. Shapley, (1962) "College Admissions and the Stability of Marriage", JSTOR, pp 9-15.

[2] Gusfield, Dan & Robert W. Irving (1989) The Stable Marriage Problem: Structure and Algorithms, MIT press Cambridge.

[3] Biró, Péter & Eric McDermid, (2011) "Matching with Sizes (Or Scheduling with Processing Set Restrictions)", Elsevier, .
[4] iQua, "Stable Matching in Networking." http://iqua.ece.toronto.edu/spotlights/matching/ (accessed 03/15, 2014).
[5] The match "National Resident Matching Program." http://www.nrmp.org/ (accessed 03/15, 2014).
[6] Cheng, Christine, Eric McDermid, & Ichiro Suzuki, (2008) "A Unified Approach to Finding Good Stable Matchings in the Hospitals/Residents Setting", Elsevier, Vol. 400, No. 1, pp 84-99.
[7] Kamiya, Toshihiro, Shinji Kusumoto, & Katsuro Inoue, (2002) "CCFinder: A Multilinguistic Token-Based Code Clone Detection System for Large Scale Source Code", IEEE, Vol. 28, No. 7, pp 654-670.
[8] Roy, Chanchal K. , (2009) "Detection and Analysis of Near-Miss Software Clones", IEEE, pp 447-450.
[9] Basit, Hamid Abdul & Stan Jarzabek, (2009) "A Data Mining Approach for Detecting Higher-Level Clones in Software", IEEE, Vol. 35, No. 4, pp 497-514.
[10] Rieger, Matthias, Stéphane Ducasse, & Michele Lanza, (2004) "Insights into System-Wide Code Duplication", IEEE, pp 100-109.
[11] Roy, Chanchal Kumar & James R. Cordy, (2007) Citeseer, Vol. 541, pp 115.
[12] Mayrand, Jean, Claude Leblanc, & Ettore M. Merlo, (1996) "Experiment on the Automatic Detection of Function Clones in a Software System using Metrics", IEEE, pp 244-253.
[13] Jiang, Zhen Ming, Ahmed E. Hassan, & Richard C. Holt, (2006) "Visualizing Clone Cohesion and Coupling", IEEE, pp 467-476.
[14] Lakhotia, Arun, Junwei Li, Andrew Walenstein, & Yun Yang, (2003) "Towards a Clone Detection Benchmark Suite and Results Archive", IEEE, pp 285-286.
[15] Kontogiannis, Kostas. , (1997) "Evaluation Experiments on the Detection of Programming Patterns using Software Metrics", IEEE, pp 44-54.
[16] Baxter, Ira D., Andrew Yahin, Leonardo Moura, Marcelo Sant'Anna, & Lorraine Bier, (1998) "Clone Detection using Abstract Syntax Trees", IEEE, pp 368-377.
[17] Burd, Elizabeth & John Bailey, (2002) "Evaluating Clone Detection Tools for use during Preventative Maintenance", IEEE, pp 36-43.
[18] Li, Zhenmin, Shan Lu, Suvda Myagmar, & Yuanyuan Zhou, (2006) "CP-Miner: Finding Copy-Paste and Related Bugs in Large-Scale Software Code", IEEE, Vol. 32, No. 3, pp 176-192.
[19] Kapser, Cory & Michael W. Godfrey, (2004) "Aiding Comprehension of Cloning through Categorization", IEEE, pp 85-94.
[20] Bellon, Stefan, Rainer Koschke, Giuliano Antoniol, Jens Krinke, & Ettore Merlo, (2007) "Comparison and Evaluation of Clone Detection Tools", IEEE, Vol. 33, No. 9, pp 577-591.

AUTHORS

Hosam Al Hakami received his B.Sc. degree in Computer Science from King Abdulaziz University, Saudi Arabia. He received his MSc degree in Internet Software Systems from Birmingham University, Birmingham, UK. He is studying now towards his PhD degree at the Faculty of Technology in De Montfort University, Leicester, UK.

Dr.Feng Chen was awarded his BSc, Mphil and PhD at Nankai University, Dalian University of Technology and De Montfort University in 1991, 1994 and 2007. As research outputs, he has published over 30 research papers in the area of software evolution and distributed computing.

Dr.Helge Janicke is heading the Software Technology Research Laboratory at De Montfort University, Leicester (UK). He is leading the research theme on Computer Security and Trust. His research interests are in area of software engineering where he is primarily looking at cyber security, in particular access control and policy-based system management.