

Stable Marriage Problem Based Adaptation for Clone Detection and Service Selection



Hosam Hasan M. Al Hakami

STRL

De Montfort University

This thesis is submitted in partial fulfilment of the requirements for the
degree of

Doctor of Philosophy

March 2015

Except as otherwise permitted under the Copyright, Designs and Patents Act 1988, this thesis may only be produced, stored or transmitted in any form or by any means with the prior permission in writing of the author. The author asserts his/her right to be identified as such in accordance with the terms of the Copyright, Designs and Patents Act 1988.

Declaration

I declare that the work described in this thesis was originally carried out by me during the period of registration for the degree of Doctor of Philosophy at De Montfort University, Leicester, United Kingdom, from October 2009 to November 2014. It is submitted for the degree of Doctor of Philosophy at De Montfort University. Apart from the degree that this thesis is currently applying for, no other academic degree or award was applied for by me based on this work.

Hosam Hasan M. Al Hakami

March 2015

Acknowledgements

In the name of Allah, the Most Merciful and the Most Gracious, I gave praise and thanks to Him for supporting me with the strength to complete this thesis, and for providing me with knowledgeable and caring individuals during the study process.

This thesis could not have been completed without the recommendations, advice and encouragement of many people.

Special thanks to my supervisor Feng Chen, who oversaw this thesis and offered wise direction. I am sincerely thankful and appreciative of this invaluable assistance.

I would like to thank my second supervisor Helge Janicke for his constant guidance, personal attention, suggestions, endless encouragement and full support during the last three years of my study.

I also extend my sincere thanks and appreciation to my colleagues at the Faculty of Technology, who gave me guidance and advice through my study. Special thanks to my wife, for her love, patience and support during my whole study, and to my lovely daughter Remas. I also wish to express thanks to my mother and father for their love and encouragement.

Finally, I would like thank to all of my relatives and friends, who have always been there when I needed them most.

Publications

- **Al Hakami, H**, Aldabbas, H., & Alwada'n, T. (2012). Comparison between Cloud and Grid computing: Review Paper. *International Journal on Cloud Computing: Services and Architecture (IJCCSA)*, 2(4):1–21.
- **Al Hakami, H** & Chen, F (2014) "SMP-based Dual Proposed Matching Scheme for Service Selection", *7th Saudi Students Conference-UK*, Edinburgh, UK.
- **Al Hakami, H**, Chen, F., & Janicke, H (2014) "SMP-based Clone Detection", *Second International Conference on Software Engineering*, pp 89-101. AIRCC.
- **Al Hakami, H**, Chen, F., & Janicke, H (2014) An Extended Stable Marriage Problem Algorithm for Clone Detection, *International Journal of Software Engineering & Applications (IJSEA)*, 5(4):103-122.
- **Al Hakami, H**, Chen, F., & Janicke, H (2014) "SMP-based Service Matching", *Science and Information Conference*, pp 620-625. IEEE.
- **Al Hakami, H**, Chen, F., & Janicke, H SMP-Based Approach for Intelligent Service Interaction: submitted to *International Journal of Advanced Computer Science and Applications (IJACSA)*.

Abstract

Current software engineering topics such as clone detection and service selection need to improve the capability of detection process and selection process. The clone detection is the process of finding duplicated code through the system for several purposes such as removal of repeated portions as maintenance part of legacy system. Service selection is the process of finding the appropriate web service which meets the consumer's request. Both problems can be converted into a matching problem.

Matching process forms an essential part of software engineering activities. In this research, a well-known mathematical algorithm Stable Marriage Problem (SMP) and its variations are investigated to fulfil the purposes of matching processes in software engineering area. We aim to provide a competitive matching algorithm that can help to detect cloned software accurately and ensure high scalability, precision and recall. We also aim to apply matching algorithm on incoming request and service profile to deal with the web service as a clever independent object so that we can allow the services to accept or decline requests (equal opportunity) rather than the current state of service selection (search-based), in which service lacks of interacting as an independent candidate.

In order to meet the above aims, the traditional SMP algorithm has been extended to achieve the cardinality of many-to-many. This adaptation is achieved by defining the selective strategy which is the main engine of the new adaptations. Two adaptations, Dual-Proposed and Dual-Multi-Allocation, have been proposed to both service selection and clone detection process. The proposed approach (SMP-based) shows very competitive results compare to existing software clone approaches, especially in identifying type 3 (copy with further modifications such update, add and delete statements) of cloned software. It performs the detection process with a relatively high precision and recall compare to the CloneDR tool and shows good scalability on a middle sized program. For service selection, the proposed approach has several advantages such as service protection and service quality. The services gain equal opportunity against the incoming requests. Therefore, the intelligent service interaction is achieved, and both stability and satisfaction of the candidates are ensured.

This dissertation contributes to several contributions firstly, the new extended SMP algorithm by introducing selective strategy to accommodate many-to-many matching problems, to improve overall features. Secondly, a new SMP-based clone detection approach to detect cloned software accurately and ensures high precision and recall. Ultimately, a new SMP-based service selection approach allows equal opportunity between services and requests. This led to improve service protection and service quality.

Case studies are carried out for experiments with the proposed approach, which show that the new adaptations can be applied effectively to clone detection and service selection processes with several features (e.g. accuracy). It can be concluded that the match based approach is feasible and promising in software engineering domain.

Table of Contents

List of Figures	xvii
List of Tables	xxi
Nomenclature	xxii
1 Introduction	1
1.1 Background & Motivation	2
1.2 Research Objectives	3
1.3 Research Questions	4
1.4 Original Contributions	4
1.5 Research Methodology	5
1.6 Criteria of Success	8
1.7 Thesis Outline	9
2 Background and Related Research	11
2.1 Introduction	12
2.2 Stable Matching Problem	13

Table of Contents

2.2.1	Overview	13
2.2.2	Stable Marriage Problem	15
2.2.2.1	The Gale-Shapley Algorithm	15
2.2.2.2	Optimal Stable Marriage Problems	22
2.2.2.3	Stable Marriage with Incomplete Lists (SMI)	23
2.2.2.4	Stable Marriage with Ties (SMT)	24
2.2.2.5	Stable Marriage with Ties and Incomplete List	26
2.2.3	Hospitals/Residents Problem	26
2.2.4	Stable Roommates Problem	28
2.3	Clone Detection	30
2.3.1	Overview	30
2.3.2	Clone Relation Terms	30
2.3.3	Definition of Code Cloning	33
2.3.4	Detection Techniques and Tools	34
2.3.4.1	Text-based Technique	35
2.3.4.2	Metrics-based Technique	37
2.3.4.3	Token-based Technique	42
2.3.4.4	Tree-based Technique	43
2.3.4.5	PDG-based Technique	46
2.4	Service Selection	47
2.4.1	Overview	47
2.4.2	Qos-based Service Selection	48

2.4.3	Semantic Matching	49
2.4.4	Service Availability	51
2.5	Search-Based Optimisation	53
2.5.1	Overview	53
2.5.2	SBSE Ingredients	54
2.5.3	Common Search Algorithms	55
2.5.3.1	Hill Climbing	55
2.5.3.2	Simulated Annealing	56
2.5.3.3	Genetic Algorithms	56
2.6	Conclusion	57
3	SMP Extensions	59
3.1	Overview	60
3.2	Dual Proposed	63
3.2.1	Overview	63
3.2.2	Dual Proposed Algorithm	64
3.2.3	Selective Strategy	66
3.2.4	Semantic Equivalence	71
3.2.5	Evaluation	72
3.3	Dual Multi Allocation	73
3.3.1	Overview	73
3.3.2	Dual Multi Allocation Algorithm	74
3.4	Conclusion	76

Table of Contents

4	SMP-Based Clone Detection	79
4.1	Overview	80
4.2	Similarity Measurements	80
4.3	Metrics	81
4.4	Detection Process	82
4.5	SMP-based Clone Detection	84
4.5.1	The Scenario	89
4.5.1.1	The stage of Building the Preference Lists	89
4.5.1.2	The Stage of Running the SMP Algorithm	93
4.5.1.3	Extended SMP Algorithm (Dual-Multi-Allocation) for Clone Detection	94
4.5.2	Discussion	95
4.6	Conclusion	96
5	SMP-Based Service Matching	99
5.1	Overview	100
5.2	Dual-Proposed for Service Selection	101
5.2.1	Dual-Proposed Scheme	101
5.2.2	Service Matching	103
5.3	Cloud Service Availability	107
5.4	Discussion	109
5.5	Conclusion	109

6	Evaluation	111
6.1	Clone Detection	112
6.1.1	Introduction	112
6.1.2	Case Study (Job Search System)	113
6.1.2.1	Overview	113
6.1.2.2	Related Technology	114
6.1.2.3	System Design	115
6.1.3	Clone Detection Experiment	116
6.1.3.1	Recall and Precision	120
6.1.3.2	Discussion	123
6.2	Service Selection	125
6.2.1	Case Study	125
6.2.1.1	A Hotel Reservation Example	125
6.2.1.2	Discussion	127
6.2.2	Experiment for Service Selection	129
6.3	Conclusion	132
7	Conclusion and Future Work	135
7.1	Summary of the Thesis	136
7.2	Contributions Revisited	136
7.3	Success Criteria Revisited	138
7.4	Thesis Limitations	139
7.5	Future Work	139

Table of Contents

References	141
Appendix A Screen-shots	151
A.1 Clonesets Snapshots	154
Appendix B Source Code	155

List of Figures

2.1	Instance C1 of SMP	18
2.2	Instance C2 of SMP	20
2.3	A Stable Marriage Instance of Size 4	21
2.4	Case C of SMI	23
2.5	Case C of SMT	25
2.6	Clone pair and Clone class	31
2.7	Normalization operations on source code elements	36
2.8	Tools and calculated metrics	40
2.9	Service Selection process	48
2.10	Description Framework Of Fqos	50
2.11	Overall Architecture of SBSE Approach	55
3.1	Dual Propose Technique	63
3.2	love's degree	66
3.3	Selective Strategy Scheme	68
3.4	Example of selective strategy	69

List of Figures

3.5	Semantic Equivalence Candidates	71
3.6	Semantic Equivalence Candidates (SMP-based)	72
3.7	Multi-proposed Technique	73
4.1	Detection Process for Program Analysis	83
4.2	Metrics level-based	85
4.3	Detection Process for Program Analysis	87
4.4	High presentation of code fragment along with its measurements	88
4.5	Detection Process for Program Analysis	89
4.6	Methods preference	93
4.7	Dual Multi Allocation.	95
5.1	Dual Proposed	102
5.2	Services Matching with Dual Propose Algorithm	103
5.3	A stable marriage instance of size 8	105
5.4	Service Matching in Cloud.	108
6.1	Three-tier architecture	115
6.2	Three-tier description of the project	116
6.3	Number of clone candidates for program netbeans-javadoc and job search project.	118
6.4	Number of clone candidates for program netbeans-javadoc and job search project.	119
6.5	Number of detected software clones for job search project.	120

6.6	Recall and Precision	121
6.7	Recall.	123
6.8	Precision.	124
6.9	General Example of web services	126
6.10	SMP-based service selection	127
6.11	Hotel Reservation Request	128
6.12	Hotel Reservation Service.	129
6.13	Hotel Reservation Example.	130
6.14	Execution of the main SMP Service Optimal.	131
6.15	Execution of the main SMP Request Optimal.	132
6.16	Execution of the dual-proposed SMP.	133
A.1	Job search project browsed by CloneDR	151
A.2	Some statistics of job search project based on CloneDR	152
A.3	Job search project analysed by CloneDR	152
A.4	Some statistics of netbeans-javadoc project based on CloneDR	153
A.5	netbeans-javadoc project analysed by CloneDR	153
A.6	Clone set of type 1	154
A.7	Clone set of type 2	154

List of Tables

2.1	Metrics-based Detection Techniques.	41
2.2	Detection Methods of Token-based.	44
2.3	Detection Methods of Tree-based.	45
2.4	Detection Methods of PDG-based.	47
2.5	Obstacles to and opportunities for growth of cloud computing.[10]	52
3.1	New Man-Oriented	70
3.2	New woman-Oriented	70
4.1	Coupling Metrics.	90
4.2	Method Metrics.	90
4.3	Metrics of Source file 1.	91
4.4	Metrics of Source file 2.	91
4.5	Metrics Weight.	92
5.1	ServiceView	104
5.2	RequestView	104

List of Tables

5.3	Service-view	106
5.4	Request-view	106
6.1	Job search system	117
6.2	Snapshot of some metrics in the job search system	117
6.3	Number of actual clone for program netbeans-javadoc and job search project.	118
6.4	Number of clone candidates for program netbeans-javadoc and job search project.	119
6.5	Number of detected software clones for job search project.	120

Chapter 1

Introduction

Objectives

- To provide an overview of the research problems and motivations.
 - To identify the scope of the thesis.
 - To present the research objectives and questions.
 - To describe the criteria of success.
 - To describe the research methodology.
 - To introduce the outline of the thesis.
-

1.1 Background & Motivation

Matching processes are the corner stones of the widely spread software applications, in which the software approaches can be judged as perfect (fulfil the standard requirement) or poor. Current software areas such as clone detection and service selection need to increase the capability of its match process to accomplish the intelligence assignment between its candidates. Stable Marriage Problem (SMP) algorithm is one of the well-known mathematical matching algorithms, which is used for multiple purposes and several criteria with different derived versions[55]. Therefore, with some amendments to this algorithm in order to take place and replace the current matching process of such software applications, the intelligent is achieved.

The matching process performance needs to be challenging in several aspects of criteria for instance speed, accuracy, flexibility, stability and intelligence (satisfied matched candidates). Especially to accommodate the SMP algorithm to wide range of software matching processes, the current state of the SMP algorithm lacks the satisfaction of both matched candidates and increase the fairness of the matching process. Therefore, the original SMP algorithm need to be adapted to suit wide range of software engineering disciplines.

Clone detection is the process of finding duplicated code through the system for several purposes such as removal of repeated portions as maintenance part of legacy system. On the other hand, service selection is the process of finding the appropriate web service which meets the consumer's request. Basically, the service listed as a WSDL file in the directory UDDI to be picked up by the request. The current service selection process performs a simple search-based technique which only considers the consumer's view.

This needs a proper engine to increase the performance of these processes, as for example, current clone detection process lacks to identify some cloned software types. Also, current service selection process lacks to a suitable syntactic matching-based behaviour that should ensure an independent interaction between the services and requests. As a result, stable marriage problem may be applied in order to improve the overall process in both topics.

1.2 Research Objectives

This research aims to investigate a well-known mathematical algorithm Stable Marriage Problem (SMP algorithm) and its variations to fulfil the purposes of matching processes in software engineering area. The research objectives presented in this thesis are as follows:

- To improve the current status of the clone detection process to increase its overall performance, as most of the current approaches lack of detecting some types of software clones.
- Also, to improve the process of service selection by introduce matching-based approach to ensure an intelligent service interaction between fully independent services and incoming clients' requests.
- To evaluate our proposed approach and demonstrate that it can be ideally used in different software engineering disciplines such as clone detection and service selection.

1.3 Research Questions

The principle research question in this study is:

- **How to effectively modify the current state of SMP algorithm to be efficiently applied in Software Engineering area?**

Based on this research question we investigate some disciplines in software engineering and therefore, two sub questions are formulated as follows:

- Does the new adapted algorithm SMP-based helps in detecting cloned software (Clone Detection)?
- Does the new adapted algorithm SMP-based improves the current service selection process?

1.4 Original Contributions

This dissertation mainly contributes to the following:

- The new extended SMP algorithm by introducing selective strategy to accommodate many-to-many matching problems, to improve some features such as precision, fairness and satisfaction (see Chapter 3).
- A new SMP-based clone detection approach to detect cloned software accurately and ensure high scalability, precision and recall (see Chapter 4).

- A new SMP-based service selection approach to allow equal opportunity between services and requests, to improve service protection and service quality, rather than the current state of search-based service selection (see Chapter 5).

1.5 Research Methodology

The cornerstone of setting off any type of research is the choosing of the most appropriate research method. It is a crucial step as it makes the whole work as organized as possible and presented in precise way leading to facilitate the research to the researchers. However, due to the different domains, the research methods are varied. [17, 39, 91] Following is a brief description about some common research methods:

- **Experimental Research:** This type of research method usually is used to test a proposed hypothesis.
- **Qualitative Research:** This type of research method required the researcher to directly involve in performing some activities with people to consider their opinion using conversation and direct observation.
- **Exploratory Research:** This type of research considered when new problem of a domain is identified.
- **Quantitative Research:** This type of research reflects the objectivity of the researcher as an independent of the research process, the researcher in this method usually design a questionnaire form.

- **Action Research:** This type of research intended to find an urgent solution for unexpected problem for example controlling unknown dangerous virus.
- **Conceptual Research:** This type of research method is related to ideas and theory to develop new concepts.
- **Empirical Research:** This type of research method tests a wide range of empirical studies to prove the utility of the known solution on a large scale.
- **Fundamental Research:** Also called pure research or basic research, this type of research method focuses on expanding knowledge, without specific applications or products in mind.
- **Non-experimental Research:** in this type of research method the hypothesis is not important to be considered.

In computer science domain, as it is usually in the form of research projects, all of the aforementioned types of research methods above are not suitable. Research projects in computer science domain focuses on developments of new methods and systems. Therefore, as stated by [27, 77], the following research method which is known as constructive research method takes place in development of solutions for previously identified problem situation. This type of constructive research method refers to contributions to knowledge being developed as new algorithms, frameworks, models, or techniques.

The constructive research approach is a research procedure for producing innovative constructions (novel) which help to tackle issues faced in the real world and, by that means,

to make a contribution to the theory of the discipline in which it is applied. By developing a construction, something that differs profoundly from anything, which existed before, is created: novel constructions bring forth, by definition, new reality.

In philosophy, the application of constructive research can be found in those cases where the world is constructed, step by step, from supposedly basic elements like objects, time-space slices, observations, thoughts, or logical relations. Mathematical algorithms and new mathematical entities provide theoretical examples of constructions. In medicine it can be found that the constructive approach in the development of new treatment or medicine [64]. Computer languages - is an example of developing a pure construction.

The need to the constructive research approach has several aspects appear when:

- implies a very close involvement and co-operation between the researcher and practitioners in a team-like manner, in which experiential learning is expected to take place,
- focuses on real-world problems felt relevant to be solved in practice,
- produces an innovative construction meant to solve the initial real-world problem,
- is explicitly linked to prior theoretical knowledge, and
- pays particular attention to reflecting the empirical findings back to theory [85].
- includes an attempt for implementing the developed construction and thereby a test for its practical applicability.

As long as our research focuses on how to improve a current mathematical algorithm to be used for solving some existing issues with different fields, the constructive method approach is used for this purpose. This research constructs a novel strategy “selective strategy” which is the main engine of the proposed adaptations in this thesis. The constructive methodology helps to validate the proposed framework which trying to tackle real problems in software engineering state of art.

The constructive research approach is used in this thesis, as it fulfils the requirements of this research. Our approach considers the significant tasks of development, implementation and evaluation.

1.6 Criteria of Success

- Adapting the current state of Stable Marriage Problem to the field of software engineering.
- Applying the extended algorithm to clone detection and increase the performance efficiency of the current clone detection techniques and emphasise the accuracy.
- Detecting type 3 (a copy with further modifications such update, add and delete statements) of software clones using both selective group of metrics and SMP-based approach.
- Applying the extended algorithm to add a sense of matching to the current state of service selection process to protect the services.

- Evaluating a medium-sized case study to demonstrate the scalability of the proposed approaches.

1.7 Thesis Outline

This section provides the proposed structure of the authors' PhD thesis.

- **Chapter 1: Introduction**

This chapter gives an overview of the problem, motivation and approach to the solution. Also, it shows the fundamentals of this thesis by introduce the research objectives, research questions, main contributions, research methodology and criteria of success.

- **Chapter 2: Background and Related Work** SMP and its derived algorithms, clone detection, selection services and search based software engineering are the main topics of this thesis. All these topics and their relevant are deeply reviewed and investigated. This chapter is a cornerstone of the thesis which shows a full knowledge about aforementioned basics.

- **Chapter 3: SMP Extensions.**

This chapter presents the development of current Stable Marriage Problem to be more suitable for Software Engineering area. This is illustrated in define the selective strategy which forms the main engine of the new adaptations. This chapter shows two extensions respectively “Dual Proposed Algorithm” and “Dual Multi Allocation Algorithm”.

- **Chapter 4: SMP-Based Clone Detection**

This chapter is concerned with designing the SMP-based clone detection framework and provides a small prototype to detect duplicated code. This is based on the introduced adaptation in chapter 3. A concrete example is introduced to explain the adapted algorithm.

- **Chapter 5: SMP-Based Service Selection.**

This chapter is concerned with correlating services and requests using the SMP-based service selection approach. This is based on the introduced adaptation in chapter 3. A concrete example is introduced to explain the adapted algorithm. The service availability to redirect coming requests to similar services is also discussed.

- **Chapter 6: Evaluation.**

This chapter shows the design of experiments and concerned with evaluating our approach against other approaches. A medium-sized experiment provided to test the SMP-based approach to clone detection and to analyse the obtained results against a solid benchmark. Also, this chapter provides an automated experiment to test the SMP-based approach to service selection.

- **Chapter 7: Conclusion and Future Work.**

This chapter concludes the whole research and provide some recommendations. Also, it shows some limitations of the proposed approach. Finally, it provides some possible future work and directions.

Chapter 2

Background and Related Research

Objectives

- To provide the background information of this thesis.
 - To present state of the art stable marriage problem algorithms.
 - To review the work related to matching schemes in several software applications.
 - To introduce state of the art search-based optimisation.
-

2.1 Introduction

In this chapter, we introduce several relevant topics to gain a solid background of the required knowledge, which helps to sharpen and improve our prospective approach. We correlate these topics to try to fit an adaptation of SMP algorithm to software engineering areas.

As long as the Stable Marriage Problem (SMP) algorithm which invented around 50 years ago, helped in several applications such as matching applicants to jobs and residents to hospitals under the cardinality of one-to-one and one-to-many, respectively. Therefore, we investigate these algorithms to try to improve their current state to ensure the cardinality of many-to-many for software context purpose.

Many researchers in [25, 99] define the types of cloned software as following:

- **Type 1:** Identical code fragments, except for variations in whitespace, layout and comments.
- **Type 2:** Syntactically identical fragments except for variations in identifiers, literals, types, whitespace, layout and comments.
- **Type 3:** Copied fragments with further modifications such as changed, added or removed statements, in addition to variations in identifiers, literals, types, whitespace, layout and comments.
- **Type 4:** Two or more code fragments that perform the same computation but are implemented by different syntactic variants.

As the current approaches of the clone detection lack of accuracy (low precision and low recall) and lack of recognising some types of cloned software such as type 3 and type 4, we investigate their current approaches, tools, techniques and used algorithms to fully understand their backgrounds. Thus, trying to accommodate the new adaptation of the SMP algorithm to solve these issues and improve the overall performance of the current detection process.

Also, we investigate the current state of art of service selection process which lack of a proper selective process, as most of the current approaches perform search-based techniques. Recent approaches of service selection consider some semantic techniques to enhance the overall selection process using ontology alignment. As long as the current approaches of service selection do not consider the view of the services provided by the service providers, it should be a way to consider both views of costumers and services. This needs a matching algorithm which can consider both parties.

2.2 Stable Matching Problem

2.2.1 Overview

It has been noticed in some crucial large-scale practical applications that there are several matching problems that need to be solved. In a matching problem there are three aspects that need to be considered; the candidates of the different sets about to assigned, preference list and capacities and stability. The preference list of the candidate is a set of opposite candidates, ordered based on the most preferable candidate over the opposite set, that he/she

prefers to assign to. The capacity of the candidate is the most allowable number of candidates that he/she can assign to.

The stability ensures that there are no two candidates that would prefer each other to current assignees. By this concept of satisfied stability the stable matching is formed. For instance finding a job among several jobs and assigning it to the right job seeker. Assigning the students to the best of their choices of which department to study in is a similar issue. Similarly, allocating the final year students to the final projects and supervisors. Add to these the process of distributing the medical graduation students to the hospitals as trainees.

All these cases could form unfair problems if the preciseness of the matchmaking process is lost. Most of these cases need an optimality to satisfy all contributors on both sides. A matchmaker is a high responsible person who controls the process of matching by applying a suitable formula on each case to accomplish the optimal matching.

The idea of stable marriage problem is about finding a matching between men and women, based on the preference lists that consist of the members of the opposite gender for each candidate. The result of the matching process must be stable, which means that there is none of the pair members both men and women have any motivation or influence to break down the relation. This problem was introduced in 1962 in the seminal paper of Gale and Shapley, and it has interested many researchers from different backgrounds such as mathematics, economics, game theory, computer science, etc.

In this chapter we discuss the stable marriage problem and its variants derived algorithms, and other related matching algorithms. Also, applicable areas in software engineering are introduced to see the possibility of use aforementioned matching algorithms in two different

levels; code level and architecture level. Moreover, we introduce semantic matching which deals with data side. In section 2.1 we express the main algorithm of stable marriage problem. The SMP algorithm involves two sets of candidates (men, women), where each candidate ranks the opposite candidates in strict order of preference.

In section 2.2 we describe another type of stable marriage problem known as the Hospitals/Residents problem which reflects different relation (many-to-one). We then move to section 2.3 to describe the Stable Roommates problem which is one of the related Stable Marriage Problems. In section 2.4 we introduce the clone detection in which similar code fragments are matched for various purposes. We briefly introduce the semantic matching and its uses of ontology alignment in section 2.5. Finally in section 2.6 we explore some search based techniques and how they differ from matching techniques.

2.2.2 Stable Marriage Problem

2.2.2.1 The Gale-Shapley Algorithm

The stable marriage problem 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 [35]. Thus, the output has to be stable, which means that the matched pair is satisfied and both of them have no incentive to disconnect. A matching 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 [18].

David Gale and Lloyd in their paper [35] in 1962 described the Stable Marriage Problem for a first time, and formally defined and introduced their algorithm as a solution. A case C of the stable mating SM involves the same number of men and women and we denote to by x . The preference list is essential in this algorithm for each man and woman, which is built by ranking all x members of the opposite gender. The set of men and set of women are indicated to in the case C as follows:

$$W = w_1, w_2, \dots, w_n$$

$$M = m_1, m_2, \dots, m_n$$

The preference lists in stable matching SM are considered as complete, when all members of the case C are ranked.

Algorithm 1 Basic Gale-Shapley algorithm[89]

```
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 to whom  $m$  has not yet proposed;
5: if  $w$  is free then
6: assign  $m$  and  $w$  to be engaged to each other
7: else
8: if  $w$  prefers  $m$  to her fiancé  $m'$  then
9: assign  $m$  and  $w$  to be engaged and  $m'$  to be free
10: else
11:  $w$  rejects  $m$  and  $m$  remains free
12: end;
output the stable matching consisting of the  $n$  engaged pairs
```

We denote to an initial pre-step to a matching M by A as assignments which is a set of pairs of woman and man $(m, w) \in MW$, if a pair $(m, w) \in A$ this means that w is assigned to a

man m in A and m is assigned to w in A . Also, it means that w is m 's partner in A and m is w 's partner in A . For example, let $A(s)$ refers to s 's spouse in A , where $s \in M \cup W$. If $A(s) \neq \emptyset$, this means that s is assigned in A , otherwise it is not assigned in A . With the same indication used with assignments, if $(m, w) \in M$ from either men-to-women or women-to-men. This means that w is matched to a man m in M and m is matched to w in M . Also, it means that w is m 's partner in M and m is w 's partner in M .

The issue of blocking pair appears when one of the partners in matching M would like to engage to another partner from different pair in matching M whom he/she prefers more than his/her current partner. This affects the whole pairs and blocks matching M . Consequently, the matching M is not stable. The following pair is considered as a blocking pair: $(m, w) \in MW$ be a blocking pairs w prefers m to $M(w)$ and m prefers w to $M(m)$.

So, we say that the pair (m, w) is stable if both w and m matched to each other in a stable matching M . Figure 2.1 shows an example case of stable matching (SM), in which the preference lists are ordered from right to left, start with the most preferable (we use this order for all stable matching (SM) examples in this thesis). $M = (m1, w1), (m2, w3), (m3, w2)$ is an example of stable matching in case C1 of Figure 2.1.

The algorithm is presented in two possible views based on who has the right to propose (men or women). These two orientations are woman-oriented algorithm and the man-oriented algorithm. In the man-oriented algorithm each man m proposes to the first woman w on his preference' list to whom he did not propose before. If the woman w is free, then she becomes engaged to the man m . Otherwise, if the woman w prefers a man m' to her current husband-to-be the man m , then she rejects her current fiancé the man m . consequently, the

Men's preferences

```
m1 : w1 w3 w2
m2 : w1 w2 w3
m3 : w2 w1 w3
```

Women's preferences

```
w1 : m1 m3 m2
w2 : m3 m1 m2
w3 : m1 m2 m3
```

Fig. 2.1 Instance C1 of SMP algorithm.

woman w becomes engaged to the new man m' and the man m becomes free. Otherwise the woman w keeps her current husband-to-be to m and rejects m' which remains free. This procedure continues until all the men engaged. The process of woman-oriented algorithm is similar to the man-oriented algorithm; the different is that the women make proposals. The algorithm relies on the strategy of “deferred acceptance” in proceeding this process.

Consequently, based on the aforementioned orientations, the output is either of man-optimal and woman-pessimal or woman-optimal and man-pessimal stable matchings. Each man in man-optimal stable matching is feature from obtaining the best partner and thus, each woman assigns to her worst partner. Accordingly, each woman in woman-optimal algorithm is feature from obtaining her best partner among stable matchings, and as a result each man obtains his worst partner.

Algorithm 2 Extended Gale-Shapley algorithm[43]

```

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;

```

The following are two possible matchings for the case C2 which illustrated in Figure 2.2:

$$M_z = (m_3, w_4), (m_2, w_1), (m_1, w_2), (m_4, w_3)$$

$$M_0 = (m_3, w_2), (m_2, w_1), (m_1, w_4), (m_4, w_3)$$

Whereas M_z and M_0 are woman-optimal and man-optimal, respectively. In some situation both man-optimal and woman-optimal are the same, like the following unique stable matching presented in case C1 in the previous example of Figure 2.1:

$$M_z = M_0 = (m_2, w_3), (m_3, w_2), (m_1, w_1)$$

Gale and Shapley claimed that their algorithm implemented in complexity of

$$n^2 - 2n + 2$$

However, later Knuth [68] confirmed that the algorithm involved at time of $O(n^2)$ which is indicated in the next theorem.

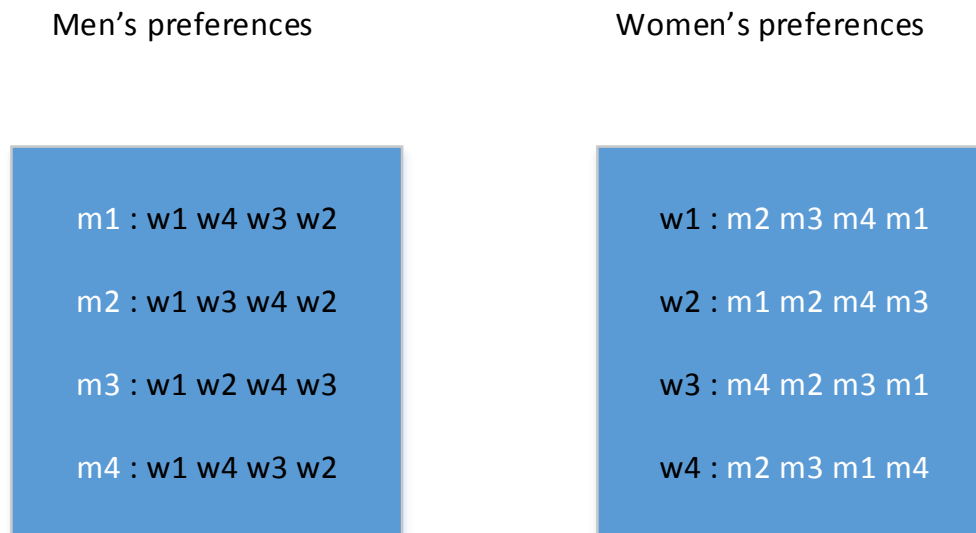


Fig. 2.2 Instance C2 of SMP algorithm.

Theorem 1. *Any algorithm to find a stable matching or to check if a given matching is stable or to determine whether a given pair is stable requires $O(n^2)$ time in the worst case, even when both the preference lists and ranking arrays are given as input.*

Theorem 2. *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.*

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 [43].

Theorem 3. *In the man-optimal stable matching, each woman has the worst partner that she can have in any stable matching.*

The following example in Figure 2.3 gives different output for both man-optimal and woman optimal for the same instance which formed out of 4 elements.

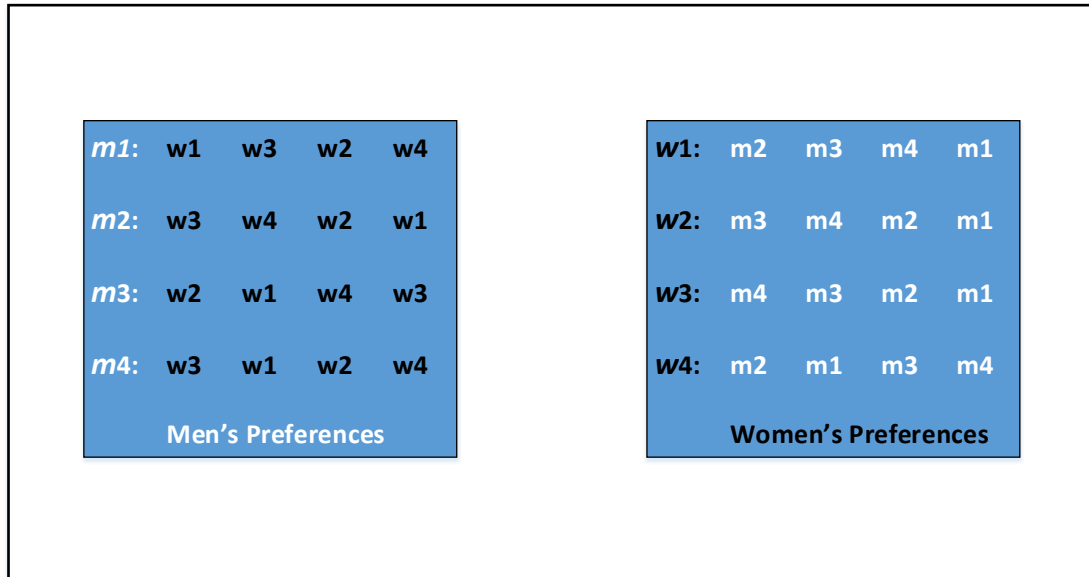


Fig. 2.3 A Stable Marriage Instance of Size 4.

The stable matching generated by both man-oriented and women-oriented versions is:

$$\text{Man-oriented} = M_0 = (1,4), (2,3), (3,2), (4,1)$$

$$\text{Woman-oriented} = M_z = (1,4), (2,1), (3,2), (4,4)$$

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.

2.2.2.2 Optimal Stable Marriage Problems

In this section we show a defined variant of stable matching that introduced by O'Malley in his thesis [93]. This variation is designed to find an optimal stable matching. He defined the rank of an agent p on an agent q 's list, denoted by $rank(p, q)$, to be the position of q on p 's list. Let I be an instance of sm, where M is the set of men and W is the set of woman in I . Let M be a stable matching in I , and let p be some agent in I . We define the cost of p with respect to M , denoted by $costM(p)$, to be $rank(p, M(p))$.

First he defined the regret of a matching M as follows:

$$r(M) = \max_{p \in M \cup W} costM(p)$$

He claimed that M has minimum regret if $r(M)$ is minimised over all stable matchings in I . Gusfield [41] described an $O(n^2)$ algorithm which finds a minimum regret stable matching given an instance of SM .

Also, he defined the cost of a matching M as follows:

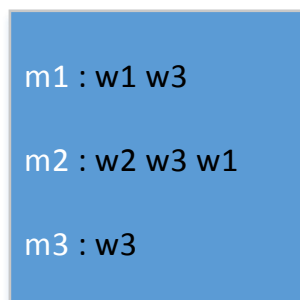
$$c(M) = \sum_{p \in M \cup W} costM(p)$$

Knuth [68], with Irving et al. [53] were the first who introduce the problem of egalitarian stable matching, which is the minimised stable matchings in in C , they introduced an algorithm which required $O(n^4)$ to solve this problem. However, a sophisticated algorithm has been introduced later by Feder [32] to find the egalitarian stable matching in $O(n^3)$.

2.2.2.3 Stable Marriage with Incomplete Lists (SMI)

From the title of this section it appears that the preference lists of candidates varies according to their desire. This occurs when a candidate find that a member of the opposite gender is unacceptable. Thus, the candidate disappears from the list of that member. This is clearly depicted in Figure 2.4. If one of the candidates let say a man m_j appears on a woman w_i 's preference list, this means that both m_j and w_i are acceptable to each other.

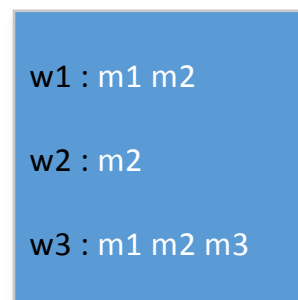
Men's preferences



A blue rectangular box containing three lines of text representing men's preference lists. The first line is 'm1 : w1 w3', the second is 'm2 : w2 w3 w1', and the third is 'm3 : w3'.

```
m1 : w1 w3
m2 : w2 w3 w1
m3 : w3
```

Women's preferences



A blue rectangular box containing three lines of text representing women's preference lists. The first line is 'w1 : m1 m2', the second is 'w2 : m2', and the third is 'w3 : m1 m2 m3'.

```
w1 : m1 m2
w2 : m2
w3 : m1 m2 m3
```

Fig. 2.4 Case C of SMI.

It can be noticed that the preference lists in SMI consistently interact according to any changes occurred from both genders. Figure 2.4 shows that man m_3 disappears from the preference list of woman w_1 which means, it is not accepted by woman w_1 . This implies that the woman w_1 disappears from the list of preference of man m_3 . The stability definition of SMI is amended from the form of SM, since it contains unacceptable candidates. Therefore, there are slight alerts to considering a blocking pair of M [43], if:

- **m and w are not matched in M , but m and w find each other acceptable.**
- **m is either unmatched in M , or prefers w to his partner in M .**
- **w is either unmatched in M , or prefers m to her partner in M .**

Gale and Sotomayor [36] stated that in a case of SMI, the same set of women and men are matched in all stable matchings, as noted below.

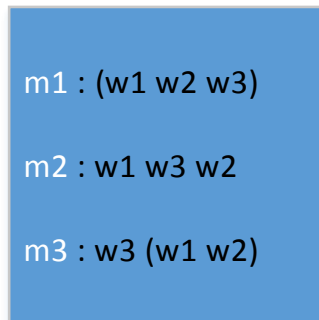
Theorem 4. *In an instance of SMI, the same set of men and women are matched in all stable matchings.*

2.2.2.4 Stable Marriage with Ties (SMT)

In this version, ties are used to determine indifferent between two or more candidates, as shown in Figure 2.5. It appears that m_3 is indifferent between w_1 and w_2 and prefers w_3 to both of them whereas m_1 does not prefer any candidate to others, as it is indifferent between all of the women candidates. This leads to various definitions of the stability such as super-stability, strong stability and weak stability. From their names, these suggest to

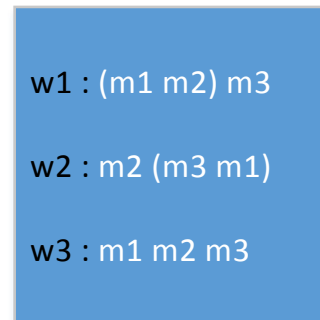
check the stability of matched pairs among a matching. Therefore, a blocking pair is defined according to the aforementioned variations as follows:

Men's preferences



m1 : (w1 w2 w3)
 m2 : w1 w3 w2
 m3 : w3 (w1 w2)

Women's preferences



w1 : (m1 m2) m3
 w2 : m2 (m3 m1)
 w3 : m1 m2 m3

Fig. 2.5 Case C of SMT.

- **Super stability:** each of m and w either indifferent between them or strictly prefers the other to their partner in matching M.
- **Strong stability:** Whichever, m either strictly prefers w to his partner in M or is indifferent between them, and w strictly prefers m to her partner in M, or w either strictly prefers m to her partner in M or is indifferent between them, and m strictly prefers w to his partner in M.
- **Weak stability:** both m and w strictly prefer each other to their partners in M.

According to the aforementioned definitions, a matching M is said to be super, strongly and weakly stable when blocking pairs disappear. An algorithm to find a super stable

matching is defined by Irving [53], which runs in time $O(n^2)$. Also, he provides another algorithm to find a strongly stable matching, which requires $O(n^4)$. Both algorithms rely on the strategy of “deferred acceptance”.

2.2.2.5 Stable Marriage with Ties and Incomplete List

As previously mentioned in SMI, however, this algorithm has ties which considered in the process of matching. The stability can be achieved by combining the ties with incompleteness. For instance, when two applicants tie (in terms of their qualifications) to accomplish a certain job, the system should find a way to sort it out by using such algorithm.

With regards to our research some of aforementioned algorithms with slight amendment may help to find a way of admitting an optimal matching. Also, there are some algorithms derived from the main algorithm supporting different approaches such as non-bipartite (more than two sets).

2.2.3 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 Matching Program is a real existing example that takes a place in the United States, which helps to match around 30,000 residents to hospitals yearly.

It is clear that every hospital can accommodate more than one resident and many residents can be assigned to one hospital, which implies the cardinality of one-to-many. Hospital resident (HR) problem is introduced to manage such issue. There are two sets involved in this algorithm, a set H of x hospitals and a set R of y residents. Respectively, every hospital h has a capacity c that indicates the number of available positions. Each hospital ranks its acceptable residents in R in strict preference order and each resident ranks his/her acceptable hospitals in H in strict preference order. In matching M , the hospital h should appear no more than c pairs, and every resident r should appear in only one pair. It can be said that a pair $(r, h) \notin M$ is a blocking pair in matching M if:

- (i) r prefers h to the hospital he/she is assigned to or r is unmatched and finds h acceptable and,
- (ii) h prefers r to one of the residents assigned to it or h is not filled to capacity c and finds r acceptable.

Algorithm 3 Hospital-oriented algorithm [55]

```

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;
```

Gale and Shapley in [35] are first introduced the HR problem along with the simple “deferred acceptance” strategy which acts as an engine of the algorithm. They proved the existing of stable matching in every instance of HR, which found in time of $O(nm)$. In [22] Cheng et al. discussed how to find the possible stable matchings. Also, they checked an HR instance and its stable matchings and introduced meta-rotations concept.

Algorithm 4 Resident-oriented algorithm [55]

```
1: assign all residents to be free
2: assign all hospital to be totally unsubsidised;
3: while (some resident  $r$  is free) and ( $r$  has a nonempty list) do
4: begin
5:  $h :=$  first hospital on  $r$ 's list;  $r$  "proposes" to  $h$ 
6: if  $h$  is fully subscribed then
7: begin
8:  $r' :=$  worst resident provisionally assigned to  $h$ ;
9: assigned  $r'$  to be free
10: end;
11: provisionally assign  $r$  to  $h$ ;
12: if  $h$  is fully subscribed then
13: begin
14:  $s :=$  worst resident provisionally assigned to  $h$ ;
15: for each successor  $s'$  of  $s$  on  $h$ 's list do
16: remove  $s'$  and  $h$  from each other's lists
17: end;
18: end;
```

2.2.4 Stable Roommates Problem

Unlike stable marriage problem, the stable roommate algorithm is a non-bipartite extension of the main stable marriage problem [43]. The algorithm is applied on the candidates of the same set of even cardinality, so the idea of SRP is based on building the preference lists for each member from same set in strict order. A matching is a set of n unordered pairs of

members whereas $2n$ reflects the number of participants. The matching is stable if there is no pair (x, y) , each of whom prefers the other to their assigned roommate in M . Such a pair is said to block M .

Algorithm 5 Phase 1 of the stable roommates algorithm [34]

```

1: assign each person to be free
2: while some free person  $x$  has a nonempty listdo
3: begin
4:  $y :=$  first person on  $x$ 's list;  $x$  proposes to  $y$ 
5: if some person  $z$  is semiengaged to  $y$ ; then
6: assign  $z$  to be free;  $y$  rejects  $z$ 
7: assign  $x$  to be semiengaged to  $y$ 
8: for each successor  $x'$  of  $x$  on  $y$ 's list do
9: delete the pair  $x', y$  from the preference table
10: end;
```

SR has been firstly studied by Gale and Shaply [35], and gave an example that does not confess a stable matching. However, later on the problem have been solved by Irvin [54] when the problem stood by Knuth[68] when he defined an algorithm that required time of $O(n^2)$. Afterward the structural aspects of SR have been explored by Gusfield [42], also a comperhansive discussion of SR is given by Gusfield and Irving in [43]. Also, Tan [104] announced the concept of a stable partition in an instance I of SR in which he provides a ‘succinct certificate’ in the case that I does not admit a stable matching. Subsequently, Feder et al. [33] showed an $O(n \log^3 n)$ parallel algorithm to find a stable matching if exists, for a given instance of SR.

Algorithm 6 Phase 2 of the stable roommates algorithm [34]

```
1:  $T :=$  phase 1 table;
2: while (some list in  $T$  has more than one entry) and (no list in  $T$  is empty)do
3: begin
4: find a rotation  $\rho$  exposed in  $T$ ;
5:  $T := T/\rho$  eliminate  $\rho$  then
6: end;
7: if some list in  $T$  is empty then
8: report instance unsolvable
9: else
10: output  $T$ , which is a stable matching
```

2.3 Clone Detection

2.3.1 Overview

Clone detection is a crucial field that has been intensively conducted by researchers and practitioners for the last two decades to enhance software systems and therefore, improves the maintainability for the future lifespan of the software system. Although clone detection is a wide spread research problem over many years; it is considered as a fuzzy terminology, since the researchers have differently defined it according to various 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.

2.3.2 Clone Relation Terms

Clones are usually detected as one of two forms or terms; either clone pair or clone classes. These two terms focus on the similarity relation of two or more pieces of software clones. [62] Describe an equivalence relation to the similarity relation, such as symmetric relation

and transitive relation. It can be said that there is a clone-relation between two fragments of code if they have the same sequences (characters, strings, tokens etc.) From Figure 2.6 below we can express the meaning of clone pair and clone classes based on the clone relation:

<u>Fragment 1:</u>		<u>Fragment 2:</u>		<u>Fragment 3:</u>
...	
for (int i=1; i<n; i++) { sum = sum + i; }	a	for (int i=1; i<n; i++) { sum = sum + i; }	a	
if (sum < 0) { sum = n - sum; }	b	if (sum < 0) { sum = n - sum; }	b	if (result < 0) { result = m - result; }
...		while (sum < n) { sum = n / sum ; }	c	while (result < m) { result = m / result }
	

Fig. 2.6 Clone pair and Clone class.[97][5]

- 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 2.6 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 [13].

- **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 1, 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 a collection of clone pairs that sharing the same code fragment [96, 98]
- **Clone Communities:** as termed in [88], it is another name of the Clone classes that reflects the aggregation of related code fragments which form a clone pairs.
- **Clone Class Family:** researchers in [96] revealed the term of clone class family to group aggregate all clone classes of same domain.
- **Super Clone:** as have been outlined by [58] 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 [13]. Therefore, it can be classified as both a class clone (in early stage of clustering similar fragments of code) and super clone.

2.3.3 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 [73, 78](Lakhotia, Li et al. 2003, Kontogiannis 1997). However, a fragments of code that have identical or similar code fragments in the source code are considered to be code clones. Regardless of the changes that have been applied on a certain code clone, if still within 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. [15](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 [19, 98](Burd, Bailey 2002, Roy, Cordy 2007) in which fragment of code is called clone when there is more existences of that fragment in the source code with or without “minor” modifications. However, a number of researchers [63, 73, 82](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 [16, 73, 98](Roy, Cordy 2007, Kontogiannis 1997, Bellon, Koschke et al.

2007)they attempt to automatically unify the result sets of multiple detectors, to try to solve the differential detector-based output.

2.3.4 Detection Techniques and Tools

The following are some features (dimensions) which form corner stones and clarify the several facets of the clone detection techniques:

- **Source Representation:** it is the final form or representation of the code fragments being compared in the comparison phase to meet the algorithm requirements. This can be achieved using the different types of transformations/normalizations or filtering.
- **Source Transformation/Normalization:** some clone detection approaches apply a kind of transformation / normalization or filtering on the original source code to get a suitable code format in order to apply a comparison algorithm. However, some other approaches only remove the comments and the whitespaces.
- **Comparison Algorithm:** one of the major concerns or issues that can affect the performance of the clone detection process is the choice of the appropriate detection or matching algorithm. Several approaches apply different data mining/information retrieval algorithms.
- **Clone Granularity:** there are two types of granularity clones fixed and free granularity. A returned fixed granularity clone is the pre-defined block size of the considered code fragments such as (e.g., function, begin-end brackets etc.). However, if there is no

certain considered limit or size of the code portion block then they are called free granularity clones.

- **Clone Similarity:** This feature represents the type of code clones that can be found by some detection techniques such as exact match clone, parameterized match or near-miss clones.

2.3.4.1 Text-based Technique

This technique is purely based on the text or string methods, so in this approach the raw source code is considered as sequence of lines and strings. Code segments are matched with each other to detect the same sequences of text or strings which are not related to structural elements of the language. The detected sequences are returned as clone pair by the detection technique. Some text-based approaches perform a slight transformation/normalization on the code fragments before setting off the comparison process, whereas normally the raw source code is directly used in the matched process. The following are some commonly used filtering/transformation/normalization techniques in some approaches:

- **Normalization:** basic normalization can be applied on the raw source code (Table 2.7).
- **Whitespaces:** considers and removes all whitespaces including tabs, new line(s) and other blanks spaces.
- **Comments:** eliminates all comments used in the source code.

Text-based approaches differ from one another, as they are based on different techniques such as fingerprints and dot plot. Ducasse et al. [28] present one of the most recent text-based

Operation	Language element	Example	Replacement
1	Literal string	"Abort"	"..."
2	Literal character	'y'	'.'
3	Literal integer	42	1
4	Literal decimal	0.314159	1.0
5	Identifier	counter	p
6	Basic numerical type	int, short, long, double	num
7	Function name	main	foo()

Fig. 2.7 Normalization operations on source code elements.[28, 98]

approaches that is based on dot plot. The scatter plot is a two dimensional chart formed by two axes of source code. In this approach the comparison units are lines of program. The dot appears when x and y are equal based on the calculated hash value of both lines. The diagonals in dot plots can identify the clones, as dot plot can further be used to display the information of code clones. String-based dynamic pattern matching is applied by Ducasse et al. on dot plot to match the lines of code. Another similar approach SDD is identified by Lee and Jeong[80], applies an n-neighbor technique to detect type-3 clones.

As the approach of Ducasse et al is not sure about recognizing the type-3 clones Wettel and Marinescu [110] provide an extension to this approach to find nearmiss clones using dot plots. They use the algorithm that relates the neighboring lines to detect some forms Type-3 clones. The approach of Marcus and Maletic [87] applies latent semantic indexing (LSI) to source code to identify and retrieve the two similar code portions, based on the similar used comments and identifiers of the retrieved code fragments. This approach limits to identify abstract data types (ADTs) which are high level concept clones.

However, Johnson is a key person who applies the technique of fingerprints in his approach on substrings of the source code [59, 60]. Initially, he uses the hash technique

to hash certain portion of code of fixed number of lines (the window). Then they classify the sequences of lines which have the same hash value as clones, using the known sliding window techniques as well as the incremental hash function. The sliding window technique helps to recognise code clones of variant lengths, as it is frequently applied. Also, Manber [86] uses in his approach the technique of fingerprints to detect similar source files, based on the subsequences of the main keywords.

2.3.4.2 Metrics-based Technique

In Metrics-based approaches, several software metrics are gathered from clone fragments to derive its measurement in order to be compared instead of comparing the actual portions directly. These metrics values are related to various scopes such as a package, a class or a method and then these values are compared to detect code clones over these blocks. The source files are normally parsed to Abstract Syntax Trees representation as a pre-process to calculate the software metrics.

There are several software metrics tools which can be used for code measurements. [83] (Lincke et al. 2008) have made selection set of software metrics tools according to analyzable languages, metrics calculated, and availability/license type. They found that the majority of metrics tools available can derive metrics for many programming languages such as Java programs, UML and C/C++. They state that about half of the tools are rather simple “code counting tools” which calculates Lines of Code (LOC) metric. However, they consider the other half as more sophisticated software metrics such as CBO (Coupling Between Object

classes). The following are some of the final selected software metrics tools by (Lincke et al. 2008) [83]:

- **OOMeter** this software metrics tool accepts Java/C# source code and UML models in XMI and calculates various metrics, developed by Alghamdi et al [2].
- **Eclipse Metrics Plug-in 1.3.6** this is an open source metrics calculation and dependency analyzer plugin for the Eclipse IDE. It calculates several metrics and catches cycles in package and type dependencies, developed by Frank Sauer.
- **CCCC** this is an open source command-line tool. It analyses C++ and Java code, proposed by Chidamber & Kemerer and Henry & Kafura.
- **Understand for Java** this is a metrics tool for Java source code.
- **Dependency Finder** this is an open source tool for analyzing compiled Java code. It extracts dependency graphs and mines them for useful information.
- **Semmler** is an Eclipse plug-in which allows searching for bugs and measure code metrics by providing and SQL querying languages for object oriented code.
- **Analyst4j** this is an Eclipse IDE plug-in which allows several features such as search, metrics and report generation for Java programs.
- **Eclipse Metrics Plug-in 3.4** this is an open Source tool which calculates various metrics during, developed by Lance Walton.

Linke et al also considered some software metrics derived by aforementioned tools and they base their selected metrics on the class unit, as they argue that this unit is the natural block of object oriented software systems and most metrics have been calculated on class level. The following are some considered software metrics in their study:

- **LOC** (Lines Of Code) calculates the lines of code of a specified unit [52].
- **NOM** (Number Of Methods) calculates the methods in a class [102].
- **LCOM-CK** (Lack of Cohesion of Methods) describes the lack of cohesion between the methods of a class [23]. It is proposed by Chidamber & Kemerer
- **CBO** (Coupling Between Object classes) gives the number of classes to which a class is coupled [23].
- **NOC** (Number Of Children) is the number of subclasses to a certain class in its block [23].
- **RFC** (Response For a Class) reflects the number of methods which can be executed in response to an object of the class [23].
- **DIT** (Depth of Inheritance Tree) represents the maximum inheritance path from the class to the main root class [23].
- **WMC** (Weighted Methods per Class) it is the total of weights for the methods of a class [23]. However, using Cyclomatic Complexity software metric method weight can be achieved [108].

- **LCOM-HS** (Lack of Cohesion of Methods, proposed by Henderson-Sellers) describes the lack of cohesion between the methods of a class [102].

The following table shows the software metrics calculated by metrics tools where ‘x’ indicates that a certain metric can be calculated by a certain metric tool.

Tools	Metrics								
	CBO	DIT	LCOM-CK	LCOM-HS	NOC	NOM	RFC	TCC	WMC
Name									
Analyst4j	X	X	X		X	X	X		X
CCCC	X	X			X	X			
Chidamber & Kemmerers Java Metrics	X	X	X		X	X	X		
Dependency Finder		X			X	X			
Eclipse Metrics Plugin 1.3.6		X		X	X	X			X
Eclipse Metrics 3.4			X	X					X
OOMeter	X	X	X		X			X	
Semmler		X	X	X	X	X	X		
Understand for Java	X	X	X		X	X			
VizzAnalyzer	X	X	X		X	X	X	X	X

Fig. 2.8 Tools and calculated metrics.[83]

Davey et al. [26] apply neural networks algorithm on code fragments (begin end block) and considered a certain features of specified code blocks. Their approach identifies the exact, parameterized, and near-miss clones.

Balazinska et al. [11] have proposed their tool SMC (similar methods classifier) in which dynamic matching and metrics craterisation are combined. A similar approach was defined by Kontogiannis et al. [74] based on two different ways to find code clones. The first one uses direct comparison of the metrics values of the specified clone granularity. The second

compares the specified blocks (code fragments), a statement-by-statement basis using a dynamic programming (DP) technique, as the small distance is considered as clone that is caused by cut-and-paste habits. Both Patenaude et al. [94] and Mayrand et al. [88] take into account several metrics which consider some features such as names and control flow of functions to match functions with same or close metrics values as code clones.

However, Calefato et al. [20] have showed that Metrics-based approaches have been applied in web documents to detect redundant web pages and clones.

Table 2.1 shows a comparison of different approaches of metrics-based techniques with respect to several properties.

Table 2.1 Metrics-based Detection Techniques.

Properties	Lanubile	Mayrand	Kontogiannis
Code Representation	Several metrics (LOC,ELOC,CLOC)	Several metrics	Feature vectors
Comparison Granularity	visual inspection of each function	Metrics of individual functions	Metrics of each begin-end block
Complexity	Not specified	Polynomial $O(n^3)$, n=statements contains, clones	$O(n^2)$,n=input size
Comparison Technique	metrics values of visual inspection	21 function metrics with 4 points of comparison	Numerical comparison of metrics values
Clone Similarity	Identical, nearly Identical, similar and distinct	Exact and near-miss	Partial and near-miss
Clone Granularity	Free, function	Free, function	Free, begin-end blocks)
Output Type	Functions with metrics values	Clone Class and Clone Pair	Clone Pair

2.3.4.3 Token-based Technique

Token approaches (lexical approaches) transform/parsed/lexed the source code into a sequence of tokens using compiler-style lexical analysis. These tokens are scanned to detect duplicated subsequences of tokens, as a result the matched tokens from the original code fragments are retrieved as clones. This technique is much better than the textual approach in term of detecting the minor code changes such as formatting and spacing.

Baker's tool Dup [8, 9] is one of the best tools that represent this approach. She used a lexical analyser to chop the line of the program to se-quence of tokens. However, there are two types of tokens that appear in this stage respectively; parameter tokens (identifiers and literals) and non-parameter tokens. The parameter tokens are encoded based on their occurrence in the line (position index) which helps in detecting type-2 clones whereas in the non-parameter tokens a hashing function is used. Suffix tree is used to present the prefixes of sequence of symbols. Common prefix means that tree suffixes share the same set of edges, which can be considered a clone.

CCFinder [62] of Kamiya et al.is another recent and efficient token-based clone detection in this approach. Each line of the program is chopped to tokens using a lexer. The whole tokens of a certain source file are then chained as a single sequence. Then, the token se-quence is transformed (based on the specified transformation rules such as add, change or delete tokens). Special tokens are used to be replaced by the identifiers (with considering types and names) across the source file, making code portions with different variable names clone pairs. The similar sub-sequences are searched among the transformed token sequence using suffix-tree based sub-string matching algorithm to be returned as clone pairs/clone

classes. Finally, a suitable mapping is applied between the already obtained information of clone pair/ clone class the token-sequences and the clone pair/ clone class information of the original source code. Several tools are based on the CCFinder, such as RTF [14], which enhances the process of detecting clones by allowing the user to tailor tokenization; by using a more memory-efficient suffix-array in place of suffix trees. Also, Gemini [106] uses scatter plots to display near-miss clones. Another pioneered clone detection technique is CP-Miner [81, 82], in which a frequent subsequence data mining technique [1] is used to find similar sequences of the original tokenised statements.

Some of the aforementioned techniques are used to detect plagiarism. SIM [38] is one of the plagiarism detection tools, which uses the dynamic programming string alignment technique to compare token sequences. Also, JPlag [95] and Winnowing [100] are examples of plagiarism detection tools which is based on token techniques.

2.3.4.4 Tree-based Technique

Tree-matching approaches detect code clones by matching the similar sub-trees obtained by parsing the source code (an abstract syntax tree). Several techniques of tree matching are used to search the related source code of the corresponding subtrees which are returned as clone pairs. Compared to token-based approach, the AST is more sophisticated in order of detecting code clones as it is based on the structure of the program rather than variable names, literal values. CloneDr [15] which is lunched by Baxter et l., is one of the best tools uses AST. The subtrees which are obtained by parsing the trees of the program are hashed into buckets which compares the contained subtrees to each other using a tolerant tree matching.

Table 2.2 Detection Methods of Token-based.

Properties	Kamiya et al.	Baker	Li et al.
Code Representation	Sequence of normalized, transformed & parameterized tokens	Parameterized token string	Collection of short sequences of numbers
Comparison Granularity	Token	Sequence of tokens	Tokens sequence
Complexity	$O(n)$, n =lines of code	$O(m+n)$, m =number of detections, n =lines of code	$O(n^2)$, n =Lines of code
Comparison Technique	Tokens of Suffix-tree	Suffix-tree based Token matching	Frequent subsequence mining technique
Clone Similarity	Exact or near miss possibly with gaps	Exact and parameterized matches	Exact and near miss with gaps
Clone Granularity	Free, threshold-based of tokens (30 tokens)	Free and threshold-based (minimum 15 lines)	Free and threshold-based (of functions)
Output Type	Clone pair and Clone Class	Clone pair and Clone Class	Clone pair

Evans and Fraser [30] have proposed an approach in which structural abstraction of a program is achieved to handle exact and near-miss clones with gaps. Yang [112] has proposed an approach which can effectively manage the syntactic differences between the compared subtrees using dynamic programming approach. Wahler et al. [107] have converted the AST to XML to find exact and parametrised clones at abstract level. Then they have applied a data mining technique to find code clones.

However, there are several recent approaches which facilitate the comparison process of subtree by applying alternative simple tree representations rather than considering the full subtree. Falke, et al. [31], who have serialized the AST subtrees as AST node sequences for

which a suffix tree is then constructed which allows to detect the syntactic clones faster (at the speed of token-based techniques). Tairas and Gray [103] have proposed an approach based on suffix trees, using Microsoft's new Phoenix framework. Jiang et al. [56] in their tool Deckard have proposed a novel approach to detect similar trees, in which vectors are computed to approximate the structure of ASTs in a Euclidean space. Then locality sensitive hashing (LSH) is applied to aggregate the similar vectors using Euclidean distance metric to detect the related code clones.

Table 2.3 Detection Methods of Tree-based.

Properties	Evans	Wahler et al	Yang
Code Representation	AST in XML	AST in XML	Parse tree
Comparison Granularity	AST Node	one line	Token (Tree node)
Complexity	Not specified	$O(k.n^2)$, n=statements contains, clones, K=maximal size clones	$O(T1T2)$, T1 = nodes of first tree, T2=nodes of second tree
Comparison Technique	Graph theoretic Approach	Frequent Itemset	Tree matching
Clone Similarity	Exact and near miss with gap	Exact and parameterized matches	Exact and near miss
Clone Granularity	Free, threshold-based	Free, threshold-based (5 statements)	Free, program/ segment)
Output Type	HTML document with clone information	Not specified	Just displays with pretty-printing

2.3.4.5 PDG-based Technique

Program Dependency Graph (PDG)-based approaches [70, 76, 84] semantically considers the source code details due to the high abstraction representation of the source code. Expressions and statements are presented by the nodes of the graph whereas control and data dependencies are represented by the edges. PDG provides more precise information than the syntactic approaches. As the PDG holds crucial information of a program such as control flow and data flow, thus it can facilitate matching the corresponding similar subgraphs (code clones) easily in a semantic way using matching algorithm. The approaches of the PDG are reliable and robust for code management. However, they lack of scalability to huge systems.

Komondoor and Horwitz [70, 72] have proposed one of the most leading approaches in this technique known as PDG-DUP which uses program slicing to detect isomorphic PDG subgraphs [109]. They have also applied a sophisticated approach which aggregates the detected code clones with keeping the semantics of the source code [69, 71]. This has been used to support software refactoring by automating the procedure extraction. Furthermore, Gallagher and Lucas [37] have conducted a slicing based clone analysis experiment by arguing the raised question: Are decomposition slices clones? going through the all variables in the whole system by computing program slices. They have showed the pros and cons to the raised arguments.

Chen at al. [21] proposes an approach for code compaction which considers syntactic structure and data flow. The technique has several features in embedded systems. Liu at al. [84] announce their tool which helps in plagiarism detection purposes. Krinke [76] also

proposes an iterative PDG-base technique (k-length patch matching) to finding maximal similar subgraphs.

Table 2.4 Detection Methods of PDG-based.

Properties	Krinke	Komondoor et al.	Liu et al.
Code Representation	Fine Grained PDGs	set of PDGs of procedures	set of PDGs without control dependencies
Comparison Granularity	PGD subgraphs PDGs	PGD Node	PGD Node
Complexity	Non-polynomial	Not Available	NP-Complete
Comparison Technique	patch matching of k-length	Isomorphic PDG subgraph matching using backward slicing	Isomorphic subgraph matching
Clone Similarity	Exact and Semantic	non-contiguous, reordered, intertwined	Exact and Near-miss
Clone Granularity	Free, threshold-based, length limited similar path	Free, slicing-based	Fixed, programs
Output Type	Clone Class	Clone pair and Clone Class	Plagiarized pair of procedures

2.4 Service Selection

2.4.1 Overview

Service selection is the process of choosing the appropriate service which fulfils the client's requirement. Service selection is a key step to affect the whole process of service composition. Different approaches are applied to match service providers and consumers. Typically the selection relies on the search function of Universal Description, Discovery and Integration (UDDI) which often provides a relatively poor search facility by allowing only a keyword

based search of web services. Consequently, returning unrelated services since UDDI registry encompasses both checked and unchecked services [57]. Huang et al. in [51] present an efficient service selection scheme that considers non-functional characteristics while choosing web services by service requesters. Their model considers different data types, and uses multiple criteria decision making (MCDM) with weighted sum model (WSM) to help service requesters evaluate services numerically. They proof using experimental results that their service selection scheme performs much better than the enumerative method, and outperforms other related methods in terms of efficiency and effectiveness. The Figure 2.9 below illustrates the whole process through the three major stages of Huang's model.

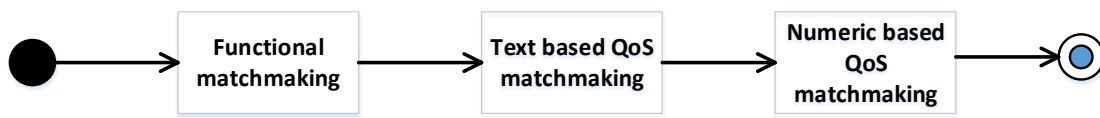


Fig. 2.9 Service Selection process [51].

2.4.2 QoS-based Service Selection

Various definitions of Quality of Service (QoS) has been provided previously; Schmidt in [101] defines the QoS as the degree to which a system, component or process meets customer or user needs or expectations". Alrifai and Risse [6] propose a solution that combines global optimisation with local selection techniques to prevent local selection strategy fails in handling global QoS requirements. They use distributed local selection to find the best web services that satisfy these local constraints. Baldoni et al. [12] show that the semantic matchmaking techniques used in the services is lacking preservation and they provide an

approach to overcome these limitations. Yu et al. [113] present a broker-based architecture to facilitate the selection of QoS-based services. Recent approaches of service selection are deeply involved in semantic manner using ontology which has drawn the attention of many researchers [66][92][40][61][65].

Discovering the recommended web services needed by users among the vast resources is a significant challenge. Zhong et al. [10] discuss functional attributes and non-functional attributes of the web services resources from the resources balanced perception of FQoS (Functional Quality of Service) and QoS. They propose a web service description, discovery and selection mechanism towards balanced perceptions of FQoS and QoS. In this mechanism, FQoS is more used to describe and discover services, and QoS is more used to select services. Due to the perspective differences, they point out several problems on the research on service functions and quality of service. There are big differences among QoS indicators. Due to the similarity between the functional and non-functional attributes, there is an ambiguity of specifying service functions out of QoS [105]. Figure 2.10 below illustrates a brief depiction about FQoS.

2.4.3 Semantic Matching

Semantic terminology reflects the cleverness of the process that find the similarity of corresponding concepts that somehow are related to main candidates. In this section we show a brief description of ontology alignment. It is a significant process that contributes to the foundation of semantic Web, which can help to solve the heterogeneousness problem. There are several alignment methods which are classified based on their approaches and strategies.

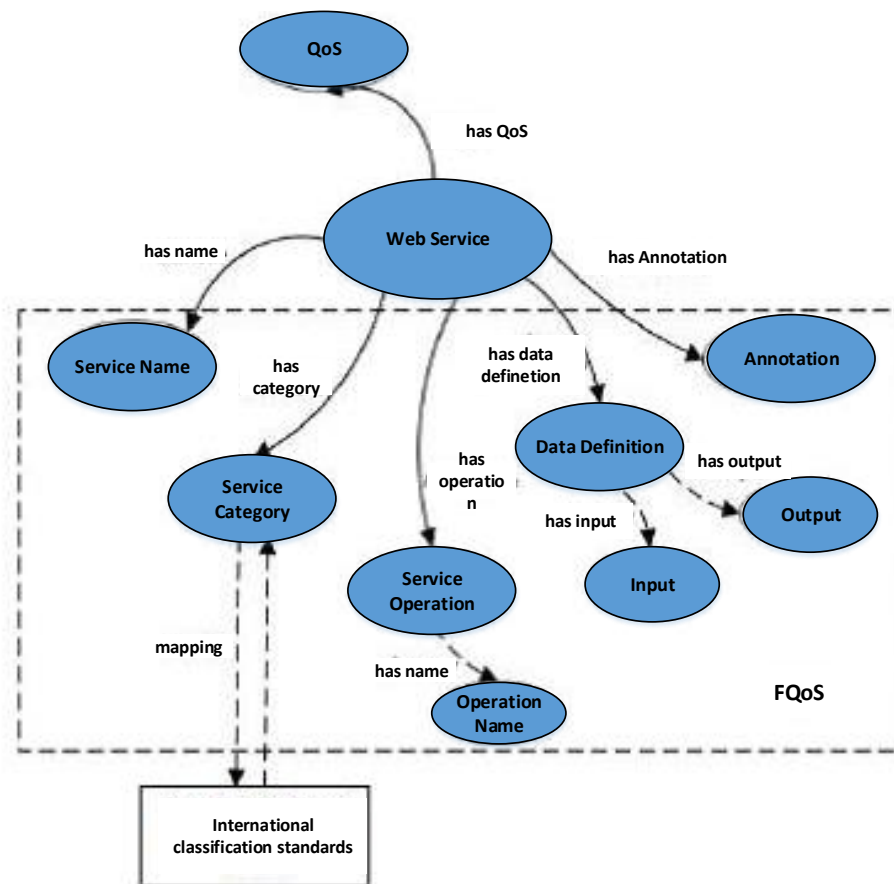


Fig. 2.10 Description Framework Of FQoS [10].

Some of these methods are based on lexical and linguistic treatments [75]. Whereas other methods act as hybrids and comprehend the lexical treatments, which rely on the structural of the ontologies to be aligned [111]. These methods helped to complete the classic techniques of matching, which reveals the effectiveness or the wealth of the ontologies representative language. Ontologies represent the concepts used in a domain; therefore, each ontology should reflect a whole related domain. However due to the parallel use, different ontologies which related to a certain domain may developed independently. Thus, Ontology matching [29] is used for creating mappings between ontologies.

Many matching processes (matchers) of ontology alignment systems use several approaches to measure the similarity between concepts that related to two different ontologies. Based on the context of similarity measurements, these matchers relate to different classifications such as lexical, structural or extensional matchers. Cross in [24] goes further and tries to ensure the quality of mappings produced by the Ontology Alignments systems, by investigating the use of semantic similarity measures. This help to evaluate ontology alignment especially with respect to precision [24]. Semantic alignment quality is based only on the produced mappings and has no knowledge of missed mappings. Therefore, it is more related to the precision than recall.

2.4.4 Service Availability

The main purpose of designing the Clouds is to provide services to users and providers needs to be made up for sharing their capability and resources [4]. Armbrust et al. [7] discuss many obstacles of growth cloud computing paired with the suggested solutions (opportunities) to overcome these issues as depicted in Table 2.5. Also, many organisations worry about the availability of the computing services, which therefore affect the cloud computing workflow. Some software as a service (SaaS) products set a high standard in this regard to prevent the issue of service unavailability such as Google Search [7]. Table 2.5 shows the services outages and the causes, for Amazon Simple Storage Service (S3), AppEngine and Gmail in 2008.

As services providers need to meet the requirement of the consumers, providing a mechanism of matching the services description to the required services is very significant.

Table 2.5 Obstacles to and opportunities for growth of cloud computing.[10]

Service and Outage	Duration	Date
S3 outage: authentication service overload leading to unavailability	2 hours	2/15/08
S3 outage: Single bit error leading to gossip protocol blowup	6-8 hours	7/20/08
AppEngine partial outage: programming error	5 hours	6/17/08
Gmail: site unavailable due to outage in contacts systems	1.5 hours	8/11/08

Meanwhile, massive services of the Internet need to assign the services to the requesters efficiently and automatically. Therefore, providing service matching approach can tackle above issues by redirecting the related request to the most similar available services when the current service is busy. In this paper SMP algorithm will be investigated to improve the process of service selection.

The main SMP algorithm performs a one-to-one cardinality over the matched candidates. Furthermore, the relationship of one-to-many is presented in one of the extended SMP algorithms which called Hospital resident (HR), basically many residents assigned to one hospital [22]. Whereas in this approach a many-to-many relationship is needed. Hence, the original SMP algorithm is extended to cope with the new requirements of this proposed approach.

Many-to-many relationship is needed in both clone detection and service selection processes. This helps the services for instance to interact according to their views based on the profile and the history of the costumers (incoming requests), which secure the services from unwanted malicious costumers. Based on this background chapter 5 shows the use of

the extended Gale-Shapley algorithm in our application to enhance the process of service selection.

2.5 Search-Based Optimisation

2.5.1 Overview

Search concept is a corner stone of many processes such as a sophisticated search approaches and matching processes. The simplest search process is random search that randomly generates solutions. However, such a search process works without any aspect of intelligence, therefore the sophisticated search processes are guided by fitness function that supported by some optimization techniques for instance simulated annealing, hill climbing simulated annealing and genetic algorithms [45].

Search Based Software Engineering (SBSE) is an approach in which search optimization techniques are applied to manage some obstacles raised in Software Engineering processes and products using generic, flexible, robust and scalable computational search. SBSE is widely applicable and successful approach that provides a mechanism for managed automation of software engineering activities [46].

The importance of SBSE appears clearly in providing insights and decision support. Also, it is crucial to transfer the problems of software engineering to be machine-based search rather than human-based search by using the evolutionary computation paradigms and several methods of the metaheuristic search. This is to benefit from both machines' reliability and

humans' creativity, rather than relying on humans, which avoid error prone that makes the features of the engineering process very expensive [49].

Matching is a process in which we must apply a search process that tries to find the candidate based on certain standards; however we consider using some sophisticated search processes in SMP algorithm to apply a dual search process in which both seekers looking to satisfy their own criteria or features.

Search-based optimization shares valuable ideas, which might intersected in future works of other optimization techniques. As long as the SMP algorithm is match-based, we tried to borrow some search based techniques to be more suitable for some match-based processes. This might reveal a new term for future research direction known as match-based software engineering.

2.5.2 SBSE Ingredients

There are two basic factors for performing SBSE [44, 48, 49]:

- **The presentation of the problem.**
- **The fitness function.**

Figure 2.11 shows these two ingredients which are the basis of Search Based Software Engineering to implement search based optimisation algorithms and get results. Software engineers use a proper representation for their problem. However, the fitness function guides the search process for optimal or near optimal elements in a search space of possible solutions using software metrics [47, 49].

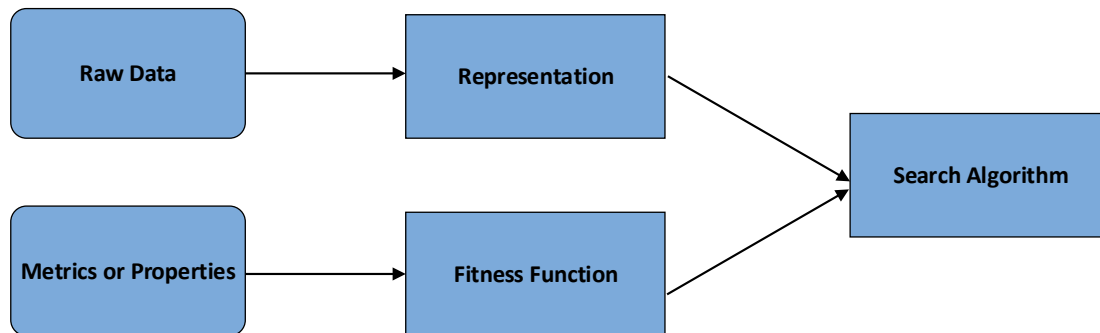


Fig. 2.11 Overall Architecture of SBSE Approach [49].

2.5.3 Common Search Algorithms

SBSE benefits from well-known search optimization techniques such as hill climbing, simulated annealing and genetic algorithms. These algorithms add a sense of cleverness compared to the basic random search which is considered as unguided search that usually fail to find the optimal solutions.

Both Simulated Annealing and Hill climbing are considered as local searches due to their way of work, as they refer to one candidate solution at a time, choosing ‘moves’ based on the neighbourhood of that candidate solution. However, genetic algorithms are said to be global searches.

2.5.3.1 Hill Climbing

Hill climbing is the simplest search algorithm that uses fitness information in the form of fitness function[49]. A point of the search space is randomly selected by the algorithm of Hill Climbing. After that the algorithm checks the raised solutions in the ‘neighbourhood’ of the original. If the neighbouring candidate solutions are similar and found of improved

fitness, the search ‘moves’ to that new solution and explores the neighbourhood of it for better solutions, and so on until no improved neighbours can be found for the current candidate solution. This solution is locally optimal.

2.5.3.2 Simulated Annealing

Originally, the name of Simulated Annealing comes from the analogy of the technique with the chemical process of annealing. Later, Kirkpatrick et al. [67] proposed the algorithm as the basis of the search mechanism.

Simulated Annealing relies on a certain variable namely temperature, and without restarting the search process, it attempts to escape local optima. At the start the temperature is high and the free movement through the search space is allowed. As the search progresses, the temperature reduces, consequently, moving to poorer solutions becomes unlikely. Ultimately, freezing point is reached, leading the algorithm to acts as if Hill Climbing. A lower solution is calculated by consider the acceptance probability as $p = e^{\frac{-\delta}{t}}$, where t presents the current temperature value, and δ indicates the difference in fitness value between the neighbouring inferior solution and the current solution.

2.5.3.3 Genetic Algorithms

The name Genetic Algorithm reflects the analogy between encoding of candidate solutions as a sequence of simple components, and the genetic structure of an individual [90]. Such components are referred to as individuals or chromosomes. Normally, a Genetic Algorithm uses a binary representation, i.e. candidate solutions are encoded as strings of 1s and 0s.

Population refers to the set of candidate solutions that are currently concerned; however, generation refers to each successive population considered. The first generation is made up of randomly selected individuals. Some domain information about problem is determined by selected chromosomes of population, which may increase the chances of the search converging on a set of highly-fit candidate solutions. Therefore, each individual of the population is evaluated for fitness. Holland [50] introduces the original Genetic Algorithm which includes the selection of 'fitness-proportionate' which determines the expected times of selecting an individual for reproduction, that is proportionate to the individual's fitness in comparison with the rest of the population. However, fitness-proportionate selection has been criticised because highly-fit chromosomes appearing early in the advance of the search tend to dominate the selection process, leading the search to converge earlier on one sub-area of the search space [49].

Eventually, the following generation of the population is selected for the reinsertion stage, and the new individuals are evaluated for fitness. The GA continues in this loop until it reaches a solution of global optimal.

2.6 Conclusion

This chapter reviews the relevant topics to the main research topic, such as Stable Marriage Problem and its variants. Also, the Clone Detection is deeply reviewed and investigated as it is one of the main parts of the conducted case studies. The current status of service selection is reviewed.

Several search-based algorithms are investigated to be compared with the proposed matching-based approach. This chapter gives a solid knowledge of the relevant areas, which is a cornerstone to represent and understand the proposed research topic. This section summarises the reviewed topics as follows:

- Stable Marriage Problem and its variants, Clone Detection and Service selection are reviewed.
- To compare the matching-based approach, some search-based algorithms are introduced.

Chapter 3

SMP Extensions

Objectives

- Leveraging SMP algorithms in software engineering.
 - Defining the selective strategy.
 - Presenting the extended SMP algorithm.
-

3.1 Overview

Our research is focusing on increasing the efficiency of the stable matching algorithms with respect to software engineering field. We concentrate on the Gale and Shapely algorithm (Stable Marriage Problem) SMP; trying to increase the detection range of similar candidates to be matched semantically using ontologies. Also, as long as SMP algorithm allows only the candidates of the first set (Men) to propose to their first choices, our research devoted to increase the fairness of SMP algorithm by allowing the candidates of the second set (Women) to make their own choices i.e. propose to the best of their choices of the opposite set. Moreover, we found another way to increase the fairness by allowing the candidates of the first set (Men) to enter the competition again to propose for their second choices even if they have already been assigned to their first choices; to fulfil the best choices (wishes) of the candidates of the second set (Women); this will affect the cardinality of the SMP algorithm to be one-to-many rather than the original one-to-one; it is similar to the HR problem.

Our approach shows the possibility of extending the SMP algorithm to enhance the matching process in software engineering area, as already known SMP algorithm has many derived algorithms such as SMPI, SMPIT. We would like to dynamically change the SMP algorithm to meet some purposes of software matching and other discipline. We add some modifications to the preferences of the SMP algorithm to update the sets of preferences (to receive new inputs during the matching process for example). We also mentioned the possibility of analysing the pseudo code using metrics, and then applying the SMP algorithm to find code similarity.

The main aim of this research is the development of a comprehensive matching framework to accomplish a high-level of features, to help in multi purposes and different aspects in Software Engineering. It has been noticed that SMP algorithm does not allow the candidates from the same set to involve into an agreement in which they can speed up the process of choosing their preferences upon the opposite candidates. However, this is considered in our approach trying as best as possible to apply more fair assignments. For example, if A1 has not been allocated to B1 which is its first choice, the algorithm will do the best to fit A1 with its second choice and so on (trying to alleviate the problem).

In semantic sense, we would derive equivalence candidates from each candidate to increase the chances of detecting related candidates as have been shown in Figure 3.5 below. We build the preferences in two different phases (or two different lists of candidates, clarified later). First, it is build based on the original candidates as normal. Second, it is build based on the equivalence candidates which in this case acts as original ones as have been depicted in Figure 3.6.

Our approach, also considers a cardinality of many to many, which means a man can be engaged to more than one woman and vice versa. To fulfil the requirements of matching process of some software engineering areas (Clone detection etc.); the SMP algorithm needs to be properly modified and consider such cardinality. It is a valued change to SMP algorithm as currently it is at the cardinality of one to one. These modifications met the needs of clone detection process, as long as there are many similar fragments of codes match many similar fragments of code in the opposite set in the same instance.

We improve the SMP algorithm to fulfil the required purposes by adding three different extensions; firstly, allowing dual proposes; secondly, allowing dual multi allocation and finally deriving the equivalence candidates using ontologies.

3.2 Dual Proposed

3.2.1 Overview

Since the SMP algorithm allows only the candidates of the first set (Men) to propose to their first choices, our research devoted to increase the fairness of SMP algorithm by allowing the candidates of the second set (Women) to make their own choices i.e. propose to the best of their choices of the opposite set.

Our approach considers a dual proposed technique that allows the candidates from both sets to propose for their preferences. This slight amendment has enhanced the precision of the matching process, it is illustrated in Figure 3.1.

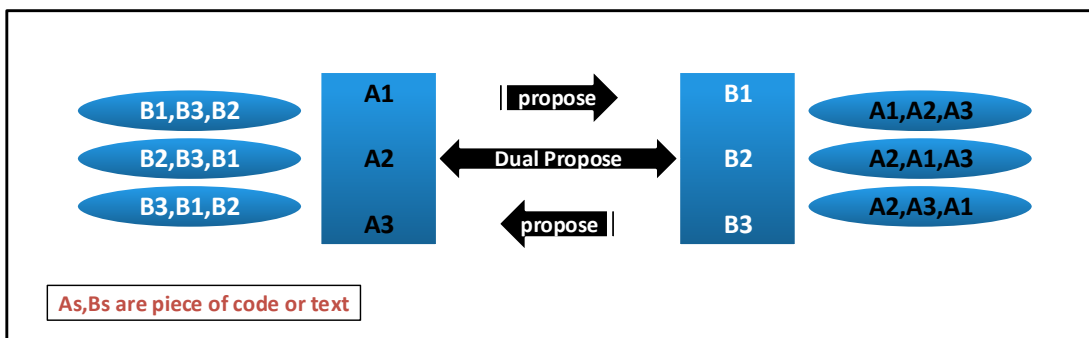


Fig. 3.1 Dual Propose Technique.

Practically, this process gives two different stable matched pairs; respectively man-optimal and woman-optimal. Thus, we enclose a novel way of assigning men and women to each other; by add a selective strategy. However, this strategy helps to judge which of the two pairs is more optimal. In the late section of this chapter the selective strategy is explained. Figure 3.2 shows a partial example of the strategy.

3.2.2 Dual Proposed Algorithm

Dual proposed algorithm acts as both man-optimal and woman-optimal. The combination of these two versions supplemented with a selective strategy using love's degree factor, result in complete concrete dual proposed scheme.

The algorithm of Dual Proposed consists of two phases followed by the selective strategy as following (Algorithm 7):

- **Phase 1 man-optimal algorithm.**

- **Phase 2 woman-optimal algorithm.**

- **Apply selective strategy.**

The above algorithm formed of three main parts. First and second parts use the original SMP algorithm. First part presents the man-oriented algorithm in which men have the priority and gain the best choices against the women. Second part presents the women-oriented algorithm in which women have the priority and gain the best choices against the men. The final part is the selective strategy which selects the ultimate output based on the two different outputs of the first and second parts.

Steps 1 to 11 illustrate the original Man-oriented algorithm (SMP algorithm). In these steps men get paired to their best choices. A man m may is assigned to a women w yields to the relation of one-to-one. The output of these steps is assigned to the matching M_1 . Men and women act as if services and requests which need to be mapped according to the relation provided by the algorithm.

Algorithm 7 Dual Proposed Algorithm

```

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;
11: assign  $M1$  to the man-optimal pair;
12: assign each person to be free
13: while some woman  $w$  is free do
14: begin
15:  $m :=$  first man on  $w$ 's list;
16: if some woman  $s$  is engaged to  $m$  then
17: assign  $s$  to be free;
18: assign  $w$  and  $m$  to be engaged to each other;
19: for each successor  $w'$  of  $w$  on  $m'$  list do
20: delete the pair( $w', m$ )
21: end;
22: assign  $M2$  to the woman-optimal pair;
23: apply selective strategy on  $M1$  and  $M2$ 
24: end;

```

Steps 12 to 22 illustrate the original Woman-oriented algorithm (SMP algorithm). However, in these steps the women gain their best choices against men as previously showed in steps 1 to 11 where men get their best choices. The ultimate output of these steps is assigned to the matching $M2$.

Step 23 applies the selective strategy which is defined in section 3.2.3 to get the final results and providing a relation of many-to-many based on both $M1$ and $M2$.

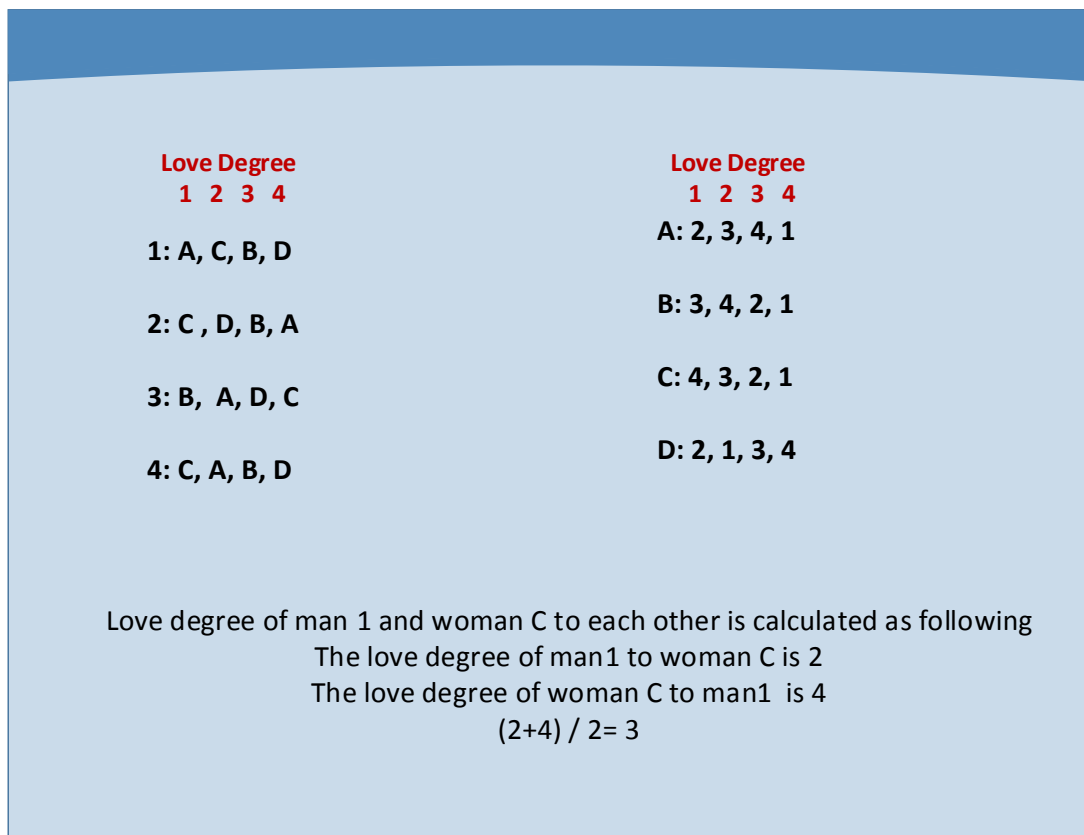


Fig. 3.2 love's degree factor.

3.2.3 Selective Strategy

The selective strategy acts as the main engine for the two proposed adaptations, and uses some factors (love degree and contrast degree) to select the optimal pairs. The selective strategy considers the contrast's degree factor between the elements in order of choosing the optimal pair. This factor helps when two different pairs has the same love's degree. Also, when more than one candidate have the same love's degree with a certain candidate, then the right candidate will be chosen. The contrast's degree reflects the difference between the

assigned candidates. For example the contrast's degree between man1 and woman C is 2 as depicted in Figure 3.2.

The selective 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 (services etc.). However, to converge these views, the degree of love for both participating (in the same pair) candidates is added and then the result is divided by two. The final result is the love's degree of the pair.

Whereas, the contrast's degree reflects the difference between the actual love's degree of the involved candidates (services etc.). Thus, the most preferable pair is that with small difference in its contract'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 3.3 depicts the selective strategy scheme.

With regards to the example in Figure 5.3 the output based on this strategy slightly differs. We provides several solutions some of which allow a candidate to be assigned to more than one candidates from the opposite set of participants, which reflects the cardinality of "many to many". However, as overall comprehensive observation the new algorithm provides slightly different matching pairs from the original man-optimal algorithm.

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, with regards to the recent extensions to the main SMP algorithms the need to a strategy to choose the optimal pair is raised and essential.

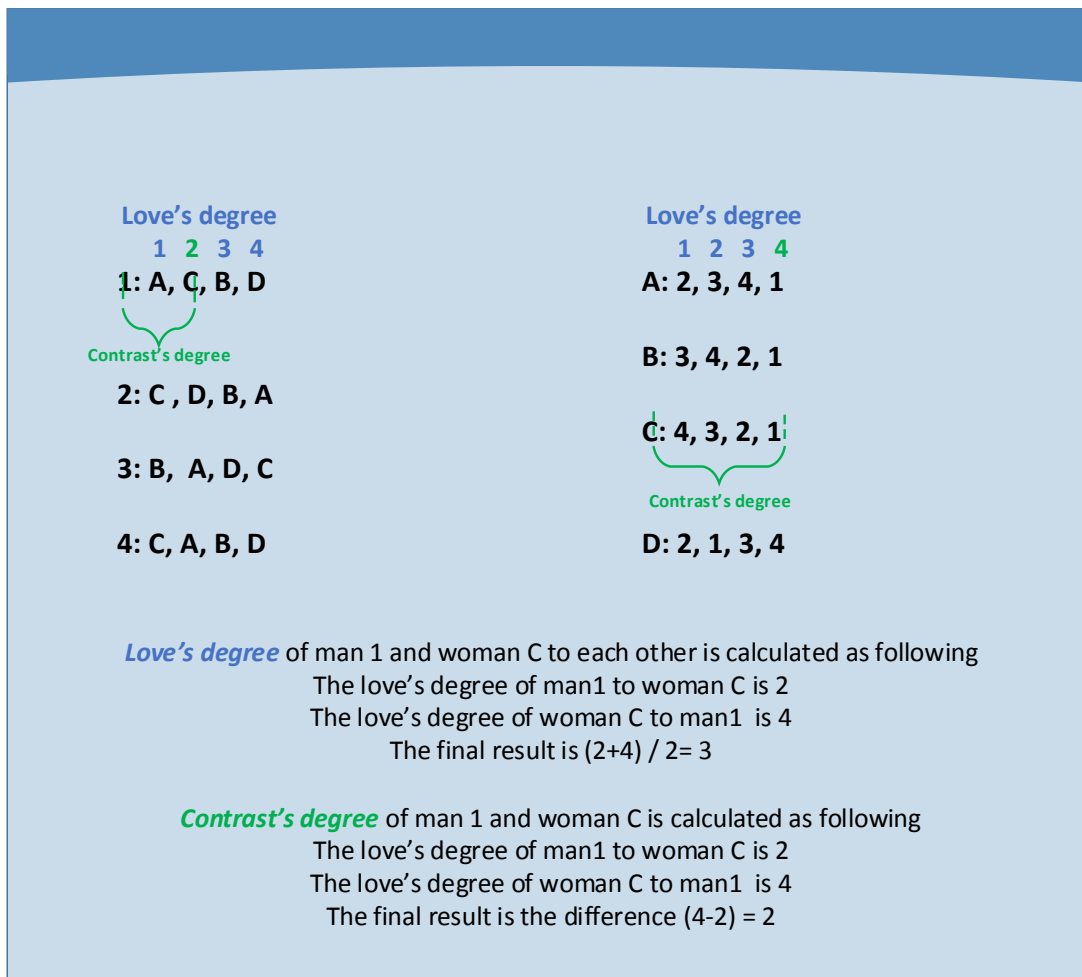


Fig. 3.3 Selective Strategy Scheme

Therefore, we provide a competitive Selective Strategy to support the newly built extension, at its ultimate parts to help choosing the optimal pair.

In our approach, a cardinality of many to many is considered, which means the man can be engaged to more than one woman and vice versa. To fulfil the requirements of matching process of some software engineering areas (Service Selection etc.); the SMP needs to be

properly modified and consider such cardinality. It is a valued change to SMP algorithm as currently it is at the cardinality of one to one.

These modifications met the needs of clone detection process, as long as there are many similar fragments of codes which match many similar fragments of code in the opposite set in the same instance. Our extension of SMP algorithm encloses a novel way of assigning men and women to each other; by add a love's degree factor. Figure 3.2 shows the love degree extension and gives an example of how it is calculated.

The following example in Figure 3.4 illustrates the process of the introduced strategy as presented in algorithm 7 against the original SMP algorithm:

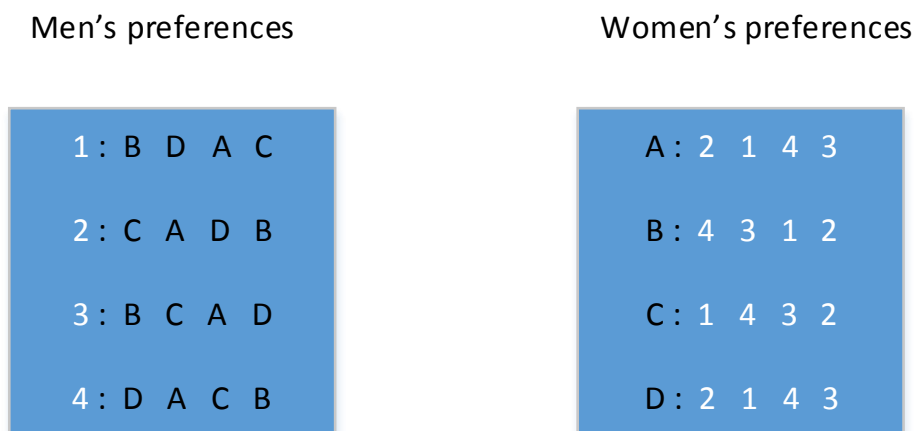


Fig. 3.4 Example of selective strategy.

The original SMP algorithm shows the following results. These results are different according to who starts to propose (man-oriented or woman-oriented).

$$\text{Man - Optimal} = M0 = (1,D), (2,C), (3,B), (4,A)$$

$$Woman - optimal = Mz = (1,D), (2,A), (3,B), (4,C)$$

Based on the previous output, it can be seen that men 2 and 4 are assigned to two different women in both orientations. Therefore, the selective strategy is applied to give more precise solution. The following two tables give the new solutions based on the factors used in the selective strategy based on the output of the original SMP algorithm which finds the final solutions in a quadratic time.

Table 3.1 New Man-Oriented

M0	Love degree	Contrast degree	Weight
(1,D)	2	0	1
(2,C)	3	3	3
(3,B)	2	1	2
(4,A)	2	1	2

$$NM0 = New - Man - oriented = (1,D), (2,A), (3,B), (4,A).$$

Table 3.2 New woman-Oriented

Mz	Love degree	Contrast degree	Weight
(1,D)	2	0	1
(2,A)	2	1	2
(3,B)	2	1	2
(4,C)	3	1	2

$$NMz = New - Woman - oriented = (1,D), (2,A), (3,B), (4,A), (4,C)$$

Selective strategy is calculated using the union of both new outputs $NM0$ and NMz as follows:

$$NM0 \cup NMz = (1,D), (2,A), (3,B), (4,A)$$

3.2.4 Semantic Equivalence

This feature is attached to the previous extensions to increase the detection range of similar candidates to be matched semantically using ontologies. This should be implemented through two different phases as illustrated in figures[3.5 and 3.6].

Figure 3.5 presents the initial stage of the process; which gets the synonyms of the main original candidates using ontology.

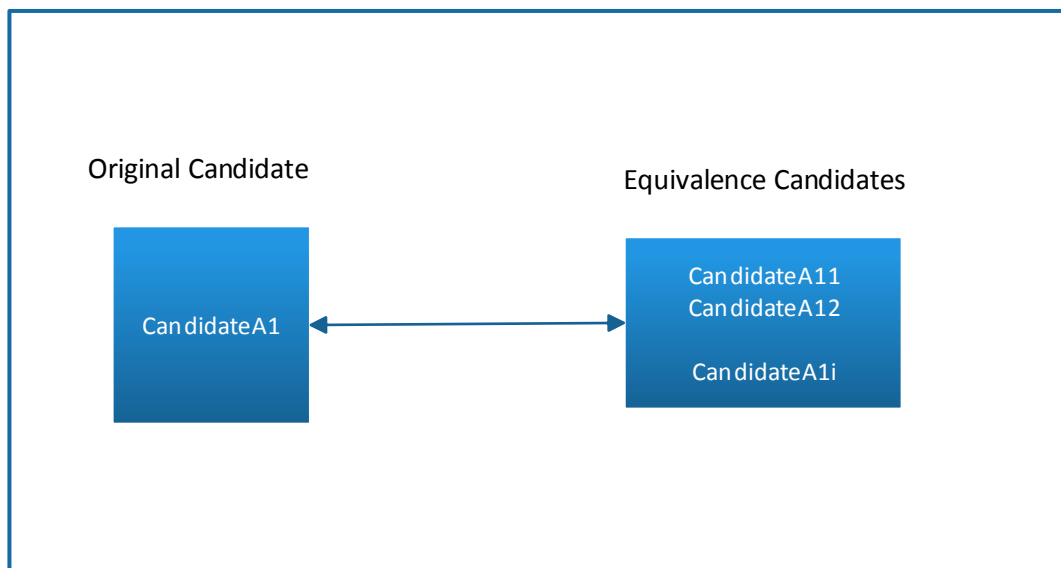


Fig. 3.5 Semantic Equivalence Candidates.

Figure 3.6 shows the virtual candidates which illustrated in newly formed two sets. These candidates derived originally from the main candidates as depicted in Figure 3.5. This final stage of the proposed extension matches the virtual candidates using the SMP algorithm.

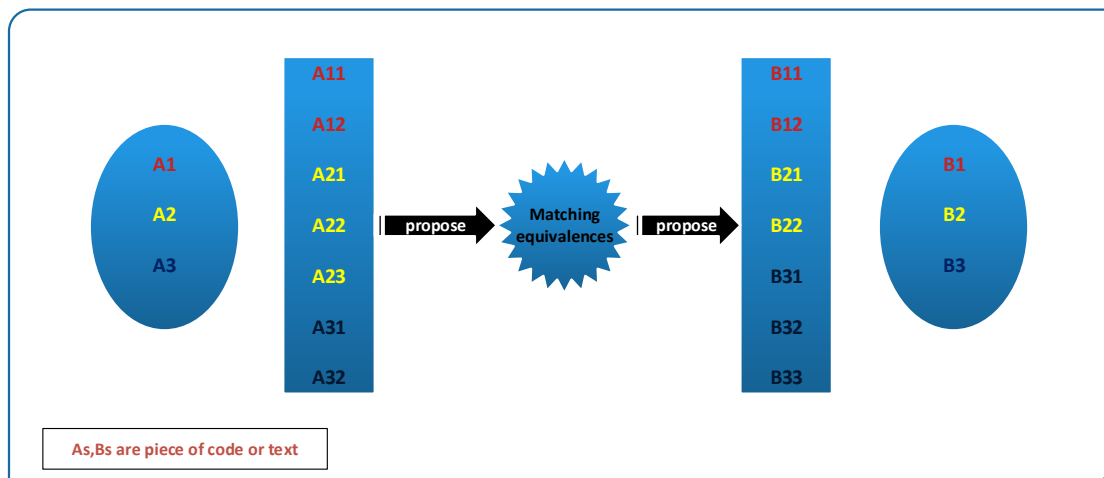


Fig. 3.6 Semantic Equivalence Candidates (SMP-based).

3.2.5 Evaluation

The presented matching scheme covers a noticeable hole of the current state of the original SMP algorithm and perfectly suited to the world of software engineering, serving in several aspects the needed allocation process that requires a high consideration of both sides of matched candidates. Our Scheme has increased the quality of allocating the similar services to each other, through considering the desire of the matched services, which result in increased satisfaction of the candidates in each pair, reflects a high stability. However, the Scheme has some limitation in terms of the time feature, as the consideration of the candidates' desire needs more computation and recursion to fulfil and reach the highly required stability. The implementation shows a smoothly scalable Scheme which performed in a large environment. It shows efficiency in its performance and the final results.

3.3 Dual Multi Allocation

3.3.1 Overview

This extension allows the candidates of the both first and second set to enter the competition and propose again for a certain times to their preferences. Each candidate of the first set may have more than one matched participants of the second set and vice versa. Since, the current state of the stable marriage problem algorithm at the cardinality of “one to one” and later covered the “one to many” which is presented in Hospital and Resident problem; this extension will consider the “many to many” cardinality. Such extension serves wide range of real software applications that already exists.

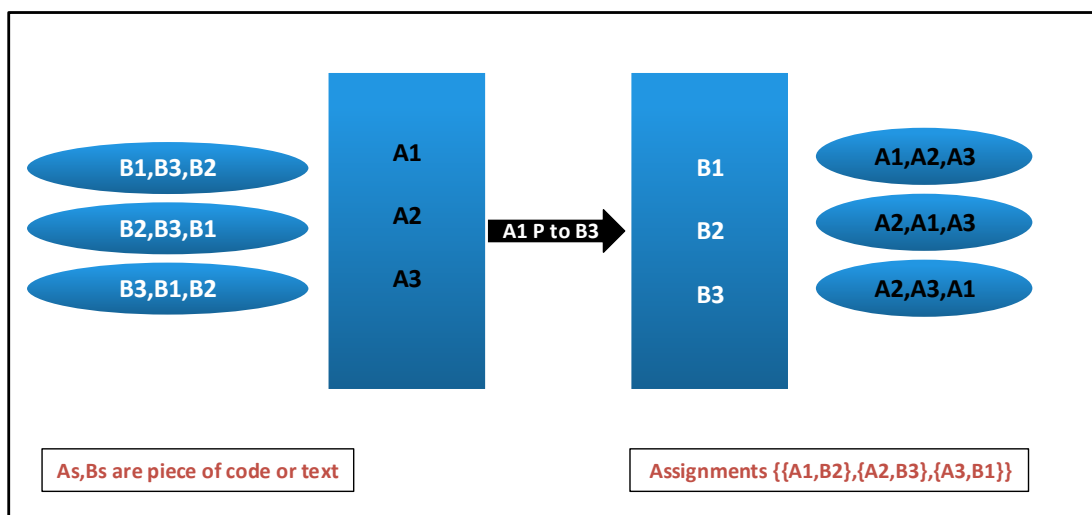


Fig. 3.7 Multi-proposed Technique.

Therefore, dual multi allocation considers multiple matched participants with different level of love's degree factor to the main candidates of both sets. Results in several stable matched pairs that annotated by the stable's degree.

3.3.2 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 of two phases followed by the selective strategy as following:

- **Phase 1 Hospital-Oriented-Man algorithm.**
- **Phase 2 Hospital-Oriented-Woman algorithm.**
- **Apply selective strategy.**

The above algorithm formed of three main parts. First part presents the Hospital-oriented algorithm in which hospitals have the priority and gain the best choices against the residents. Second part presents the residents-oriented algorithm in which residents have the priority and gain the best choices against the hospitals. Final part is the selective strategy which selects the ultimate output based on the two different outputs of the first two parts.

Steps 1 to 11 illustrate the original Hospital-oriented algorithm (HR). In these steps men act as hospitals and the women act as residents. Therefore, a man m may allocated by

Algorithm 8 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 selective strategy on  $M1$  and  $M2$ 
24: end;

```

many women w . the relation here is many-to-one. The output of these steps is assigned to the matching $M1$. Men and women act as if software methods which need to be mapped according to the relation provided by the algorithm.

Steps 12 to 22 illustrate the original Hospital-oriented algorithm (HR). However, in these steps the women act as hospitals and the men act as residents as previously showed in steps 1 to 11. The ultimate output of these steps is assigned to the matching $M2$.

Step 23 applies the selective strategy which is defined in section 3.2.3 to get the final results. Figure 4.1 shows an example of using algorithm 8 where the methods candidates are mapped to the similar methods based on their software metrics values. The time to reach the final solutions cost a polynomial time.

3.4 Conclusion

This chapter introduces the heart of our framework “selective strategy” which is used as the main engine of the new extensions. The chapter shows how to improve the current State of clone detection by introduces the extended Dual-Multi-Allocation algorithm.

Also, it shows the second main adaptation, Dual-Proposed algorithm which improves the current state of service selection. The use of semantic matching is discussed to strengthen the SMP-based approach in the process of service selection.

The problems of clone detection and service selection are similar however, the basic are different as the current status of service selection apply one to many relationship whereas the relation of clone detection process is arbitrary to fixed (hash function). Therefore, the two extensions are introduced to reach final purpose of this application which is to achieve the cardinality of many to many.

This section summarises the key points of this chapter as follows:

- The main engine of the introduced extensions is defined as “selective strategy”.
- The Dual-Multi-Allocation extension is introduced, which helps to improve the current state of Clone Detection (see chapter 4).

- The Dual-Proposed extension is introduced, which helps to improve the process of service selection (see chapter 5).
- It is discussed that how to strengthen the SMP-based approach using semantic matching.

Chapter 4

SMP-Based Clone Detection

Objectives

- To provide a clone detection scenario based on adapted SMP algorithm.
 - To present the SMP-based approach to clone detection.
 - To show a concrete clone detection example.
-

4.1 Overview

Code cloning is a severe problem that negatively affects industrial software and threatens intellectual property. This Chapter 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.

Clone detection has been 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 Chapter, 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. cyclomatic complexity (CC) and the number of calls from or to a method (MCIN & MCOU). The clone detection process should therefore consider the values of both parties.

4.2 Similarity Measurements

Several techniques for software clone detection have been introduced over so far. We provide background concepts of the process of clone detection. Also, most of clone detection

techniques are classified and presented with regard to different aspects. The aim of this Chapter is: providing a new method of detecting the clones, and to introduce our competitive detection method (SMP-based). Our method focuses on two facets. First, we introduce a new matched algorithm (Stable Marriage Problem) to the world of clone detection. Second, we use a variable granularity to detect maximum number of possible clones.

4.3 Metrics

Since our process is method-based we use some metrics to extract some features out of the matched fragments, such as method rank, number of the parameters and nested depth method. However, our approach might be extended to be a variable adaptive by considering a higher abstract level such as classes.

In the following we present some predefined metrics based on both methods and classes:

Based on the numeric metrics values, we will build the preferences list of each element (clone), from the opposite source file. We use a java plug-in with eclipse, known as Metrics 1.3.6 which calculates some metrics such as following:

- **NOM** presents the number of methods in a class.
- **LCOM-HS** describes the lack of cohesion between methods in the class.
- **NOC** indicates the number of subclasses of a certain class.
- **DIT** is the maximum inheritance path from a class to the root class.
- **WMC** indicates the total weights of the methods in a class.

Based on the above metrics, we specify the characteristics of each method. However, in order to build up the preferences lists of every single method, the list needs to be ordered based on the most similar attributes of the opposite methods.

4.4 Detection Process

In Figure 4.1 we present a high level view of our detection method that SMP-based

In the following, we give short description of these phases: The phase abstraction is the basic transformation of the source code to source units, in our case presented as subtree of AST. The representation of a source unit is based on the used approach such as text-based, token-based, AST-based, or metric-based. However, in the first preparation phase, we prepare for classifying the cases by applying some annotations, for example case without parameters are A. Also, we can extend these annotations for several purposes (e.g. enables to remove getter and setter methods, remove redundant parentheses). In the next step which is the Assembly, we group the similar cases into several sets, these cases are represented as candidates (elements of their related sets). In the second preparation step, we unify these sets into a unique set and create a set of preferences for each candidate. These preferences are actual clones from the opposite source file. Thus there is link somehow between the matched files in this phase. In the pre-ultimate important stage which is the matching phase, we apply the SMP algorithm on the sets (two sets fileA and fileB) with its preference lists, forming an optimal clone pairs.

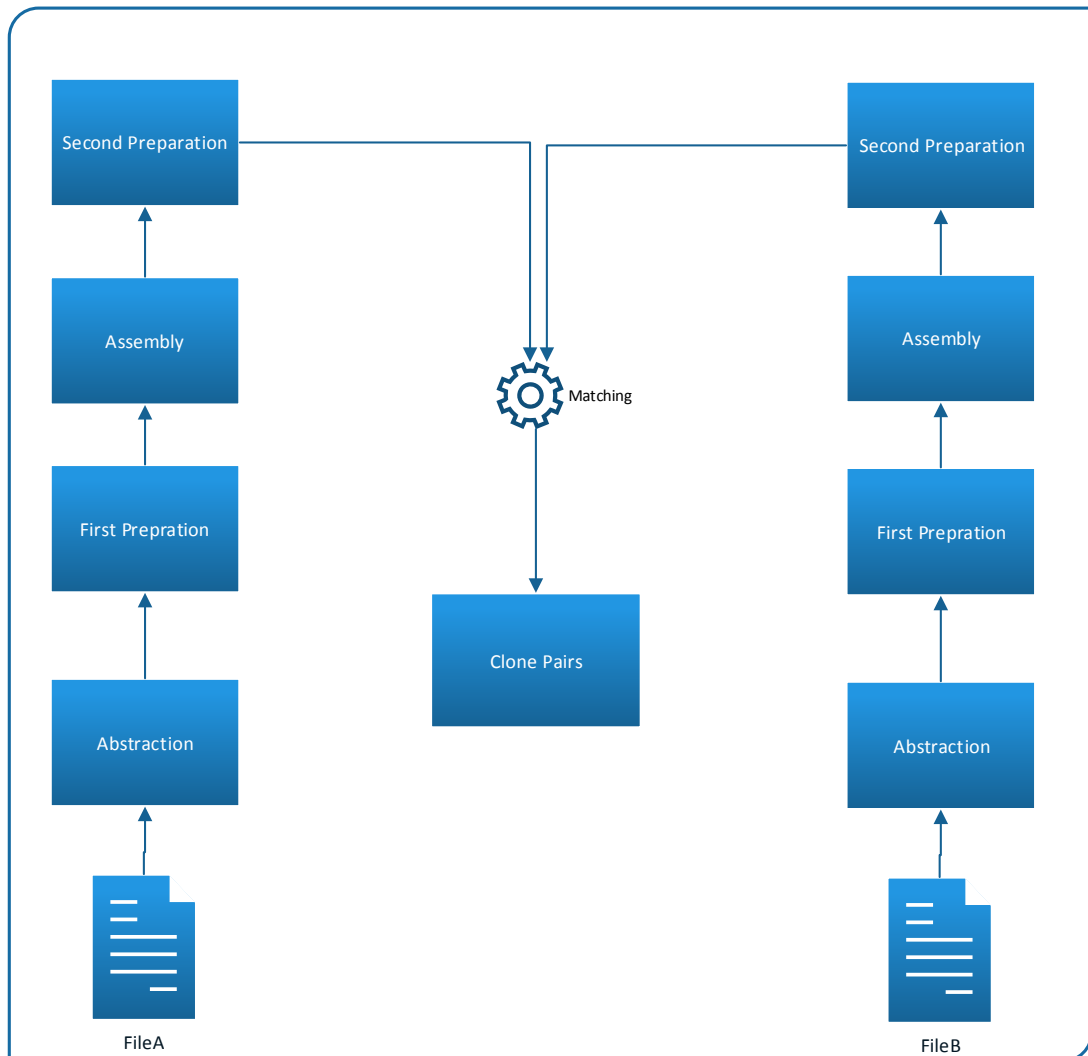


Fig. 4.1 Detection Process for Program Analysis.

4.5 SMP-based Clone Detection

Our experiment has been designed to concentrate on enhancing several features such as accuracy, satisfaction and stability of the current clone detection approaches. We have specified certain metrics to help in recognising the aspects of the related and similar portions of codes.

We proofed the remarkable efficiency of our approach by carrying out a case study on two middle sized source files. Each file has got a minimum of 100 of specified blocks. We have got a set of metrics 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. First, we took two different source files of the same program language with certain threshold-based of clone granularity.

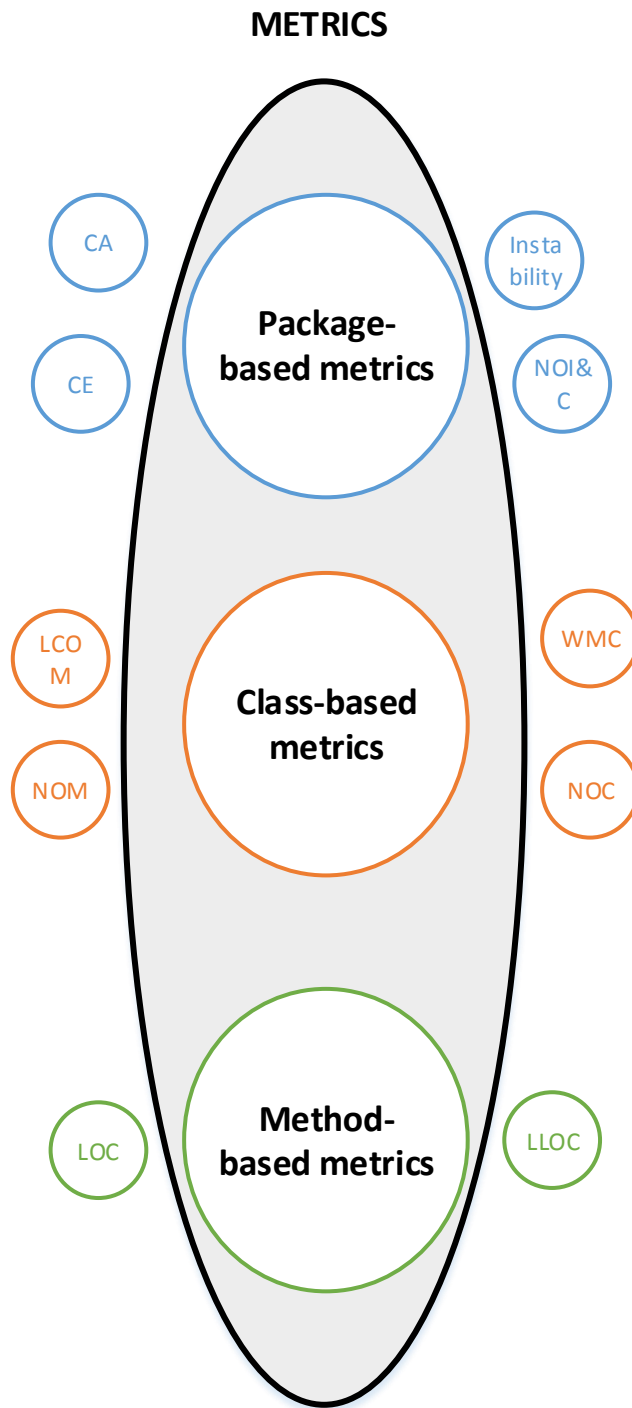


Fig. 4.2 Metrics level-based.

However the granularity-level may vary at certain points and stages to different abstract levels (e.g. class, package etc.) based on the situation of the matched source files. Figure 4.2 depicts several metrics based on abstraction levels.

The process has two different phases as following: **Phase1**: we build the preference list of each clone of first source file (in this case the criteria is the similarity) from the second source file's clones, starting by the most similar block and so on. We do the previous step for the second file. **Phase2**: we apply the original SMP algorithm with observing some appointed features for both the used algorithm (e.g. speed & performance) and the status of the detected clones (e.g. accuracy). After that we swap the original SMP algorithm with our extended algorithm with considering the specified features.

How does it work?

To apply the SMP algorithm in clone detection, it needs first to build the preference lists of both code fragments. This can be achieved by 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 4.3 depicts a general prototype of code clones (method-based).

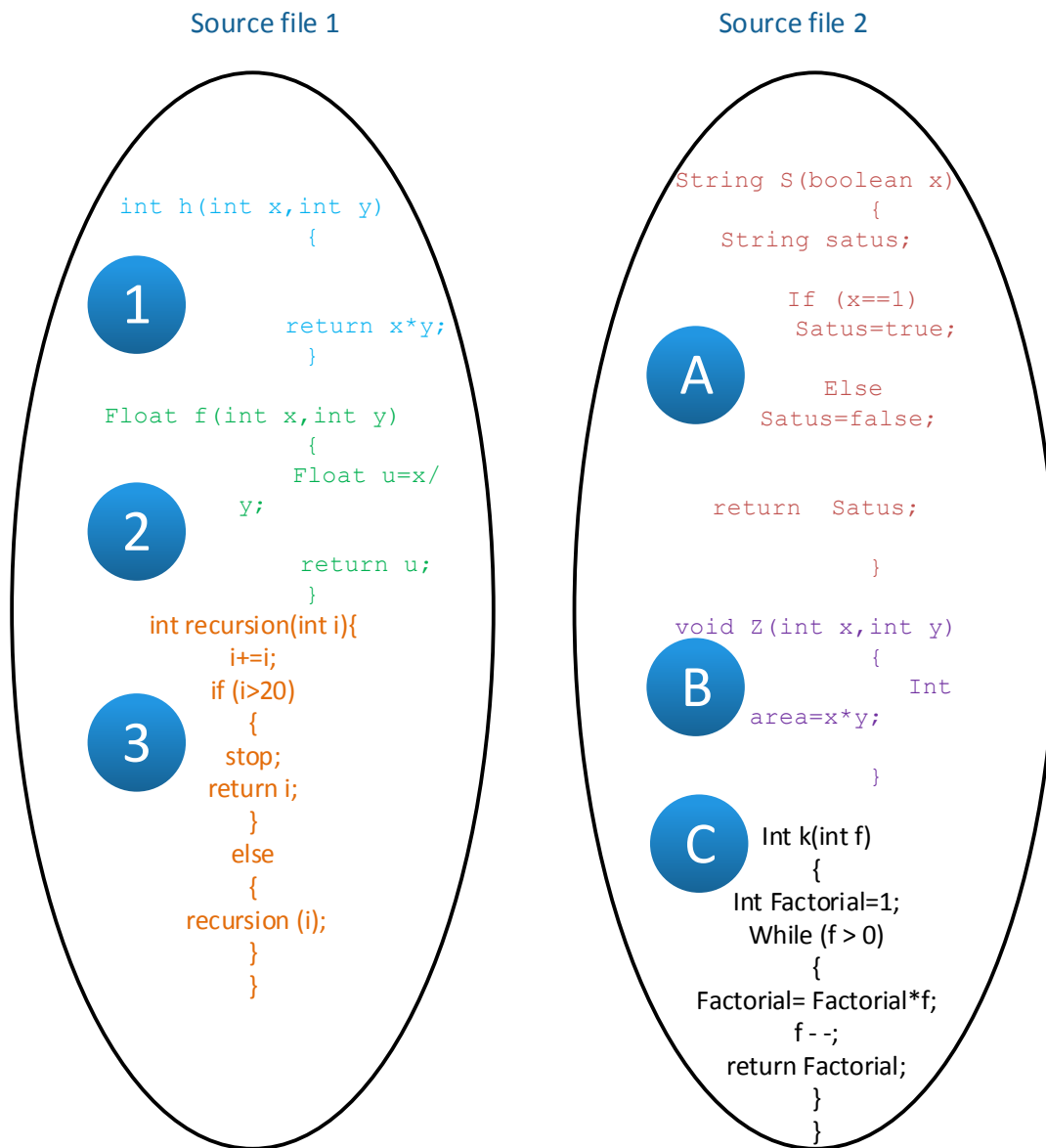


Fig. 4.3 Detection Process for Program Analysis.

So let's zoom-in into one of these clones and get the assigned metrics' values to understand its nature, ease of cooperating to act as a candidate (see Figure 4.4).

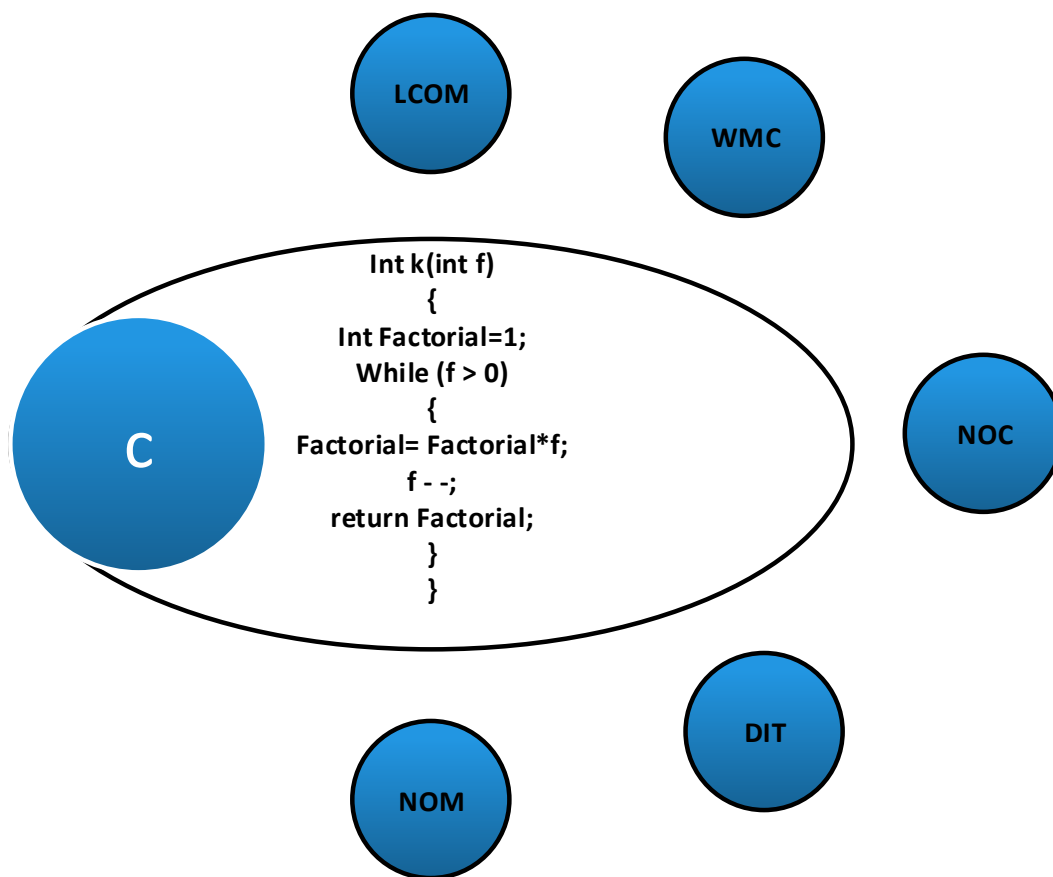


Fig. 4.4 High presentation of code fragment along with its measurements.

We have considered some metrics which reflect the aspects of each clone; however, we have avoided the others due to the scope (clone-granularity) which we specified to act as a clone candidate.

4.5.1 The Scenario

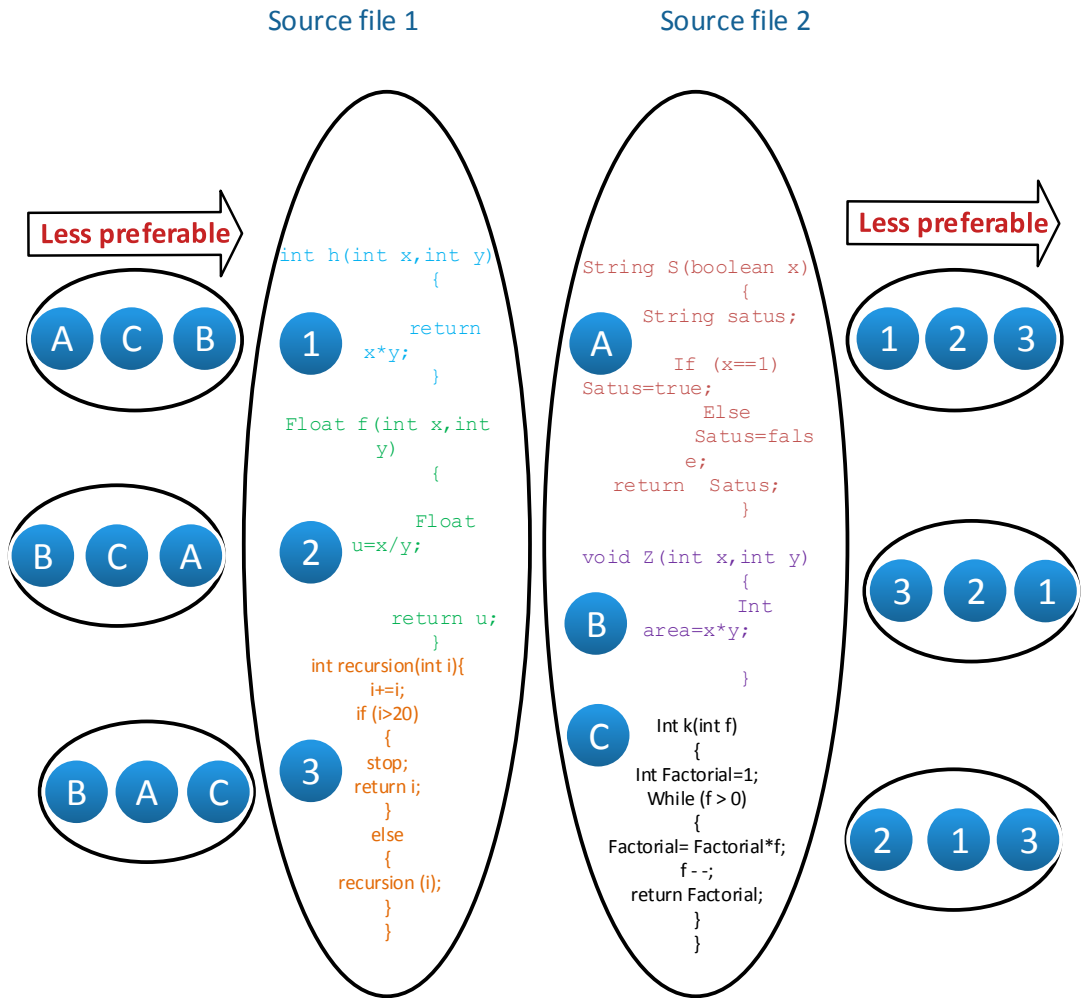


Fig. 4.5 Detection Process for Program Analysis.

4.5.1.1 The stage of Building the Preference Lists

The blocks of the first source file look for the similar blocks of the second source file, based on the values of the metrics of each fragments of codes. We use java plug-in with eclipse

1.3.3 known as Metrics 1.3.6 which gives the values of several metrics, to calculate the required metrics that we wish to consider in this process. We mentioned some of these metrics in an early section of this chapter.

Table 4.1 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
MCOU	Number of calls from a method

Table 4.2 Method Metrics.

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

SMP-based approach is similar to the approach presented by Mayrand et al. [88] as both approaches use several metrics to identify functions with similar metrics values as code clones. Also, as seen above metrics are calculated from names, layout, expressions, and (simple) control flow of functions. As a result functions with similar metrics values are clone. The main different between these two approaches is the algorithm being used, which accordingly SMP-based approach compete other approaches to obtain better results.

Table 4.3 Metrics of Source file 1.

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

Table 4.4 Metrics of Source file 2.

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

As seen with several approaches for instance CloneDR, the user can set the basic criteria of the search process which influence the end results according to their requirements for example, similarity threshold which reflects the ratio of identical to total code in the clone. Also, starting depth is one of these criteria which determine minimum tree depth in the clone (see Figure A.1).

SMP-approach gives a priority to each candidate (method) to order similar codes (possible clones) as a preference lists. Similar codes means acceptable to the minimum requirement or to a certain threshold. Also, with regards to the importance of certain software metric this means that some software metrics may more valuable than others in order to run this certain matching process. This reflects the status of the current approaches in term of detecting software clones as some of these tools are good in detecting some types of software clone which undetectable by others.

Methods in SMP-based approach are engaged to each other based on a semi-semantic way. This is done by selecting a group of software metrics that reflect the deep structure of the method (method hierarchy). This is the main reason that the SMP-based approach outperforms other approaches in detecting type 3. These software metrics are weighted individually and as a group based on the prior requirements.

The weight may vary based on the values of software metrics (Variable weight) for example if both NBV and NBP weighted by 20% as individual, this may be increased when they have very close metrics' values with other method so the chance of the similarity (detecting clone) is increased. However, there are certain thresholds of the metrics' values indicate that both methods are not similar; it can be called unwanted method. The following table4.5 shows the initial weight of the conducted software metrics.

Table 4.5 Metrics Weight.

Abbreviations	Description	Weight
LOC	Number lines of code	10%
Nbp	Number of parameters	20%
Nbv	Number of variables declared in the method	20%
CC	McCabe's Cyclomatic Complexity	25%
NBD	Nested Block Depth	25%

From the weight priority above, it can be noticed that the SMP-based approach focuses on the structure of the method to find the cloned methods. However, SMP-based approach pairs the most similar methods to each other and discards the unwanted methods from the competition at early stage, which means not listed in the preference list of the method being

matched with their similar methods. This is a filtering stage based on the software metrics values and forms the pre-step to the build the preference list that required in the algorithm.

With regards to the weight priority in tables 4.3 and 4.4, the preference lists of involved candidates appear as follows (Figure 4.6):

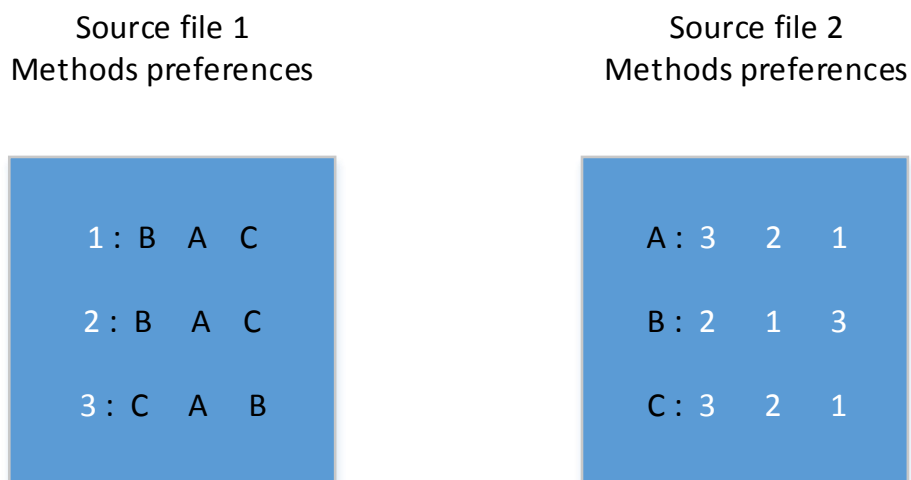


Fig. 4.6 Methods preference.

4.5.1.2 The Stage of Running the SMP Algorithm

The stage of running the SMP algorithm each block of the first source file should be engaged to one and only one block of the second source file and vice versa. Therefore, the blocks of the first source file start to propose to their best choices of their preference list. The matching result out of running the SMP algorithm based on the above Figure 4.5 is M (a set of matched pairs).

4.5.1.3 Extended SMP Algorithm (Dual-Multi-Allocation) for Clone Detection

SMP algorithm 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 algorithm 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 adaptation has enhanced the precision of the matching process; it is illustrated in Figure 4.7 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 algorithm 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 selective strategy. This strategy helps to choose the pairs which form similar code clones to a certain threshold. This algorithm results in several stable matching pairs with dissimilar allocated candidates based on love's degree, which can be

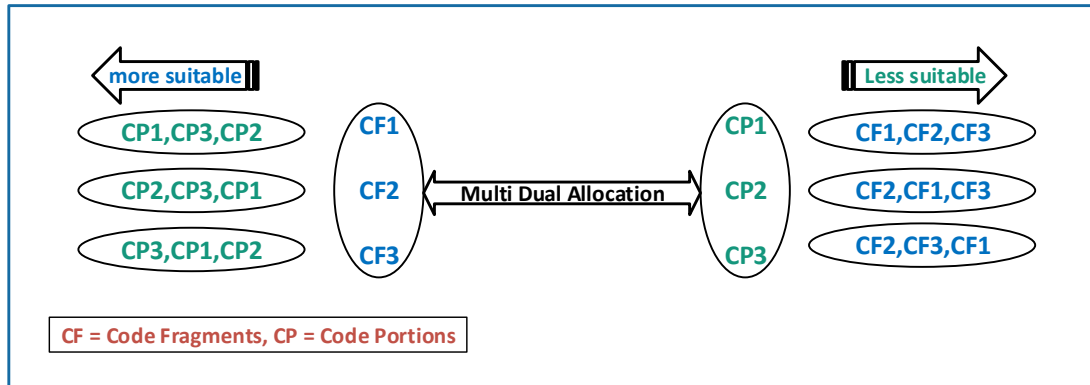


Fig. 4.7 Dual Multi Allocation.

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.

However, to increase the fairness of this assignments between the similar fragments of the aforementioned different source files, a new strategy has been defined and added to the adapted SMP algorithm (Chapter 3). Also to make the detection process more accurate and satisfiable.

4.5.2 Discussion

The output of the previously depicted prototype in Figure 4.3 shows a remarkable efficiency of our approach. 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. It is noticed that the extended algorithm benefits from some features such as performance and the accuracy of the detected software clones.

This means that it is possible to develop match making code fragments that are 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 were undetectable with most of previous clone detection approaches. However, the time complexity is challenging in this newly adapted algorithm, which is still the same as the original SMP (polynomial time).

4.6 Conclusion

This chapter introduces some software metrics based on different abstract levels to help in measuring the software portions. Also, two-phase workflow for Clone Detection is presented, to show the steps of SMP-based approach.

A middle-sized prototype is tested to apply the new adaptation of the SMP (Dual-Multi-Allocation). This section summarises the key points as follows:

- Some software metrics based on different abstract levels, which help in measuring the software portions are introduced.
- A two-phase workflow for Clone Detection is presented:
 - Building the candidates' preference lists.
 - Applying the new adapted algorithm.

- To apply the new adaptation of the SMP (Dual-Multi-Allocation) a small prototype is tested.

Chapter 5

SMP-Based Service Matching

Objectives

- To provide a service selection scenario based on adapted SMP algorithm.
 - To present the SMP-based approach to service selection.
 - To show a concrete service selection example.
-

5.1 Overview

Service-oriented computing is establishing itself as a new computing paradigm in which services advertise their capabilities within a network, and then are used, composed and orchestrated by other services and end-users. Whilst many approaches to matching service providers with consumers of their services have been developed in the past, the proposed approach in this paper takes a different view of the problem in that it does not look for the fittest individual utility, but views it as a constrained optimisation problem that matches between a set of services and a set of service requests. Our approach addresses this problem using an adaptation of the well known stable-marriage problem and demonstrates how matching between services and requests to a certain threshold can be expressed. This will contribute a fair assignment between services and requests based on their preferences. As the current state of the service selection process considers only the view of requests, the proposed approach can ensure several features to the services such as service protection and service quality, e.g. it can ensure the preservation of service availability by redirecting a coming request to a similar service if the current service is busy.

Recent research of service selection focuses on matching behaviour, trying to interact with services in a smart way. This can positively impact several relevant big topics on service-oriented computing, such as cloud computing which provides both software and resources as services, and service composition in which aggregation of services are composed to automate a certain job.

In this Chapter we will apply a variant of the stable marriage problem to service oriented computing (SOC) and use it to match between service providers and service consumers.

Current selection algorithms take the QoS (Quality of Services) constraints of the requestor (service consumer) into account, but do not consider the preferences of the service provider in the process. Our work expands this to include preferences that a service provider may have towards service consumers. Example preferences of a service provider may include returning customers, payment options and country of origin. The web service selection process should therefore consider the values of both parties.

5.2 Dual-Proposed for Service Selection

5.2.1 Dual-Proposed Scheme

Dual proposed is one of two main adaptations presented in this thesis. The proposed approach considers a dual proposed technique that allows the candidates from both sets to propose to their preferences. This crucial modification has enhanced the precision of the matching process; it is illustrated in figure 5.1 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 service selection the concept of similarity plays an important role. Therefore, aforementioned extension of the current state of SMP algorithm is necessary to be effectively applied in such applications.

A novel matching scheme is needed to achieve smart interaction between the services and their corresponding requests. This led to a quality satisfaction for selecting the appropriate services to the equivalent requests as the original SMP algorithm base its matching process on the desire of the candidates of the first set regardless of the similarity factor. However, in

the process of service selection the similarity between a certain service and the corresponding request forms the corner stone. This enhancement increases the novelty of SMP algorithm, as it will help to contribute and serve many applications in different aspects of service matching or selection.

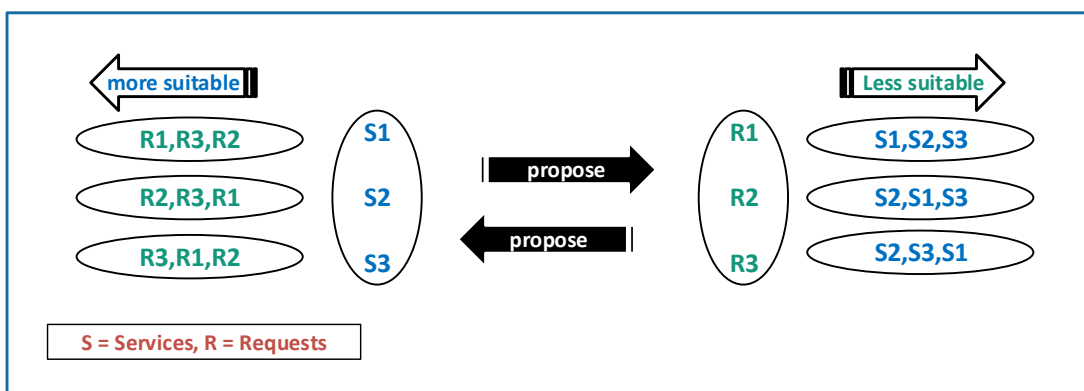


Fig. 5.1 Dual Proposed.

Practically, this process gives two different stable matched pairs; respectively man-optimal and woman-optimal. Thus, we enclose a novel way of assigning men and women to each other by adding a selective strategy. This strategy helps to choose the most optimal pair among the final result.

This significant adaptation increases the performance of the original SMP algorithm especially in the process of service selection. The importance of this extended algorithm is accomplished by implicating aforementioned selective strategy in Chapter 3.

5.2.2 Service Matching

To apply the SMP algorithm in services, it needs first to build the preference lists of both service requirements and service providers. This can be achieved using service profile to specify the most similar related participants (service requirements or service providers). Thus, each service requirement needs to strictly order the service providers based on the similarity and vice versa (similar to Figure 7). However, the SMP algorithm performs a single assignment (one-to-one) for the involved candidates, which at some points does not help especially in the case of allocating more than one service requirements to the right service provider for example. Dual propose algorithm has been proposed to fulfil this requirement which widely needed in such fields. Figure 5.2 below illustrates a brief depiction about our approach.

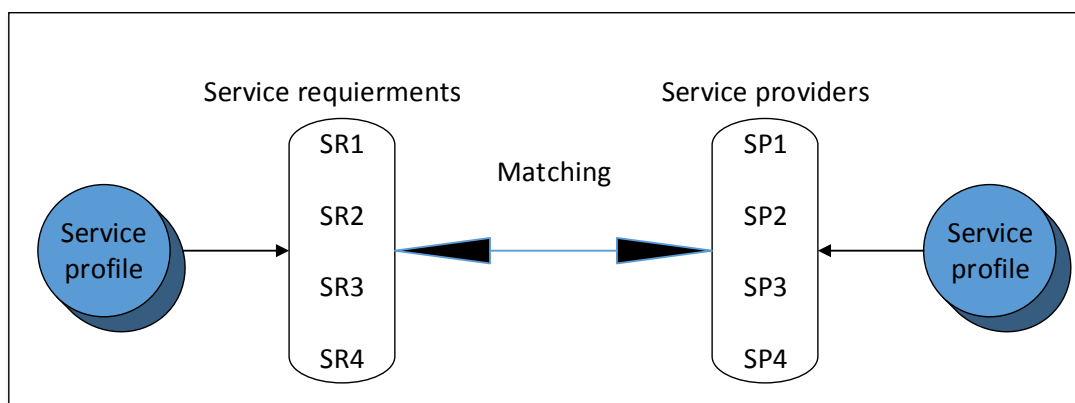


Fig. 5.2 Services Matching with Dual Propose Algorithm

In the following, we apply our approach to the previous example, presented in Figure 2.3:

Table 5.1 ServiceView

M0	Love degree	Contrast degree	Weight
(SR1,SP4)	2	0	1
(SR2,SP3)	3	3	3
(SR3,SP2)	2	1	2
(SR4,SP1)	2	1	2

ServiceView = *New – Man – oriented* = (SR1, SP4), (SR2, SP1), (SR3, SP2), (SR4, SP1).

Table 5.2 RequestView

Mz	Love degree	Contrast degree	Weight
(SR1,SP4)	2	0	1
(SR2,SP1)	2	1	2
(SR3,SP2)	2	1	2
(SR4,SP3)	3	1	2

RequestView = *New – Woman – oriented* = (SR1, SP4), (SR2, SP1), (SR3, SP2), (SR4, SP1), (SR4, SP3).

The combination of both new versions, each candidate may be assigned to more than one candidates and this based upon the features considered by the selective strategy.

Similarly The following example in Figure 5.3 considers the selective strategy with different symbols. This example shows different output from the main SMP algorithm, which allows a candidate to be assigned to more than one candidate from the opposite set of participants, which reflects the cardinality of many-to-many. The new algorithm provides different matching pairs from the original man-optimal algorithm.

$M0 = \{ (S1,R5) , (S2,R3) , (S3,R8) , (S4,R6) , (S5,R7) , (S6,R1) , (S7,R2) , (S8,R4) \}$

S1: R5 R7 R1 R2 R6 R8 R4 R3	R1: S5 S3 S7 S6 S1 S2 S8 S4
S2: R2 R3 R7 R5 R4 R1 R8 R6	R2: S8 S6 S3 S5 S7 S2 S1 S4
S3: R8 R5 R1 R4 R6 R2 R3 R7	R3: S1 S5 S6 S2 S4 S8 S7 S3
S4: R3 R2 R7 R4 R1 R6 R8 R5	R4: S8 S7 S3 S2 S4 S1 S5 S6
S5: R7 R2 R5 R1 R3 R6 R8 R4	R5: S6 S4 S7 S3 S8 S1 S2 S5
S6: R1 R6 R7 R5 R8 R4 R2 R3	R6: S2 S8 S5 S3 S4 S6 S7 S1
S7: R2 R5 R7 R6 R3 R4 R8 R1	R7: S7 S5 S2 S1 S8 S6 S4 S3
S8: R3 R8 R4 R5 R7 R2 R6 R1	R8: S7 S4 S1 S5 S2 S3 S6 S8
Services' Preferences	Requests' Preferences

Fig. 5.3 Service marriages (Similarity-based).

So, the output after applying our strategy is as following:

New-Service-Optimal= $M'0 = \{ (S1,R5) , (S2,R3) , (S3,R2) , (S4,R8) , (S5,R7) , (S6,R1) , (S7,R7) , (S8,R4) \}$.

$Mz = \{ (S1,R3) , (S2,R6) , (S3,R2) , (S4,R8) , (S5,R1) , (S6,R5) , (S7,R7) , (S8,R4) \}$

So, the output after applying our strategy is as following:

New-Request-Optimal= $M'z = \{ (R1,S5) , (R1,S6) , (R2,S3) , (R3,S2) , (R4,S8) , (R5,S6) , (R6,S4) , (R7,S5) , (R8,S3) \}$. = $\{ (S2,R3) , (S3,R2) , (S3,R8) , (S4,R6) , (S5,R1) , (S5,R7) , (S6,R1) , (S6,R5) , (S8,R4) \}$.

From above results we can derive the dual-propose-multi-allowance as following:

Service-view \cup Request-view = $\{ (S1,R5) , (S2,R3) , (S3,R2) , (S3,R8) , (S4,R6) , (S4,R8) , (S5,R1) , (S5,R7) , (S6,R1) , (S6,R5) , (S7,R7) , (S8,R4) \}$.

Table 5.3 Service-view

Pairs (m,w)	Love degree	Contrast degree	Weight
(S1,R5)	1	5	3
(S2,R3)	2	2	2
(S3,R8)	1	5	3
(S4,R6)	6	1	4
(S5,R7)	1	1	1
(S6,R1)	1	3	2
(S7,R2)	1	4	3
(S8,R4)	3	2	3

Table 5.4 Request-view

Pairs (m,w)	Love degree	Contrast degree	Weight
(R1,S5)	1	3	2
(R2,S3)	3	3	3
(R3,S1)	1	7	4
(R4,S8)	1	2	2
(R5,S6)	1	3	2
(R6,S2)	1	7	4
(R7,S7)	1	2	2
(R8,S4)	2	5	4

Whereas, Service-view \cap Request-view = { (S2,R3) , (S3,R2) , (S5,R7) , (S6,R1) }.

The above results are used as threshold between the views of both parties. The union includes more pairs which are more relevant to the view of one party. However, the intersection is the view that is agreed by both parties.

5.3 Cloud Service Availability

As seen in the previous sections, there are several fields can benefit from our approach to achieve a considerable improvements. Our service matching approach reliably tackles the issue of service unavailability by redirecting the related request or requests to the most similar and available service or services when the current service is busy.

Applying SMP algorithm in services needs first building the preference lists of both services from matched cloud domains. This can be achieved using service profile to specify the most similar related participants (services in the other pool). Thus, each service from $p1$ needs to strictly order the services from $p2$ based on the similarity and vice versa. However, the original SMP algorithm performs a single assignment (one-to-one) for the involved candidates, which at some points does not help especially in the case of allocating more than one service to the right service in matched cloud domain for example. Dual propose algorithm has been proposed to fulfil this requirement which widely needed in such fields. Figure 5.4 below illustrates a brief depiction about our approach.

The result is matching of services from different domains or pools that take into account the preferences of both parties. This means that our approach is able to develop match making services that trying to optimise the pairings from both perspectives fairly.

This presented matching scheme covers a noticeable hole of the current state of the original SMP algorithm and perfectly assigned to the world of software engineering, serving in several aspects the needed allocation process that requires a high consideration of both sides of matched candidates. Also, our approach shows the ability to enhance the quality of allocating the similar services to each other, through considering the desire (similar

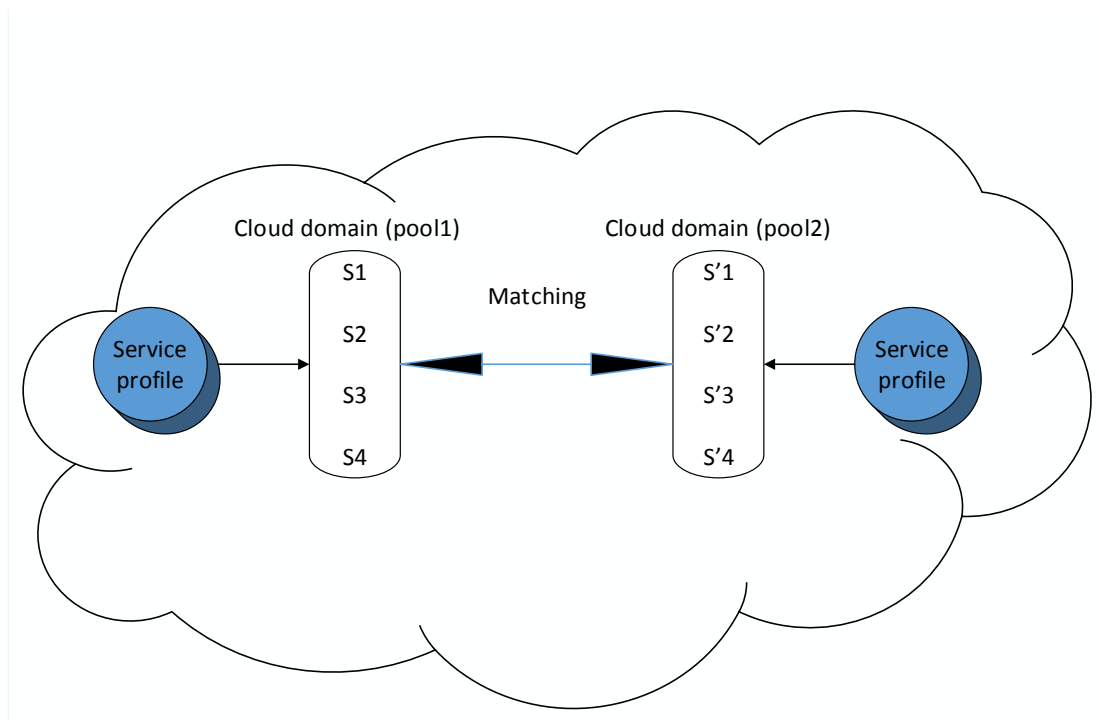


Fig. 5.4 Service Matching in Cloud.

or equivalence) of the matched services, which result in increased the satisfaction of the candidates in each pair, expecting a high stability. However, the Scheme should expect some limitation in terms of the time feature, as the consideration of the candidates' desire needs more computation and recursion to fulfil and reached the highly required stability. The implementation shows a competitive scheme which relatively performed in a large environment.

5.4 Discussion

Our contribution in this chapter is the selective strategy, that compromises the preferences of the service provider and the service request in a service oriented computing environment. Our scheme can increase the quality of allocating the services to the corresponding requests, through considering the desire of the matched candidates, which results in increased satisfaction of the candidates in each pair. This leads to more stable service bindings and has the potential to reduce inefficiencies that are incurred through re-binding and alternative lookups. However, as long as the proposed scheme uses the adaptation of Dual proposed, the scheme would require a polynomial time to complete the pairing process.

In our future work we would like to look at how dynamic constraints such as incurred by load and execution constraints on the providers would influence the stability of the matches. We also plan to evaluate our approach against strategies that consider only the interests of the service consumer.

5.5 Conclusion

This chapter introduces a new adaptation to the topic of service selection and discusses many relevant issues such as service availability. A middle sized prototype based on the defined adaptation is tested and shows that the adapted algorithm tackles such issues.

This section summarises the key points of this chapter as follows:

- A new adaptation of SMP Algorithm (Dual-Proposed) is applied for the process of Service Selection.

- To apply the new adaptation of the SMP (Dual-Proposed), a small prototype is tested.
- The service availability is discussed and how the new adapted algorithm can help to overcome this issue.

Chapter 6

Evaluation

Objectives

- To provide a medium-sized experiment to test the SMP-based approach to clone detection and to analyse the obtained results against a solid benchmark.
 - To provide an automated experiment to test the SMP-based approach to service selection.
-

6.1 Clone Detection

6.1.1 Introduction

There are several clone detection techniques and tools for different interest of purposes. Therefore, there are some properties or parameters which can be used to compare the existing techniques or approaches of clone detection. These properties are relatively considered as challenges of detecting software clones. In the following we mention some of these parameters:

- **Recall:** a good approach should be able to identify all relevant software clones of a system. Some of these clones may not be detected due to the reshape of the syntactic forms and therefore the similarity with the original, might not be recognised, although there is a relationship between the code fragments.
- **Precision:** the tool should detect software clone with a high precision among the retrieved clones, so less of false positive (discussed more in the next section).
- **Robustness:** the tool should be able to tackle different type of code clones and can find any hidden relationships between the software clones with higher recall and precision.
- **Scalability:** a good tool should be able to detect software clones in large system and handle complex systems efficiently (speed & memory).

The proposed approach tries (SMP-based) to maximise the range of considering the possible corresponding code clones. However, the final results show that based on the basic of the

original algorithm the time complexity may needs to be considered, which is quadratic time (polynomial time), as most of the metrics techniques' computational time is linear time.

Several related materials are considered in order of carrying out this experiment as follows:

1) Many software metrics tools to calculate the required values of code fragments such as Eclipse Metrics Plug-in 1.3.6

2) Also some pioneered metrics-based approaches as benchmarks to be compared against our approach in several facets; such as eMetrics proposed by Fabio et al [20, 79].

3) Moreover, different samples (snapshots of used source code) which have been already considered by other researchers (exists tools) to give a precise comparison.

6.1.2 Case Study (Job Search System)

6.1.2.1 Overview

The job search web application project provides services for both jobseekers and companies to seek their needs. It allows different organizations to upload different categories of job advertisements to the system to be viewed by appropriate applicants. The system has an automatic matching facility that matches the job specification with the applicant's preferences when the conditions are met. The job search engine will shows only the related jobs to what has been searched for. In this thesis, the job search web application has been utilized as a case study, in order to apply our approach detecting the possible similar code (code clones) within the whole project.

6.1.2.2 Related Technology

This section gives a brief overview of the technologies that have been used to implement this project. These technologies are as flow:

- **Java programming.** It is a professional object oriented programming language that was developed by Sun Microsystems. It is independent platform which means that it can be executed on many different operating systems [3]. In our case, it makes this case study more suitable to use it with our approach as many clone detection tools have used several java-based cases to evaluate their approaches.
- **Hibernate.** Hibernate is an ORM tool for Java that uses XML files as a replacement for the database tables in the design and implementation. However, it has been aimed to be used with the database in this project. Hibernate has been developed by Gavin King and a team of Java developers and is supported by Red Hat. Hibernate is used to perform operations on the database by mapping the instance variables in the classes with the columns in the database [3].
- **Spring Framework.** Spring Framework has many features, organised in seven modules.
- **Wicket Framework** Wicket is an open source component-oriented Java web application framework [3].
- **Aegis** which is also called Spring Security is a power full and flexible security solution that provides applications with comprehensive authentication and authorization.

- **PostgreSQL** is one of the most popular databases that have a strong reputation in terms of reliability and data integrity. It is easily capable to operate on top of many different operating systems.
- **JavaScript** is the scripting language of the Web that adds functionality, validates forms; detects browsers, etc. This project has already used this language to validate forms and apply some functionality.

6.1.2.3 System Design

The system consists of three major tiers namely presentation tier, business tier and data tier. Each of these tiers or levels performs part of the functional requirements of the system. They are all inter-connected to achieve the overall objective of the system as seen in following figure 6.1

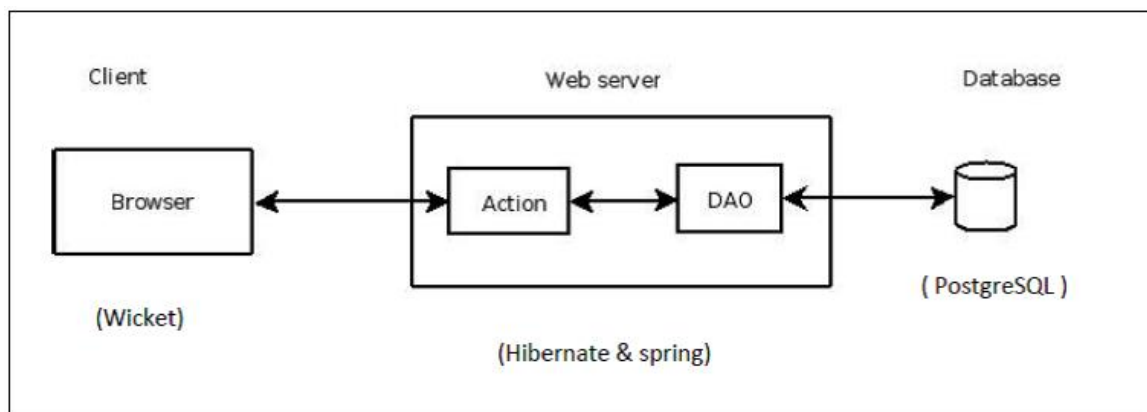


Fig. 6.1 Three-tier architecture.[3]

The following table 6.2 describes the layers of the three-tier architecture and shows how this web application has adopted each of them:

Layer	Description
Presentation layer	<p>Since this layer represents the user interface and reduces duplicating in the code and gain reusability, it is considered as the most important layer in the three- tier architecture.</p> <p>In this web application, Wicket presents this layer. It deals with user interaction by receiving the inputs and displaying the output.</p>
Business layer	<p>This tier acts as a medium between presentation tier and data tier. Also, it performs the requests of presentation tier in one hand and manages operations on data of Data tier on the other hand.</p> <p>In job search system java, Hibernate and Spring is the base of this tier.</p>
Data layer	<p>It is the platform where all required data are stored in.</p> <p>In this system, Hibernate and PostgreSQL are used to implement this layer.</p>

Fig. 6.2 Three-tier description of the project.

6.1.3 Clone Detection Experiment

An example (a Java code of a job search system) has taken place over a medium size of source files with around 460 methods as appears in the following tables. However, the way of calculating metrics is reflecting the priority of wanted aspects of each block. This specified

merged metrics are justified to accomplish the purpose of detecting as many as possible of the code clones, achieving high recall. Moreover, the precision of the retrieved code fragments needs to be considered, avoiding both false-negative and false-positive.

Table 6.1 Job search system

#Files	Size	#Methods	#LOC	Language
73	260kb	495	6085	Java

Table 6.1 shows the details of the job search system, which has been used to extract software clones using our approach (SMP-based). Table 6.2 shows some calculated metrics in the job search system.

Table 6.2 Snapshot of some metrics in the job search system

Method	Metrics				
	LOC	Nbp	Nbv	CC	NBD
AdminPage	118	0	2	3	3
ManageType	61	0	6	3	4
ChangePass	40	1	2	2	3
NewJob	49	1	11	2	3
JobSearch	122	2	6	5	4
UpdateAdmin	48	1	6	2	2

We have used the CloneDR tool to find the duplicated code in job search project. The CloneDR tool has got several parameters which influence the ultimate results of the software clone pairs, such as minimum mass (lines) and similarity threshold. Table 6.3 shows actual cloned software in both job search project and netbeans-javadoc.

Figure 6.3 below shows the detected clone candidates, actual clone candidates and three types of clone software in both job search project and netbeans-javadoc.

Table 6.3 Number of actual clone for program netbeans-javadoc and job search project.

	Cands	Actual cands	Type1	Type2	Type3
Jobsearch	41	38	19	16	3
netbeans-javadoc	82	70	27	38	5

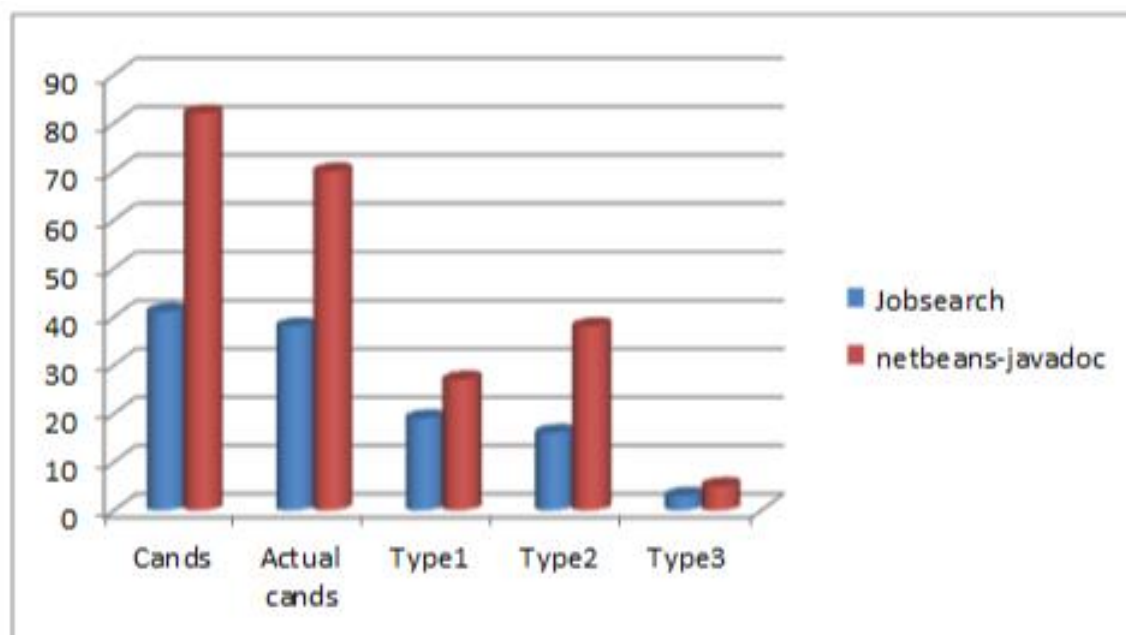


Fig. 6.3 Number of clone candidates for program netbeans-javadoc and job search project.

To avoid bias, all actual candidates are inspected, to make sure that the candidates are real software clones. The results therefore, are subjected to examination by author who has investigated software clone candidates which were detected by SMP-Based approach. Figure 6.4 displays the total detected code clones and their derived types based on the inspected clone candidates, comparing CloneDR to SMP-Based.

Table 6.4 Number of clone candidates for program netbeans-javadoc and job search project.

	SMP-approach				CloneDR			
	Type1	Type2	Type3	Total	Type1	Type2	Type3	Total
Jobsearch	17	11	13	41	19	16	3	38
netbeans-javadoc	23	29	16	68	27	38	5	70

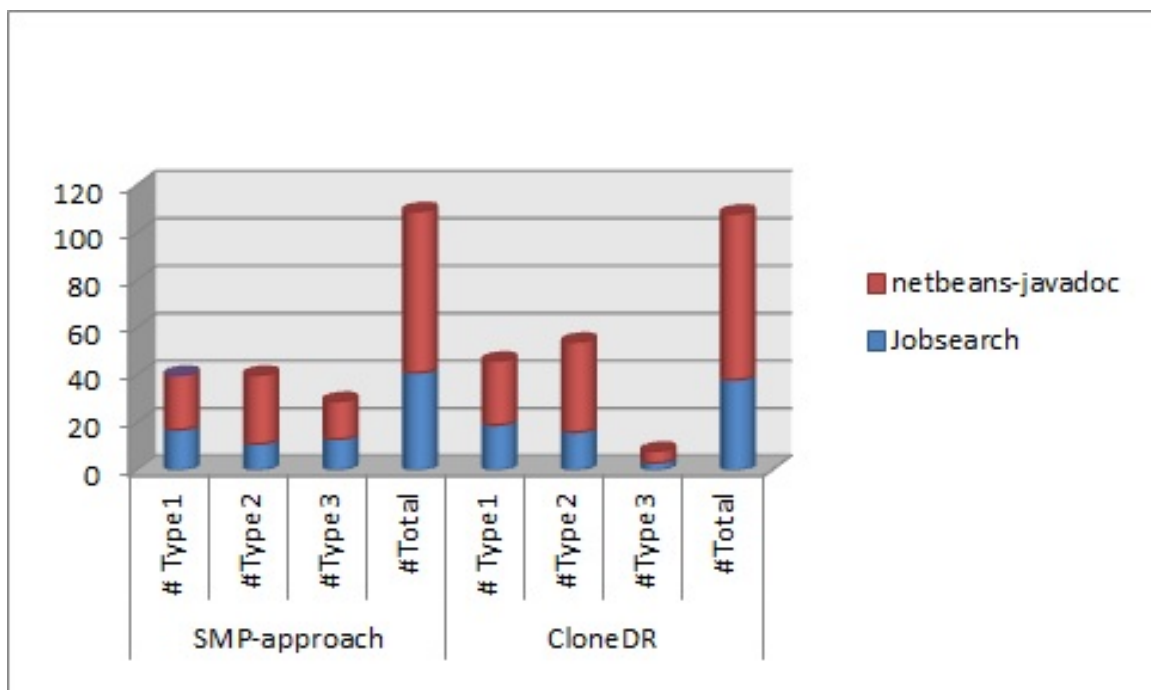


Fig. 6.4 Number of clone candidates for program netbeans-javadoc and job search project.

Also, we have considered the experiment conducted by Bellon et al. in their valuable article [16] and examine its findings to evaluate our approach. Figure 6.5 depicts several clone detection tools including our technique, finding software clones based on their types.

Table 6.5 Number of detected software clones for job search project.

	Type1	Type2	Type3	Total
Data set	40	20	10	70
SMP-approach	17	11	13	41
Baker	29	17	6	52
Kamiya	33	12	5	50
Rieger	30	13	6	49

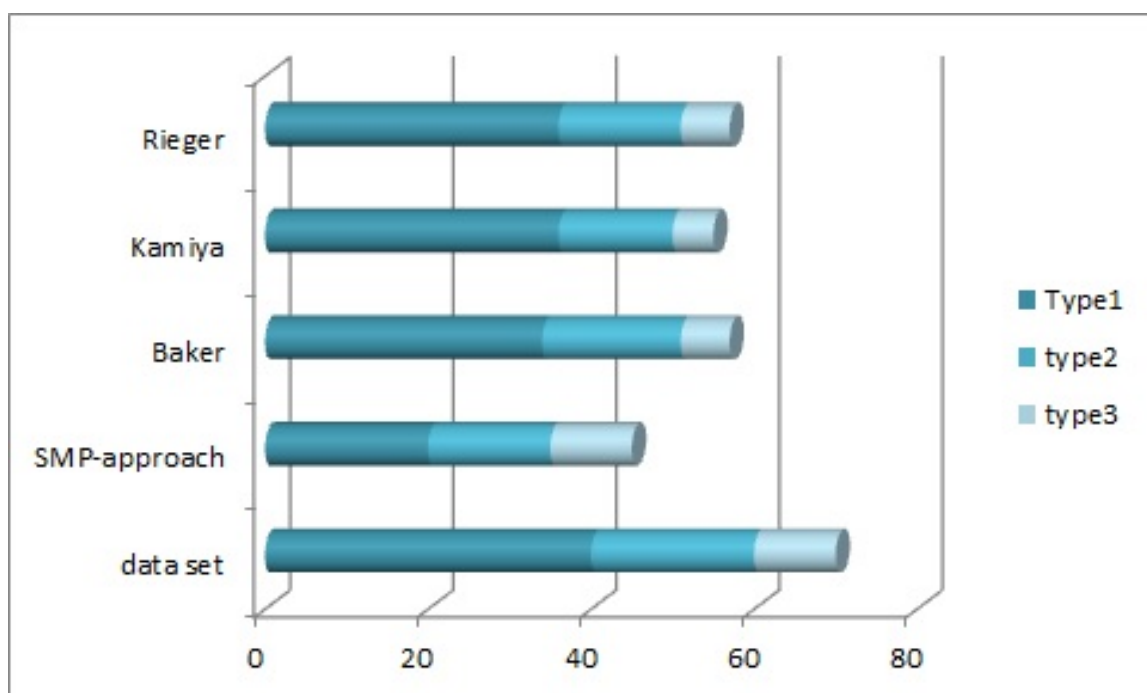


Fig. 6.5 Number of detected software clones for job search project.

Some examples of software clones types are listed in Appendix A (see figure A.7).

6.1.3.1 Recall and Precision

In this section we demonstrate two important measures that we use in some of the evaluation parts of this chapter. To measure the effectiveness of a retrieval method in information

retrieval, there are two important measures; recall and precision. Figure 6.6 below shows the calculation of these measures. These can be calculated if we suppose there exists set A of actual clones precisely inspected by human. However, it is very expensive to determine A using manual inspection, which means that the final results are subjective to the inspector.

The tools of clone detection will identify some set C of candidate clones. The ideal set C should be as close as possible to the actual clones of set A. However, the set of candidates D presents a real software clones. Precision measures the fraction of candidates that are actually clones, and recall measures the fraction of actual clones that are identified as candidates. Achieving both high recall and precision reflects an ideal approach.

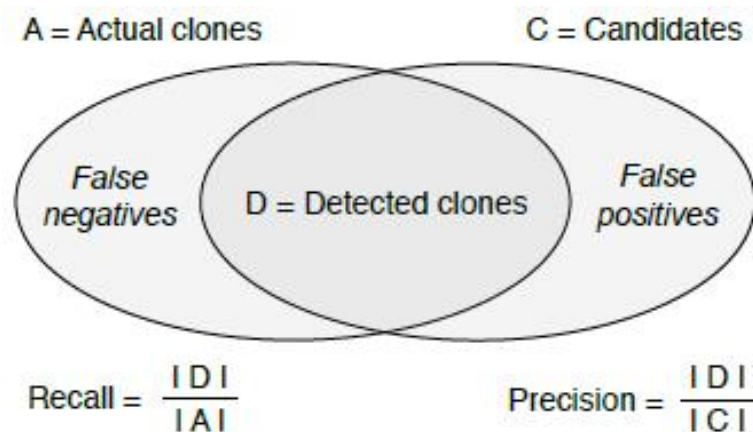


Fig. 6.6 Recall and precision.

The following are two definitions that used for calculating recall ad precision of our approach (SMP-based) and CloneDR.

Definition 1:

- Let $AllCands(P, T, t)$ denote the reported candidates (possible software clones) of a program P by conducted tools T with respect to clone type t .
- Let $Cands(P, T, t)$ denote the reported candidates (possible software clones) of program P by tool T with respect to clone type t .
- Let $InspectedCands(P, T, t)$ denote all candidates (possible software clones) of tool T for program P and clone type t that examined by an independent inspector.
- Let $ActuCands(P, T, t)$ denote the reported actual clone candidates of program P by tool T with respect to clone type t by the inspector.
- Let $IntersectCands(P, T, t)$ denote all candidates (possible software clones) of conducted tools of a program P with respect to clone type t that are in common.

Definition 2:

$$Recall = \frac{ActuCands(P, T, t)}{AllCands(P, T, t)},$$

$$Precision = \frac{ActuCands(P, T, t)}{Cands(P, T, t)}$$

The following figure 6.7 shows the recall of our approach (SMP-based) against CloneDR based on different bases, as defined in Definition 1 and 2.

The following figure 6.8 shows the precision of our approach (SMP-based) against CloneDR based on different bases, as defined in Definition 1 and 2. We have noticed that our approach can detect more software clones of type 3 compared to other tools, which indicate a deep understanding of the code geologies, whereas the most recognised code clones are type 1.

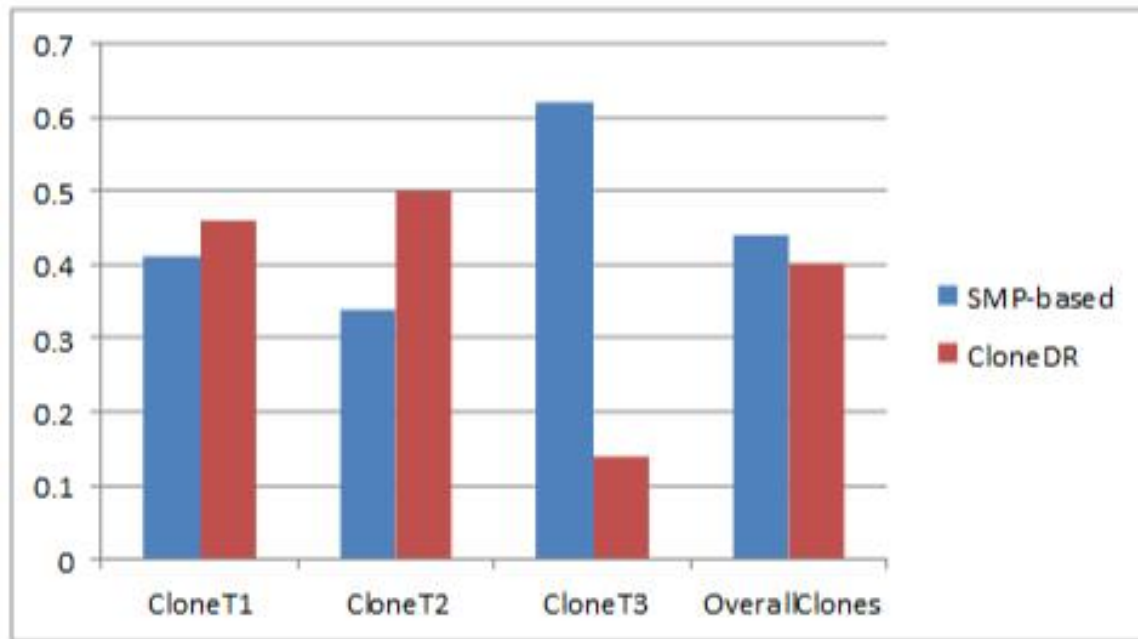


Fig. 6.7 Recall; see Definition 2.

The difference of the results refers to the considered metrics in aforementioned approaches, as SMP-based has used some metrics which present the nature of the code methods rather than focusing on the syntactic facet of the software clone.

6.1.3.2 Discussion

The experiment shows that the SMP-based approach can precisely identify a type-3 clone which is a copy with further modifications more than syntactic changes. The main features of SMP-based approach are competitively outweigh some already existing approaches in several facets such as increasing the spot of the detection process to try to detect every possible clone smell. However, this approach lacks two related features respectively time complexity and speed as the used algorithm is executed in a quadratic time, which impacts the overall

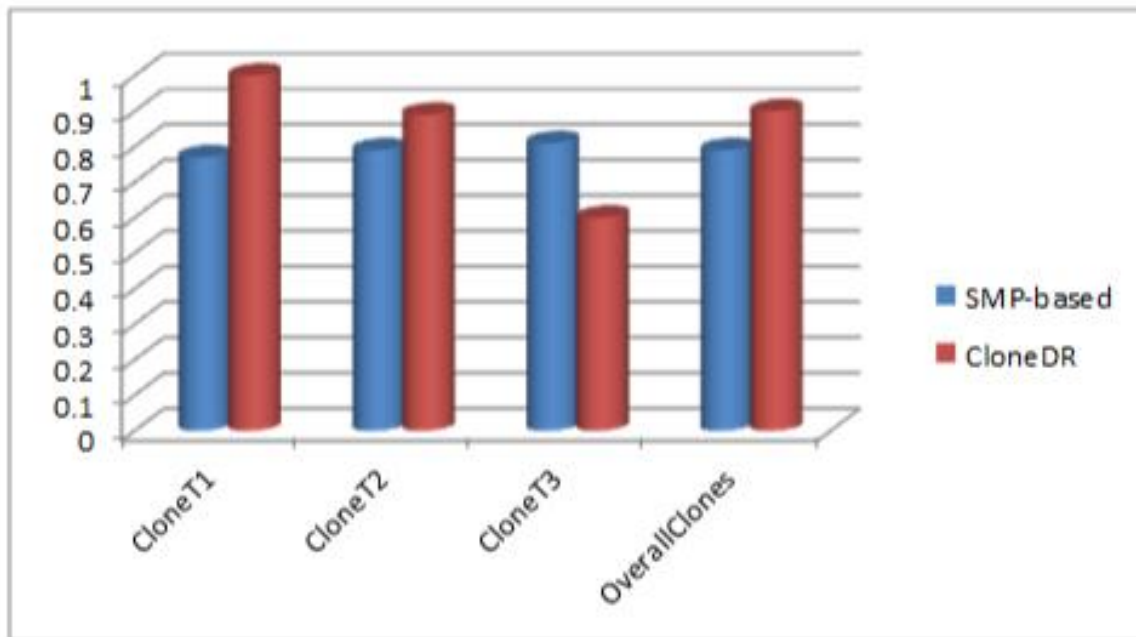


Fig. 6.8 Precision; see Definition 2.

scalability. The clone granularity has been set as method-based blocks in which every method acts as a candidate (possible software clone).

A remarkable efficiency of the proposed approach have been evaluated by carrying out a case study on two medium size source files, each file has more than 100 specified blocks. Also, a set of metrics are predefined to help each candidate to build up its own preference list in order to apply the SMP algorithm. We observed 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. Similarly, allowing the many-to-many relationship to increase the

range of clones (high recall, high precision) that are undetectable with most of the previous clone detection approaches. The time complexity is the same as the original SMP algorithm (polynomial time).

6.2 Service Selection

6.2.1 Case Study

To apply the SMP algorithm in service selection, it needs first to build the preference lists of both requests and services. This can be achieved using service profile to specify the most similar related participants (services or requests). Each request needs to strictly order the service based on the similarity and vice versa (similar to Figure 5.3). 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 service requests to the right service for example. Dual-proposed algorithm has been proposed to fulfil this requirement.

6.2.1.1 A Hotel Reservation Example

Figure 6.9 is a general example of web service composition scenario.

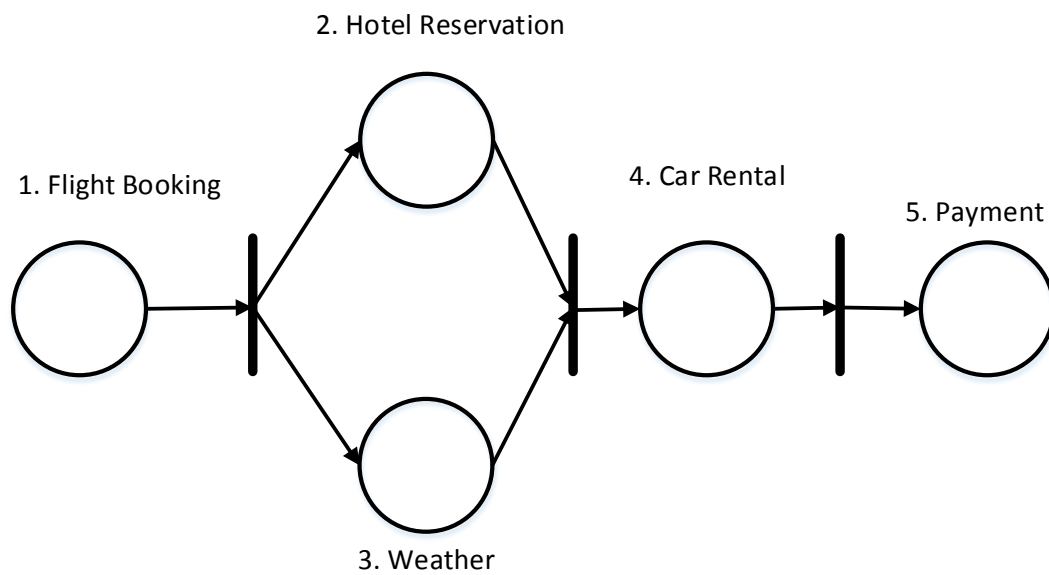


Fig. 6.9 General Example of web services.

Figure 6.10 shows an engaged service, hotel reservation, to the most suitable requests using SMP algorithm based on the QoS and the values of both candidates. There are different hotel services to show the rooms available, the style of the rooms etc. There are also some hotel reservation requests from the customers or agents. Then the extended dual-proposed algorithm can be applied to choose the best services.

The result is matching of service providers (hotels) and consumers (guests) that do not only take into account the preferences of the respective consumers, but also those of the providers. This means that we are now able to develop match making services that not only decide on the basis of the request's preferences, but are actually trying to, within the current set of services requests and services, optimise the pairings from both perspectives fairly.

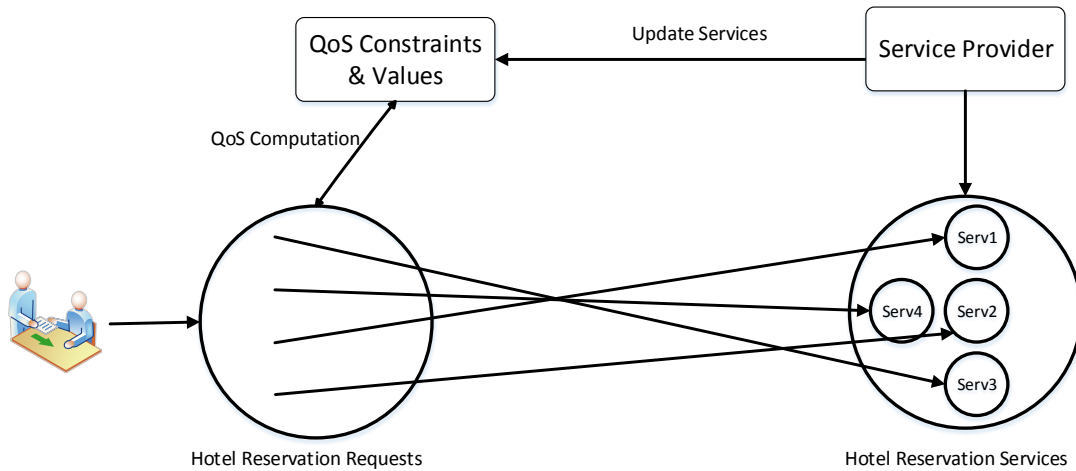


Fig. 6.10 SMP-based service selection.

6.2.1.2 Discussion

The network administrator should act as the stakeholder and take advantage of monitoring the process of matching services to requests. This provides reliable information such as profile and history details of both services and requests. The result is a service interaction in which optimal assignments of these two parties ensure safety, reliability, dependability and stability to their candidates.

Based on the characteristics of both services and requirements of customers' requests, our approach can start triggering the matching process to allocate the most proper customers' requests to the most appropriate services and considers both views using our smart selective strategy.

Our approach contributes to the service selection by adding the matching behaviour rather than the traditional existing search-based techniques which picks up the required services from the UDDI directory regardless of the constraints of the chosen services.

Web services should be dealt with as independent objects that have their own view about incoming consumers' request and are aware of its requirements. Therefore, SMP-based approach gives services rights to whether accept or decline the customers' requests, with regards to their conditions. This can corroborate the concept of equal opportunity which in turn ensures the stability.

Moreover, this approach helps services to check the trust level of the client, which supports both safety and reliability. On demand, the service may choose to reject the client because he demanding highly computational task that may impact other ongoing tasks in the service to meet a quality of service that it promised to other clients.

The following Diagrams 6.11 and 6.12 show some considered facets of both consumer's request and web service respectively, in a simple prototype example of hotel reservation.

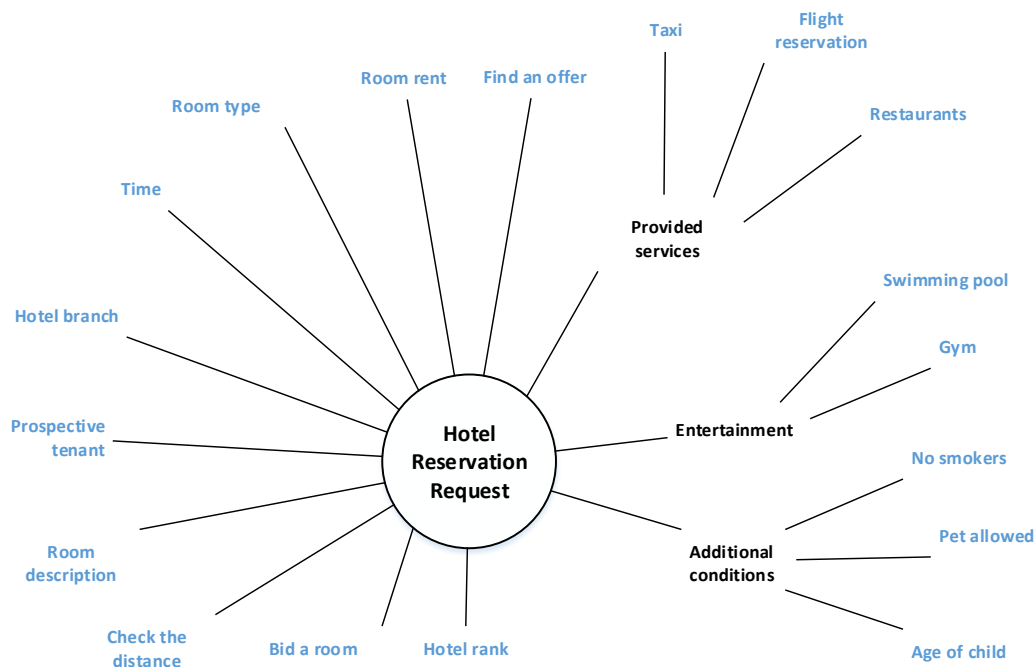


Fig. 6.11 Hotel Reservation Request.

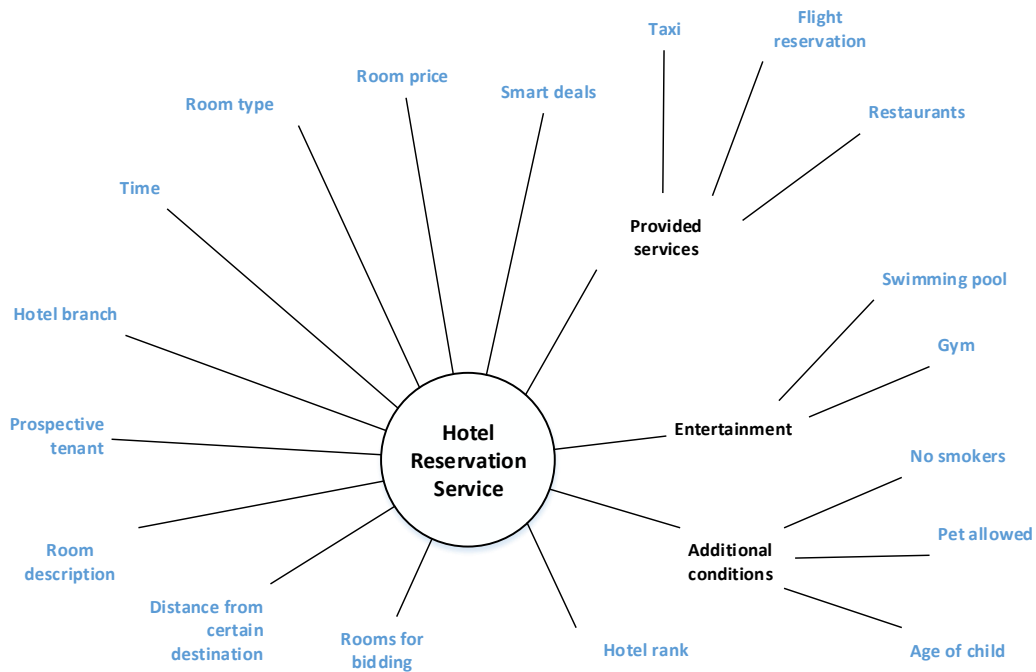


Fig. 6.12 Hotel Reservation Service.

The following Figure 6.13 depicts an early pre-step of matching consumer's requests to the services of hotel reservation using SMP-Based approach.

6.2.2 Experiment for Service Selection

In this part we apply dual-proposed algorithm for selection process of the services to the relevant requests. We assume a many to many relationship in which the service can serve more than one request at a time, also the request is served by more than one service to fulfil its requirements. The following screen-shots 6.14, 6.15 and 6.16 present a prototype of java program that designed to show the execution of both the original SMP algorithm beside the newly adapted dual-proposed algorithm.

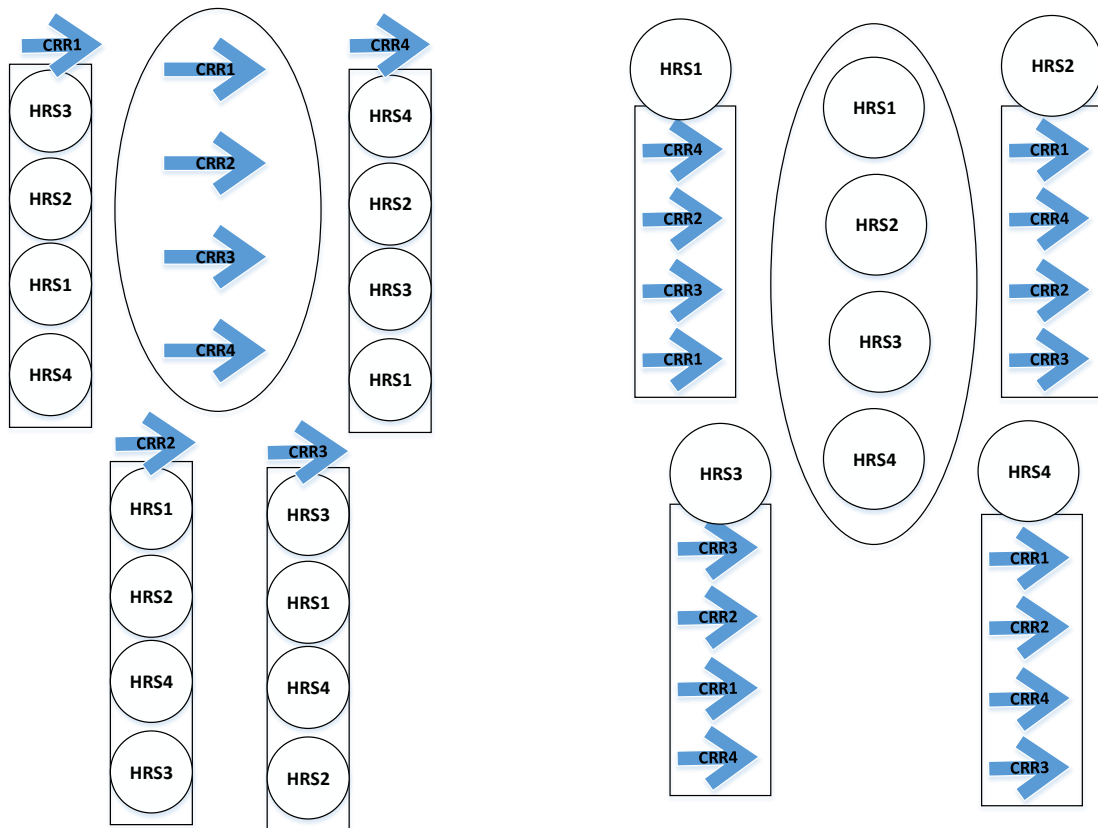


Fig. 6.13 Hotel Reservation Example.

The preference lists are randomly generated for each candidate and for each service. Then we choose to run either of the two algorithms. The results appear in the right part of the screen along with the execution time.

Figure 6.14 shows the service-optimal based on the original SMP algorithm. This gives the priority to the party of services whereas in Figure 6.15 the best choices are selected by the requests. This not satisfies the service selection process as long as the relation based on this main SMP algorithm is still one-to-one.

However, results in Figure 6.16 reflects clearly the relation of many-to-many between the services and the requests as this meet the purpose of the two direction approach that

The screenshot displays the 'Stable Marriage Problem Simulation' interface. On the left, there are two text areas: 'R1' through 'R8' listing preferences for requests, and 'Services Preferences' listing preferences for services. Below these are buttons for 'Generate Preferences', 'Run Regular SMP', and 'Run Extended SMP'. At the bottom left, there are checkboxes for 'Requests Optimal' and 'Services Optimal' (which is checked), and a 'Clear' button. The right side of the interface shows the calculated results for a 'Service Optimal' match, including LD, CONT, and WGT values for each request-service pair, and the final stable matches.

```

R1 S2 S6 S4 S5 S8 S7 S3 S1
R2 S8 S3 S2 S4 S6 S7 S1 S5
R3 S4 S1 S7 S5 S2 S3 S8 S6
R4 S4 S3 S1 S2 S7 S5 S8 S6
R5 S4 S5 S3 S2 S7 S6 S8 S1
R6 S3 S2 S4 S5 S8 S7 S6 S1
R7 S7 S6 S3 S1 S2 S5 S4 S8
R8 S8 S2 S7 S5 S3 S6 S4 S1

Services Preferences:
S1 R8 R4 R1 R6 R3 R2 R7 R5
S2 R4 R2 R7 R5 R3 R8 R6 R1
S3 R5 R2 R7 R4 R8 R3 R1 R6
S4 R8 R7 R4 R3 R6 R1 R5 R2
S5 R6 R7 R8 R3 R2 R1 R5 R4
S6 R6 R1 R5 R7 R2 R4 R8 R3
S7 R7 R2 R4 R6 R1 R8 R5 R3
S8 R6 R1 R8 R5 R4 R7 R2 R3

R6 => S1 LD= 8 CONT = 4 WGT = 6
R7 => S7 LD= 1 CONT = 0 WGT = 0
R8 => S8 LD= 1 CONT = 2 WGT = 1

Stable Matches (Services + Requests):
Service Optimal

S1 => R6 LD= 4 CONT 4 WGT = 4
S2 => R2 LD= 2 CONT 1 WGT = 1
S3 => R1 LD= 7 CONT 0 WGT = 3
S4 => R4 LD= 3 CONT 2 WGT = 2
S5 => R3 LD= 4 CONT 0 WGT = 2
S6 => R5 LD= 3 CONT 3 WGT = 3
S7 => R7 LD= 1 CONT 0 WGT = 0
S8 => R8 LD= 3 CONT 2 WGT = 2

This is the result of using the calculated weight and
contrast. From point of view of requests:

R1 => S3
R2 => S2 R3 => S6 R4 => S4
R5 => S6 R6 => S1 R7 => S7
R8 => S8

Result:
Stable Matches (Services + Requests):
Service Optimal

S1 => R6
S2 => R2 S3 => R1 S4 => R4
S5 => R3 S6 => R5 S7 => R7
S8 => R8

```

Fig. 6.14 Execution of the main SMP Service Optimal.

consider the view of the services party. It can be seen that the selective strategy is applied and the relevant factors (love degree, contrast degree and weight) are calculated in the right area of the output Figure 6.16.

The final results shows that our algorithm accurately performs the selection process for the wanted purpose. The final pairs are accordingly stable and achieve the satisfaction, with regard to their preferences. However, the computational time still in concern, the execution time of both dual-proposed and the main SMP algorithm is almost the same, running in a quadratic time (polynomial time).

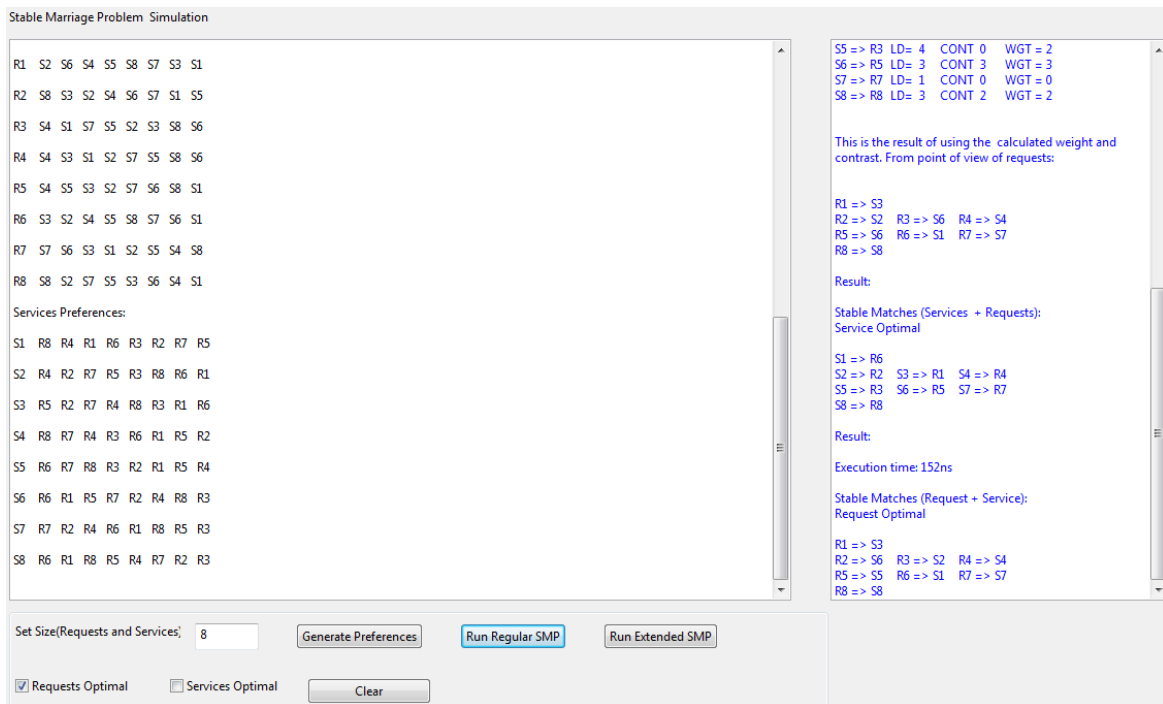


Fig. 6.15 Execution of the main SMP Request Optimal.

6.3 Conclusion

This chapter discusses two different experiments for two issues, to support the proposed methods. This gives a chance to compare the introduced method to the previous approaches for both clone detection and service selection. The obtained results show that the proposed approach outperforms the current approaches in overall performance.

This section summarises the key points of this chapter as follows:

- Two experiments for both clone detection and service selection are presented.
- Compare the SMP-based approaches to the existing approaches.
- The obtained results are analysed to evaluate some properties such as accuracy and overall performance.

Stable Marriage Problem Simulation

```

R1 S2 S6 S4 S5 S8 S7 S3 S1
R2 S8 S3 S2 S4 S6 S7 S1 S5
R3 S4 S1 S7 S5 S2 S3 S8 S6
R4 S4 S3 S1 S2 S7 S5 S8 S6
R5 S4 S5 S3 S2 S7 S6 S8 S1
R6 S3 S2 S4 S5 S8 S7 S6 S1
R7 S7 S6 S3 S1 S2 S5 S4 S8
R8 S8 S2 S7 S5 S3 S6 S4 S1

Services Preferences:
S1 R8 R4 R1 R6 R3 R2 R7 R5
S2 R4 R2 R7 R5 R3 R8 R6 R1
S3 R5 R2 R7 R4 R8 R3 R1 R6
S4 R8 R7 R4 R3 R6 R1 R5 R2
S5 R6 R7 R8 R3 R2 R1 R5 R4
S6 R6 R1 R5 R7 R2 R4 R8 R3
S7 R7 R2 R4 R6 R1 R8 R5 R3
S8 R6 R1 R8 R5 R4 R7 R2 R3
    
```

Execution time: 2330ns

Stable Matches (Request + Service):
Request Optimal

```

R1 => S3 LD= 7  CONT = 0  WGT = 3
R2 => S6 LD= 5  CONT = 0  WGT = 2
R3 => S2 LD= 5  CONT = 0  WGT = 2
R4 => S4 LD= 1  CONT = 2  WGT = 1
R5 => S5 LD= 2  CONT = 5  WGT = 3
R6 => S1 LD= 8  CONT = 4  WGT = 6
R7 => S7 LD= 1  CONT = 0  WGT = 0
R8 => S8 LD= 1  CONT = 2  WGT = 1
    
```

Stable Matches (Services + Requests):
Service Optimal

```

S1 => R6 LD= 4  CONT 4  WGT = 4
S2 => R2 LD= 2  CONT 1  WGT = 1
S3 => R1 LD= 7  CONT 0  WGT = 3
S4 => R4 LD= 3  CONT 2  WGT = 2
S5 => R3 LD= 3  CONT 0  WGT = 2
S6 => R5 LD= 3  CONT 3  WGT = 3
S7 => R7 LD= 1  CONT 0  WGT = 0
S8 => R8 LD= 3  CONT 2  WGT = 2
    
```

This is the result of using the calculated weight and contrast. From point of view of requests:

```

R1 => S3
R2 => S2  R3 => S6  R4 => S4
R5 => S6  R6 => S1  R7 => S7
R8 => S8
    
```

Set Size(Requests and Services): 8

Generate Preferences Run Regular SMP Run Extended SMP

Requests Optimal Services Optimal Clear

Fig. 6.16 Execution of the dual-proposed SMP algorithm.

Chapter 7

Conclusion and Future Work

Objectives

- To summarise the whole thesis and provide some recommendations.
 - To show the limitations of our approach.
 - To provide some possible future works.
-

7.1 Summary of the Thesis

This chapter concludes the thesis as detailed in the former chapters. Comparing and reviewing the work to the existing relevant approaches in the literature. The thesis demonstrates that the developed selective strategy utilises the SMP algorithm effectively for enhancing the task of both service selection and clone detection. The proposed modifications to the SMP algorithm further enhance the efficacy of the algorithm, improving the cardinality to achieve many-to-many relationship and therefore, improving both the detection process of the code clones and matching process of service selection. Similar software applications could also benefit from the newly adapted algorithm. The capabilities and potential of the approach are experimentally validated, and compared with either the relevant techniques, or original approaches in the literature. Ultimately, the chapter states a number of directions for future works. The thesis aggregates and relates together a number of relatively big topics, starting with utilising stable marriage problems in software engineering.

7.2 Contributions Revisited

The principle contributions of this thesis are as follows:

- First, we extend the original SMP algorithm for purpose of software context, to ensure a proper correlation process between the involved candidates. The two extensions respectively named as *Dual proposed* and *Dual Multi Allocation*. The first extension is based on the original SMP algorithm, whereas the second adaptation is based on the hospital resident algorithm (HR) originally derived from the main SMP algo-

rithm. These extensions accommodate the main idea of the SMP in order to improve and enhance the current state of software clone detection and the process of service selection.

- Second, we design a semi-automated experiment on several real programs (netbean-javadoc), and also contribute to the results of Bellon's study. As the final results are subjects to an independent expert programmer who manually checks the detected clone candidates, he notices that most of suspected software clone candidates are real cloned software. He states that SMP-based approach shows very competitive results compared to some existing software clone approaches, especially in identifying type 3 (a copy with further modifications such update, add and delete statements) of cloned software and he regards these results very encouraging. Also, the SMP-based technique acts very well with regards to the accuracy, showing a high percentage of recall of 44% which is just above the percentage achieved by CloneDR, and accomplishes a good percentage of precision of 79% compared to some previous approaches. The main drawback of the approach is that it requires a polynomial time $O(n^3)$ to run the algorithm, however, as the compare granularity fixed to the method-based level, the number of candidates are much lower than some other approaches which rely on very small blocks for instance line-based or token-based. This alleviates the problem of time complexity and puts the SMP-based approach in acceptable range of other approaches.
- Finally, we utilise the SMP-based algorithm to the process of service selection where a basic search-based algorithm is presented. The SMP-based ensures matching behaviour

to this process, and deals with each service as an independent object which has the right to whether accept or decline a certain request based on their constraints. The initial results shows that the SMP-based approach preserves the service quality, as the busy service will either redirect or simply reject the coming request. Moreover, the SMP-based helps services to protect themselves against malicious requests by checking the consumer's profile, and therefore, ensures both safety and reliability.

7.3 Success Criteria Revisited

We emphasize the criteria of success of our approach as follows:

- The adaptation of the main SMP algorithm (SMP-based) is successfully extended to process the relationship of many-to-many, which helps in software engineering context, and is applied to matching process of both clone detection and service selection, which performs well in correlating similar candidates.
- SMP-based approach outperforms the current clone detection, especially in recognising type 3 of software clones, and shows high precision and high recall.
- Service selection process clearly benefits from the SMP-based approach by ensuring an intelligent matching interaction between services and requests.
- All aforementioned steps are experimentally tested and showed competitive results.

7.4 Thesis Limitations

In this section, we show the limitations of this thesis as follows:

- Our SMP-based approach less efficient, because of its nature as the original SMP algorithm is performed in a polynomial time $O(n^3)$.
- Our SMP-based approach is limited to detect software clones of type 1,2 and 3, and faces difficulties to recognise most of semantically equivalence code fragments (type 4).
- In service selection, we assumed that the services are completely independent which requires a third party to ensure the workflow of the matching process between the candidates of both parties. We regard this as a new paradigm that has not yet been tested in reality.

7.5 Future Work

We make several recommendations in relation to future research:

- To improve the SMP-based approach to automatically and dynamically compare different levels of software clone granularities, as its current status is fixed to method-based only.
- To improve SMP-based approach by consider some semantic techniques such as ontology alignments which positively affects the matching process among the textual part of paired elements. This will ensure a perfect service selection.

- To extend the SMP-based approach to detect type 4 of cloned software, which performs with same computations but are implemented by different syntactic variables.
- To resolve the issue of service availability by correlating the similar cloud services from different pools of services in order to provide similar services

References

- [1] Agrawal, R. and Srikant, R. (1995). Mining sequential patterns. In *Data Engineering, 1995. Proceedings of the Eleventh International Conference on*, pages 3–14. IEEE.
- [2] Alghamdi, J. S., Rufai, R. A., and Khan, S. M. (2005). Oometer: A software quality assurance tool. In *2011 15th European Conference on Software Maintenance and Reengineering*, pages 190–191. IEEE Computer Society.
- [3] AlHakami, H. (2009). Job search web application. Master’s thesis, The University of Birmingham.
- [4] AlHakami, H., Aldabbas, H., and Alwada’n, T. (2012). Comparison between cloud and grid computing: review paper. *International Journal on Cloud Computing: Services and Architecture*, 2(4):1–21.
- [5] AlHakami, H., Chen, F., and Janicke, H. (2014). An extended stable marriage problem algorithm for clone detection. *International Journal of Software Engineering & Applications (IJSEA)*, 5:103–122.
- [6] Alrifai, M. and Risse, T. (2009). Combining global optimization with local selection for efficient qos-aware service composition. In *Proceedings of the 18th international conference on World wide web*, pages 881–890. ACM.
- [7] Armbrust, M., Fox, A., Griffith, R., Joseph, A. D., Katz, R., Konwinski, A., Lee, G., Patterson, D., Rabkin, A., Stoica, I., et al. (2010). A view of cloud computing. *Communications of the ACM*, 53(4):50–58.
- [8] Baker, B. S. (1993). A program for identifying duplicated code. *Computing Science and Statistics*, pages 49–49.
- [9] Baker, B. S. (1995). On finding duplication and near-duplication in large software systems. In *Proceedings of 2nd Working Conference on Reverse Engineering*, pages 86–95. IEEE.
- [10] BALANCED, S. M. T. (2013). Web service description framework and selection mechanism towards balanced perceptions of fqos and qos. *Journal of Theoretical and Applied Information Technology*, 47(1).

- [11] Balazinska, M., Merlo, E., Dagenais, M., Lague, B., and Kontogiannis, K. (1999). Measuring clone based reengineering opportunities. In *Software Metrics Symposium, 1999. Proceedings. Sixth International*, pages 292–303. IEEE.
- [12] Baldoni, M., Baroglio, C., Martelli, A., Patti, V., and Schifanella, C. (2008). Service selection by choreography-driven matching. In *Emerging Web Services Technology, Volume II*, pages 5–22. Springer.
- [13] Basit, H. and Jarzabek, S. (2009). A data mining approach for detecting higher-level clones in software. *Software Engineering, IEEE Transactions on*, 35(4):497–514.
- [14] Basit, H. A. and Jarzabek, S. (2007). Efficient token based clone detection with flexible tokenization. In *Proceedings of the the 6th joint meeting of the European software engineering conference and the ACM SIGSOFT symposium on The foundations of software engineering*, pages 513–516. ACM.
- [15] Baxter, I., Yahin, A., Moura, L., Sant’Anna, M., and Bier, L. (1998). Clone detection using abstract syntax trees. In *Software Maintenance, 1998. Proceedings. International Conference on*, pages 368–377. IEEE.
- [16] Bellon, S., Koschke, R., Antoniol, G., Krinke, J., and Merlo, E. (2007). Comparison and evaluation of clone detection tools. *Software Engineering, IEEE Transactions on*, 33(9):577–591.
- [17] Bergh, D. and Ketchen, D. (2009). *Research Methodology in Strategy and Management*. Number v. 5 in Research methodology in strategy and management. Emerald Group Publishing Limited.
- [18] Biró, P. and Mcdermid, E. (2014). Matching with sizes (or scheduling with processing set restrictions). *Discrete Applied Mathematics*, 164:61–67.
- [19] Burd, E. and Bailey, J. (2002). Evaluating clone detection tools for use during preventative maintenance. In *Source Code Analysis and Manipulation, 2002. Proceedings. Second IEEE International Workshop on*, pages 36–43. IEEE.
- [20] Calefato, F., Lanubile, F., and Mallardo, T. (2004). Function clone detection in web applications: A semiautomated approach. *Journal of Web Engineering*, 3:3–21.
- [21] Chen, W.-K., Li, B., and Gupta, R. (2003). Code compaction of matching single-entry multiple-exit regions. In *Static Analysis*, pages 401–417. Springer.
- [22] Cheng, C., McDermid, E., and Suzuki, I. (2008). A unified approach to finding good stable matchings in the hospitals/residents setting. *Theoretical Computer Science*, 400(1):84–99.
- [23] Chidamber, S. R. and Kemerer, C. F. (1994). A metrics suite for object oriented design. *Software Engineering, IEEE Transactions on*, 20(6):476–493.

- [24] Cross, V. and Hu, X. (2011). Using semantic similarity in ontology alignment. *Ontology Matching*, page 61.
- [25] Cuomo, A., Santone, A., and Villano, U. (2012). A novel approach based on formal methods for clone detection. In *Proceedings of the 6th International Workshop on Software Clones*, pages 8–14. IEEE Press.
- [26] Davey, N., Barson, P., Field, S., Frank, R., and Tansley, D. (1995). The development of a software clone detector. *International Journal of Applied Software Technology*, 1(3/4):219–236.
- [27] Dodig-Crnkovic, G. (2010). Constructivist research and info-computational knowledge generation. In Magnani, L.; Carnielli, W. P. C., editor, *Model-Based Reasoning In Science And Technology - Abduction, Logic, and Computational Discovery (Studies in Computational Intelligence)*, pages 359–380. Springer Verlag.
- [28] Ducasse, S., Nierstrasz, O., and Rieger, M. (2006). On the effectiveness of clone detection by string matching. *Journal of Software Maintenance and Evolution: Research and Practice*, 18(1):37–58.
- [29] Euzenat, J. and Shvaiko, P. (2007). *Ontology matching*, volume 18. Springer Berlin.
- [30] Evans, W. S., Fraser, C. W., and Ma, F. (2009). Clone detection via structural abstraction. *Software Quality Journal*, 17(4):309–330.
- [31] Falke, R., Frenzel, P., and Koschke, R. (2008). Empirical evaluation of clone detection using syntax suffix trees. *Empirical Software Engineering*, 13(6):601–643.
- [32] Feder, T. (1994). Network flow and 2-satisfiability. *Algorithmica*, 11(3):291–319.
- [33] Feder, T., Megiddo, N., and Plotkin, S. A. (1994). A sublinear parallel algorithm for stable matching. In *Proceedings of the fifth annual ACM-SIAM symposium on Discrete algorithms*, pages 632–637. Society for Industrial and Applied Mathematics.
- [34] Fleiner, T., Irving, R. W., and Manlove, D. F. (2011). An algorithm for a super-stable roommates problem. *Theoretical Computer Science*, 412(50):7059–7065.
- [35] Gale, D. and Shapley, L. (2013). College admissions and the stability of marriage. *The American Mathematical Monthly*, 120(5):386–391.
- [36] Gale, D. and Sotomayor, M. (1985). Some remarks on the stable matching problem. *Discrete Applied Mathematics*, 11(3):223–232.
- [37] Gallagher, K. and Layman, L. (2003). Are decomposition slices clones? In *Program Comprehension, 2003. 11th IEEE International Workshop on*, pages 251–256. IEEE.
- [38] Gitchell, D. and Tran, N. (1999). Sim: a utility for detecting similarity in computer programs. In *ACM SIGCSE Bulletin*, volume 31, pages 266–270. ACM.

- [39] Goddard, W. D. and Melville, S. (2004). *Research methodology: An introduction*. jutaonline. co. za.
- [40] Guo, G., Yu, F., Chen, Z., and Xie, D. (2011). A method for semantic web service selection based on qos ontology. *Journal of Computers*, 6(2):377–386.
- [41] Gusfield, D. (1987). Three fast algorithms for four problems in stable marriage. *SIAM Journal on Computing*, 16(1):111–128.
- [42] Gusfield, D. (1988). The structure of the stable roommate problem: efficient representation and enumeration of all stable assignments. *SIAM Journal on Computing*, 17(4):742–769.
- [43] Gusfield, D. and Irving, R. W. (1989). *The stable marriage problem: structure and algorithms*.
- [44] Harman, M. (2007a). The current state and future of search based software engineering. In *2007 Future of Software Engineering*, pages 342–357. IEEE Computer Society.
- [45] Harman, M. (2007b). Search based software engineering for program comprehension. In *Program Comprehension, 2007. ICPC'07. 15th IEEE International Conference on*, pages 3–13. IEEE.
- [46] Harman, M., Burke, E., Clark, J., and Yao, X. (2012a). Dynamic adaptive search based software engineering. In *Proceedings of the ACM-IEEE international symposium on Empirical software engineering and measurement*, pages 1–8. ACM.
- [47] Harman, M. and Clark, J. (2004). Metrics are fitness functions too. In *Software Metrics, 2004. Proceedings. 10th International Symposium on*, pages 58–69. IEEE.
- [48] Harman, M. and Jones, B. F. (2001). Search-based software engineering. *Information and Software Technology*, 43(14):833–839.
- [49] Harman, M., McMinn, P., de Souza, J. T., and Yoo, S. (2012b). Search based software engineering: Techniques, taxonomy, tutorial. In *Empirical Software Engineering and Verification*, pages 1–59. Springer.
- [50] Holland, J. H. (1975). *Adaptation in natural and artificial systems*. University of Michigan Press.
- [51] Huang, A. F., Lan, C.-W., and Yang, S. J. (2009). An optimal qos-based web service selection scheme. *Information Sciences*, 179(19):3309–3322.
- [52] Humphrey, W. S. (1997). *Introduction to the personal software process*. Addison-Wesley Professional.
- [53] Irving, R., Leather, P., and Gusfield, D. (1987). An efficient algorithm for the "optimal" stable marriage. *Journal of the ACM (JACM)*, 34(3):532–543.

- [54] Irving, R. W. (1985). An efficient algorithm for the "stable roommates" problem. *Journal of Algorithms*, 6(4):577–595.
- [55] Iwama, K. and Miyazaki, S. (2008). A survey of the stable marriage problem and its variants. In *Informatics Education and Research for Knowledge-Circulating Society, 2008. ICKS 2008. International Conference on*, pages 131–136. IEEE.
- [56] Jiang, L., Mishherghi, G., Su, Z., and Glondu, S. (2007). Deckard: Scalable and accurate tree-based detection of code clones. In *Proceedings of the 29th international conference on Software Engineering*, pages 96–105. IEEE Computer Society.
- [57] Jiang, M., Ding, Z., and Liu, J. (2012). Service selection based on behavior matching. *Journal of Software*, 7(9):1950–1959.
- [58] Jiang, Z., Hassan, A., and Holt, R. (2006). Visualizing clone cohesion and coupling. In *Software Engineering Conference, 2006. APSEC 2006. 13th Asia Pacific*, pages 467–476. IEEE.
- [59] Johnson, J. H. (1993). Identifying redundancy in source code using fingerprints. In *Proceedings of the 1993 conference of the Centre for Advanced Studies on Collaborative research: software engineering-Volume 1*, pages 171–183. IBM Press.
- [60] Johnson, J. H. (1994). Visualizing textual redundancy in legacy source. In *Proceedings of the 1994 conference of the Centre for Advanced Studies on Collaborative research*, page 32. IBM Press.
- [61] Junhao, W., Jianan, G., Zhuo, J., and Yijiao, Z. (2011). Semantic web service selection algorithm based on qos ontology. In *Service Sciences (IJCSS), 2011 International Joint Conference on*, pages 163–167. IEEE.
- [62] Kamiya, T., Kusumoto, S., and Inoue, K. (2002). Ccfinder: a multilinguistic token-based code clone detection system for large scale source code. *Software Engineering, IEEE Transactions on*, 28(7):654–670.
- [63] Kapsner, C. and Godfrey, M. (2004). Aiding comprehension of cloning through categorization. In *Software Evolution, 2004. Proceedings. 7th International Workshop on Principles of*, pages 85–94. IEEE.
- [64] Kasanen, E. and Lukka, K. (1993). The constructive approach in management accounting research. *Journal of management accounting research*, (5):243–264.
- [65] Keskes, N., Lehireche, A., and Rahmoun, A. (2010). Web services selection based on context ontology and quality of services. *Int. Arab J. e-Technol.*, 1(3):98–105.
- [66] Khutade, P. A. and Phalnikar, R. (2014). Qos aware web service selection and ranking framework based on ontology. *International Journal of Soft Computing and Engineering*, 4(3):77–81.

- [67] Kirkpatrick, S., Jr., D. G., and Vecchi, M. P. (1983). Optimization by simulated annealing. *science*, 220(4598):671–680.
- [68] Knuth, D. (1997). *Stable marriage and its relation to other combinatorial problems: An introduction to the mathematical analysis of algorithms*, volume 10. Amer Mathematical Society.
- [69] Komondoor, R. and Horwitz, S. (2000). Semantics-preserving procedure extraction. In *Proceedings of the 27th ACM SIGPLAN-SIGACT symposium on Principles of programming languages*, pages 155–169. ACM.
- [70] Komondoor, R. and Horwitz, S. (2001). Using slicing to identify duplication in source code. In *Static Analysis*, pages 40–56. Springer.
- [71] Komondoor, R. and Horwitz, S. (2003). Effective, automatic procedure extraction. In *Program Comprehension, 2003. 11th IEEE International Workshop on*, pages 33–42. IEEE.
- [72] Komondoor, R. V. (2003). *Automated duplicated-code detection and procedure extraction*. PhD thesis, UNIVERSITY OF WISCONSIN.
- [73] Kontogiannis, K. (1997). Evaluation experiments on the detection of programming patterns using software metrics. In *Reverse Engineering, 1997. Proceedings of the Fourth Working Conference on*, pages 44–54. IEEE.
- [74] Kontogiannis, K. A., DeMori, R., Merlo, E., Galler, M., and Bernstein, M. (1996). Pattern matching for clone and concept detection. In *Reverse engineering*, pages 77–108. Springer.
- [75] Kotis, K., Vouros, G. A., and Stergiou, K. (2006). Towards automatic merging of domain ontologies: The hcone-merge approach. *Web Semantics: Science, Services and Agents on the World Wide Web*, 4(1):60–79.
- [76] Krinke, J. (2001). Identifying similar code with program dependence graphs. In *Reverse Engineering, 2001. Proceedings. Eighth Working Conference on*, pages 301–309. IEEE.
- [77] Labro, E. and Tuomela, T.-S. (2003). On bringing more action into management accounting research: process considerations based on two constructive case studies. *European Accounting Review*, 12(3):409–442.
- [78] Lakhotia, A., Li, J., Walenstein, A., and Yang, Y. (2003). Towards a clone detection benchmark suite and results archive. In *Program Comprehension, 2003. 11th IEEE International Workshop on*, pages 285–286. IEEE.
- [79] Lanubile, F. and Mallardo, T. (2003). Finding function clones in web applications. In *Software Maintenance and Reengineering, 2003. Proceedings. Seventh European Conference on*, pages 379–386. IEEE.

- [80] Lee, S. and Jeong, I. (2005). Sdd: high performance code clone detection system for large scale source code. In *Companion to the 20th annual ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications*, pages 140–141. ACM.
- [81] Li, Z., Lu, S., Myagmar, S., and Zhou, Y. (2004). Cp-miner: A tool for finding copy-paste and related bugs in operating system code. In *OSDI*, volume 4, pages 289–302.
- [82] Li, Z., Lu, S., Myagmar, S., and Zhou, Y. (2006). Cp-miner: Finding copy-paste and related bugs in large-scale software code. *Software Engineering, IEEE Transactions on*, 32(3):176–192.
- [83] Lincke, R., Lundberg, J., and Löwe, W. (2008). Comparing software metrics tools. In *Proceedings of the 2008 international symposium on Software testing and analysis*, pages 131–142. ACM.
- [84] Liu, C., Chen, C., Han, J., and Yu, P. S. (2006). Gplag: detection of software plagiarism by program dependence graph analysis. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 872–881. ACM.
- [85] Lukka, K. (2000). The key issues of applying the constructive approach to field research. *Reponen, T.(ed.)*.
- [86] Manber, U. et al. (1994). Finding similar files in a large file system. In *Usenix Winter*, volume 94, pages 1–10.
- [87] Marcus, A. and Maletic, J. I. (2001). Identification of high-level concept clones in source code. In *Automated Software Engineering, 2001.(ASE 2001). Proceedings. 16th Annual International Conference on*, pages 107–114. IEEE.
- [88] Mayrand, J., Leblanc, C., and Merlo, E. (1996). Experiment on the automatic detection of function clones in a software system using metrics. In *Software Maintenance 1996, Proceedings., International Conference on*, pages 244–253. IEEE.
- [89] McDermid, E. and Irving, R. W. (2011). Popular matchings: structure and algorithms. *Journal of combinatorial optimization*, 22(3):339–358.
- [90] McMinn, P. (2004). Search-based software test data generation: a survey. *Software Testing, Verification and Reliability*, 14(2):105–156.
- [91] McNabb, D. (2002). *Research Methods in Public Administration and Nonprofit Management: Qualitative and Quantitative Approaches*.
- [92] Nagy, W., Mokhtar, H. M., and El-Bastawissy, A. (2011). A flexible tool for web service selection in service oriented architecture. *International Journal of Advanced Computer Science and Applications*, 2(12):191–201.

- [93] O'Malley, G. (2007). *Algorithmic aspects of stable matching problems*. PhD thesis, University of Glasgow.
- [94] Patenaude, J.-F., Merlo, E., Dagenais, M., and Laguë, B. (1999). Extending software quality assessment techniques to java systems. In *Program Comprehension, 1999. Proceedings. Seventh International Workshop on*, pages 49–56. IEEE.
- [95] Prechelt, L., Malpohl, G., and Philippsen, M. (2002). Finding plagiarisms among a set of programs with jplag. *J. UCS*, 8(11):1016.
- [96] Rieger, M., Ducasse, S., and Lanza, M. (2004). Insights into system-wide code duplication. In *Reverse Engineering, 2004. Proceedings. 11th Working Conference on*, pages 100–109. IEEE.
- [97] Roy, C. K. (2009). Detection and analysis of near-miss software clones. In *Software Maintenance, 2009. ICSM 2009. IEEE International Conference on*, pages 447–450. IEEE.
- [98] Roy, C. K. and Cordy, J. R. (2007). A survey on software clone detection research. Technical report, Citeseer.
- [99] Roy, C. K., Cordy, J. R., and Koschke, R. (2009). Comparison and evaluation of code clone detection techniques and tools: A qualitative approach. *Science of Computer Programming*, 74(7):470–495.
- [100] Schleimer, S., Wilkerson, D. S., and Aiken, A. (2003). Winnowing: local algorithms for document fingerprinting. In *Proceedings of the 2003 ACM SIGMOD international conference on Management of data*, pages 76–85. ACM.
- [101] Schmidt, M. E. (2000). *Implementing the IEEE software engineering standards*. Sams.
- [102] Sellers, B. H. (1996). Object-oriented metrics. measures of complexity.
- [103] Tairas, R. and Gray, J. (2006). Phoenix-based clone detection using suffix trees. In *Proceedings of the 44th annual Southeast regional conference*, pages 679–684. ACM.
- [104] Tan, J. J. (1991). A necessary and sufficient condition for the existence of a complete stable matching. *Journal of Algorithms*, 12(1):154–178.
- [105] Tao, C.-H. and Feng, Z.-Y. (2010). Novel qos-aware web service recommendation model. *Application Research of Computers*, 10:83.
- [106] Ueda, Y., Kamiya, T., Kusumoto, S., and Inoue, K. (2002). On detection of gapped code clones using gap locations. In *Software Engineering Conference, 2002. Ninth Asia-Pacific*, pages 327–336. IEEE.
- [107] Wahler, V., Seipel, D., von Gudenberg, J. W., and Fischer, G. (2004). Clone detection in source code by frequent itemset techniques. In *SCAM*, volume 4, pages 128–135.

- [108] Watson, A. H., McCabe, T. J., and Wallace, D. R. (1996). Structured testing: A testing methodology using the cyclomatic complexity metric. *NIST special Publication*, 500(235):1–114.
- [109] WEISER, M. (1984). Program slicing. *IEEE TRANSACTIONS ON SOFTWARE ENGINEERING*, 10(4).
- [110] Wettel, R. and Marinescu, R. (2005). Archeology of code duplication: Recovering duplication chains from small duplication fragments. In *Symbolic and Numeric Algorithms for Scientific Computing, 2005. SYNASC 2005. Seventh International Symposium on*, pages 8–pp. IEEE.
- [111] Xu, P., Wang, Y., Cheng, L., and Zang, T. (2010). Alignment results of sobom for oaei 2010. *Ontology Matching*, page 203.
- [112] Yang, W. (1991). Identifying syntactic differences between two programs. *Software: Practice and Experience*, 21(7):739–755.
- [113] Yu, T., Zhang, Y., and Lin, K.-J. (2007). Efficient algorithms for web services selection with end-to-end qos constraints. *ACM Transactions on the Web (TWEB)*, 1(1):6.

Appendix A

Screen-shots

Here are some snapshots which are taken during the experiments, which relevant to the ClonDR tool and applied on both job search and netbeans-javadoc projects.

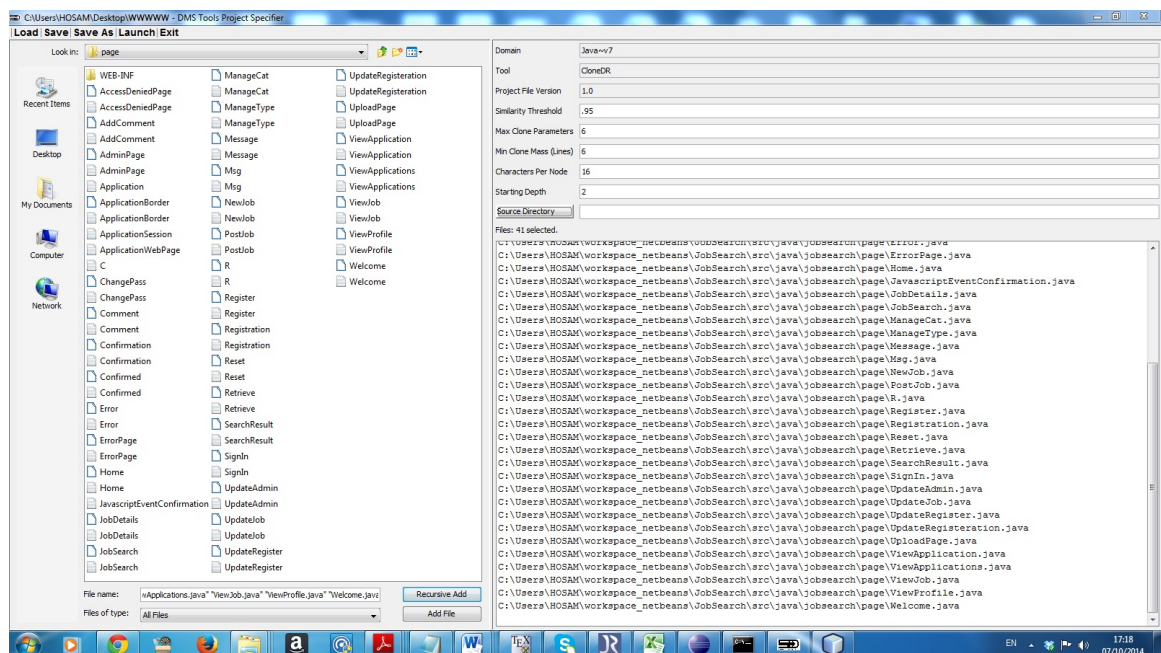


Fig. A.1 Job search project browsed by ClonDR.

Appendix 1

Evaluation Clone Detection Report for Project File:

C: /Users/HOSAM/Desktop/qqq

Table of Contents

1. Detection Parameters
2. Files Analyzed
3. Clone Detection Statistics
4. CloneSets by Size
5. CloneSets by Parameters
6. Detected CloneSets

Detection Parameters	
Value	
Similarity Threshold	95%
Maximum parameter count	6
Minimum Mass (Lines)	6.0
Characters per node	16
Starting height	2

Clone Detection Statistics	
Statistic	Value
File Count	64
Total Source Lines of Code (SLOC)	5548
Estimated SLOC before preprocessing	5611
Expanded SLOC after preprocessing	5407
Total CloneSets	41
Exact-match CloneSets	19
Near-miss CloneSets	22
Number of cloned SLOC	715
SLOC in clones %	12.7%
Estimated removable SLOC	304
Possible SLOC reduction %	5.42%
Possible SLOC reduction in expanded file %	5.62%

CloneSets by Clone Size	
Clone Size (SLOC)	Number of CloneSets
1	6
2	1
3	3
4	2
5	2

Fig. A.2 Some statistics of job search project based on CloneDR.

CloneSets by Clone Size	
Clone Size (SLOC)	Number of CloneSets
1	6
2	1
3	3
4	2
5	2
6	5
7	2
8	5
9	4
10	3
11	1
12	1
13	3
14	1
15	2
19	1
20	1

CloneSets by Parameter Count	
Number of Parameters	Number of CloneSets
0	19
1	7
2	6
3	6
4	2
5	1

Detected CloneSets (Sorted by Total Mass of CloneSet)					
CloneSet Details	Clone Mass	Clones in CloneSet	Parameter Count	Clone Similarity	Syntax Category [Sequence Length]
CloneSet1	6	5	1	0.990	statement_sequence[6]
CloneSet2	9	2	0	1.000	import_statements[9]
CloneSet3	13	2	5	0.970	statement_sequence[11]
CloneSet4	20	2	3	0.951	statement_sequence[14]
CloneSet5	10	2	1	0.994	statement_sequence[9]
CloneSet6	1	2	0	1.000	parameters
CloneSet7	19	2	1	0.994	executable_statement
CloneSet8	9	2	0	1.000	statement_sequence[9]
CloneSet9	5	2	0	1.000	statement_sequence[5]
CloneSet10	1	2	0	1.000	parameters

Evaluation Version: Only 10 samples of medium size (maximum 30 lines) of the 41 detected clones are reported here.

Fig. A.3 Job search project analysed by CloneDR.

Appendix I

Evaluation Clone Detection Report for Project File:

C:/Users/HOSAM/Desktop/qoq

Table of Contents

1. Detection Parameters
2. Files Analyzed
3. Clone Detection Statistics
4. CloneSets by Size
5. CloneSets by Parameters
6. Detected CloneSets

Detection Parameters	
Value	
Similarity Threshold	95%
Maximum parameter count	6
Minimum Mass (Lines)	6.0
Characters per node	16
Starting height	2

Clone Detection Statistics	
Statistic	Value
File Count	101
Total Source Lines of Code (SLOC)	14360
Estimated SLOC before preprocessing	14461
Expanded SLOC after preprocessing	13140
Total CloneSets	82
Exact-match CloneSets	27
Near-miss CloneSets	55
Number of cloned SLOC	2615
SLOC in clones %	18.1%
Estimated removable SLOC	1285
Possible SLOC reduction %	8.89%
Possible SLOC reduction in expanded file %	9.78%

CloneSets by Clone Size		
Clone Size (SLOC)	Number of CloneSets	
2	5	
3	3	
4	9	
5	12	

Fig. A.4 Some statistics of netbeans-javadoc project based on CloneDR.

14	2
16	3
17	1
18	1
19	1
20	2
22	1
23	1
24	1
26	2
27	2
29	1
31	1
34	1
43	2
47	1
73	1
86	1

CloneSets by Parameter Count		
Number of Parameters	Number of CloneSets	
0	27	
1	16	
2	14	
3	8	
4	9	
5	7	
6	1	

Detected CloneSets (Sorted by Total Mass of CloneSet)					
CloneSet	Clone Mass	Clones in CloneSet	Parameter Count	Clone Similarity	Syntax Category [Sequence Length]
CloneSet1	27	5	4	0.989	statement_sequence[22]
CloneSet2	47	3	5	0.971	class_body_declarations[2]
CloneSet3	43	2	0	1.000	nested_class_declaration
CloneSet4	31	2	3	0.979	statement_sequence[12]
CloneSet5	26	2	5	0.974	statement_sequence[26]
CloneSet6	17	3	0	1.000	statement_sequence[17]
CloneSet7	34	2	6	0.960	type_declaration
CloneSet8	16	3	4	0.968	statement_sequence[16]
CloneSet9	43	2	0	1.000	block
CloneSet10	26	2	1	0.997	class_body_declarations[9]

Evaluation Version: Only 10 samples of medium size (maximum 30 lines) of the 82 detected clones are reported here.

Fig. A.5 netbeans-javadoc project analysed by CloneDR.

A.1 Clonesets Snapshots

Here we list different types of software clones which are recognised by both approaches.

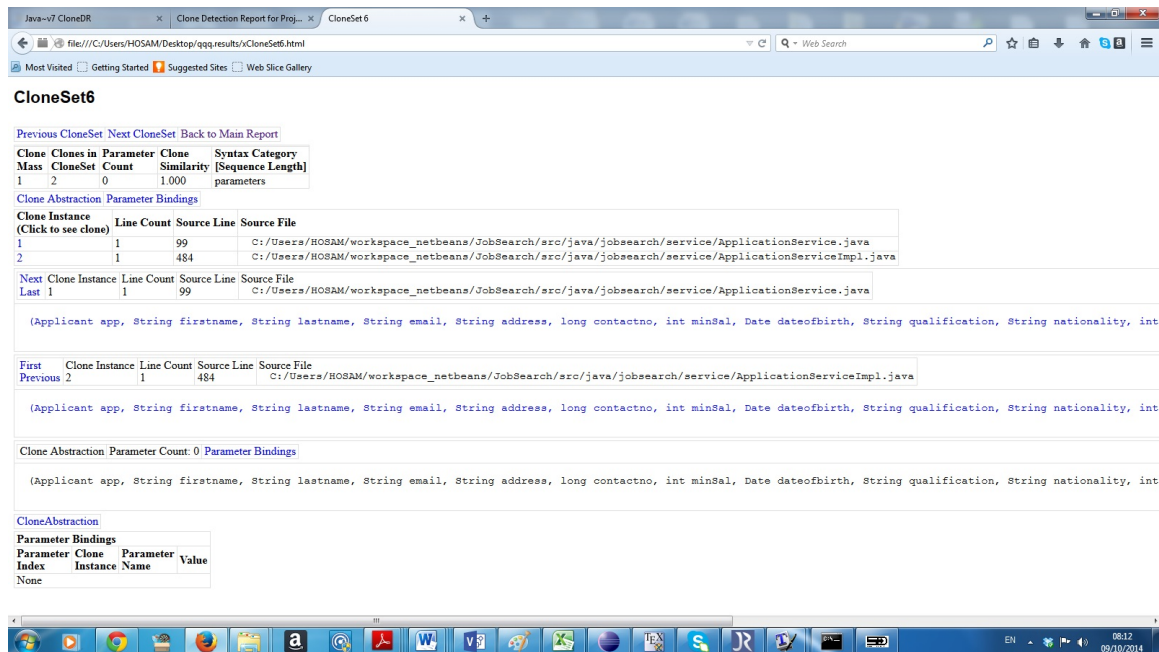


Fig. A.6 Clone set of type 1 (Identical clone).

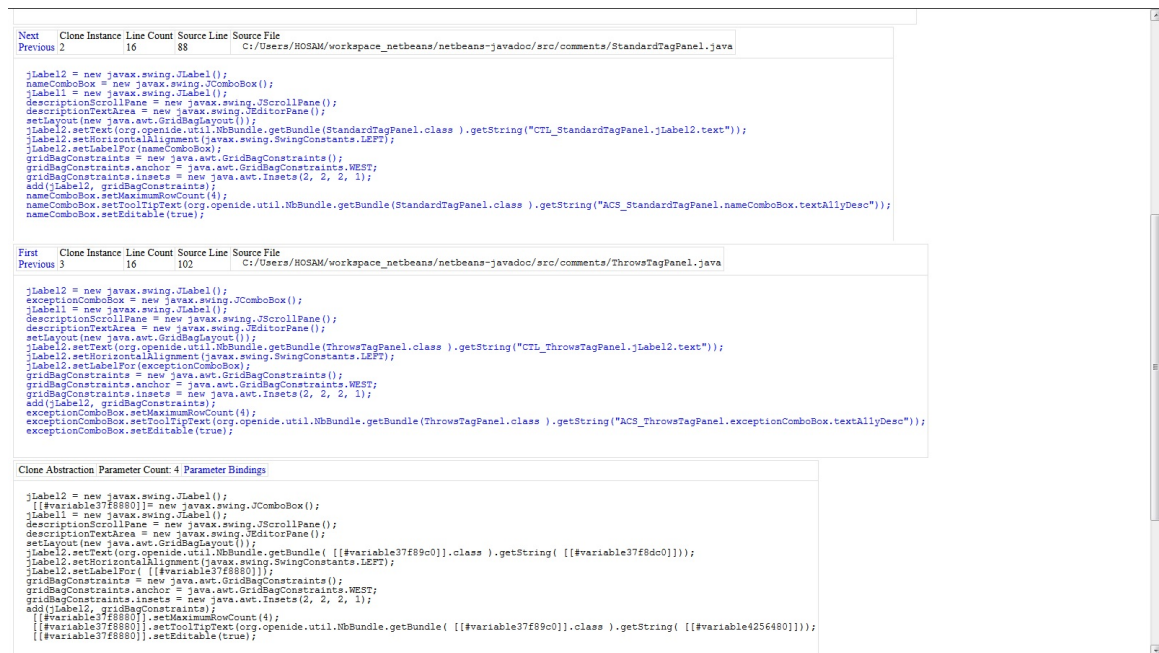


Fig. A.7 Clone set of type 2.

Appendix B

Source Code

In this part, we provide the source code of the program simulation based on our adaptation. This program presents the selection process among services and requests. The program built in Java using eclipse-standard-kepler.

```
=====StableMarriage=====

import java.util.*;

/**
 * An implementation of the stable marriage algorithm
 *
 * _____
 * @author Hosam AlHakami (STRL, DMU, UK)
 * hosam.alhakami@myemail.dmu.ac.uk
 * _____ */

public class StableMarriage {
    // Number of men (=number of women)
    private int n;

    // Preference tables (size nxn)
```

```
public int[][] jobPref;
public int[][] applPref;

private static final boolean DEBUGGING = false;
private Random rand = new Random();

/**
 * Creates and solves a random stable marriage problem of size n, where n is
 * given on the command line.
 */

/**
 * Creates a marriage problem of size n with random preferences.
 */
public StableMarriage(int n) {
    this.n = n;
    jobPref = new int[n][];
    applPref = new int[n][];
    for (int i = 0; i < n; i++) {
        jobPref[i] = new int[n];
        createRandomPrefs(jobPref[i]);
        applPref[i] = new int[n];
        createRandomPrefs(applPref[i]);
    }
}

/**
 * Puts the numbers 0 .. v.length - 1 in the vector v in random order.
 */
private void createRandomPrefs(int[] v) {
    // Create a vector with the values 0, 1, 2, ...
    for (int i = 0; i < v.length; i++)
        v[i] = i;
}
```

```

// Create a random permutation of this vector.
for (int i = v.length - 1; i > 0; i--) {
    // swap v[i] with a random element v[j], j <= i.
    int j = rand.nextInt(i + 1);
    int temp = v[i];
    v[i] = v[j];
    v[j] = temp;
}
}

/**
 * Returns a stable marriage in the form an int array v, where v[i] is the
 * man married to woman i.
 */
public int[][] stableJob() { // job optimal
    // Indicates that woman i is currently engaged to
    // the man v[i].

    int[][] current = new int[n][n];
    final int NOT_ENGAGED = -1;
    for (int i = 0; i < current.length; i++)
        current[i][0] = NOT_ENGAGED;

    // List of jobs that are not currently engaged.
    LinkedList<Integer> freeJobs = new LinkedList<Integer>();
    for (int i = 0; i < current.length; i++)
        freeJobs.add(i);

    // next[i] is the next applicant to whom i has not yet proposed.
    int[] next = new int[n];

    // computeRanking();
    while (!freeJobs.isEmpty()) {

```

```

    int m = freeJobs.remove();
    int w = jobPref[m][next[m]];
    next[m]++;
    printDebug("m=" + m + "_w=" + w);
    if (current[w][0] == NOT_ENGAGED) {
        current[w][0] = m;
    } else {
        int m1 = current[w][0];
        if (prefers(w, m, m1, "j")) {
            current[w][0] = m;
            freeJobs.add(m1);
        } else {
            freeJobs.add(m);
        }
    }
}

// calculate LD
for (int i = 0; i < current.length; i++) {
    int c;
    for (int j = 0; j < jobPref.length; j++) {

        c = jobPref[i][j];
        if (c == current[i][0]) {
            current[i][1] = j + 1;
        }
    }
}

// calculate CONT
for (int i = 0; i < current.length; i++) {
    int c = 0, d = 0;
    for (int j = 0; j < jobPref.length; j++) {

```

```

        if (current[i][0] == jobPref[i][j]) {
            c = j + 1;
        }
    }

    for (int j = 0; j < applPref.length; j++) {
        if (applPref[current[i][0]][j] == i) {
            d = j + 1;
        }
    }

    current[i][2] = Math.abs(c - d);
    current[i][3] = Math.abs((current[i][1] + current[i][2]) / 2); // calculate

}

return current;
}

public int[][] stableAppl() {

    // Indicates that job i is currently engaged to
    // the man v[i].
    int[][] current = new int[n][n];
    final int NOT_ENGAGED = -1;
    for (int i = 0; i < current.length; i++)
        current[i][0] = NOT_ENGAGED;

    // List of applicants that are not currently engaged.
    LinkedList<Integer> freeAppl = new LinkedList<Integer>();
    for (int i = 0; i < current.length; i++)

```

```
        freeAppl.add(i);

    int[] next = new int[n];

    while (!freeAppl.isEmpty()) {
        int m = freeAppl.remove();
        int w = applPref[m][next[m]];
        next[m]++;
        printDebug("m=" + m + " _w=" + w);
        if (current[w][0] == NOT_ENGAGED) {
            current[w][0] = m;
        } else {
            int ml = current[w][0];
            if (prefers(w, m, ml, "a")) {
                current[w][0] = m;
                freeAppl.add(ml);
            } else {
                freeAppl.add(m);
            }
        }
    }
}

// calculate LD
for (int i = 0; i < current.length; i++) {
    int c;
    for (int j = 0; j < jobPref.length; j++) {

        c = applPref[i][j];
        if (c == current[i][0]) {
            current[i][1] = j + 1;
        }
    }
}
```

```

// calculate CONT
for (int i = 0; i < current.length; i++) {
    int c = 0, d = 0;
    for (int j = 0; j < applPref.length; j++) {
        if (current[i][0] == applPref[i][j]) {
            c = j + 1;
        }
    }

    for (int j = 0; j < jobPref.length; j++) {
        if (jobPref[current[i][0]][j] == i) {
            d = j + 1;
        }
    }

    current[i][2] = Math.abs(c - d);
    current[i][3] = Math.abs((current[i][1] + current[i][2]) / 2); // calculate
}

return current;
}

/**
 * Returns true iff w prefers x to y.
 */
private boolean prefers(int w, int x, int y, String optimal) {
    for (int i = 0; i < n; i++) {
        int pref;
        if (optimal == "j") {
            pref = applPref[w][i];

```

```
        } else {
            pref = jobPref[w][i];
        }
        if (pref == x)
            return true;
        if (pref == y)
            return false;
    }
    // This should never happen.
    System.out.println("Error_in_applPref_list_" + w);
    return false;
}

public void printMarriageReg(int [][] m, String optimal) {
    int i;
    if (optimal == "j") { // jobs optimal
        System.out.println("Stable_Matches_(Request+_Service):" + "\n"
            + "Request_Optimal");
        for (i = 0; i < m.length; i++) {
            System.out.print("R" + (i + 1) + "_=>" + "S" + (m[i][0] + 1)
                + "\t\t");
            if (i % 3 == 0) {
                System.out.print("\n");
            }
        }
    }

    } else if (optimal == "a") { // applicants optimal
        System.out.println("Stable_Matches_(Services+_Requests):_" + "\n"
            + "Service_Optimal");
        for (i = 0; i < m.length; i++) {
            System.out.print("S" + (i + 1) + "_=>" + "R" + (m[i][0] + 1)
                + "\t\t");
            if (i % 3 == 0) {
```



```

        System.out.print("\n");
    }
}

}

}

}

public void printMarriageExt(int [][] m, String optimal) {
    int i;
    if (optimal == "j") { // jobs optimal
        System.out.println("Stable_Matches_(Request+_Service):" + "\n"
            + "Request_Optimal");
        for (i = 0; i < m.length; i++) {
            System.out.print("R" + (i + 1) + "_=>" + "S" + (m[i][0] + 1)
                + "\t" + "_LD=" + m[i][1] + "\t" + "_CONT="
                + m[i][2] + "\t" + "_WGT=" + m[i][3] + "\n");
            /*
             * if(i % 2 ==0 ) { System.out.print("\n"); }
             */
        }
    } else if (optimal == "a") { // applicants optimal
        System.out.println("Stable_Matches_(Services+_Requests):" + "\n"
            + "Service_Optimal");
        for (i = 0; i < m.length; i++) {
            System.out.print("S" + (i + 1) + "_=>" + "R" + (m[i][0] + 1)
                + "\t" + "_LD=" + m[i][1] + "\t" + "_CONT="
                + m[i][2] + "\t" + "_WGT=" + m[i][3] + "\n");
            /*
             * if(i % 2 ==0 ) { System.out.print("\n"); }
             */
        }
    } else if (optimal == "e") {

```

```
        for (i = 0; i < m.length; i++) {
            System.out.print("R" + (i + 1) + "=>" + "S" + (m[i][0] + 1)
                + "\t\t");

            if (i % 3 == 0) {
                System.out.print("\n");
            }
        }

    }

}

private void printDebug(String s) {
    if (DEBUGGING) {
        System.out.println(s);
    }
}

}
```