The University of Nottingham

UNITED KINGDOM · CHINA · MALAYSIA

AN INVESTIGATION OF MULTI-OBJECTIVE HYPER-HEURISTICS FOR MULTI-OBJECTIVE OPTIMISATION

by

Mashael Maashi, BSc, MSc

Thesis submitted to the University of Nottingham for the degree of Doctor of Philosophy

April 2014

Contents

Abstract

In this thesis, we investigate and develop a number of online learning selection choice function based hyper-heuristic methodologies that attempt to solve multi-objective unconstrained optimisation problems. For the first time, we introduce an online learning selection choice function based hyper-heuristic framework for multi-objective optimisation. Our multi-objective hyper-heuristic controls and combines the strengths of three well-known multi-objective evolutionary algorithms (NSGAII, SPEA2, and MOGA), which are utilised as the low level heuristics. A choice function selection heuristic acts as a high level strategy which adaptively ranks the performance of those low-level heuristics according to feedback received during the search process, deciding which one to call at each decision point. Four performance measurements are integrated into a ranking scheme which acts as a feedback learning mechanism to provide knowledge of the problem domain to the high level strategy. To the best of our knowledge, for the first time, this thesis investigates the influence of the move acceptance component of selection hyper-heuristics for multi-objective optimisation. Three multi-objective choice function based hyper-heuristics, combined with different move acceptance strategies including All-Moves as a deterministic move acceptance and the Great Deluge Algorithm (GDA) and Late Acceptance (LA) as a non-deterministic move acceptance function.

GDA and LA require a change in the value of a single objective at each step and so a well-known hypervolume metric, referred to as *D* metric, is proposed for their applicability to the multi-objective optimisation problems. *D* metric is used as a way of comparing two non-dominated sets with respect to the objective space. The performance of the proposed multi-objective selection choice function based hyper-heuristics is evaluated on the Walking Fish Group (WFG) test suite which is a common benchmark for multi-objective optimisation. Additionally, the proposed approaches are applied to the vehicle crashworthiness design problem, in order to test its effectiveness on a real-world multi-objective problem. The results of both benchmark test problems demonstrate the capability and potential of the multi-objective hyper-heuristic approaches in solving continuous multi-objective optimisation problems. The multi-objective choice function Great Deluge Hyper-Heuristic (HHMO_CF_GDA) turns out to be the best choice for solving these types of problems.

Acknowledgements

I would like to express my respect and gratitude to my supervisor Professor Graham Kendall for giving me this opportunity to pursue a PhD at the University of Nottingham. I feel very lucky to be one of his students. Sincere gratitude to Dr Ender Özcan who joined Professor Kendall soon after in the supervision of my PhD research. I would like to thank both of them for their ideas, insights, trust, patience, support and valuable guidance, as well as sound advice and suggestions throughout my PhD. They are, and always will be, sources of inspiration, and ensured that my project remained on track. I appreciate their kind help and encouragement throughout my PhD journey which has greatly helped me to overcome difficulties, gain confidence and move forward.

My thanks also go to the Automated Scheduling, Optimisation and Planning (ASAP) Research Group and the School of Computer Science at the University of Nottingham for their help and assistance. I would like to acknowledge the many colleagues who helped make the time enjoyable especially my friends Dr Huanlai Xing and Shahriar Asta. I would like to thank Dr. Dario Landa-Silva from University of Nottingham for his great help and collaboration during my PhD study. I would also express my deep appreciation to Professor Kalyanmoy Deb from Indian Institute of Technology Kanpur for his advice and suggestions.

I would like to express my gratefulness to my Ph.D viva examiners; Dr Dario Landa-Silva (Internal examiner) and Professor Peter Fleming from University of Sheffield (external examiner), for providing insightful comments which has greatly helped to improve the quality of my thesis.

Dedicated to my parents - the greatest Dad and Mum - without their prayers, support and encouragement, I would not have come to the UK to pursue my research. I am grateful to Allah that they are in my life and what they dream about comes true.

Dedicated to Grandma, my sisters - Safa, Mervet, Manal, Khould, my brother Abdulaziz, and all my nephews and nieces for their prayers, love, care and patience. Thanks for being a good sister(s)/brother and for all the late night/early morning chats. Although you were all far from me in distance, your hearts live inside my soul.

A special dedication to my sister Marwah (my soulmate), who shared with me all the foreignness of life; as well as joyful and happy and even sad and difficult moments. Thanks for her inspiration, support, encouragement, prayers, care, patience, travelling and cooking☺. It would have not been possible for me to finish my PhD without her warm heart.

To the who that is my life companion, my husband, Abdulrahman. Allah sent you into my life at the *right* moment as Mr. *Right*. Having you in my life makes it easier, more enjoyable, colourful, full of happiness and full of love. Thanks for love, care, encouragement, prayers and patience. Love you Babe.

Many thanks also to all of my friends - Amani, Mona, Shatha, Mashael, Hend, Zhorah, Azhar and Afaf - for listening to me and supporting me. I have enjoyed these years and my life would not have been as enjoyable without you all.

Lastly, I am highly grateful to the University of Tabuk and the Saudi Cultural Bureau in the UK and Ireland for making available the financial funding without which I would not have been able to undertake my research. I would like also to thank all those who painstakingly guided me during the project. Thanks for your prayers, help and co-operation.

Mashael S. Maashi
4, December 2013

آهداء وشكر

الحمدلله رب العالمين...الحمدلله اولاً واخيراَ.. الحمدلله حمداً كثيراً طيباً مباركاً فيه.. الحمدلله عدد خلقه ورضا نفسه ومداد كلماته .. وعدد أوراق الشجر وحباب الرمل وزخات المطر.. الحمدلله مل السموات والأرض ومن فيهن وعدد انفاس البشر وقطرات البحر.. الحمد لله الذي تتم بنعمته الصالحات.. الحمد لك ياالله لا لمخلوق سواك.. فالخير منك واليك والفضل فضلك والعلم علمك .. ربي هذا العلم جزء ضئيل من علمك الكبير .. فما أوتينا من العلم إلا قليلاً .. اسألك ياالله أن تتقبله مني وتنفعي به في حياتي ومماتي وتنفع به خلقك.. وتعطي والديّ اجر هذا العلم ،،، آمين .

آهداء الى والديّ أعظم أب وأم.. د. سليمان معشي ود. عائشة العبدلي – هذه الاطروحة آهداء بسيط لكما على صبركما وتحملكما لبعدي عنكما وعلى كل الآلام التى سببتها غربتي لكما. لم احصل على الدكتوراة انا.. فانتم من حصل عليها ..لولا الله ومن ثم دعواتكما الصادقة لي والتي تلهج بها قلوبكما قبل السنتكما ليل نهار ودعمكما وتشجيعكما ورعايتكما الدائمة لما استطعت الإلتحاق بالبعثة ومواصلة تعليمي والحصول على درجة الدكتوراة .
ممتنة إلى الله كونكما في حياتي وأشكره على استجابته لدعواتكما وجعل حلمكما لي واقع جميل. ادعوا الله ان يطيل لي في عمركما ويجعلكما دوماً ذخراً وسنداً لي. أحبكم بابا وماما.

آهداء الى ستى طيبة اطال الله لي في عمرها والى روح ستي فاطمة والى أخواتي العزيزات – صفاء ، مرفت، منال، خلود ، و أخي العزيز عبد العزيز ، والى جميع أولاد وبنات اخواتي. شكراً لدعائكم ، حبكم ، اهتمامكم وصبركم . شكراً لكونكم نعم الأخوات /الأخ . شكراً على كل المحادثات والاتصالات الجميلة في الصباح الباكر وآواخر الليل. على الرغم من بعدكم عني في المسافة إلا أن قلوبكم كانت ومازالت تعيش دوماً داخل روحي.

آهداء خاص لأختي مروة (توأم الروح)، التى شاركتني الغربة بحلوها ومرها.. بكل لحظاتها البهيجة والسعيدة وحتى الحزينة والصعبة منها. لم يكن من الممكن لي إنهاء برنامج الدكتوراه دون وجودك وعطفك وقلبك الدافئ . شكراً لإلهامك، دعمك، تشجيعك ، دعواتك ولرعايتك لي عند مرضي وصبرك العظيم في لحظات إنهياري وضعفي. شكراً على رفقتك الحلوة في كل رحلاتي. شكراً على كل الوجبات والأطباق الشهية واللذيذة التى صنعتها يديك !! :)

آهداء مغلف بالحب إلى شريك حياتي .. زوجي وحبيبي - عبد الرحمن.. احمد الله على أنه جعلك لي نصيب فانت نعم الزوج والصديق والحبيب. وجودك في حياتي جعلها أسهل وأكثر متعة ..مليئة بالسعادة والفرح والحب. ممتنة لحبك ولرعايتك واهتمامك وعطفك وتشجيعك ودعواتك وصبرك ولقلبك الكبير الطيب. قد لا تراك عيني في بعض الأحيان الا أنك في القلب دوماً حاضراً .أحبك بي بي.

الشكر الجزيل لجميع صديقاتي الصدوقات - أماني، منى، شذى، مشاعل،ايمان، هند، ازهار، زهرة وعفاف.
شكراً لاستماعكم لي ودعمكن وقوفكن دوماً جانبي في الغربة. وبوجودكن تحولت مرارة الغربة الى سكر. لولاكن لما اصبحت كل تلك السنوات من الاغتراب ذكرى جميلة وممتعة.

أتقدم بخالص احترامي وامتناني لمشرفي الدراسي البروفيسور غراهام كيندال على أتاحة الفرصة لي لتحضير الدكتوراه في جامعة نوتنغهام. إني محظوظة جداً وفخورة كوني أحد طلابه. خالص امتناني أيضاً للدكتور اندر أوزكان الذي إنضم مع البروفيسور كيندال للإشراف على بحثي في الدكتوراه . أود أن أشكرهما على كل الافكار، الثقة، الصبر، الدعم والتوجيهات القيمة. فضلاً على تقديمهما المشورة والاقتراحات وحرصهم على نجاح بحثي. هما كانوا وسيكونوا دوماً مصدر إلهام لي. ممتنة لمساعدتهما وتشجيعهما الدائم طوال مرحلة الدكتوراه . فقد كان لهما الفضل بعد الله بمساعدتي للتغلب على الصعوبات، وكسب الثقة والمضي قدماً.

كما أتوجه بالشكر إلى مجموعة البحث (الجدولة الآلية والتخطيط الأمثل) وكلية علوم الحاسب الآلي في جامعة نوتنغهام على مساعدتهم ودعمهم. وكما أود أن أشكر العديد من الزملاء الذين جعلوا وقت الدراسة ممتع وخاصة أصدقائي الدكتور خوليلاين شينغ وشهريار آستا. وأود ايضاً أن اشكرالدكتور داريو سيلفا لاندا، من جامعة نوتنغهام على تعاونه معي خلال دراستي للدكتوراه . وأود أيضا أن أعرب عن تقديري العميق لبروفيسور كليموني ديب من المعهد الهندي للتكنولوجيا في كانبور- الهند على تقديمه المشورة والاقتراحات.

وكما أود أن أعبر عن عميق امتناني للممتحني مناقشة الدكتوراة : الدكتور داريو سيلفا لاندا، (ممتحن داخلي) والبروفيسور بيتر فليمنج من جامعة شيفيلد (ممتحن خارجي)، على تعليقاتهم الثاقبة التي ساعدت إلى حد كبير في التحسين من جودة أطروحتي واخراجها بأفضل شكل.

اتقدم ايضاً بجزيل الشكر الي جامعة تبوك والملحقية الثقافية السعودية في المملكة المتحدة وايرلندا لتمويلهم ودعمهم المادي. وآخيراً .. لا انسى من الشكر والعرفان كل شخص دعمني خلال مدة دراستي واغترابي لو باتصال .. رسالة جوال.. بابتسامة أو حتى دعوة صادقة في ظهر الغيب.

مشاعل سليمان معشي
نوتنجهام- ليستر
المملكة المتحدة
شتاء ٢٠١٤/٢٠١٣

List of Figures

List of Tables

1 Introduction

1.1 Background and Motivations

Many real-world problems are complex. Due to their (often) NP-hard nature, researchers and practitioners frequently resort to problem tailored heuristics to obtain a reasonable solution in a reasonable amount of time. Hyper-heuristics are emerging methodologies designed to generate high quality solutions in an attempt to solve difficult computational optimisation problems by performing a search over the space of heuristics rather than searching the solution space directly. One of their main aims is to raise the level of generality of search methodologies, and to automatically adapt the algorithm by combining the strength of each heuristic and making up for the weaknesses of others. This process requires the incorporation of a learning mechanism into the algorithm to adaptively direct the search at each decision point for a particular state of the problem or the stage of search. Hyper-heuristics have a strong link to Operations Research in terms of finding optimal or near-optimal solutions to computational search problems. It is also firmly linked to a branch of Artificial Intelligence in terms of machine learning methodologies (Burke et al., 2010). In a hyper-heuristic approach, different heuristics (or heuristic components) can be selected, generated or combined to solve a given optimisation problem in an efficient way. Generally, there are two recognized types of hyper-heuristics: *selection* and *generation* hyper-heuristics. A selection hyper-heuristic framework manages a set of low level heuristics and chooses the *best* one at any given time using a performance measure for each low level heuristic. This type of hyper-heuristic comprises two main stages: *heuristic selection* and *move acceptance* strategy.

Hyper-heuristics have drawn increasing attention from the research community in recent years, although their roots can be traced back to the 1960s. Numerous hyper-heuristic papers have been published and several studies are still being undertaken in this area of research. However, the majority of research in this area has been limited to single-objective optimisation. Hyper-heuristics for multi-objective optimisation problems is a relatively new area of research in Operational Research and Evolutionary Computation (Burke et al., 2010; Özcan et al., 2008). To date, few studies have been identified that deal with hyper-heuristics for multi-objective problems. Burke et al. (2003a) proposed a hyper-heuristic for multi-objective problems which was based on tabu search (TSRoulette Wheel). Veerapen et al. (2009) presented another multi-objective hyper-heuristic approach that comprised two phases. An online selection hyper-heuristic, Markov chain based, (MCHH) has been investigated in McClymont and Keedwell (2011). Gomez and Terashima-Marỉn (2010) propose a new hyper-heuristic based on the multi-objective evolutionary algorithm NSGAII (Deb and Goel, 2001). A hyper-heuristic-based encoding was proposed by Armas et al. (2011) and Miranda et al. (2010) for solving strip packing and cutting stock problems. An adaptive multi-method search called AMALGAM is proposed by Vrugt and Robinson (2007). A multi-strategy ensemble multi-objective evolutionary algorithm called MS-MOEA for dynamic optimisation is proposed by Wang and Li (2010). In Furtuna et al. (2012) a multi-objective hyper-heuristic for the design and optimisation of a stacked neural network is proposed. Rafique (2012) presented a multi-objective hyper-heuristic optimisation scheme for engineering system design problems. Vázquez-Rodríguez and Petrovic (2013) proposed a multi-indicator hyper-heuristic for multi-objective optimisation.Len et al. (2009) proposed a hypervolume-based hyper-heuristic for a dynamic-mapped multi-objective island-based model. Bai et al. (2013) proposed a multiple neighbourhood hyper-heuristic for two-dimensional shelf space allocation problem. Kumari et al. (2013) presented a multi-objective hyper-heuristic genetic algorithm (MHypGA) for the solution of Multi-objective Software Module Clustering Problem.

None of the above studies have used multi-objective evolutionary algorithms (MOEAs), only in Rafique (2012), Gomez and Terashima-Marỉn (2010) and Vrugt and Robinson (2007), and no continuous and standard multi-objective test problems have been studied, only in McClymont and Keedwell (2011), Vrugt and Robinson (2007), Len et al. (2009) and Vázquez-Rodríguez and Petrovic (2013). Moreover, none of the previous hyper-heuristics make use of the components specifically designed for multi-objective optimisation that we introduce in this thesis. Our multi-objective hyper-heuristic framework addresses four main research areas, these being: multi-objective evolutionary algorithms, hyper-heuristics, meta-heuristics and multi-objective test problems. This thesis highlights the lack of scientific study that has been conducted in these areas and investigates the design of a hyper-heuristic framework for multi-objective optimisation and develops hyper-heuristic approaches for multi-objective optimisation (HHMOs) to solve continuous multi-objective problems. We focus on an online learning selection hyper-heuristics for multi-objective optimisation and their hybridisation with multi-objective evolutionary algorithms which controls and combines the strengths of three well-known multi-objective evolutionary algorithms (NSGAII (Deb and Goel, 2001), SPEA2 (Zitzler et al., 2001) and MOGA (Fonseca and Fleming, 1998)). The performance of the multi-objective hyper-heuristic approaches (HHMOs), when combined with a choice function that uses different move acceptance strategies such as all-moves, a great deluge algorithm (Dueck, 1993) and late acceptance (Burke and Bykov, 2008) is also studied.

1.2 Aims and Scope

References to multi-objective hyper-heuristics are scarce. This research, combines hyper-heuristic methodologies and multi-objective evolutionary algorithms in one approach in order to tackle multi-objective problems, in particular, continuous unconstrained real-valued problems.

The main aim of this research is to investigate hyper-heuristic approaches for multi-objective optimisation problems based on multi-objective evolutionary algorithms (MOEAs), in order to produce a set of high quality solutions (i.e. not necessarily optimal) compared with the existing approaches in the MOEA literature.

In order to achieve this aim, several objectives are outlined as follows:

- Study existing meta-heuristics for single-objective and multi-objective optimisation.
- Understand existing hyper-heuristic methodologies particularly those based on heuristic selection.
- Understand existing multi-objective evolutionary algorithms and identifying their strengths and weakness.
- Investigate existing multi-objective test problems and identifying their desirable features.
- Investigate a hyper-heuristic method based on heuristic selection with a deterministic move acceptance strategy.
- Investigate a hyper-heuristic method based on heuristic selection with a non-deterministic move acceptance strategy.
- Develop hyper-heuristic approaches to effectively and efficiently address multi-objective optimisation problems, demonstrating their effectiveness and efficiency on both benchmark test problems and a real-world problem.

In this thesis, a hyper-heuristic for multi-objective optimisation (HHMO) is investigated using three common multi-objective evolutionary algorithms NSGAII (Deb and Goel, 2001), SPEA2 (Zitzler et al., 2001) and MOGA (Fonseca and Fleming, 1993) as low level heuristics. The choice function acts as the selection mechanism. Four performance metrics; the algorithm effort (Tan et al., 2002), the ratio of non-dominated individuals (Tan et al., 2002), the uniform distribution of a non-dominated population (Srinivas and Deb, 1994), and the hypervoulme (Zitzler and Thiele, 1999) are used in the framework to serve as a feedback mechanism. The use of different move acceptance strategies; All-Moves, GDA (Dueck, 1993) and LA (Burke and Bykov, 2008), combined with a choice function is also investigated. The scope of this investigation is limited to continuous unconstrained problems. Combinatorial or discrete problems are not considered. The Walking Fish Group test suite (WFG) (Huband et al., 2006) is used as our benchmark dataset. The multi-objective design of vehicle crashworthiness problem (Liao et al., 2008) is used as a real-world application.

1.3 Overview of the Thesis

Our multi-objective hyper-heuristic framework addresses four main research areas; multi-objective evolutionary algorithms, hyper-heuristics, meta-heuristics and multi-objective test problems. Each area of research is discussed in this thesis. In chapter 2, a literature review of multi-objective evolutionary algorithms, hyper-heuristics and meta-heuristics are discussed. Chapter 2 also provides a description of well-known methodologies that address multi-objective optimisation and identify their strengths and weaknesses. A review of the scientific research on the subject is also presented. In chapter 3, the multi-objective test problems are identified and discussed. A description of the most common multi-objective test problems with an analysis of their features is given.

In this thesis, a hyper-heuristic for multi-objective optimisation is investigated through two methods: 1) Heuristic selection with a deterministic move acceptance strategy. 2) Heuristic selection with a non-deterministic move acceptance strategy. This investigation is based on three common multi-objective evolutionary algorithms; NSGAII (Deb and Goel, 2001), SPEA2 (Zitzler et al., 2001) and MOGA (Fonseca and Fleming, 1993) which act as low level heuristics, and the choice function is used as the selection method. In chapter 4, the details of the choice function based hyper-heuristic framework for multi-objective optimisation is described. Also a description of the learning feedback mechanism and the ranking scheme that is used within the hyper-heuristic framework is given.

Chapter 5 presents an online learning selection choice function all-moves based hyper-heuristic (HHMO_CF_AM). All-Moves is used as a deterministic move acceptance strategy. The proposed approach is tested and compared against the individual low level heuristics and other multi-objective hyper-heuristics from the scientific literature over the Walking Fish Group (WFG) test suite (Huband et al., 2006), a common benchmark for multi-objective optimisation.

An investigation of using non-deterministic move acceptance strategies, combined with a choice function as a heuristic selection method is provided in Chapters 6 and 7. We integrate *D* metric into the non-deterministic move acceptance criterion in order to convert the multi-objective optimisation to the single-objective optimisation without having to define values weights for the various objectives.

In Chapter 6, a selection choice function great deluge based hyper-heuristics (HHMO_CF_GDA) is proposed, developed and tested on the WFG test suite. The use of *D* metric within great deluge is discussed and described. Also an investigation of tuning the rain speed parameter (*UP*) of GDA is carried out.

In Chapter 7, a selection choice function late acceptance based hyper-heuristic (HHMO_CF_LA) is proposed. The use of *D* metric within late acceptance is presented. The comparison of the proposed approach and other multi-objective selection hyper-heuristics approaches, from Chapters 5 and 6, over the WFG test suite is investigated.

The three multi-objective hyper-heuristics, that are proposed in Chapters 5, 6 and 7, are applied to a real-world problem in Chapter 8. A description and formulation of the real-world multi-objective problem - the design of vehicle crashworthiness - is provided. A well-known multi-objective evolutionary algorithm and our three hyper-heuristics are compared and evaluated over four instances of this problem. Also an investigation of tuning the number of decision points for these hyper-heuristics is presented.

Finally, conclusions and recommendations for future work are presented in Chapter 9.

1.4 Contributions of the Thesis

The contributions of this thesis are as follows:

- The thesis investigates hyper-heuristics hybridised with multi-objective evolutionary algorithms (MOEAs) in order to tackle multi-objective problems. For the first time, a general design of a multi-objective hyper-heuristic framework based on a choice function is proposed in this thesis. The framework is flexible and could incorporate any meta-heuristic for multi-objective optimisation. Three online learning multi-objective selection choice function based hyper-heuristic are combined with three different move acceptance strategies (HHMO_CF_AM, HHMO_CF_GDA and HHMO_CF_LA). The first approach uses All-Moves as a deterministic move acceptance strategy and the other two approaches that are used GDA (Dueck, 1993) and LA (Burke and Bykov, 2008) respectively as additional non-deterministic move acceptance strategies. We show that those approaches, using a non-deterministic move acceptance strategy, outperform the approach that uses a deterministic move acceptance strategy on the test instances used in this thesis.

- This thesis presents a ranking scheme to measure the performance of low level heuristics, which also provides an online learning mechanism. The ranking scheme is simple and flexible and any number of low level heuristics can be incorporated.

- The thesis, for the first time, introduces *D* metric - a binary hypervolume (Zitzler, 1999) - integrating this idea into the non-deterministic move acceptance strategies (GDA and LA) in a multi-objective hyper-heuristic framework. The *D* metric is employed as the comparison tool in both move acceptance criteria in order to covert the multi-objective problem to a single-objective problem without having to define weights for each term.

- An application of a real-world problem on our multi-objective choice function based hyper-heuristics is investigated to see their performance on a real-world problem and measure the level of generality they we are able to achieve. It is shown that our methods produce better quality solutions when compared to other methods.

1.5 Academic Papers Produced

Maashi, M., Kendall, G., and Özcan, E. (2012). A choice function based hyper-heuristic for multi-objective optimisation. The 3rd Student Conference on Operational Research (SCOR 2012). April, Abstract.

Maashi, M., Kendall, G., and Özcan, E. (2012). A great deluge based learning hyper-heuristic for multi-objective optimisation. *The 54th Operation Research Annual Conference(OR54)*, September. Available at: http://www.theorsociety.com/DocumentRepository/Browse.aspx?CatID=3, (Accessed: 17th April 2013), Abstract.

Maashi, M., Özcan, E. and Kendall, G. (2014). A multi-objective hyper-heuristic based on choice function, *Expert Systems with Applications,* 41(9): 4475-4493.

Maashi, M., Kendall, G., and Özcan, E. (2014). Choice function based hyper-heuristics for multi-objective optimization, *Applied Soft Computing,* in review.

Maashi, M., Kendall, G., and Özcan, E. (2014). Comparison of Multi objective Hyper-heuristics on tri-objective WFG test problems. The 7th Saudi Students Scientific Conference(SSCUK2013), Edinburgh, UK, February, Abstract.

2 Literature Review

This chapter reviews three research areas; multi-objective evolutionary algorithms, meta-heuristics and hyper-heuristics.

2.1 Multi-objective Evolutionary Algorithms (MOEAs)

A multi-objective problem (MOP) comprises several objectives (two or more), which need to be minimised or maximised depending on the problem. A general definition of a MOP (Van Veldhuizen and Lamont, 2000) is:

An MOP minimises $\mathrm{F(x)} = (\mathrm{f_1(x)}, \ldots, \mathrm{f_k(x)})$ *subject to* $\mathrm{g_i(x)} \leq 0;\ \mathrm{i} = 1, \ldots, \mathrm{m}, \mathrm{x} \in \Omega$*. An MOP solution minimises the components of a vector* $\mathrm{F(x)}$*, where* x *is an n-dimensional decision variable vector* $(X = \mathrm{x_1}, \ldots, \mathrm{x_n})$ *from some universe Ω.*

An MOP consists of n decision variables, m constraints, and k objectives. The MOP's evaluation function, $\mathrm{F} : \Omega \rightarrow \Lambda$ maps decision variable vectors $(X = \mathrm{x_1}, \ldots, \mathrm{x_n})$ to vectors $(\mathrm{Y} = \mathrm{a_1}, \ldots, \mathrm{a_k})$. The mapping between the decision variable space and objective function space for multi-objective optimisation is represented in Figure 2.1.

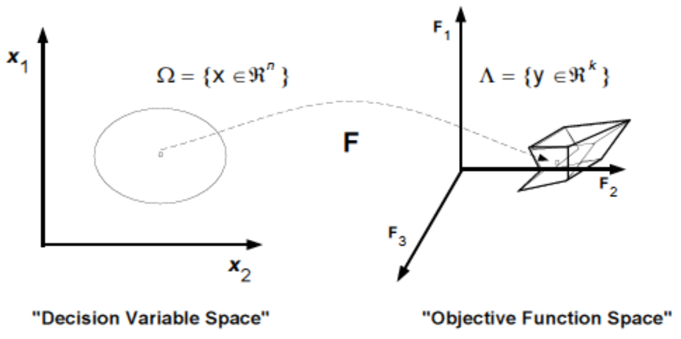

x_1
x_2
$\Omega = \{x \in \Re^n\}$
F
F_1
F_2
F_3
$\Lambda = \{y \in \Re^k\}$
"Decision Variable Space"
"Objective Function Space"

Figure 2.1: The mapping of Multi-objective spaces. Reprinted from (Van Veldhuizen and Lamont, 2000).

The relationship between a pair of objectives can be dependent and independent (Purshouse and Fleming, 2003a). Dependent objectives refers to objectives are harmony or conflict. If objectives are in conflict with each other, i.e. an improvement in one objective leads to deterioration in other, multi-objective optimisation techniques are required to solve this case (Tan, 2002). However, if two objectives are in harmony, i.e. an improvement in one objective leads naturally to improvement in the other, the objectives can be converted into a single-objective and tackled as a single optimisation problem (Tan, 2002). Independent objectives refer to the objectives are not affect each other. In this case, the objectives can be solved completely separately from each other (Purshouse and Fleming, 2003a).

Historically, a MOP was solved by converting the problem to a single-objective problem, due to the lack of multi-objective optimisation (MOO) methodologies to find a set of optimal solutions instead of a single optimum solution (Deb, 2005). However, many MOO techniques have now been proposed; so that it is possible to find the so-called Pareto-optimal solutions.

From a decision maker's perspective, multi-objective optimisation techniques are divided into three classes (Landa-Silva et al., 2004; Van Veldhuizen and Lamont, 2000; Coello et al., 2007a):

- **A *priori* approach** (decision-making and then a search)
  In this class, the objective preferences or weights are set by the decision maker prior to the search process. An example of this is aggregation-based approaches such as the weighted sum approach. The disadvantage of this approach is the requirement of the decision maker's experience to define the weights of the criteria values, which is usually a complex task, requiring a lot of experience (Petrovic and Bykov, 2003).

- **A *posteriori* approach** (a search and then decision-making)
  The search is conducted to find solutions for the objective functions. Following this, a decision process selects the most appropriate solutions (often involving a trade off). Multi-objective evolutionary optimisation (MOEA) techniques, whether non Pareto-based or Pareto-based, are examples of this class. MOEA techniques will be discussed later (see Section 2.1.3).

- ***Interactive* or *progressive approach*** (search and decision-making simultaneously)
  In this class, the preferences of the decision maker(s) are made and adjusted during the search process.

The scientific literature proposes three methods to evaluate the quality of the solutions for any MOP (Coello et al., 2007a). The first method is objective combination, which is the classical method to aggregate the objectives into a single scalar value by using a weighted function, after allocating weights to the objective criteria (Zitzler et al., 2000; Landa-Silva et al., 2004). The second method is where one objective is optimised, while the other objectives are defined as constraints. The drawback of this method is the difficulty in deciding which objective function should be optimised at any given point (Coello et al., 2007a). Pareto-based evaluation is the third method used to evaluate the quality of MOP solutions. In this method, all objectives are optimised simultaneously applying *Pareto dominance* concepts (see the next subsection) and using a vector for the values of all objectives and their solutions fitness. The two first methods are much simpler than the last one but they are more subjective and not straightforward. Furthermore, the last method is more methodical, more practical and less subjective compared to the others (Deb, 2005).

2.1.1 Pareto Dominance

The idea behind the *dominance* concept is to generate a preference between MOP solutions since there is no information regarding objective preference provided by the decision maker. This preference is used to compare the *dominance* between any two solutions (Coello et al., 2007a; Tan et al., 2002). A more formal definition of Pareto dominance (for minimisation case) is as follows (Coello et al., 2007a):

A vector $u = (u_1, \dots, u_k)$ is said to *dominate* another vector $v = (v_1, \dots, v_k)$ (denoted by $u \precsim v$) according to k objectives if and only if u is partially less than v, i.e., $\forall i \in \{1, \dots, k\}, u_i \leq v_i \wedge \exists\, i \in \{1, \dots, k\} : u_i < v_i$.

In other words, a solution is known as *non-dominated* if there is no other solution that is better than it in all objectives. All *non-dominated* solutions are also known as *the admissible set of the problem, non-inferior* or *the Pareto optimal* sets (Landa-Silva et al., 2004). The corresponding *Pareto optimal* set, with respect to the objective space, is known as the *non-dominated frontier*, *the trade-off surface* or *the Pareto optimal front* (Gandibleux and Ehrgott, 2005). In the rest of thesis, the terms *Pareto* *optimal set* (PS) and *Pareto optimal front* (POF) will be used. An example of Pareto optimal front in two objective space is shown in Figure 2.2.

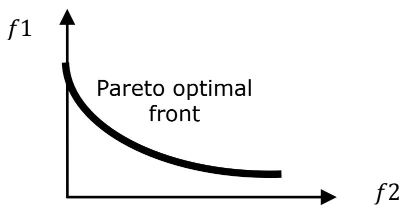
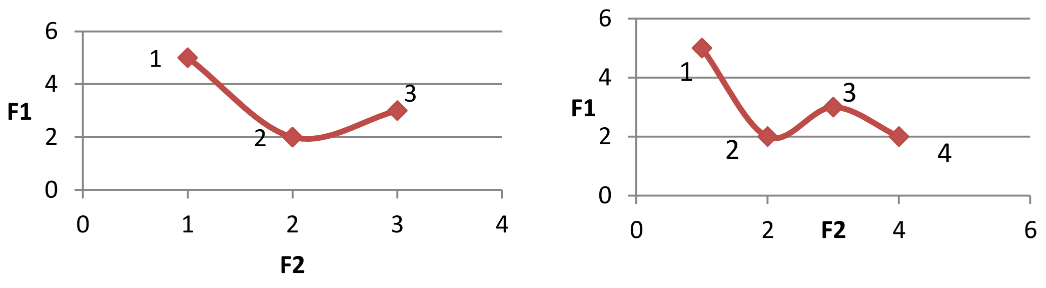


Figure 2.2: An example of Pareto optimal front in two objective space.

To further illustrate this idea, a solution x is known as *strictly dominates* if it is better than another solution x' in all objectives. While a solution x is known as *loosely dominates* if it is better than another solution x' in some objectives but it is equivalent to another solution x' at least in one objective (Landa-Silva et al., 2004). See Figure 2.3 for an illustration of these two concepts.



(a) (b)

Figure 2.3: Examples of strictly and loosely dominates solutions in the minimisation optimisation problem: in (a) the solution number 2 strictly dominates, in (b) the solutions numbers 2 and 4 are loosely dominates.

2.1.2 MOEAs Background

The idea of evolutionary algorithm(s) (EAs) is analogues to Darwin's principal of the biological evolution mechanism which adopted the concept of "survival-of-the-fittest" (Darwin, 1859). Many EA researchers would argue that evolutionary algorithm(s) are more suitable to deal with multi-objective optimisation problems (Deb and Goldberg, 1989; Bäck, 1996; Fonseca and Fleming, 1998; Deb, 2001; Coello et al., 2007a; Anderson et al., 2007; Zhang and Li, 2007; Miranda et al., 2010) because of their population-based nature, which means they can find Pareto optimal sets (trade-off solutions) in a single run, allowing a decision maker to select a suitable compromise solution. However, the task of an MOEA is not simply to find a Pareto optimal set that corresponds to the objectives of a particular problem. It is more complicated than that (Deb, 2005). MOEAs are multiple-objective in nature. Therefore, its task is also to minimise the distance of the Pareto optimal front and then maximise the extension of the Pareto optimal set (Zitzler et al., 2000).

According to Gandibleux and Ehrgott (2005), an EA comprises several components, which are the population, the evolutionary operators, (including crossover and mutation), the ranking method, the guiding method, the clustering method, the elite solutions archive, the fitness measurement and the penalty strategy. These components are discussed in more depth later in this section.

When applying an EA to a MOP, two important issues have to be considered (Zitzler et al., 2000): (i) Guiding the search towards the Pareto optimal set via an appropriate fitness assignment and selection strategies, and (ii) maintaining a diverse Pareto optimal set to obtain a well-distributed Pareto optimal front. It is worth noting that the EA may not find a diverse Pareto optimal set in some cases because of the Pareto optimal set's characteristics such as convexity, non-convexity, non-uniformity etc. (Zitzler et al., 2000). According to Coello et al. (2007a), a convex set is defined as that of all pairs of two points x and y in a set of points in n-dimensional space (see Figure 2.4 for examples of convexity, non-convexity sets).

Furthermore, three elements can determine the quality of the obtained Pareto optimal set (Landa-Silva et al., 2004; Zitzler et al., 2000):

i. The extent of the Pareto optimal set i.e. how many solutions are in the Pareto optimal set.

ii. The distance of the Pareto optimal front i.e. the closeness of the Pareto optimal front and the obtained front. Note that in some MOPs the Pareto optimal front is unknown.

iii. The distribution of the Pareto optimal front i.e. the depth of the coverage of the Pareto optimal front.

Examples of good and bad approximate Pareto fronts are shown in Figure 2.5.

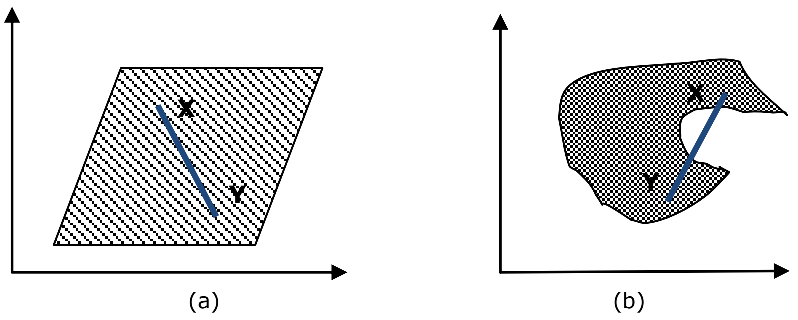


Figure 2.4: Examples of convexity, non-convexity sets. A set is convex if the line segment connecting any two points in the set lies entirely inside the set. in (a), an example of convex Pareto optimal front , in (b), an example of non-convex Pareto optimal front.

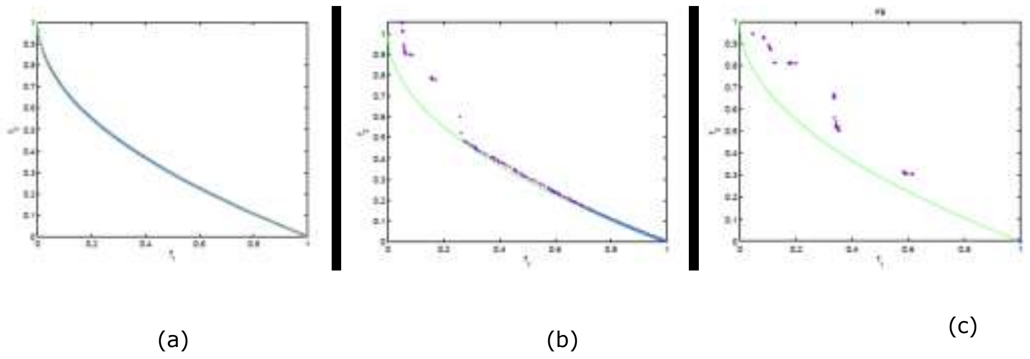


Figure 2.5: Examples of good and bad approximate Pareto fronts. In (a) a good example of approximate Pareto front, it is well-distributed over the Pareto optimal front. (b) and (c) are poor examples of approximate Pareto fronts. In (b) the distribution of approximate Pareto front not uniform and in (c) the approximate Pareto front is not well spread across the Pareto optimal front. Reprinted from (Li & Zhang, 2009).

With regard to the distribution of the Pareto optimal set, there are many techniques proposed in the literature to improve it (Burke et al., 2003a). These include:

- Tuning weights.
- Clustering or niching methods.
- Fitness sharing.
- Cellular structures and adaptive grids.
- Restricted mating sets.
- Relaxed forms of the dominance relation.

The tuning weights strategy is used to guide the search towards the target region of the Pareto optimal front by pushing the current solution towards that region. Examples of approaches that have employed this strategy are found in Czyzak and Jaszkiewicz (1998) and Ishibuchi et al. (2002).

Clustering (niching) methods aim to obtain a well-distributed Pareto optimal front via a fitness assignment based on the number of solutions on the given area (a measure of the crowding area). Examples of approaches that have employed this method are found in Lu and Yen (2002) and Socha and Kisiel-Dorohinicki (2002).

The fitness sharing technique aims to find a uniform (so-called equidistant) distribution of the Pareto optimal front (Van Veldhuizen and Lamont, 2000) by reducing the fitness of solutions in a particular area that are close together (Burke et al., 2003a). The fitness sharing method can be either phenotypic-based, with respect to the objective function space, or genotypic-based, with respect to the decision variable space (Horn et al., 1994). A brief introduction of phenotype and genotype terms can be found in the section on genetic algorithms (Section 2.2.9). The genotypic-based method is often employed by the operational research community because they are more concerned with the variable space in order to obtain a well-distributed Pareto optimal set (Benson and Sayin, 1997). However, setting appropriate values to the sharing parameter σ_{share} is not an easy task due to the necessity of a priori shape and the separation of the niche information for the problem at hand. Therefore, fitness sharing performance can be affected by the population size (Van Veldhuizen and Lamont, 2000).

Cellular structures and adaptive grid techniques aim to uniformly distribute the solutions over the Pareto optimal front. The micro genetic algorithm 2 (micro-GA2) (Pulido and Coello, 2003) is an example of an approach that has used this technique. In this approach, an online adaptation is made using Pareto ranking and an external memory.

The restricted mating method aims to reduce the probability of generating new, similar solutions by recombining these two solutions based on the degree of similarity between them (Burke et al., 2003a). However, this method is not always effective for some MOPs (Van Veldhuizen and Lamont, 2000).

Relaxed forms of the dominance relation aim to allow a small detriment in one or many objectives according to a relaxation factor, called ϵ-dominance, if a *large* improvement in other objective(s) is acquired. However, an improvement in the objectives' values can compensate for this relaxation (Coello et al., 2007a).

2.1.3 MOEA Methodologies

Schaffer (1985) proposed a non-Pareto based approach, namely the vector evaluated genetic algorithm (VEGA). It is considered as the first MOEA that has been formally proposed (Zitzler et al., 2000. In each generation, the population is divided into sub-populations based on the number of objectives. Each sub-population attempts to optimise a certain objective. Then these sub-populations are shuffled together and mutation and crossover operators are applied in order to generate the new population. The main drawback of VEGA is its inability to converge to non-convex areas of the Pareto optimal front.

Since 1985, various other MOEA techniques have been presented in the scientific literature. The most common ones being: MOGA (Fonseca and Fleming, 1993), NSGA (Srinivas and Deb, 1994), PESA (Knowles and Corne, 2000), SPEA (Zitzler and Thiele; 1999), MOMGA (Van Veldhuizen, 1999) and NPGA (Horn et al., 1994). However, several MOEA techniques are still emerging, while many existing MOEA techniques are being modified to create new versions. A survey of MOEAs can be found in Zhou et al. (2011) and Giagkiozis et al. (2013).

Tan et al. (2002) classifies MOEAs into three groups, with respect to their implementation strategies (selection methods and cost assignments). The three groups are naïve approaches, non-aggregation approaches and Pareto-based approaches. However, some other researchers classify MOEAs from a different perspective. Fonseca and Fleming (1995) classify MOEAs with respect to their algorithmic basis and Coello et al. (2007a) classify them with respect to the decision maker's viewpoint (see Section 2.1).

Pareto-based approaches are classical MOEAs. This section focuses on the Pareto-based approaches particularly MOGA, NGSA, SPEA, NPGA and

MOMOGA, because they are efficient and effective and they also incorporate much of the known MOEA theory (Van Veldhuizen and Lamont, 2000).

2.1.4 Multi-objective Genetic Algorithm (MOGA)

MOGA was proposed by Fonseca and Fleming (1993). In MOGA, the Pareto ranking scheme is used i.e. each solution in the current population is given a rank based on their dominance rank. All solutions in the Pareto optimal set have a rank of 1. A niche-formation method (fitness sharing) is employed in phenotypic-based cases to maintain a well-distributed population over the POF (Coello et al., 2007a). The average value of the fitness for all solutions that have the same rank is assigned to these solutions. A modified version of this algorithm has been proposed by Fonseca and Fleming (1998). This version employed restricted sharing between solutions that have the same rank and the distance between two solutions is computed and compared to the key sharing parameter σ_{share}. While MOGA is efficient and easy to implement, its fitness sharing method prevents two vectors that have the same value in the objective space existing simultaneously unless the fitness sharing is genotypic-based. The pseudo code of MOGA is shown in algorithm 1.

2.1.5 Non-dominated Sorting Genetic Algorithm (NSGA)

The original version of the Non-dominated Sorting Genetic Algorithm (NSGA) was proposed by Srinivas and Deb (1994). It employs a dominance depth based on the Pareto ranking scheme (Van Veldhuizen and Lamont, 2000). Moreover, a dummy fitness value, proportional to the population size, is used to classify all solutions in the Pareto optimal set. The fitness sharing method is quite similar to that used in MOGA but it is genotypic-based and applied to each level to maintain the diversity of the population and to obtain a uniform distribution of the POF (Zitzler et al., 2000). Once all solutions in the population are classified, the first Pareto front is assigned to the maximum fitness value. Therefore, the first Pareto front must have more copies than the other solutions in the population. A stochastic remainder selection strategy is employed for this purpose (Coello et al., 2007a). The complexity of NSGA is exhibited in its fitness sharing mechanism which assigns the fitness values to solutions in the current population. Knowles and Corne (2000), and many other researchers, have reported that NSGA has a poorer performance than MOGA. It is also more sensitive to the sharing parameter σ$_{share}$ than MOGA. However, some researchers point out that NGSA helps obtain a well-spread POF (Coello et al., 2007a). The pseudo code of NSGA is shown in algorithm 2.

Algorithm 1: MOGA algorithm

```
 1: procedure MOGA $(N', g, f_k(x))$ ▻ $N' member\ evolved\ g\ generations\ to\ solve\ f_k(x)$
 2:     Initialise Population P′
 3:     Evaluate Objective Values
 4:     Assign Rank Based on Pareto dominance
 5:     Compute Niche Count
 6:     Assign Linearly Scaled Fitness
 7:     Shared Fitness
 8:     for $i = 1$ to $g$ do
 9:         Selection via Stochastic Universal Sampling
10:         Single Point Crossover
11:         Mutation
12:         Evaluate Objective Values
13:         Assign Rank Based on Pareto dominance
14:         Compute Niche Count
15:         Assign Linearly Scaled Fitness
16:         Assign Shared Fitness
17:     end for
18: end procedure
```

Reprinted from (Coello et al., 2007a)

Algorithm 2: NSGA algorithm

```
 1: procedure NSGA $(N', g, f_i(x_k))$ ▻ $N' member\ evolved\ g\ generations\ to\ solve\ f_k(x)$
 2:     Initialise Population P′
 3:     Evaluate Objective Values
 4:     Assign Rank Based on Pareto dominance in Each $Wave$
 5:     Compute Niche Count
 6:     Assign Shared Fitness
 7:     for $i = 1$ to $g$ do
 8:         Selection via Stochastic Universal Sampling
 9:         Single Point Crossover
10:         Mutation
11:         Evaluate Objective Values
12:         Assign Rank Based on Pareto dominance in Each $Wave$
13:         Compute Niche Count
14:         Assign Shared Fitness
15:     end for
16: end procedure
```

Reprinted from Coello et al., 2007a)

A modified version of NSGA was proposed by Deb and Goel (2001). The modified version (NSGAII), is a non-explicit building block MOEA technique that incorporates the concept of elitism (Deb, 2005; Coello et al., 2007a). The solutions compete, then each solution is ranked and sorted based on its Pareto optimal level.

Genetic operators are applied to generate a new group of children who are then merged with parents in the population (Coello et al., 2007a). Furthermore, a niching method based on crowding distance is used during the selection process in order to maintain a diverse Pareto front (Zhang and Li, 2007). The pseudo code of NSGAII is shown in algorithm 3.

Algorithm 3: NSGAII algorithm

```
 1: procedure NSGAII $(N', g, f_i(x_k))$ ▻ $N'$ member evolved $g$ generations to solve $f_k(x)$
 2:     Initialise Population P′
 3:      Generate random population- size $N'$
 4:      Evaluate Objective Values
 5:     Assign Rank (level) Based on Pareto dominance - $sort$
 6:     Generate Child Population
 7:        Binary Tournament Selection
 8:        Recombination and Mutation
 9:     for $i = 1$ to $g$ do
10:         for each Parent and Child in Population do
11:            Assign Rank (level) Based on Pareto dominance - $sort$
12:            Generate sets of nondominated vectors along $PF_{known}$
13:            Loop (inside) by adding solutions to next generation starting from the
               $first$ front until $N'$ individuals found determine crowding distance
               between points on each front
14:         end for
15:         Select points (elitist) on the lower front (with lower rank) and are
            outside a crowding distance
16:         Create next generation
17:         Binary Tournament Selection
18:         Recombination and Mutation
19:     end for
20: end procedure
```

Reprinted from (Coello et al., 2007a)

Although NSGAII is more efficient than NSGA, it still has some drawbacks. It cannot simply generate an approximate set in some regions of the search space, particularly unpopulated regions (Coello and Pulido, 2001). In addition, NSGAII performs very badly when used for many-objectives optimisation (Purshouse and Fleming, 2007). As the number of objectives increase, the proportion of the space becomes lager and the solutions returned can be quite far from the Pareto optimal front. As result of this, the algorithm biased towards poor proximity solutions to the Pareto optimal front (Jaszkiewicz, 2001a; Purshouse and Fleming, 2007). Although, the algorithm could obtain very good spread across the Pareto optimal front, it faces difficult to achieve a good proximity.

2.1.6 Strength Pareto Evolutionary (SPEA)

The first version of Strength Pareto Evolutionary Algorithm (SPEA) was proposed by Zitzler and Thiele (1999). It integrates different desirable features in MOEAs which are (i) the use of the concept of dominance in the evaluation and selection process, (ii) the use of an external archive (secondary population) of the Pareto optimal set that was previously obtained, and (iii) the use of clustering and niching methods (Landa-Silva et al., 2004). In each generation, the Pareto optimal set is added to the secondary population. The solutions in the secondary population are used to evaluate the fitness values for the solution in the current population by summing the solutions' rank in the secondary population (Landa-Silva et al., 2004; Van Veldhuizen and Lamont, 2000).

The Pareto ranking scheme, based on the dominance count and rank, is employed, which means any distance measurement such as niche radius is not required (Coello et al., 2007a). The secondary population participates in the selection process, which leads to an increase in the population size. Therefore, a clustering technique, namely the average linkage method, is adopted to deal with this issue (Coello et al., 2007a). The pseudo code of SPEA is shown in algorithm 4.

Algorithm 4: SPEA algorithm

```
 1: procedure SPEA $(N', g, f_k(x))$ ▷ $N'$ member evolved $g$ generations to solve $f_k(x)$
 2:        Initialise Population P′
 3:        Create empty external set $E'$ ($|E'| < |P'|$)
 4:        for $i = 1$ to $g$ do
 5:              $E' = E' \cup ND(P')$ ▷ Copy members evaluatuing to be nondominated of P to E
 6:              $E' = ND(E)$ ▷ keep only member evaluating to nondominated vectors in E
 7:              Prune $E'$ (using clustering) if max capacity of $E'$ is exceeded
 8:              $\forall_{i \in P'}$ Evaluate $(P'_i)$ ▷ Evalute fitness for all members of $E'$ and P′
 9:              $\forall_{i \in E'}$ Evaluate $(E'_i)$
10:              $\mathcal{MP} \leftarrow \mathcal{T}(P' \cup E')$ ▷ Use binary tournament selection with
11:                                  ▷ replacement to select individuals from $E' + P'$
12:                                  ▷ (multiset union)until the mating pool is full
13:              Apply crossover and mutation on $\mathcal{MP}$
14:        end for
15: end procedure
```

Reprinted from (Coello et al., 2007a)

Despite SPEA generally having a good performance, it has some potential weak points in terms of fitness assignment, density estimation and archive truncation, which may affect SPEA's quality (Gandibleux and Ehrgott, 2005). To overcome these, an updated version called SPEA2 was proposed by Zitzler et al. (2001). SPEA2 differs from the previous version in three aspects: (i) it incorporates a fine-grained fitness assignment strategy which considers the number of individuals for each solution that dominates it and which it is dominated by, (ii) it uses a nearest neighbour density estimation technique in order to increase the efficiency of the search, and (iii) it improves the archive truncation method that guarantees the preservation of boundary points by replacing the average linkage method used in the previous version. The experimental results show that SPEA2 performs well in terms of diversity and distribution as the number of objectives increases. In addition, it significantly outperforms its predecessor SPEA. The pseudo code of SPEA2 is shown in algorithm 5.

Algorithm 5: SPEA2 algorithm

```
 1: procedure SPEA2 $(N', g, f_k(x))$
 2:     Initialise Population P′
 3:     Create empty external set E′ (|E′| < |P′|)
 4:     for $i = 1$ to $g$ do
 5:         Compute fitness of each individual in P′ and E′
 6:         Copy all individual evaluating to nondominated vectors P′ and E′ to E′
 7:         Use the truncation operator to remove elements from E when the capacity of
            the file has been extended
 8:         If the capacity of E′ has not been exceeded then use dominated individuals in P′ to fill E′
 9:         Perform binary tournament selection with replacement to fill the mating pool
10:         Apply crossover and mutation to the mating pool
11:     end for
12: end procedure
```

Reprinted from (Coello et al., 2007a)

2.1.7 Niched Pareto Genetic Algorithm (NPGA)

The Niched Pareto Genetic Algorithm (NPGA) was proposed by Horn et al. (1994). It uses the tournament selection scheme based on Pareto dominance ranking. Two randomly selected solutions are compared against ~10% of the population. If one of them is dominated while the other is not, the Pareto optimal set is selected. If both selected solutions are dominated or non-dominated, the fitness sharing scheme (equivalence class sharing) is employed to decide the results of the tournament. The pseudo code of NPGA is shown in algorithm 6 (Coello et al., 2007a).

NPGA has some difficulties in terms of the convergence towards the POF. To overcome this, an improved Niched Pareto Genetic Algorithm called NPGA2 was proposed by Erickson et al. (2001). In NPGA2, Pareto ranking and tournament selection schemes are used. NPGA2 evaluates the niche counts based on the next generation, instead of the current generation, using a continuously updated fitness sharing. The pseudo code of NPGA2 is shown in algorithm 7.

Algorithm 6: NPGA algorithm

```
 1: procedure NPGA  $(N', g, f_k(x)) \triangleright N' member\ evolved\ g\ generations\ to\ solve\ f_k(x)$
 2:     Initialise Population P'
 3:     Evaluate Objective Values
 4:     for $i = 1$ to $g$ do
 5:         Specialized Binary Tournament Selection
 6:         Begin
 7:             if Only Candidate 1 dominated then
 8:                 Select Candidate 2
 9:             else if Only Candidate 2 dominated then
10:                 Select Candidate 1
11:             else if Both are Dominated or Nondominated then
12:                 Perform specialized fitness sharing
13:                 Return Candidate with lower niche count
14:             end if
15:         End
16:         Single Point Crossover
17:         Mutation
18:         Evaluate Objective Values
19:     end for
20: end procedure
```

Reprinted from (Coello et al., 2007a)

Algorithm 7: NPGA2 algorithm

```
 1: procedure NPGA2  $(N', g, f_k(x)) \triangleright N' member\ evolved\ g\ generations\ to\ solve\ f_k(x)$
 2:     Initialise Population P'
 3:     Evaluate Objective Values
 4:     for $i = 1$ to $g$ do
 5:         Specialized Binary Tournament Selection using rank as domination
degree
 6:         Begin
 7:             if Only Candidate 1 dominated then
 8:                 Select Candidate 2
 9:             else if Only Candidate 2 dominated then
10:                 Select Candidate 1
11:             else if Both are Dominated or Nondominated then
12:                 Perform specialized fitness sharing
13:                 Return Candidate with lower niche count
14:             end if
15:         End
16:         Single Point Crossover
17:         Mutation
18:         Evaluate Objective Values
19:     end for
20: end procedure
```

Reprinted from (Coello et al., 2007a)

2.1.8 Multi-objective Messy Genetic Algorithm (MOMGA)

The Multi-objective Messy Genetic Algorithm (MOMGA) was proposed by Van Veldhuizen (1999). The algorithm is an extended version of the Messy Genetic Algorithm that is designed for a MOP. It is an explicit building block technique that comprises three stages: (i) the initialisation stage where building blocks of the population are generated in the partially enumerative initialisation process, (ii) the primordial stage where a tournament selection scheme is applied on the population, and finally (iii) the juxtapositional stage where a recombination of Messy GA operators are applied to build up the population.

The main advantage of MOMGA is that it is very powerful. However, it has some difficulties related to the population size. Its population size grows exponentially when the size of the building block increases. Many modified versions of MOMGA have been proposed. MOMGA-II described in Zydallis et al. (2001) is comprises three stages: the initialisation stage, the building filtering stage and the juxtapositional stage. The first two stages are different from MOMGA. MOMGA-III is the MOMGA recorded in an object-oriented form. The pseudo code of MOMGA and MOMGA-II are shown in algorithm 8 and algorithm 9 respectively.

Algorithm 8: MOMGA algorithm

```
 1: procedure MOMGA  $(N', g, f_k(x))$
 2:     for $i = 1$ to $epoch$ do
 3:     ▻ $PEI\ Phase$
 4:         Perform Partially Enumerative Initialisation
 5:         Evaluate each population member's fitness with respect to k templates
 6:         ▻ $Primordial\ Phase$
 7:         for $i = 1$ to $Max\ Pirmordial\ Generations$ do
 8:             Perform Tournament Thresholding Selection
 9:             if Appropriate number of generations accomplished then
10:                 Reduce Population Size
11:             end if
12:         end for
13:         ▻ $Juxtapositional\ Phase$
14:         for $i = 1$ to $Max\ Juxtapositional\ Generations$ do
15:             Cut-and-Slice
16:             Evaluate Each Population member's fitness with respect to k
                templates
17:             Perform Tournament Thresholding Selection and Fitness Sharing
18:             $P_{known}(t) = P_{current}(t) \cup P_{known}(t-1)$
19:         end for
20:         Update k templates  ▻ $Using\ best\ known\ value\ in\ each\ objective$
21:     end for
22: end procedure
```

Reprinted from (Coello et al., 2007a)

Algorithm 9: MOMGA-II algorithm

```
 1: procedure MOMGA-II  $(N', g, f_k(x))$
 2:     for $n = 1$ to $k$ do
 3:     Perform Probabilistically Complete Initialisation
 4:     Evaluate each population member's fitness with respect to k templates
 5:     ▻ Buliding Block Filtering BBF Phase
 6:       for $i = 1$ to Max Numbers of BBF Generations do
 7:           if BBF Required Based Off of Input Schedule then
 8:               Perform BBF
 9:           else Perform Tournament Thresholding Selection then
10:           end if
11:       end for
12:         ▻ Juxtapositional Phase
13:       for $i = 1$ to Max Juxtapositional Generations do
14:           Cut-and-Slice
15:               Evaluate Each Population member's fitness with respect to k
                  templates
16:               Perform Tournament Thresholding Selection and Fitness Sharing
17:               $P_{known}(t) = P_{current}(t) \cup P_{known}(t-1)$
18:         end for
19:         Update k competitive templates
▻ Using best known value in each objective
20:     end for
21: end procedure
```

Reprinted from (Coello et al., 2007a)

2.1.9 Overview of Many-objectives Optimisation

As the focus of this thesis is on multi-objective optimisation (two or three objectives), only a brief overview on many-objectives optimisation is presented in this section.

Recently, more attention is has been paid from EAs research to the many-objective optimisation (Purshouse and Fleming, 2004, 2007). In many-objective optimisation, the number of objectives is more than two and three. It might involve a large number of objectives. Unlike multi-objective optimisation, many-objectives optimisation faces some difficulties in terms diversity of solutions and obtaining an accurate approximation of the Pareto optimal front. These difficulties are known as dominance resistance and speciation (Purshouse and Fleming, 2004). Many-objectives optimisation also faces challenge when the objectives are in harmony. The traditional MOEA that designed for multi-objective optimisation cannot deal with many-objectives optimisation effectively. As the number of objectives increases, the proportion of non-dominated solution in the objective space becomes very large. So the selection pressure based on dominance is less effective which causes poor searching in seeking a good approximation of the Pareto front. To overcome this, some suggestions have been proposed (Adra and Fleming, 2011) such as modifying Pareto dominance by using different ranking schemes, use of goals and preference information to limit the search space, and employing different diversity management strategies. For more details see (Purshouse and Fleming, 2003b).

2.1.10 Overview of Performance Metrics for Multi-objective Optimisation

The comparison of the quality of solutions for multi-objective optimisation is more complex than single-objective problems. The number of non-dominated individuals should be maximised, the distance of the non-dominated front should be minimised, i.e. the resulting non-dominated set should be distributed uniformly as much as possible and converge well toward the POF.

In the scientific literature, many performance metrics have been proposed to measure different aspects of the quality and quantity of the resulting non-dominated set. See (Van Veldhuizen, 1999; Coello et al., 2007a). Some of these metrics require knowledge of the true Pareto front, whilst others do not. Some metrics, known as unary metrics, are designed to evaluate the performance of each algorithm independently of other algorithms. While other metrics, known as binary metrics, are designed to compare two non-dominated sets to each other. Deb (2001) classifies the performance metrics into three classes- metrics for convergence, metrics for diversity and metrics for both convergence and diversity. Knowles and Corne (2002) classifies the performance metrics based on the outperformance relations between two non-dominated sets into strong, weak and complete outperformance of one non-dominated set to another.

The quality of the obtained Pareto optimal set can be determined by three criteria (Landa-Silva et al., 2004; Zitzler et al., 2000):

(i) The extent of the Pareto optimal set i.e. how many solutions are in the Pareto optimal set? Ratio of non-dominated individuals (RNI) (Tan et al., 2002) and Error ratio (ER) (Van Veldhuizen, 1999) are examples of metrics that measure this criterion.

(ii) The distance of the Pareto optimal front, i.e. the closeness of the Pareto optimal front and the obtained non-dominated front. Examples of unary metrics that measure this criteria are the size of space covered metric (SSC or S-metirc) (Zitzler and Thiele, 1999), generational distance (GD) (Van Veldhuizen and Lamont, 1998b) and inverted generational distance (IGD) (Coello and Cruz Cortès, 2005). C metric and D metric (Zitzler, 1999) are examples of binary metrics that measure this criterion.

(iii) The distribution of the Pareto optimal set i.e. the depth of the coverage of the Pareto optimal front. Uniform distribution of a non-dominated population (UD) (Srinivas and Deb, 1994) and Spacing metric (Δ) (Deb and Jain, 2002) are examples of metrics that measure this criterion.

Beside the above three criteria, the computational time of the algorithm can be considered as a criterion to evaluate the performance of an optimiser, i.e. the time that an algorithm needs to obtain a non-dominated set should be minimised. Algorithm effort (AE) (Tan et al, 2002) is an example of metrics that measure this criterion.

Some of the performances that measure the above criteria are described as follows:

- **The size of space covered (SSC)**

SSC is a hypervolume presented by Zitzler and Thiele (1999). It is also known as the S-metric. This metric evaluates the size (volume) of the objective functions space covered by the solutions around the POF. Let X be a population and $x_i \in X$, the function $SSC(X)$ gives the volume enclosed by the union of the polytopes in the objective domain, where each polytope formed by the intersection of the following hyperplanes arising out of, along with the axes i.e. any point within the polytopes is always dominated by at least one x_i in X. SSC does not require knowledge of the true POF but it requires a reference point as the origin of the objective space. A lager value of SSC indicates better quality of non-dominated set which means a smaller distance to the true POF.

- **Uniform distribution of a non-dominated population (UD)**

UD is a unary metric presented by Srinivas and Deb (1994). It evaluates the distribution of non-dominated individuals over the POF. The distribution should be as uniform as possible to gain consistent gaps among neighbouring individuals in the population. Let X be a set of nondominated individual, *UD* defined as

$$UD(X) = \frac{1}{1+S_{nc}} \tag{2.1}$$

where S_{nc} is the standard deviation of niche count of the overall set of non-dominated set X. The UD metric does not require prior knowledge of the true POF. A lager value of UD indicates better quality of non-dominated set which means the non-dominated front is spread well along the POF.

- **Algorithm effort (AE)**

AE measures the computational effort of an algorithm to obtain the Pareto optimal set (Tan et al., 2002). It computes the ratio of the total number of function evaluations over a fixed period of simulation time. It ranges from [0,∞). A smaller value of AE indicates better performance which means the optimiser requires less time to obtain non-dominated solutions.

- **Ratio of non-dominated individuals (RNI)**

RNI is presented by Tan et al. (2002). It evaluates the fraction of non-dominated individuals $nondom\_inds$ in the population X. RNI defined as:

$$RNI(X) = \frac{nondom-inds}{size\ of\ X} \tag{2.2}$$

It ranges from [0,1]. If *RNI*=1, this indicates that all individuals for a given population are non-dominated and *RNI*=0 indicates that none of the individuals in the population are non-dominated. Although RNI gives an indication of the solution quality, it does not show how these solutions are good in terms of the diversity and the convergence towards the POF.

.

- **Generational distance (GD)**

GD is a unary metric presented by Van Veldhuizen and Lamont (1998b). It measures the distance (convergence) of the approximation non-dominated front A to the true POF B. GD defined as:

$$GD\ (A,B) \equiv \frac{1}{|A|}\left(\sum_{i=1}^{|A|} dist\ (a_i, B^p)\right)^{\frac{1}{p}} \qquad (2.3)$$

A smaller value of GD is more desirable and it indicates that the approximation non-dominated front is closer to the POF. The GD metric requires a prior knowledge of the true POF.

- **Inverted generational distance (IGD)**

IGD is a unary metric presented by Coello and Cruz Cortès (2005). It is opposite of the metric of GD. It measures the distance from a set of reference points (ideally the true POF) B to the approximation non-dominated set A. IGD defined as:

$$GD\ (A,B) \equiv \frac{1}{|B|}\left(\sum_{i=1}^{|B|} dist\ (b_i, A^p)\right)^{\frac{1}{p}} \qquad (2.4)$$

A smaller value of IGD is more desirable and it indicates that the approximation non-dominated front is closer to the POF.

- **Coverage difference of two sets (*D* metric)**

D metric (Zitzler, 1999) is an extended version of the hypervolume, also so-called the size of space covered metric (SSC) (Zitzler and Thiele, 1999). The SSC metric does not compute the coverage difference of two sets *A* and *B* when compared to each other, i.e it cannot be used to decide if one set entirely dominates the other. However, *D* metric computes the coverage difference of two non-dominated sets (initial/current non-dominated set) *A* and (candidate non-dominated set) *B* with respect to the objective space. $D(A,B)$ denotes the size of the space dominated by *A* and not dominated by *B* while $D(B,A)$ denotes the size of the space dominated by *B* and not dominated by *A*:

$$D(A,B) = SSC(A+B) - SSC(B) \tag{2.5}$$

$$D(B,A) = SSC(A+B) - SSC(A) \tag{2.6}$$

$D(A,B) < D(B,A)$ then *B* dominates *A*. In other words, the non-dominated front of *B* (front 2) is better than the non-dominated front of *A* (front 1) with respect to the *D* metric. An example of this is illustrated in Figure 2.6.

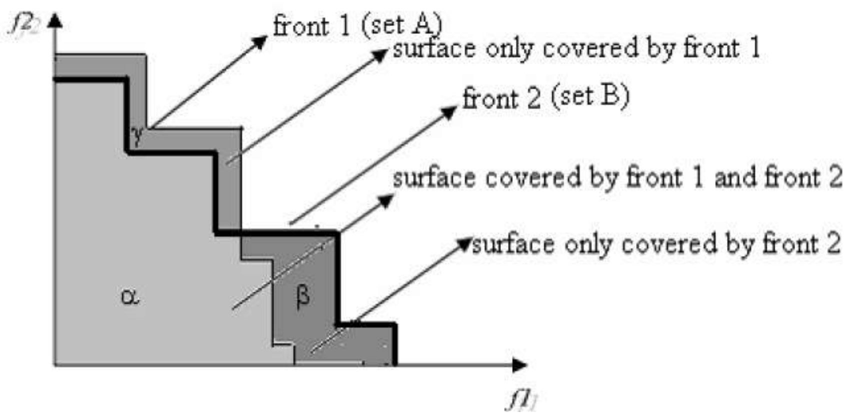

f2
front 1 (set A)
surface only covered by front 1
front 2 (set B)
surface covered by front 1 and front 2
surface only covered by front 2
γ
α
β
f1

Figure 2.6 :Example of *D* metric for two sets A and B and their fronts (front 1) and (front 2) respectively. Reprinted from (Grosan et al., 2003).

The relative size of the region (in the objective space) for a maximisation problem that is dominated by *A* and not dominated by *B* is suggested by Zitzler (1999):

$$D'(A,B) = \frac{D(A,B)}{V} \tag{2.7}$$

$$where\ V = \prod_{i=1}^{k} (f_i^{max} - f_i^{min}) \tag{2.8}$$

f_i^{max}, f_i^{min} represent the maximum, minimum values respectively for the objective f_i.

2.1.11 Studies on the Comparison of MOEAs

Generally, most MOEAs have common strategies that are employed in their search process. However, they are different in the way that they apply these strategies. MOGA and NSGA both apply the selection process after they have evaluated the rank values. However, MOGA classifies the solutions based on the ranking scheme using linear or exponential interpolation and applies the sharing scheme in the objective space, while NSGA uses dummy fitness values assigned to the solutions and applies the sharing scheme in the decision variable space (Van Veldhuizen and Lamont, 2000).

Furthermore, MOGA, NSGA, SPEA, NPGA and MOMGA incorporate fitness sharing schemes in order to obtain a uniform distribution of the POF. However, the mating restriction strategy is not always employed in any of them. A secondary population is also not always required in MOEAs, except in the case of SPEA. Van Veldhuizen and Lamont (2000) and Horn (1997) believe that any MOEA must use a secondary population for all Pareto optimal sets that have been found previously. Since MOEA(s) have a stochastic nature and the solutions are found in a particular generation, they are not necessarily found again in other generations. The second population helps to keep the desirable solutions in the population at the end of the search. In addition, some studies (Zitzler et al., 2000; Tan et al., 2002) report that elitism is a significant element used to enhance MOEA performance. For example, an NSGA with elitism performs as well as SPEA (Zitzler et al., 2000). The common strategies are employed in the search process for the five MOEAs- MOGA, NSGA, SPEA, NPGA and MOMGA- are presented in Table 2.1.

MOEA / **Strategies**	**MOGA**	**NSGA**	**SPEA**	**NPGA**	**MOMGA**
Fitness Sharing Schemes	√	√	√	√	√
Mating Restriction Strategy	√				
Secondary Population			√		
Elitism		√*	√		

Table 2.1: The common strategies are employed in the search process for the five MOEAs (MOGA, NSGA, SPEA, NPGA and MOMGA). * The elitism strategy is employed in the second version of NSGA (NSGAII) only.

In the scientific literature, some studies have compared MOEAs' performance and quality against each other. Zitzler et al. (2000) conducted a systematic comparison on eight algorithms including: five MOEAs (MOGA, NPGA, VEGA, NSGA and SPEA), two weighted-sum based approaches (SOEA and HLGA (Hajela and Lin, 1992)) and a random search strategy called RAND. These algorithms were run on six domain-independent test functions that provided sufficient complexity. The empirical results confirm that all MOEAs perform better than the RAND. Nevertheless, HLGA, NPGA and MOGA, in some cases, do not convergence well towards the POF. It is an interesting point that NSGA performs better than other none-elitist MOEAs in terms of distance and distribution along the POF, while SPEA has the best overall performance. In addition, the study demonstrates that NSGA with elitism performs similar to SPEA. Furthermore, the size of the population significantly affects the performance of EAs to cover the POF.

Another comparison study of MOEAs is provided by Tan et al. (2002). The study compares ten MOEAs which are VEGA, HLGA, NPGA, MOGA, NSGA, SPEA, MIMOGA (Murata & Ishibuchi, 1995), IMOEA (Khor et al. 2000), EMOEA (Khor et al. 2001) and a MOEA proposed by Tan et al. (1999). The ten MOEAs were run on four benchmark tests considering six performance measures- Ratio of non-dominated individuals (RNI), Uniform distribution of a non-dominated population (UD), Algorithm effort (AE), the hypervolume- Size of space covered (SSC), Noise sensitivity (NS) and Average best performance (ABP)- to examine the strength and weakness of each algorithm. Generally, the experimental results show that there is no existing algorithm that has the best performance in the all performance measures. In addition, the results confirm that elitism and sharing methods positively affect the performance of SPEA, MOEA, IMOEA and EMOEA in terms of distribution and convergence towards the POF. MIMOGA has relatively the lowest Algorithm effort (AE) in all benchmark tests while Tan's MOEA and IMOEA have the highest (better) Ratio of non-dominated (RNI) for all benchmark tests. HLGA and NPGA have relatively low noise sensitivity. MIMOGA, NPGA, MOGA and NSGA have moderate ABP performance while SPEA, MOEA, IMOEA and EMOEA perform well.

A comparison study for SPEA2, NSGAII and MOGA on ZDT4 and ZDT6 problems (Zitzler et al., 2000) was presented in Watanabe et al. (2002). With respect to the RNI metric, NSGAII has better performance than the others on

ZDT4. However, SPEA2 outperforms MOGA and NSGAII for the same metric on ZDT6. The authors concluded this study by stating that SPEA2 has an advantage with regard to its accuracy over NSGAII. While NSGAII is superior to SPEA2 in finding wide spread solutions.

Khare et al. (2003) conducted another comparative study for NSGAII, SPEA2 and PAES on four test problems (DTLZ1, DTLZ2, DTLZ3 and DTLZ6) with 2-8 objectives. Three performance metrics were used for convergence and diversity of the obtained non-dominated set and the running time. SPEA2 performs better than NSGAII in terms of convergence for a small number of objectives. However, both perform similarly for a higher number of objectives. SPEA2 and NSGAII have good performance with respect to the diversity, but they have some difficulties in the closeness of the obtained non-dominated set to the POF. In comparison, PAES (Liu et al., 2007) performs very well in converging to the POF but it fails in diversity and it requires a higher computational time as the number of objectives increases. However, NSGAII requires a less computational time compared to the others.

In Bradstreet et al. (2007) another comparative study between NSGAII and SPEA2 on the WFG test problems with 24 real values and a different scale of objectives. For two objectives, NSGAII is superior to SPEA2 on the WFG test problems with respect to the SSC metric. In contrast, SPEA2 outperforms NSGAII on all WFG problems expect WFG3 in three objectives with respect to the same metric.

We can note from two last studies that the number of objectives can affect the performance of an algorithm. SPEA2 works well with a high number of objectives for WFG and a low number of objectives for DTLZ. The opposite is true for NSGAII. We can also observe from these comparative studies that an algorithm can perform better than another algorithm with respect to a specific metric on a certain problem, while another algorithm performs better than another algorithm with respect to another metric for the same problem. Also an algorithm can perform differentially according to the number of objectives. All these observations could be an advantage when combining different algorithms in a hyper-heuristic framework for multi-objective optimisation to derive the strengths of the algorithms and avoid their weaknesses. These observations also supported by the No Free Lunch Theorem (Wolpert and Macready, 1997).

2.2 Meta-heuristics

The term *Meta-heuristic* was coined by Glover (1986). It refers to a general algorithmic search framework that is utilised for solving complex optimisation problems, instead of using classical approaches such as mathematical and dynamic programming (Bianchi et al., 2009). *Meta-heuristics* have the ability to find feasible solutions for problems of realistic size in reasonable computation time (Bianchi et al., 2009). Sörensen and Glover (2013) define Meta-heuristics as:

"A Meta-heuristic is a high-level problem-independent algorithmic framework that provides a set of guidelines or strategies to develop heuristic optimization algorithms".

It is worth noting that a problem-specific implementation of a heuristic optimisation algorithm is also referred to as a meta-heuristic. In the context of this thesis, meta-heuristics comprise a high level strategy that aims to explore the search space via the use of local search procedures in order to search for (approximate) optimal solutions and to escape from local optima. Moreover, some meta-heuristic techniques may employ learning mechanisms such as using memory in order to increase the efficacy of the search process (Blum and Roli, 2003).

In the scientific literature, common meta-heuristics such as simulated annealing (Kirkpatrick et al.,1983), tabu search (TS) (Glover,1986), genetic algorithms (GA) (Holland, 1975; Goldberg, 1989), ant colony optimisation (Dorigo et al., 1996), scatter search (Glover et al., 2000) and variable neighbourhood search (VNS) (Hansen and Mladenovic, 1999) have been successfully applied to solve different combinatorial optimisation problems (see (Corne et al., 1999; Voß et al., 1999; Glover and Kochenberger, 2003)). Further discussion of some meta-heuristics is presented in the following sections.

2.2.1 Algorithm Complexity and Problem Complexity

Algorithm complexity refers to the resources required of an algorithm that is required to solve a given problem (Garey and Johnson, 1979;

Rayward-Smith, 1986). The efficiency of an algorithm is measured in terms of execution time (the number of steps in the algorithm) and memory (the amount of memory that is needed to run the algorithm). The time complexity is as a function of the size of the input. In other words, it refers to the number of basic operations that are performed by an algorithm for its worst-case behaviour. The big O notation is used to describe the performance or complexity of an algorithm. The computational complexity of a problem is assessed by the time complexity of an algorithm that can be found to solve the problem efficiently (Garey and Johnson, 1979).

Optimisation problems can be divided into two major classes, P and NP. A P problem can be defined as an algorithm that can solve a problem in polynomial time. An NP problem can be defined as an algorithm that can solve a problem in *non-deterministic* polynomial time. For more details (see (Garey and Johnson, 1979, Rayward-Smith, 1986)). If there is a deterministic algorithm for a problem, a non-deterministic algorithm can be simply constructed for the problem, i.e. $P \subseteq NP$. This leads to, the most important open question in computational complexity theory, whether P=NP or P≠NP? To date, no efficient (polynomial) algorithms have been found for any NP problems, which supports the assumption that P≠NP, but this is still not proved. An example of an NP problem is the classic Travelling Salesmen Problems.

A special class of NP problems are NP-complete problems. These are the hardest class of problem in NP. The theory of NP-completeness was presented by Cook (1971). If P=NP then all NP-complete problems can be efficiently solved. All NP-complete problems could belong to P. However, NP-complete problems belong to the set NP−P.

Given the open P=NP question, exact algorithms cannot always be used to solve a given instance of an optimisation problem efficiently due to the time complexity being bounded by an exponential function (we may be able to solve small instances but this becomes impractical as the instance size increases). So, heuristic methods, or approximation algorithms, are generally more suitable to solve such problems since they can often produce near optimal solutions, or at least produce solutions of acceptable quality in reasonable computational time.

2.2.2 Intensification and Diversification

In the context of meta-heuristics, the concepts of *intensification* and *diversification* have a significant effect on the search behaviour. Intensification refers to exploiting the accumulated search experience whereas diversification refers to exploring the search space (Blum and Roli, 2003; Bianchi et al., 2009). A dynamic balance between these concepts is required in the search process. On the one hand we want to explore those areas of the search space, than just those currently providing good quality solutions (intensification). On other hand we also want to explore previously unvisited areas of the search space (diversification) (Blum and Roli, 2003). It is worth mentioning that the terms exploitation and exploration can sometimes be used instead of intensification and diversification (Blum and Roli, 2003). However, this may lead, in some cases, to different meanings. For example, exploitation and exploration may infer short term methods limited to randomness, whereas intensification and diversification may infer medium and long term methods based on the usage of memory. In meta-heuristics, the use of the local search strategy in simulated annealing is an example of intensification, while the use of tabu lists in tabu search is an example of diversification (Bianchi et al., 2009).

2.2.3 Meta-heuristics Classification

In the scientific literature, there are different points of view concerning the classification of meta-heuristics approaches. Glover and Laguna (1997) and Blum and Roli (2003) classify meta-heuristics into four main classes:

1) Nature-inspired and non-nature inspired methods
2) Dynamic and static objective functions
3) Memory usage and memory-less methods
4) Population-based and the single-point search methods

According to the origins of the search method, meta-heuristics divide into two groups; nature-inspired and non-nature inspired methods. Examples of these groups are genetic algorithms and tabu search respectively. Criticisms of this classification have been made for two reasons (Blum and Roli, 2003): (i) some hybrid meta-heuristics approaches cannot be categorised based on this classification. For example, memetic approaches that employ a local search mechanism and a genetic algorithm fit into both classes; and (ii) it is sometimes hard to classify an approach to one of the two categories. For example, tabu search belongs to the non nature-inspired category (memory-inspired), but it can be difficult to decide whether the use of memory belongs to the same class as well.

Another school of thought classifies meta-heuristics into two classes, dynamic and static objective functions. The first class changes the representation of the objective function during the search process. An example of this is Guided Local Search (GLS) (Voudouris and Tsang, 1999). On the contrary, the second class retains the representation of the objective function with no change.

Furthermore, memory usage and memory-less methods are important classifications of meta-heuristics according to the way that the algorithm makes use of search history. Memory usage can use short or long term memory. Short term memory keeps track of the moves and visited solutions whereas long memory accumulates synthetic parameters of the search. Memory-less methods usually tend to use the information to decide the next moves in the search process. Nowadays, memory is considered an essential element in successful meta-heuristic approaches (Blum and Roli, 2003).

The classification of population-based search and single-point approaches refers to the number of solutions that are maintained during the search process at each iteration (Glover and Laguna, 1997). In population-based meta-heuristics, a number of points (known as the population) are provided in order to evolve a new generation. Genetic algorithms, evolution strategies, ant colony optimisation, and scatter search are examples of population-based methods. In single-point search, only one solution is maintained during the search process. Single-point search based methods are also known as trajectory methods which share the same characteristics as a trajectory in the search space during the search process, and incorporate local search strategies (Blum and Roli, 2003). Tabu search, simulated annealing, iterated local search (Lourenco et al., 2003) and variable neighbourhood search are examples of single-point search methods. Since the population-based concept plays a significant role in hyper-heuristic for multi-objective optimisation (HHMO) that is proposed in this thesis, this classification of meta-heuristics is more suited to HHMO than the others.

2.2.4 Local Search

The key idea behind a local search algorithm is attempting to find the optimum (or an approximate) solution through exploring the neighborhoods of the current solution and comparing new solutions with the incumbent solution. If the new solution is better, then the current solution is replaced by the new one. The simplest form of local search is an iterative improvement algorithm. The algorithm starts with an initial solution and then explores the neighbourhood of that solution in order to find a better one. When a better solution is found, the current solution replaces it. This process is repeated until the current solution is better than all its neighborhood solutions.

In the context of local search, the strategy to improve a solution depends on the type of heuristic that is used in the algorithm (Lourenco at al., 2003). Random walk, simple descent and steepest descent are examples of local search heuristics. In random walk, a solution is selected randomly from the search space. This heuristic is usually combined with other methods and used as a diversification strategy. Simple descent is a typical local search strategy. It is also known as hill climbing. At each iteration, a random solution is selected. If the selected solution improves the objective value then it is accepted and the previous solution is replaced by it. Steepest descent is different from previous local search heuristics. This heuristic evaluates each solution in neighbourhood, and accepts the best solution that generates a better objective value. If there is no better objective value, the algorithm terminates. This method can be computationally expensive for large-sized neighbourhoods.

The main drawback of local search algorithms is that they can easily become trapped in a local optima, i.e. the solution is not necessarily the global optimal, because the search terminates once no better solutions can be found. An optimal solution (global) can be in some area of the search space that has not yet been explored (Focacci at al., 2003). To overcome this problem, some techniques have been presented that allow the algorithm to escape from local optima by accepting a worse solution (Aarts et al., 2005).

The local search algorithm terminates according to some conditions such as the number of iterations, elapsed CPU time or until there is no further improvement in the current solution for a given number of iterations.

2.2.5 Simulated Annealing

Simulated annealing (SA) is a search algorithm that was proposed by Kirkpatrick et al. (1983). It is considered the first meta-heuristic approach to use an explicit method which accepts worse solutions in order to escape from local optima (Henderson, 2003). Initially, SA was used to tackle combinatorial optimisation problems (often within the discrete problem domain). More recently, it has been extended to include continuous problems (Henderson, 2003). The concept of SA is based on the Metropolis algorithm for statistical mechanics developed by Metropolis et al. (1953). The Metropolis algorithm is a model for simulating the physical annealing process with solid materials like metals and glass (Bianchi et al., 2009). These materials are placed in a heat bath under a high temperature and then gradually cooled according to an appropriate cooling schedule until they reach a thermal equilibrium state (Dowsland, 1995 and Henderson, 2003).

In the context of meta-heuristics, SA incorporates thermodynamic behaviour into the local search strategy (Henderson, 2003), and the search process combines two local search heuristics; random walk and iterative improvement (Dowsland, 1995). It also employs a predefined neighbourhood structure of the search space (Bianchi et al., 2009). The algorithm starts with a high temperature and an initial solution. This solution can be either randomly selected or heuristically constructed (Blum and Roli, 2003). During the search process, the temperature is slowly decreased based on a cooling schedule (Dowsland, 1995). At each iteration, a solution of the neighbourhood is selected and evaluated and then compared with the incumbent solution. If it is better than the current one, it is accepted and replaces it to become the current solution. Otherwise, worse solutions are accepted according to a probabilistic function of temperature and the difference of objective function values for the new and current solutions (Dowsland, 1995 and Bianchi et al., 2009). The pseudo code of SA is shown in algorithm 10.

Two important issues can affect the performance of SA. Firstly, the choice of neighbourhood structure (Aarts and Korst, 1998), and secondly, the choice of a cooling schedule (Blum and Roli, 2003). There are two types of cooling schedule, static and dynamic schedules. In a static cooling schedule there is no change in the parameter values during the execution time. With a dynamic cooling schedule the parameters are adaptively changed during execution time (Aarts et al., 2005). It is generally not easy to choose an appropriate cooling schedule. In some cases in SA, the temperature is reduced and reaches a very low value. So, an increase of cost function values will be impossible and SA can lead to a local minimum. To overcome this problem, a reheating scheme may be used when a local minimum has been detected, in order to escape from it (Thanh and Anh, 2009). An example of SA's approaches that use the reheating scheme is simulated annealing with non-monotonic reheating (Osman, 1993). The key idea of this approach is that whenever the occurrence of a local minimum is detected, the use of reheating scheme aims to escape from it by doubling the temperature at which the best solution was obtained.

Algorithm 10: SA algorithm

```
 1: procedure SA
 2:        Initialise ($i_{start}$ , $C_0$ , $L_0$)
 3:        $k := 0$;
 4:        repeat
 5:          for $l = 1$ to $L_k$ do
 6:             Generate (j from $S_i$)
 7:             if $f$ (j) $\leq f(i)$ then $i := j$
 8:             else
 9:             if exp ($\frac{f(i)-f(j)}{C_k}$) $> random[0,1)$ then $i := j$
10:          end for
11:        $k := k + 1$;
12:        Calculate_Length($L_k$) ;
13:        Calculate_Control($C_k$) ;
14:     until stop_criterion
15:   end procedure
```

Reprinted from (Aarts et al., 2005)

Although SA is simple and flexible, a cooling schedule needs to be defined for each problem in order for the algorithm to work effectively (Hussin, 2005). Moreover, good quality cooling schedules (either static or dynamic schedules) which can find a global optimal can be particularly slow.

2.2.6 The Great Deluge Algorithm

The great deluge algorithm (GDA) is a meta-heuristic local search algorithm proposed by Dueck (1993). It is considered a reasonable alternative to other meta-heuristic algorithms such as simulated annealing (SA) (Kirkpatrick et al., 1983) and tabu search (TS) (Glover,1986), because of its simplicity and dependency on fewer parameters (Petrovic et al., 2007). GDA always accepts improving moves, while a worsening move is accepted only if it is better than a *threshold* (target improvement) at a given step. In a generic GDA approach, the threshold changes gradually over time, e.g. increases linearly.

In the case of a maximisation problem, the GDA algorithm starts with an initial water level, which is equivalent to the quality of the initial solution. The water level is increased gradually (usually linearly) at each iteration, during the search, according to a predefined rate referred to as *Rain Speed* (*UP*). A worsening solution is accepted if the quality of the solution is greater than or equal to the water level. This process is reversed for a minimisation problem. The algorithm terminates when there is no change in the solution quality within a predefined time or when the maximum number of iterations is exceeded. The pseudo code of a GDA (for maximisation problem) is shown in algorithm 11.

The main advantage of GDA is that it is simple and much easier to implement when compared to the other meta-heuristics, such as SA or evolutionary algorithms. Moreover, a better quality of solutions could be produced with a longer search time (Burke et al., 2004). GDA requires fewer input parameters; in fact it only has one parameter, rain speed (*UP*). The value of *UP* is usually a small fraction greater than 0, and less than 0.03 (Scott and Geldenhuysys, 2000). Dueck (1993) provided various recommendations regarding *UP*. For example, a suggestion is that *UP* value should be on average smaller than 1% of the average distance between the quality of the current solution and the water level. So the water level can be calculated for the j solution using:

$$LEVEL = LEVEL - UP\ (LEVEL - \ f(j)) \tag{2.9}$$

The value of *UP* can also be calculated based on the time allocated for search and defining upper/lower bounds of an estimated quality of solution (Petrovic. et al.,2000). However, both of those parameters depend on the problem dimensions and can affect the quality of final solution for a given problem (Telfar, 1995).

An extended GDA with reheating was proposed by McMullan and McCollum (2007). The idea is similar to the reheating scheme utilised in SA. The reheating (re-levelling in the GDA context) aims to widen the boundary condition, via improving the rain speed, in order to allow a worsening move to be accepted and avoid becoming trapped in a local optimum. If there is no improvement, water level is reset and re-levelling strategy is applied using a new rain speed value based on the number of total moves in the process.

Algorithm 11: GDA algorithm

```
 1: procedure GDA
 2:         Begin
 3:             Choose an initial configureuration $i$
 4:             Choose an initial rain speed $UP$ >0
 5:             Choose an initial water level  $LEVEL$>0  ⊳ $LEVEL = f(i)$
 6:             repeat
 7:                 Choose a neighbor $j \ni N(i,Q)$
 8:                     if $f(j) < LEVEL$ then
 9:                         $i := j$
10:                         $LEVEL = LEVEL + UP$
11:                     end if
12:       until (termination criteria are satisfied)
13:  end procedure
```

Reprinted from (Dueck,1993)

2.2.7 Tabu Search

Tabu search (TS) is a dynamic neighbourhood search technique (Stützle, 1999) that was first proposed by Glover (1986). It has been applied to many combinatorial optimisation problems (Gendreau, 2003; Hussin, 2005); for example, the Robust Tabu Search to the QAP problem (Taillard, 1991), and the Reactive Tabu Search to the MAXSAT problem (Battiti and Protasi, 1997) and to assignment problems (Dell'Amico et al., 1999).

Glover and Laguna (1997) define TS as follows:

> *"Tabu search is a meta-heuristic that guides a local heuristic search procedure to explore the solution space beyond local optimality."*

Tabu search is an advanced form of local search that employs the steepest descent heuristic and adaptive memory (Bianchi et al., 2009). The main aim of using memory and the search history is to avoid local optima and promote the exploration process (Blum and Roli, 2003; Gendreau, 2003). Furthermore, the key feature of TS is that it incorporates three specific concepts these being best improvement, tabu lists and aspiration criteria (Bianchi et al., 2009). Best improvement refers to always accepting a solution of the neighbourhood, whether it is better or worse than the current solution (Bianchi et al., 2009). However, that can result in the acceptance of solutions that were already previously accepted which may result in cycling. So, a short term memory that employs the tabu list concept is implemented to avoid this (Gendreau, 2003). Tabu lists prevent the recently visited solution being revisited by storing the attributes of these solutions. In the tabu list, some information about the search is stored to use it in the strategic guidance of the search (Bianchi et al., 2009). The length of the tabu list (so-called tabu tenure) is crucial for the performance of the algorithm. A small tabu tenure limits the search to small regions of the search space whereas a large tabu tenure results in the search exploring larger regions (Blum and Roli, 2003).

An aspiration criterion is a condition that has to be satisfied in order to remove a solution from the tabu list (Gendreau, 2003). One example of this is removing a specific solution from the tabu list, if it obtains a better objective value than the best value previously found (Gendreau, 2003). In the scientific literature, the aspiration criteria can be either time-dependent or time-independent. However, the choice of aspiration criteria is particularly critical because it can affect the search results (Gendreau, 2003). Other important control parameters that can affect the search results are tabu tenure and the structure of the neighbourhood.

The most popular termination conditions used for TS is the number of iterations, the CPU time or until no improvement in the object value has been found for a given number of iterations.

2.2.8 Late Acceptance

The late acceptance (LA) is recently proposed iterative search method proposed by Burke and Bykov (2008). It won an international competition to automatically solve the Magic Square problem (Burke and Bykov, 2012). It is based on the hill-climbing framework. The idea is delaying the comparison between the cost of current solution and previous solution. The comparison does not happen immediately, the cost of current solution is compared to the solution obtained after a number of moves to allow acceptance of worsening moves.

This method is very simple, easy to implement and yet powerful. It is also not sensitive to initialisation. It has a single input parameter, which is the length of array (L_{fa}) that contains the cost function values of the current solutions in the previous several iterations. In the context of LA, all values of the current cost function for the previous iterations are maintained in a list of a fixed length (L_{fa}), which is the only input parameter of LA. The last element of that list is compared with the cost value of candidate solution, in order to accept the move or reject it. If the candidate cost is better, or is equal to the last element, then the candidate solution is accepted and its cost is inserted into the beginning of the list, while the last element is removed from the end of the list. This process is repeated until it meets a stopping condition.

In order to avoid the shifting of the whole list at each iteration and reduce the processing time of LA, it is suggested to employ the *virtual* shifting of the list; the list beginning V is calculated by using:

$$V = i \ mod \ L_{fa} \tag{2.10}$$

where mod represents the remainder of integer division, i^{th} is the current iteration, L_{fa} the length a fitness array ($fa = f_0, f_1, f_2, \dots, f_{L_{fa}-1}$). At each iteration i^{th}, the candidate cost is compared with the value of C_v. Then after the acceptance procedure, the current cost is assigned to C_v, if it is accepted. The pseudo code of a LA is provided in algorithm 12.

At the beginning of the search, the fa can be filled by the initial cost value. In order to obtain the LA unique properties, it is intuitive that the length of the fitness array L_{fa} should be less than the number of iterations and equal to or greater than two. However, if L_{fa} is equal to one or zero, the LA performs as greedy hill-climbing (Burke and Bykov, 2008).

Algorithm 12: LA algorithm

```
1: procedure LA
2:     begin
3:         Produce an initial conFigureuration $s$
4:         Calculate initial cost function $C(s)$
5:         for all $k \in \{0, \dots, l_{fa} - 1\}$ dol 0 $C_k = C(s)$
6:         Assign the initial number of iterations $i = 0$
7:         repeat
8:             Construct a candidate solution $s*$
9:             Calculate its cost function $C(s*)$
10:                $V = i \ mod \ l_{fa}$
11:                if $C(s*) \leq C_v$ or $C(s*) \leq C(s)$ then
12:                        Accept candidate ($s = s*$)
13:                    Insert cost value into the list $C_v = C(s)$
14:                end if
15:            Increment the number of iterations $i = i + 1$
16:    until ( a chosen stopping condition)
17: end procedure
```

Reprinted from (Burke and Bykov, 2008)

2.2.9 Genetic Algorithms

In the scientific literature covering meta-heuristics, various population-based algorithms (so-called Evolutionary Computation (EC)) are presented, including genetic algorithms (GA) in (Fraser, 1957; Bremermann, 1958; Holland, 1975; Goldberg, 1989), evolution strategies (ES) by Rechenberg (1965), genetic programming by Koza (1992), ant colonies (AC) by Dorigo et al. (1996) and scatter search (SS) by Glover et al. (2000).

As described in the meta-heuristics classification in Section 2.2.3, population-based methods deal with a set of solutions (population) whereas single-point search methods such as simulated annealing and tabu search (see Sections 2.2.5 and 2.2.7) maintain only a single solution.

The ideas underpinning GAs were first proposed independently by Fraser (1957) and Bremermann (1958), although much of the important work can also be attributed to Holland (1975). Genetic algorithms (GA) are a stochastic search method, sometimes known as an evolutionary algorithm (EA). It is the most common population-based meta-heuristic (Sastry et al., 2005). It is based on the idea of "*Survival of the fittest*" presented by Darwin (1859). This natural concept of evolution is adopted as a search mechanism in all evolutionary computation algorithms (Reeves, 2003).

Unlike other meta-heuristics, the representation of solutions in GAs is quite different. The decision variables (chromosomes) that encode the solutions of problems are called *"genotypes",* whereas the candidate solutions of problems that represent the solutions themselves are called *"phenotypes"* or individuals (Goldberg, 1989; Reeves, 2003). In this context, a set of individuals (solutions) is called a population and each iteration during the search is called a generation. In addition, the solutions can be encoded as finite-length strings of binary or real numbers, or many other encodings (Goldberg and Rudnick, 1991).

A typical GA comprises six main stages as follows (Goldberg, 1989; Sastry et al., 2005):

1) Initialisation
2) Evaluation

3) Selection
4) Recombination (crossover)
5) Mutation
6) Replacement

Stages 2 to 6 are repeated in every generation until the algorithm is terminated by some criteria such as a maximum number of generations or a given number of fitness evaluations (Reeves, 2003). The pseudo code of a GA is shown in algorithm 13.

In the initialisation stage (step 2), an initial population of solutions is generated, typically randomly, in the search space. When the population is created, the fitness value of each solution in the population is evaluated by a fitness function (step 3) (Sastry et al., 2005). The solutions with higher fitness values are selected (step 5), usually stochastically, in order to separate the good solutions from the poorer ones (Sastry et al., 2005). The selection process can be accomplished by many proposed selection strategies including roulette-wheel selection, tournament selection, stochastic universal selection and ranking selection (Goldberg and Rudnick, 1991). For example, a solution with the highest fitness has the highest probability of being selected in roulette-wheel selection (Bianchi et al., 2009).

Algorithm13: The Genetic algorithm

```
 1: procedure GA
 2:        Initialise  $the\ population$
 3:        Evaluate  each $individuals$
 4:        repeat
 5:          Select  $individuals$ for $recombination$
 6:          for $gentention = 1$ to $Max_{genteation}$  do
 7:             Recombine $individuals$ generating new ones
 8:             Mutate the new $individuals$
 9:             Evaluate  each $individuals$
10:          Replace old $individuals$ with the new ones
11:          end for
12:      until (a  chosen stopping condition)
13:   end procedure
```

Reprinted from (Goldberg, 1989).

The choice of an appropriate selection strategy has a significant effect on the guidance of the search (Goldberg and Rudnick, 1991). After the selection stage, genetic operators (crossover and mutation) are applied to the selected solutions (steps 7 and 8) in order to create a new population (offspring) for the next generation. Crossover and mutation are executed in the recombination and mutation stages respectively. In the recombination stage (step 7), two or more solutions (parents) from the current generation are combined to generate hopefully better new solutions (children) for the next generation. Crossover can typically occur at one point or two points (known as one-point and two-point crossover) depending on the method of that is used (Goldberg et al., 1989). Many crossover operators have been proposed, for example, Partially Matched Crossover (PMX) and Simulated Binary Crossover (SBX). In the mutation stage (step 8), a change is made to an individual solution. The mutation in a GA is considered as a subsidiary operation that is used to increase the diversity of the population (Sastry et al., 2005). A typical example of mutation is bit-flip. The last stage (step 10) is replacement. The aim of this stage is to replace the old population with the new one for the next generation (Goldberg et al., 1989; Sastry et al., 2005). Examples of replacement methods are steady-state replacement, elitist replacement and generation-wise replacement.

In the context of GAs, the diversification strategy is accommodated by mutation, while intensification is accommodated by crossover operators and the selection process. However, mutation and population size have a critical impact on the scalability and performance of the algorithm. A small population size results in limited search exploration while a large population size results in long computational time (Reeves, 2003). Furthermore, too high a mutation rate can affect the diversity of the population (Goldberg, 1999; Reeves, 2003).

The main disadvantage of GA is the requirement of the fitness function. Some complex real-world problems such as structural optimisation problems cannot be tackled by GA, because it requires hours (sometimes days) of computational time for fitness evaluation (Reeves, 2003). Possible alternatives are to use approximated fitness or delta evaluation.

2.2.10 Other Meta-heuristic Algorithms

Many other meta-heuristic approaches have been proposed in the scientific literature, whether they belong to population-based approaches scatter search, or they belong to single-point search class such as variable neighbourhood search (VNS), iterated local search (ILS). Some may fit into both classes such as memetic algorithms (MA).

Ant colony optimisation (ACO) is a constructive meta-heuristic introduced by Dorigo et al. (1996). It simulates the behaviour of the real ants and the way they deposit pheromone to communicate with other ants. ACO use artificial pheromone trials as an indirect communication mechanism to distribute information among artificial ants (agents) in order to produce new solutions (Dorigo and Stützle, 2003).

Scatter search (SS) is a deterministic population-based alternatives for evolutionary algorithms that was introduced by Glover (1977). The key idea of this approach is to attempt to obtain better solutions through the construction of new solutions by linear combinations. Its strategy is based on the concept of combining decision rules and constraints in integer programming (Glover at al., 2003). Scatter search involves five major procedures: diversification generation, improvement, updating of a reference set, generation of subsets and the combining of solution procedures. For more details see (Glover at al., 2003).

Variable neighbourhood search (VNS) is a dynamic meta-heuristic approach proposed by Hansen and Mladenovic (1999). The algorithm provides several degrees of freedom to implement a wide range of variants (Blum and Roli, 2003) through dynamically changing neighbourhood structures. The basic design of VNS is different to other meta-heuristics. On some occasions only a few parameters may be needed, or none at all (Hansen, 2005). A standard VNS comprises three main phases: shaking, local search and move. The main aim of the shaking phase is to apply perturbation to a solution in order to make it a starting point for the local search (Hansen, 2005). In the context of VNS, the neighbourhoods are randomly chosen; then a solution of neighbourhoods is chosen (often randomly) as a starting point for the local search. Once the local search is terminated, the new solution that is found is compared with the initial solution. If it is better, the initial solution is replaced by the new one. Otherwise, a new iteration is started, including a new shaking phase with different neighbourhoods (Blum and Roli, 2003; Hansen, 2005).

Iterated local search (ILS) (Lourenco at el., 2003) is a stochastic local search method, and is a simple and powerful meta-heuristic approach (Martin et al., 1991; Stützle, 1999; Lourenco et al., 2003). It uses local search using an initial solution. Once the local optimum is found, the perturbation strategy is used in order to escape from it, then the local search restarts. A typical ILS includes three main processes: the choice of the initial solution, acceptance criteria and perturbation. The perturbation operators are particularly important (Blum and Roli, 2003); a larger perturbation makes the algorithm behave as a random restart local search whereas a small perturbation may result in an inability to escape from the local optima.

Memetic algorithms (MA) are a meta-heuristic that incorporates a local search strategy within an evolutionary algorithm. It was proposed by Moscato (1999). The most common memetic algorithms utilise genetic algorithm, carrying out a local search on each member of population in every generation. In the context of the memetic algorithm, the methods of individual learning usually include some knowledge of the problem at hand. These methods can be deterministic or stochastic. Moreover, many other studies have been presented in the literature in hybrid evolutionary algorithms and hill climbing strategies using multi-local searchers known as multimemes. Multimeme algorithms adaptively select from a set of local search procedures. Example of multimeme approaches can be found in Krasnogor and Smith, (2002) and Krasnogor (2002) and Krasnogor and Gustafson (2004).

2.2.11 Multi-objective Meta-heuristic

Meta-heuristics were originally designed to tackle single-objective optimisation problems. They have been extended to tackle multi-objective problems in a single run, without converting it to a single-objective problem, for example, by linearly weighting each objective. Multi-objective evolutionary algorithms such as MOGA (Fonseca and Fleming, 1993) and NSGA (Srinivas and Deb, 1994) (see Section 2.1.3) have had significant success in the multi-objective field due to their suitability to tackle such types of problems. However, a number of multi-objective meta-heuristics based on local search, such as simulated annealing and tabu search have been successfully applied to various multi-objective problems (Landa-Silva et al., 2004). The most common application that has been successfully tackled by multi-objective local search are multicriteria scheduling problems including flowshop scheduling problems and machine scheduling problems (see (Blazewicz et al., 1996; Baykasoglu et al., 1999 ; Gandibleux and Freville, 2000; Jaszkiewicz, 2001a)).

Thompson and Dowsland (1996) proposed a multi-phased simulated annealing algorithm to solve the examination timetabling problem. In this approach, the problem is formulated as a graph colouring problem and has two phases. The first phase aims to satisfy all the hard constraints (which is the first objective). The second phase aims to minimise the violations of soft constraints (which is the second objective). Moreover, Ulungu (1993) presented a multi-objective simulated annealing approach (MOSA). The author used simulated annealing to tackle a problem with multiple objectives (maybe two or three objectives). Another multi-objective simulated annealing based approach was proposed by Nam and Park (2000). The approach obtains good results when compared to MOEAs.

Gandibleux et al. (1997) presented the first multi-objective tabu search approach, the so-called MOTS. In this approach, special aspiration criteria, intensification and diversification strategies are designed for the multi-objective class, and a scalarising function and a reference point are used to enumerate a set of possible good solutions.

Jaszkiewicz (2001b) introduced a hybrid multi-objective approach based on genetic algorithm and local search. This is the so-called MOGLS. In this approach, local improvement heuristics are combined with crossover operators. Another hybrid method of a multi-objective approach has been proposed by Barichard and Hao (2002) known as the MOGTS, it is based on a combination of a genetic algorithm and tabu search. It is applied to the multi-constraint knapsack problem and showed competitive results. Li and Landa-Silva (2011) present an adaptive evolutionary multi-objective approach. Based on simulated annealing, it is called EMOSA. It incorporates simulated annealing and adapts weight vectors corresponding to various subproblems. The proposed approach is applied to the multi-objective knapsack problem and the multi-objective travelling salesman problem. It outperforms six multi-objective meta-heuristic algorithms from the literature.

2.3 Hyper-heuristics

Some real-world problems are complex. Due to their (often) NP-hard nature, researchers and practitioners frequently resort to problem tailored heuristics to obtain a reasonable solution in a reasonable amount of time. Hyper-heuristics are methodologies that operate on a search space of heuristics rather than directly searching the solution space for solving hard computational problems, with one of the key aims being to raise the level of generality. Many real-world computational problems have been solved successfully using state-of-the-art approaches and meta-heuristics techniques such as tabu search, genetic algorithms and simulated annealing (Burke et al., 2013). However, this success is often limited to a particular class of problem (or even particular problem instances) that has been solved using a specific implementation (Burke et al., 2013). The same implementation often cannot solve a new instance of the same problem unless the related parameters are properly tuned. Such methods are usually expensive to transfer to, and maintain, for new problems (Burke et al., 2013; Qu & Burke, 2009). Hyper-heuristics approaches have been proposed in order to raise the level of generality of search methodologies (Burke et al., 2010). Moreover, hyper-heuristics produce general search algorithms that are applicable for solving a wide range of the problems in different domains (Burke et al., 2010; Burke et al., 2013; Özcan et al., 2008; Ross, 2005).

In a hyper-heuristic approach, different heuristics (or heuristic components) can be selected, generated or combined to solve a given optimisation problem in an efficient way. In their simplest form hyper-heuristics are a search methodology that encompasses a *high level strategy* (which could be a meta-heuristic) that controls the search over a set of heuristics (heuristic components) rather than controlling a search over a direct representation of the solutions (Burke et al., 2010, 2013). In other words, hyper-heuristics performs as a "*heuristic scheduler*" within a set of low level heuristics using deterministic or non–deterministic methods; it is also sometimes termed *Move acceptance strategies* (Özcan et al., 2008).

Burke et al. (2013) define Hyper-heuristics as follows:

"A search method or learning mechanism for selecting or generating heuristics to solve computational search problems".

This definition will apply to the use of the term "hyper-heuristic" throughout this thesis. According to the recent definition of meta-heuristic, proposed by Sörensen and Glover (2013), we can define hyper-heuristics as a set of meta-heuristics.

To date, numerous hyper-heuristics papers have been published and several studies are being undertaken in this area of research. However, the notion of hyper-heuristics is not new. According to Burke et al. (2010, 2013) the idea of hyper-heuristics was first proposed in the early 1960s. Fisher and Thompson (1961, 1963) and Crowston et al. (1963) proposed the idea of a combination of dispatching rules (priority) to solve production scheduling problems so these combined rules were demonstrated to be superior to any rule taken in isolation. They also describe a method of combination by using "*probabilistic learning*" that simulated the mechanism of reinforcement learning in humans. Although computational search methodologies were still not mature at that time, the learning method proposed is similar to a stochastic local search algorithm performing in the space of scheduling rules' sequences (Burke et al. 2013). The main important conclusions from Fisher and Thompson's (1963) study are "*(1): an unbiased random combination of scheduling rules is better than any of them taken separately, and (2) learning is possible*".

The first time the term *hyper-heuristics* appeared was in a technical report by Denzinger et al. (1997) to illustrate a protocol that combines a range of Artificial Intelligence (AI) algorithms. Cowling et al (2000) used the term in a peer-reviewed conference paper to present the idea of the heuristic selection in scheduling a sales summit. The ideas in this paper was further developed and applied to scheduling problems in (Cowling et al, 2001, 2002a,b,c).

2.3.1 The Concept of Hyper-heuristics

In a hyper-heuristic approach, different heuristics can be selected, generated or combined to solve a given optimisation problem in an efficient way. Since each heuristic has its own strengths and weaknesses, one of the aims of hyper-heuristics is to automatically inform the algorithm by combining the strength of each heuristic and making up for the weaknesses of others. This process requires the incorporation of a learning mechanism into the algorithm to adaptively direct the search at each decision point for a particular state of the problem or the stage of search. It is obvious that the concept of hyper-heuristics has strong ties to Operational Research (OR) in terms of finding optimal or near-optimal solutions to computational search problems. It is also firmly linked to artificial intelligence (AI) in terms of machine learning methodologies (Burke et al., 2013). In the context of hyper-heuristics, learning knowledge control mechanisms plays a significant role in applying the appropriate low level heuristic at each decision point. Moreover, these mechanisms guide the search adaptively to improve the search methodologies (Burke et al., 2013).

The general framework of the hyper-heuristic is illustrated in Figure 2.7. Usually, in a hyper-heuristic framework, there is a clear separation between the high level hyper-heuristic approach (also referred to as strategy) and the set of low level heuristics or heuristic components. It is assumed that there is a domain barrier between them (Burke et al, 2003b). The purpose of domain barrier is to give the hyper-heuristics a higher level of abstraction. This also increases the level of generality of hyper-heuristics by being able to apply it to a new of problem without changing the framework. Only a set of problem-related heuristics are supplied.

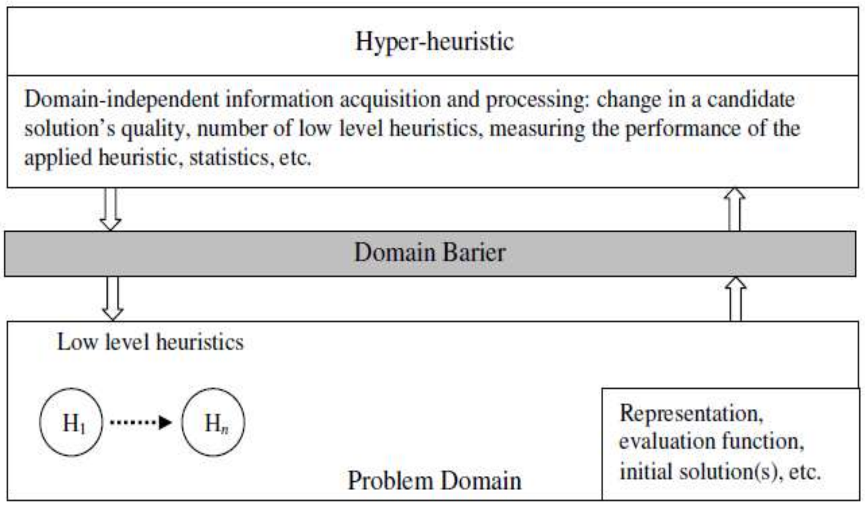

Hyper-heuristic
Domain-independent information acquisition and processing: change in a candidate solution's quality, number of low level heuristics, measuring the performance of the applied heuristic, statistics, etc.
Domain Barier
Low level heuristics
H_1
H_n
Problem Domain
Representation, evaluation function, initial solution(s), etc.

Figure 2.7: A generic hyper-heuristic framework. Reprinted from (Burke et al., 2003b).

The barrier allows only problem domain independent information to flow from the low level to the high level, such as the fitness/cost/penalty value (measured by an evaluation function, indicating the quality of a solution) (Hussin, 2005). Low level heuristics or heuristic components are the problem domain specific elements of a hyper-heuristic framework; hence they have access to any relevant information, such as candidate solution(s). The high level strategy can be a (meta-) heuristic or a learning mechanism (Burke et al.,2003b). The task of the high level strategy is to guide the search intelligently and adapt according to the success/failure of the low level heuristics or combinations of heuristic components during the search process, in order to enable the reuse of the same approach for solving different problems (Qu and Burke, 2009). Thus, the high level strategy does not change while both the low level heuristics or heuristic components and the evaluation function require changing when tackling a new problem.

2.3.2 Hyper-heuristics Classification

Two types of hyper-heuristic methodologies can be identified in the literature (Burke et al., 2013): (i) heuristic selection methodologies: (meta-)heuristics to choose (meta-)heuristics, and (ii) heuristic generation methodologies: (meta-)heuristics to generate new (meta-)heuristics from given components. Selection hyper-heuristics produce sequences of heuristics which lead to good quality solutions while generation hyper-heuristics produce new heuristics. For both hyper-heuristic methodologies, there are two recognized types of heuristics: (i) constructive heuristics which process a partial solution(s) and build a complete solution(s), (ii) perturbative heuristics which operate on complete solution(s). The notation of *constructive* and *perturbative* indicates how the search through the solution space is managed by the low level heuristics (Burke et al., 2013). However, a new direction of hybrid approaches of hyper-heuristics might include a combination of heuristic selection and heuristic generation methodologies, or a combination of construction and perturbation heuristics (Burke et al., 2010). The selection hyper-heuristics based on perturbative heuristics is the focus of this thesis. More on generation hyper-heuristics can be found in (Burke et al., 2013; Burke et al., 2010; Ross, 2005).

An orthogonal classification of hyper-heuristics is provided in Burke et al. (2010) (see Figure 2.8) depending on: (i) the nature of the heuristic search space and (ii) the source of feedback during the search process. Hyper-heuristics can be used to select or generate constructive or perturbative heuristics which determine the nature of the heuristic search space. However, a new research direction of hybrid hyper-heuristics might include a combination of heuristic selection and heuristic generation methodologies, or a combination of constructive and perturbative heuristics. A hyper-heuristic can employ no learning, online learning (getting feedback from the search process while solving an instance), or offline learning (getting feedback via training over a selected set of instances to be utilized for solving unseen instances). A hyper-heuristic which combines simple random heuristic selection with a method of accepting improving and equal quality moves is an example which uses a no learning approach (Özcan et al., 2008). If a hyper-heuristic incorporates a mechanism to adaptively guide the search process and enable the approach to make informed decisions about selecting or generating a low level heuristic, then it is a learning hyper-heuristic. Machine learning techniques are commonly used in hyper-heuristics. For example, reinforcement learning (based on reward/punishment) is employed as an online learning method for heuristic selection in hyper-heuristics (Cowling et al., 2002c). Genetic programming is frequently used as an offline learning hyper-heuristic which learns via the evolutionary process (Burke et al, 2009). In this thesis, we present an online learning selection hyper-heuristic based on perturbation heuristics (see Chapters 4-8).

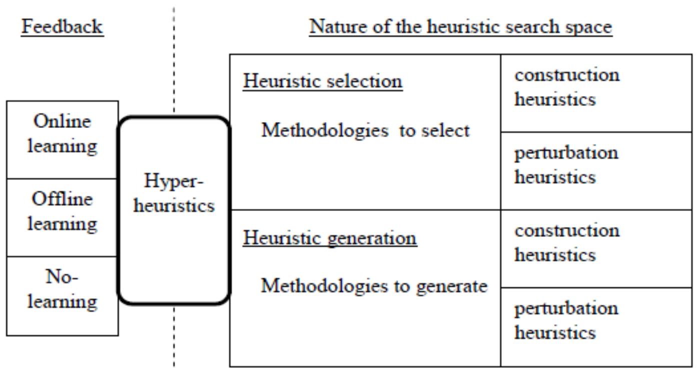


Figure 2.8: A classification of Hyper-heuristic. Reprinted from (Burke et al., 2010)

2.3.2.1 Selection Methodologies

In the context of selection hyper-heuristics, the search space involves a set of widely known and understood heuristics. These heuristics are decomposed into their primary components in order to solve a particular problem (Burke et al., 2010). Heuristic selection methodologies can be based on either *perturbative low level heuristics* or the *construction low level heuristics*.

Selection hyper-heuristics based on perturbation heuristics perform a search using two successive stages (Burke et al., 2013; Özcan et al, 2008): (meta-)heuristic selection and acceptance. An initial solution (a set of initial solutions) is iteratively improved using the low level (meta-)heuristics until some termination criteria is satisfied. During each iteration, the (meta-)heuristic selection decides which low level (meta-)heuristic will be executed next based on some criteria (perhaps randomly). After the selected (meta-)heuristic is applied to the current solution (a set of solutions), a decision is made whether to accept the new solution(s) or not using an acceptance method. The low level (meta-)heuristics in a selection hyper-heuristic framework are, in general, human designed heuristics which are fixed before the search starts.

A wide variety of selection hyper-heuristics based on perturbation heuristics are proposed using different heuristic selection and acceptance strategies in different domains: packing, vehicle routing, timetabling, channel assignment, component placement, personnel scheduling, planning and shelf space allocation (Burke et al., 2010). Most of the existing selection hyper-heuristics are based on perturbative low level heuristics, and favour single-point search.

More elaborate acceptance mechanisms have been introduced and there is a growing body of comparative studies which evaluate the performance of different heuristic selection and acceptance combinations (Burke et al., 2013). Cowling et al. (2002c) investigated the performance of different hyper-heuristics, combining different heuristic selection, with different move acceptance methods on a real world scheduling problem. Simple Random, Random Descent, Random Permutation, Random Permutation Descent, Greedy and Choice Function were introduced as heuristic selection methods. The authors utilised the following deterministic acceptance methods: All-Moves accepted and Only Improving moves accepted. The hyper-heuristic, combining Choice Function with All-Moves acceptance, performed the best. In Kendall et al. (2002) the choice function based hyper-heuristic was proposed and applied to nurse scheduling and sales summit scheduling. The study shows that the choice function hyper-heuristic is successful in making effective use of low level heuristics, due to its ability of learning the dynamics between the solution space and the low level heuristics to guide the search process towards better quality solutions. Burke et al. (2003c) proposed reinforcement learning with tabu search methodology in order to solve rostering problems. The approach is tested on two problems, concerning university timetabling and nurse rostering. The results were comparable to other state-of-the-art approaches.

Bai and Kendall (2005) proposed an approach using simulated annealing as a non-deterministic move acceptance strategy in order to apply it to a shelf space allocation problem. In this approach, improving solutions are always accepted, and worsening moves are accepted based on the Metropolis criterion. The results show that the Simple Random in hyper-heuristics simulated annealing based produces a better solution than Simple Random-Only Improving, Simple Random All-Moves, Greedy Only Improving and Choice Function All-Move. Dowsland et al. (2007) present simulated annealing with reheating as a non-deterministic move acceptance strategy in order to determine shipper sizes for storage and transportation in relation to a packing problem. Reinforcement-Learning with tabu search (RLTS) selection heuristic strategy is employed. The experimental data are generated based on actual data from a cosmetics company. The study's results show that simulated annealing with reheating and RLTS outperform the simpler local search strategy of Random Descent, Bai et al (2012) presents an extended hyper-heuristics framework based on the above studies. The proposed hyper-heuristic uses a reinforcement learning mechanism with a short term memory as a heuristic selection and SA with a reheating scheme as a move acceptance method. The proposed approach evaluated on different problem domains including nurse rostering, course timetabling and bin packing. Pisinger and Ropke (2007) developed an approach using simulated annealing based on a linear cooling rate as an acceptance strategy and applied it to five different vehicle routing problems. A large neighbourhood search framework is employed. The approach was tested over a wide range of vehicle routing benchmark instances. The experimental results confirm that the strategies used in the approach can produce better solutions over many instances.

In Özcan et al. (2008) the performance of seven different heuristic selection methods (Simple Random, Random Descent, Random Permutation, Random Permutation Descent, Greedy, Choice Function and Tabu Search) combined with five acceptance methods (All-Moves, Only Improving, Improving & Equal, Exponential Monte Carlo with Counter and Great Deluge) were investigated. The resultant hyper-heuristics were tested on fourteen benchmark functions against genetic and memetic algorithms. The empirical results confirmed the success of memetic algorithms over genetic algorithms and the performance of a choice function based hyper-heuristic was comparable to the performance of a memetic algorithm. Özcan et al. (2009)

used late acceptance as the non-deterministic move acceptance strategy with the best combination heuristic selection methods in order to solve exam timetabling problems. The results show that Simple Random combined with late acceptance outperforms Simple Random combined with other heuristic selection methods like Greedy, Choice Function, Reinforcement-Learning and Reinforcement-Learning Tabu Search. In Gibbs et al. (2011) the performance of different hyper-heuristics are compared with different components emphasising the influence of learning heuristic selection methods for solving a sports scheduling problem. The experimental result shows that the proposed approach is slightly better than the other approaches that use choice function as heuristic selection and great deluge as an acceptance criteria for solving a sports scheduling problem.

In Özcan and Kheiri (2011) a greedy heuristic selection strategy was presented which aims to determine low level heuristics with good performance based on the trade-off between the change (improvement) in the solution quality and the number of steps taken. This method performs well with respect to the competition hyper-heuristics on four problem domains. Berberoglu and Uyar (2011) compared the performance of combining twenty four learning and non-learning selection hyper-heuristics and seven mutational and hill-climbing heuristics. The study shows that Random Permutation Descent Only Improving performed the best on a short-term electrical power scheduling problem.

Recently, a wide empirical analysis was conducted in Burke et al. (2012) to compare many Monte Carlo based hyper-heuristics for examination timetabling. The experimental results show that choice function simulated annealing with reheating performs well. Another study was conducted by Bilgin et al. (2007) using a set of eight heuristic selection strategies (Simple Random, Random Gradient, Random Permutation, Random Permutation Gradient, Greedy, Choice function, Reinforcement Learning and Tabu Search) and five move acceptance strategies (All-Moves, Only Improving, Improving & Equal, Great Deluge Algorithm and Exponential Probability Function based on the computation time and a counter of consecutive (EMCQ)) which were tested on different timetabling benchmark problems. The study showed that there is no one strategy that dominates every other combination strategies. Vinkö and Izzo (2007) proposed a new distributed solver based on cooperatively standard versions of some stochastic solvers. The proposed approach outperforms the stand alone classical methods. Biazzini et al. (2009) presented a set of distributed hyper-heuristic based on an island model. The approach was compared against other hyper-heuristics over a set of real parameter optimisation problems.

Misir et al. (2011) proposed a move acceptance method referred to as Adaptive Iteration Limited List-based Threshold Acceptance (AILLA). The proposed move acceptance is compared to other move acceptance strategies including LA, SA, GDA and Improving & Equal. All the comparison methods are combined with Simple Random heuristic selection. The results show that AILLA and Late acceptance outperform the others. Misir et al. (2012) extend the work by presenting a heuristic selection based on heuristic dynamic learning. The approach that combined AILLA and this heuristic selection was the winner of the CHeSC competition. Drake et al. (2012) presented a hyper-heuristic employing a variant of the choice function as a heuristic selection with a simple new initialisation and update scheme. Demeester et al. (2012) presented Simple Random based hyper-heuristics using different move acceptance strategies including Improving or Equal, GDA, SA, LA and Steepest Descent Late Acceptance for examination timetabling. The experimental results show that the Simple Random SA hyper-heuristic performs the best over exam benchmark datasets. A recent study on hyper-heuristics for continuous optimisation in dynamic environments is proposed by Kiraz et al. (2011). The proposed approach uses a parameterised Gaussian mutation to create different low level heuristics. The experimental results show that the choice function Improving & Equal hyper-heuristic outperforms Simple Random Improving & Equal hyper-heuristic.

There are a number of hyper-heuristic approaches in the literature based on evolutionary algorithms. An example of a hyper-heuristic approach based on a genetic algorithm can be in Dorndorf and Pesch (1995). Although the term of hyper-heuristic was not created by the authors, the concept of a hyper-heuristic was employed through a probabilistic learning strategy based on the principles of evolution. The proposed algorithm was applied to solving job shop scheduling problems. Another example of a hyper-heuristic approach based on a genetic algorithm can be found in Hart et al. (1998). This approach was used for handling a set of low level heuristics to solve a chicken catching and transportation problem. Ross and Marin-Blazquez (2005) also present a messy genetic algorithm hyper-heuristic based on graph colouring heuristics to tackle class and exam timetabling problems. The key idea behind their approach was to devise an algorithm to find the problem states through a set of labelled points which it refers to as a heuristic. The approach produces fast problem-solving algorithms compared with other existing algorithms. A messy genetic algorithm is employed by Terashima Marın et al. (2008) for class and exam timetabling problems. The proposed offline approach shows an ability to produce good quality solutions. Cobos et al (2011) present different variants of evolutionary approaches under a multi-point based search framework. The proposed approaches are tested on different combinations of heuristic selection and move acceptance methods on the document clustering problems. Grobler et al. (2012) presents a hybrid approach on a set of meta-heuristics including a genetic algorithm, particle swarm optimisation variants, CMA-ES and differential evolution that was combined with local search under a multi-point hyper-heuristic framework.

In selection hyper-heuristics based on construction heuristics, an initial solution is empty and is then built up gradually via the use of constructive heuristics. A complete solution is obtained at the end of the run (Burke et al., 2013). Various construction low level heuristic based approaches are proposed using a variety of high level strategies in different domains. According to a recent survey conducted by Burke et al. (2013), the popular high level strategy used in heuristic selection based on constructive heuristics are hill-climbing, genetic algorithms, tabu search, iterated local search, variable neighbourhood search, fuzzy systems, case-based reasoning, classifier systems, messy genetic algorithms and scatter search. In addition, the common domains which have applied heuristic selection based on constructive heuristics are packing, vehicle routing, timetabling and production scheduling and constraint satisfaction domains.

2.3.2.2 Generation Methodologies

As the focus of this thesis is on selection hyper-heuristic methodologies, only a brief review of the literature on generation hyper-heuristic methodologies is presented in this section.

Generation hyper-heuristic methodologies refer to generating new heuristics from the basic components of existing heuristics, known as a set of building blocks. Generation hyper-heuristic methodologies can be based on either construction low level heuristics or perturbation low level heuristics (Burke et al., 2010). In the context of heuristic generation, the search space involves a set of basic components of known and understood heuristics. A new heuristic is generated to produce the solution for a given problem at the end of a run (Burke et al., 2013). Although generation hyper-heuristics aim to generate a new heuristic automatically, using building blocks of heuristics, the heuristic components still have to be designed by humans (Burke et al., 2010). Generation hyper-heuristics have some advantages in terms of their ability to produce a better solution than human-designed heuristics. In addition, they require less (human) time and human resources to be applied to various problem instances. However, they do have some disadvantages in the short term regarding their computational cost (Burke et al., 2013).

The most common generation hyper-heuristics are genetic programming-based. That is because of this methodology's suitability to represent heuristics in an effective way (Jakobovic et al., 2007). Genetic programming (Koza, 1992) is an evolutionary computation technique that operates on a population of computer programs. However, other generation approaches have been developed based on the squeaky wheel optimisation methodology (Joslin and Clements,1999; Aickelin et al.,2009; Burke and Newall, 2004).

Various automated generation hyper-heuristic approaches have been proposed in different problem domains including the travelling salesman problem, satisfiability testing (SAT), production scheduling, cutting and packing, boolean satisfiability, binary decision diagrams, constraint satisfaction and compiler optimisation. An example of the generation approach for boolean satisfiability is presented by Bader-El-Den and Poli (2007). The approach uses genetic programming to produce local search heuristics. In their study, traditional crossover and mutation operators are used within various heuristic generation methodologies. Burke et al. (2006; 2007a,b) propose the generation of construction heuristics using genetic programming. The proposed approach was evaluated on the bin packing problems. The study confirms the applicability of the approach to these types of problems. Further, the results show that the approach can beat the human-designed heuristics in terms of its ability to perform better over a new instance of a particular class of heuristic rather than new instance of a different class. Keller and Poli (2007) propose a genetic-programming hyper-heuristic approach to evolve local search heuristics in order to solve travelling salesman problems. The evolved heuristics show good performance over two TSP benchmark instances. Pillyay (2008) conducted an analysis of performance in genetic programming systems under three representations (alternative encodings, fixed length and variable length) for the examination timetabling problem. The study shows that fixed length representation perform badly. In Özcan and Parkes (2011) present a hyper-heuristic for generating constructive heuristics (policies). The whole process is formulated as a tuning process where there are many parameters in the system.

In term of multi-objective approaches, Tay and Ho (2008) propose a genetic programming hyper-heuristic approach to evolve dispatching rules to solve multi-objective job-shop problems in production scheduling. The dispatching rules generated performed better than single dispatching rules. Allen et al. (2009) present an empirical study comparing the quality of genetic programming heuristics and human heuristics that were designed to solve 3D knapsack packing problems. The results indicate that the generated heuristics perform competitively against state-of–the-art approaches. Kumar et al. (2009) propose multi-objective genetic programming for the minimum spanning tree problem. The diameter and cost of the trees serve as objectives. In this approach, the evolved heuristics are used to generate the Pareto optimal front and produced good quality solutions compared with existing heuristics.

This section has reviewed the papers in the area of research that are particularly relevant to this thesis. For comprehensive surveys and examples see (Burke et al, 2013). Some valuable guidelines for implementing a hyper-heuristic approach can also be found in Ross (2005).

2.3.3 Multi-objective Hyper-heuristics Approaches

Hyper-heuristics have recently seen an increase in attention from researchers. Although many hyper-heuristics papers have been published, they are still mainly limited to single-objective optimisation. The hyper-heuristics for multi-objective optimisation problems is a new area of research in Evolutionary Computation and Operational Research (Özcan et al., 2008; Burke et al., 2013). To date, few studies, have been identified that deal with hyper-heuristics for multi-objective problems (see Table 2.2).

The first approach (Burke et al., 2003a) is a multi-objective hyper-heuristic based on tabu search (TSRoulette Wheel). The key feature of this paper lies in choosing a suitable heuristic at each iteration to tackle the problem at hand by using tabu search as a high-level search strategy. The proposed approach was applied to space allocation and timetabling problems and produced results with acceptable solution quality. An adaptive multi-method (multi-point) search called AMALGAM is proposed in Vrugt and Robinson (2007). It employs multiple search algorithms; NSGAII (Deb and Goel, 2001), PSO (Kennedy, 2001), AMS (Haario et al., 2001), and DE (Storn and Price, 1997) simultaneously using the concepts of multi-method search and adaptive offspring creation. AMALGAM is applied to a number of continuous multi-objective test problems and it was superior to other methods. It was also applied to solve a number of water resource problems and it yielded very good solutions (Raad et al., 2010; Zhang et al., 2010}. Veerapen et al. (2009) present a multi-objective hyper-heuristic approach comprising two phases: the first phase aims to produce an efficient Pareto front (this may be of low quality based on the density), while the second phase aims to deal with a given problem in a flexible way to drive a subset of the population to the desired Pareto front. This approach was evaluated on the multi-objective travelling salesman problems with eleven low level heuristics. It is compared to other multi-objective approaches from the literature which reveals that the proposed approach generates good quality results but future work is still needed to improve the methodology. Len et al. (2009) propose a hypervolume-based hyper-heuristic for a dynamic-mapped multi-objective island-based model. The proposed method shows its superiority when compared to the contribution based hyper-heuristic and other standard parallel models over the WFG test problems (Huband et al., 2006). A new hyper-heuristic based on the multi-objective evolutionary algorithm NSGAII (Deb and Goel, 2001) is proposed in Gomez and Terashima-Marỉn (2010). The main idea of this method is in producing the final Pareto-optimal set, through a learning process that evolves combinations of condition-action rules based on NSGAII. The proposed method was tested on many instances of irregular 2D cutting stock benchmark problems and produced promising results. A multi-strategy ensemble multi-objective evolutionary algorithm called MS-MOEA for dynamic optimization is proposed in Wang and Li (2010). It combines different strategies including a memory strategy and genetic and differential operators to adaptively create offspring and achieve fast convergence speed. Experimental results show that MS-

MOEA is able to obtain promising results. In McClymont and Keedwell (2011) an online selection hyper-heuristic, Markov chain based, (MCHH) is investigated. The Markov chain guides the selection of heuristics and applies online reinforcement learning to adapt transition weights between heuristics. In MCHH, hybrid meta-heuristics and Evolution Strategies were incorporated and applied to the DTLZ test (Deb et al., 2002) problems and compared to a (1+1) evolution strategy meta-heuristic, a random hyper-heuristic and TSRoulette Wheel (Burke et al., 2003a). The comparison shows the efficacy of the proposed approach in terms of Pareto convergence and learning ability to select good heuristic combinations. Further work is needed in terms of diversity preserving mechanisms. The MCHH was applied to the WFG test problems (Huband et al., 2006), the experiments shows efficacy of the method but future work is still needed in terms of acceptance strategies to improve the search (McClymont and Keedwell, 2011). The MCHH has also been applied to real-world water distribution networks design problems and produced competitive results (McClymont et al., 2013). In Miranda et al. (2010) and Armas et al. (2011), a hyper-heuristic-based codification is proposed for solving strip packing and cutting stock problems with two objectives that maximise the total profit and minimise the total number of cuts. Experimental results show that the proposed hyper-heuristic outperforms single heuristics. In Furtuna et al. (2012) a multi-objective hyper-heuristic for the design and optimisation of a stacked neural network is proposed. The proposed approach is based on NSGAII combined with a local search algorithm (Quasi-Newton algorithm). Rafique (2012) presented a multi-objective hyper-heuristic optimisation scheme for engineering system design problems. A genetic algorithm, simulated annealing and particle swarm optimisation are used as low-level heuristics. Vázquez-Rodríguez and Petrovic (2013) proposed a multi-indicator hyper-heuristic for multi-objective optimisation. This was approach based on multiple rank indicators that taken from NSGAII (Deb & Goel, 2001), IBEA (Zitzler and Künzli, 2004) and SPEA2 (Zitzler et al., 2001). Len et al. (2009) proposed a hypervolume-based hyper-heuristic for a dynamic-mapped multi-objective island-based model. Bai et al. (2013) proposed a multiple neighbourhood hyper-heuristic for two-dimensional shelf space allocation problem. The proposed hyper-heuristic was based on a simulated annealing algorithm. Kumari et al. (2013) present a multi-objective hyper-heuristic genetic algorithm (MHypGA) for the solution of Multi-objective Software Module Clustering Problem. In MHypGA, different methods of selection, crossover and mutation operations of genetic algorithms are incorporated as a low level heuristics.

None of the above studies have used multi-objective evolutionary algorithms (MOEAs), with the exception of Gomez and Terashima-Marín (2010), Vrugt and Robinson (2007) and Rafique (2012) and no continuous and standard multi-objective test problems studied, except in except in McClymont and Keedwell (2011), Vrugt and Robinson (2007), Len et al. (2009) and Vázquez-Rodríguez and Petrovic (2013). Moreover, none of the previous hyper-heuristics make use of the components specifically designed for multi-objective optimisation that we introduce in this thesis.

Component name	**Application domain/ test problems**	**Reference(s)**
Tabu search	Space allocation, timetabling Travelling salesman problems	Burke et al. (2003) Veerapen et al. (2009)
Markov chain, evolution strategy	Real-world water distribution networks design /DTLZ, WFG	McClymont and Keedwell (2011)
NSGAII	Irregular 2D cutting stock Strip packing and Cutting stock	Gomez and Terashima-Marín (2010) de Armas et al. (2011) and Miranda et al.(2010)
NSGAII, quasi-Newton algorithm	Stacked neural network	Furtuna et al. (2012)
Number of operations from NSGAII, SPEA2 and IBEA	A number of continuous multi-objective test problems	Vázquez-Rodríguez and Petrovic (2013)
Number of selection, crossover and mutation operations of evolutionary algorithms	Software module clustering	Kumari et al. (2013)
Hypervolume	Dynamic-mapped island-based model/ WFG	Len et al. (2009)
Particle swarm optimisation, adaptive metropolis algorithm, differential evolution	Water resource problems/ a number of continuous multiobjective test problems	Vrugt and Robinson(2007), Raad et al. (2010) and Zhang et al. (2010)
Memory strategy, genetic and differential operators	Dynamic optimization problems/a number of continuous multi-objective test problems	Wang and Li (2010)
Genetic algorithm, simulated annealing, particle swarm optimization	Engineering system design problems/a number of classical multi-objective test problems	Rafique (2012)
Simulated annealing	Shelf space allocation	Bai et al. (2013)

Table 2.2: Heuristic components and application domains of hyper-heuristics for multi-objective optimisation.

2.3.4 Multi-objective Selection Hyper-heuristics versus Hybrid Methods for Multi-objective Optimisation

According to Ke Tang in Vrugt et al. (2010), the idea of combining multiple algorithms is not new at all, and can be traced back to 1980s. In the context of multi-objective and evolutionary computation, many methods are presented utilising this idea, such as adaptive multi-method algorithms (Vrugt and Robinson, 2007) and multi-strategy ensemble algorithms (Wang and Li, 2010).

The adaptive multi-method/strategy ensemble algorithms rely on running multiple algorithms (such as MOEAs or evolution strategies) simultaneously and adaptively creating the offspring. Both methods are closely similar to selection hyper-heuristics for multi-objective optimisation problems. Other researchers would argue that the adaptive multi-method/strategy ensemble algorithms are hyper-heuristic methods. According to Burke et al. (2013), the hyper-heuristics defined in Section 2.3. It is hard to classify the adaptive multi-method/strategy ensemble algorithms as selection or generation hyper-heuristics. However, we cannot remove them from the umbrella of hyper-heuristics, as they are combining different heuristics/ meta- heuristics. These methods are similar to the multi-objective selection hyper-heuristic methods in term of the incorporation of different algorithms. However, they are different from selection hyper-heuristics in their concept. Selection hyper-heuristic rely on two concepts: a selection mechanism and an acceptance move strategy. Both concepts are not adopted in the adaptive multi-method/strategy ensemble algorithms. Moreover, multiple heuristic/meta-heuristics run concurrently in the adaptive multi-method/strategy ensemble algorithms. Each heuristic/meta-heuristics produce a different population of offsprings, and then all produced offsprings are evaluated to evolve a new population of offspring by an adaptive creation offspring strategy. In multi-objective selection hyper-heuristics, a sequence of heurstic/meta-heuristic is executed during the search, i.e. one heurstic/meta-heuristic is selected and applied at each stage (iteration/decision point) of the search. The high level strategy in hyper-heuristics evaluates the performance of a set of heurstic/meta-heuristic in order to improve the population of solutions.

In this thesis, a new online learning selection hyper-heuristic framework which supports multi-point search and cooperative low level meta-heuristics for multi-objective optimisation is proposed. Further details of this hyper-heuristic framework are discussed in Chapter 4.

2.4 Summary

Our multi-objective hyper-heuristic framework that is investigated in this thesis addresses multi-objective evolutionary algorithms, hyper-heuristics, meta-heuristics research areas. This chapter has reviewed previous research work for those areas.

In this chapter also we provided a description of well-known methodologies that address multi-objective optimisation and identify their strengths and weaknesses. In this chapter, we reviewed the previous research for multi-objective hyper-heuristics. None of the previous hyper-heuristics make use of the components particularly designed for multi-objective optimisation that we introduce in this thesis.

Several multi-objective test problems have been proposed in the literature; for example, real-world problems, combinatorial optimisation problems, discrete or integer-based problems, noisy problems, dynamic problems, and problems with side constraints. In the next chapter, we presents an overview and discusses the multi-objective optimisation test problems in specifically the continuous unconstrained problems.

3 Multi-objective Optimisation Test Problems

A multi-objective problem (MOP) comprises several objectives (two or more), which need to be minimised or maximised depending on the problem. Each objective has some measure as to the quality of the solution. It is essential that MOEA algorithms are tested over a number of problems in order to have a clear perception of their strengths and weaknesses. To accomplish this effectively, it is crucial to first develop a strong understanding and undertake a precise analysis of the test problems at hand. In the MOEAs literature, several multi-objective test problems have been proposed; for example, continuous problems, combinatorial optimisation problems, discrete or integer-based problems, noisy problems, dynamic problems, problems with side constraints and even real-world problems (see Coello et al., 2007b). However, some of the multi-objective test problems do not fully examine the characteristics of EAs. Also they sometimes have defects in their design such as not being scalable in terms of parameters/objectives, or only being suitable for simple algorithms (Huband et al., 2006). In order to fully understand the features of test problems for multi-objective optimisation, some important definitions and test problems features are described in this chapter.

3.1 Definitions of the Test Problems' Features

Pareto one-to-one or Pareto many-to-one:

If the mapping between the Pareto optimal set and the Pareto optimal front (the fitness landscape) is one-to-one. The problem, in this case, is called *Pareto one-to-one.* Otherwise, if the fitness landscape is many-to-one the problem is called *Pareto many-to-one* (see Figure 3.1).

Flat regions:

A characteristic of many-to-one fitness landscapes is when a connected open subset of parameter space maps to a singleton. The problem with flat *regions* occurs when a tiny perturbation of the parameters in regions do not change the objective values.

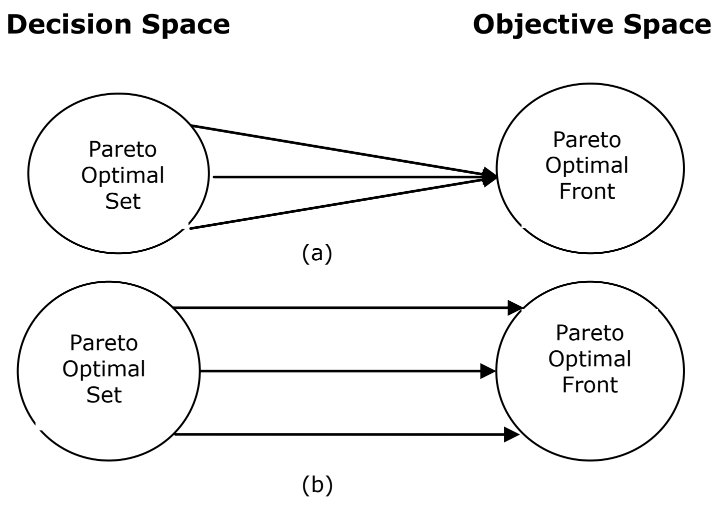


Figure 3.1: Examples of the mapping between the Pareto optimal set and the Pareto optimal front (the fitness landscape). In (a) Pareto many-to-one, (b) Pareto one-to-one.

Modality*:*

A problem can be described as a *multimodal problem* if it has a multimodal objective which includes multiple local optima in the objective space. Otherwise, if there is only a single optimum with the objective function, the problem is described as a *unimodal* problem (see Figure 3.2).

Deception:

Deception is a special case of multimodality. If the objective function has at least two optima (a true optimum and a deceptive optimum) then it can be called a *deceptive objective*, and the problem which consists of this objective function can be called a *deceptive* problem (see Figure 3.2).

Bias:

In the fitness landscape, an evenly distributed sample of parameter vectors in the search space maps to an evenly distributed set of objective vectors in the fitness space, but the mapping from the Pareto optimal set to the Pareto optimal front can be biased if significant variation occurs in distribution. The variation is known as *bias*. It is worth mentioning that bias has a significant effect on the convergence speed toward the Pareto optimal front (POF).

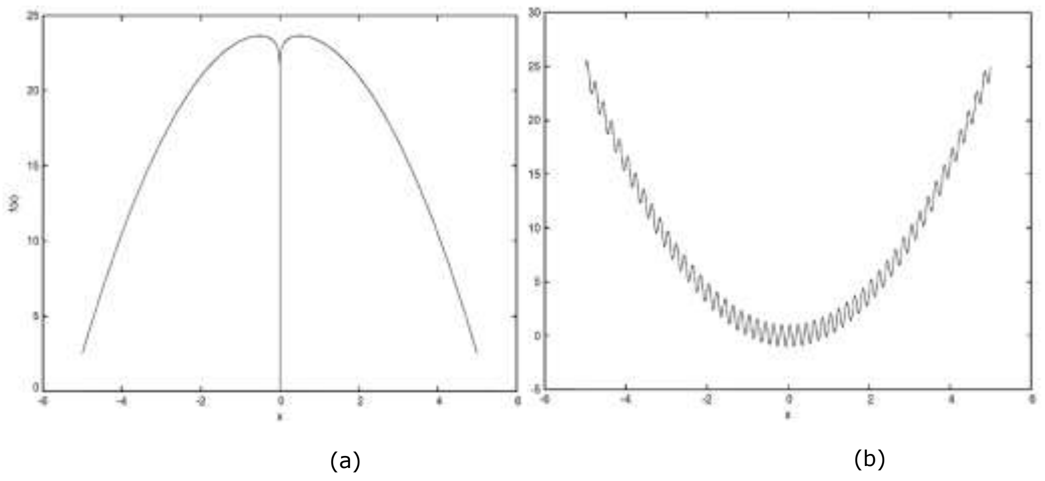


Figure 3.2: Examples of deceptive and multimodal objectives. In (a) a deceptive multimodal objective. (b) a nondeceptive multimodal objective. Reprinted from (Huband et al., 2006).

Separability:

It refers to the parameter dependencies. If every objective of a problem is *separable*, then it is a *separable* problem. Otherwise, it is a *nonseparable* problem.

Pareto Front Geometries:

The geometry of the Pareto optimal front can be *convex*, *concave*, *degenerate*, *connect*, *discrete.* It can also consist of different geometry fronts which are known as *mixed* fronts (see Figure 3.3). A front is a convex front, if it covers its convex hull. In contrast, if it is covered by its convex hull, it is a concave front. A linear front is one that is both concave and convex. A degenerate front is a front that is less than the number of dimensions in the objective space such as front that only a point in two objectives and a line segment in a three objective problem. (Huband et al., 2006). A connected front is often referred to as continuous while a disconnected front is often referred to as discontinuous. A mixed front is one with consists of strictly convex, strictly concave, or linear front.

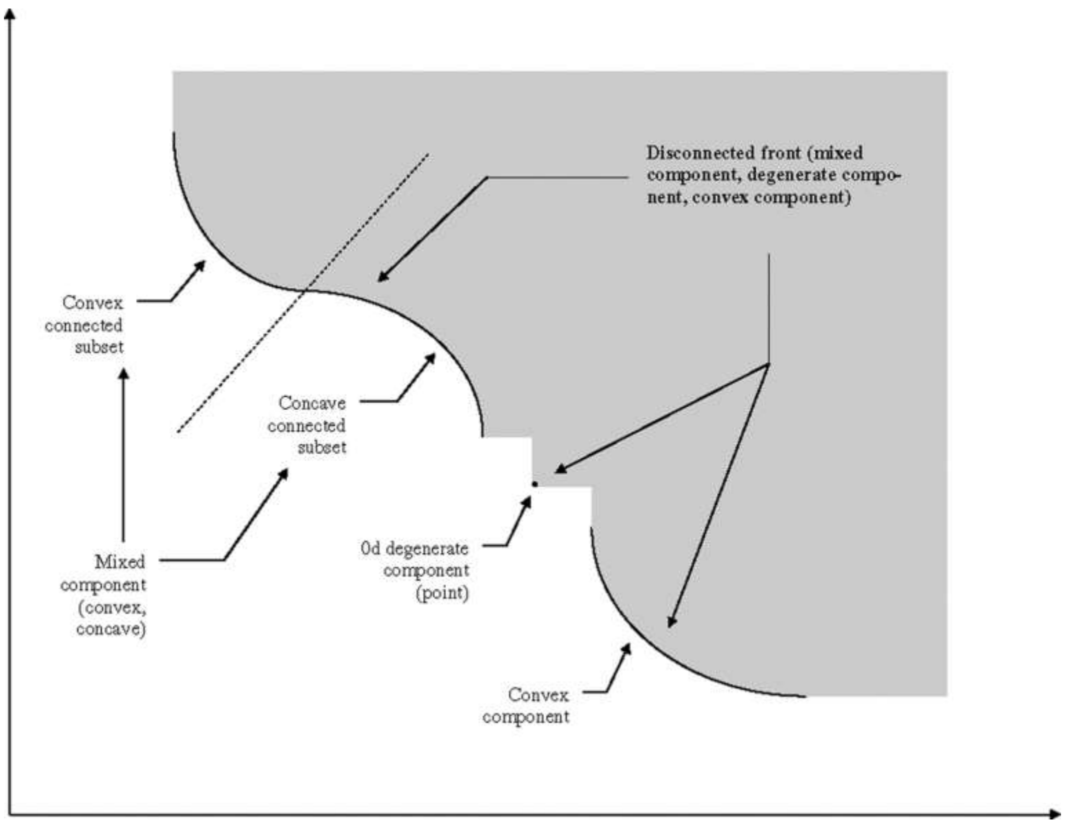

Disconnected front (mixed component, degenerate component, convex component)
Convex connected subset
Concave connected subset
Mixed component (convex, concave)
0d degenerate component (point)
Convex component

Figure 3.3: Example of mixed geometry front consists of a half-convex and half-concave component, a degenerate zero dimensional point, and a convex component. Reprinted from (Huband et al., 2006).

3.2 The Features of the Test Problems

In the scientific literature, various features for multi-objective optimisation test problems are presented. Those features are designed to make the problems difficult enough to examine algorithmic performance. Examples of these features are deception (Goldberg, 1987; Whitley, 1991), multimodality (Horn and Goldberg, 1995), noise (Kargupta, 1995) and epistasis (Davidor, 1990). Moreover, other features of test problems are suggested in Deb (1999) such as multimodality, deceptive, isolated optimum and collateral noise. These features can cause difficulties for evolutionary optimisers in terms of converging to the Pareto optimal front (POF) and maintaining the population diversity. Furthermore, some characteristics of the POF such as convexity or non-convexity, discreteness, and non-uniformity could cause difficulties in term of the population diversity (Zitzler et al., 2000). Branke (1999) asserted that the test problems should be simple and straightforward in order to understand the behaviour of the optimisation algorithm more easily. In addition, they should be describable and analyzable, and their parameters should be tunable. Nevertheless, they should be complicated enough to provide a true reflection of real world problems.

The main features of test problems for multi-objective optimisation presented in Deb et al. (2002) include the simplicity of formation, scalability to any number of decision variables, scalability to any number of objectives, accurate and specific knowledge of the shape and location of the Pareto fronts, finding a widely distributed set of Pareto solutions, and the capability to overcome the difficulty in converging to the true Pareto front. Furthermore, Huband et al. (2006) introduced the following key features of multi-objective test problems which present varying degrees of problem difficulty for the multi-objective optimisers:

- Pareto Optimal Front Geometry such as convex, linear, concave, mixed, degenerate and disconnected.
- Parameter Dependencies which refer to the problem and whether the objective is separable or nonseparable.
- Bias refers to whether the test problem may or may not be biased.
- Many-to-One Mappings which refer to the fitness landscape, which are either one-to-one or many-to-one.
- Modality refers to the problem objective; this may be unimodal or multimodal (can also be deceptive multimodality).

Huband et al. (2006) introduce some useful recommendations for designing multi-objective test problems including:

- No extremal parameters to the test problem in order to prevent exploitation by truncation operators.
- No medial parameters for the test problem in order to prevent exploitation by intermediate recombination.
- Scalability in the number of decision variables.
- Scalability in the number of objectives.
- The parameters of the test problem should have domains of dissimilar magnitude to encourage an optimiser to scale the strengths of the mutation operator.
- Knowledge of the POF in order to support the analysis of the results.

It can be seen that some of the recommendations of Huband et al. (2006) are identical to the features described by Deb et al. (2002).

3.3 Test Suite for Multi-objective Optimisation

Typically, a test suite should include different test problems which consist of a wide range of characteristics and features as mentioned in Section 3.2. However, it is impractical to have a test suite that incorporates all possible combinations of features. The test suites most commonly employed as benchmark multi-objective problems in the MOEA literature are the ZDT test suite (Zitzler et al., 2000), the DTLZ test suite (Deb et al., 2002) , the WFG (Huband et al., 2006) and more recently LZ09 (Li and Zhang, 2009). It good to note ZDT, DTLZ and WFG test suites have been used by MOHH approaches which presented in Section 2.3.3. The problem features in ZDT, DTLZ and WFG test suites are presented in Table 3.1.

Test features		**ZDT**	**DTLZ**	**WFG**
Pareto 1-1				√
Pareto M-1		√	√	√
Flat Regions				√
Modality	Unimodality	√	√	√
	Multimodality	√	√	√
Deception		√		√
Bias		√	√	√
Pareto Front known		√	√	√
Separability	Separable	√	√	√
	Nonseparable			√
Scalability	No of Parameters		√	√
	No of objectives		√	√
Front Geometry	Convex	√		√
	Concave	√	√	√
	Disconnected	√		√
	Degenerate			√
	Linear		√	√
	Mixed			√

Table 3.1: Listing of Test Problem Features in ZDT, DTLZ and WFG test suites.

3.3.1 ZDT Test Suite

This was introduced in Zitzler et al. (2000) and consists of six test problems. All the problems are separable and complicated enough to enable comparison over a variety of multi-objective evolutionary approaches. They also include some features which make the problems sufficiently difficult for optimisers such as multimodality, non-convexity and deception. For all problems of ZDT, the global optimum has the same variable values for different decision variables and objectives and the POF is known (Huang et al., 2007). In addition, the ZDT test suite has been widely used by many researchers in MOEAs. Therefore, test results are available and can be easily accessed. However, ZDT has some limitations. In terms of scalability, the number of decision variables and objectives only has one decision variable with two objectives. Moreover, none of its test problems has fitness landscapes with flat regions, a degenerate Pareto front or even non-separable features. In addition, the only deceptive problem is binary encoded. Also the global optimum for all ZDT problems lies on the lower bound, or in the centre of the search bounds (Huang et al., 2007). The ZDT test functions are presented in Table 3.2.

3.3.2 DTLZ Test Suite

This was introduced in Deb et al. (2002) and consists of seven different test problems. Similar to ZDT, the global optimum of DTLZ test problems has the same values for decision variables and objectives, all its problems are separable (Huang et al., 2007), and the POF is known. However, it differs from ZDT in terms of its scalability. DTLZ is scalable to any number of objectives and distance parameters. However, DTLZ has several shortcomings. For all problems, the global optimum is situated in the centre of the search range or on the bounds. None of these problems has fitness landscapes with flat regions, deceptive or non-separable features. Moreover, the number of decision variables is always strongly tied to the number of objectives (Huband et al., 2006). In addition, the increase in the number of objectives may cause difficulties for an optimiser to find the Pareto solutions (Deb et al., 2002; Kokolo et al., 2001). The DTLZ test functions are presented in Table 3.3.

3.3.3 WFG Test Suite

The Walking Fish Group's test suite (WFG) was created in Huband et al. (2006). It consists of nine test problems. The benchmark problems fully satisfy the recommendations set out in Section 3.2. The WFG is designed only for real valued parameters with no side constraints which make the problems

ZDT1	$f_1(x_1) = x_1$ $g(x_2,\dots,x_M) = 1 + 9 \cdot \sum_{i=2}^{m} x_i \;/\; (m-1)$ $h(f_1, g) == 1 - \sqrt{f_1 / g}$ *subject to* $0 \le x_i \le 1$
ZDT2	$f_1(x_1) = x_1$ $g(x_2,\dots,x_M) = 1 + 9 \cdot \sum_{i=2}^{m} x_i \;/\; (m-1)$ $h(f_1, g) == 1 - (f_1 / g)^2$ *subject to* $0 \le x_i \le 1$
ZDT3	$f_1(x_1) = x_1$ $g(x_2,\dots,x_M) = 1 + 9 \cdot \sum_{i=2}^{m} x_i \;/\; (m-1)$ $h(f_1, g) == 1 - \sqrt{f_1 / g} - (f_1 / g)\, sin(10\pi\, x_{i_1})$ *subject to* $0 \le x_i \le 1$
ZDT4	$f_1(x_1) = x_1$ $g(x_2,\dots,x_M) = 1 + 10\,(m-1) + \sum_{i=2}^{m} x_i^2 - 10\, cos(10\pi\, f_1) \;/\; (m-1)$ $h(f_1, g) == 1 - \sqrt{f_1 / g}$ *subject to* $-5 \le x_m \le 5,\; 0 \le x_1 \le 1$
ZDT5	$f_1(x_1) = 1 + u(x_1)$ $g(x_2,\dots,x_M) = \sum_{i=2}^{m} v(u(x_i))$ $h(f_1, g) == 1 / f_1$ *subject to* $v(u(x_i)) = 2 + u(x_i) \;\; if\; u(x_i) < 5$ $= 1 \qquad if\; u(x_i) = 1$
ZDT6	$f_1(x_1) = 1 + exp(-4x_1) sin^6(6\pi x_1)$ $g(x_2,\dots,x_M) = 1 + 9 \cdot ((\sum_{i=2}^{m} x_i \;/\; (m-1))^{0.25}$ $h(f_1, g) == 1 - (f_1 / g)^2$ *subject to* $0 \le x_i \le 1$

Table 3.2: ZDT test functions. Reprinted from (Zitzler et al., 2000)

easy to analyse and implement. The features of the WFG dataset are seen as the common choice for most MOEA researchers (Huband et al., 2006). Unlike most of the multi-objective test suites such as ZDT and DTLZ, the WFG test suite has powerful functionality; and a number of instances that have features not included in other test suites. The benchmark problems are non-separable problems, deceptive problems, a truly degenerate problem, and a mixed-shape Pareto front problem. In addition, WFG is scalable to any number of parameters and objectives, and the numbers of both distance- and position-related parameters can be scaled independently (Huband et al., 2006). The properties of the WFG problems are presented in Table 3.4.

DTLZ1	*MIN* $f_1(x) = 0.5\, x_1 x_2 \ldots x_{M-1}(1 + g(X_M))$ $\vdots \quad \vdots$ *MIN* $f_{M-1}(x) = 0.5\, x_1 (1 - x_2)(1 + g(X_M))$ *MIN* $f_M(x) = 0.5\,(1 - x_2)(1 + g(X_M))$ *subject to* $0 \leq x_i \leq 1$ $g(x_M) = 100\,(\lvert X_M \rvert + \sum_{xi \in X_M} (x_i - 0.5)^2 - \cos 20\pi(x_i - 0.5))$
DTLZ2	*MIN* $f_1(x) = (1 + g(X_M)) \cos(x_1\pi / 2) \ldots \cos(x_M - 1\pi / 2)$ $\vdots \quad \vdots$ *MIN* $f_M(x) = (1 + g(X_M)) \sin(x_1\pi / 2))$ *subject to* $0 \leq x_i \leq 1$ $g(x_M) = \sum_{xi \in X_M} (x_i - 0.5)^2$
DTLZ3	*As DTLZ2 with the g function given in DTLZ1*
DTLZ4	*As DTLZ2 with different meta-variable mapping:* $x_i \longrightarrow x_i^{\alpha}$ *where* $\propto = 100$
DTLZ5	*As DTLZ2 with different mapping of* $\theta_i h_M$ $\theta_i = \frac{\pi}{2(1+g(r))}(1 + 2g(r)x_i)$ *for* $i = 2,3,\ldots,$*(M-1)* $t_{i=1:k}^{1}$ *where* $g(x_M) = \sum_{xi \in X_M} x_i^{\;0.1}$
DTLZ6	*MIN* $f_1(X_1) = x_1$ $\vdots \quad \vdots$ *MIN* $f_M(X) = (1 + g(X_M))\, h(f_1, f_2, \ldots, f_{M-1}, g)$ *subject to* $0 \leq x_i \leq 1$ *where* $g(X_M) = 1 + \frac{9}{\lvert X_M \rvert} \sum_{xi \in X_M} x_i$ $h = M - \sum_{i=1}^{M-1} \left[\frac{f_i}{1+g} (1 + \sin(3\pi f_i))\right]$
DTLZ7	*MIN* $f_j(X) = \frac{1}{\lfloor \frac{n}{M} \rfloor} \sum_{i=\lfloor (j-1)\frac{n}{M} \rfloor}^{\lfloor j\frac{n}{M} \rfloor} x_i$ *subject to* $0 \leq x_i \leq 1$ $g_i(X) = f_M(X) + 4f_i(X) - 1 \geq 0$ $g(x_M) = 2f_M(X) + \min_{\substack{i,j=1 \\ i \neq j}}^{M-1} [f_i(X) + f_j(X)] - 1 \geq 0$

Table 3.3: DTLZ test functions. Reprinted from (Deb et al., 2002)

All WFG test problems are continuous problems that are constructed based on a vector that corresponds to the problem's fitness space. This vector is derived through a series of transition vectors such as multimodality and non-separability. The complexity of the problem can be increased according to the number of transition vectors. The WFG test functions are presented in Table 3.5.

The main advantage of the WFG test suite is that it is an excellent tool for comparing the performance of EAs over a range of test problems, and it has been shown to have a more comprehensive set of challenges when compared to DTLZ using NSGAII in Huband et al. (2006). Therefore, the WFG test suite has been selected to be the benchmark test suite employed in our multi-objective hyper-heuristics that we present in this thesis.

Problem	Obj.	Separability	Modality	Bias	Geometry
WFG1	$f_{1:M}$	separable	uni	polynomial, flat	convex, mixed
WFG2	$f_{1:M-1}$	non-separable	uni	no bias	Convex, disconnected
	$f_{1:M}$	non-separable	multi	no bias	
WFG3	$f_{1:M}$	non-separable	uni	no bias	liner, degenerate
WFG4	$f_{1:M}$	separable	multi	no bias	concave
WFG5	$f_{1:M}$	separable	deceptive	no bias	concave
WFG6	$f_{1:M}$	non-separable	uni	no bias	concave
WFG7	$f_{1:M}$	separable	uni	parameter dependent	concave
WFG8	$f_{1:M}$	non-separable	uni	parameter dependent	concave
WFG9	$f_{1:M}$	non-separable	multi, deceptive	parameter dependent	concave

Table 3.4: The properties of the WFG problems. Reprinted from (Huband et al., 2006).

3.3.4 Other Test Suites

The LZ09 test suite was created in Li and Zhang (2009) and consists of nine problems with complicated Pareto fronts in decision space. All its problems are continuous multimodal constrained problems that designed to deal with two objectives, except LZ09-F6, which is a tri-objective. The main advantages of problems with complicated Pareto set shapes (PSs) that they are offer a challenge for MOEAs. However, LZ09 is considered a relatively new test suite, few test results are available in the original study and in later work (e.g. Nebro and Durillo, 2010; Batista et al., 2010; Durillo, 2011; Loshchilov, 2011).

WFG1	$h_{M=1} : M = convex_m$ $h_M = mixed_M$ (with $\alpha = 1$ *and* $A = 5$) $t^1_{i=1:k} = y_i$ $t^1_{i=k+1:n} = S\_linear(y_i ,0.35)$ $t^2_{i=1:k} = y_i$ $t^2_{i=k+1:n} = b\_flat(y_i ,0.8,0.75,0.85)$ $t^3_{i=1:n} = b\_poly(y_i ,0.02)$ $t^4_{i=1:M-1} = r\_sum(\{y_{(i-1)k/(M-1)} + 1 , \dots , y_{ik/(M-1)}\}, \{2((i -$ $1k/(M - 1, \dots ,2ik/(M - 1)\})$ $t^4_M = r\_sum(\{y_{k+1} , \dots , y_{n)}\}, \{2(k + 1), \dots .2n\})$
WFG 2	$h_{M=1} : M = convex_m$ $h_M = disc_M$ (with $\alpha = \beta = 1$ *and* $A = 5$) $As\ t^1\ from\ WFG1. (Linear\ shift.)$ $t^2_{i=1:k} = y_i$ $t^2_{i=k+1:k+l/2} = r\_nonsep(\{y_{k+2(i-k)-1} , y_{k+2(i-k)}\},2)$ $t^3_{i=1:M-1} = r\_sum(\{y_{(i-1)k/(M-1)} + 1 , \dots , y_{ik/(M-1)}\}, \{1, \dots ,1\})$ $t^3_M = r\_sum(\{y_{k+1} , \dots , y_{k+l/2)}\}, \{1, \dots 1\})$
WFG 3	$h_{M=1} : M = linear_m(degenerate)$ $As\ t^{1:3}\ from\ WFG2. (Linear\ shift, non$ $-\ separable\ reduction, and\ weighted\ sum\ reduction.)$
WFG 4	$h_{M=1} : M = concave_m$ $t^1_{i=1:n} = S\_multi(y_i ,30,10,0.35)$ $t^2_{i=1:M-1} = r\_sum(\{y_{(i-1)k/(M-1)} + 1 , \dots , y_{ik/(M-1)}\}, \{1, \dots ,1\})$ $t^2_M = r\_sum(\{y_{k+1} , \dots , y_{n)}\}, \{1, \dots 1\})$
WFG5	$h_{M=1} : M = concave_m$ $t^1_{i=1:n} = S\_decept(y_i ,0.35,0.001,0.05)$ $As\ t^2\ from\ WFG4. (weighted\ sum\ reduction.)$
WFG 6	$h_{M=1} : M = concave_m$ $As\ t^1\ from\ WFG1. (Linear\ shift.)$ $t^2_{i=1:M-1} = r\_nonsep(\{y_{(i-1)k/(M-1)} + 1 , \dots , y_{ik/(M-1)}\}, k/(M - 1))$ $t^2_M = r\_nonsep(\{y_{k+1} , \dots , y_{n)}\}, l)$
WFG 7	$h_{M=1} : M = concave_m$ $t^2_{i=1:k}$ $= b\_param(y_i, r\_sum(\{y_{(i-1)} , \dots , y_n\}, \{1, \dots ,1\}), \frac{0.98}{49.98}, 0.02,50)$ $t^2_{i=k+1:n} = y_i$ $As\ t^1\ from\ WFG1. (Linear\ shift.)$ $As\ t^2\ from\ WFG4. (weighted\ sum\ reduction.)$
WFG8	$h_{M=1} : M = concave_m$ $t^1_{i=1:k} = y_i$ $t^1_{i=k+1:n}$ $= b\_param(y_i, r\_sum(\{y_1 , \dots , y_{i-1}\}, \{1, \dots ,1\}), \frac{0.98}{49.98}, 0.02,50)$ $As\ t^1\ from\ WFG1. (Linear\ shift.)$ $As\ t^2\ from\ WFG4. (weighted\ sum\ reduction.)$
WFG9	$h_{M=1} : M = concave_m$ $t^1_{i=1:n-1}$ $= b\_param(y_i, r\_sum(\{y_{i+1} , \dots , y_n\}, \{1, \dots ,1\}), \frac{0.98}{49.98}, 0.02,50)$ $t^1_n = y_n$ $t^2_{i=1:k} = S\_decept(y_i ,0.35,0.001,0.05)$ $t^2_{i=k+1:n} = S\_multi(y_i ,30,95,0.35)$ $As\ t^2\ from\ WFG6. (non - separable\ reduction.)$

Table 3.5: WFG test functions. Reprinted from (Huband et al., 2006)

Van Veldhuizen's test suite was created in Van Veldhuizen (1999) which consists of seven multi-objective test problems. The main drawbacks of Van

Veldhuizen's problems are that they were designed for only two or three decision variables and are not scalable in terms of the number of objectives. In addition, none of these problems has any deceptive, flat regions or many-to-one fitness landscapes (Huband et al., 2006).

Deb (1999) introduced a toolkit for creating test problems for multi-objective optimisation. Deb's toolkit incorporates three functions: a distribution function to assess the optimiser's performance in terms of the diversify along the POF, a distance function to assess the optimiser's performance in terms of convergence towards the POF, and a shape function to specify the shape of the POF. Deb's toolkit has shortcomings; it was designed to construct a problem with two objectives only, and no problems with flat regions, degenerate or even mixed Pareto front geometries are provided. Moreover, no real valued deceptive functions are considered in the toolkit.

3.4 Other Test Functions Problems for Multi-objective Optimisation

In the MOEA literature, various test problems have been presented. However, some test problems had shortcomings in terms of the simplicity of construction and the scalability of the number of parameters and objectives (Deb et al., 2002). For instance, Schaffer (1985) presented two test problems (SCH1 and SCH1). Both problems were scalable but only to single decision variable. Poloni et al. (2000) presented a test problem (POL) that has only two decision variables. Fonseca and Fleming (1995) and Kursawe (1990) introduced their own test problems, FON and KUR respectively. Both test problems were scalable to any number of decision variables but were not scalable in terms of the number of objectives. Viennet (1996) introduced a test problem (VNT) that was scalable to only three objectives.

3.5 Summary

Several multi-objective test problems have been proposed in the scientific literature such as real-world problems, combinatorial optimisation problems, discrete or integer-based problems, noisy problems, dynamic problems, and problems with side constraints. In this thesis, we focus on continuous unconstrained real-valued problems. It is essential that algorithms are tested in order to have a clear perception of their strengths and weaknesses. To accomplish this effectively, it is crucial to first develop a strong understanding and undertake a precise analysis of the test problems at hand. This chapter has reviewed the multi-objective test problems that are particularly relevant to this thesis. The most common multi-objective test problem such as the ZDT test suite (Zitzler et al., 2000), the DTLZ test suite (Deb et al., 2002) and the WFG (Huband et al., 2006) are identified and discussed. A description of those problems with an analysis of their features is given as well. The WFG test suite has been selected to be the benchmark test suite employed in our multi-objective hyper-heuristics that we present in this thesis, as it has been shown to have a more comprehensive set of challenges among other test suites (Huband et al., 2006).

In next chapter, we discuss design issues related to the development of hyper-heuristics for multi-objective optimisation. And we propose an online learning selection choice function based hyper-heuristic for multi-objective optimisation.

4 A Multi-objective Hyper-heuristic Framework

Burke et al. (2003b) provide a generic hyper-heuristic framework (see Section 2.3.1). Soubeiga (2003) presents general guidelines for designing an effective framework for a hyper-heuristic for single-objective optimisation. Burke et al. (2003a) discussed a framework for hyper-heuristic for multi-objective combinatorial problems. However, no further investigations, nor any related information, are given for how to build a hyper-heuristic for multi-objective optimisation to deal specifically with continuous problems. In this chapter, we discuss design issues related to the development of hyper-heuristics for multi-objective optimisation. And we propose an online learning selection choice function based hyper-heuristic for multi-objective optimisation. A choice function is utilised as a selection mechanism for the proposed framework.

4.1 A Selection Choice Function Hyper-heuristic Framework

The design of the framework for our multi-objective hyper-heuristic is inspired by two facts. Firstly, there is no existing algorithm that excels across all types of problems. In the context of multi-objective optimisation, no single MOEA algorithm has the best performance with respect to all performance measures in all types of multi-objective problems. Some comparison studies in MOEAs which emphasises this idea are presented in Section 2.1.11. This fact is also supported by the No Free Lunch Theorem (Wolpert and Macready,1997). Secondly, the hybridisation or combining different (meta)heuristics/algorithms into one framework could yield promising results compared to (meta)heuristics/algorithms when used alone. In Section 2.3, we reviewed many studies that support this fact. According to those facts, we are looking to gain an advantage of combining different algorithms in a hyper-heuristic framework for multi-objective optimisation to get benefit from the strengths of the algorithms and avoid their weaknesses.

The idea of hybridising a number of algorithms (heuristics) into a selection hyper-heuristic framework is straightforward and meaningful. However, many design issues related to the development of hyper-heuristics for multi-objective optimisation require more attention when designing such a framework to be applicable and effective.

The main components of the hyper-heuristic framework are low level heuristics, selection method, learning mechanism and move acceptance method. The choosing of these components is critical. In our opinion, all components are important and could affect the performance of the hyper-heuristics. For instance, if we employ very powerful low level heuristics and a poor move acceptance method, we have less chance of producing high quality of solutions. This is especially true if we employ a *complete* algorithm as a low level heuristic and this algorithm produces a good quality solution. With a poor move acceptance method, the obtained solution could be rejected. The reverse is also true. Therefore, each component in the hyper-heuristic framework plays a significant role in improving the quality of both the search and the eventual solution. The components of the hyper-heuristic in the context of multi-objective optimisation are discussed in depth in as follows:

- **Low level heuristics:**

  The choice of appropriate low level heuristics is not an easy task. Many questions arise here, *what* heuristics (algorithms) are suitable to deal with multi-objective optimisation problems. Are priori approaches or a posteriori approaches more suitable? Are non Pareto-based or a Pareto-based more applicable? (see Section 2.1). As one of hyper-heuristic aims is raising the level of generality, a posteriori approach is more suitable to achieve this aim. Unlike the priori approaches, there is no need to set objective preferences or weights prior to the search process in the posteriori approach such as MOEAs which based on Pareto dominance. Moreover, we agree with many researchers (Deb and Goldberg, 1989; Bäck, 1996; Fonseca and Fleming, 1998; Deb, 2001; Coello et al., 2007a; Anderson et al., 2007; Zhang and Li, 2007; Miranda et al., 2010) that evolutionary algorithms are more suitable in dealing with multi-objective optimisation problems because of their population-based nature, which means they can find Pareto optimal sets (trade-off solutions) in a single run, which allows a decision maker to select a suitable compromise solution (with respect to the space of the solutions). In the context of multi-objective hyper-heuristics, a decision maker here could be a selection method that decides which is the best low level heuristic to select at each decision point (with respect to the space of the heuristics).

The main aim of hyper-heuristics is to draw on the strengths of individual low level heuristics and avoid their weaknesses. This motivates us to make use of classical Pareto-based MOEAs (NSGAII, SPEA2 and MOGA) to act as low level heuristics within our hyper-heuristics framework, as their features are more likely (in our view) to generate high quality solutions. In other words, we reuse the conventional MOEAs to benefit from their strengths even if they have some shortcomings. The features of classical MOEAs make them suitable to enable us to investigate their combined use within a multi-objective hyper-heuristic framework. Although NSGAII, SPEA2 and MOGA are no longer considered *state-of-the-art* MOEAs, more powerful population-based methods such as decomposition-based approaches MOEA/Ds (e.g. (Li and Zhang, 2009; Li and Landa-Silva, 2011)) and indicator-based approaches (e.g. (Auger et al.,2012; Bader and Zitzler, 2011)) may outperform them. However, they are still viewed as a baseline for MOEA. Moreover, they incorporate much of the known MOEA theory (Van Veldhuizen and Lamont, 2000). Comparative studies, which support this decision, are presented in Section 2.1.11.

- **Selection method:**

As a selection hyper-heuristic relies on an iterative process, the main questions arise here are *what* is an effective way can use to choose an appropriate heuristic at each decision point? And *how* to choose this heuristic i.e. which criteria can be considered when choosing a heuristic? In single-objective cases, this criterion is easy to determine by measuring the quality of the solution such as the objective/cost value and time. However, this is more complex when tackling a multi-objective problem. The quality of the solution is not easy to assess. There are many different criteria that should be considered such as the number of non-dominated individuals and the distance between the non-dominated front and the POF. We will discuss this later when dealing with learning and the feedback mechanism (will discuss later). As we aim to keep the framework simple, we should keep in a higher level of abstraction as much as possible. Therefore, we do not employ any information about problem-specific such as the number of objectives nor information about the nature of the solution space. We focus more on the performance of the low level heuristics. This will boost the intensification element. So, a heuristic with the best performance will be chosen more frequently to exploit the search area. We are not only looking for the intensification but we also give attention to diversification. We attempt to achieve a kind of balance between the intensification and diversification when choosing a heuristic. Selection methods based on randomisation support only the diversification by exploring unvisited areas of the search space. Reinforcement learning (RL) (Sutton and Barto, 1998) that use, as a selection method, support intensification by rewarding and punishing each heuristic based on its performance during the search using a scoring mechanism. An example of this can be found in Nareyek (2003). The choice function that is used as a selection method in hyper-heuristics provides a balance between intensification and diversification. The choice function addresses the trade-off between the undiscovered areas of the search space and the past performance of each heuristic. The experimental results demonstrate that the choice function based hyper-heuristic outperforms other random based hyper-heuristics over shelf space allocation problems (Bai, 2005). In addition, the computational results show the choice function all-moves based hyper-heuristic is superior to other hyper-heuristics that combine different selection methods with different move acceptance methods on a project presentation problem (Cowling et al., 2002c). The choice function meets our requirements for the selection method. Moreover, it was successful when used as a selection method in the hyper-heuristic for single-objective optimisation (Soubeiga, 2003). For these reasons, we have decided to employ the choice function as a selection method and to act as a high level strategy in our multi-objective hyper-heuristic framework. More details about the choice function are provided in Section 4.3.

- **Learning and feedback mechanism:**

Not all hyper-heuristic approaches incorporate a learning mechanism (see Section 2.3.2). However, a learning mechanism is strongly linked to the selection method. An example of this is a random hyper-heuristic which is classified as an offline learning approach (Burke et al., 2010), because the random selection does not provide any kind of learning. In the context of our multi-objective hyper-heuristic framework, a learning process is an essential element in the choice function to do its task as a selection method effectively. The learning mechanism guides the selection method to which *best* heuristic should be chosen at each decision point. We mean by a *best* heuristic the heuristic that produces solutions with good quality. As we mention previously, the measurement of the quality of the solution for multi-objective problems requires us to assess different aspects of the non-dominated set in the objective space (see Section 2.1.9). As inspiration from the first fact, that mentioned earlier, is that no single MOEA excels across all performance measures (Tan et al., 2002). Therefore, we employ a learning mechanism based on different measures using the ranking scheme to provide a feedback about the quality of the solutions. We do not aim to choose a heuristic that performs well with respect to all measures. This cannot be achieved anyway in accordance with the No Free Lunch Theorem (Wolpert and Macready, 1997). But we aim to select a heuristic that performs well in most measures. More details about the learning mechanism that is employed in our multi-objective hyper-heuristic are provided in Section 4.2.

- **Move acceptance method:**

The selection hyper-heuristic framework comprises two main stages: selection and move acceptance methods (Burke at al., 2010). In the scientific literature, many methods are presented that act as move acceptance strategies in hyper-heuristics (see Section 2.3.2.1) a move acceptance criterion can be *deterministic* or *non-deterministic*. A deterministic move acceptance criterion produces the same result, given the configuration (e.g. proposed new solution etc). A non-deterministic move acceptance criteria may generate a different result even when the same solutions are used for the decision at a same given time. This could be because the move acceptance criterion depends on time or it might have a stochastic component while making the accept/reject decision. Examples of deterministic move acceptance criteria are All-Moves, Only-Improving and Improving & Equal. In All-Moves, the candidate solution is always accepted whether a move worsens or improves the solution quality. The candidate solution in Only-Improving criteria is accepted only if it improves the solution quality, while in Improving & Equal criteria, the candidate solution is accepted only if it improves or equal to the current solution. For non-deterministic move acceptance criteria, the candidate solution is always accepted if it improves the solution quality, while worsening solutions can be accepted based on an acceptance function including the great deluge algorithm (GDA) (Dueck, 1993), late acceptance (Burke and Bykov, 2008), monte carlo (Glover and Laguna, 1995) and simulated annealing (Kirkpatrick et al., 1983) . In this thesis, we investigate a multi-objective choice function based hyper-heuristic using different move acceptance methods including deterministic (All-Moves) and non-deterministic strategies (GDA and LA). These investigations are presented and discussed in Chapters 5-7. To the best of the authors' knowledge, this thesis, for the first time, investigates the influence of the move acceptance as a component in a selection hyper-heuristic for multi-objective optimisation. Since no similar work has been reported in the literature, this investigation is a useful reference not only for the work presented in this thesis but also for other researchers interested in selection hyper-heuristics for multi-objective optimisation. We decided to employ GDA and LA as a move acceptance component in our multi-objective hyper-heuristic choice function as they are both simple and depend on a small number of parameters (Petrovic et al., 2007). Moreover, it was successful with single-objective optimisation (Kendall and Mohamad, 2004). We also note that no work has been reported in the scientific literature that utilises GDA and LA as a move acceptance component within a hyper-heuristic framework for multi-objective optimisation.

The multi-objective choice function based hyper-heuristic framework is shown in Figure 4.1. The choice function acts as the high level strategy and three well-known multi-objective evolutionary algorithms (NSGAII, SPEA2, and MOGA) act as low level heuristics. The choice function considers the performances of low level heuristics in order to select a suitable heuristic as the search progresses. This process adaptively ranks the performance of low level heuristics with respect to the performance metrics, deciding which one to call at each decision point.

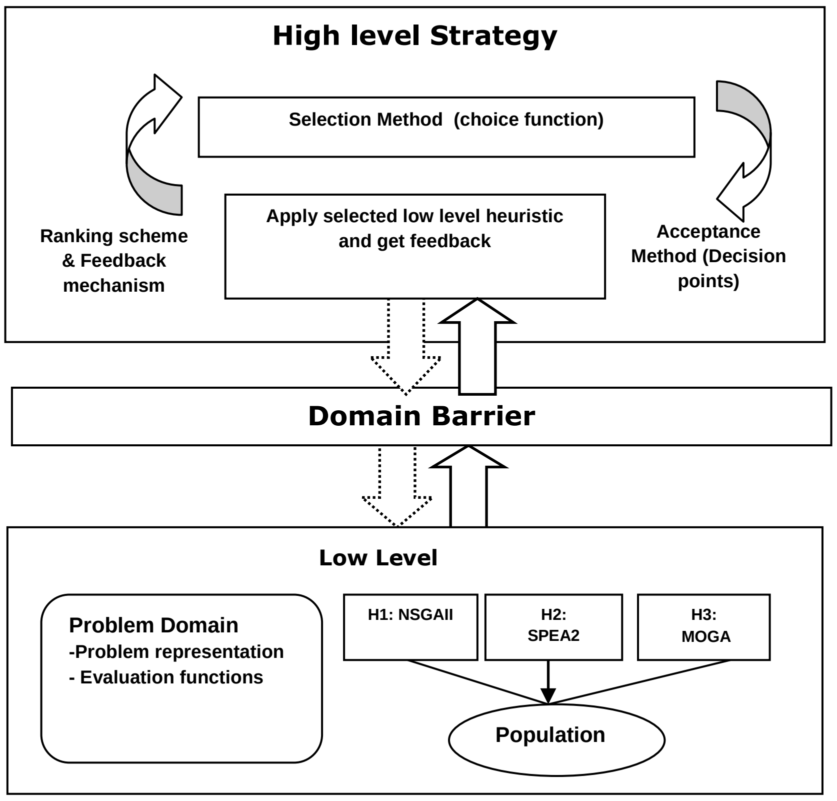

High level Strategy
Selection Method (choice function)
Apply selected low level heuristic and get feedback
Ranking scheme & Feedback mechanism
Acceptance Method (Decision points)
Domain Barrier
Low Level
Problem Domain
-Problem representation
- Evaluation functions
H1: NSGAII
H2: SPEA2
H3: MOGA
Population

Figure 4.1: The proposed framework of the hyper-heuristic choice function based for multi-objective optimisation problems. In this framework, the choice function acts as a high level strategy and three well-known multi-objective evolutionary algorithms (NSGAII, SPEA2, and MOGA) act as low level heuristics.

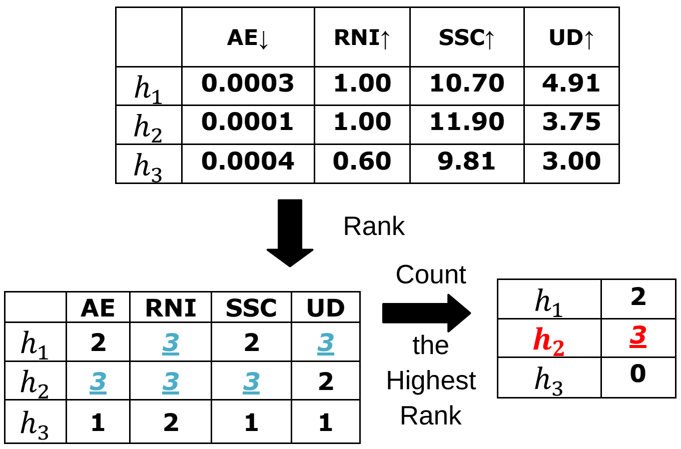
In this framework. the high level strategy does not have any knowledge of the problem domain and solutions. This is a separation of domain information known as the domain barrier. To provide the knowledge of the problem domain to the high level strategy, a number of performance metrics are utilised as a feedback mechanism. More details about the feedback mechanism are presented in the next section. The high level strategy selects one low level heuristic at each decision point according to the information obtained from the feedback mechanism. Note that the three low level heuristics operate in an encapsulated way. Each heuristic has its own characteristics described in Section 2.1. There is no direct information exchange between low level heuristics but they are sharing the same population. The framework is flexible and could incorporate any MOEA(s) for multi-objective optimisation in future work. The framework designed to make used for the complete algorithm as low level heuristic. No much information required from the low level heuristic, only the number of function evaluations and objectives as input and non-dominated solutions as output.

4.2 The Online Learning Feedback Mechanism and the Ranking Scheme

Four performance metrics are selected to be indicators for the feedback mechanism. These performance metrics are as follows (see Section 2.1.9. for more details):

- Algorithm effort (AE) (Tan et al., 2002
- Ratio of non-dominated individuals (RNI) (Tan et al., 2002)
- Size of space covered or S-metric Hypervolume (SSC) (Zitzler and Thiele, 1999).
- Uniform distribution of a non-dominated population (UD) (Srinivas and Deb, 1994)

The motivation behind choosing these metrics is that they have been commonly used for performance comparison of MOEAs to measure different aspects of the final non-dominated solutions in the objective space (Tan et al., 2002). In addition, they do not require prior knowledge of the POF, which means that our framework, is suitable for tackling real-world problems in future studies. The task of the performance metrics is to provide information about the performance of the low level heuristics. It is to provide an online learning mechanism in order to guide the high level strategy during the search and determine which low level heuristic should be selected next. Since those metrics are not in the same scalar units, it is difficult to determine the best heuristic with respect to the four performance metrics. Therefore, we use a ranking scheme to score the performance of heuristics. This ranking scheme is simple and flexible and enables us to incorporate any number of low level heuristics and performance indicators. Unlike the ranking scheme used in Vázquez-Rodríguezand Petrovic (2012), which ranks the algorithms based on their probabilities against the performance indicators' using a mixture of experiments, our ranking scheme relies on sorting the low level heuristics in descending order based on the highest ranking among the other heuristics. For N number of low level heuristics and M number of performance metrics, N heuristics are ranked according to their performances against M metrics. For a particular metric m_i , $i \in M$, a heuristic h_j , $j \in N$ with the best performance among other heuristics assigns the highest rank, which is equal to N. Then another heuristic with the second best performance is ranked as $N-1$ and so on. If two heuristics have the same performance, both heuristics are assigned the same rank. This ranking process is applied for all M metrics. After all heuristics are ranked against all metrics, the frequency of the highest rank for each heuristic is counted. A heuristic with the largest frequency count of the highest rank is more desirable. An example of how the ranking scheme works using the four performance metrics to rank three low level heuristics is described in Figure 4.2.



	AE↓	RNI↑	SSC↑	UD↑
h_1	0.0003	1.00	10.70	4.91
h_2	0.0001	1.00	11.90	3.75
h_3	0.0004	0.60	9.81	3.00

	AE	RNI	SSC	UD
h_1	2	3	2	3
h_2	3	3	3	2
h_3	1	2	1	1

h_1	2
h_2	3
h_3	0

Figure 4.2: An example of how three low level heuristics, denoted as h_1, h_2 and h_3 are ranked against four performance metrics of AE, RNI, SSC, and UD. The ↓ and ↑ show that heuristics are ranked in decreasing and increasing order for the given metric, respectively, 3 indicating the top ranking heuristic. Each row in the top table represents each low level heuristic's performance with respect to the four metrics. Each row in the leftmost table represents each heuristic's rank among other heuristics for each metric. The rightmost table represents the frequency of each heuristic ranking the top over all metrics.

As we are not only looking for the heuristic that has the best performance, but also aiming to have a larger number of non-dominated individuals, the frequency count of the highest rank for a heuristic h_i is summed with its RNI rank using:

$$\forall i \in N \quad f_i(h_i) = freq_{highest\_rank}(h_i) + RNI_{rank}(h_i) \qquad (4.1)$$

where N represents the number of the low level heuristics and $f_i(h_i)$ reflects a performance of heuristic h_i. In the example presented in Figure 4.2, the performance value of h_2 is equal to 6 using Equation 4.1. In the case of two heuristics h_i and h_j having the same value of $f_i(h_i)$ and $f_j(h_j)$, we consider the heuristic that has a higher count of the second highest rank $(N-1)$.

4.3 The Choice Function Meta-heuristic Selection Method

The key idea behind the use of a choice function as a selection mechanism in a hyper-heuristic is guiding the search by choosing a heuristic at each decision point based on its historical performance and the time passed since the last call to the heuristic. This selection process supports both intensification and diversification which provides a kind of learning for the hyper-heuristic. If a heuristic performs well, the choice function will choose it to exploit the search area. Even a heuristic that does not perform well still has a chance to be called in order to explore new areas of the search space.

Cowling et al. (2002c) and Kendall et al. (2002) propose a choice function based hyper-heuristic for a single-objective problem that employs the choice function as a heuristic selection method which adaptively ranks the low level heuristics (h_i) using:

$$CF(h_i) = \alpha f_1(h_i) + \beta f_2(h_j, h_i) + \delta f_3(h_i) \tag{4.2}$$

where f_1 measures the individual performance of each low level heuristic, f_2 measures the performance of pairs of low level heuristics invoked consecutively, and finally, f_3 is the elapsed CPU time since the heuristic was last called. Both f_1 and f_2 support intensification while f_3 supports diversification. The parameter values for α, β and δ are changed adaptively based on a similar idea to reinforcement learning. The choice function based hyper-heuristic was applied to nurse scheduling and sales summit scheduling. The study shows that the hyper-heuristic combining Choice Function with All-Moves acceptance performed the best when compared to the other methods. The study also shows that the choice function hyper-heuristic is successful in making effective use of low level heuristics, due to its ability to learn the dynamics between the solution space and the low level heuristics to guide the search process towards better quality solutions. For more details, see (Soubeiga, 2003).

The formula in Equation 4.2 was extended for multi-criteria decision making (MCDM) in Soubeiga (2003) as:

$$\forall l,\ CF_l\,(h_i) = \alpha_l\, f_{1l}(h_i) + \beta_l f_{2l}(h_i, h_j) + \frac{\delta}{c} f_3(h_i) \qquad (4.3)$$

Each individual criterion l has its own choice function. The choice function $CF_l\,(h_i)$ reflects the overall performance of each low level heuristic h_i with respect to each criterion l. Of course, Equation in 4.2 is still valid if several criteria are aggregated into one objective function.

In this thesis, we propose a modified version of the choice function heuristic selection method as a component in our multi-objective selection hyper-heuristic. The modified choice function is formulated as

$$\forall i \in N,\ CF_i(h_i) = \alpha f_{1i}(h_i) + f_{2i}(h_i) \qquad (4.4)$$

where $f_{1i}(h_i)$ is computed using Equation 4.1 based on the ranking scheme described earlier in Section 4.2. It measures the individual performance of each low level heuristic h_i. $f_{2i}(h_i)$ is the number of CPU seconds elapsed since the heuristic was last called. $f_{1i}(h_i)$ provides an element of intensification while $f_{2i}(h_i)$ provides an element of diversification, by favouring those low level heuristics that have not been called recently. α is a large positive value (e.g. 100). It is important to strike a balance between f_1 and f_2 values, so that they are in the same scalar unit. Experiments to tune α are conducted in Chapter 5. The low level heuristic h_i with the largest value of $CF_i(h_i)$ is the heuristic that is applied for the next iteration of the search.

Equation 4.4 differs from Equations 4.2 and 4.3 as it is adjusted to deal with a given multi-objective optimisation problem, but their goal is the same, measuring the overall performance of a low level heuristic h_i. Unlike Equation 4.3 which reflects the performance of low level heuristics with respect to the criteria (objective values), Equation 4.4 reflects the overall performance of low level heuristics with respect to the performance metrics that measures the resulting non-dominated set in the objective space. Our multi-objective hyper-heuristic works at a high level of abstraction, no information for problem-specific is required such as the number of objectives nor for the nature of the solution space, only the number of low level heuristics. This advantage makes our framework suitable to apply to single-objective optimisation by replacing the performance metrics and low level heuristics to those which are designed for single-objective problems.

4.4 Summary and Remarks

Hyper-heuristics have drawn increasing attention from the research community in recent years, although their roots can be traced back to the 1960's. They perform a search over the space of heuristics rather than searching over the solution space directly. Research attention has focussed on two types of hyper-heuristics: selection and generation. A selection hyper-heuristic manages a set of low level heuristics and aims to choose the best heuristic at any given time using historic performance to make this decision, along with the need to diversify the search at certain times.

References to a hyper-heuristic framework for multi-objective optimisation are scarce. Burke et al., (2003b) provide a generic hyper-heuristic and Soubeiga (2003) presents general guidelines for designing a framework for hyper-heuristics. Burke et al. (2003a) discussed a framework for a hyper-heuristic for multi-objective combinatorial problems. No further investigations nor any related information are given for how to build a hyper-heuristic for multi-objective optimisation in particular continuous problems. This chapter has addressed the design issues related to the development of hyper-heuristics for multi-objective optimisation. The framework of our multi-objective hyper-heuristic is inspired by two facts: (i) no existing algorithm that excels across all types of problems, and (ii) the hybridisation or combining different (meta)heuristics/algorithms into one framework could yield promising results compared to (meta)heuristics/ algorithms on their own. Accordingly, we discussed each component of a hyper-heuristic framework from the multi-objective prospective including the low level heuristics, the selection method, the learning and feedback mechanisms and finally the move acceptance method.

Hyper-heuristic frameworks, generally, impose a domain barrier which separates the hyper-heuristic from the domain implementation along with low level heuristics. Moreover, this barrier does not allow any problem specific information to be passed to the hyper-heuristic itself during the search process. We designed our framework in the same modular manner, making it highly flexible and its components reusable and easily replaceable. Our online selection choice function based hyper-heuristic for multi-objective (HHMO_CF) controls and combines the strengths of three well-known multi-objective evolutionary algorithms (NSGAII, SPEA2, and MOGA), which are utilised as the low level heuristics. The motivation behind choosing these MOEAs is that they are efficient and effective and they also incorporate much of the known MOEA theory (Van Veldhuizen and Lamont, 2000). The choice function utilised, as a selection method, acts as a high level strategy which adaptively ranks the performance of three low-level heuristics, deciding which one to call at each decision point. Four performance metrics (AE, RNI, SSC and UD) act as an online learning mechanism to provide knowledge of the problem domain to the selection mechanism.

There is strong empirical evidence showing that different combinations of heuristic selection and acceptance methods in a selection hyper-heuristic framework yield different performance in single-objective optimisation (Burke et al., 2012). In the next three chapters, we will investigate the proposed multi-objective choice function based hyper-heuristic combined with different move acceptance strategies including All-Moves as deterministic move acceptance and Great Deluge (GDA) and Late Acceptance (LA) as non-deterministic move acceptance.

5 A Heuristic Selection Using Deterministic Move Acceptance Strategy

In the previous chapter, we presented the framework for an online learning selection hyper-heuristic for multi-objective optimisation. The key feature of the proposed selection hyper-heuristic is the use of a modified choice function as a selection method based on ranking low level heuristics according to their performance. This chapter investigates the proposed multi-objective choice function based hyper-heuristic when combining All-Moves as a move acceptance strategy.

5.1 Choice Function All-Moves for Selecting Low Level Meta-heuristics (HHMO_CF_AM)

In single-objective optimisation, Cowling et al., (2002c) investigate the performance of different hyper-heuristics, combining different heuristic selection, with different move acceptance methods on a real world scheduling problem. Simple Random, Random Descent, Random Permutation, Random Permutation Descent, Greedy and Choice Function were introduced as heuristic selection methods. The authors utilised the following deterministic acceptance methods: All-Moves accepted and Only Improving moves accepted. The hyper-heuristic combining Choice Function with All-Moves acceptance performed the best. In this chapter, we investigate the performance of the proposed multi-objective choice function based hyper-heuristic, utilising All-Moves as a deterministic acceptance strategy, meaning, that we accept the output of each low level heuristic whether it improves the quality of the solution or not. We use the multi-objective hyper-heuristic framework that we proposed in Chapter 4. Three well-known multi-objective evolutionary algorithms (NSGAII, SPEA2, and MOGA), act as the low level heuristics.

The multi-objective choice function all-moves based hyper-heuristic (HHMO_CF_AM) is shown in Algorithm 10. Initially, a greedy algorithm is executed to determine the best low level heuristic to be selected for the first iteration (steps 2-6). All three low level heuristics are run (step 3). Then, the three low level heuristics are ranked by using Equation 4.1 and their choice function values are computed by using Equation 4.4 (steps 4 & 5). The low level heuristic with the largest choice function value is selected (step 6) to be applied as an initial heuristic (step 8). Then, for all low level heuristics, the ranking mechanism is updated (step 9). The choice function values are also computed and updated (step 10). According to the updated choice function values, the low level heuristic with the largest choice function value is selected to be applied in the next iteration (step 11). This process is repeated until the stopping condition is met (steps 7-12). Note that the greedy algorithm is applied only once at the beginning of the search, in order to determine which low level heuristic to apply first. Then, only one low level heuristic is selected at each iteration.

Algorithm 10: Multi-objective Choice Function All-Moves based Hyper-heuristic

```
 1: procedure HHMO_CF_AM $(H)$ $whereas\ H$ is a set of the low level heuristics
 2:  Initialisation
 3:  Run $h, \forall h \in H$ for $ng$ function evaluations
 4:  Rank $h, \forall h \in H$ based on the ranking scheme
 5:  Get $CF(h), \forall h \in H$
 6:  Select $h$ with the largest $CF(h)$ as an initial heuristic
 7: repeat
 8:    Execute the selected $h$ for $ng$ function evaluations
 9:    Update the rank of $h, \forall h \in H$ based on the ranking scheme
10:    Update $CF(h), \forall h \in H$
11:    Select $h$ with the largest $CF(h), \forall h \in H$
12: until (termination criteria are satisfied)
13: end procedure
```

Our multi-objective selection choice function based hyper-heuristic (HHMO_CF) involves N multi-objective meta-heuristics as low level heuristics for solving k-objective optimisation problems. Each low level heuristics executes a fixed number of function evaluations ng where n is the size of population and g is the number of generations. Because of the high level abstraction in HHMO_CF, the number of objectives in k is not considered. HHMO_CF executes for a fixed number of iterations (decision points) (NDP) as computational resource is always limited. In each iteration, HHMO_CF evaluates ng function evaluations. That is, HHMO_CF executes for $\mathrm{NDP} \times \mathrm{ng}$ function evaluations. Regardless of the computational cost for low level heuristics are used, the high level strategy; the selection choice function method in (Steps 9 & 10) ranks N low level heuristics with respect to M performance metrics. So the computational cost of the choice function at each iteration is $\mathrm{N} \times \mathrm{M}$. HHMO_CF takes linear time to execute; $\mathrm{ng} \times \mathrm{N} \times \mathrm{M}$. We note that N and M are negligible. In the best case, HHMO_CF only requites $\mathrm{O(ng)}$ basic operations per iteration to achieve an approximation Pareto front which has a comparable quality to that obtained by the low level heuristic when run individually. The experiments observation shows that there is no notable difference between the execution time of our method and other low level heuristics run on their own. It is good to note that all the methods are executed the same number of function evaluations.

5.2 Performance Comparison of Multi-objective Choice Function Based Hyper-heuristic and Low Level Heuristics

A set of experiments using the WFG test suite is conducted to see the performance difference between using each individual multi-objective meta-heuristic (NSGAII, SPEA2, and MOGA) run on its own and the proposed HHMO_CF_AM selection hyper-heuristic that combines them. Although NSGAII and SPEA2 have previously been applied to the WFG test suite in Bradstreet et al. (2007), we repeat the experiments, including MOGA, under our own experimental settings. For short, we refer to the HHMO_CF_AM as HH_CF.

5.2.1 Performance Evaluation Criteria

The comparison of the quality of solutions for multi-objective optimisation is more complex than single-objective problems. The number of non-dominated individuals should be maximised, the distance to the non-dominated front should be minimised, i.e. the resulting non-dominated set should be distributed uniformly as much as possible and converge well toward the POF. Because of that, we use three performance metrics RNI, SSC, and UD, to assess the quality of approximation sets in different aspects. In addition, we use the students test (t-test) as the statistical test while comparing the average performances of a pair of algorithms with respect to a metric averaged over 30 trials. The null hypothesis is as follows:

$$\begin{cases} H_0 \, the\ performance\ of\ a\ pair\ of\ algorithms\ \ have\ the\ same\ means \\ H_1 \, the\ performance\ of\ a\ pair\ of\ algorithms\ \ have\ different\ means \end{cases}$$

We assume two independent samples, unequal variance and one-tailed distribution with 95% confidence level. We aim to reject the null hypothesis and accept the alternative hypothesis and demonstrate the performance of HH_CF is statistically different from the performance of other algorithms. We use the following notation. Given two algorithms P and Q, P:Q $+$ $(-)$ indicates that P performs better/worse than Q on average and this performance difference is statistically significant. The ~ sign indicates that both algorithms deliver a similar performance. The notation n/a means the *t*-test is not applicable since the performances of both algorithms are completely equal.

5.2.2 Experimental Settings

All experimental parameters are chosen accordingly to that commonly used in the literature for continuous problems. Nine test problems for the WFG suite (WFG1-WFG9) have 24 real parameters including four position parameters, 20 distance parameters and two objectives. All settings for the test suite are fixed using the same settings proposed in the previous studies (Zitzler et al., 2000; Huband et al., 2006).

According to Voutchkov and Keane (2010) and Chow and Regan (2012), an algorithm could reach better convergence by 6,250 generations. Therefore, the HH_CF was terminated after 6,250 generations. That is, HH_CF runs for a total of 25 iterations (stages). In each iteration, one low level heuristic is applied and this is executed for 250 generations with a population size equal to 100. The secondary population of SPEA2 is set to 100. The execution time takes about 10-30 minutes depending on the given problem. In order to make a fair comparison, each low level heuristic is used in isolation and is terminated after 6,250 generations. For the WFG problems, 30 independent trials were run for each algorithm with a different random seed. For all three low level heuristics, the simulated binary crossover (SBX) operator is used for recombination and a polynomial distribution for mutation (Deb and Agrawal, 1995). The crossover and mutation probability were set to 0.9 and 1/24 respectively. The distribution indices for crossover and mutation were set to 10 and 20 respectively. In the measure of SSC, the reference points for WFG problems with k objectives was set $r_i = (0, i * 2), i = 1, \ldots, k$; (Huband et al., 2006). The distance sharing σ for the UD metric and MOGA was set to 0.01 in the normalised space. These settings were used for SSC and UD as a feedback indicator in the ranking scheme of HH_CF and as a performance measure for the comparison. All algorithms were implemented with the same common sub-functions using Microsoft Visual C++ 2008 on an Intel Core2 Duo 3GHz\2G\250G computer.

5.2.3 Tuning of α parameter

In our multi-objective hyper-heuristic framework, we employ a modified choice function, a selection mechanism using Equation 4.4 (see Section 4.3). The parameter value for α is important to strike a balance between f_1 and f_2

values, as they are not in the same scalar unit. However, the choice of the right value is not trivial. We conducted initial experiments to determine the right value of α that leads to obtain solutions with good quality. In this experiment, we used three values in different ranges (small, middle and large (10,100 and 1000) respectively). Four instances of the WFG with two objectives (WFG1, WFG4, WFG6 and WFG8) are selected as they require a varied execution time ranging approximately between 10-25 minutes and they are run 30 times.

The performance values of HH_CF using the different values of α (10, 100 and 1000) with respect to the performance metrics (RNI, SSC and UD) on the selected WFG problems are summarised in Table 5.1. For each performance metric, the average, minimum, maximum and standard deviation values are computed. A higher value indicates a better performance. We can observe that HH_CF has the highest (best) average of RNI when α=1000. However, HH_CF has the highest (best) averages of SSC and UD metrics when α=100. We note that HH_CF has the worst performance with respect to three metrics when α=10. These results can be explained by answering some questions, what is a good balance between $f_1(h)$ and $f_2(h)$ to reach in a satisfactory level (i.e. producing good solutions). How does intensification and diversification affect the quality of the solutions during the search? In the case of a small α, more attention is given to $f_2(h)$ and to the diversification factor as well. Thus, no consideration for $f_1(h)$ and the intensification factor. The choice function acts as a random selection method; a low level heuristic (h) is invoked regardless of its performance; the learning mechanism is not effective. In contrast, a large α gives more focus to $f_1(h)$ and for the intensification factor as well. The low level heuristic (h) with the best performance is always invoked during the search and no other low level heuristics are considered. An example of this, let's say $f_1(h)$=6 and $f_2(h) =$ 90.718 seconds, Based on this, the selection of the heuristics relies on $f_2(h)$ when α=10 while it relies on $f_1(h)$ when α=1000. In case of α=100, a balance between $f_1(h)$ and $f_2(h)$ can be made. In the first few iterations of the search, the intensification factor gives a low level heuristic, that performs well, a chance to exploit the search area. Then as f_2 increases during the search, the selection method invokes a low level heuristic which is not currently performing well, in order to explore unvisited search areas. The value changing between $f_1(h)$ and $f_2(h)$ leads to a balance between intensification and diversification. In Figures 5.1 and 5.2, we provide an example of this situation for WFG1. In Figure 5.1, the average performance values of RNI, SSC and UD metrics for the HH_CF during the search with different settings α=(10,100, 1000) on WFG1 is visualised. Also the average heuristic utilisation rate which indicates how frequently a given heuristic is chosen and applied during the whole search process across all runs on WFG1 for the HH_CF with different α values is computed and illustrated in Figure 5.2.

From both Figures 5.1 and 5.2, we note that the performance of HH_CF during the search when α=10 with respect to RNI is reduced, and it fluctuated with respect to the SSC and UD metrics. This is due to the absence of the intensification factor and the strong effect of the diversification factor on the algorithm which result in the heuristics being called (almost randomly). The performance of HH_CF during the search when α=1000 with respect to the three metrics is relatively the same. Although the performance of HH_CF has slightly increased during the search and it does obtain better solutions, diversification factor is not having any effect. This is clear in Figure 5.2, where NSGAII has the highest utilisation rate as it performs well and MOGA have not been executed at all. This is because of the effect of the intensification factor. However, the performance of HH_CF during the search when α=100, with respect to the three metrics, is reflecting the good balance between intensification and diversification. In Figure 5.2, HH_CF with α=100 shows a heuristic with the best performance for many iterations because of the effect of the intensification factor, but it also gives a chance for other heuristics to be called because of the diversification factor. This is shown in Figure 5.1, all heuristics are invoked even if they do not perform well. From the above observations, α=100 is the best value compared to the others that obtains better solutions for HH_CF on selected WFG problems. Therefore, α is set to100 for our HH_CF in the experiments that are presented in the rest of this chapter.

WFG	α	RNI				SSC (HV)				UD			
		AVG	MIN	MAX	STD	AVG	MIN	MAX	STD	AVG	MIN	MAX	STD
1	10	0.1080	0.0400	0.2000	0.0593	3.1031	0.3251	8.11719	3.1183	0.3873	0.3364	0.7656	0.1401
	100	0.8800	0.2800	1.0000	0.2539	**12.1386**	9.0338	12.5130	0.9101	**0.4428**	0.3490	0.6945	0.1007
	1000	**1.0000**	1.0000	1.0000	**0.0000**	10.5048	6.4887	10.5168	**0.0113**	0.4101	0.3890	0.4284	**0.0152**
4	10	0.2340	0.2000	0.2500	0.1949	9.1308	8.7872	9.3591	0.2104	0.4932	0.4798	0.5505	0.0894
	100	0.5443	0.4800	0.6400	0.0452	**9.6588**	9.5331	9.6643	0.0176	**0.5596**	0.4752	0.6317	**0.0361**
	1000	**1.0000**	1.0000	1.0000	**0.0000**	9.6510	9.5000	9.6632	**0.0038**	0.4118	0.3955	0.4379	0.1574
6	10	0.2800	0.1600	0.2800	0.0438	8.8502	7.9699	9.1721	0.4976	0.5088	0.6231	0.7646	0.0544
	100	0.4720	0.4000	0.5600	0.0412	**9.3687**	9.1500	9.3810	**0.0542**	**0.5962**	0.5042	0.6479	0.0363
	1000	**1.0000**	1.0000	1.0000	**0.0000**	9.3346	9.2759	9.4105	0.0695	0.4155	0.3992	0.4337	**0.0135**
8	10	0.0640	0.0400	0.0800	**0.0219**	6.8731	5.0792	7.5603	7.5603	06668	0.5358	0.7387	0.8315
	100	0.2627	0.2000	0.4400	0.0454	**8.3033**	8.1155	8.5676	0.1224	**0.7886**	0.6294	1.0000	0.2627
	1000	**0.9000**	0.4000	1.0000	0.3000	7.6730	7.5321	7.7276	**0.0797**	0.4772	0.4125	0.6948	**0.1218**

Table 5.1: The Performance of multi-objective choice function based hyper-heuristic (HH_CF) using different values of α parameter in the choice function selection method.

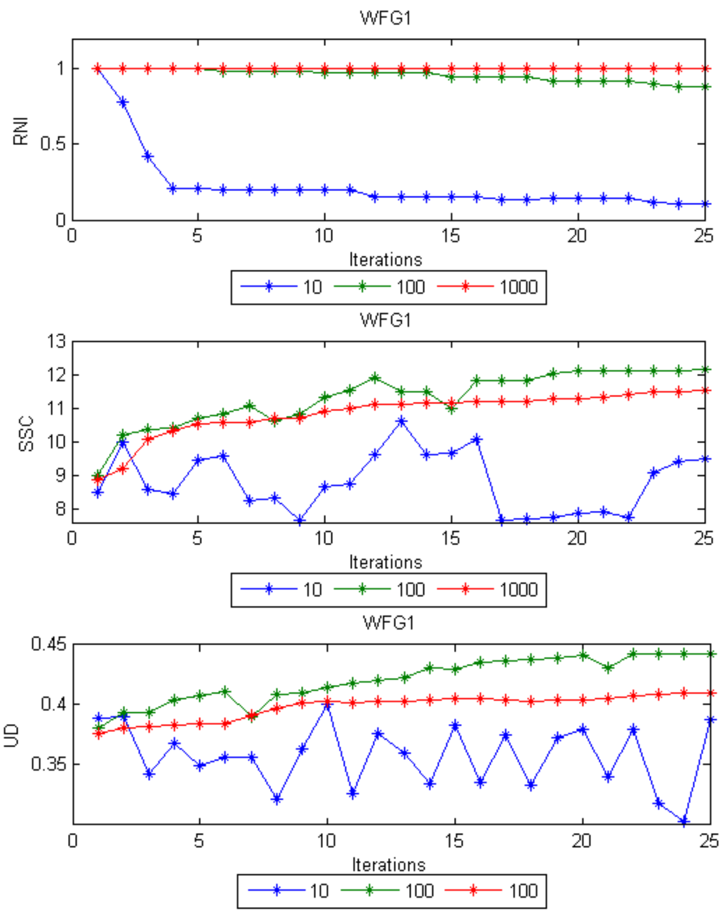


Figure 5.1: The performance of HH_CF with respect to the measure RNI, SSC and UD during the search which were averaged over 30 trials for different α settings (10, 100, 1000) on WFG1.

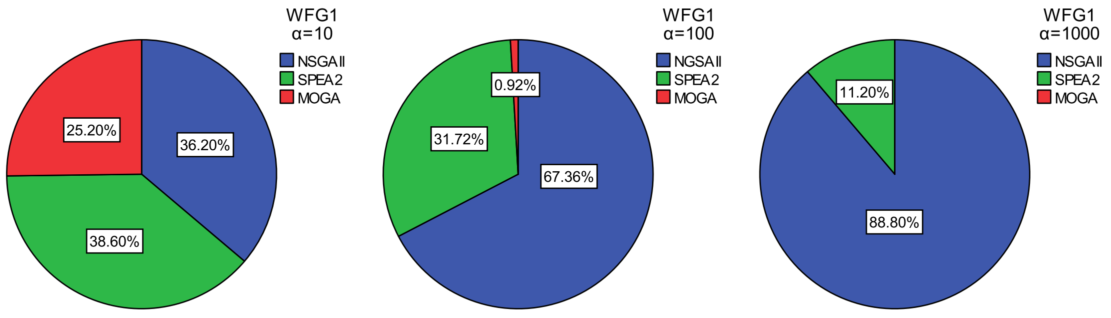


Figure 5.2: The average heuristic utilisation rate over 30 trials for the low level heuristics (NSGAII, SPEA2 and MOGA) in HH_CF using different α settings (10, 100, 1000) on the WFG1.

5.2.4 Comparison Results and Discussion

NSGAII, SPEA2, MOGA and HH_CF are tested on the nine WFG test problems under the same experimental settings described in Section 5.2.2. Table 5.2 summarises the average, minimum, maximum and standard deviation values pairs for each algorithm with respect to RNI, SSC and UD over 30 trials. For all performance metrics, a higher value indicates a better performance. HH_CF has a higher RNI value than MOGA while it has a lower value than NSGAII and SPEA2 for WFG1. HH_CF has the highest value of SSC and UD metrics among the methods. We can put WFG5 and WFG6 in this category. For WFG2 and WFG3, HH_CF has a RNI value similar to MOGA and lower than the others. With respect to SSC, HH_CF has higher values than SPEA2 and MOGA and similar to NSGAII. However, HH_CF has the highest value among other methods in the measure of UD. For WFG4 and WGF7, HH_CF has the lowest (worst) RNI value and the highest UD value. HH_CF has a higher value than MOGA and similar to NSGAII and SPEA2 with respect to the SSC metric. For WFG8 and WFG9, HH_CF has the lowest value with respect to RNI and SSC metrics, and the highest value with respect to UD metric.

These performance results with respect to RNI, SSC and UD are also displayed as box plots in Figures 5.3, 5.4 and 5.5 in order to provide a clear visualisation of the distribution of the simulation data of the 30 independent runs. The statistical *t*-test comparing our proposed HH_CF and the three low level heuristics (NSGAII, SPEA2 and MOGA), when used in isolation for the three performance metrics (RNI, SSC and UD) are given in Table 5.3. We can note that HH_CF and the other algorithms are statistically different in the majority cases.

In Figure 5.3, NSGAII and SPEA2 perform better than the others and produce the highest value of RNI for all datasets. This performance variation is statistically significant as illustrated in Table 5.3. Moreover, NSGAII and SPEA2 perform the same across all benchmarks with respect to RNI. However, HH_CF and MOGA produce relatively low values for this metric. HH_CF performs significantly better than MOGA on two instances of WFG1 and WFG5 and vice-versa for two instances of WFG8 and WFG9. For the rest of the instances, they deliver the same performance. This indicates that HH_CF performs badly according to the metric of RNI and produces a low number of non-dominated solutions than other algorithms, except for MOGA.

In Figure 5.4, the performance of HH_CF for SSC is relatively better than SPEA2 and MOGA across all test problems except for WFG9. HH_CF performs significantly better than SPEA2 and MOGA on eight instances of WFG (see Table 5.3). HH_CF also performs better than NSGA2 in WFG1, WFG5 and WFG6. This performance variation is statistically significant as illustrated in Table 5.3. HH_CF performs significantly better than NSGAII on three instances of WFG1 and WFG5, WFG6.

In Figure 5.5, it can be seen that HH_CF has the highest uniform distribution UD value across all test problems. This indicates that HH_CF is superior to the other algorithms on all WFG instances in terms of the distribution of non-dominated individuals over the POF. This performance variation is statistically significant as illustrated in Table 5.3. HH_CF performs significantly better than the other methods on all nine instances of WFG. Although HH_CF performs similarly to NSGAII in WFG2, WFG3, WFG4 and WFG7, HH_CF performs significantly slightly better than NSGAII on three instances of WFG2, WFG4 and WFG7 (see Tables 5.2 and 5.3). For WFG8 and WFG9, HH_CF does not perform well compared to the others, except MOGA. HH_CF performs significantly worse than NSGAII and SPEA2 where HH_CF performs significantly better than MOGA as shown in Table 5.3.

We note from all the above results that HH_CF performs worse than the low level heuristics when used in isolation with respect to the RNI metric, and it produces a lower number of non-dominated solutions for most of the WFG problems. However, HH_CF performs very well and produces non- dominated solutions that distribute uniformly well over the POF with respect to the UD metric when compared to the other methods. HH_CF also performs better than the others in most of the WFG problems and produces non-dominated solutions with high diversity that cover a larger proportion of the objective space with respect to the SSC metric, except for WFG8 and WFG9 where it failed to converge towards the POF. As WFG8 and WFG9 have a significant bias feature, HH_CF may have difficulties coping with bias.

WFG	Methods	RNI				SSC (HV)				UD			
		AVG	MIN	MAX	STD	AVG	MIN	MAX	STD	AVG	MIN	MAX	STD
1	HH_CF	0.8800	0.2800	1.0000	0.2539	**12.1386**	9.0338	12.5130	0.9101	**0.4428**	0.3490	0.6945	0.1007
	NSGAII	**1.0000**	1.0000	1.0000	**0.0000**	11.6041	11.0016	12.3570	0.3880	0.4003	0.3727	0.4327	**0.0140**
	SPEA2	**1.0000**	1.0000	1.0000	**0.0000**	6.4931	6.4811	6.5063	**0.0066**	0.4099	0.3760	0.4420	0.0148
	MOGA	0.2650	0.1300	0.6300	0.1140	4.2184	3.5399	6.3178	0.6727	0.2117	0.1535	0.3718	0.0478
2	HH_CF	0.2293	0.1600	0.3600	0.0545	**11.0219**	10.6407	12.3894	0.3042	**0.7278**	0.6223	1.0000	0.0661
	NSGAII	**1.0000**	1.0000	1.0000	**0.0000**	10.8199	10.8057	10.8249	**0.0041**	0.3747	0.3497	0.3988	**0.0112**
	SPEA2	**1.0000**	1.0000	1.0000	**0.0000**	10.7898	10.2636	11.9569	0.7935	0.2874	0.2217	0.3488	0.0305
	MOGA	**1.0000**	1.0000	1.0000	**0.0000**	9.7959	7.1533	10.1943	0.6978	0.5414	0.4294	0.6910	0.0597
3	HH_CF	0.6027	0.5200	0.6800	0.0445	11.8940	11.3990	11.9867	0.0853	**0.5450**	0.4959	0.6136	0.0289
	NSGAII	**1.0000**	1.0000	1.0000	**0.0000**	**11.9185**	11.9046	11.9306	**0.0063**	0.4244	0.3980	0.4448	0.0120
	SPEA2	**1.0000**	1.0000	1.0000	**0.0000**	11.4062	11.3664	11.4541	0.0189	0.4289	0.4110	0.4436	**0.0078**
	MOGA	0.6070	0.5200	0.9600	0.0400	11.2921	10.9930	11.4508	0.1393	0.4468	0.3819	0.5116	0.0324
4	HH_CF	0.5443	0.4800	0.6400	0.0452	**9.6588**	9.5331	9.6643	0.0176	**0.5596**	0.4752	0.6317	0.0361
	NSGAII	**1.0000**	1.0000	1.0000	**0.0000**	9.6460	9.6518	9.6683	**0.0041**	0.4132	0.3879	0.4402	0.0151
	SPEA2	**1.0000**	1.0000	1.0000	**0.0000**	9.1853	9.1599	9.2091	0.0133	0.4058	0.3725	0.4301	**0.0133**
	MOGA	0.5800	0.4900	0.7100	0.0540	8.9968	8.4897	9.3057	0.2056	0.4594	0.3940	0.5610	0.0387
5	HH_CF	0.8537	0.6000	1.0000	0.1723	**9.2899**	9.1526	9.2984	0.5744	**0.4779**	0.4279	0.5744	0.0468
	NSGAII	**1.0000**	1.0000	1.0000	**0.0000**	9.2857	9.2672	9.2904	**0.0043**	0.3958	0.3705	0.4271	0.0129
	SPEA2	**1.0000**	1.0000	1.0000	**0.0000**	9.2860	9.1952	9.2968	0.0214	0.4360	0.4222	0.4538	**0.0087**
	MOGA	0.6820	0.6000	0.7400	0.0360	8.8946	8.4904	9.1028	0.4171	0.4184	0.3583	0.4690	0.0272
6	HH_CF	0.4720	0.4000	0.5600	0.0412	**9.3687**	9.1500	9.3810	**0.0542**	**0.5962**	0.5042	0.6479	0.0363
	NSGAII	**1.0000**	1.0000	1.0000	**0.0000**	9.3503	9.1883	9.4401	0.0605	0.4082	0.3091	0.4479	0.0247
	SPEA2	**1.0000**	1.0000	1.0000	**0.0000**	8.7135	8.4494	9.0349	0.1851	0.3761	0.3461	0.4068	**0.0158**
	MOGA	0.4990	0.4300	0.5900	0.0420	8.8878	8.5542	9.0785	0.1345	0.4786	0.3929	0.5712	0.0367
7	HH_CF	0.6173	0.4000	0.7200	0.0653	**9.6606**	9.2261	9.6911	0.0926	**0.5289**	0.4734	0.6743	0.0416
	NSGAII	**1.0000**	1.0000	1.0000	**0.0000**	9.6579	9.5053	9.6704	**0.0294**	0.4048	0.3766	0.4220	0.0117
	SPEA2	**1.0000**	1.0000	1.0000	**0.0000**	9.2481	9.2109	9.2724	0.0161	0.4082	0.3777	0.4333	**0.0116**
	MOGA	0.6300	0.5100	0.7600	0.0550	9.1685	8.6489	9.3474	0.1799	0.4331	0.3539	0.4980	0.0415
8	HH_CF	0.2627	0.2000	0.4400	0.0454	8.3033	8.1155	8.5676	0.1224	**0.7886**	0.6294	1.0000	0.1245
	NSGAII	**1.0000**	1.0000	1.0000	**0.0000**	**8.7155**	8.6912	8.7391	**0.0140**	0.4178	0.3980	0.4404	0.0123
	SPEA2	**1.0000**	1.0000	1.0000	**0.0000**	8.3957	8.3509	8.4412	0.0199	0.4069	0.3907	0.4226	**0.0083**
	MOGA	0.4790	0.4000	0.6000	0.0460	8.0762	7.4237	8.9192	0.2777	0.4490	0.3679	0.5644	0.0450
9	HH_CF	0.6410	0.4000	0.8000	0.0896	8.6132	8.2356	9.2519	0.2236	**0.5142**	0.4141	0.6432	0.0525
	NSGAII	**1.0000**	1.0000	1.0000	**0.0000**	**8.7650**	8.5787	9.2673	0.2960	0.3955	0.3641	0.4294	0.0163
	SPEA2	**1.0000**	1.0000	1.0000	**0.0000**	8.7091	8.5700	9.0416	**0.1967**	0.4303	0.4031	0.4488	**0.0106**
	MOGA	0.8260	0.6700	0.9700	0.0900	8.5723	8.2357	8.9845	0.2259	0.3693	0.2803	0.4257	0.0350

Table 5.2: The average performance of HH_CF compared to the low level heuristics on the WFG test problems with respect to the ratio of non-dominated individuals (RNI), the hypervolume (SSC) and the uniform distribution (UD).

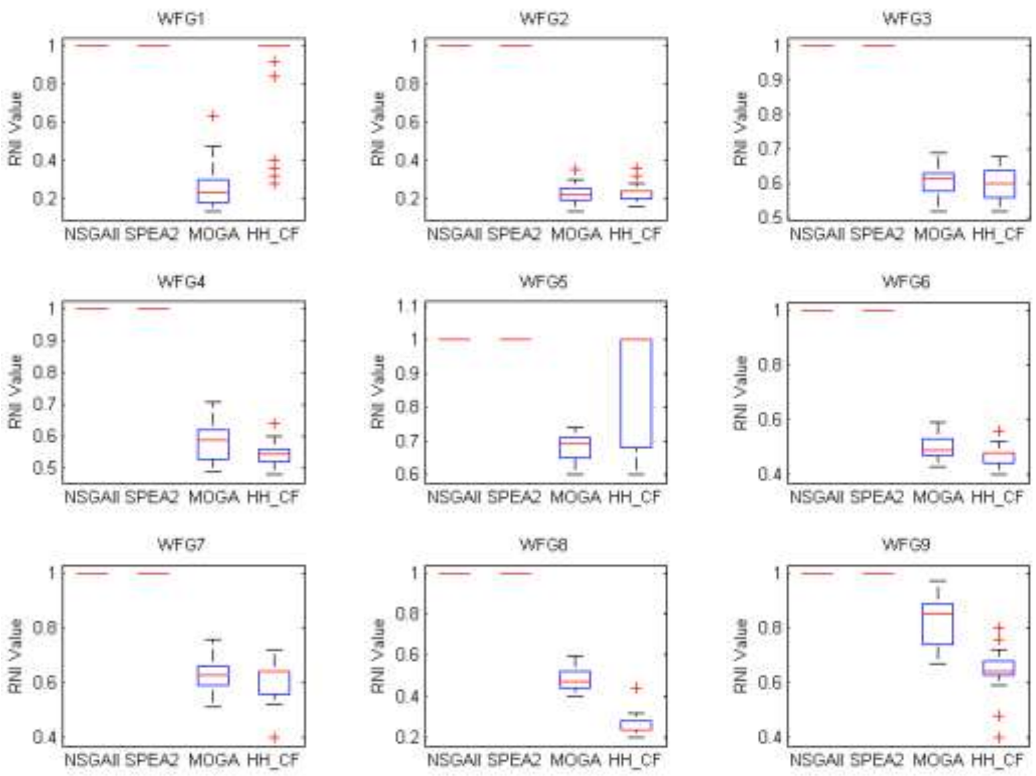


Figure 5.3:.Box plots of NSGAII, SPEA2, MOGA and HH_CF, for the measure of ratio of non-dominated individuals (RNI) on the WFG test functions.

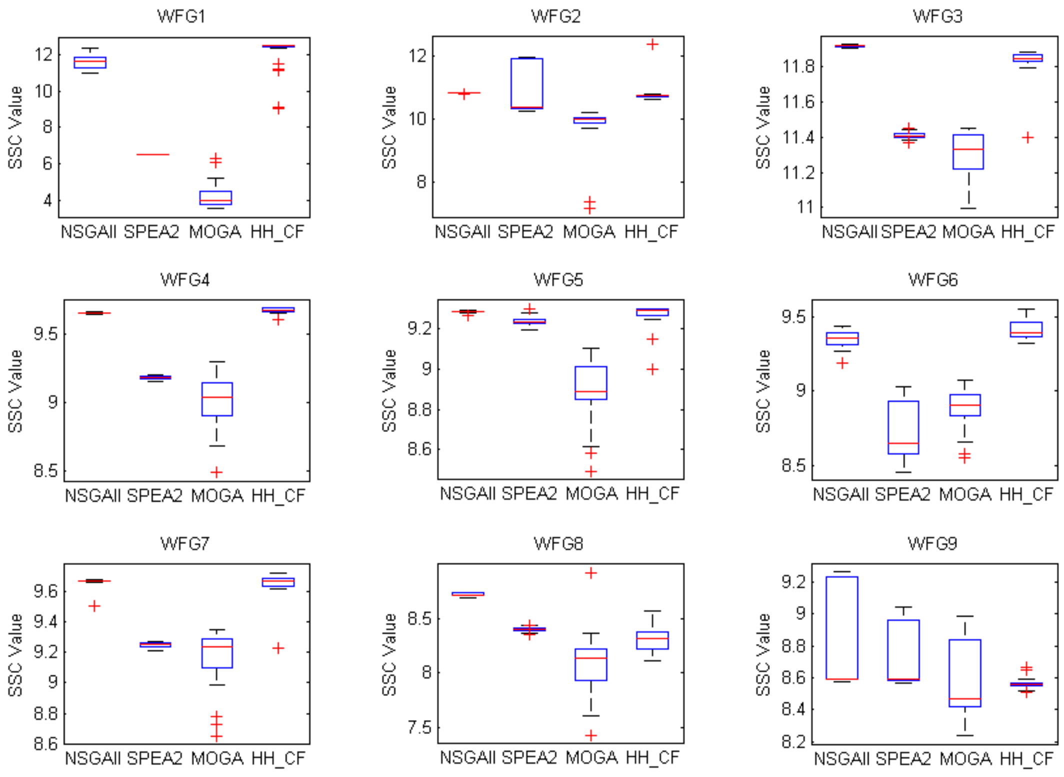


Figure 5.4: Box plots of NSGAII, SPEA2, MOGA and HH_CF for the measure of hypervolume (SSC) on the WFG test functions.

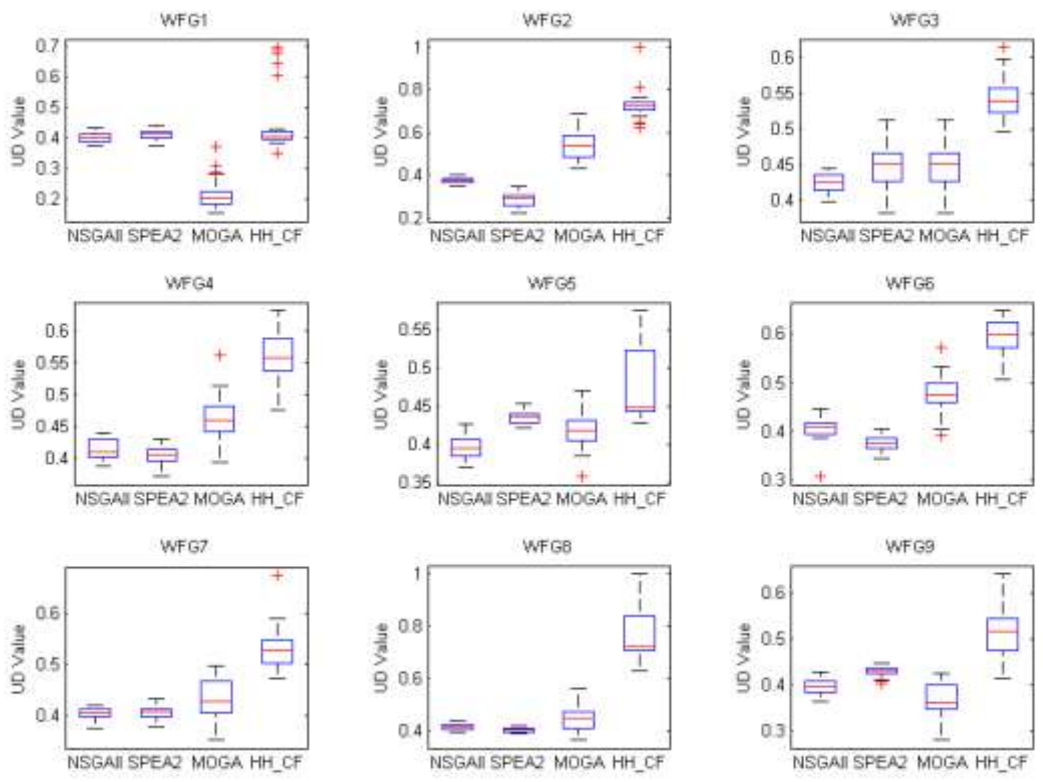


Figure 5.5: Box plots of NSGAII, SPEA2, MOGA and HH_CF for the uniform distribution (UD) of non-dominated population on the WFG test functions.

Generally, HH_CF produces competitive results across most of the WFG problems with respect to two of the performance metrics (SSC and UD) out of the three metrics. Although HH_CF obtains a low number of solutions, it produces very good solutions in terms of diversity and convergence when compared to the low level heuristics when used in isolation. HH_CF can benefit from the strengths of the low level heuristics. Moreover, it has the ability to intelligently adapt to calling combinations of low level heuristics. To understand how the HH_CF could obtain these results, we analyse the behaviour of the low level heuristics in the next sub-section.

5.2.5 Behaviour of Low Level Heuristics

We compute the average heuristic *utilisation rate* which indicates how frequently a given low level heuristic is chosen and applied during the search process, across all runs, in order to see which low level heuristic is used more frequently. The results are presented in Figure 5.6. The average heuristic utilisation rate of NSGAII is at least 44% and is the highest among all the low level heuristics for each problem, except for WFG5 for which SPEA2 is chosen

Problem	Methods	Metrics		
		RNI	SSC	UD
WFG1	HH_CF:NSGAII	-	+	+
	HH_CF:SPEA2	-	+	+
	HH_CF:MOGA	+	+	+
	NSGAII:SPEA2	n/a	+	-
	NSGAII:MOGA	+	+	+
	SPEA2:MOGA	+	+	+
WFG2	HH_CF:NSGAII	-	~	+
	HH_CF:SPEA2	-	+	+
	HH_CF:MOGA	~	+	+
	NSGAII:SPEA2	n/a	~	+
	NSGAII:MOGA	+	+	-
	SPEA2:MOGA	+	+	-
WFG3	HH_CF:NSGAII	-	~	+
	HH_CF:SPEA2	-	+	+
	HH_CF:MOGA	~	+	+
	NSGAII:SPEA2	n/a	+	+
	NSGAII:MOGA	+	+	-
	SPEA2:MOGA	+	+	-
WFG4	HH_CF:NSGAII	-	~	+
	HH_CF:SPEA2	-	+	+
	HH_CF:MOGA	-	+	+
	NSGAII:SPEA2	n/a	+	+
	NSGAII:MOGA	+	+	-
	SPEA2:MOGA	+	+	-
WFG5	HH_CF:NSGAII	-	+	+
	HH_CF:SPEA2	-	+	+
	HH_CF:MOGA	+	+	+
	NSGAII:SPEA2	n/a	+	-
	NSGAII:MOGA	+	+	-
	SPEA2:MOGA	+	+	+
WFG6	HH_CF:NSGAII	-	+	+
	HH_CF:SPEA2	-	+	+
	HH_CF:MOGA	~	+	+
	NSGAII:SPEA2	n/a	+	+
	NSGAII:MOGA	+	+	-
	SPEA2:MOGA	+	-	-
WFG7	HH_CF:NSGAII	-	~	+
	HH_CF:SPEA2	-	+	+
	HH_CF:MOGA	~	-	+
	NSGAII:SPEA2	n/a	+	~
	NSGAII:MOGA	+	+	-
	SPEA2:MOGA	+	+	-
WFG8	HH_CF:NSGAII	-	-	+
	HH_CF:SPEA2	-	-	+
	HH_CF:MOGA	-	+	+
	NSGAII:SPEA2	n/a	+	+
	NSGAII:MOGA	+	+	-
	SPEA2:MOGA	+	+	-
WFG9	HH_CF:NSGAII	-	-	+
	HH_CF:SPEA2	-	-	+
	HH_CF:MOGA	-	+	+
	NSGAII:SPEA2	n/a	+	-
	NSGAII:MOGA	+	+	+
	SPEA2:MOGA	+	+	+

Table 5.3: The *t*-test results of HH_CF and low level heuristics on the WFG test problems with respect to the ratio of non-dominated individuals (RNI), the hypervolume (SSC) and the uniform distribution (UD).

most frequently with a utilisation rate of 55.72% during the search process. It explains why HH_CF has either a similar or relatively better convergence to the POF for most of the test problems when compared with NSGAII. It indicates that NSGAII performs best among other low level heuristics in most of the WFG problems. The authors theorise that HH_CF, therefore, prefers NSGAII and it becomes preferable to be chosen more frequently than the other low level heuristics. Our result is consistent with the result in Bradstreet et al. (2007) that shows that the best performance is achieved by NSGAII on the WFG test functions with two objectives.

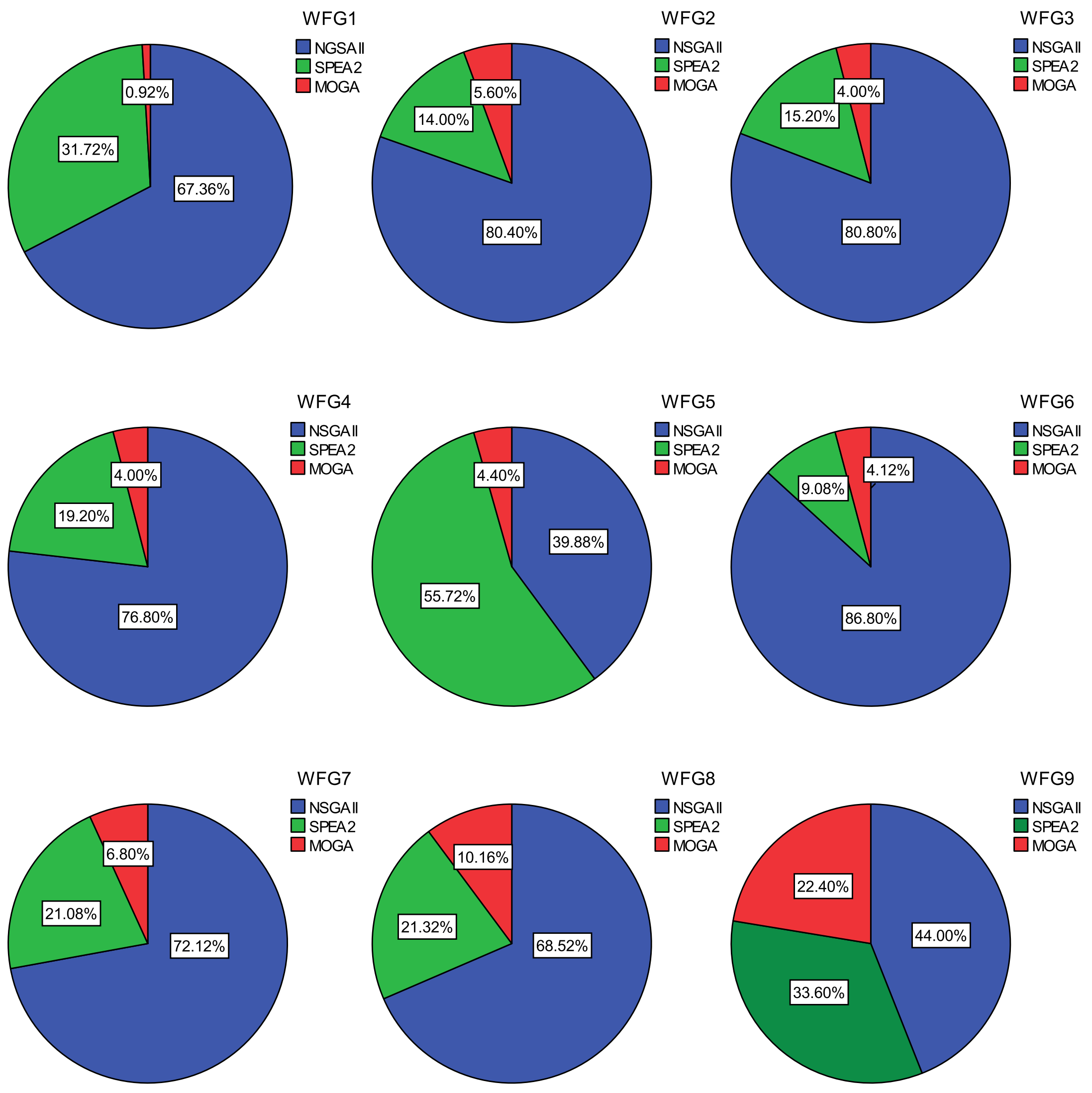


Figure 5.6: The average heuristic utilisation rate for the low level heuristics (NSGAII, SPEA2 and MOGA) in HH_CF on the WFG test suite.

The performance of MOGA is not that good on the WFG benchmark, thus it is invoked relatively less frequently during the search process because of the diversification factor f_2 in the selection choice function method (see Sections 4.1 and 4.3). However, MOGA still influences the performance of HH_CF, negatively, in particular with respect to the ratio of number of non-dominated individuals (RNI). This is due to that fact that MOGA does not have any archive mechanism or preserving strategy to maintain the non-dominated solutions during the search. Although the selection choice function method provides a kind of balance between the intensification (f_1) and diversification (f_2) when choosing a heuristic, HH_CF obtains a low ratio of non-dominated individuals (RNI) which indicates poor diversification. This is because of our multi-objective hyper-heuristics do not incorporate any archive mechanisms to maintain the non-dominated solutions during the search. So when MOGA is called, it produces a low number of non-dominated individuals, leading to poor diversification. The average utilisation rate of MOGA is the highest for WFG8 (10.16%) and WFG9 (22.40%) among other WFG problems. This utilisation rate explains why the performance of HH_CF is the worst performing approach in terms of RNI. HH_CF also faces some difficulty while solving WFG8 and WFG9 in terms of convergence as well.

In order to see the effectiveness of each chosen low level heuristic on the performance of HH_CF, we looked into the performance of the low level heuristics with respect to the RNI, SSC and UD metrics at twenty five decision points during the search process. We observe that some problems are following a specific pattern to invoke the low level heuristics during the search. Each problem has its own pattern. For example, for WFG3, NSGAII is invoked and executed for the first seven consecutive decision points. Then SPEA2 is invoked for the next four decision points, followed by one iteration of MOGA. Then NSGAII is chosen for the rest of the search. More of these patterns are illustrated in Figure 5.7.

In order to analyse these results, we divide the WFG instances into four categories based on the performance of HH_CF compared to the three low level heuristics being used in isolation with respect to RNI, SSC and UD as listed below:

(i) WFG1,WFG5 and WFG6:

- RNI: Better performance than MOGA and worse than NSGAII and SPEA2
- SSC: The best performance among NSGAII, SPEA2 and MOGA
- UD: The best performance among NSGAII, SPEA2 and MOGA

(ii) WFG2 and WFG3:

- RNI: Similar performance to MOGA and worse than NSGAII and SPEA2
- SSC: Better performance than SPEA2 and MOGA and similar to NSGAII
- UD: The best performance among NSGAII, SPEA2 and MOGA

(iii) WFG4 and WGF7:

- RNI: The worst performance among NSGAII, SPEA2 and MOGA
- SSC: Better performance than SPEA2 and MOGA and similar to NSGAII
- UD: The best performance among NSGAII, SPEA2 and MOGA

(iv) WFG8 and WFG9:

- RNI: The worst performance among NSGAII, SPEA2 and MOGA
- SSC: The worst performance among NSGAII, SPEA2 and MOGA
- UD: The best performance among NSGAII, SPEA2 and MOGA

For each category described above, except the last one, we have selected a sample problem to visualise the low level call patterns. WFG5 for the first category, WFG3 for the second category and WFG4 for the third category. For the last category, no specific pattern has been observed. The selected three problems have different problems features in terms of separability and modality (Huband et al., 2006). The average of RNI, SSC and UD values versus decision point plots across selected benchmark problems (WFG3, WFG4 and WFG5) are shown in Figure 5.7. Each step in the plot is associated with the most frequently selected low level heuristics across 30 trials. Since we employed All-Moves as an acceptance strategy, some moves are accepted even if it worsens the solution quality.

From Figure 5.7, it is clear that MOGA, during the search, produces a worse solution with respect to RNI, and this solution is accepted which affects the performance of HH_CF. However, some worsening moves are able to produce better solutions. This can be noted in the performance HH_CF with respect to the UD metric. SPEA2 produces low quality solutions in terms of the distribution along the POF, but this helps it to escape from the local optimum and obtain better solutions at the end. This is also true with respect to the SSC performance indicator. In addition, we note that HH_CF has an advantage over MOGA and outperforms the three MOEAs methods with respect to the distribution of non-dominated individuals over the POF. It also has an advantage over NSGAII in terms of convergence, in that it performs better than all other methods in some problems while performing better or similar to NSGAII on the other problems. However, HH_CF does not have an advantage over NSGAII and SPEA2 with respect to the non-dominated individuals in the population. HH_CF performs poorly because of MOGA's effect.

It can be concluded that our choice function based hyper-heuristic can benefit from the strengths of the low level heuristics. And it can avoid the weaknesses of them (partially), as the poor performance of MOGA affects the performance of HH_CF badly in the metric of RNI by producing a low number of non-dominated individuals. We can avoid this by employing another acceptance move strategy instead of All-Moves. A non-deterministic acceptance strategy could accept worsening moves within a limited degree and help improve the quality of the solutions. However, HH_CF has the ability to intelligently adapt to calling combinations of low level heuristics.

5.3 Performance Comparison of Multi-objective Choice Function Based Hyper-heuristic to the Other Multi-objective Approaches

We conduct some experiments to examine the performance of our proposed multi-objective choice function based hyper-heuristic (HH_CF) compared to two multi-objective approaches; a random hyper-heuristic (HH_RAND) and the adaptive multi-method search (AMALGAM) (Vrugt and Robinson, 2007). In a random hyper-heuristic (HH_RAND), we employ a simple random selection instead of the choice function selection this is used in HH_CF. No ranking scheme, nor a learning mechanism, is embedded into

HH_RAND. In HH_RAND, we use the same three low level heuristics that are used in HH_CF.

5.3.1 Performance Evaluation Criteria

The hypervolume (SSC) (Zitzler and Thiele, 1999), the generational distance (GD) (Van Veldhuizen and Lamont, 1998b) and the inverted generational distance (IGD) (Coello and Cruz Cortès, 2005) metrics were used to compare the performance of multi-objective approaches for this set of experiments. The GD and IGD measure the distance (convergence) between the approximation non-dominated front and the POF. A smaller value of GD and IGD is more desirable and it indicates that the approximation non-dominated front is closer to the POF. In addition, we use *t*-test for the average performance comparison of algorithms and the results are discussed using the same notation as provided in Section 5.2.1.

5.3.2 Experimental Settings

All experimental parameters are chosen to be the same as those commonly used in the scientific literature for continuous problems (Zitzler et al., 2000; Huband et al., 2006). All methods were applied to the nine WFG test problems with 24 real values and two objectives. In order to keep the computational costs of the experiments to an affordable level, all the methods were executed for 25,000 evaluation functions with a population size of 100 and 250 generations in each run. Depending on the given problem, the execution time of HH_CF and HH_RAND for one run takes about 5-12 minutes. Both HH_CF and HH_RAND are executed for 2,500 evaluation functions at each iteration. Other parameter settings of AMALGAM are identical to those used in Vrugt and Robinson (2007). We used the Matlab implementation of AMALGAM obtained from the authors via personal communication. We implemented a C++ interface between AMALGAM and the WFG test suite's C++ code. All other experimental settings are fixed the same as discussed in Section 5.2.2.

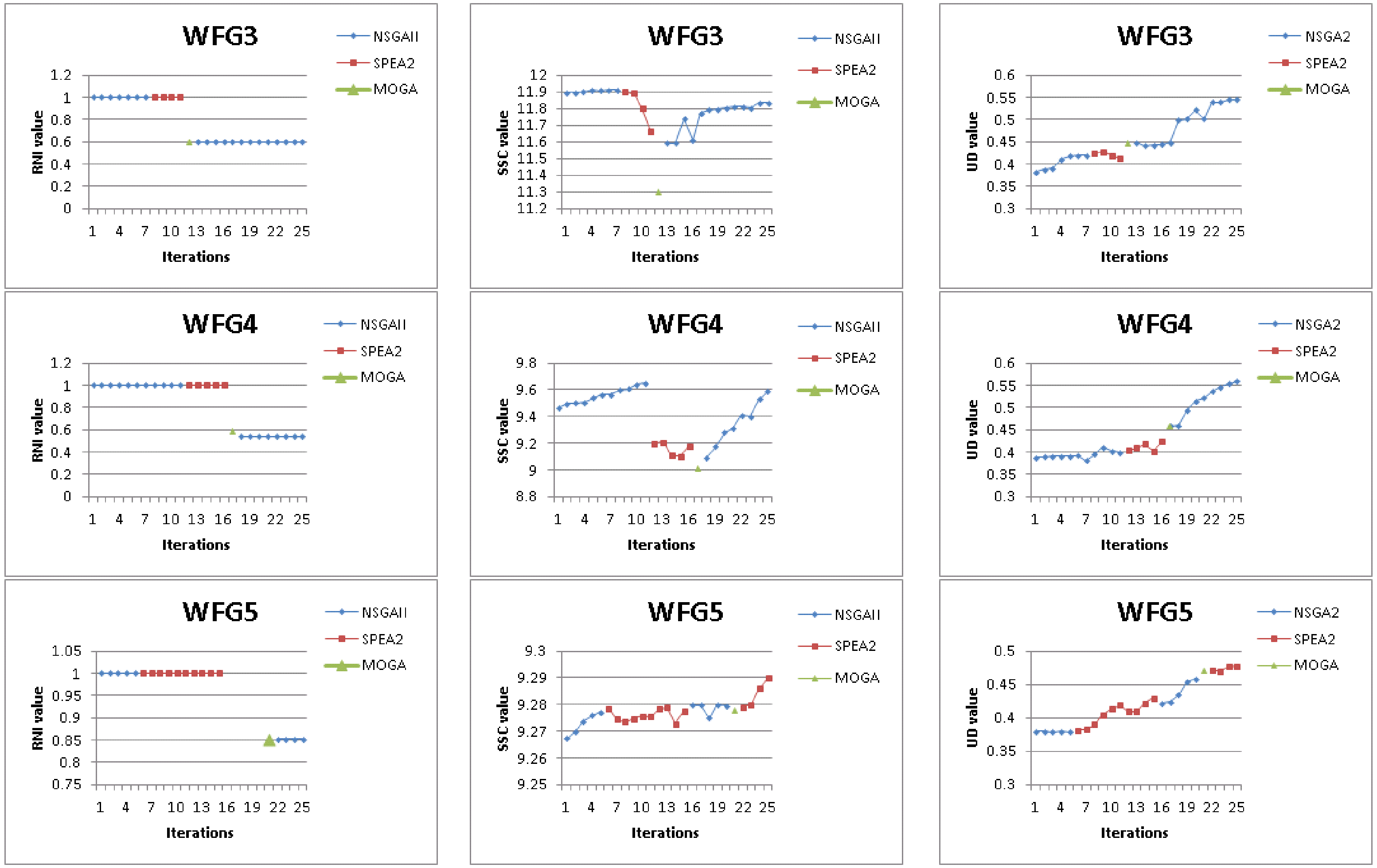


Figure 5.7: The average of RNI,SSC and UD values versus decision point steps plots across selected benchmark problems (the WFG3, WFG4 and WFG5). Each step in the plot is associated with the most frequently selected low level heuristics across 30 trial

5.3.3 Experimental Results and Discussion

The performance values of HH_CF and the other hyper-heuristic methods with respect to the performance metrics SSC, GD and IGD on the WFG problems are summarised in Table 5.4. For each performance metric, the average, minimum, maximum and standard deviation values are computed.

These performance results with respect to SSC, GD and IGD are also displayed as box plots in Figures 5.8, 5.9 and 5.10 in order to provide a visualisation of the distribution of the simulation data of the 30 independent runs. The statistical *t*-test comparing our proposed HH_CF and other multi-objective hyper-heuristics for the metrics (SSC, GD and IGD) are given in Table 5.5. The results show that the HH_CF performs better than the other algorithms in most cases. As expected, HH_CF achieves better coverage and diversity than HH_RAND according to both metrics. This is due to the learning mechanism used in HH_CF which adaptively guides the search towards the POF. Interestingly, HH_RAND performs better than AMALGAM according to the hypervolume metric except in WFG9. However, HH_RAND performs worse than AMALGAM according to the GD metric over all of the problems while it better in all problems with respect to IGD except in WFG9. This performance variation is statistically significant as illustrated in Table 5.5. HH_RAND performs significantly better than AMALGAM for the SSC metric on eight instances of WFG except in WFG9. HH_RAND also performs significantly better than AMALGAM for the IGD metric on all instances except in WFG9. HH_RAND also performs significantly better than AMALGAM for the GD metric on three instances of WFG1, WFG6 and WFG7 while it performs significantly similar to AMALGAM on one instance of WFG5 where it performs significantly worse than AMALGAM for the rest.

Compared to AMALGAM, HH_CF performs better with respect to the convergence and diversity on most of the WFG problems. According to the SSC metric, HH_CF produced non-dominated solutions that cover a larger proportion of the objective space than AMALGAM on all WFG problems except for WFG9. In Table 5.5, HH_CF performs significantly better than AMALGAM on eight instances of WFG except for WFG9 where AMALGAM performs significantly better than HH_CF on this instance. The superiority of HH_CF on the SSC metric is due to the stronger selection mechanism and the effective ranking scheme that relies on choosing a heuristic with the best SSC value at the right time (decision point) to guide the search to move toward more spaces around the POF. This result is more reliable as shown in Figure 5.8.

According to the metrics of GD and IGD, HH_CF is superior to AMALGAM on most of WFG problems as reported in Table 5.4 and displayed as box plots in Figure 5.9 and 5.10. In Table 5.5, HH_CF performs significantly better than AMALGAM on five instances out of nine including WFG1, WFG2, WFG5, WFG6, and WFG7 for the metric of GD. And HH_CF performs significantly better than AMALGAM on all instances except in WFG9 for the metric of IGD. Again, this result is due to the online-learning selection mechanism and the ranking scheme in HH_CF. The ranking scheme maintains the past performance of low level heuristics using a set of performance indicators that measure different aspects of the solutions. During the search process, the raking scheme creates a balance between choosing the low level heuristics and their performance according to a particular metric. This balance enhances the algorithm performance to yield better solutions that converge toward the POF as well as distribute uniformly along the POF. However, AMALGAM performs significantly better than HH_CF on the other four instances for GD and one instance for IGD (see Tables 5.4 and 5.5). This might be because of the nature of the problems that present difficulties for HH_CF to converge toward the POF or might slow down the convergence speed such as the bias in WFG8, WFG9 and the multimodality of WFG4. It is good to report that AMALGAM has better performance according to the both metrics; SSC, GD and IGD in WFG9. This is shown in Table 5.5, where AMALGAM performs significantly better than others on one instance of WFG9.

For each problem, we computed the 50% attainment surface for each algorithm, from the 30 fronts after 25,000 evaluation functions. In Figures 5.11 and 5.12, we have plotted the POF and the 50% attainment surface of the algorithms. HH_CF shows good convergence and uniform distribution for most datasets. It seems clear that HH_CF has converged well on the POF in WFG1 and WFG2 compared to other algorithms. Moreover, HH_CF produced solutions that covered larger proportions of the objective space compared to the other algorithms. AMALGAM has poor convergence most problems. It has fewer solutions with poor convergence for WFG2. And it has no solutions over the middle-lower segments of the POF for WFG3, WFG5, WFG6, WFG7, and WFG8 and no solutions over the upper-middle segments of the POF for WFG4.

WFG	Methods	SSC (HV)				GD				IGD			
		AVG	MIN	MAX	STD	AVG	MIN	MAX	STD	AVG	MIN	MAX	STD
1	HH_CF	**12.0044**	11.8430	12.2044	0.8301	**0.00774**	0.00340	0.04660	0.01106	**0.00102**	0.00039	0.00393	0.00098
	HH_RAND	7.0258	2.4467	7.5580	0.7877	0.02420	0.02899	0.03556	0.00143	0.00583	0.00340	0.00658	0.00078
	AMALGAM	7.7902	7.2863	8.2485	**0.1941**	0.02917	0.02620	0.03290	**0.00155**	0.00312	0.00276	0.00352	**0.00016**
2	HH_CF	**11.0102**	10.9907	11.2940	**0.2033**	**0.00046**	0.00090	0.00320	**0.00049**	**0.00051**	0.00022	0.00064	0.00008
	HH_RAND	9.7547	7.0023	9.7798	0.5078	0.01680	0.00031	0.04145	0.01089	0.00191	0.00123	0.00330	0.00058
	AMALGAM	1.7582	1.6036	6.1053	0.8210	0.00099	0.00030	0.01930	0.00346	0.00413	0.00412	0.00414	**0.00001**
3	HH_CF	**11.7550**	11.5650	11.8066	0.0743	0.00068	0.00030	0.00280	0.00045	**0.00075**	0.00068	0.00082	0.00005
	HH_RAND	11.0290	10.8800	11. 0833	0.1490	0.00384	0.00220	0.02252	0.00357	0.00081	0.00045	0.00100	0.00009
	AMALGAM	6.6890	6.6752	6.6980	**0.0049**	**0.00036**	0.00031	0.00041	**0.00002**	0.00272	0.00272	0.00272	**0.00000**
4	HH_CF	**9.5610**	9.5331	9.6700	0.0143	0.00097	0.00075	0.00151	0.00019	**0.00036**	0.00030	0.00043	**0.00003**
	HH_RAND	9.2052	8.7032	9.2991	0.0145	0.00405	0.00329	0.00499	0.00053	0.00066	0.00060	0.00072	0.00004
	AMALGAM	3.5687	3.5509	3.5838	**0.0075**	**0.00081**	0.00059	0.00070	**0.00005**	0.00194	0.00190	0.00200	**0.00003**
5	HH_CF	**9.2701**	8.7531	9.2954	0.5343	**0.00273**	0.00244	0.00333	0.00032	**0.00058**	0.00054	0.00069	**0.00003**
	HH_RAND	9.2577	9.2152	9.2784	0.0556	0.00255	0.00245	0.00269	0.00010	0.00066	0.00055	0.00077	0.00005
	AMALGAM	6.3554	6.2404	6.3766	**0.0323**	0.00281	0.00268	0.00381	**0.00028**	0.00126	0.00124	0.00137	**0.00003**
6	HH_CF	**9.3579**	9.0433	10.2011	0.0530	**0.00225**	0.00151	0.00391	**0.00056**	**0.00065**	0.00050	0.00078	**0.00007**
	HH_RAND	9.3119	9.1005	9.4231	0.0501	0.00334	0.00227	0.00452	0.00052	0.00077	0.00072	0.00080	0.00002
	AMALGAM	6.3554	6.2404	6.3766	**0.0323**	0.00298	0.00142	0.00554	0.00123	0.00193	0.00181	0.00217	0.00011
7	HH_CF	**9.6498**	9.2261	9.6540	0.0901	**0.00047**	0.00044	0.00136	0.00025	**0.00030**	0.00023	0.00037	0.00003
	HH_RAND	9.1184	8.1243	9.1685	0.3473	0.00425	0.00309	0.00582	0.00067	0.00037	0.00030	0.00041	0.00004
	AMALGAM	3.9171	3.9115	3.9263	**0.0035**	0.00067	0.00041	0.00051	**0.00003**	0.00345	0.00342	0.00347	**0.00001**
8	HH_CF	**8.2843**	8.0165	8.6621	0.1451	0.00442	0.00358	0.00498	0.00043	**0.00072**	0.00058	0.00088	0.00008
	HH_RAND	8.1089	7.0121	8.6760	0.3867	0.01140	0.00677	0.01510	0.00207	0.00086	0.00050	0.00100	0.00012
	AMALGAM	3.0945	3.0419	3.1293	**0.0213**	**0.00241**	0.00216	0.00281	**0.00017**	0.00243	0.00242	0.00245	**0.00001**
9	HH_CF	8.5981	8.2011	9.2660	0.2143	0.00528	0.00143	0.00639	0.00145	0.00183	0.00041	0.00218	0.00055
	HH_RAND	8.4697	8.1138	8.8453	0.3059	0.00602	0.00044	0.00755	0.00167	0.00337	0.00302	0.00395	0.00024
	AMALGAM	**9.0676**	8.6088	9.1480	**0.1140**	**0.00113**	0.00098	0.00156	**0.00011**	**0.00026**	0.00024	0.00032	**0.00002**

Table 5.4: The performance of HH_CF compared to multi-objective hyper-heuristics on the WFG test problems with respect to the Hypervolume (SSC), the generational distance (GD) and the inverted generational distance (IGD).

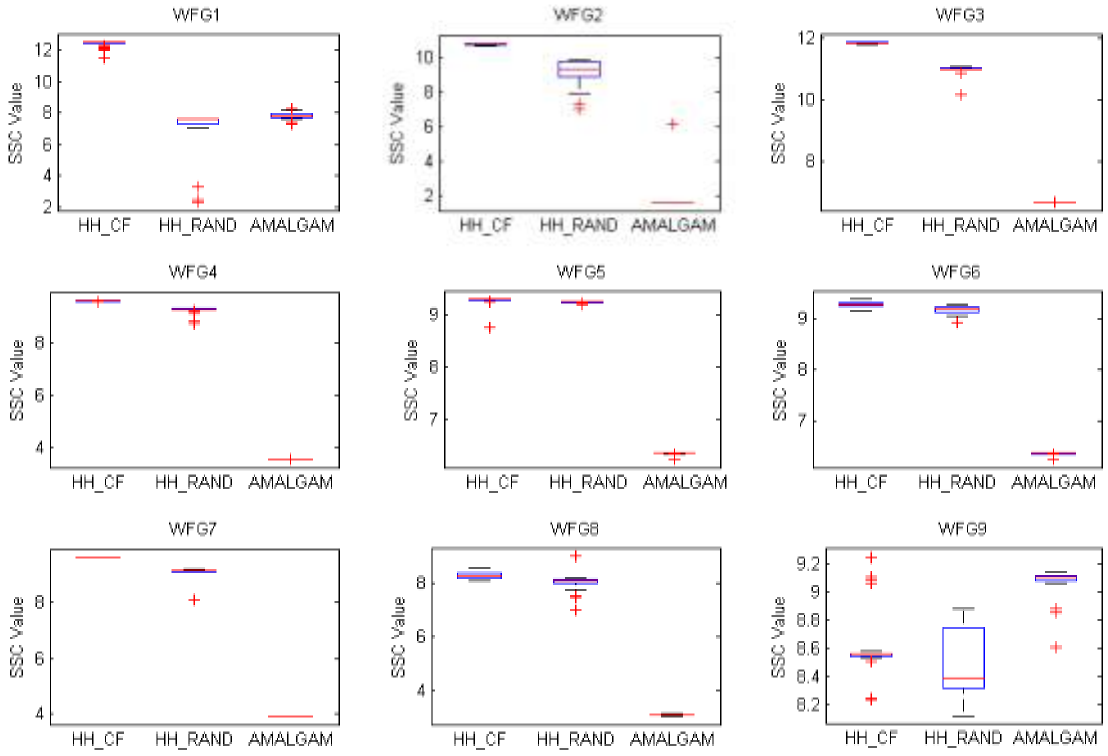


Figure 5.8: Box plots of HH_CF, HH_RAND, and AMALGAM for the measure of hypervolume (SSC) on the WFG test functions.

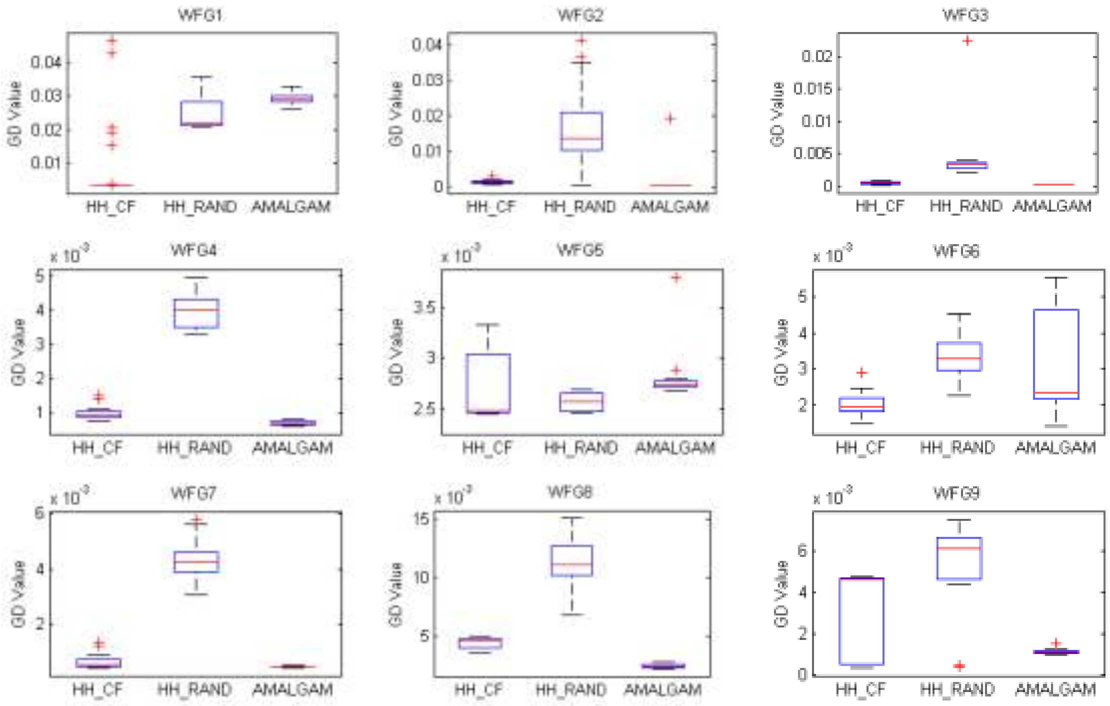


Figure 5.9: Box plots of HH_CF, HH_RAND, and AMALGAM for the measure of generational distance (GD) on the WFG test functions.

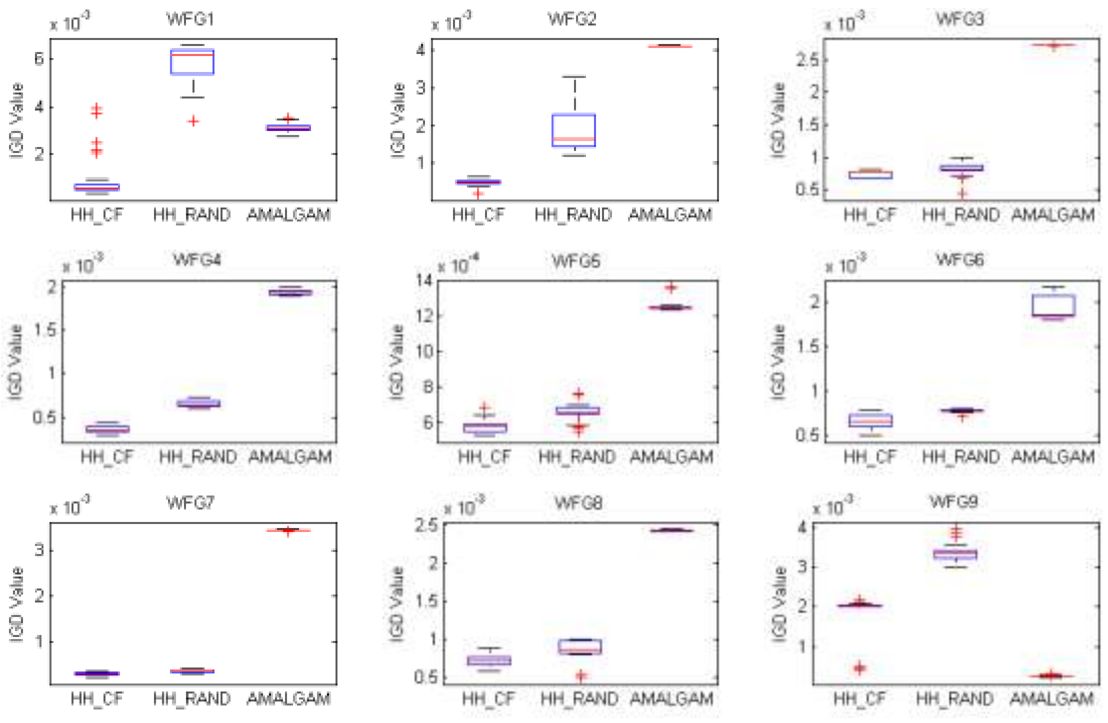


Figure 5.10: Box plots of HH_CF, HH_RAND, and AMALGAM for the measure of inverted generational distance (IGD) on the WFG test functions.

Problem	Methods	Metrics		
		SSC	GD	IGD
WFG1	HH_CF:HH_RAND	+	+	+
	HH_CF:AMALGAM	+	+	+
	HH_RAND:AMALGAM	+	+	+
WFG2	HH_CF:HH_RAND	+	+	+
	HH_CF:AMALGAM	+	+	+
	HH_RAND: AMALGAM	+	-	+
WFG3	HH_CF:HH_RAND	+	+	~
	HH_CF:AMALGAM	+	-	+
	HH_RAND: AMALGAM	+	-	+
WFG4	HH_CF:HH_RAND	+	+	+
	HH_CF:AMALGAM	+	-	+
	HH_RAND:AMALGAM	+	-	+
WFG5	HH_CF:HH_RAND	+	+	~
	HH_CF:AMALGAM	+	+	+
	HH_RAND:AMALGAM	+	~	+
WFG6	HH_CF:HH_RAND	+	+	~
	HH_CF:AMALGAM	+	+	+
	HH_RAND:AMALGAM	+	+	+
WFG7	HH_CF:HH_RAND	+	+	~
	HH_CF:AMALGAM	+	+	+
	HH_RAND:AMALGAM	+	+	+
WFG8	HH_CF:HH_RAND	+	+	~
	HH_CF:AMALGAM	+	-	+
	HH_RAND:AMALGAM	+	-	+
WFG9	HH_CF:HH_RAND	+	+	+
	HH_CF:AMALGAM	-	-	-
	HH_RAND:AMALGAM	-	-	-

Table 5.5: The *t*-test results of HH_CF,HH_RAND and AMALGAM on the WFG test problems with respect to the hypervolume (SSC), the generational distance(GD) and the inverted generational distance (IGD).

It can be concluded that all the above results demonstrate the effectiveness of HH_CF in terms of its ability to intelligently adapt to calling combinations of low level heuristics and outperforming other hyper-heuristics for multi-objective optimisation (HH_RAND and AMALGAM) for solving these kind of problems.

5.4 Summary and Remarks

This chapter presented an online selection choice function based hyper-heuristic for multi-objective optimisation (HHMO_CF) (or HH_CF for short) employing All-Moves as an acceptance strategy. This is meaning that we accept the output of each low level heuristic whether it improves the quality of the solution or not. Four performance metrics (Algorithm effort (AE), Ratio of non-dominated individuals (RNI), Size of space covered (SSC) and Uniform distribution of a non-dominated population (UD)) act as an online learning mechanism to provide knowledge of the problem domain to the high level strategy.

We have conducted a number of experiments to analyse HH_CF and compared its performance to the low level heuristics (NSGAII, SPEA2 and MOGA), when used in isolation over the nine WFG test functions which we utilise as our benchmark instances. We have also conducted a number of experiments to examine the performance of our proposed HH_CF, comparing with two multi-objective hyper-heuristics; a random hyper-heuristics (HH_RAND) and the adaptive multi-method search AMALGAM over the same benchmark instances.

The experimental results shows that the choice function all-moves based hyper-heuristic can benefit from the strengths of the low level heuristics. Moreover, it has the capability to intelligently adapt to calling combinations of low level heuristics. Our hyper-heuristic performs well in terms of the distribution of non-dominated individuals along the POF and obtains competitive results in terms of converging towards the POF. However, it performs poorly with respect to the number of non-dominated solutions in the population. Another acceptance strategy instead of All-Moves can be employed to avoid this and improve the quality of solutions. This is investigated in Chapters 6 and 7.

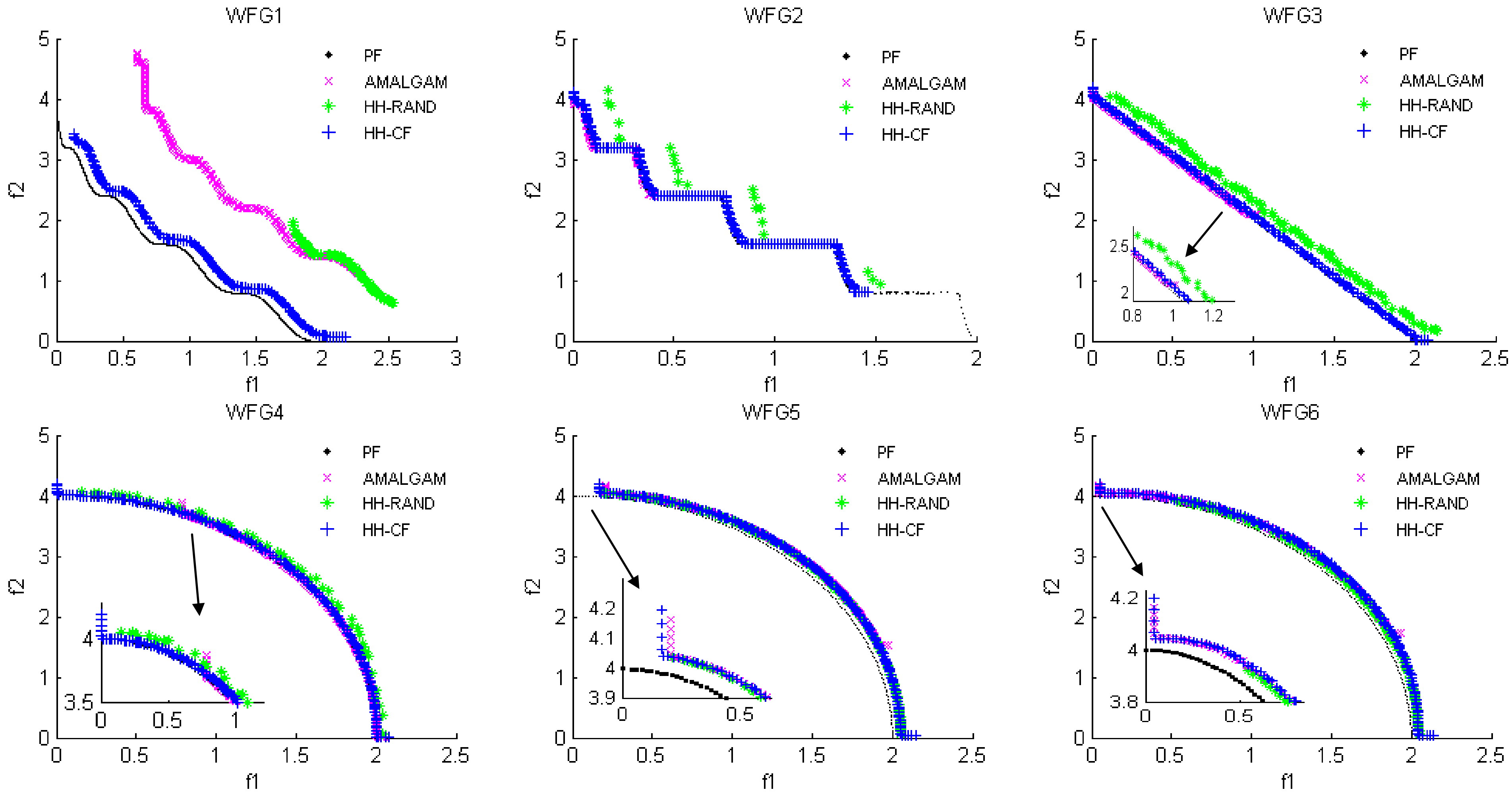


Figure 5.11: Pareto optimal front and 50% attainment surfaces for AMALGAM, HH_RAND and HH_CF after 25,000 evaluation functions on the WFG1-WFG6 test functions.

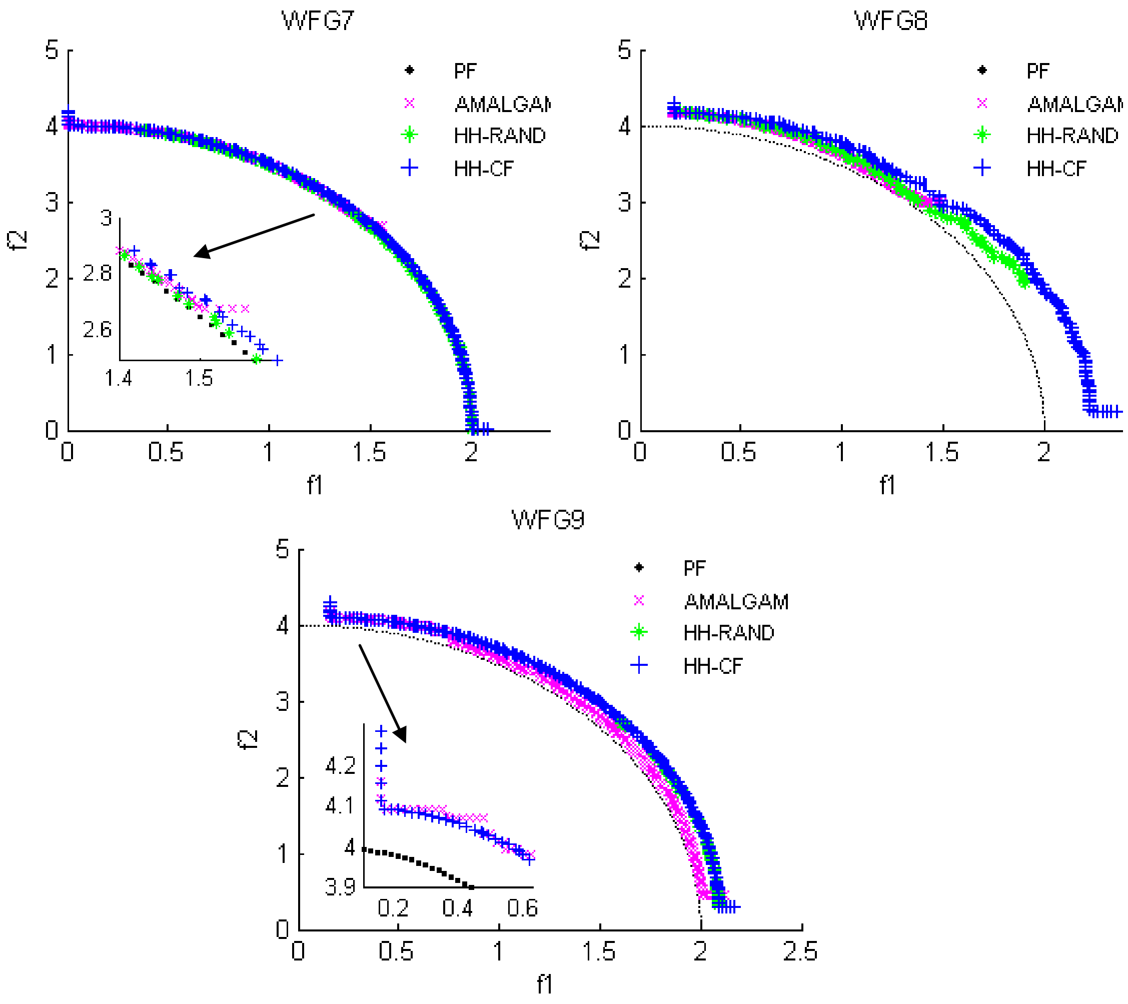


Figure 5.12: Pareto optimal front and 50% attainment surfaces for AMALGAM, HH_RAND and HH_CF after 25,000 evaluation functions on the WFG7-WFG9 test functions.

6 A Heuristic Selection Using Great Deluge as a Non-Deterministic Move Acceptance Strategy

In the previous chapter, we presented a choice function heuristic selection combined with All-Moves as an acceptance strategy for multi-objective optimisation. Our multi-objective choice function based hyper-heuristic used the WFG test suit as our benchmark instances. It showed good performance and produces good quality solutions in terms of the diversity and convergence towards the POF. As All-Moves accepts all solutions of each low level heuristic, whether it improves the quality of the solution or not, the choice function all-moves based hyper-heuristic fails to avoid the MOGA weakness by accepting solutions with poor quality in terms of the number of non-dominated solutions. To overcome this, we propose to use another move acceptance strategy instead of All-Moves that accepts worsening moves within a limited degree and help improve the quality of the solutions. This chapter investigates the performance of the choice function based hyper-heuristic when combining great deluge (GDA) (Dueck, 1993) as an acceptance criteria. We also investigate the sensitivity of our choice function based hyper-heuristic using different parameter settings for the great deluge algorithm.

6.1 The Great Deluge Algorithm as a Move Acceptance Criteria

In the scientific literature, there are many studies that investigate GDA and its variants in tackling various optimisation problems. However, the majority of them are applied to optimisation problems with a single-objective. Petrovic et al. (2007) proposed a case based reasoning methodology with GDA for solving examination timetabling problems. In Bykov (2003) GDA is applied to thirteen benchmark problems for examination timetabling. The experimental result shows that GDA yields the best result for the majority of the problems when compared to a time predefined simulated annealing approach. A new hybridised method based on a genetic algorithm and GDA is proposed in Al-Milli (2010). The approach tackles course timetabling problems, producing good quality solutions for standard benchmark problems. In Scott and Geldenhuysys (2000) the performance of a GDA was compared to tabu search (TS) for graph colouring. The results show that GDA was able to obtain better colourings, particularly for large graphs in shorter times. GDA was applied to the travelling salesman problem (TSP) in Telfar (1995). In Dhouib (2000), a multi start great deluge approach was proposed to optimise two continuous engineering design problems. The simulation results show that this approach performs better than SA and a genetic algorithm. McMullan and McCollum (2007) proposed an extended version of GDA using a reheating (relevelling) technique. This GDA variant was applied to a dynamic job scheduling problem, producing better results in most cases when compared to SA. Another extended version of GDA was proposed by Baykasoglu et al. (2011). This method was applied to two problems; industrial process control and a simulation model of a job shop, yielding promising results. Nourelfath et al. (2007) presented a hybrid approach combining GDA and ant colony optimisation. This approach was applied to the discrete facility layout problem (FLP) and tested on quadratic assignment problem (QAP) benchmarks. The experimental results indicate that the hybrid algorithm outperforms many other meta-heuristics. Nahas et al. (2010) proposed another version of the GDA called the Iterated Great Deluge (IDA) to solve the dynamic facility layout problem. The method produces competitive results. An extension to the GDA was proposed by Burke and Bykov (2006). This approach, called *Flex-Deluge,* introduces a flexibility coefficient that controls the move acceptance and is. This GDA variant performed well for solving exam timetabling problems. Another variant of GDA combined with evolutionary operators was proposed by Landa-Silva and Obit (2009). GDA utilises a non-linear rate of change for the threshold. This hybrid evolutionary approach, applied to a university course timetabling problem, performed better for solving four out of eleven instances. Pramodh and Ravi (2007) presented four variants of GDA on three different benchmarks from banks for predicting bankruptcy.

The GDA is not only employed as a meta-heuristic to solve optimisation problems. It is also used in many hyper-heuristic approaches as an acceptance move strategy. Özcan et al. (2010) shows a reinforcement learning great deluge hyper-heuristics and reinforcement learning late acceptance are promising when applied to examination timetabling, and produced good quality solutions when compared to some other approaches in the literature. Kendall and Mohamad (2004) presented a variant of a GDA based hyper-heuristics. It was applied to channel assignment benchmarks. The experimental results show simple random-great deluge produced good results when compared to a constructive heuristic and a genetic algorithm. In addition, a variant of the GDA hyper-heuristic approach including flex deluge (FD), non-linear (NLGD) and extended great deluge (EGD) is proposed in Sin and Kham (2012). These approaches were applied to large scale and highly constrained timetabling problems and tested on exam timetabling benchmark problems. The experimental results demonstrate that NLGD produced the best results compared to other approaches in the literature. In Gibbs et al. (2011) the performance of different hyper-heuristics are compared with different components emphasising the influence of learning heuristic selection methods for solving a sports scheduling problem. It have been shown that the proposed approach is slightly better than the other approaches that use choice function as heuristic selection and great deluge algorithm as an acceptance criteria for solving a sports scheduling problem.

An important observation is that all the above GDA studies deal with single-objective optimisation problems. However, there is only one study that has proposed the GDA for multi-objective optimisation (Petrovic and Bykov, 2003). This method is based on a trajectory that guides the search dynamics by changing the criteria weights of the cost function values. This method was applied to a set of real-world timetabling problems, producing high quality solutions. We decide to employ GDA as a move acceptance component in our multi-objective hyper-heuristics choice function as GDA is simple and depends on fewer parameters (Petrovic et al, 2007). Moreover, it was successful with single-objective optimisation (Kendall and Mohamad, 2004). And no work has been reported in the literature that utilises the GDA as a move acceptance component within a hyper-heuristic framework for multi-objective optimisation. Details about the great deluge algorithm are formally discussed in Section 2.2.6. GDA, as move acceptance strategy, requires computation of the change in the value of a single objective at each step and so the *D* performance metric (Zitzler, 1999) is proposed for its applicability to multi-objective optimisation problems.

6.2 The Great Deluge and *D* Metric

In the context of move acceptance criterion, the quality measure of the current solution and the candidate solution is essential in order to make a decision regarding an acceptance decision. For the single-objective optimisation problem, fitness can be used. However, this is not applicable in multi-objective optimisation. In multi-objective problems, the output is a set of solutions (a non-dominated set). We propose the use of the *D* metric (Zitzler, 1999) as a way of comparing two non-dominated sets with respect to the objective space. In this thesis, we use *D* metric, integrating into move acceptance criterion, particularly GDA, in order to convert multi-objective optimisation to single-objective optimisation without definition of criteria weights. This is similar to the concept that is used in indicator-based multi-objective optimisers (e.g. (Auger et al.,2012; Wang et al., 2013; Bader and Zitzler, 2011)), where a multi-objective problem is converted to a single-objective problem by optimising the quality indicator instead of optimising a set of objective functions simultaneously. In an indicator-based evolutionary algorithm, such as ESP (Huband et al., 2003), SMS-EMOA (Beume et al., 2007), the hypervoulme is integrated into environmental selection. In our multi-objective choice function hyper-heuristic, the *D* metric is integrated into the move acceptance strategy. Our goal is to maximise the underlying *D* metric as follows.

$$\begin{aligned}
&LEVEL = D(A,B)\\
&\boldsymbol{if}\ \ D(B,A) > LEVEL\\
&\boldsymbol{then}\ \ A = B\\
&\qquad LEVEL = LEVEL + UP
\end{aligned} \tag{6.1}$$

A is a non-dominated front which represents an initial solution and *B* is is a non-dominated front which represents a candidate solution from the neighbourhood. The water level is assigned initially to $D(A,B)$. Note that we are always looking to get a higher value (maximise) of $D(B,A)$ in order to accept the candidate solution *B*, so the condition $D(B,A) > D(A,B)$ or $D(B,A) > LEVEL$ should be valid (see subsection 2.1.9). In the acceptance case, *B* is accepted and the water level is increased linearly according to a predefined speed rate (*UP*) which is usually a small fraction greater than 0 less than 0.03 (Scott and Geldenhuysys, 2000).

6.3 Choice Function Great Deluge for Selecting Low Level Meta- heuristics (HHMO_CF_GDA)

In this section, we propose a multi-objective choice function based hyper-heuristic combining it with great deluge as a non-deterministic acceptance strategy (HHMO_CF_GDA). We use the same multi-objective hyper-heuristic framework that we proposed in Chapter 4 including the ranking scheme and learning mechanism. Three well-known (as previously) multi-objective evolutionary algorithms (NSGAII, SPEA2, and MOGA), act as the low level heuristics. The pseudo code of the proposed HHMO_CF_GDA for multi-objective optimisation is shown in algorithm 11.

Algorithm 11: Multi-objective Choice Function Great Deluge based Hyper-heuristic

```
 1: procedure HHMO_CF_GDA $(H)$ $whereas\ H$ is a set of the low level heuristics
 2:  Initialisation
 3:  Run $h, \forall h \in H$
 4:  Rank $h, \forall h \in H$ based on the ranking scheme
 5:  Get $CF(h), \forall h \in H$
 6:  Select $h$ with the largest $CF(h)$ as an initial heuristic
 7:  Execute the selected $h$ and produce a front $A$
 8: repeat
 9:    Update the rank of $h, \forall h \in H$ based on the ranking scheme
10:    Update $CF(h), \forall h \in H$
11:    Select $h$ with the largest $CF(h), \forall h \in H$
12:    Execute the selected $h$ and produce a front $B$
13:    $LEVEL = D(A,B)$
14:    If $D(B,A) > LEVEL$
15:       $A = B$
16:       $LEVEL = LEVEL + UP$
17: until (termination criteria are satisfied)
18: end procedure
```

Initially, a greedy algorithm is applied to determine the best low level heuristic *h* to be selected for the first iteration (steps 2-6). All low level heuristics *H* are executed (step 3). Then, the low level heuristics are ranked based on the ranking scheme using Equation 4.1 (step 4) and their choice function values are computed using Equation 4.4 (step 5). The low level heuristic *h* with the largest choice function value *CF*(*h*) is selected to be applied at the next iteration and it produces the non-dominated front *A* (a current solution) (steps 6 & 7). Then, for all low level heuristics *H*, the ranking mechanism is updated (step 9). The choice function values are also computed and updated (step 10). According to the updated choice function values, the low level heuristic *h* with the largest choice function value *CF*(*h*) is executed and it produces the non-dominated front *B* (a candidate solution) (steps 11 & 12). In steps 13-15, the acceptance procedure GDA is applied. As we are aiming to maximise *D(B,A),* the condition in (step 14) should be valid in order to accept the candidate front *B* (step 15). In the case of acceptance, the water level is increased linearly based on a predefined rain speed rate (*UP*). This process is repeated until the stopping condition is met which is a fixed number of iterations (steps 8-17). Note that the greedy algorithm is applied only once at the beginning of the search, in order to determine which low level heuristic to apply first. Then, only one low level heuristic is selected at each iteration.

6.4 Performance Comparison of Choice Function Great Deluge Hyper-heuristics

As a preliminary framework, we combine great deluge as a move acceptance with simple random as a heuristic selection method. A low level heuristic was selected randomly at each iteration in the search process. According to the results that reported in Chapter 5, we believe that a simple random selection strategy is not that successful as it does not retain any knowledge about the performance of low level heuristics on which to base future decisions. To examine our assumption, we conduct an initial experiment to compare the performance of great deluge when combined with simple random and a choice function as a selection method under the multi-objective hyper-heuristic framework. For the choice function great deluge based hyper-heuristic, we use the same multi-objective hyper-heuristic framework that presented in Chapter 4, including the ranking scheme and learning mechanism, and the same experimental settings that were used in Section 5.2.2. The rain speed parameter (*UP*) is initially assigned to 0.03 as recommended in the literature (Scott and Geldenhuysys, 2000). As we expected the comparison revealed that choice function great deluge based hyper-heuristic outperforms the simple random great deluge based hyper-heuristic on the WFG1 benchmark with respect to the three performance metrics; RNI, SSC and UD (see Table 6.1). The choice function great deluge based hyper-heuristic also performs well when compared to the pervious hyper-heuristic method, choice function all-moves, that presented in Chapter 5.

Metric	Methods	AVG	MIN	MAX	STD
RNI	CF-GDA	**0.9480**	0.1600	1.0000	**0.1894**
	CF-AM	0.8800	0.2800	1.0000	0.2539
	SR-GDA	0.6423	0.0300	1.0000	0.4124
SSC	CF-GDA	**12.2380**	8.3703	12.5154	**0.7870**
	CF-AM	12.1386	9.0338	12.5130	0.9101
	SR-GDA	8.2421	5.3700	8.4240	2.5423
UD	CF-GDA	0.4066	0.2083	0.8000	**0.0988**
	CF-AM	**0.4428**	0.3490	0.6945	0.1007
	SR-GDA	0.2937	0.2501	0.3900	0.2834

Table 6.1: The performance of the choice function great deluge based hyper-heuristic (CF-GDA), choice function all-moves hyper-heuristic (CF-AM) and the simple random great deluge based hyper-heuristic (SR-GDA) with respect to the metrics of ratio of non-dominated individuals (RNI), size of space covered (SSC), and uniform distribution (UD) of non-dominated population onWFG1.

We note from the Table 6.1 that both hyper-heuristics that have utilised a choice function as a heuristic selection method outperforms the hyper-heuristic that used a random selection method. Unlike the random selection strategy, the choice function considers the performances of low level heuristics in order to select a suitable heuristic as the search progresses. The learning mechanism is essential in our multi-objective hyper-heuristic framework. It plays a large role in guiding the high level strategy (selection method) and deciding which low level heuristic to call at each decision point.

6.4.1 Tuning of Rain Speed Parameter (*UP*)

One of the reasons for choosing the great deluge algorithm (GDA) as a move acceptance component within our multi-objective hyper-heuristic framework is due to its simplicity and dependency on fewer parameters (Petrovic et al, 2007). In fact, GDA has one parameter which is the rain speed (*UP*). In the literature, it recommended to set the *UP* to 0.03 or less (Scott and Geldenhuysys, 2000). However, the choice of the rain speed value is not trivial, bearing in mind that the suggestion for *UP*=0.03 was for single-objective problems. Coming up with the right value requires domain knowledge, such as, the target upper limit (for our case) and a specific number of moves that we conduct during the search until we reach that target level. The rain speed (*UP*) was fixed at 0.03 during the initial experiments.

From our observation during the experiments, many questions have arisen regarding *UP*. What is the *best* value for this parameter? Does it depend on the given problem? How does changing *UP* influence the quality of solutions? How about if we narrow/widen the level boundary? Will the solution be improved? To answer these questions, we conducted a number of experiments to investigate the effectiveness of the speed rain parameter on the quality of solutions. We assigned different rain speed parameter values comparing to our default parameter of 0.03. These settings include 0.3 as a large value and 0.0003 as a small value. Please note the 0.3 value does not come with any recommendation from the literature. However, we set *UP*=0.3 in order to examine the effectiveness of rain speed parameter and how this could affect the acceptance process and the quality of solutions. The water level is increased linearly according to a predefined rain speed rate.

6.4.2 Experimental Settings and Performance Evaluation Criteria

We use the same experimental settings that we presented in Section 5.2.2. Nine test problems for the WFG suite (WFG1-WFG9) have 24 real parameters including four position parameter, 20 distance parameters and two objectives. HHMO_CF_GDA was terminated after 6,250 generations. That is, HHMO_CF_GDA runs for a total of 25 iterations. In each iteration, one low level heuristic is applied and is executed for 250 generations, with a population size equal to 100. The secondary population of SPEA2 is set to 100. For the WFG problems, 30 independent trials were run for each algorithm with a different random seed. For GDA, the rain speed (*UP*) is assigned to three values (0.3, 0.03 and 0.0003). HH_CF_GDA was implemented with the same common sub-functions using Microsoft Visual C++ 2008 on an Intel Core2 Duo 3GHz\2G\250G computer.

Three performance metrics are used to assess the quality of approximation sets in different aspects including ratio of non-dominated individuals (RNI), the hyper-volume (SSC), and Uniform distribution of non-dominated individuals (UD). For all performance metrics, a higher value indicates a better performance. In addition, *t*-test is used as a statistical test for pairwise mean performance comparison of three version on

HHMO_CF_GDA using different UP values (0.3, 0.03 and 0.0003). The null hypothesis is as follows:

$$\begin{cases} H_0 \; the\ performance\ of\ a\ pair\ of\ algorithms\ \ have\ the\ same\ means \\ H_1 \; the\ performance\ of\ a\ pair\ of\ algorithms\ \ have\ different\ means \end{cases}$$

The following notation is used while reporting the results. Given a pair of algorithms, P and Q (denoted as $P{:}Q$), The + ($-$) indicates that the algorithm P performs better/worse than Q on average with respect to the given metric and this performance difference is statistically significant within a confidence interval of 95%. The $\pm$ ($\mp$) indicates that P performs slightly better (worse) than Q without any statistically significance. The n/a means the *t*-test is not applicable since the performances of both algorithms are completely equal.

6.4.3 Experimental Results and Discussion

The average, minimum, maximum and standard deviation values pairs for HHMO_CF_GDA using different rain speed (*UP*) values with respect to RNI, SSC and UD over 30 trials are provided in Table 6.2. The pairwise mean performance comparisons (using *t*-test) of HHMO_CF_GDA using different *UP* settings are provided in Table 6.3. We refer to the HHMO_CF_GDA using the *UP* values (0.3, 0.03 and 0.0003) as GDA1, GDA2 and GDA3 respectively.

HHMO_CF_GDA with the smallest *UP* value (GDA3) performs the best. We note from the Tables 6.2 and 6.3 that the pairwise performance differences of GDAs are statistically significant for all benchmark functions, except for the metric RNI where GDA1, GDA2 and GDA3 perform the same. GDA1 and GDA2 perform significantly similar on average with respect to the measure of SSC and UD. With respect to the measure of SSC, GDA3 is statistically significant better than GDA1 and GDA2 for all benchmark instances. GDA3 performs statistically better than the others in terms of distribution along the POF (UD) in all test instances except WFG1 and WFG2.

The results are illustrated in Figures 6.1, 6.2 and 6.3. We can see from Figure 6.1, the water level when the rain speed set to *UP*=0.03 has been increased more quickly compared when the rain speed set to *UP*=0.0003. The rapid growth of the water level freezes the boundary condition in the early stages of the search as is the case when *UP*=0.3 (see Figure 6.4). This leads

WFG	Metric	UP	Method	AVG	MIN	MAX	STD
1	RNI	0.3	GDA1	0.9390	0.1100	1.0000	**0.1441**
		0.003	GDA2	**0.9480**	0.1600	1.0000	0.1894
		0.0003	GDA3	0.9357	0.3100	1.0000	0.1821
	SSC	0.3	GDA1	11.6170	7.9112	12.0140	**0.6905**
		0.003	GDA2	12.2380	8.3703	12.5154	0.7870
		0.0003	GDA3	**12.9388**	8.2543	12.9966	1.2517
	UD	0.3	GDA1	0.3561	0.2022	0.5841	0.0753
		0.003	GDA2	**0.4066**	0.2083	0.8000	0.0988
		0.0003	GDA3	0.3941	0.2047	0.5952	**0.0698**
2	RNI	0.3	GDA1	**1.0000**	1.0000	1.0000	**0.0000**
		0.003	GDA2	**1.0000**	1.0000	1.0000	**0.0000**
		0.0003	GDA3	**1.0000**	1.0000	1.0000	**0.0000**
	SSC	0.3	GDA1	10.8023	10.5912	11.3981	**0.0045**
		0.003	GDA2	**10.8310**	10.6391	12.4274	0.3034
		0.0003	GDA3	11.8148	10.7433	11.8258	0.0146
	UD	0.3	GDA1	0.3710	0.3497	0.3805	**0.0057**
		0.003	GDA2	**0.3756**	0.3550	0.4187	0.0144
		0.0003	GDA3	0.3729	0.3609	0.3862	0.0064
3	RNI	0.3	GDA1	**1.0000**	1.0000	1.0000	**0.0000**
		0.003	GDA2	**1.0000**	1.0000	1.0000	**0.0000**
		0.0003	GDA3	**1.0000**	1.0000	1.0000	**0.0000**
	SSC	0.3	GDA1	11.7543	11.8356	11.9196	**0.0054**
		0.003	GDA2	11.8930	11.8620	11.9201	0.0151
		0.0003	GDA3	**11.9197**	11.9094	11.9296	0.0064
	UD	0.3	GDA1	0.4190	0.3788	0.4578	0.0133
		0.003	GDA2	0.4224	0.3874	0.4575	0.0129
		0.0003	GDA3	**0.4252**	0.4059	0.4580	**0.0120**
4	RNI	0.3	GDA1	**1.0000**	1.0000	1.0000	**0.0000**
		0.003	GDA2	**1.0000**	1.0000	1.0000	**0.0000**
		0.0003	GDA3	**1.0000**	1.0000	1.0000	**0.0000**
	SSC	0.3	GDA1	9.5921	9.5234	9.6100	**0.0032**
		0.003	GDA2	9.6181	9.5821	9.6376	0.0146
		0.0003	GDA3	**9.6642**	9.6210	9.6650	0.0100
	UD	0.3	GDA1	0.4101	0.3510	0.4163	0.0122
		0.003	GDA2	0.4115	0.3710	0.4415	0.0157
		0.0003	GDA3	**0.4145**	0.3879	0.4423	**0.0112**
5	RNI	0.3	GDA1	**1.0000**	1.0000	1.0000	**0.0000**
		0.003	GDA2	**1.0000**	1.0000	1.0000	**0.0000**
		0.0003	GDA3	**1.0000**	1.0000	1.0000	**0.0000**
	SSC	0.3	GDA1	9.2682	9.0977	9.2866	0.0094
		0.003	GDA2	9.2771	9.2607	9.2928	**0.0084**
		0.0003	GDA3	**9.2964**	9.1526	9.2984	0.4023
	UD	0.3	GDA1	0.4083	0.3683	0.4399	**0.0041**
		0.003	GDA2	0.4110	0.3772	0.4481	0.0235
		0.0003	GDA3	**0.4395**	0.4238	0.4579	0.0086
6	RNI	0.3	GDA1	**1.0000**	1.0000	1.0000	**0.0000**
		0.003	GDA2	**1.0000**	1.0000	1.0000	**0.0000**
		0.0003	GDA3	**1.0000**	1.0000	1.0000	**0.0000**
	SSC	0.3	GDA1	9.3394	9.2008	9.4683	**0.0543**
		0.003	GDA2	9.3421	9.2102	9.4715	0.0581
		0.0003	GDA3	**9.3745**	9.2346	9.4787	0.0628
	UD	0.3	GDA1	0.4108	0.3711	0.4255	**0.0045**
		0.003	GDA2	0.4115	0.3749	0.4287	0.0129
		0.0003	GDA3	**0.4128**	0.3992	0.4308	0.0083
7	RNI	0.3	GDA1	**1.0000**	1.0000	1.0000	**0.0000**
		0.003	GDA2	**1.0000**	1.0000	1.0000	**0.0000**
		0.0003	GDA3	**1.0000**	1.0000	1.0000	**0.0000**
	SSC	0.3	GDA1	9.6391	9.5754	9.6522	0.0154
		0.003	GDA2	9.6402	9.5869	9.6571	0.0187
		0.0003	GDA3	**9.6650**	9.6596	9.6700	**0.0028**
	UD	0.3	GDA1	0.4011	0.3630	0.4321	**0.0144**
		0.003	GDA2	0.4038	0.3660	0.4345	0.0162
		0.0003	GDA3	**0.4085**	0.3792	0.4565	0.0151

WFG	Metric	UP	Method	AVG	MIN	MAX	STD
8	RNI	0.3	GDA1	**1.0000**	1.0000	1.0000	**0.0000**
		0.003	GDA2	**1.0000**	1.0000	1.0000	**0.0000**
		0.0003	GDA3	**1.0000**	1.0000	1.0000	**0.0000**
	SSC	0.3	GDA1	8.5643	8.4132	8.6588	0.0132
		0.003	GDA2	8.5783	8.4534	8.6667	0.0150
		0.0003	GDA3	**8.7279**	8.6708	8.7389	**0.0120**
	UD	0.3	GDA1	0.4210	0.3920	0.4599	**0.0050**
		0.003	GDA2	0.4228	0.4040	0.4610	0.0150
		0.0003	GDA3	**0.4248**	0.3948	0.5933	0.0341
9	RNI	0.3	GDA1	0.9801	0.7500	1.0000	**0.0073**
		0.003	GDA2	0.9866	0.7600	1.0000	0.0518
		0.0003	GDA3	**0.9893**	0.8000	1.0000	0..4193
	SSC	0.3	GDA1	8.7299	8.5498	9.4277	**0.2487**
		0.003	GDA2	8.7313	8.5554	9.4465	0.2693
		0.0003	GDA3	**8.7689**	8.5789	9.4346	0.3054
	UD	0.3	GDA1	0.4021	0.3611	0.4559	**0.0099**
		0.003	GDA2	0.4088	0.3657	0.4606	0.0210
		0.0003	GDA3	**0.4111**	0.3661	0.6141	0.4442

Table 6.2: The average performance of HHMO_CF_GDA using a different UP settings (0.3, 0.03, 0.0003) donated as GDA1, GDA2 and GDA3 on the WFG test problems with respect to the ratio of non-dominated individuals (RNI), the hypervolume (SSC) and the uniform distribution (UD).

Problem	Methods	Metrics		
		RNI	SSC	UD
WFG1	GDA1:GDA2	+	−	−
	GDA1:GDA3	±	−	−
	GDA2:GDA3	−	−	+
WFG2	GDA1:GDA2	n/a	−	−
	GDA1:GDA3	n/a	−	∓
	GDA2:GDA3	n.a	−	±
WFG3	GDA1:GDA2	n/a	−	−
	GDA1:GDA3	n/a	−	−
	GDA2:GDA3	n.a	−	−
WFG4	GDA1:GDA2	n/a	−	∓
	GDA1:GDA3	n/a	−	−
	GDA2:GDA3	n.a	−	−
WFG5	GDA1:GDA2	n/a	−	∓
	GDA1:GDA3	n/a	−	−
	GDA2:GDA3	n.a	−	−
WFG6	GDA1:GDA2	n/a	∓	∓
	GDA1:GDA3	n/a	−	-
	GDA2:GDA3	n.a	−	∓
WFG7	GDA1:GDA2	n/a	∓	∓
	GDA1:GDA3	n/a	−	∓
	GDA2:GDA3	n.a	−	∓
WFG8	GDA1:GDA2	n/a	∓	∓
	GDA1:GDA3	n/a	−	∓
	GDA2:GDA3	n.a	−	∓
WFG9	GDA1:GDA2	∓	∓	∓
	GDA1:GDA3	∓	−	−
	GDA2:GDA3	∓	−	−

Table 6.3: The *t*-test results of HHMO_CF_GDA using a different *UP* settings (0.3, 0.03, 0.0003) donated as GDA1, GDA2 and GDA3 on the WFG test problems with respect to the ratio of non-dominated individuals (RNI), the hypervolume (SSC) and the uniform distribution (UD).

to accept the good moves in few number of decision points in the beginning of the search, while all other moves are rejected for the rest of the search. However, the slow growth of the water level provides a wider space in the search to accept more moves as the case when *UP*=0.0003. This helps to improve solutions by escaping from the local optimum. From Figure 6.1, we note that for both settings of *UP* (0.03 and 0.0003) in WFG1, there is no effect on the acceptance criteria, i.e. for all decision points, all moves are accepted since the boundary limit is under the candidate solutions level.

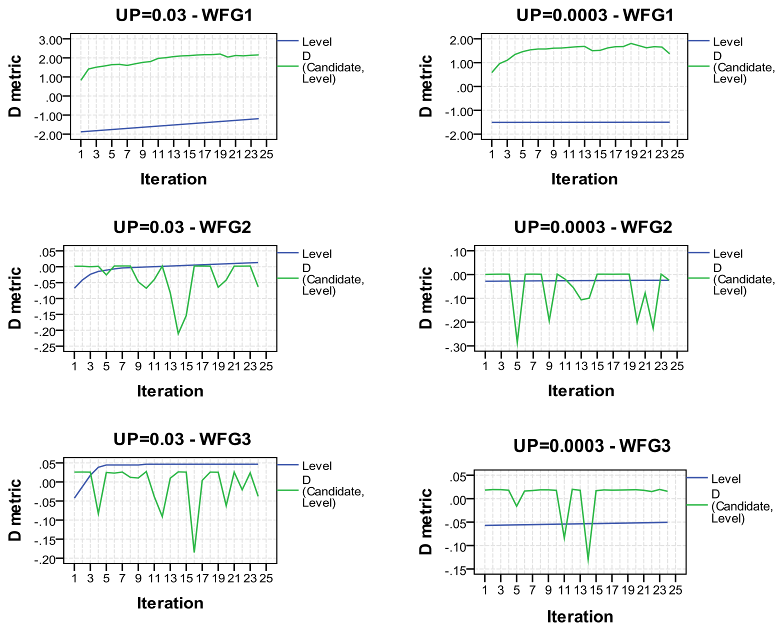


Figure 6.1: The performance of *D* metric (Green line) and Level (Blue line) during the search across 25 decision points for HHMO CF GDA with different sizes of *UP* (0.03 and 0.0003) on the WFG test suite – Continue.

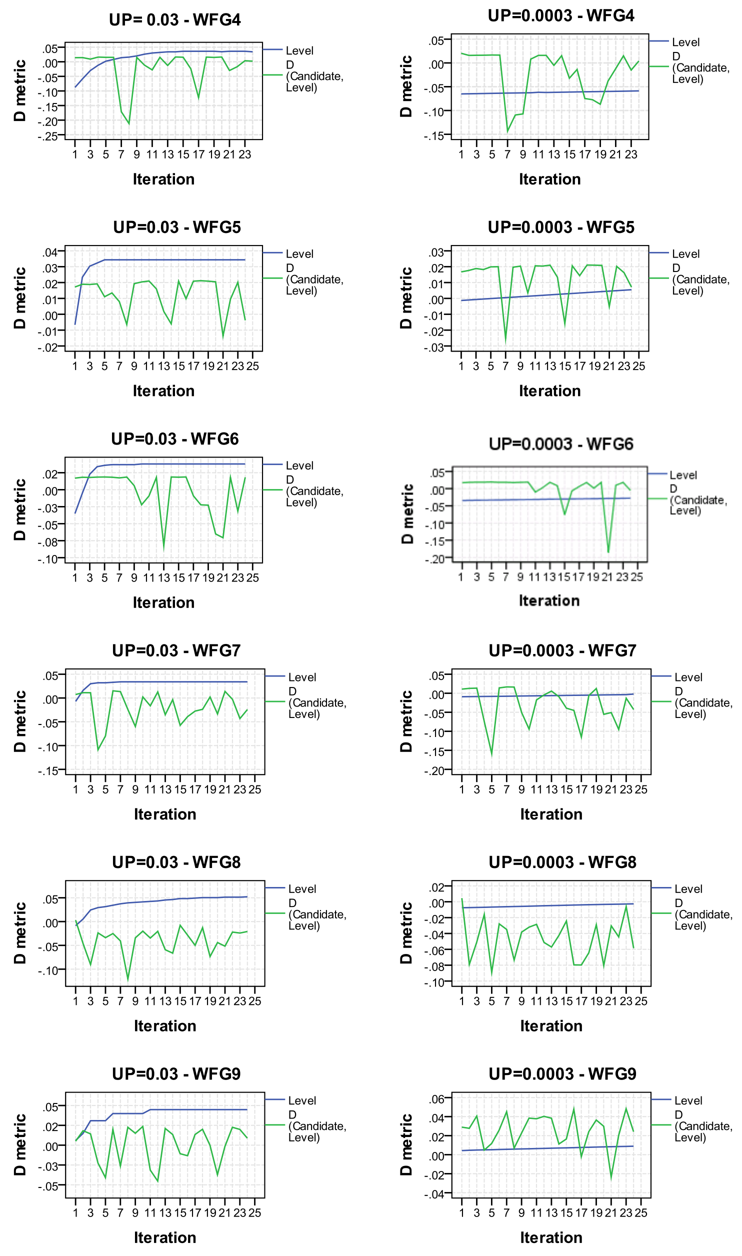


Figure 6.1: Continue- the performance of *D* metric (Green line) and Level (Blue line) during the search across 25 decision points for HHMO CF GDA with different sizes of *UP* (0.03 and 0.0003) on the WFG test suite.

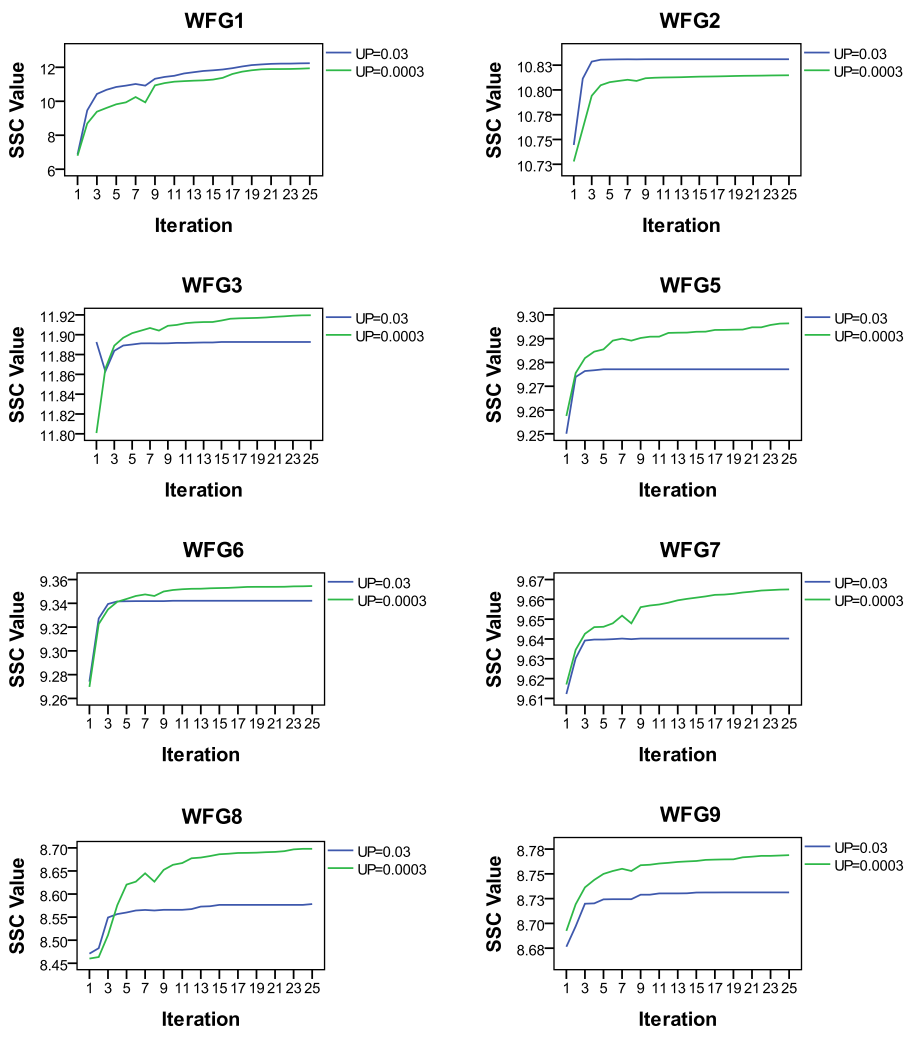


Figure 6.2: The performance HHMO_CF_GDA with different *UP* sizes (0.03 and 0.0003) during the search across 25 decision points with respect to the size of space covered metric (SSC) during on the WFG test suite.

From Figures 6.2 and 6.3, HHMO_CF_GDA always performs better during the search with respect SSC and UD metrics when the *UP* is small for all WFG problems except WFG1 and WFG2. In general, the smaller rain speed value allows for the acceptance of more moves with worse solution quality. This helps escape from the local optimum and produce better solution. This is clear in Figure 6.4. The HHMO_CF_GDA with the large *UP* value (0.3) has the worst performance in WFG4. There is no change in the values of the SSC and UD metrics which means no moves were accepted during the search. Moves acceptance has been frozen in the 6th iterations because the level rose too quickly. While in the 0.0003 case, the level rose slightly which gives the GDA more boundary space to accept more moves. So, HHMO_CF_GDA with the

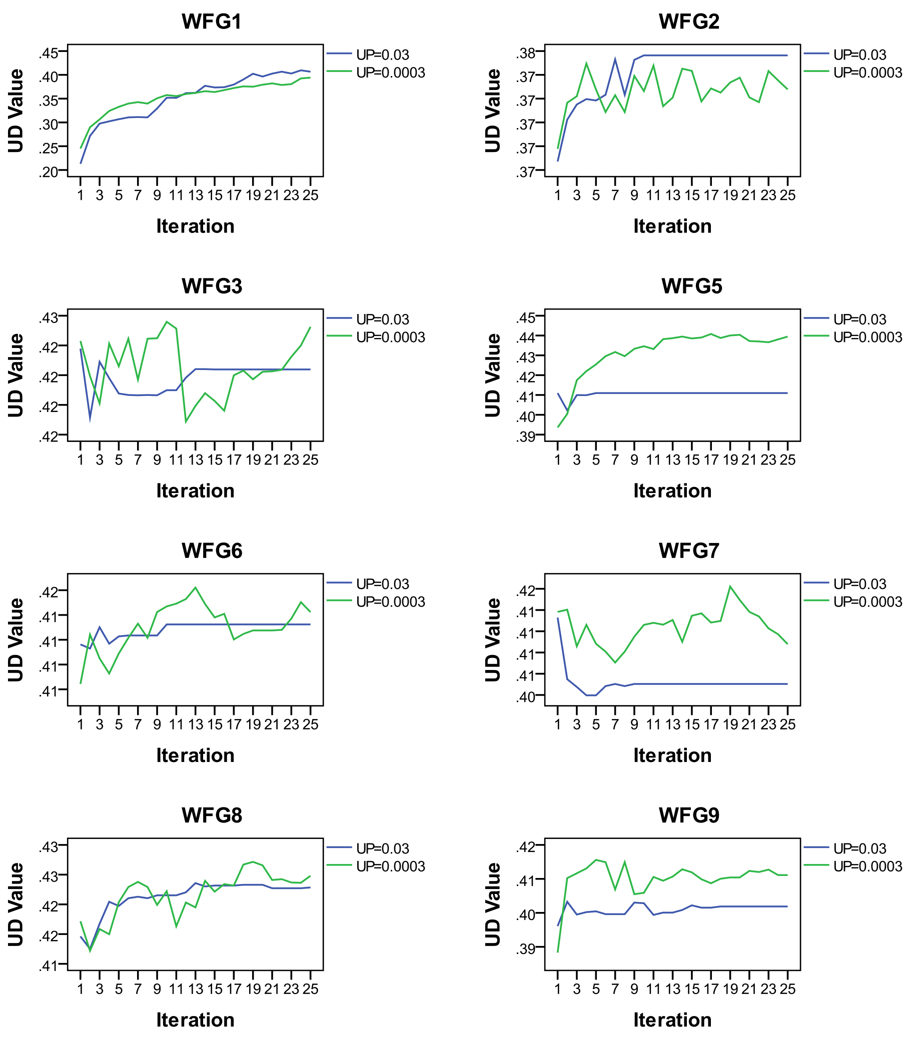


Figure 6.3: The performance HHMO_CF_GDA with different *UP* sizes (0.03 and 0.0003) during the search across 25 decision points with respect to the uniform distribution metric (UD) on the WFG test suite.

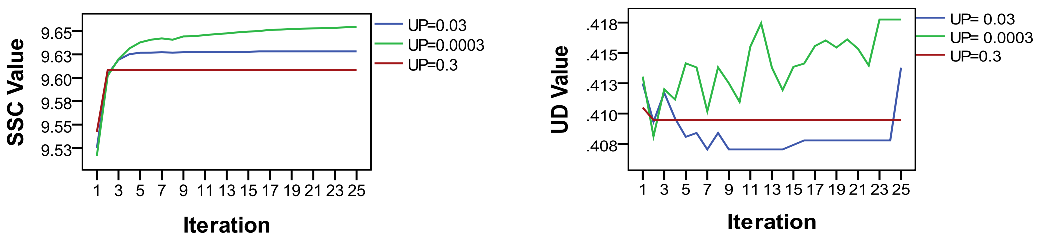


Figure 6.4: The performance HHMO_CF_GDA with different *UP* sizes (0.3 ,0.03 and 0.0003) during the search across 25 decision points with respect to the size of space covered metric (SSC) and the uniform distribution metric (UD) on WFG4.

smallest *UP* value can produce better solutions. The reasons behind this are the level boundary increased quickly with the large *UP* value which leads to reject many moves up to the level.

The average *heuristic utilisation rate,* which indicates how frequently a given low level heuristic is chosen and applied during the search process across all runs on the WFG problems for the HHMO_CF_GDA with *UP* values 0.03 and 0.0003, is computed and illustrated in Figure 6.5. Although the *heuristic utilisation rate* addresses the selection method (choice function) in HHMO_CF_GDA, it can also give some insights about how many moves can be accepted or rejected based on GDA as an acceptance criteria with different *UP* settings. It is clear from Figure 6.5 that acceptance moves mainly happens mostly when *UP*=0.0003, and the most rejected moves happen when *UP*=0.03. This demonstrates that the smaller rain speed value provides a wider boundary space to accept more moves. In WFG1, all moves have been accepted for both rain speed values. This supports the results of Figure 6.1, where the acceptance criteria does not affect the move acceptance because of the wide boundary space. From the above observations, we conclude that GDA with a smaller rain speed value produces better solutions for the WFG test problems.

6.5 Summary and Remarks

We have presented a selection choice function based hyper-heuristic for multi-objective optimisation utilising a great deluge algorithm as a non-deterministic move acceptance strategy (HHMO_CF_GDA). The hyper-heuristic proposed in this chapter differs from the hyper-heuristic that was proposed in Chapter 5 in terms of a move acceptance criteria. Although both hyper-heuristics used the same multi-objective hyper-heuristic framework presented in Chapter 4, choice function great deluge based hyper-heuristic employed a great deluge as a move acceptance method instead of all-move acceptance method which was employed in choice function all-moves based hyper-heuristic (HHMO_CF_AM). The motivation for choosing GDA as an acceptance criteria is that it is simple and does not depend on many parameters, this requiring less effort for parameter tuning. More importantly, encouraging results have been reported in the literature for single-objective optimisation, but there are only a few studies on their application to multi-objective optimisation (e.g., (Petrovic and Bykov, 2003)).

In the context of move acceptance criterion, the quality measure of the current solution and the candidate solution is essential in order to make a decision regarding an acceptance decision. For the single-objective optimisation problem, fitness can be used. However, this is not applicable in multi-objective optimisation. In multi-objective problems, the output is a set of solutions (a non-dominated set). In this thesis, for the first time, we propose the use of *D* metric (Zitzler, 1999) integrating this into the move acceptance criterion, particularly GDA as a way of comparing two non-dominated sets with respect to the objective space, in order to covert the multi-objective optimisation to the single optimisation without definition of criteria values' weights.

We conducted an initial experiment to compare the performance of the proposed great deluge based hyper-heuristics combining the choice function as a selection method and great deluge based hyper-heuristics combined with simple random as a selection method under the multi-objective hyper-heuristic framework. The choice function great deluge outperforms the simple random great deluge over the WFG1 benchmark with respect to the three performance metrics; RNI, SSC and UD. The learning mechanism is essential in our multi-objectives hyper-heuristic framework. It plays a large role in guiding the high level strategy (selection method) in deciding which low level heuristic to call at each decision point. In the absence of a learning mechanism, our multi-objective hyper-heuristic is not that successful. Findings in Chapter 5 support this. The choice function great deluge based hyper-heuristic outperforms the pervious hyper-heuristic method, choice function all-moves based hyper-heuristic. Findings in Chapter 7 will further confirm this.

We experimented with the proposed choice function great deluge based hyper-heretics with different settings of the rain speed parameters (*UP*) to investigate the effectiveness of this parameter on the move acceptance. We assigned different rain speed parameter values; large (0.3), medium (0.03) and small (0.0003) to examine how these setting affect the algorithm and the quality of solutions that ultimately returned. The experimental results show that HHMO_CF_GDA with the smallest *UP* value (0.0003) performs the best for the WFG test problems. In general, the smaller rain speed value allows for the acceptance of more moves that helps escape from the local optimum and produce better solution.

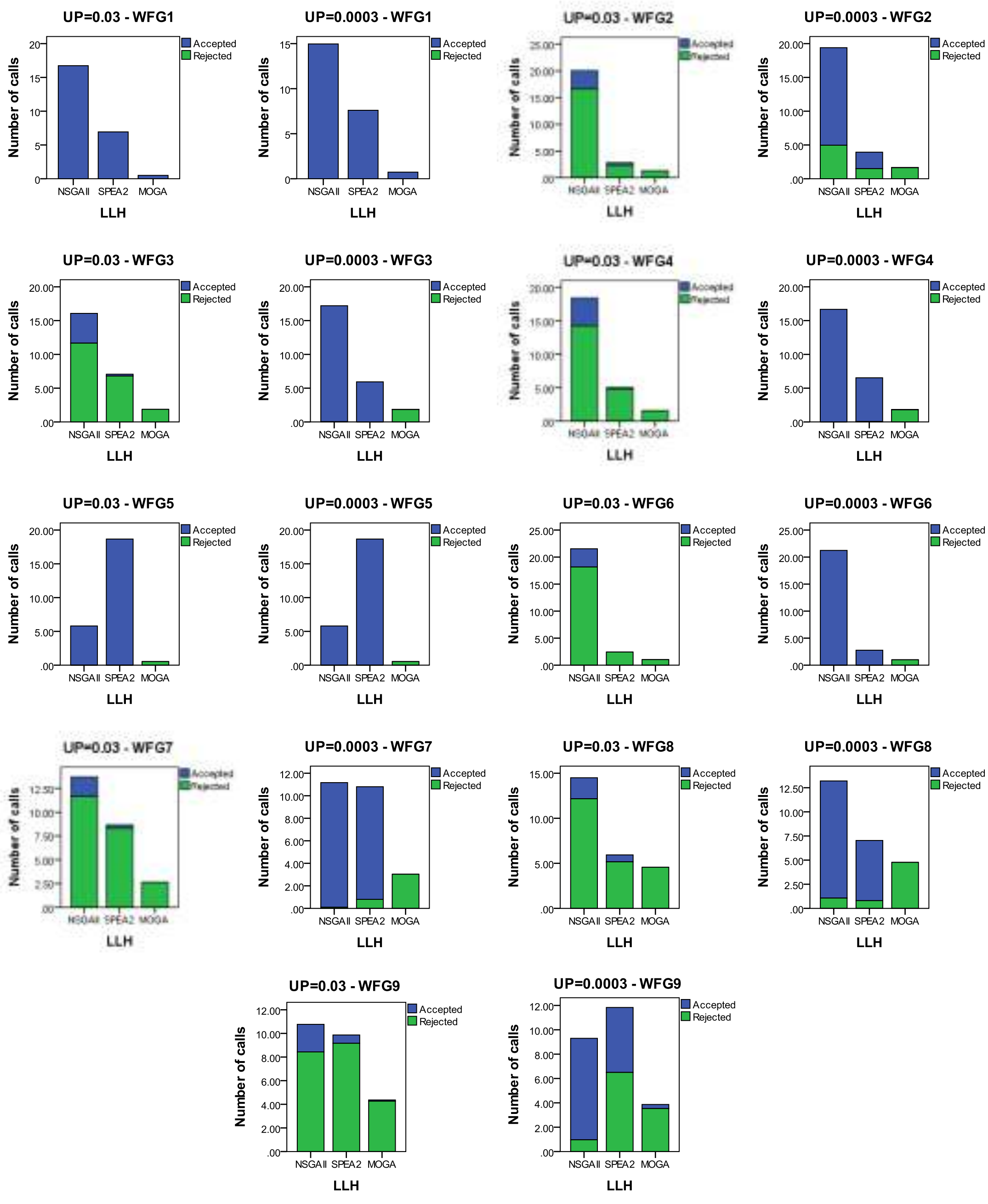


Figure 6.5: The average heuristic utilisation rate for low level heuristic during the search in HHMO_CF_GDA with different sizes of UP (0.03 and 0.0003) on the WFG test suite.

7 A Heuristic Selection Using Late Acceptance as a Non-Deterministic Move Acceptance Strategy

In the previous chapter, we investigated the performance of a selection choice function based hyper-heuristic that utilised the great deluge algorithm (GDA) as a non-deterministic move acceptance criterion. The *D* metric was integrated into GDA as a way of comparing two non-dominated sets in the objective space based on the given acceptance criteria. In this chapter, we further investigate the performance of the choice function based hyper-heuristic that combines the late acceptance strategy (LA) as a non-deterministic move acceptance criterion. We will also conduct computational experiments to compare the performance of the three multi-objective choice function based hyper-heuristic combined with different move acceptance strategies including all-moves, great deluge and late acceptance that were presented in Chapters 5, 6 and this chapter respectively. The comparison will be conducted over the bi-objective and tri-objective Walking Fish Group (WFG) test functions. This chapter is structured as follows. Sections 7.1 and 7.2 introduce late acceptance as a component in a choice function based hyper-heuristic. In Section 7.3, a choice function late acceptance based hyper-heuristic for multi-objective optimisation (HHMO_CF_LA) is proposed. This is followed by computational experiments over bi-objective and tri-objective WFG test function in Sections 7.4 and 7.5 respectively. Section 7.6 concludes the chapter.

7.1 Late Acceptance Strategy as Move Acceptance Criteria

Since late acceptance (LA) is a new methodology, there are only a limited number of studies in literature. There are very few investigations of variant studies, and no multi-objective studies. In Özcan et al. (2009), the late acceptance strategy was combined with different heuristic selection methods (simple random, greedy, reinforcement learning, tabu search and choice function) and applied to examination timetabling problem. The experiments show that the random heuristic selection with late acceptance performs well among other combination methods. In Burke and Bykov (2012) an experimental comparison of LA was presented, along with other well-known search methodologies (simulated annealing (SA), threshold accepting (TA) and GDA) on the travelling salesman and exam timetabling problems. The results show that LA is more reliable and powerful than the others. In Verstichel and Berghe (2009), a number of local search heuristics were combined with the best improving move strategy and LA was presented for solving the lock scheduling problem. The experimental results show that LA has a positive effect on the performance of the heuristics. In Abuhamdah, (2010) and Abuhamdah and Ayob (2010) a variant of LA using randomized descent algorithm (LARD) is proposed to solve university course timetabling. The results demonstrate that the proposed method can beat the original LA in many cases. In Tierney (2013) LA is applied to solve a central problem in the liner shipping industry. LA shows promising performance but it could not beat SA on the same data sets. Yuan et al. (2013) employed LA to solve a two-sided assembly line balancing problem with multiple constraints. The computational results show the effectiveness of LA to solve this kind of problem when compared to an integer programming model and the lower bounds of the problem instances.

LA is successful for single-objective optimisation and it is simple, depending on few parameters. Therefore, we employ LA as a component within our choice function based hyper-heuristic framework for multi-objective optimisation. To the best of our knowledge, no multi-objective LA based studies have been investigated, nor has any work that utilises the LA as a move acceptance component within a hyper-heuristic framework for multi-objective optimisation been reported in the literature. Details about the late acceptance strategy are discussed in Section 2.2.8.

7.2 Late Acceptance and *D* Metric

In a similar way that the *D* metric was integrated into GDA (see Section 6.2), we also integrate *D* metric into LA as a move acceptance strategy.. This is similar to the concept that was used in indicator-based multi-objective optimisers (e.g. (Auger et al.,2012; Wang et al., 2013; Bader and Zitzler, 2011)), Our goal is to maximise the underlying *D* metric, integrating as an acceptance criterion, in order to accept (or reject) a candidate solution (a candidate non-dominated set).

LA is modified to employ the *D* metric. The pseudo code of LA with *D* metric is shown in algorithm 12.

Algorithm 12: The Late Acceptance with *D* Metric

```
 1: procedure LA (A,B, $i$ )
 2:         Calculate $D(A,B)$
 3:         for all $k \in \{0,\dots,l_{fa}-1\}$ doI   0 $C_k = D(A,B)$
 4:             $V = i \; mod \; l_{fa}$
 5:             Calculate $D(B,A)$
 6:             if $D(B,A) \geq C_v$ or $D(B,A) \geq D(A,B)$ then
 7:                 Accept candidate $A = B$
 8:                 Insert cost value into the list $C_v = D(B,A)$
 9:             end if
10: end procedure
```

For an i iteration, *A*, *B* fronts are produced as an initial front and a candidate front respectively. The fitness array is filled by the value of $D(A,B)$ (step 3). Since we are aiming to accept the candidate front *B*, the condition $D(B,A) > D(A,B)$ should be valid (see Section 2.1.9) (step 6). Note that we are always looking to get a higher value (maximise) of $D(B,A)$ in order to accept the candidate solution *B*. In the acceptance case, front *B* is accepted (step 7). The value of $D(B,A)$ is inserted in the fa (step 8), and the value of $D(A,B)$ or C_v is removed from the fa. Note the insertion and removing processes are made virtually in an i iteration using Equation 2.10 (step 4).

7.3 Choice Function Late Acceptance for Selecting Low Level Meta- Heuristics (HHMO_CF_LA)

In this section, we propose multi-objective choice function based hyper-heuristic combined with late acceptance as a non-deterministic acceptance strategy (HHMO_CF_LA). We use the same multi-objective hyper-heuristic framework that was proposed in Chapter 4, including the ranking scheme and the learning mechanism. Three well-known multi-objective evolutionary algorithms (NSGAII, SPEA2, and MOGA), act as low level heuristics. The pseudo code of HHMO_CF_LA for multi-objective optimisation is shown in algorithm 13.

Algorithm 13: Multi-objective Choice Function Late Acceptance based Hyper-heuristic

```
 1: procedure HHMO_CF_LA $(H)$ $whereas\ H$ is a set of the low level heuristics
 2:  Initialisation
 3:  Run $h, \forall h \in H$
 4:  Rank $h, \forall h \in H$ based on the ranking scheme
 5:  Get $CF(h), \forall h \in H$
 6:  Select $h$ with the largest $CF(h)$ as an initial heuristic
 7:  Execute the selected $h$ and produce a front $A$
 8:  Assign the initial number of iterations $i = 0$
 9:  repeat
10:   Update the rank of $h, \forall h \in H$ based on the ranking scheme
11:   Update $CF(h), \forall h \in H$
12:   Select $h$ with the largest $CF(h), \forall h \in H$
13:   Execute the selected $h$ and produce a front $B$
14:   Call the late acceptance procedure LA (A, B, $i$)
        ▻ Algorithm 12: The Late Acceptance with D Metric
15:  Increment the number of iterations $i = i + 1$
16: until (termination criteria are satisfied)
17: end procedure
```

Similar to the previous two multi-objective choice function based hyper-heuristic HHMO_CF_AM and HHMO_CF_GDA, that were proposed in Chapters 5 and 6 respectively, a greedy algorithm is applied at the beginning of the search to determine the best low level heuristic *h* to be selected for the first iteration (steps 2-6). All low level heuristics *H* are executed simultaneously (step 3). Then, the low level heuristics are ranked, based on the ranking scheme using Equation 4.1 (step 4), and their choice function values are computed using Equation 4.4 (step 5). The low level heuristic *h* with the largest choice function value *CF*(*h*) is selected and applied at the next iteration and it produces the non-dominated front *A* (a current solution) (steps 6 & 7). Then, for all low level heuristics *H*, the ranking mechanism is updated (step 9). The choice function values are also computed and updated (step 11). According to the updated choice function values, the low level heuristic *h* with the largest choice function value *CF*(*h*) is called to apply and it produces the non-dominated front *B* (a candidate solution) (steps 12 & 13). In step 14, the acceptance procedure; late acceptance LA (A,B,i) is called and applied using the parameters that were obtained from the search (see algorithm 12). This process is repeated until the stopping condition is met which is a fixed number of iterations (steps 9-17). Note the HHMO_CF_LA is operated in a similar manner to the HHMO_CF_GDA unless the move acceptance criteria that employed are different.

7.4 Comparison of Multi-objective Hyper-heuristics-the Case of Bi-objective

In this section, we conduct experiments over the bi-objective Walking Fish Group (WFG) benchmark dataset (Hunband et al., 2006) to evaluate the performance of our three multi-objective choice function based hyper-heuristics for multi-objective optimisation using different move acceptance strategies including all-moves as a deterministic move acceptance, and the great deluge algorithm and late acceptance as non-deterministic move acceptance functions. Experiments are also conducted to investigate the influence of using non-deterministic move acceptance strategies; great deluge algorithm and late acceptance on the performance of online learning selection choice function based hyper-heuristic for multi-objective optimisation.

7.4.1 Performance Evaluation Criteria

We used five performance metrics to measure the quality of approximation sets from different aspects: (i) ratio of non-dominated individuals (RNI) (Tan et al., 2002), (ii) hypervolume (SSC) (Zitzler and Thiele, 1999) (iii) uniform distribution of a non-dominated population (UD) (Srinivas and Deb, 1994), (iv) generational distance (GD) (Van Veldhuizen and Lamont, 1998b) and (v) inverted generational distance (IGD) (Coello and Cruz Cortès, 2005). A higher value considering one of those performance metrics indicates that non-dominated solutions have a good quality, except for GD and IGD, where a lower value indicates that the approximation nondominated front is closer to the POF.

We have compared the mean performance of three multi-objective choice function based hyper-heuristics; HHMO_CF_AM, HHMO_CF_GDA and HHMO_CF_LA across multiple trials with respect to the metrics across multiple trials. *t*-test is used as a statistical test for pairwise mean performance comparison of selection hyper-heuristics.

7.4.2 Experimental Settings

All experimental parameters are chosen according to those commonly used in the literature for continuous problems (see (Zitzler et al. (2000) and

Huband et al. (2006)). We use the same parameter settings that were used in Sections 5.2.2 and 6.4.2 for a fair comparison. In the measure of SSC and *D* metric for GDA and LA, the reference points for WFG problems with k objectives was set $r_i = (0, i * 2), i = 1, \ldots, k$ (Huband et al., 2006). As for HHMO_CF_GDA, the rain speed (*UP*) is set to 0.0003 based on the empirical experiments that are presented in Chapter 6. The length of the fitness array l_{fa} in HHMO_CF_LA is set to 5 as recommended in Burke and Bykov (2012). All methods are implemented using Microsoft Visual C++ 2008 on an Intel Core2 Duo 3GHz\2G\250G computer.

7.4.3 Experimental Results and Discussion

The average, minimum, maximum and standard deviation considering the performance metrics, including RNI, SSC, UD, GD and IGD for each WFG problem generated by each hyper-heuristic across 30 trials are provided in Tables 7.1 and 7.2. From this point onward, each hyper-heuristic will be referred to by move acceptance method utilised within each hyper-heuristic.

The pairwise mean performance comparison of different selection choice function based hyper-heuristics, each using a different move acceptance method, are provided in Table 7.3 based on *t*-test. The box plots of RNI, SSC, UD, GD and IGD values for each bi-objective WFG benchmark function using AM, GDA and LA are also illustrated in Figures 7.1, 7.2, 7.3, 7.4 and 7.5. The performance of the choice function based hyper-heuristics are statistically different in the majority cases. In general, AM, GDA, LA are statistically different from each other (i.e. we reject the null hypothesis). In the overall, GDA performs the best.

From Figure 7.1, we note that the selection hyper-heuristic using GDA and LA perform better than the one using AM on average with respect to the measure of ratio non-dominated solutions (RNI). The pairwise performance differences of GDA and LA from AM are statistically significant for all benchmark functions with respect to RNI, except WFG1. GDA and LA perform relatively similar (see Table 7.2).

WFG	Methods	RNI				SSC (HV)				UD			
		AVG	MIN	MAX	STD	AVG	MIN	MAX	STD	AVG	MIN	MAX	STD
1	AM	0.8800	0.2800	1.0000	0.2539	12.1386	9.0338	12.5130	**0.9101**	**0.4428**	0.3490	0.6945	0.1007
	GDA	0.9357	0.3100	1.0000	0.1821	**12.9388**	8.2543	12.9966	1.2517	0.3941	0.2047	0.5952	0.0698
	LA	**0.9950**	0.8400	1.0000	**0.0292**	12.1867	6.4458	12.3515	0.9967	0.3117	0.1178	0.3800	**0.0521**
2	AM	0.2293	0.1600	0.3600	0.0545	11.0219	10.6407	12.3894	0.3042	**0.7278**	0.6223	1.0000	0.0661
	GDA	**1.0000**	1.0000	1.0000	**0.0000**	**11.8148**	10.7433	11.8258	**0.0146**	0.3729	0.3609	0.3862	**0.0064**
	LA	**1.0000**	1.0000	1.0000	**0.0000**	11.8139	10.7242	11.9365	0.1567	0.3716	0.3158	0.4055	0.0156
3	AM	0.6027	0.5200	0.6800	0.0445	11.8940	11.3990	11.9867	0.0853	**0.5450**	0.4959	0.6136	0.0289
	GDA	**1.0000**	1.0000	1.0000	**0.0000**	**11.9197**	11.9094	11.9296	**0.0064**	0.4252	0.4059	0.4580	0.0120
	LA	**1.0000**	1.0000	1.0000	**0.0000**	11.9093	11.8232	11.8933	0.0162	0.4222	0.3976	0.4352	**0.0094**
4	AM	0.5443	0.4800	0.6400	0.0452	9.6588	9.5331	9.6643	0.0176	**0.5596**	0.4752	0.6317	0.0361
	GDA	**1.0000**	1.0000	1.0000	**0.0000**	**9.6642**	9.6210	9.6650	**0.0100**	0.4145	0.3879	0.4423	**0.0112**
	LA	**1.0000**	1.0000	1.0000	**0.0000**	9.6512	9.5685	9.6330	0.0141	0.4150	0.3860	0.4402	0.0143
5	AM	0.8537	0.6000	1.0000	0.1723	9.2899	9.1526	9.2984	0.5744	**0.4779**	0.4279	0.5744	0.0468
	GDA	**1.0000**	1.0000	1.0000	**0.0000**	**9.2964**	9.1526	9.2984	0.4023	0.4395	0.4238	0.4579	**0.0086**
	LA	**1.0000**	1.0000	1.0000	**0.0000**	9.2772	9.2580	9.2859	**0.0080**	0.4170	0.3733	0.4484	0.0213
6	AM	0.4720	0.4000	0.5600	0.0412	9.3687	9.1500	9.3810	0.0542	**0.5962**	0.5042	0.6479	0.0363
	GDA	**1.0000**	1.0000	1.0000	**0.0000**	**9.3745**	9.2346	9.4787	0.0628	0.4128	0.3992	0.4308	**0.0083**
	LA	**1.0000**	1.0000	1.0000	**0.0000**	9.3711	9.2495	9.4553	**0.0474**	0.4136	0.3927	0.4377	0.0129
7	AM	0.6173	0.4000	0.7200	0.0653	9.6606	9.2261	9.6911	0.0926	**0.5289**	0.4734	0.6743	0.0416
	GDA	**1.0000**	1.0000	1.0000	**0.0000**	**9.6650**	9.6596	9.6700	**0.0028**	0.4085	0.3792	0.4565	0.0151
	LA	**1.0000**	1.0000	1.0000	**0.0000**	9.6641	9.6172	9.6550	0.0100	0.4112	0.3878	0.4342	**0.0133**
8	AM	0.2627	0.2000	0.4400	0.0454	8.3033	8.1155	8.5676	0.1224	**0.7886**	0.6294	1.0000	**0.1245**
	GDA	**1.0000**	1.0000	1.0000	**0.0000**	**8.7279**	8.6708	8.7389	**0.0120**	0.4248	0.3948	0.5933	0.0341
	LA	**1.0000**	1.0000	1.0000	**0.0000**	8.4859	8.3572	8.6371	0.0754	0.4128	0.3832	0.4488	0.0136
9	AM	0.6410	0.4000	0.8000	0.0896	8.6132	8.2356	9.2519	**0.2236**	**0.5142**	0.4141	0.6432	0.0525
	GDA	0.9893	0.8000	1.0000	0.4193	**8.7689**	8.5789	9.4346	0.3054	0.4111	0.3661	0.6141	0.0210
	LA	**0.9973**	0.9200	1.0000	**0.0146**	8.7132	8.5373	9.2002	0.2518	0.3953	0.3508	0.4201	**0.0144**

Table 7.1: The performance of selection choice function based hyper-heuristics using different move acceptance strategies including all-moves (AM), great deluge algorithm (GDA) and late acceptance (LA) on the bi-objective WFG test problems with respect to the metrics; the ratio of non-dominated individuals (RNI), the hypervolume (SSC), the uniform distribution (UD).

WFG	Methods	GD				IGD			
		AVG	MIN	MAX	STD	AVG	MIN	MAX	STD
1	AM	**7.740E-03**	3.400E-03	4.660E-02	1.106E-02	**7.300E-04**	3.900E-04	2.930E-03	**6.500E-04**
	GDA	8.240E-03	3.400E-03	4.400E-02	1.110E-02	1.020E-03	3.900E-04	4.940E-03	1.170E-03
	LA	1.534E-02	8.200E-03	4.110E-02	**7.260E-03**	2.400E-03	1.680E-03	4.740E-03	7.200E-04
2	AM	1.460E-03	9.000E-04	3.200E-03	4.900E-04	4.400E-04	2.200E-04	5.400E-04	6.000E-05
	GDA	**4.500E-04**	4.000E-04	8.000E-04	**8.000E-05**	**3.500E-04**	3.500E-04	3.600E-04	**0.000E+00**
	LA	7.000E-04	4.000E-04	4.500E-03	7.300E-04	3.700E-04	3.500E-04	7.800E-04	8.000E-05
3	AM	6.800E-04	3.000E-04	2.800E-03	4.500E-04	6.853E-04	6.831E-04	7.229E-04	7.169E-06
	GDA	**2.000E-04**	1.900E-04	3.000E-04	**5.000E-05**	**6.835E-04**	6.830E-04	6.839E-04	**2.502E-07**
	LA	4.100E-04	3.000E-04	6.000E-04	7.000E-05	6.836E-04	6.813E-04	6.856E-04	1.184E-06
4	AM	9.700E-04	7.500E-04	1.510E-03	1.900E-04	2.563E-04	1.951E-04	3.311E-04	3.384E-05
	GDA	**4.700E-04**	4.300E-04	5.800E-04	4.000E-05	**1.297E-04**	1.169E-04	1.613E-04	**1.027E-05**
	LA	6.100E-04	5.400E-04	7.500E-04	**5.000E-05**	1.475E-04	1.343E-04	1.830E-04	1.099E-05
5	AM	2.730E-03	2.160E-03	2.440E-03	3.200E-04	5.439E-04	5.281E-04	5.987E-04	2.246E-05
	GDA	**2.450E-03**	2.430E-03	2.430E-03	**1.000E-05**	**5.292E-04**	5.278E-04	5.304E-04	**6.672E-07**
	LA	2.510E-03	2.460E-03	2.460E-03	3.000E-05	5.394E-04	5.293E-04	5.612E-04	8.862E-06
6	AM	2.250E-03	1.500E-03	3.900E-03	5.600E-04	5.523E-04	4.265E-04	7.191E-04	6.749E-05
	GDA	**2.000E-03**	1.310E-03	2.700E-03	3.500E-04	**4.441E-04**	2.850E-04	5.791E-04	7.680E-05
	LA	2.050E-03	1.420E-03	2.550E-03	**2.700E-04**	4.470E-04	3.089E-04	5.503E-04	**5.602E-05**
7	AM	4.700E-04	4.400E-04	1.360E-03	2.500E-04	2.206E-04	1.736E-04	4.141E-04	5.025E-05
	GDA	**3.300E-04**	2.600E-04	4.100E-04	**4.000E-05**	**1.191E-04**	1.090E-04	1.392E-04	**7.968E-06**
	LA	4.100E-04	2.900E-04	5.000E-04	**4.000E-05**	1.323E-04	1.096E-04	1.471E-04	1.185E-05
8	AM	4.420E-03	3.580E-03	4.980E-03	4.300E-04	6.195E-04	4.806E-04	7.753E-04	7.767E-05
	GDA	**3.890E-03**	3.580E-03	5.850E-03	3.800E-04	**3.634E-04**	3.426E-04	4.198E-04	1.397E-05
	LA	4.410E-03	4.080E-03	4.710E-03	**1.500E-04**	4.205E-04	3.863E-04	4.572E-04	**1.371E-05**
9	AM	5.280E-03	1.430E-03	6.390E-03	**1.450E-03**	9.545E-04	3.122E-04	1.176E-03	**2.444E-04**
	GDA	**3.640E-03**	4.100E-04	5.500E-03	1.950E-03	**7.879E-04**	1.369E-04	1.025E-03	3.908E-04
	LA	3.770E-03	5.700E-04	4.950E-03	1.690E-03	8.312E-04	1.787E-04	1.031E-03	3.538E-04

Table 7.2: The performance of selection choice function based hyper-heuristics using different move acceptance strategies including all-moves (AM), great deluge algorithm (GDA) and late acceptance (LA) on the bi-objective WFG test problems with respect to the metrics; the generational distance (GD) and the inverted generational distance (IGD).

Problem	Methods	Metrics				
		RNI	SSC	UD	GD	IGD
WFG1	AM:GDA	∓	−	+	±	+
	AM:LA	−	−	+	+	+
	GDA:LA	∓	+	±	±	+
WFG2	AM:GDA	−	−	+	−	−
	AM:LA	−	−	+	−	−
	GDA:LA	n/a	±	±	+	+
WFG3	AM:GDA	−	−	+	−	∓
	AM:LA	−	−	+	−	∓
	GDA:LA	n/a	+	±	+	±
WFG4	AM:GDA	−	−	+	−	−
	AM:LA	−	−	±	−	−
	GDA:LA	n/a	+	∓	+	+
WFG5	AM:GDA	−	−	+	−	−
	AM:LA	−	±	+	−	∓
	GDA:LA	n/a	+	+	+	±
WFG6	AM:GDA	−	−	+	−	−
	AM:LA	−	−	+	−	−
	GDA:LA	n/a	+	∓	+	±
WFG7	AM:GDA	−	−	+	−	−
	AM:LA	−	−	+	−	−
	GDA:LA	n/a	+	∓	+	±
WFG8	AM:GDA	−	−	+	−	−
	AM:LA	−	−	+	−	−
	GDA:LA	n/a	+	+	+	+
WFG9	AM:GDA	−	−	+	−	−
	AM:LA	−	−	+	−	−
	GDA:LA	∓	+	+	±	+

Table 7.3: The *t*-test results of multi-objective choice function based hyper-heuristics methodologies using AM, GDA and LA as a move acceptance criterion on the bi-objective WFG test problems with respect to the metrics; the ratio of non-dominated individuals (RNI), the hypervolume (SSC), the uniform distribution (UD) and the generational distance (GD).

From Figure 7.2 and Table 7.2, GDA has the best overall mean performance when compared to AM and LA. With respect to the measure of the hypervolume (SSC), this performance difference is statistically significant across all WFG problems, except WFG2. For this instance, GDA performs slightly better than LA. In addition, LA delivers a significantly better performance than AM for all WFG problems, except WFG5. Similarly, GDA delivers a significantly better mean performance when compared to AM and LA with respect to the measure of generational distance (GD) for all benchmark functions, except WFG1 and WFG9 (See Figure 7.4 and Table 7.2).

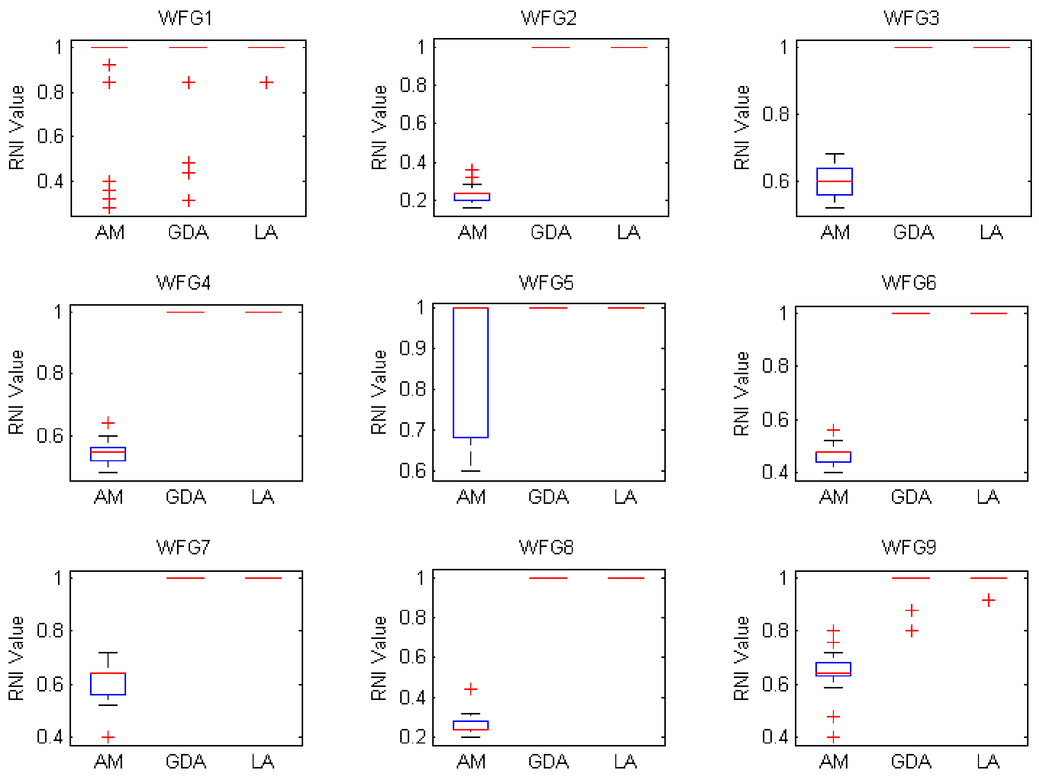


Figure 7.1: Box plots of multi-objective choice function based hyper-heuristics methodologies using AM, GDA and LA as a move acceptance criterion on the bi-objective WFG test problems for the measure of ratio non-dominated solutions (RNI).

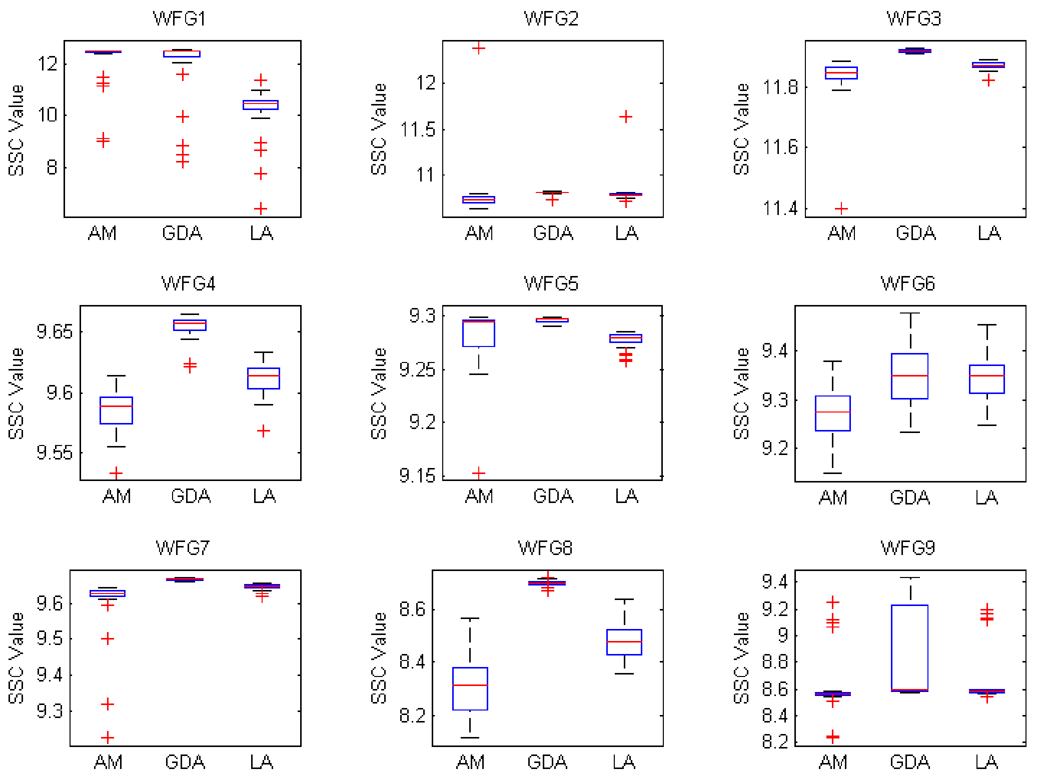


Figure 7.2: Box plots of multi-objective choice function based hyper-heuristics methodologies using AM, GDA and LA as a move acceptance criterion on the bi-objective WFG test problems for the measure of hypervoulme (SSC).

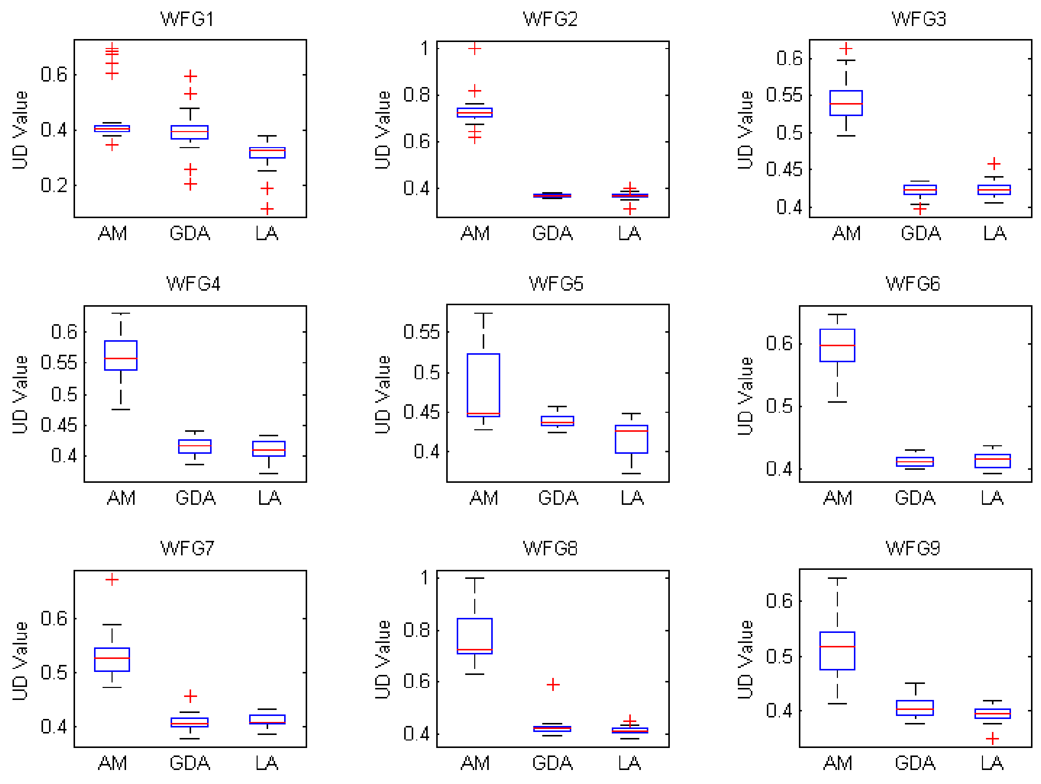


Figure 7.3: Box plots of multi-objective choice function based hyper-heuristics methodologies using AM, GDA and LA as a move acceptance criterion on the bi-objective WFG test problems for the measure of uniform distribution (UD).

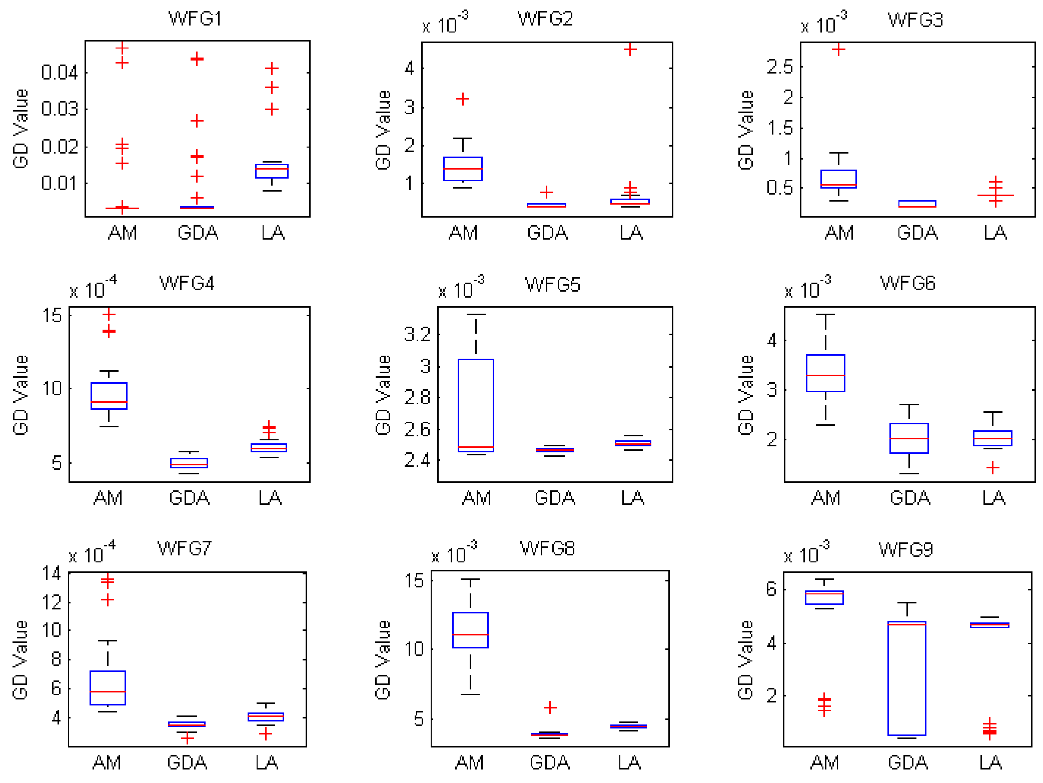


Figure 7.4: Box plots of multi-objective choice function based hyper-heuristics methodologies using AM, GDA and LA as a move acceptance criterion on the bi-objective WFG test problems for the measure of generational distance (GD).

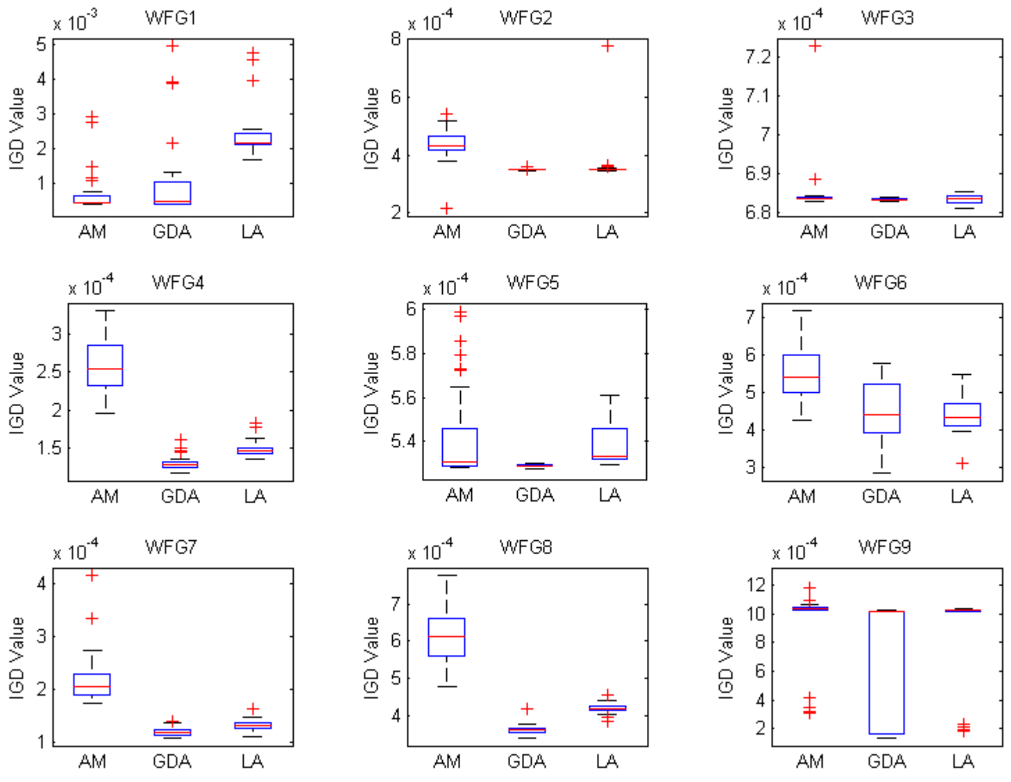


Figure 7.5: Box plots of multi-objective choice function based hyper-heuristics methodologies using AM, GDA and LA as a move acceptance criterion on the bi-objective WFG test problems for the measure of inverted generational distance (IGD).

For WFG1, AM performs slightly better than GDA and significantly better than LA, while for WFG9, LA performs significantly better than AM and GDA performs slightly better than AM. With respect to the measure of inverted generational distance (IGD), GDA performs significantly better than AM in all instances except in WFG1. In addition, GDA performs significantly better than LA in four instances of WFG2, WFG4, WFG8 and WFG9 while it performs significantly similar to LA in the rest (see Figure 7.5).

Although non-deterministic move acceptance methods improve the overall mean performance of the hyper-heuristic with respect to RNI, SSC, GD and IGD, AM performs the best with respect to the measure of the uniform distribution of non-dominated solutions (UD) (see Figure 7.3). The performance differences from GDA and LA are statistically significant for all problems, except WFG4, for which AM still performs slightly better than LA. GDA and LA have relatively similar performance across all WFG problems (see Table 7.2). The success of AM with respect to UD might be due to the use of the *D* metric into acceptance procedure. Since *D* metric is a binary hypervolume measure that is designed to compare two sets of non-dominated solutions with respect of their convergence towards the POF, there is no consideration regarding how uniformly these solutions are distributed along the POF. This might also be a reason for why non-deterministic move acceptance produces high quality solutions in terms of the convergence towards the POF.

7.4.4 Behaviour of Acceptance Strategies

To further understand how the move acceptance strategies, AM, GDA and LA, are performing and how their performances could affect the quality of the solutions, we compute the average *accepted/rejected move rates* which indicates how frequently a move (solution) that is produced from the three low level heuristics is accepted/ rejected under different acceptance methods AM, GDA and LA. Figure 7.6 illustrates the average number of heuristic invocations of each low level heuristic selected and applied at 25 consecutive decision points (stages/iterations) during the search process over all runs. Each bar in the plot also indicates the average number of accepted and rejected Pareto fronts. A similar pattern for the choice of low level heuristics during the search process has been observed in Figure 7.6 on almost all WFG problems considering the three hyper-heuristics. This is high likely due to the use of the same heuristic selection mechanism (choice function). However, the pattern in the plots for accepted or rejected Pareto fronts produced by the chosen low level heuristic varies for a given problem depending on the move acceptance strategy that the hyper-heuristic employs. NSGAII is always selected more than the other low level meta-heuristics regardless of the move acceptance method, except for WFG5 and WFG9. For WFG5, SPEA2 is the most frequently chosen algorithm regardless of the move acceptance component of the hyper-heuristic during the search process. On the other hand, SPEA2 is frequently chosen when GDA is used as the move acceptance algorithm on WFG9. The performance of MOGA is the worst among three hyper-heuristics on the WFG problems; thus it is invoked relatively less frequently during the search process in all test problems for all methods.

Overall, NSGAII appears to be a good choice for solving the WFG problems. Our observations are consistent with the result obtained in Bradstreet et al. (2007) showing that the best performance is achieved by NSGAII on the bi-objective WFG test suite. This indicates that NSGAII is a good choice for solving the WFG problems. We theorise that the multi-

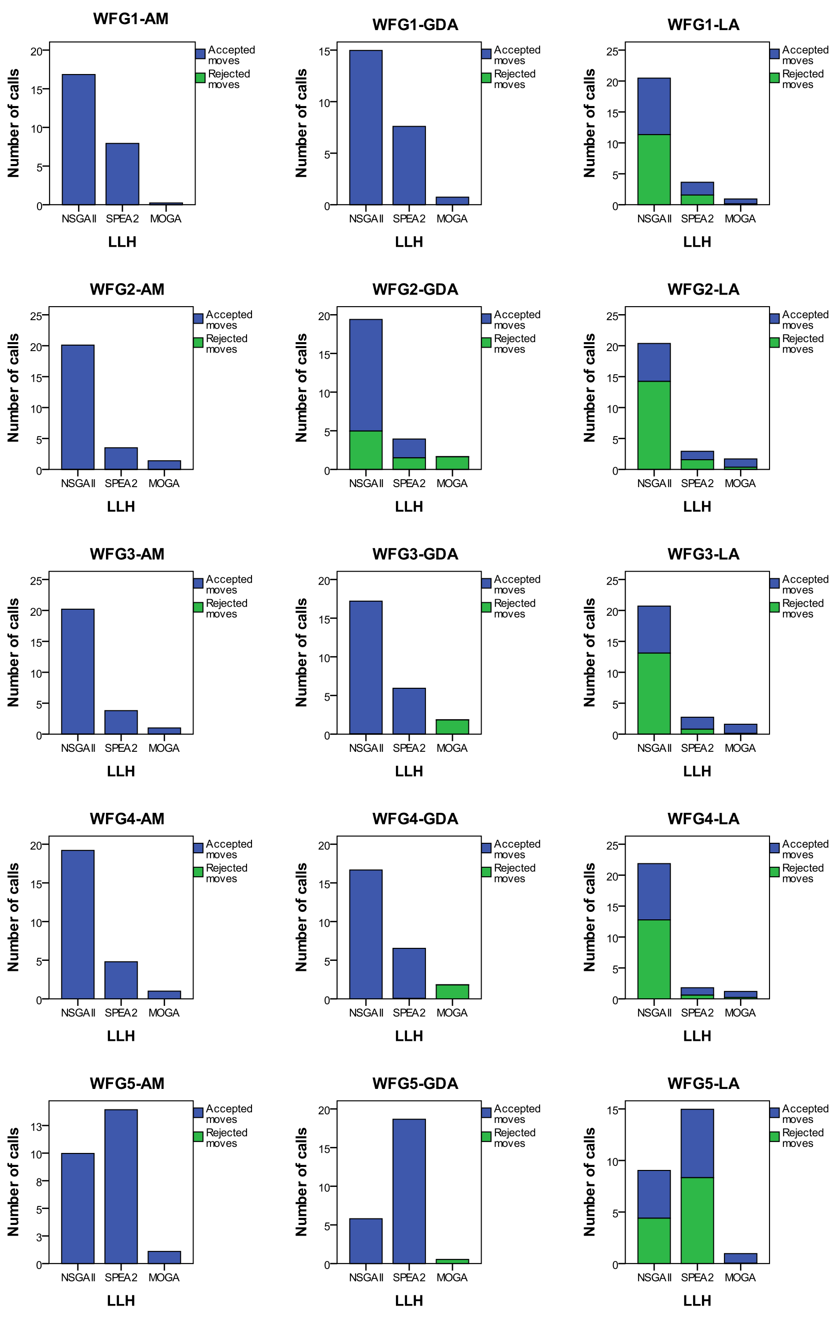


Figure 7.6: The average number of low level meta-heuristic invocations (NSGAII, SPEA2 and MOGA) and accepted/rejected moves produced by selection hyper-heuristics using AM, GDA and LA over the bi-objective WFG test problems- continue.

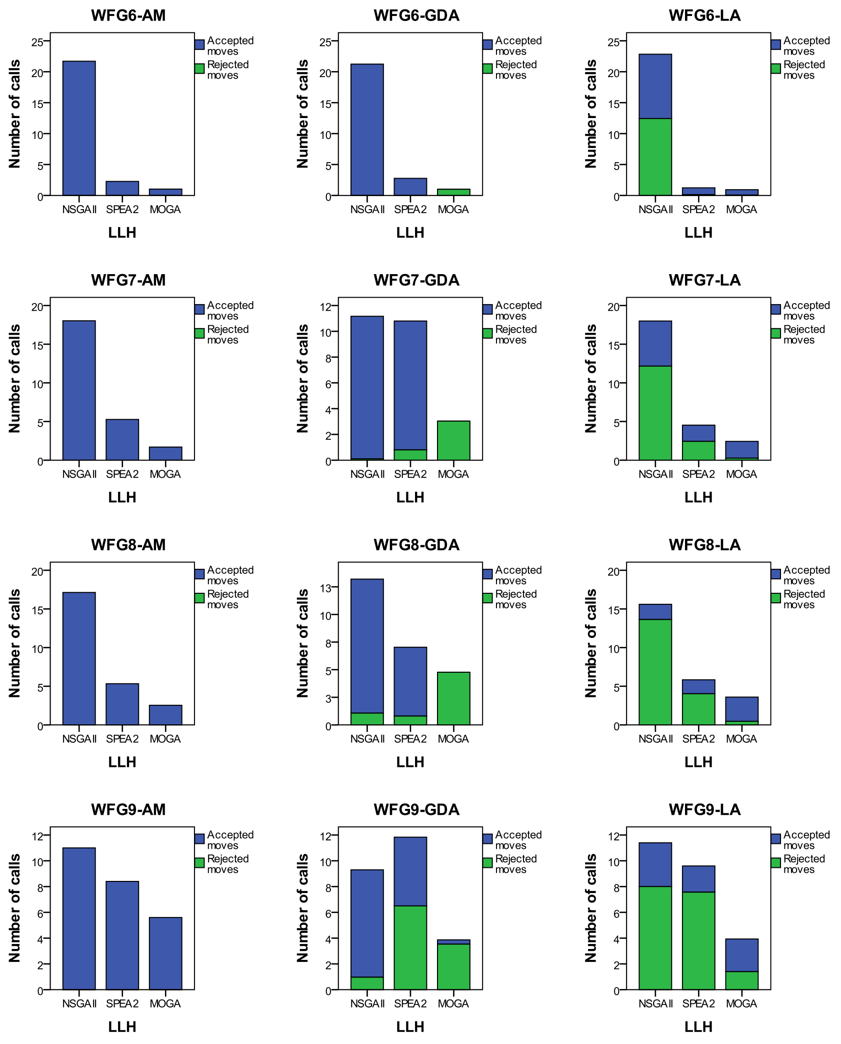


Figure 7.6: Continue- the average number of low level meta-heuristic invocations (NSGAII, SPEA2 and MOGA) and accepted/rejected moves produced by selection hyper-heuristics using AM, GDA and LA over the bi-objective WFG test problems.

objective choice function hyper-heuristic, therefore, prefers NSGAII and it becomes preferable to be chosen more frequently than the other low level heuristics.

Figure 7.6 shows that there is only one case in which all moves are accepted when a non-deterministic strategy is used, that is GDA for WFG1. The rate of moves rejected for LA is higher than that for GDA on all test problems regardless of the low level meta-heuristic employed, except for MOGA, where LA accepts more moves (solutions) than GDA on almost all problems. These observations offer some explanation as to why the performance of GDA is better than LA in terms of convergence towards the

POF: (i) The good moves that are accepted in GDA are rejected in LA, and (ii) as MOGA does not perform well in the WFG test problem and it is invoked relatively less frequently during the search process, LA accepts all MOGA's moves (solutions) while GDA rejects them. LA produces better solutions than AM. So, the non-deterministic acceptance strategies (GDA and LA) beat the deterministic acceptance strategy (AM). In addition, GDA and LA appear to positively affect the performance of the multi-objective choice function based hyper-heuristic when used as the move acceptance strategy over the bi-objective WFG test problems.

7.5 Comparison of Multi-objective Hyper-heuristics- the Case of Tri-objective

More experiments are conducted to evaluate the performance of the three proposed selection online learning choice function based hyper-heuristics for multi-objective optimisation (HHMO_CF_AM, HHMO_CF_GDA and HHMO_CF_LA) over tri-objective Walking Fish Group (WFG) benchmark dataset (Huband et al., 2006). The performance of our selection choice function based hyper-heuristics is compared to the well-known multi-objective evolutionary algorithm, SPEA2 (Zitzler et al., 2001) as well. The motivation behind choosing SPEA2 to compare against our multi-objective hyper-heuristic is that SPEA2 performs well on the WFG problems in three objectives (Bradstreet et al., 2007). For brevity, we will refer to three multi-objective choice function based hyper-heuristics; HHMO_CF_AM, HHMO_CF_GDA and HHMO_CF_LA as AM, GDA and LA respectively.

7.5.1 Performance Evaluation Criteria

We used three performance metrics- the hypervolume- size of space converged (SSC) (Zitzler and Thiele, 1999), generational distance (GD) (Van Veldhuizen and Lamont, 1998b) and inverted generational distance (IGD) (Coello and Cruz Cortès, 2005)- to assess the quality of approximation sets in both diversity and convergence aspects. In addition, we use the students test (*t*-test) statistic to compare the mean performance of SPEA2 and three choice function based multi-objective hyper-heuristics using different acceptance criteria; AM, GDA and LA across multiple trials with respect to the metrics across multiple trials. We use the same notation that was presented in Section 7.4.1.

7.5.2 Experimental Settings

All experimental parameters are chosen according to those commonly used in the scientific literature for the tri-objective problems (Huang et al., 2007; Zielinski and Laur, 2007). The nine test problems (WFG1-WFG9) with three objectives have 24 real parameters including four position parameter and 20 distance parameters. For each problem, we run 30 independent trials with a different random seed. For fair comparison, all methods in each run were executed 300,000 evaluation functions in order to keep the computational costs of the experiments in an affordable level. In other words, all hyper-heuristics are run for a total of 30 stages (iterations). In each stage, a low level heuristic is chosen and applied to execute 100 generations with a population size equal to 100 (10,000 evaluation functions). SPEA2 executed for 300,000 evaluation functions (3000 generations in total with primary and secondary population sizes equal to 100). Other parameter settings are identical to those used in Section 7.4.2. All methods were implemented with the same common sub-functions using Microsoft Visual C++ 2008 on an Intel Core2 Duo 3GHz\2G\250G computer.

7.5.3 Experimental Results and Discussion

The statistical *t*-test results of comparing three multi-objective choice function based hyper-heuristics (AM, GDA and LA) and SPEA2 with respect to the three performance metrics (SSC, GD and IGD) on the nine WFG test problems are given in Table 7.4. We can note that our multi-objective choice function based hyper-heuristics are statistically different from SPEA2 in the majority cases (i.e. we reject the null hypothesis) except in AM whilst performs similar to SPEA2 in most cases for the SSC metric.

The performance values of SPEA2 and our three multi-objective hyper-heuristic methodologies (AM, GDA and LA) with respect to the performance metrics (SSC, GD and IGD) on the tri-objective WFG function are summarised in Table 7.5. For each performance metric, the average, minimum, maximum and standard deviation values are shown. We also visualise the distribution of the simulation data of the 30 independent runs for the comparison methods with respect to these performance metrics shown in Figures 7.7 -7.9. A higher value indicates a better performance in SSC while a lower value indicates a better performance in GD and IGD.

We note from both Table 7.5 and Figures 7.7-7.9. GDA has the highest SSC value in five out of nine problems including WFG1, WFG3, WFG5, WFG7, WFG9 while LA has the highest SSC's value for the rest of the WFG problems. The pairwise performance differences of GDA from other methods are statistically significant for all benchmark functions with respect to the measure of hypervoulme (SSC). It is interesting to note that AM and SPEA2 are performing similarly in the majority of cases for SSC. With respect to the measure of generational distance (GD), GDA has the lowest GD's value which means it has the best performance among other methods for all WFG problems except in WFG3 where SPEA2 performs the best. In contrast, LA has the highest GD value, thus the worst performance among the comparison methods. With respect to the measure of inverted generational distance (IGD), GDA has the lowest GD value which means it has the best performance among other methods for all WFG problems except in WFG1 and WFG9.

Generally, GDA is statistically significant better than AM, LA and SPEA2 in the most cases with respect to the SSC, GD and IGD metrics. Although AM and LA perform better than SPEA2 in the measure of SSC for all WFG problems, they perform worse than SPEA2 in the measure of GD and IGD in the majority cases. The superiority of our multi-objective hyper-heuristics compared to SPEA2 in the SSC metric is because of the influence of the ranking scheme that is embedded in the selection mechanism (the choice function). The ranking scheme maintains the past performance of low level heuristics using a set of performance indicators that measure different aspects of the solutions, the SSC metric is one of these indicators.

In Figure 7.10, we have plotted the POF and the distribution of the final fronts obtained in the run with the lowest GD value of each method in each WFG problem. It is clear that the GDA is converging well (closer to the POF) compared to the other methods for all the datasets. However, GDA shows poor distribution of final solutions in WFG8 and WFG9. This could be attributed to the fact that WFG8 and WFG9 feature significant bias which causes difficulty to the algorithm to spread well along the front. We can observe that the multi-objective selection hyper-heuristic that utilised the GDA as a move acceptance criterion outperforms SPEA2 and the other move acceptance criteria AM and LA in most WFG problems with three objectives.

Problem	Methods	Metrics		
		SSC	GD	IGD
WFG1	AM:GDA	∓	−	±
	AM:LA	−	±	±
	AM:SPEA2	∓	±	±
	GDA:LA	+	+	∓
	GDA:SPEA2	+	+	∓
	LA:SPEA2	−	∓	∓
WFG2	AM:GDA	∓	−	−
	AM:LA	−	−	±
	AM:SPEA2	∓	±	∓
	GDA:LA	−	+	+
	GDA:SPEA2	∓	+	+
	LA:SPEA2	+	+	∓
WFG3	AM:GDA	−	+	∓
	AM:LA	∓	∓	+
	AM:SPEA2	+	−	∓
	GDA:LA	+	−	+
	GDA:SPEA2	+	−	+
	LA:SPEA2	±	−	+
WFG4	AM:GDA	−	−	∓
	AM:LA	−	+	∓
	AM:SPEA2	∓	±	∓
	GDA:LA	∓	+	+
	GDA:SPEA2	+	+	+
	LA:SPEA2	+	−	∓
WFG5	AM:GDA	+	+	−
	AM:LA	+	−	+
	AM:SPEA2	±	−	+
	GDA:LA	+	+	+
	GDA:SPEA2	+	+	+
	LA:SPEA2	+	−	+
WFG6	AM:GDA	−	+	∓
	AM:LA	−	−	+
	AM:SPEA2	∓	−	∓
	GDA:LA	−	+	+
	GDA:SPEA2	+	−	±
	LA:SPEA2	+	+	−
WFG7	AM:GDA	−	−	−
	AM:LA	−	+	+
	AM:SPEA2	∓	±	−
	GDA:LA	±	+	+
	GDA:SPEA2	+	+	+
	LA:SPEA2	+	−	−
WFG8	AM:GDA	−	−	∓
	AM:LA	−	∓	∓
	AM:SPEA2	−	−	−
	GDA:LA	−	+	±
	GDA:SPEA2	+	+	±
	LA:SPEA2	+	−	−
WFG9	AM:GDA	−	−	−
	AM:LA	−	∓	∓
	AM:SPEA2	−	∓	−
	GDA:LA	+	+	+
	GDA:SPEA2	+	±	∓
	LA:SPEA2	+	−	−

Table 7.4: The *t*-test results of SPEA2 and three multi-objective choice function based hyper-heuristics using all-move (AM), great deluge algorithm (GDA) and late acceptance (LA) as a move acceptance criterion with respect to the metrics; the hypervolume (SSC), the generational distance (GD) and the inverted generational distance (IGD) on the tri-objective WFG test problems.

To understand why GDA works so well as an acceptance strategy and outperforms the others, in the next subsection, we analyse the behaviour of the move acceptance strategies and how many moves are *accepted/rejected* based on these acceptance strategies.

7.5.4 Behaviour of Acceptance Strategies

In order to understand how the move acceptance strategies, AM, GDA and LA, are performing and how their performances could affect the quality of the solutions, we compute the average *heuristic utilisation rate* which indicates how frequently a given low level heuristic is chosen and applied during the search process (through 30 decision points/stages) across all runs. We also compute the average *accepted/rejected move rates* which indicates how frequently a move (solution) that is produced from the three low level heuristics (NSGAII, SPEA2 and MOGA) is accepted/ rejected under different acceptance methods (AM, GDA and LA). The results are presented in Figure 7.11.

It is clear from Figure 7.11 that all WFG problems have the same bar graph patterns for the three hyper-heuristics methods (AM, GDA and LA), as they use the same selection mechanism (choice function). Unlike the graph patterns of the choice function in the two objective case (see Section 7.4) where NSGAII has the highest average heuristic utilisation rate, SPEA2 has the highest average heuristic utilisation rate among all low level heuristics for each problem in the three objectives case. This indicates that SPEA2 performs best among other low level heuristics in all WFG problems. We theorise that multi-objective choice function based hyper-heuristics, therefore, prefers SPEA2 and it becomes preferable to be chosen more frequently than the other low level heuristics. Our result is consistent with the result in Bradstreet et al. (2007) that show the best performance is achieved by SPEA2 on the tri-objectives WFG test functions. NSGAII has the second highest average heuristic utilisation rate among all low level heuristics for each problem in all methods. As for the two objective case, the performance of MOGA is not good on the WFG problems with three objectives; thus it is invoked relatively less frequently during the search process on all test problems, for all methods

WFG	Methods	SSC (HV)				GD				IGD			
		AVG	MIN	MAX	STD	AVG	MIN	MAX	STD	AVG	MIN	MAX	STD
1	AM	107.3712	9.1829	135.3410	32.6332	4.812E-02	3.497E-02	4.910E-02	2.552E-03	**1.116E-02**	9.979E-03	1.130E-02	**2.506E-04**
	GDA	**117.8262**	32.7882	213.2110	77.3751	**4.628E-02**	4.307E-02	4.866E-02	2.029E-03	1.175E-02	1.114E-02	1.297E-02	6.065E-04
	LA	84.9291	48.9161	175.3890	33.0404	4.936E-02	6.789E-02	4.656E-02	5.061E-03	1.142E-02	1.037E-02	1.188E-02	3.313E-04
	SPEA2	114.3752	62.3193	125.9170	**23.5923**	4.871E-02	4.807E-02	4.919E-02	**2.532E-04**	1.125E-02	1.119E-02	1.133E-02	3.545E-05
2	AM	167.1861	89.2915	207.7816	45.1490	1.092E-02	5.084E-03	2.198E-02	3.936E-03	1.760E-03	1.070E-03	4.540E-03	6.627E-04
	GDA	168.3010	82.2896	208.8914	37.0173	**5.793E-03**	2.238E-03	1.366E-02	2.512E-03	**1.482E-03**	8.193E-04	2.802E-03	4.752E-04
	LA	**187.3642**	102.9195	216.7784	**35.8396**	9.523E-03	4.824E-03	1.879E-02	3.010E-03	1.822E-03	1.109E-03	2.203E-03	2.613E-04
	SPEA2	171.0259	111.9220	201.3650	42.1033	1.119E-02	9.035E-03	1.425E-02	**1.353E-03**	1.677E-03	1.389E-03	2.023E-03	**1.765E-04**
3	AM	164.4504	161.1735	165.8882	**0.8374**	1.714E-02	3.725E-03	2.461E-02	8.661E-03	6.541E-04	2.289E-04	1.103E-03	2.737E-04
	GDA	**166.2142**	164.4883	170.2537	1.4075	2.272E-02	2.115E-02	2.384E-02	**6.695E-04**	**6.007E-04**	4.123E-04	1.229E-03	**1.901E-04**
	LA	164.9405	156.9436	172.9678	3.8900	1.700E-02	6.362E-03	2.374E-02	6.760E-03	9.436E-04	4.399E-04	1.518E-03	3.575E-04
	SPEA2	155.5069	81.6381	159.0350	13.9708	**3.764E-03**	3.285E-03	4.627E-03	4.079E-04	1.237E-03	1.050E-03	1.379E-03	7.958E-05
4	AM	174.4465	172.2685	175.9692	**0.9286**	7.472E-03	6.802E-03	8.191E-03	2.959E-04	9.237E-04	8.192E-04	1.103E-03	6.690E-05
	GDA	187.5106	172.2036	195.3434	4.7233	**7.193E-03**	4.590E-03	7.808E-03	5.411E-04	**9.134E-04**	8.162E-04	1.114E-03	7.101E-05
	LA	**188.8972**	183.5044	195.8470	3.5147	7.947E-03	7.493E-03	8.451E-03	**2.531E-04**	9.767E-04	8.693E-04	1.241E-03	7.746E-05
	SPEA2	174.5163	172.3850	176.8740	1.1343	7.579E-03	7.152E-03	8.029E-03	2.640E-04	9.320E-04	8.590E-04	1.102E-03	**5.134E-05**
5	AM	169.3947	91.8980	178.5882	21.2169	2.782E-03	2.496E-03	3.375E-03	3.030E-04	7.103E-04	6.378E-04	8.505E-04	5.944E-05
	GDA	**179.0575**	171.1901	182.2547	3.0923	**2.580E-03**	2.497E-03	2.749E-03	7.342E-05	**6.807E-04**	6.455E-04	7.609E-04	2.865E-05
	LA	178.9980	172.1897	184.6083	**2.6278**	4.884E-03	3.934E-03	6.704E-03	5.983E-04	7.738E-04	7.105E-04	9.724E-04	5.322E-05
	SPEA2	168.8870	91.8184	179.1310	26.0557	2.520E-03	2.482E-03	2.578E-03	**2.189E-05**	9.252E-04	9.014E-04	9.714E-04	**1.674E-05**
6	AM	167.5519	162.3480	172.5244	**2.4692**	1.118E-02	4.877E-03	1.537E-02	2.958E-03	1.149E-03	8.090E-04	1.364E-03	1.365E-04
	GDA	178.7681	129.1464	199.9808	17.3386	**1.289E-02**	3.130E-03	1.490E-02	3.293E-03	**1.110E-03**	7.975E-04	1.795E-03	2.281E-04
	LA	**184.4600**	165.0770	201.0835	8.4301	1.490E-02	5.611E-03	1.683E-02	3.507E-03	1.306E-03	9.108E-04	1.534E-03	1.776E-04
	SPEA2	168.3973	161.8640	177.6070	3.3697	1.130E-02	4.473E-03	1.537E-02	**2.252E-03**	1.141E-03	7.331E-04	1.400E-03	**1.382E-04**
7	AM	170.9187	89.4527	175.8396	15.4664	8.132E-03	3.416E-03	1.053E-02	**1.038E-03**	1.006E-03	8.988E-04	2.583E-03	3.000E-04
	GDA	**189.2783**	149.3687	198.5367	10.1373	**6.417E-03**	2.159E-03	9.033E-03	2.197E-03	**8.992E-04**	3.045E-04	1.643E-03	2.573E-04
	LA	180.1404	171.4541	200.4325	6.2589	8.819E-03	2.870E-03	1.032E-02	1.198E-03	1.061E-03	7.941E-04	1.323E-03	8.495E-05
	SPEA2	174.4071	172.1070	176.1530	**1.0187**	8.368E-03	7.940E-03	8.808E-03	2.307E-04	9.566E-04	9.108E-04	1.006E-03	**2.503E-05**
8	AM	142.2636	142.2636	164.4628	4.2221	1.419E-02	9.033E-03	1.641E-02	2.217E-03	1.328E-03	1.129E-03	1.537E-03	1.365E-04
	GDA	174.7478	167.1404	184.4929	4.9486	**1.260E-02**	9.033E-03	1.580E-02	**1.569E-03**	**1.200E-03**	1.010E-03	1.442E-03	**1.024E-04**
	LA	**179.6973**	160.4983	186.7640	4.8295	1.345E-02	8.675E-03	1.443E-02	9.896E-04	1.312E-03	1.221E-03	1.446E-03	4.181E-05
	SPEA2	162.7549	159.5230	164.8610	**1.3546**	1.267E-02	1.196E-02	1.352E-02	4.035E-04	1.222E-03	1.164E-03	1.301E-03	3.471E-05
9	AM	163.2564	84.2574	168.3284	14.9607	6.866E-03	3.043E-03	8.897E-03	**1.191E-03**	8.819E-04	7.524E-04	1.133E-03	6.585E-05
	GDA	**177.4758**	166.6219	192.2547	4.3396	**6.107E-03**	4.049E-03	8.996E-03	8.073E-04	8.423E-04	7.670E-04	1.131E-03	8.428E-05
	LA	175.4644	193.0402	157.9340	6.4698	6.927E-03	4.039E-03	8.244E-03	9.852E-04	8.713E-04	7.756E-04	1.167E-03	7.763E-05
	SPEA2	168.0471	165.3200	170.8950	**1.2861**	6.428E-03	4.867E-03	1.286E-02	1.291E-03	**8.306E-04**	7.345E-04	9.993E-04	**5.463E-05**

Table 7.5 : The performance of multi-objective selection choice function based hyper-heuristics using different move acceptance strategies including all-moves (AM), great deluge algorithm (GDA) and late acceptance (LA) on the tri-objective WFG test problems with respect to the metrics; the hypervolume (SSC), the generational distance (GD) and the inverted generational distance (IGD).

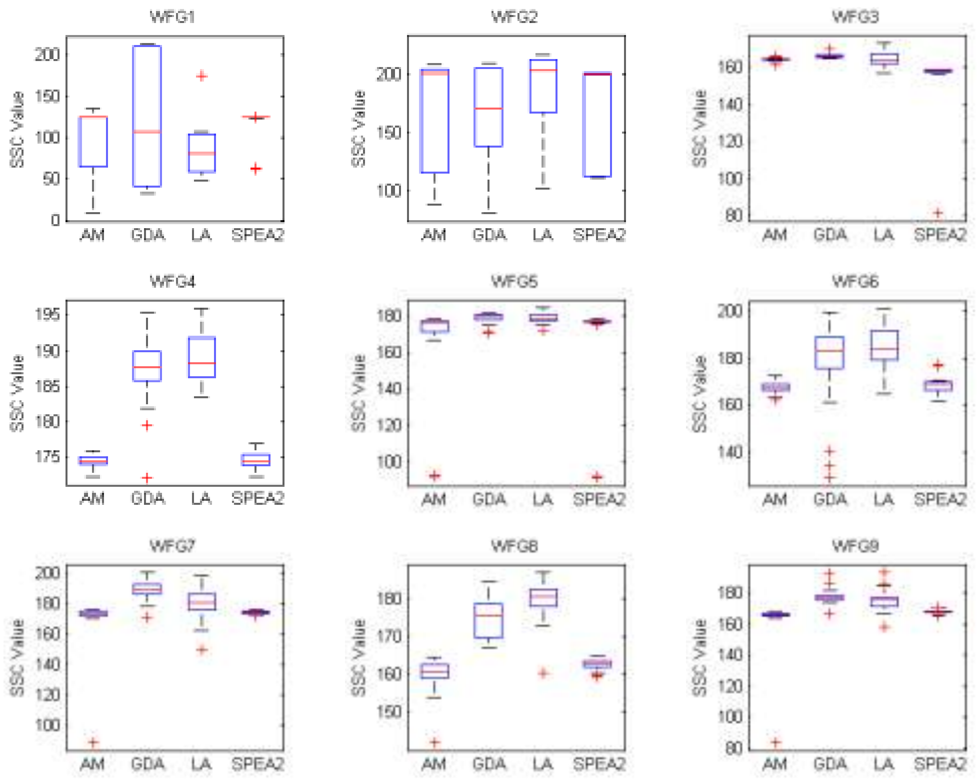


Figure 7.7: Box plots of multi-objective choice function based hyper-heuristics methodologies using AM, GDA and LA as a move acceptance criterion on the tri-objective WFG test problems for the measure of hypervoulme (SSC).

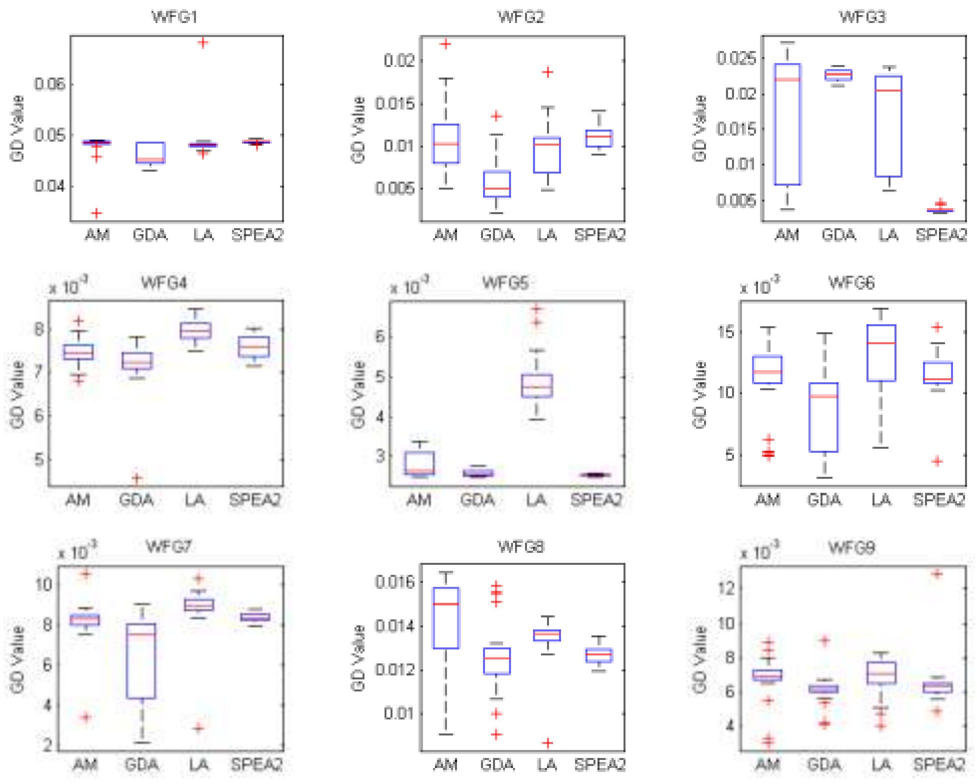


Figure 7.8: Box plots of multi-objective choice function based hyper-heuristics methodologies using AM, GDA and LA as a move acceptance criterion on the tri-objective WFG test problems for for the measure of generational distance (GD).

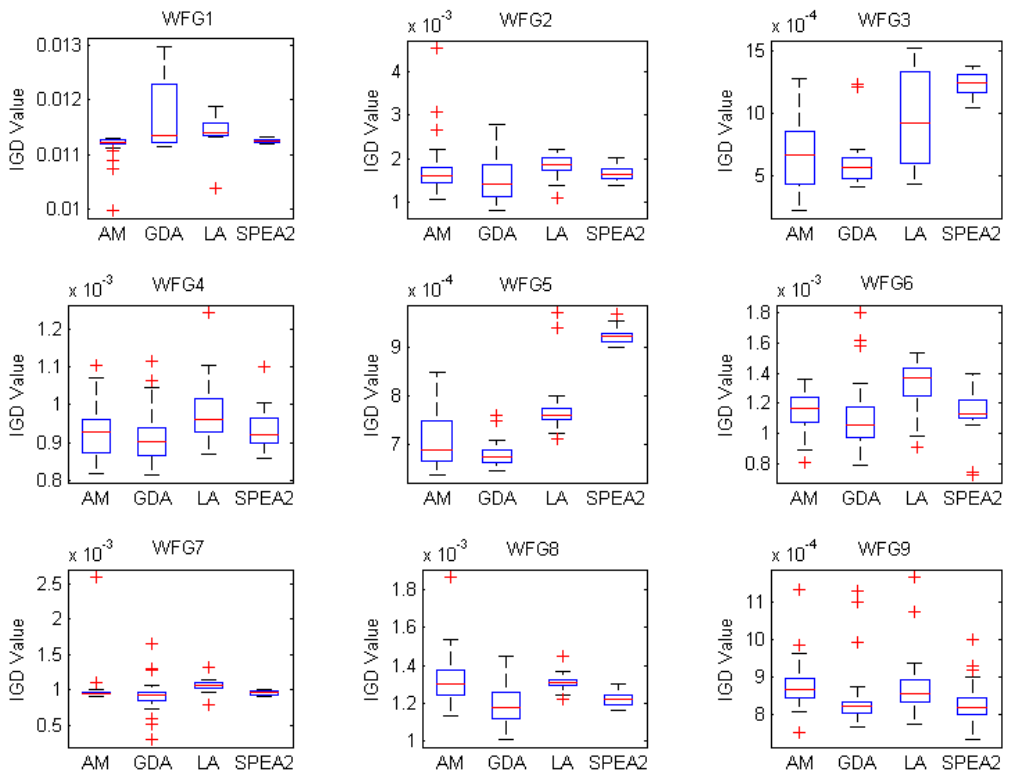


Figure 7.9: Box plots of multi-objective choice function based hyper-heuristics methodologies using AM, GDA and LA as a move acceptance criterion on the tri-objective WFG test problems for for the measure of inverted generational distance (IGD).

Figure 7.11 also gives some insights about how many moves are accepted/rejected based on the acceptance strategy that was used. We can observe that no moves are rejected for each test problem in AM, since it employs an All-Moves acceptance strategy. For each test problem in AM, SPEA2 has the highest heuristic utilisation rate among the other low level heuristics, which means that SPEA2 is invoked more frequently during the search process.

However, MOGA has a too low heuristic utilisation rate and NSGAII has a slightly higher rate than MOGA but not as high as SPEA2. This explains why AM performs relatively similar to SPEA2, in most cases, for the SSC metric (see Table 7.4). It is also clear from the graphs that the rate of rejected moves of LA is much higher than GDA on all test problems for all low level heuristics. In other words, GDA accepts moves (solutions with good quality) more than LA. These observations offer an explanation as to why the performance of GDA is better than LA in terms of convergence towards the POF. However, LA still produces better solutions than AM in most cases. This

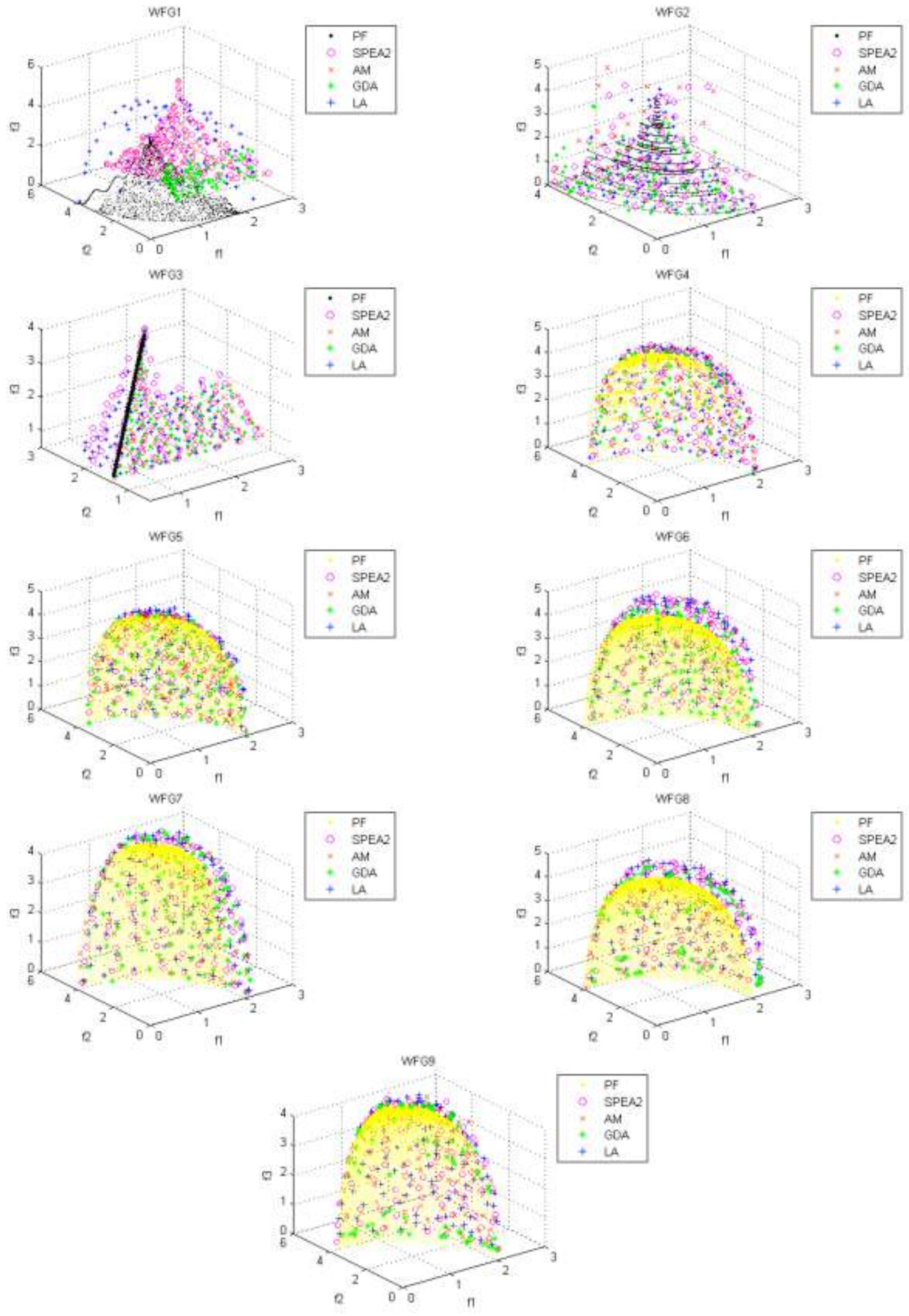


Figure 7.10: Plots of the non-dominated solutions in the objective space with the lowest GD in 30 runs of SPEA2 and three multi-objective choice function based hyper-heuristic using AM, GDA, LA as an acceptance criterion over the tri-objective WFG test functions.

.

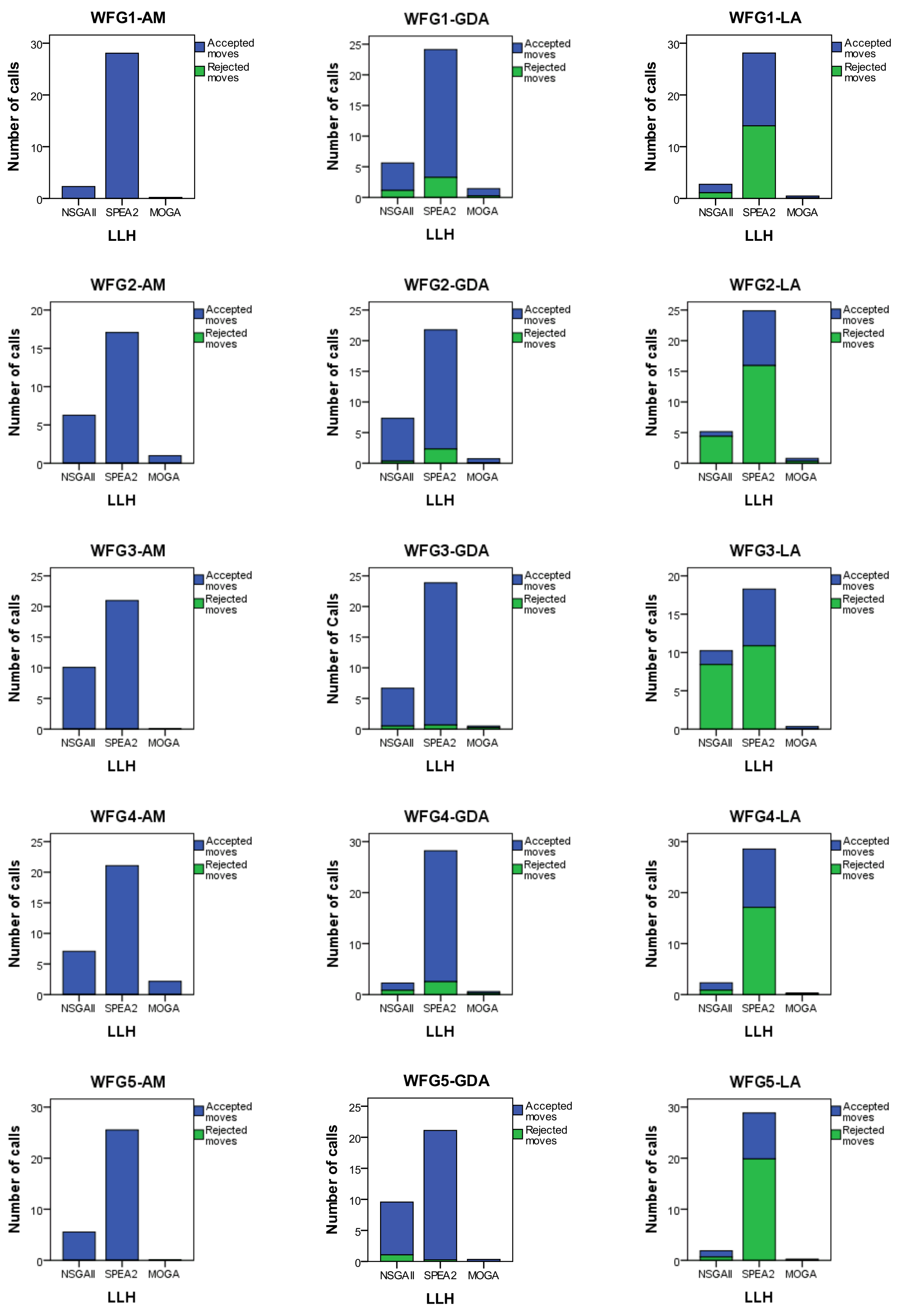


Figure 7.11: The average number of low level meta-heuristic invocations (NSGAII, SPEA2 and MOGA) and accepted/rejected moves produced by selection hyper-heuristics using AM, GDA and LA over the tri-objective WFG test problems- continue.

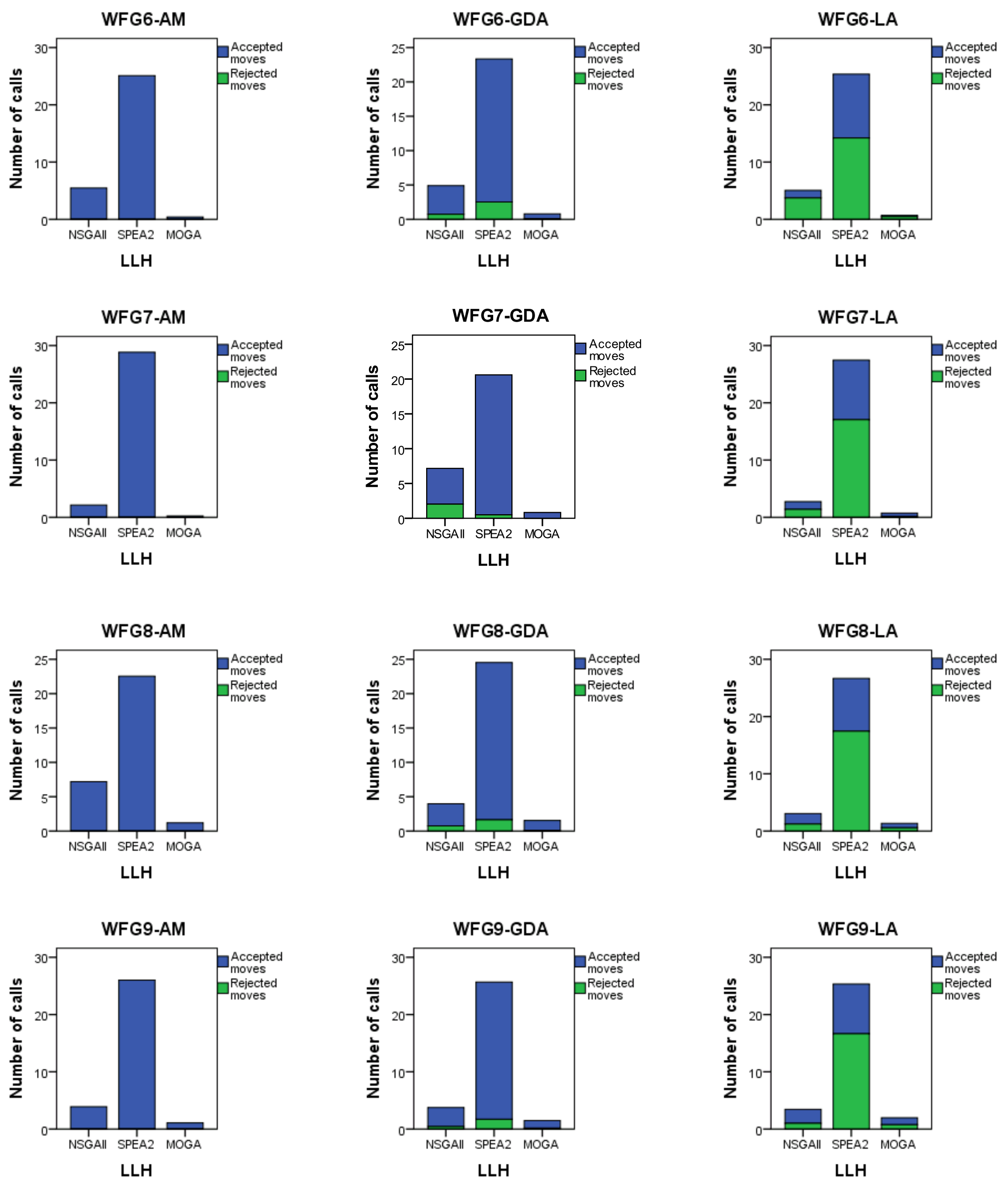


Figure 7.11: Continue- the average number of low level meta-heuristic invocations (NSGAII, SPEA2 and MOGA) and accepted/rejected moves produced by selection hyper-heuristics using AM, GDA and LA over the tri-objective WFG test problems.

indicates that is the condition criterion that used in LA help to produce solutions with acceptable quality by rejecting the worse moves (solutions) at the right decision points during the search process.

7.6 Summary and Remarks

This chapter proposed an online learning selection choice function based hyper-heuristics using late acceptance (LA) as a non-deterministic move acceptance criterion for multi-objective optimisation. To the best of our knowledge, *D* metric is used for the first time as a comparison measure between two non-dominated fronts in order to covert the multi-objective problem to a single-objective problem without definition of criteria values' weights.

The performance of the proposed multi-objective choice function late acceptance based hyper-heuristic (HHMO_CF_LA) is compared to two previous multi-objective hyper-heuristics; choice function all-moves based hyper-heuristic (HHMO_CF_AM) and choice function great deluge based hyper-heuristic (HHMO_CF_GDA) that were presented in Chapters 5 and 6 respectively. The comparison is conducted over the bi-objective Walking Fish Group (WFG) test functions benchmark for multi-objective optimisation. Additionally, the performances of the three multi-objective hyper-heuristics are compared to the well-known multi-objective algorithm, SPEA2.

The experimental results demonstrate the effectiveness of non-deterministic move acceptance strategy based methodologies. HHMO_CF_GDA and HHMO_CF_LA outperform HHMO_CF_AM over the bi-objective WFG test problems, indicating that the non-deterministic acceptance strategies improve the performance of the multi-objective selection choice function based hyper-heuristic. Moreover, this observation is supported further by empirical evidence obtained from testing those hyper-heuristics against SPEA2 over the tri-objective WFG test problems. In overall, HHMO_CF_GDA performs the best compared to other multi-objective hyper-heuristics. The superiority of multi-objective choice function great deluge based hyper-heuristic is due to the acceptance procedure that employed. The experimental result also shows that the components of the hyper-heuristics including the selection method, low level heuristics and move acceptance strategy are important and significantly affect the performance of the hyper-heuristics. The great deluge combined with choice function performs better than the great deluge combined with random selection and All-Moves combined with choice function

The benefit of using hyper-heuristics for multi-objective optimisation is shown in Table 7.6. The results of multi-objective choice function great deluge based hyper-heuristic improves solution by more than 5% when compared to the results obtained by the low level heuristics when run in isolation. This is the case except for NSGAII. The result obtained by the multi-objective choice function great deluge based hyper-heuristic is improved slightly when compared to the result obtained by NSGAII in the bi-objective

WFG problems except WFG1 and WFG2. It is good to note that the results obtained by our multi-objective choice function great deluge based hyper-heuristic improved by more than 45% in five out of nine bi-objective WFG problems, when compared to the results obtained by AMALGAM. This includes WFG2, WFG3, WFG4, WFG7 and WFG8, and more than 25% of the other test problems except WFG9. The results provide empirical evidence that combining different combination of meta-heuristics under a selection hyper-heuristic framework yields improved performance. The use of the combination of the choice function as a selection method and GDA as an acceptance strategy positively affects the performance of the multi-objective hyper-heuristics over the WFG test problems.

WFG	Methods	SSC	HH_CF_GDA improvement %
1	HH_CF_GDA	**12.9388**	
	NSGAII	11.6041	10.32
	SPEA2	6.4931	49.82
	MOGA	4.2184	67.40
	AMALGAM	7.7902	39.79
2	HH_CF_GDA	**11.8148**	
	NSGAII	10.8199	8.42
	SPEA2	10.7898	8.68
	MOGA	9.7959	17.09
	AMALGAM	1.7582	85.12
3	HH_CF_GDA	**11.9197**	
	NSGAII	11.9185	0.01
	SPEA2	11.4062	4.31
	MOGA	11.2921	5.27
	AMALGAM	6.6890	43.88
4	HH_CF_GDA	**9.6642**	
	NSGAII	9.6460	0.19
	SPEA2	9.1853	4.96
	MOGA	8.9968	6.91
	AMALGAM	3.5687	63.07
5	HH_CF_GDA	**9.2964**	
	NSGAII	9.2857	0.12
	SPEA2	9.2860	0.11
	MOGA	8.8946	4.32
	AMALGAM	6.3554	31.64
6	HH_CF_GDA	**9.3745**	
	NSGAII	9.3503	0.26
	SPEA2	8.7135	7.05
	MOGA	8.8878	5.19
	AMALGAM	6.3554	32.21
7	HH_CF_GDA	**9.6650**	
	NSGAII	9.6579	0.07
	SPEA2	9.2481	4.31
	MOGA	9.1685	5.14
	AMALGAM	3.9171	59.47
8	HH_CF_GDA	**8.7279**	
	NSGAII	8.7155	0.14
	SPEA2	8.3957	3.81
	MOGA	8.0762	7.47
	AMALGAM	3.0945	64.54
9	HH_CF_GDA	8.7689	
	NSGAII	8.7650	0.04
	SPEA2	8.7091	0.68
	MOGA	8.5723	2.24
	AMALGAM	**9.0676**	-3.41

Table 7.6: The percentage improvement for the performance of HH_CF_GDA against others methods with respect to the hypervolume (SSC) on the bi-objective WFG test functions.

8 The Real-World Problem: The Multi-objective Vehicle Crashworthiness Design

In the previous chapters we showed that our multi-objective choice function based hyper-heuristics in general, and our choice function great deluge based hyper-heuristic (HHMO_CF_GDA) in particular, can be effective when testing over both bi- and tri-objective benchmarks the Walking Fish Group (WFG) test problems. In this chapter, we further investigate the power of our multi-objective choice function based hyper-heuristics. We apply our hyper-heuristics to a real-world problem that of the multi-objective vehicle crashworthiness design. We aim to demonstrate that hyper-heuristics are not only effective on benchmarks, but that they are also applicable to a real-world problem. We also investigate the sensitivity of our choice function based hyper-heuristics, using a different size of decision points during the search. The chapter is structured as follows: In Sections 8.1 and 8.2, we describe and present the formulation of the application problem, that of the design of vehicle crashworthiness. This is followed in Section 8.3 by computational experiments and Section 8.4 presents summary and remarks.

8.1 Problem Description

In the automotive industry, crashworthiness is a very important issue to be dealt with when designing a vehicle. Crashworthiness design of real-world vehicles involves optimisation of a number of objectives including the head, injury criterion, chest acceleration and chest deflection etc (Redhe et al. 2004). However, some of these objectives may be, and usually are, in conflict with each other, i.e. an improvement in one objective value leads to deterioration in the values of the other objectives.

Multi-objective vehicle crashworthiness design was previously tackled as a single (primary) objective optimisation with multiple constraints (e.g. Redhe et al. 2004). However, it is not an easy task for most experienced design engineers to identify a primary objective from a huge number of design objectives. Alternatively, multi-objective vehicle crashworthiness design is addressed in a multi-objective framework considering different design requirements as design objectives. Fang et al. (2005) aggregated these different objectives into a single cost function in terms of weight average taken into account a weight for full-scale vehicle model, peak acceleration and energy-absorption as design objectives. In Deb (2001) an evolutionary search method has been developed to construct a multi-objective vehicle crashworthiness design based on the radial basis function. Lanzi et al. (2004) proposed a multi-objective genetic algorithm (GA) to construct multi-objective vehicle crashworthiness design by optimising composite absorber shapes under different crashworthiness requirements.

Liao et al. (2008) construct a vehicle crashworthiness design using the surrogate modelling techniques with latin hypercube sampling and stepwise regression (Krishniah, 1982). To address different safety requirements of crashworthiness design, a simulation of a full-scale vehicle model including the full frontal crash and a 40% offset-frontal crash is developed. Figure 8.1 shows the simulation results in the scenarios of the full frontal crash and the 40% offset-frontal crash. The weight of vehicle, acceleration characteristics and toe-board intrusion are addressed as the design objectives.

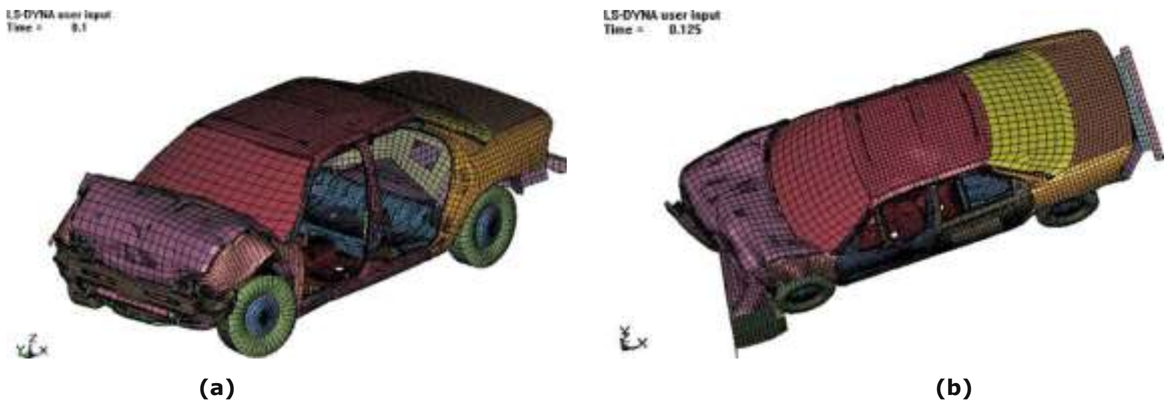


Figure 8.1: The deformed results of (a) the full frontal impact and (b) the offset-frontal impact. Reprinted from (Liao et al., 2008).

The multi-objective vehicle crashworthiness design problem has only five decision variables and no constraints (Liao et al., 2008). The output of problem provides a wider choice for engineers to make their final design decision based on Pareto solution space. In this chapter, we are tackling this problem that is presented in Liao et al. (2008) and we use it as a real-world application to our multi-objective hyper-heuristics. The decision variables of the problem represent the thickness of five reinforced members around the front as they could have a significant effect on the crash safety. See Figure 8.2 for an illustration. The mass of the vehicle is tackled as the first design objective, while an integration of collision acceleration between t_1=0.05s and t_2=0.07s in the full frontal crash is considered as the second objective function. The toe-board intrusion in the 40% offset-frontal crash is tackled as the third objective as it is the most severe mechanical injury (see Figure 8.3). The second and third objectives are constructed from the two crash conditions to reflect the extreme crashworthiness and formulated in the quadratic polynomial for the regression while the vehicle mass is formulated in a linear basis function (Marklund and Nilsson, 2001).

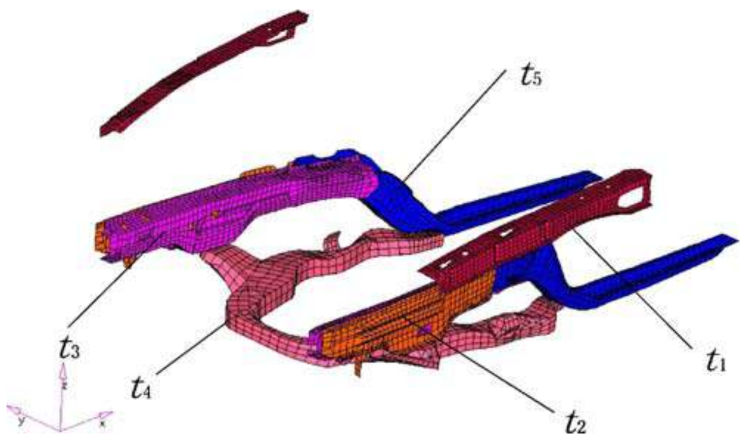


Figure 8.2: Design variables of the vehicle model. Reprinted from (Liao et al., 2008).

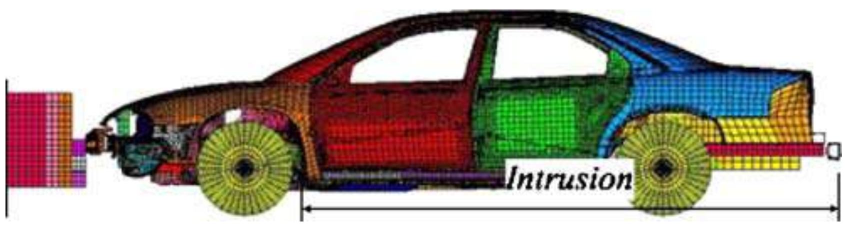


Figure 8.3: The toe board intrusion of offset-frontal crash. Reprinted from(Liao et al., 2008).

8.2 Problem Formulation

The multi-objective vehicle crashworthiness design problem involves optimisation of three objectives including the mass of the vehicle (mass), an integration of collision acceleration between t_1=0.05s and t_2=0.07s in the full frontal crash (Ain) and the toe-board intrusion in the 40% offset-frontal crash (Intrusion). The three objectives are formulated as follows:

$$Mass = 1640.2823 + 2.3573285t_1 - 2.3220035t_2 + 4.5688768t_3 + 7.7213633t_4 + 4.4559504t_5 \tag{8.1}$$

$$Ain = 6.5856 + 1.15t_1 - 1.0427t_2 + 0.9738t_3 + 0.8364t_4 - 0.3695t_1t_4 + 0.0861t_1t_5 + 0.3628t_2t_4 - 0.1106t_1^2 - 0.3437t_3^2 + 0.1764t_4^2 \tag{8.2}$$

$$Intrusion = -0.0551 + 0.0181t_1 + 0.1024t_2 + 0.0421t_3 - 0.0073t_1t_2 + 0.024t_2t_3 - 0.0118t_2t_4 - 0.0204t_3t_4 - 0.008t_3t_5 - 0.0241t_2^2 + 0.0109t_4^2 \tag{8.3}$$

So, the multi-objective design of vehicle crashworthiness problem in T decision variable space is formulated as:

$$min\ F(x) = [Mass, Ain, Intrusion]$$

$$s.t.\ \ 1mm \leq x \leq 3mm \tag{8.4}$$

$$where\ x = (t_1, t_2, t_3, t_4, t_5)^T$$

We created three more problem instances beside the original vehicle crashworthiness problem as shown in Table 8.1 after a private communication with Prof. Kalyanmoy Deb who recommended this problem. Each instance contains a pair of objectives. NSGAII was applied to the original vehicle crashworthiness problem in Liao et al. (2008) and produced reasonable results for the three objective version.

Problem Name	Objective Functions
Car1	Mass and Ain
Car2	Mass and Intrusion
Car3	Ain and Intrusion
Car4	Mass and Ain and Intrusion

Table 8.1: The multi-objective vehicle crashworthiness design problems.

8.3 Experiments and Comparison

In this section, a set of experiments are conducted over a multi-objective vehicle crashworthiness design problem as a real-world problem to evaluate the performance of our multi-objective choice function based hyper-heuristics; HHMO_CF_AM, HHMO_CF_GDA and HHMO_CF_LA. The motivation behind applying our three selection multi-objective hyper-heuristics to this problem is to investigate their performance on a real-world problem and measure the level of generality that they can achieve. The performance of three multi-objective hyper-heuristics compared to the well-known multi-objective evolutionary algorithm, NSGAII (Deb and Goel, 2001).

8.3.1 Performance Evaluation Criteria

The same performance evaluation criteria and algorithms are used as described in Section 7.4.1. Five performance metrics are used to measure the quality of the approximation sets from different aspects: (i) ratio of non-dominated individuals (RNI) (Tan et al., 2002), (ii) hypervolume (SSC) (Zitzler and Thiele, 1999) (iii) uniform distribution of a non-dominated population (UD) (Srinivas and Deb, 1994), (iv) generational distance (GD) (Van Veldhuizen and Lamont, 1998b) and (v) inverted generational distance (IGD) (Coello and Cruz Cortès, 2005). In addition, *t*-test is used as a statistical test for the average performance comparison of selection hyper-heuristics and the results are discussed using the same notation as provided in Section 7.4.1.

8.3.2 Experimental Settings

We performed 30 independent runs for each comparison method using the same parameter settings as provided in Liao et al. (2008) with a population size equal to 30. In order to make a fair comparison, we repeated NSGAII experiments conducted in Liao et al. (2008) under our termination conditions over the additional instances. All multi-objective hyper-heuristics methodologies run for a total of 75 iterations (stages) based on the empirical experiments that are presented in next subsection. In each iteration, a low level heuristic is selected and applied to execute 50 generations. So, all methods terminated after 3,750 generations. The distance sharing σ for the

UD metric and MOGA was arbitrarily set to 0.09 in the normalised space. These settings are used for the UD as a feedback indicator in the ranking scheme of the hyper-heuristic framework and as a performance measure for the comparison. As the true Pareto front is unknown, we consider the best approximation found by combining results of all considered methods and used it instead of the true Pareto front for the metrics of GD and IGD. In the measure of SSC and D metric for GDA and LA, the reference points in our experiments for k objectives can be set as $r_i = z^{nadir_i} + 0.5\,(z^{nadir_i} - z^{ideal_i})$, $i = 1,...,k$ (Li and Landa-Silva, 2011). Other experimental settings are the same as those used in Section 7.4.2. All algorithms were implemented with the same common sub-functions using Microsoft Visual C++ 2008 on an Intel Core2 Duo 3GHz\2G\250G computer.

8.3.3 Tuning of Number of Decision Points for Multi-objectives Hyper-heuristics

In the context of our multi-objective selection hyper-heuristics, the number of the decision points (NDP) is the number of moves that we conduct during the search. The NDP is an important parameter in our multi-objective hyper-heuristic framework. However, the choice of the right value of the decision points is not trivial. We conducted initial experiments to determine the right (or at least good) value of NDP that leads to solutions of good quality. The NDP relies on the other parameters such as the number of function evaluations and the number of generations. In these experiments, each decision point is executed a fixed number of generation equals to 50 with a population size equal to 30. In other words, 1500 evaluation functions are executed at each decision point (iteration or stage). For three multi-objective hyper-heuristics; HHMO_CF_AM, HHMO_CF_GDA and HHMO_CF_LA, we used four different values for NDP (25, 50, 75 and 100). The three hyper-heuristics were run for 30 times using these values with a different random seed on the original vehicle crash worthiness problem (Car4). From this point onward, each hyper-heuristic will be referred to by move acceptance method utilised within each hyper-heuristic.

The performance of the comparison methods AM, GDA and LA for the different sizes of the decision points (25, 50, 75 and 100) with respect to the performance metrics (RNI, SSC and UD) on the original vehicle crashworthiness problem (Car4) are summarised in Table 8.2.

In Table 8.2, the average, the minimum, the maximum and standard deviation values for each performance metric are computed. A higher value indicates a better performance. We can observe that the highest averages of RNI for AM are obtained with 25 and 75 decision points. The highest average of SSC and UD values are obtained with 75, 100 decision points respectively. So, no specific size of decision points for AM can obtain good results with respect to all metrics except in 75 decision points where AM obtains good results in terms of the convergence and the number of the non-dominated solutions. GDA obtains the highest averages of RNI with 25, 50 and 75 decision points. It obtains the highest averages of SSC and UD with 75 decision points. GDA obtains good results with respect to the three performance metrics with 75 decision points. LA obtains the highest averages of RNI with 25, 50 and 75 decision points. It obtains the highest average of SSC with 75 decision points, while it obtains the highest average of UD with 100 decision points. No specific size of decision points for AM can obtain good results with respect to all metrics except for 75 decision points where LA obtains good results in terms of the convergence and the number of the non-dominated solutions. We note that 75 decision points produces better solutions in most cases for the three multi-objective choice function based hyper-heuristics.

To analyse these results, we visualise the average performance values of RNI, SSC and UD metrics for the three multi-objective hyper-heuristics AM, GDA and LA during the search using different values of the decision points (NDP) (25, 50, 75 and 100) are shown in Figures 8.4-8.6. In Figure 8.4, the performance of three methods with respect to RNI using a different values of decision points are relatively the same, the smaller value of decision points obtains a higher (better) value of RNI while increasing the value of decision points leads to a lower (worse) value of RNI. This is clear in the case of 100 decision points. As our multi-objective hyper-heuristics do not incorporate any archive mechanisms to maintain the non-dominated solutions during the search, a large number of iterations (decision points) may exhibit the factor of diversification in the selection method that calls a heuristic which produces low quality solutions.

In Figure 8.5, AM and GDA and LA perform similar to each other during the search using different values of the decision points with respect to the metric of SSC. The three methods obtain a higher (better) value of SSC when

Methods	NDP	RNI				SSC				UD			
		AVG	MIN	MAX	STDDEV	AVG	MIN	MAX	STDDEV	AVG	MIN	MAX	STDDEV
AM	25	**1.00**	1.00	1.00	**0.00**	6.045E+07	3.625E+07	8.586E+07	1.669E+07	0.623	0.480	0.698	**0.044**
	50	0.93	0.75	1.00	0.06	6.631E+07	2.089E+07	8.644E+07	1.979E+07	0.480	0.200	0.640	0.140
	75	**1.00**	1.00	1.00	**0.00**	**7.381E+07**	5.315E+07	9.577E+07	**1.463E+07**	0.585	0.516	0.707	0.050
	100	0.73	0.38	0.88	0.10	6.767E+07	2.965E+07	8.660E+07	1.998E+07	**0.642**	0.491	0.732	0.047
GDA	25	**1.00**	1.00	1.00	**0.00**	7.875E+07	4.853E+07	9.587E+07	1.274E+07	0.605	0.541	0.691	**0.032**
	50	**1.00**	1.00	1.00	0.08	8.109E+07	6.294E+07	9.091E+07	**1.007E+07**	0.579	0.510	0.670	0.040
	75	**1.00**	1.00	1.00	**0.00**	**8.289E+07**	6.294E+07	9.577E+07	1.954E+07	**0.613**	0.555	0.692	0.034
	100	0.94	0.75	1.00	0.09	8.236E+07	5.910E+07	9.587E+07	1.138E+07	0.595	0.505	0.667	0.039
LA	25	**1.00**	1.00	1.00	**0.00**	7.301E+07	5.959E+07	8.800E+07	**1.167E+07**	0.584	0.494	0.694	0.056
	50	**1.00**	1.00	1.00	**0.00**	7.526E+07	5.776E+07	9.549E+07	1.379E+07	0.580	0.490	0.660	0.040
	75	**1.00**	1.00	1.00	**0.00**	**7.538E+07**	4.512E+07	9.550E+07	1.474E+07	0.582	0.302	0.641	0.062
	100	0.98	0.95	1.00	0.01	6.972E+07	4.912E+07	8.800E+07	1.207E+07	**0.600**	0.530	0.650	**0.030**

Table 8.2: The performance of multi-objective selection hyper-heuristics with different values of decision points (NDP) on the multi-objective design of vehicle crashworthiness problem (Car4) with respect to the metrics of ratio of non-dominated individuals (RNI), size of space covered (SSC), and uniform distribution (UD) of non-dominated population.

the number of decision points is higher except in the case of 100 decision points where the search is frozen and no further improvement is obtained.

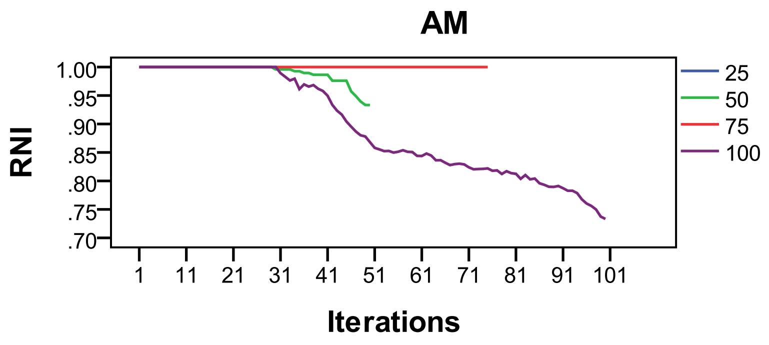
This is true for the performance of the UD metric of AM and GDA and LA in Figure 8.6. For all methods, a higher (better) value of UD is obtained when the number of decision points is higher. AM and LA obtains the best solutions with 100 decision points. While GDA obtains the best solutions with 75 decision points as the search is frozen and no further better improvement is obtained with 100 decision points.

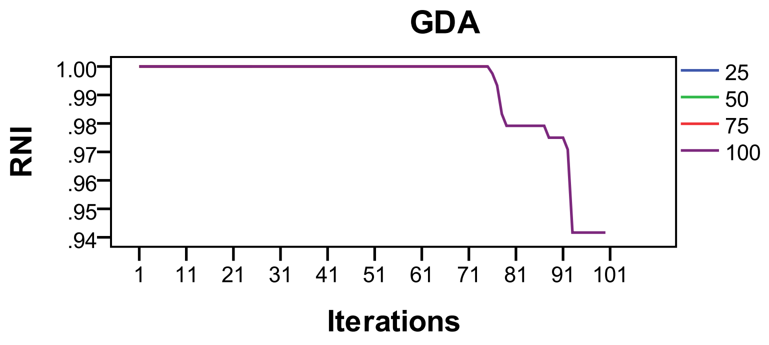
From the above observations, we conclude that a larger number (value) of decision points produces better solutions, particularly 75 decision points, according to performance metrics (RNI, SSC and UD) in the majority cases for the three methods AM, GDA and LA over the original multi-objective vehicle crashworthiness design problem (Car4). Therefore, all multi-objective hyper-heuristics methodologies run for a total of 75 decision points (iterations/stages) in our experiments over the additional instances of multi-objective vehicle crashworthiness design problems.

8.3.4 Performance Comparison of Multi-objective Hyper-heuristics and NSGAII

The mean performance comparison of AM, GDA, LA and NSGAII based on the performance metrics (RNI, SSC, UD , GD and IGD) for solving the vehicle crashworthiness problems is provided in Table 8.3. For each performance metric, the average, minimum, maximum and standard deviation values are computed. For all metrics, a higher value indicates a better performance, except in GD and IGD, where a lower value indicates a better performance. The statistical *t*-test results of NSGAII and three multi-objective choice function based hyper-heuristics (AM, GDA and LA) are given in Table 8.4. We also visualise the distribution of the simulation data of the 30 independent runs for the comparison methods with respect to these performance metrics as box plots, shown in Figures 8.7-8.11.

From Tables 8.3-8.5 and Figures 8.7-8.11, we can observe that GDA, LA and NSGAII produce a slightly higher average ratio of non-dominated individuals (RNI) compared to AM for all problems. This means that the comparison methods produce non-dominated solutions that are equal to the





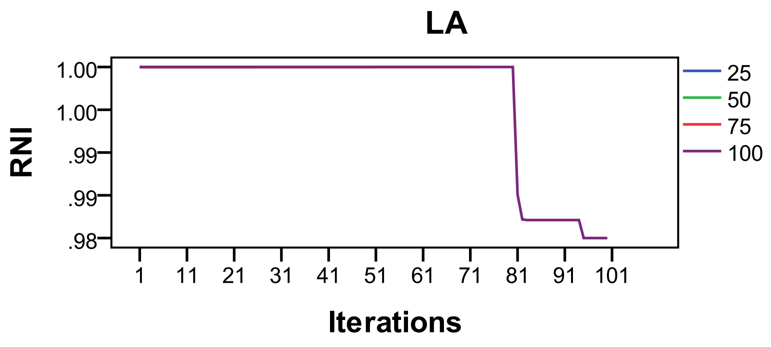
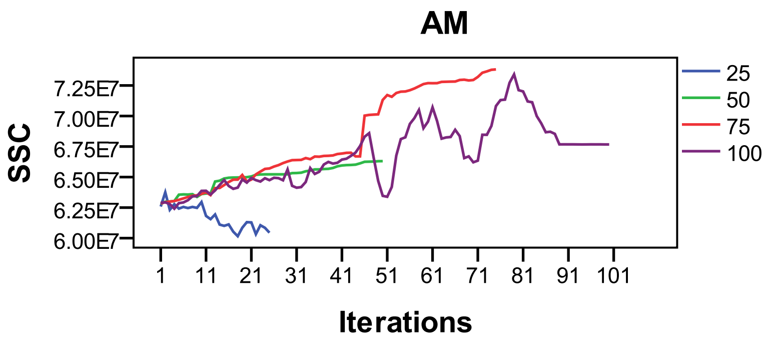
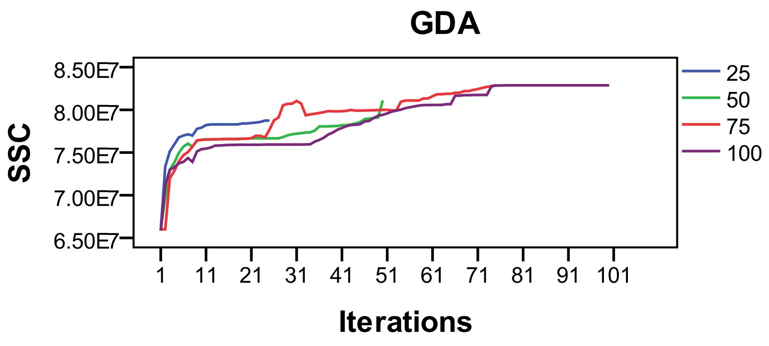


Figure 8.4: The plots showing how RNI values, averaged over 30 trials change at each decision point (iteration) for a given move acceptance method (AM, GDA and LA) combined with choice function heuristic selection considering different number of decision points while solving the vehicle crashworthiness problem (Car4).





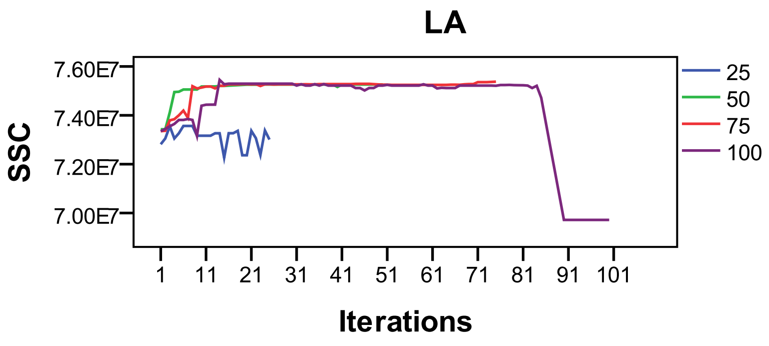


Figure 8.5: The plots showing how SSC values, averaged over 30 trials change at each decision point (iteration) for a given move acceptance method (AM, GDA and LA) combined with choice function heuristic selection considering different number of decision points while solving the vehicle crashworthiness problem (Car4).

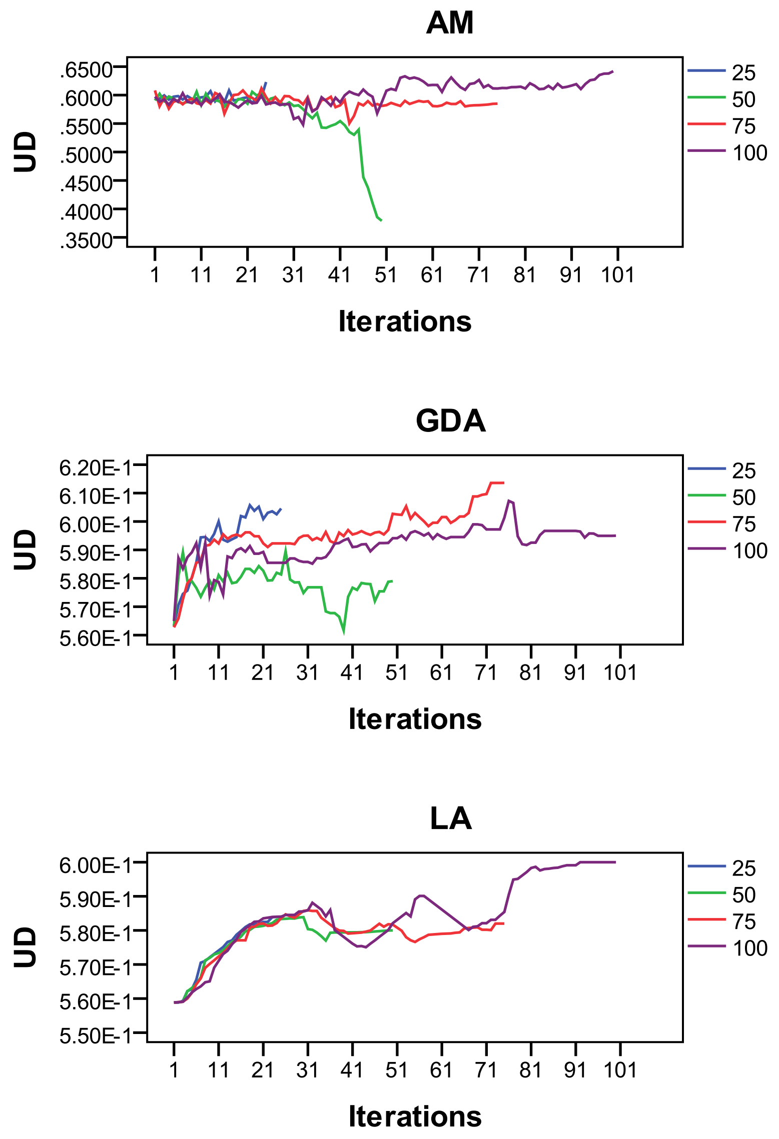


Figure 8.6: The plots showing how UD values, averaged over 30 trials change at each decision point (iteration) for a given move acceptance method (AM, GDA and LA) combined with choice function heuristic selection considering different number of decision points while solving the vehicle crashworthiness problem (Car4).

given population size and perform very well with respect to this metric. AM performs well with respect to RNI on Car4, but not for other problem instances. With respect to the hypervolume (SSC), GDA has the highest average value among the other methods for all problem instances. The performance difference of GDA from the other hyper-heuristics is statistically significant for Car1, Car3 and Car4. With respect to the measures of GD and IGD, GDA is superior to the other methods for all problem instances, except Car3, where NSGAII performs the best. This performance difference is statistically significant for Car1 and Car2. GDA performs the best considering convergence and diversity, producing solutions that converge towards the POF. Similarly, considering UD, GDA produces solutions that are distributed uniformly along the POF for all problem instances, except Car2, where NSGAII performs the best. The above observations indicate that all methods perform similarly to each other with respect to the metric of RNI over all problem instances. GDA obtains the best performance in the metrics of SSC, GD and IGD and it converges better towards the POF than the other methods. GDA is also obtains the best performance in the metric of UD and distribute more uniformly than other methods in the most problem instances.

For each problem instance, the 50% attainment surface for each method, from the 30 fronts after 3,750 generations are computed and illustrated in Figures 8.12-8.15. GDA appears to generate a good convergence for all problem instances. This can be clearly observed for Car2 and Car3 (See Figures 8.13 and 8.14), where GDA converges to the best POF with a well spread Pareto front as compared to the other approaches. In contrast, AM generates the poorest solutions in almost all cases. NSGAII and LA have similar convergence for all problem instances, except Car2, where NSGAII covered a larger proportion of objective space compared to LA.

From the above observations, we conclude that GDA outperforms NSGAII and others methods in the majority of cases. The superiority of GDA could be because of the acceptance condition criterion that was used. The hyper-heuristics for even real world multi-objective problems benefits from the use of a learning heuristic selection method as well as GDA.

problem	Method	RNI				SSC				UD			
		AVG	MIN	MAX	STD	AVG	MIN	MAX	STD	AVG	MIN	MAX	STD
Car1	NSGAII	**1.00**	1.00	1.00	**0.00**	2.296E+04	2.296E+04	2.299E+04	**1.400E-01**	0.450	0.421	0.492	0.021
	AM	0.98	0.78	1.00	0.05	2.113E+04	5.741E+03	2.255E+04	5.054E+03	0.430	0.203	0.484	0.067
	GDA	**1.00**	1.00	1.00	**0.00**	**2.298E+04**	7.703E+03	2.302E+04	3.880E+03	**0.453**	0.410	0.487	**0.020**
	LA	0.99	1.00	1.00	**0.00**	2.165E+04	7.701E+03	2.319E+04	3.983E+03	0.452	0.392	0.490	0.031
Car2	NSGAII	**1.00**	1.00	1.00	**0.00**	3.930E+04	2.109E+04	5.677E+04	1.632E+04	**0.461**	0.413	0.500	0.032
	AM	0.95	0.75	1.00	0.09	3.773E+04	6.799E+03	5.667E+04	1.707E+04	0.427	0.170	0.534	0.095
	GDA	**1.00**	1.00	1.00	**0.00**	**3.953E+04**	2.109E+04	5.680E+04	1.685E+04	0.451	0.413	0.502	**0.020**
	LA	**1.00**	1.00	1.00	**0.00**	2.107E+03	2.089E+04	5.669E+04	**1.508E+04**	0.450	0.402	0.501	0.021
Car3	NSGAII	**1.00**	1.00	1.00	**0.00**	4.174E+01	2.637E+01	4.906E+01	8.820E+00	0.464	0.411	0.510	0.022
	AM	0.98	0.63	1.00	0.08	4.058E+01	1.898E+01	4.907E+01	**1.020E+01**	0.478	0.425	0.543	0.031
	GDA	**1.00**	1.00	1.00	**0.00**	**4.175E+01**	1.930E+01	4.979E+01	9.980E+00	**0.480**	0.445	0.527	**0.021**
	LA	**1.00**	1.00	1.00	**0.00**	4.149E+01	1.977E+01	4.978E+01	9.680E+00	0.463	0.391	0.503	0.033
Car4	NSGAII	**1.00**	1.00	1.00	**0.00**	7.936E+07	4.168E+07	9.587E+07	1.595E+07	0.592	0.532	0.670	0.045
	AM	**1.00**	1.00	1.00	**0.00**	7.381E+07	5.315E+07	9.577E+07	**1.463E+07**	0.585	0.516	0.707	0.050
	GDA	**1.00**	1.00	1.00	**0.00**	**8.289E+07**	6.294E+07	9.580E+07	1.954E+07	**0.613**	0.555	0.692	**0.034**
	LA	**1.00**	1.00	1.00	**0.00**	7.538E+07	4.512E+07	9.550E+07	1.474E+07	0.582	0.302	0.641	0.062

Table 8.3: The performance NSGAII and the selection choice function based hyper-heuristics using different move acceptance strategies including all-moves (AM), great deluge algorithm (GDA) and late acceptance (LA) on the vehicle crashworthiness problems with respect to the metrics; the ratio of non-dominated individuals (RNI), the hypervolume (SSC)and the uniform distribution (UD).

problem	**Method**	**GD**				**IGD**			
		AVG	**MIN**	**MAX**	**STD**	**AVG**	**MIN**	**MAX**	**STD**
Car1	NSGAII	8.10E-04	1.10E-04	1.79E-03	4.00E-04	4.657E-04	4.117E-04	5.260E-04	**3.114E-05**
	AM	7.50E-04	1.00E-05	2.37E-03	4.70E-04	5.874E-03	3.994E-04	1.462E-02	5.990E-03
	GDA	**4.50E-04**	0.00E+00	8.70E-04	**2.00E-04**	**4.278E-04**	3.722E-04	5.817E-04	5.763E-05
	LA	8.40E-04	6.00E-05	2.72E-03	6.00E-04	6.912E-04	3.749E-04	7.866E-03	1.356E-03
Car2	NSGAII	2.45E-03	4.10E-04	9.21E-03	3.28E-03	3.174E-03	6.551E-04	6.647E-03	2.890E-03
	AM	2.30E-03	3.50E-04	1.04E-02	3.12E-03	4.974E-03	7.021E-04	1.120E-02	3.527E-03
	GDA	**1.86E-03**	3.60E-04	8.94E-03	**2.12E-03**	**3.127E-03**	6.607E-04	1.624E-02	3.630E-03
	LA	2.50E-03	3.30E-04	8.97E-03	3.34E-03	4.184E-03	6.758E-04	6.724E-03	**2.884E-03**
Car3	NSGAII	**1.01E-01**	9.68E-02	1.08E-01	4.02E-03	**9.925E-02**	6.080E-02	2.094E-01	**5.065E-02**
	AM	1.03E-01	9.79E-02	1.13E-01	**3.83E-03**	1.648E-01	6.066E-02	2.130E-01	6.292E-02
	GDA	1.03E-01	9.65E-02	1.32E-01	7.53E-03	1.264E-01	6.016E-02	2.094E-01	6.472E-02
	LA	1.03E-01	9.64E-02	1.13E-01	4.66E-03	1.420E-01	6.235E-02	2.100E-01	5.744E-02
Car4	NSGAII	2.48E-03	1.46E-03	4.21E-03	9.10E-04	4.156E-03	1.543E-03	1.289E-02	3.859E-03
	AM	2.71E-03	1.59E-03	4.06E-03	7.90E-04	4.376E-03	1.738E-03	1.288E-02	4.168E-03
	GDA	**2.11E-03**	1.10E-03	4.28E-03	**7.10E-04**	**3.552E-03**	1.661E-03	1.230E-02	3.075E-03
	LA	3.32E-03	1.70E-03	6.76E-03	1.33E-03	3.604E-03	1.525E-03	1.238E-02	**2.582E-03**

Table 8.4: The performance NSGAII and the selection choice function based hyper-heuristics using different move acceptance strategies including all-moves (AM), great deluge algorithm (GDA) and late acceptance (LA) on the vehicle crashworthiness problems with respect to the metrics; the generational distance (GD) and the inverted generational distance (IGD).

Problem	Methods	Metrics				
		RNI	SSC	UD	GD	IGD
Car1	NSGAII:AM	±	+	+	∓	+
	NSGAII:GDA	n\a	−	−	−	∓
	NSGAII:LA	±	+	−	±	±
	AM:GDA	∓	−	+	−	−
	AM:LA	∓	∓	−	−	−
	GDA:LA	±	+	±	+	+
Car2	NSGAII:AM	±	+	+	∓	±
	NSGAII:GDA	n/a	∓	±	−	∓
	NSGAII:LA	n/a	+	+	∓	+
	AM:GDA	∓	−	−	−	−
	AM:LA	∓	+	−	±	∓
	GDA:LA	n/a	+	±	+	+
Car3	NSGAII:AM	±	+	+	±	+
	NSGAII:GDA	n/a	∓	−	±	+
	NSGAII:LA	n/a	±	±	±	+
	AM:GDA	∓	−	∓	∓	−
	AM:LA	∓	−	+	±	−
	GDA:LA	n/a	±	+	±	+
Car4	NSGAII:AM	n/a	+	±	±	±
	NSGAII:GDA	n/a	−	+	∓	−
	NSGAII:LA	n/a	+	+	+	−
	AM:GDA	n/a	−	−	∓	−
	AM:LA	n/a	−	±	+	−
	GDA:LA	n/a	+	+	+	±

Table 8. 4: The *t*-test results of NSGAII and the three multi-objective choice function based hyper-heuristics methodologies using AM, GDA and LA as a move acceptance criterion on the multi-objective vehicle crashworthiness design problems with respect to the metrics; the ratio of non-dominated individuals (RNI), the hypervolume (SSC), the uniform distribution (UD), the generational distance (GD) and the inverted generational distance (IGD).

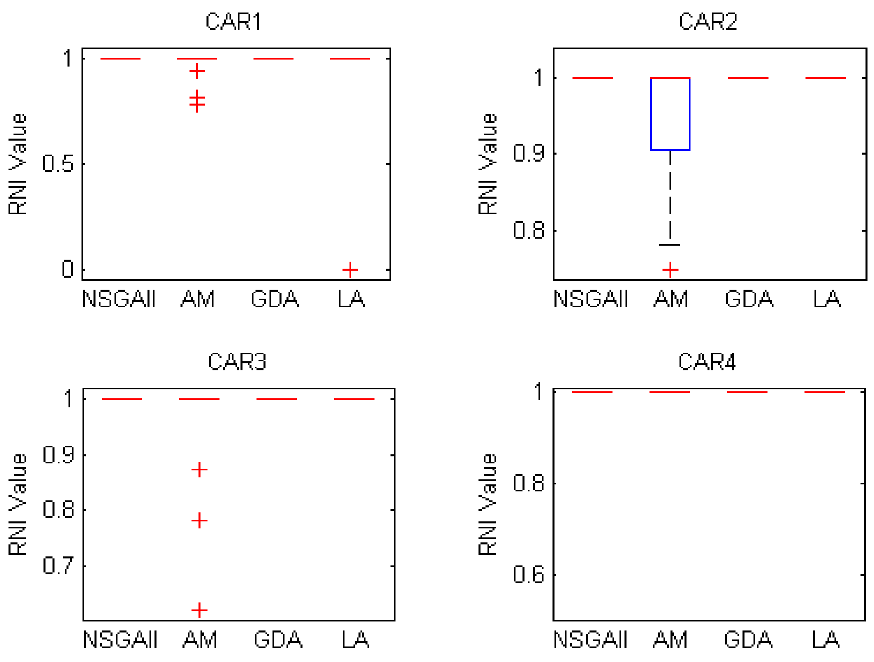


Figure 8.7: Box plots of multi-objective choice function based hyper-heuristics methodologies using AM, GDA and LA as a move acceptance criterion on the multi-objective vehicle crashworthiness design problems for the measure of ratio non-dominated solutions (RNI).

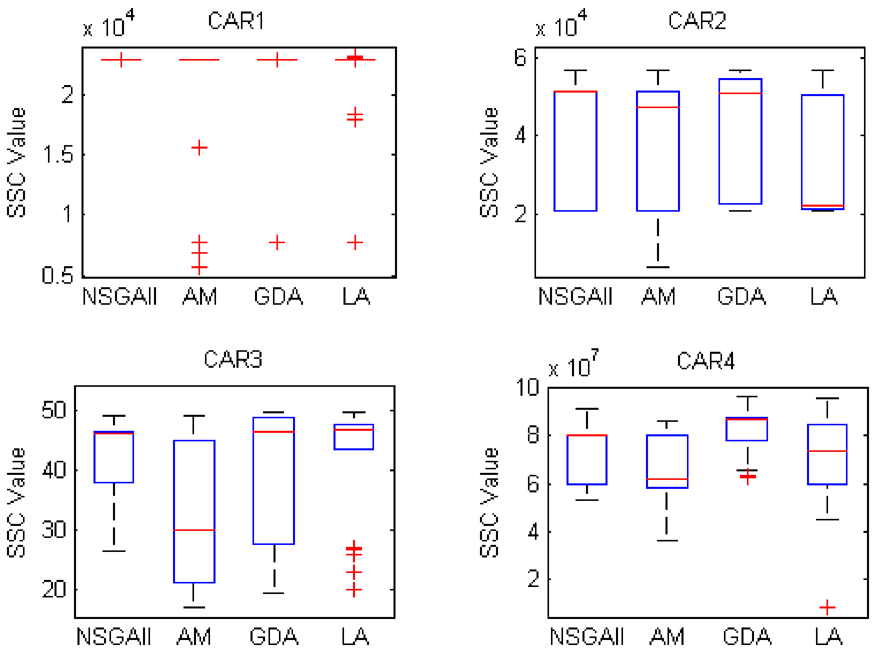


Figure 8.8: Box plots of multi-objective choice function based hyper-heuristics methodologies using AM, GDA and LA as a move acceptance criterion on the multi-objective vehicle crashworthiness design problems for the measure the hypervolume (SSC).

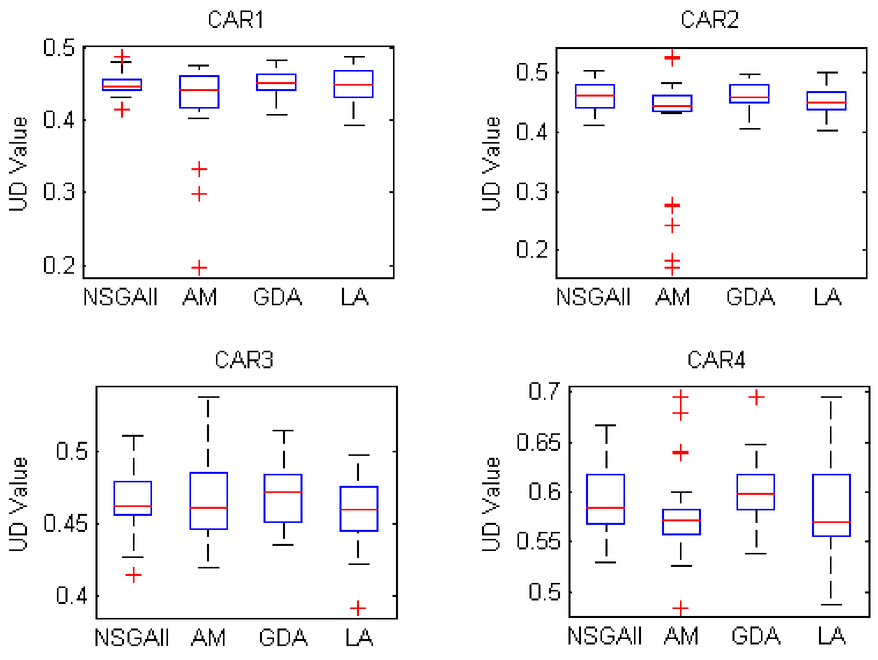


Figure 8.9: Box plots of multi-objective choice function based hyper-heuristics methodologies using AM, GDA and LA as a move acceptance criterion on the multi-objective vehicle crashworthiness design problems for the measure the uniform distribution (UD).

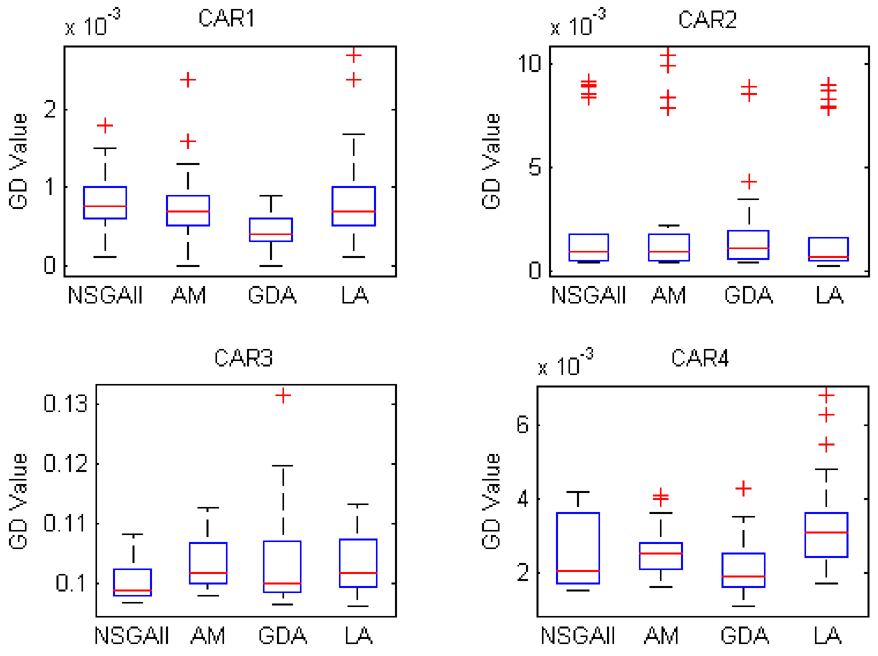


Figure 8.10: Box plots of multi-objective choice function based hyper-heuristics methodologies using AM, GDA and LA as a move acceptance criterion on the multi-objective vehicle crashworthiness design problems for the generational distance (GD).

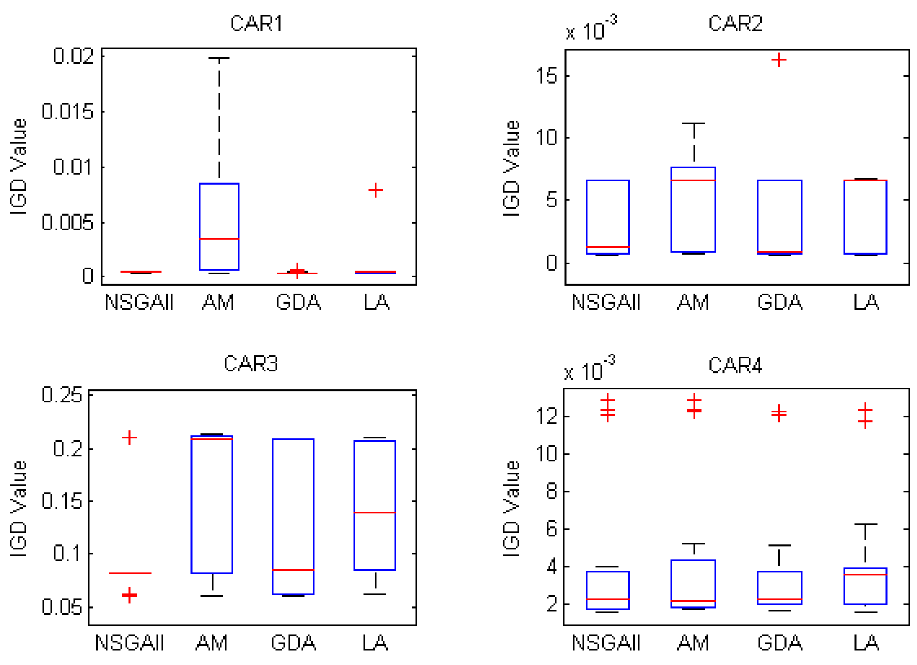


Figure 8.11: Box plots of multi-objective choice function based hyper-heuristics methodologies using AM, GDA and LA as a move acceptance criterion on the multi-objective vehicle crashworthiness design problems for the inverted generational distance (IGD).

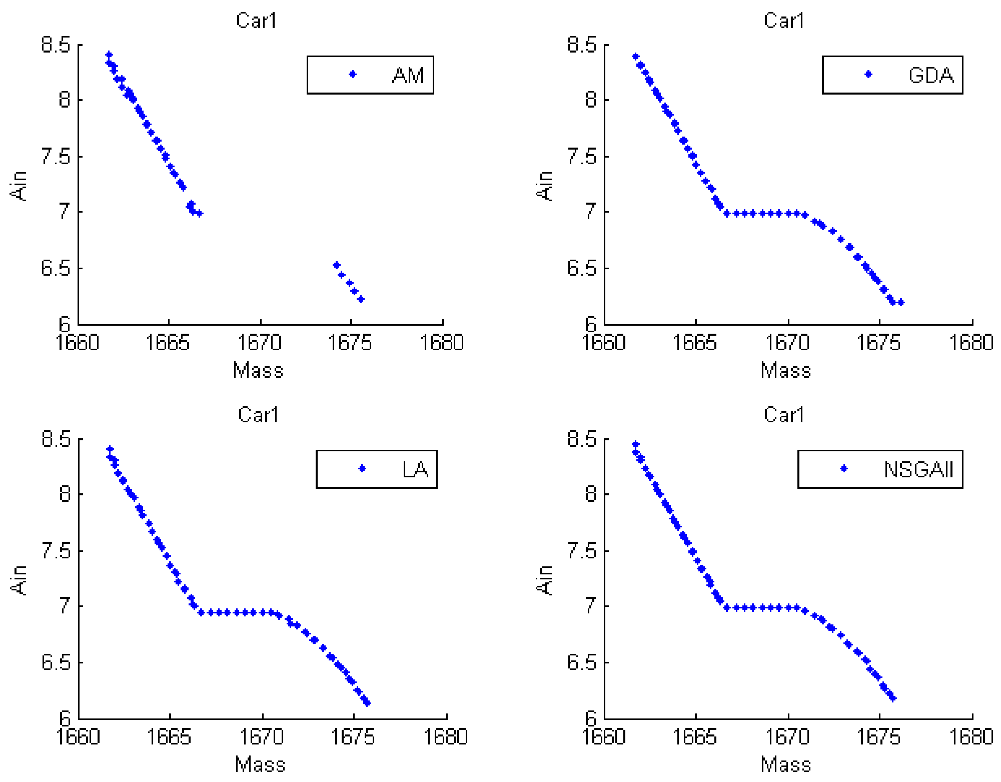


Figure 8.12: The 50% attainment surfaces for NSGAII and the three multi-objective choice function based hyper-heuristics (AM, GDA and LA) after 3,750 generations on the multi-objective design of vehicle crashworthiness problem (Car1).

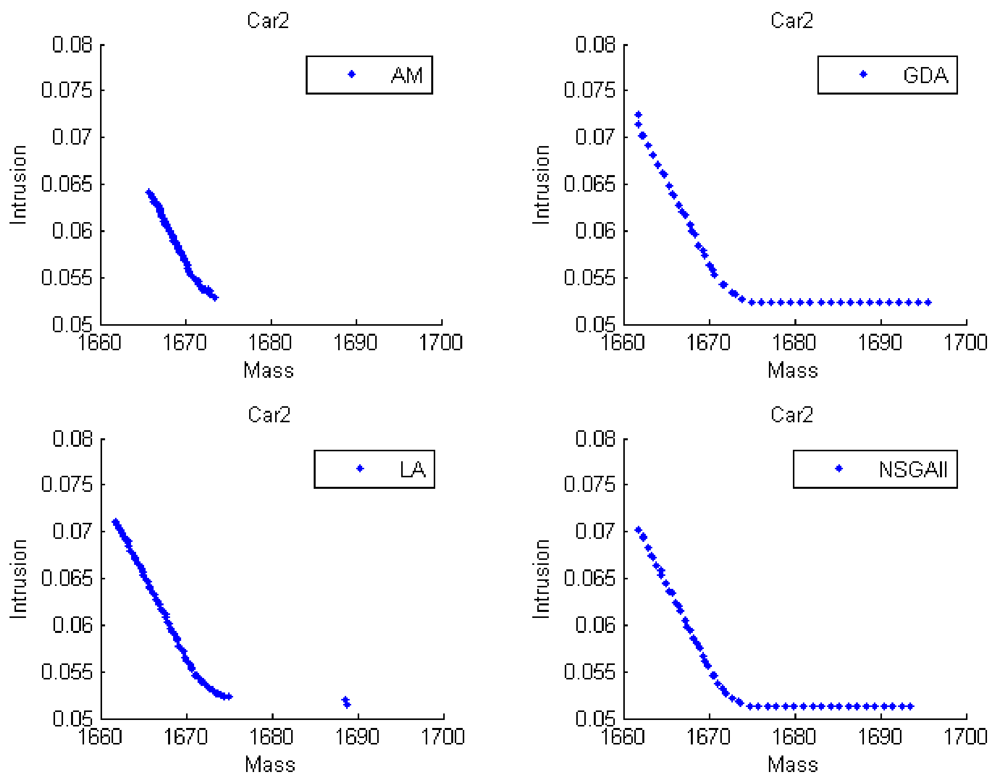


Figure 8.13: The 50% attainment surfaces for NSGAII and the three multi-objective choice function based hyper-heuristics (AM, GDA and LA) after 3,750 generations on the multi-objective design of vehicle crashworthiness problem (Car2).

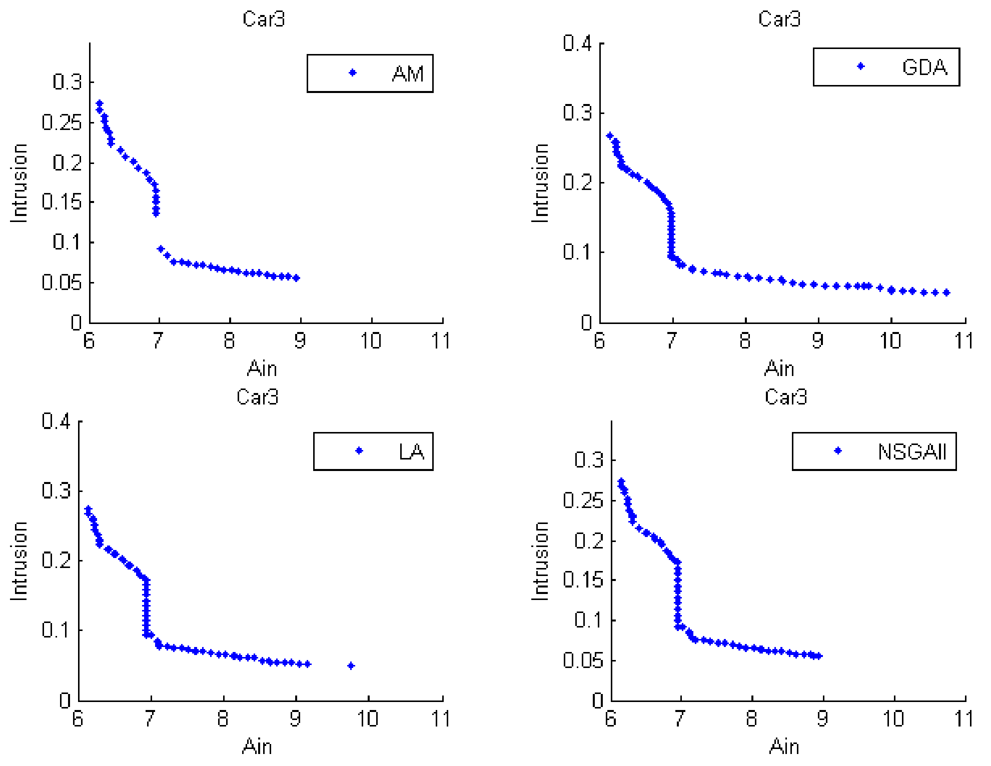


Figure 8.14: The 50% attainment surfaces for NSGAII and the three multi-objective choice function based hyper-heuristics (AM, GDA and LA) after 3,750 generations on the multi-objective design of vehicle crashworthiness problem (Car3).

8.4 Summary and Remarks

In this chapter, we have applied our multi-objective choice function based hyper-heuristics to the vehicle crashworthiness design as a real-world multi-objective problem to assess the level of generality they can achieve. The performance of our multi-objective choice function based hyper-heuristics are compared to the well-known multi-objective algorithm, NSGAII. In general, the results demonstrate the effectiveness of our selection hyper-heuristics particularly when combined with great deluge algorithm as a move acceptance criterion.

The multi-objective choice function great deluge based hyper-heuristic (HHMO_CF_GDA) beats other methods for solving both tri- objective vehicle crashworthiness design problem and bi-objective additional instances. It also benefits from the combination of GDA as an acceptance strategy and the choice function as the selection method. It is worthwhile mentioning that this result concurs with the findings in Chapter 7. In addition, HHMO_CF_GDA excels over NSGAII on all instances of the problem. HHMO_CF_GDA turns out to be the best choice for solving this problem. Although other multi-objective hyper-heuristics still produce solutions with acceptable quality in some cases, they could not perform as well as NSGAII. The reason for this relies on the move acceptance strategy they employed. A sensitivity analysis of our multi-objective choice function based hyper-heuristic was carried out and revealed a larger number of decision points (NDP) produce better solutions for the vehicles crashworthiness design problem. This indicates that the number of moves (decision point/iteration) conducted during the search could affect the performance of the multi-objective selection choice function based hyper-heuristic.

In summary, the results of the real-world problem demonstrate the capability and potential of the multi-objective hyper-heuristic approaches in solving continuous multi-objective optimisation problems.

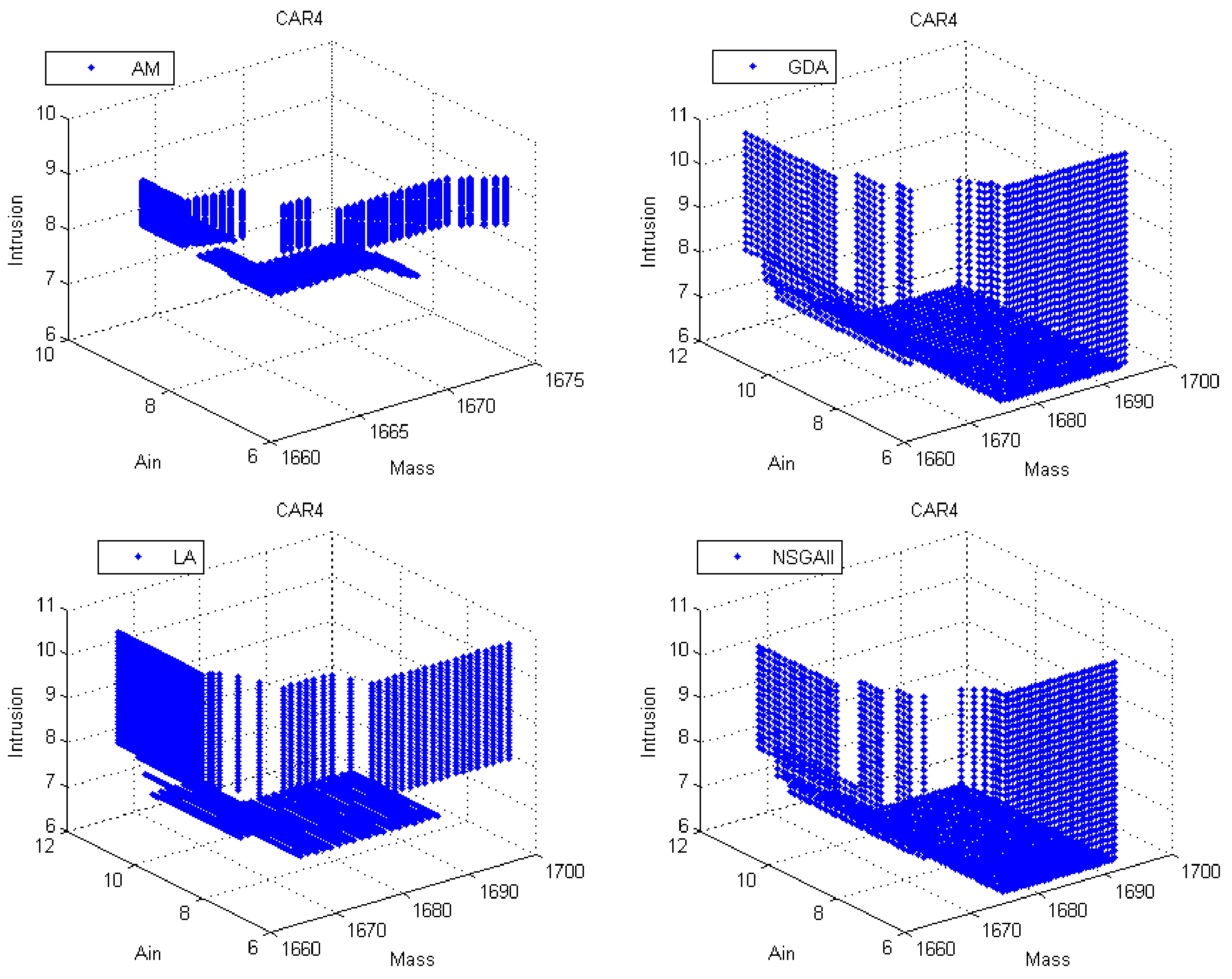


Figure 8.15: The 50% attainment surfaces for NSGAII and the three multi-objective choice function based hyper-heuristics (AM, GDA and LA) after 3,750 generations on the multi-objective design of vehicle crashworthiness problem (Car4).

9 Conclusions and Future Work

9.1 Conclusions

Hyper-heuristics are methodologies that operate on a search space of heuristics rather than solutions directly for solving hard computational problems. They have drawn increasing attention from the research community in recent years. However, the majority of hyper-heuristics studies have been limited to single-objective optimisation (Burke et al., 2013). Hyper-heuristics for multi-objective optimisation is a relatively new area of research in Operational Research and Evolutionary Computation (Burke et al., 2010; Özcan et al., 2008). Few studies were identified that deal with hyper-heuristics for multi-objective problems (e.g. (Burke et al., 2003a; Vrugt and Robinson, 2007; Veerapen et al., 2009; McClymont and Keedwell, 2011; Wang and Li, 2010; Gomez and Terashima-Marỉn, 2010)). None of these studies used multi-objective evolutionary algorithms (MOEAs), only in Rafique (2012), Gomez and Terashima-Marỉn, (2010), Vrugt and Robinson (2007), and no continuous and standard multi-objective test problems studied, only in McClymont and Keedwell (2011), Vrugt and Robinson (2007), Len et al. (2009) and Vázquez-Rodríguez and Petrovic (2013). Moreover, none of the previous hyper-heuristics made use of the components particularly designed for multi-objective optimisation that we introduced in this thesis. The main aim of this research was to investigate hyper-heuristic methodologies for multi-objective optimisation combining MOEAs with the goal of producing a set of high quality solutions (i.e. not necessarily optimal) compared to the existing approaches in the MOEA literature. The scope of this study is limited to continuous unconstrained multi-objective (two and three objectives) problems. We have investigated into the design of a generic selection hyper-heuristic framework for tackling multi-objective optimisation problems and development of effective hyper-heuristics within this multi-objective framework. The performance of different selection hyper-heuristics are tested over both benchmark test problems and real-world application. The main contributions and findings are summarised in the following subsections.

9.1.1 The Online Learning Selection Hyper-heuristic Framework for Multi-objective Optimisation

In this thesis, for the first time, we introduced an online learning selection choice function based hyper-heuristic framework for multi-objective optimisation (see Chapter 4). This framework is inspired from two facts: (i) there is no existing algorithm which excels across all types of problems, and (ii) there is empirical evidence showing that hybridisation or combining different (meta-)heuristics/algorithms could yield improved performance compared to (meta-)heuristics/ algorithms run on their own. Hyper-heuristic frameworks, generally, impose a domain barrier which separates the hyper-heuristic from the domain implementation along with low level heuristics to provide a higher level of abstraction. The domain barrier does not allow any problem specific information to be passed to the hyper-heuristic itself during the search process. We designed our framework in this same modular manner. One of advantages of the proposed framework is its simplicity. The proposed framework is highly flexible and its components reusable. It is built on an interface which allows other researchers to write their own hyper-heuristic components easily. Even the low level heuristics can be easily changed if required. If new and better performing components are found in the future, the software can be easily modified to include those components for testing. Our online selection choice function based hyper-heuristic for multi-objective optimisation (HHMO_CF) controls and combines the strengths of three well-known multi-objective evolutionary algorithms (NSGAII, SPEA2, and MOGA), which are utilised as the low level heuristics. The choice function utilised as a selection mechanism and a high level strategy which adaptively ranks the performance of three low-level heuristics, deciding which one to call at each decision point. The reason of use of the choice function as selection method is that it provides a balance between intensification and diversification. In addition, it was successful when used as a selection method in the hyper-heuristic for single-objective optimisation (Soubeiga, 2003; Bia, 2005). In our multi-objective hyper-heuristic framework, learning process is an essential component for guiding the heuristic selection method while it decides on the *most appropriate* heuristic to apply at each step of the iterative approach. The results that reported in Chapter 5 demonstrate that effectiveness of the learning multi-objective hyper-heuristic approach when compared to the one with no learning mechanism. This is understandable, as it has been observed that the learning mechanism adaptively successfully guides the search process towards the POF. In our learning multi-objective choice function based hyper-heuristic framework, we employed four performance metrics (Algorithm effort (AE), Ratio of non-dominated individuals (RNI), Size of space covered (SSC) and Uniform distribution of a non-dominated population (UD)) to act as an online learning mechanism to provide knowledge of the problem domain to the selection mechanism. The motivation behind choosing these metrics is that they have been commonly used for performance comparison of approaches for multi-objective optimisation to measure different aspects of the final non-dominated solutions in the objective space (Tan et al., 2002). In addition, they do not require a prior knowledge of the POF, which means that our framework is suitable for tackling real-world problems in future studies (see Chapter 8). Four performance metrics are integrated into a ranking scheme that we introduced in this study for the first time (see Section 4.2). The task of online learning ranking scheme is to score the performance of low level heuristics. Unlike the ranking scheme used in Vázquez-Rodríguez and Petrovic (2012) which orders the algorithms based on their probabilities against the performance indicators' using a mixture of experiments, our ranking scheme relies on sorting the low level heuristics in descending order based on the highest ranking among the other heuristics. Our ranking scheme is simple and flexible and enables us to incorporate any number of low level heuristics.

9.1.2 Three Multi-objective Choice Function Based Hyper-heuristics.

There is strong empirical evidence showing that different combinations of heuristic selection and acceptance methods in a selection hyper-heuristic framework yield different performances in single-objective optimisation (Burke et al., 2012). In this thesis, we investigated the influence of combining different acceptance methods under our online learning multi-objective choice function based hyper-heuristic framework that presented in Chapter 4. Three multi-objective choice function based hyper-heuristic combined with different move acceptance strategies including All-Moves as a deterministic move acceptance and Great Deluge Algorithm (GDA) and Late Acceptance (LA) as a non-deterministic move acceptance are presented in Chapters 5, 6 and 7 respectively.

The first multi-objective hyper-heuristic is utilised the choice function as a heuristic selection method and All-Moves as a deterministic move acceptance strategy (HHMO_CF_AM) (see Chapter 5). The choice function based hyper-heuristic was initially reported to perform well when combined with All-Moves acceptance for solving a single-objective optimisation problem (Cowling et al., 2002c). Thus, we chose All-Moves as a move acceptance strategy in our multi-objective hyper-heuristic framework, meaning that we accept the output of each low level heuristic whether it improves the quality of the solution or not.

A number of experiments are conducted to examine the performance of HHMO_CF_AM comparing to the low level heuristics (NSGAII, SPEA2 and MOGA), when used in isolation. It was shown that HHMO_CF_AM can benefit from the strengths of the low level heuristics. Unfortunately, it cannot avoid the weaknesses of them fully, as the poor performance of MOGA affects the performance of HHMO_CF_AM badly with respect to the ratio of non-dominated individual (RNI) by producing low number of non-dominated solutions. Another reason is that our multi-objective hyper-heuristic framework does not employ any archive mechanisms to maintain the number of individual in the population. To overcome this issue, we had two options: (i) employing an archive mechanism or (ii) employing a different move acceptance strategy that allows worsening moves to a limited degree. As we aim to keep our multi-objective hyper-heuristic framework in the same level of abstraction and not to break the domain barrier by incorporating an archive mechanism along the low level heuristics, the first option is ignored. So, we employed another acceptance strategy instead of All-Moves to avoid acceptance of all worsening moves.

In Chapters 6 and 7, we investigated the behaviour of great deluge algorithm (GDA) and late acceptance (LA) as non-deterministic move acceptance strategies under the choice function based hyper-heuristic framework designed for solving multi-objective optimisation problems. To the best of our knowledge, for the first time, this study investigated the influence of move acceptance component of selection hyper-heuristics for multi-objective optimisation. The motivation for choosing GDA and LA as acceptance criteria is that both are simple and do not depend on many parameters, requiring less effort for parameter tuning. More importantly, encouraging results have been reported in the literature for single-objective optimisation, but there are a few studies on their application to multi-objective optimisation (e.g. Petrovic and Bykov, 2003). The GDA and LA as move acceptance strategies require computation of the change in the value of a single-objective at each step and so the use of *D* performance metric (Zitzler, 1999) is proposed in order to be able to utilise those move acceptance methods under the proposed multi-objective framework. *D* metric is usually used in the literature as a performance metric to compare the final solutions obtained from multi-objective optimisers. In this thesis, we used *D* metric integrating into move acceptance criterion in order to covert the multi-objective optimisation to the single-objective optimisation without definition of criteria values' weights. *D* metric is used as a way of comparing two non-dominated sets of solutions in the objective space. The goal is set to as optimising (maximising) the *D* metric instead of a set of objectives simultaneously (see Sections 6.2 and 7.2). The choice function great deluge based hyper-heuristic (HHMO_CF_GDA) and choice function late acceptance based hyper-heuristic (HHMO_CF_LA) outperforms the choice function all-moves based hyper-heuristic (HHMO_CF_AM), indicating that the non-deterministic move acceptance strategies (GDA and LA) improve the performance of the multi-objective choice function based hyper-heuristic. Moreover, the multi-objective choice function based hyper-heuristics using non-deterministic move acceptance can successfully avoid accepting worse moves which result in the production of a low number of non-dominated individuals, as in the case of the original approach (HHMO_CF_AM). The main drawback of our selection multi-objective hyper-heuristic is not exhibiting the feature of multi-objective evolutionary algorithms, which act as low level heuristics. They are stochastic and the decision of the acceptance move is made after a single run only. To overcome this, we can execute each low level heuristic for many runs then make the acceptance move decision, but this is could be computationally expensive.

9.1.3 Application of Proposed Hyper-heuristics to Benchmark Test Problems and Real-world Problems

In this thesis, our multi-objective choice function based hyper-heuristics are evaluated over two problems; the Walking Fish Group (WFG) test problems (Huband et al., 2006) as our multi-objective benchmark test dataset and the multi-objective vehicle crashworthiness design problem (Liao et al., 2008) as a real-world problem.

The WFG test suite includes different test problems which consist of a wide range of characteristics and features (see Section 3.3.3). The WFG test suite has a number of instances that have features that are not included in other test suites, such as ZDT and DTLZ. Moreover, the WFG test suite is an excellent tool for comparing the performance of EAs, and they are the common choice for most MOEA researchers (Huband et al., 2006).

Our multi-objective choice function based hyper-heuristics presented in this thesis produced good results with acceptable quality over the nine WFG test problems including bi-objective and tri-objective. These results are reported in Chapters 5 and 7. We evaluated our approaches using two objective and three objective problems. In Chapter 4, the choice function heuristic selection combined with All-moves acceptance method (HHMO_CF_AM) are compared to the low level heuristics on their own. It was shown that HHMO_CF_AM performs better than MOGA over the bi-objective WFG test functions in terms of the distribution of non-dominated individuals along the POF., HHMO_CF_AM obtains competitive results performing better than NSGAII in terms of convergence towards the POF. However, HHMO_CF_AM fails to deliver a better performance as compared to NSGAII and SPEA2 in terms of number of non-dominated solutions. HHMO_CF_AM cannot avoid the weakness of MOGA with respect to this quality measure. Still, HHMO_CF_AM outperforms the adaptive multi-method search (AMALGAM) (Vrugt and Robinson, 2007) over the same test instances. The superiority of HHMO_CF_AM is due to online learning heuristic selection mechanism and the effective ranking scheme. The ranking scheme maintains the past performance of low level heuristics using a set of performance indicators that measure different aspects of the solutions. During the search process, the ranking scheme creates a balance between choosing the low level heuristics and their performance according to a particular quality metric. This balance enhances the algorithm performance to yield better solutions that converge toward the POF as well as distribute uniformly along the POF.

In Chapter 7, it was shown that the two multi-objective choice function hyper-heuristics that combined with great deluge and late acceptance as non-deterministic move acceptance criteria (HHMO_CF_GDA and HHMO_CF_LA) superior to the multi-objective choice functions based hyper-heuristic that combined with All-Moves as deterministic move acceptance criterion (HHMO_CF_AM) over both bi-objective and tri- objective WFG test functions.

The non-deterministic move acceptance methods in particularly GDA and LA improve the overall performance of the hyper-heuristic with respect to the number of the solutions, convergence and diversity. However, All-Moves still performs the best and produces better solutions in terms of the uniform distribution of non-dominated solutions. The success of HHMO_CF_AM with respect the uniform distribution of non-dominated solutions might be due to the use of the *D* metric into acceptance procedure for multi-objective non-deterministic acceptance based hyper-heuristics. Since *D* metric is a binary hypervolume measure that is designed to compare two sets of non-dominated solutions with respect of their convergence towards the POF, there is no consideration regarding how uniformly these solutions are distributed along the POF. This might also be a reason for why non-deterministic move acceptance procedures obtain high quality solutions in terms of the convergence towards the POF. In general, multi-objective choice function great deluge based hyper-heuristic (HHMO_CF_GDA) performs the best over WFG instances. The results in Chapter 7 provide an empirical evidence of mixing different combination of meta-heuristics under a selection hyper-heuristic framework yields with an improved performance. The use of the combination of the choice function as selection method and great deluge algorithm as acceptance strategy positively affect the performance of the multi-objective hyper-heuristics. The superiority of multi-objective choice function great deluge based hyper-heuristic is due to the acceptance procedure employed. Analysis of GDA behaviour as acceptance move strategy within the multi-objective choice function based hyper-heuristics framework is provided in Chapter 6.

Moreover, this observation is supported further by empirical evidence obtained from evaluating our multi-objective choice-function based hyper-heuristics against NSGAII over the vehicle crashworthiness design problems (See Chapter 8). The multi-objective choice function grate deluge based hyper-heuristic (HHMO_CF_GDA) beats others methods for solving both the original vehicle crashworthiness problem with three objectives and its bi-objective additional instances. HHMO_CF_GDA excels NSGAII over all instances of the problem. Although other multi-objective choice function based hyper-heuristics still produce solutions with acceptable quality, they could not perform better as well as NSGAII. The reason of this relies on the move acceptance strategy they are employed.

The results of both benchmark test problems (WFG) and the real-world problems (vehicle crashworthiness design) demonstrate the capability and potential of the multi-objective hyper-heuristic approaches in solving continuous multi-objective optimisation problems. The choice function great deluge based hyper-heuristic (HHMO_CF_GDA) mixing and managing population based multi-objective meta-heuristic algorithms turns out to be the best choice for multi-objective optimisation rather than running each meta-heuristic algorithm on its own.

9.2 Future Work

Our multi-objective choice function based hyper-heuristic framework which is used for managing a set of multi-objective meta-heuristics offers interesting potential research directions in multi-objective optimisation. We recommend three directions for future work as follows:

9.2.1 From the High Level Strategy Perspective

The empirical experiments demonstrate that combining different (meta)heuristic selection and move acceptance methods as components within a selection hyper-heuristic framework yield different performances in single-objective optimisation (Burke et al., 2012). In this thesis, we have adapted choice function as selection methods combined with three different acceptance methods, which are all-moves, great deluge algorithm and late acceptance, for multi-objective optimisation. More heuristic selection methods and can be adapted from previous research in single-objective optimisation and used for multi-objective optimisation. This process is not a trivial process requiring elaboration of existing methods and their usefulness in a multi-objective setting. Also other acceptance criteria such as simulated annealing (SA) and tabu search (TS) could be employed as a move acceptance component within our hyper-heuristic framework for multi-objective optimisation. As those criteria involve many parameters, this methodology would require initial experiments to tune the parameters for multi-objective settings such defining a cooling schedule and an initial temperature for SA and aspiration criterion and tabu tenure for TS.

In the context of multi-objective choice function great deluge based hyper-heuristic, it is suggested to tuning the rain speed (*UP*) parameter automatically based on the number of total moves in the search process in order to investigate great deluge algorithm as a move acceptance with re-levelling mechanism. This process requires resetting a water level (*LEVEL*) and setting a new rain speed rate (*UP*). This suggestion could improve the quality of results obtained from the original multi-objective choice function great deluge based hyper-heuristic that presented in this thesis (see Chapter 6). And make it applicable for wide range of problems. This may require further implementation of the high level strategy and more experiments could be done over the WFG test suite and other test problems.

9.2.2 From the Low Level Heuristics Perspective

Our multi-objective choice function based hyper-heuristic framework is designed to be highly flexible and its components can be reusable and easily replaceable. In this thesis, we employed and combined the strengths of three well-known multi-objective evolutionary algorithms (NSGAII, SPEA2, and MOGA) within our multi-objective selection hyper-heuristic framework (see Chapter 4). It would be interesting to employ other MOEA optimisers and other population-based methods to act as low level heuristics within the same framework. We anticipate that different low level heuristics could yield different performances. It would be so beneficial if replace MOGA with other more advance methods such as MOEA/D (Li and Zhang, 2009). There is huge numbers of low level heuristics choices possible and therefore great scope for research. Recent multi-objective hyper-heuristics studies obtain promising results. This is the case in MCHH (McClymont and Keedwell, 2011) using Evolution Strategies, and in AMALGAM (Vrugt and Robinson, 2007) using Particle Swarm Optimisation (Kennedy, 2001), Adaptive Metropolis Search (Haario et al., 2001) and Differential Evolution (Storn and Price, 1997).

9.2.3 From the Problem Domain

In this thesis, we evaluate our multi-objective choice function based hyper-heuristics over both problems: the WFG test suite (Huband et al., 2006) as our multi-objective benchmark test dataset and multi-objective vehicle crashworthiness design (Liao et al., 2008) as real-world problem. It would be interesting to test the level of generality of our multi-objective hyper-heuristics framework further on some other problems and domains including the continuous real-valued constrained, combinatorial, discrete and dynamic problems. The real-world water distribution networks design problems are applied to recent multi-objective hyper-heuristics studies in Raad et al. (2010) and McClymont et al. (2013) and produce encouraging results. In addition, extending our selection hyper-heuristics for many objectives optimisation would be an interesting direction research. This process might require adaptation of diversity management procedures and modification of Pareto-dominance.

References

Aarts, E. and Korst, J. (1998). *Simulated Annealing and Boltzman Machines*. Wiley.

Aarts, E., Korst, J., and Michiels,W. (2005). *Search Methodolgies: Introductory Tutorials in Optimization and Decision Support Techniques*, chapter Simulated Annealing. Springer.

Abuhamdah, A. (2010). Experimental result of late acceptance randomized descent algorithm for solving course timetabling problems. *IJCSNS International Journal of Computer Science and Network Security*, 10:192–200.

Abuhamdah, A. and Ayob, M. (2010). Average late acceptance randomized descent algorithm for solving course timetabling problems. In *Proceedings of Information Technology International Symposium (ITSim)*, pages 748–753.

Adra, S. and Fleming, P. (2011). Diversity management in evolutionary many-objective optimization. *IEEE Transaction on Evolutionary Computation*, 15(2):183–195.

Aickelin, U., Burke, E., and Li, J. (2009). Improved squeaky wheel optimisation for robust personnel scheduling. *IEEE Transactions on Evolutionary Computation*, 13(2):433–443.

Al-Milli, N. R. (2010). Hybrid genetic algorithms with great deluge for course timetabling. *International Journal of Computer Science and Network Security*, 10(4):283–288.

Allen, S., Burke, E., Hyde, M., and Kendall, G. (2009). Evolving reusable 3d packing heuristics with genetic programming. In *Proceedings of the ACM Genetic and Evolutionary Computation Conference (GECCO '09)*, pages 931–938.

Anderson, J. M., Sayers, T. M., and Bell, M. G. H. (2007). Optimisation of a fuzzy logic traffic signal controller by a multiobjective genetic algorithm. *IEEE Road Transport Information and Control*, 454:186–190.

Armas, J., Miranda, G., and Leòn, C. (2011). Hyperheuristic encoding scheme for multiobjective guillotine cutting problems. In *Proceedings of the 13th Annual Genetic and Evolutionary Computation Conference*, pages 1683–1690.

Auger, A., Bader, J., Brockhoff, D., and Zitzler, E. (2012). Hypervolume-based multiobjective optimization: Theoretical foundations and practical implications. *Theoretical Computer Science*, 425:75–103.

Bäck, T. (1996). *Evolutionary Algorithms in Theory and Practice*. Oxford University Press.

Bader, J. and Zitzler, E. (2011). Hype: An algorithm for fast hypervolume-based many-objective optimization. *Evolutionary Computation*, 19(1):45–76.

Bader-El-Den, M. and Poli, R. (2007). Generating sat local-search heuristics using a gp hyperheuristic framework. In *Proceedings of Artificial Evolution (EA'07)*.

Bai, R. (2005). *An Investigation Of Novel Approaches For Optimising Retail Shelf Space Allocation*. PhD thesis, School of Computer Science, University of Nottingham, UK.

Bai, R., Woensel, T., Kendall, G., and Burke, E. K. (2013). A new model and a hyper-heuristic approach for two-dimensional shelf space allocation. *Journal Operation Research*, 11:31–55

Bai, R., Blazewicz, J., Burke, E., Kendall, G., and McCollum, B. (2012). A simulated annealing hyper-heuristic methodology for exible decision support. 4OR: *A Quarterly Journal of Operations Research*, 10(1):43–66.

Bai, R. and Kendall, G. (2005). An investigation of automated planograms using a simulated annealing based hyper-heuristics. In Ibaraki, T., Nonobe, K., and Yagiura, M., editors, Metaheuristics: *Progress as Real Problem Solver, Operations Research/Computer Science Interface Series*, pages 87–108. Springer.

Barichard, V. and Hao, J. (2002). Un algorithme hybride pour le probl`eme de sac`a dos multiobjectifs. In *Proceedings of Huiti`emes Journ`ees Nationales sur la R`esolution Pratique de Probl`emes NP-Complets JNPC2002*.

Batista, L., Campelo, F., Guimares, F., and Ramírez, J. (2010). A new self-adaptive approach for evolutionary multiobjective optimization. *In Proceedings of IEEE Congress on Evolutionary Computation*, pages 1–8.

Battiti, R. and Protasi, M. (1997). Reactive search, a history-base heuristic for max-sat. *ACM Journal of Experimental Algorithmics*, 2.

Baykasoglu, A., Durmusoglu, Z. D., and Kaplanoglu, V. (2011). Training fuzzy cognitive maps via extended great deluge algorithm with applications. *Computers in Industry*, 62(2):187–195.

Baykasoglu, A., Owen, S., and Gindy, N. (1999). A taboo search based approach to find the pareto optimal set in multiple objective optimisation. *Engineering Optimization*, 31:731–748.

Benson, H. P. and Sayin, S. (1997). Towards finding global representations of the efficient set in multiple objective mathematical programming. *Naval Research Logistics*, 44:47–67.

Berberoglu, A. and Uyar, A. (2011). Experimental comparison of selection hyper-heuristics for the short-term electrical power generation scheduling problem. In Malyshkin, V., editor, *Proceedings of EvoApplications,Part II, Lecture Notes in Computer Science*, pages 444–453. Springer.

Beume, N., Naujoks, B., and Emmerich, M. (2007). Sms-emoa: multiobjective selection based on dominated hypervolume. *European Journal of Operational Research*, 181(3):1653–1669.

Bianchi, L., Dorigo, M., Gambardella, L., and Gutjahr, W. (2009). A survey on metaheuristics for stochastic combinatorial optimization. *Natural Computing*, 8(2):239–287.

Biazzini, M., Banhelyi, B., Montresor, A., and Jelasity, M. (2009). Distributed hyper-heuristics for real parameter optimization. In Proceedings of the 11th Annual Conference on Genetic and Evolutionary Computation, pages 1339–1346.

Bilgin, B., Ozcan, E., and Korkmaz, E. (2007). An experimental study on hyper-heuristics and final exam scheduling. In *Proceedings of the International Conference on the Practice and Theory of Automated Timetabling VI*, volume 3867 *of Lecture Notes in Computer Science*, pages 394–412. Springer Berlin /Heidelberg.

Blazewicz, J., Domschke, W., and Pesch, E. (1996). The job shop scheduling problem: Conventional and new solution techniques. *European Journal of Operational Research*, 93:1–33.

Blum, C. and Roli, A. (2003). Metaheuristics in combinatorial optimization: Overview and conceptual comparison. *ACM Computing Surveys*, 35(3):268–308.

Bradstreet, L., Barone, L., While, L., Huband, S., and Hingston, P. (2007). Use of the wfg toolkit and pisa for comparison of multi-objective evolutionary algorithms. In *Proceedings of IEEE Symposium on Computational Intelligence in Multi-criteria Decision-making*, pages 382–389.

Branke, J. (1999). Memory enhanced evolutionary algorithms for changing optimization problems. In *Proceedings of IEEE International Conference on Evolutionary Computation*, pages 1875–1882.

Bremermann, H. J. (1958). The evolution of intelligence. the nervous system as a model of its environment. Technical report, Department of Mathematics, University of Washington, Seattle, WA.

Burke, E., Landa-Silva, D., and Soubeiga, E. (2003a). Multi-objective hyper-heuristic approaches for space allocation and timetabling. In *MIC 2003-Meta-heuristics: Progress as Real Problem Solvers*, pages 129–158.

Burke, E., Kendall, G., Newall, J., Hart, E., Ross, P., and Schulenburg, S. (2003b). *Handbook of Meta-Heuristics*, chapter Hyper-Heuristics: An Emerging Direction in Modern Search Technology, pages 457–474. Kluwer Academic Publishers.

Burke, E., Kendall, G., and Soubeiga, E. (2003c). A tabu-search hyperheuristic for timetabling and rostering. *Journal of Heuristics*, 9(6):451–470.

Burke, E., Bykov, Y., Newall, J., and Petrovic, S. (2004). A time-predefined local search approach to exam timetabling problems. *IIE Transactions*, 36(6):509–528.

Burke, E., Hyde, M., and Kendall, G. (2006). Evolving bin packing heuristics with genetic programming. In *Proceedings of the 9th International Conference on Parallel Problem Solving from Nature (PPSN'06)*, pages 860–869.

Burke, E., McCollum, B., Meisels, A., Petrovic, S., and Qu, R. (2007a). A graph-based hyperheuristic for educational timetabling problems. *European Journal of Operational Research*, 176:177–192.

Burke, E., Hyde, M., Kendall, G., and Woodward, J. (2007a). The scalability of evolved on line bin packing heuristics. In *Proceedings of 2007 IEEE Congress on Evolutionary Computation*, pages 2530–2537.

Burke, E., Hyde, M., Kendall, G., and Woodward, J. (2007b). Automatic heuristic generation with genetic programming: Evolving a jack-of-all-trades or a master of one. In *Proceedings of the 9th ACM Genetic and Evolutionary Computation Conference (GECCO'07)*, pages 1559–1565.

Burke, E. K., Hyde, M., Kendall, G., Ochoa, G., Özcan, E., and Woodward, J. (2009). Exploring hyper-heuristic methodologies with genetic programming. In Mumford, C. and Jain, L., editors, *Computational Intelligence*, Intelligent Systems Reference Library, pages 177–201. Springer.

Burke, E., Hyde, M., Kendall, G., Ochoa, G., Özcan, E. and Woodward, J. R. (2010). *Handbook of Meta-Heuristics*, chapter A Classification of Hyper-heuristic Approaches, pages 449-468. Kluwer Academic Publishers.

Burke, E. K., Kendall, G., Misir, M., and Özcan, E. (2012). Monte carlo hyper-heuristics for examination timetabling. *Annals of Operations Research*, 196:73–90.

Burke, E. K., Gendreau, M., Hyde, M., Kendall, G., Ochoa, G., Özcan, E., and Qu, R. (2013). Hyper-heuristics: A survey of the state of the art. *Journal of the Operational Research Society*, 64:1695-1724.

Burke, E. and Newall, J. (2004). A tabu-search hyperheuristic for timetabling and rostering. *Annals of Operations Research*, 129:107–134.

Burke, E. K. and Bykov, Y. (2006). Solving exam timetabling problems with flex-deluge algorithm. In *Sixth International Conference on Practice and Theory of Automated Timetabling (PATAT 2006)*, pages 370–372.

Burke, E. K. and Bykov, Y. (2008). A late acceptance strategy in hill-climbing for exam timetabling problems. In *International Conference on the Practice and Theory of Automated Timetabling*.

Burke, E. K. and Bykov, Y. (2012). The late acceptance hill-climbing heuristic. Technical Report CSM-192, Computing Science and Mathematics, University of Stirling.

Bykov, Y. (2003). *Time-Predefined and Trajectory-Based Search: Single and Multiobjective Approaches to Exam Timetabling*. PhD thesis, School of Computer Science, University of Nottingham, UK.

Bykov, Y. (2008). The late acceptance hill-climbing algorithm for the magic square problem. In *Proceedings of International Conference on the Practice and Theory of Automated Timetabling (the PATAT 2008)*.

Chow, J. and Regan, A. (2012). A surrogate-based multiobjective metaheuristic and network degradation simulation model for robust toll pricing. Civil Engineering Working Papers.

Cobos, C., Mendoza, M., and Leon, E. (2011). A hyper-heuristic approach to design and tuning heuristic methods for web document clustering. In Proceedings of 20011 IEEE Congress on Evolutionary Computation, pages 1350–1358.

Coello, C. C. and Pulido, G. (2001). Multiobjective optimization using a micro-genetic algoritm. In *Proceedings of Genetic and Evolutionary Computation Conference*, pages 274–282.

Coello, C. C., Van Veldhuizen, D. V., and Lamont, G. (2007a). *Evolutionary Algorithms for Solving Multi-Objective Problems*. Kluwer Academic Publishers.

Coello, C. C., Van Veldhuizen, D. and Lamont, G. (2007b). *MOEA Test Suites, Evolutionary Algorithms for Solving Multi-Objective Problems*, Kluwer Academic Publishers.

Coello Coello, C. A. and Cruz Cortés, N. (2005). Solving multiobjective optimization problems using an artificial immune system. *Genetic Programming and Evolvable Machines*, 6(2):163–190.

Cook, S. (1971). The complexity of theorem-proving procedures. In *Proceedings of third annual ACM symposium on Theory of Computing*, pages 151–158.

Corne, D., Dorigo, M., and Glover, F. (1999). *New Ideas in Optimisation*. McGraw Hill.

Cowling, P., Kendall, G., and Soubeiga, E. (2000). A hyperheuristic approach for scheduling a sales summit. In *Selected Papers of the Third International Conference on the Practice And Theory of Automated Timetabling, PATAT 2000*, Lecture Notes in Computer Science, pages 176–190. Springer.

Cowling, P., Kendall, G., and Soubeiga, E. (2001). A parameter-free hyperheuristic for scheduling a sales summit. In *Proceedings of the 4th Metaheuristic International Conference*, pages 127–131.

Cowling, P., Kendall, G., and Han, L. (2002a). An adaptive length chromosome hyperheuristic genetic algorithm for a trainer scheduling problem. In *Proceedings of the 4th Asia-Pacific Conference on Simulated Evolution And Learning (SEAL'02),* pages 267–271.

Cowling, P., Kendall, G., and Han, L. (2002b). An investigation of a hyperheuristic genetic algorithm applied to a trainer scheduling problem. In *Proceedings of the Congress on Evolutionary Computation (CEC2002)*, pages 1185–1190.

Cowling, P., Kendall, G., and Soubeiga, E. (2002c). A hyper-heuristic approach to scheduling a sales summit. In *Proceedings of 3rd International Conference Practice and Theory of Automated Timetabling PATAT 2000*.

Crowston, W., Glover, F., Thompson, G., and Trawick, J. (1963). Probabilistic and parametric learning combinations of local job shop scheduling rules. *ONR Research memorandum*, 117.

Czyzak, P. and Jaszkiewicz, A. (1998). Pareto simulated annealing: A metaheuristic for multipleobjective combinatorial optimization. *Journal of Multicriteria Decision Analysis*, 7(1):34–47.

Darwin, C. (1859). *The origin of species*. Murray.

Davidor, Y. (1990). *Epistasis variance; A viewpoint on GA-hardness*. Rawlins.

Deb, K. (1999). Multi-objective genetic algorithms: Problem difficulties and construction of test problem. *Evolutionary Computation*, 7(3):205–230.

Deb, K. (2001). *Multi-Objective Optimization Using Evolutionary Algorithms*. Wiley.

Deb, K. (2005). *Introductory Tutorials in Optimization and Decision Support Methodologies*, chapter Multi-objective optimization. Search Methodologies, pages 273–316. Springer.

Deb, K. and Goldberg, D. (1989). An investigation on niche and species formation in genetic function optimization. In *Proceedings of 3rd International Conference on Genetic Algorithms*, pages 42–50.

Deb, K. and Agrawal, R. B. (1995). Simulated binary crossover for continuous search space. *Complex Systems*, 9:115–148.

Deb, K. and Goel, T. (2001). Controlled elitist nondominated sorting genetic algorithms for better convergence. In *Proceedings of Evolution Multi Criterion Optimization Conference*, pages 67–81.

Deb, K. and Jain, S. (2002). Running performance metrics for evolutionary multiobjective optimization. Technical Report KanGAL Report No. 2002004, Kanpur Genetic Algorithms Laboratory, Indian Institute of Technology Kanpur, India.

Deb, K., Thiele, L., Laumanns, M., and Zitzler, E. (2002). Scalable multi-objective optimization test problems. In *Proceedings of IEEE congress on evolutionary computation*, pages 825–830.

DellÁmico, M., Lodi, A., and Maffioli, F. (1999). Solution of the cumulative assignment problem with a well structured tabu search method. *Journal of Heuristics*, 5:123–143.

Demeester, P., Bilgin, B., Causmaecker, P., and Berghe, G. (2012). A hyperheuristic approach to examination timetabling problems: Benchmarks and a new problem from practice. *Journal of Scheduling*, 15(1):83–103.

Denzinger, J., Fuchs, M., and Fuchs, M. (1997). High performance atp systems by combining several ai methods. In *Proceedings of the Fifteenth International Joint Conference on Artificial Intelligence (IJCAI 97)*, pages 102–107.

Dhouib, S. (2000). A multi start great deluge metaheuristic for engineering design problems. In *Proceedings of the ACS/IEEE Int. Conference on Computer Systems and Applications (AICCSA)*, pages 1–5.

Dorigo, M., Maniezzo, V., and Colorni, A. (1996). Ant system: Optimisation by a colony of cooperating agents. *IEEE Transactions on Systems, Man, and Cybernetics-Part B*, 26(1):29–41.

Dorigo, M. and Stützle, T. (2003). *Handbook of Metaheuristics*, chapter The Ant Colony Optimization Metaheuristic: Algorithms, Applications, and Advances. Kluwer Academic Publishers.

Dorndorf and Pesch (1995). Evolution based learning in a job shop scheduling environment. *Computers and Operations Research*, 22:25–40.

Dowsland, K. (1995). Simulated Annealing. Advanced Topics in Computer Science:Modern Heuristic Techniques for Combinatorial Problems. McGraw-Hill International Limited.

Dowsland, K., Soubeiga, E., and Burke, E. (2007). A simulated annealing hyper-heuristic for determining shipper sizes. *European Journal of Operational Research*, 179(3):759-774.

Drake, J., Özcan, E., and Burke, E. (2012). An improved choice function heuristic selection for cross domain heuristic search. In *Parallel Problem Solving from Nature-PPSN XII*, volume 7492 of Lecture Notes in Computer Science, pages 307–316.

Dueck, G. (1993). New optimization heuristics: The great deluge algorithm and the record to record travel. *Journal of Computational Physics*, 104:86–92.

Durillo, J. (2011). *Metaheuristics for Multi-objective Optimization: Design, Analysis, and Applications*. PhD thesis, Universidad De Málaga.

Erickson, M., Mayer, A., and Horn, J. (2001). The niched pareto genetic algorithm 2 applied to the design of groundwater remedistion systems. In *Proceedings of the First International Conference on Evolutionary Multi-Criterion Optimization*, pages 681–695.

Fang, H., Rais-Rohani, M., Liu, Z., and Horstemeyer, M. (2005). A comparative study of metamodeling methods for multiobjective crashworthiness optimization. *Computers and Structures*, 83(25-26):2121-2136.

Fisher, H. and Thompson, G. (1961). Probabilistic learning combinations of local job-shop scheduling rules. In *Factory Scheduling Conference*.

Fisher, H. and Thompson, G. (1963). *Industrial Scheduling*, Chapter Probabilistic learning combinations of local job-shop scheduling rules, pages 225–251. Prentice Hall.

Fleming, P., Purshouse, R., and Lygoe, R. (2005). Many-objective optimization:an engineering design perspective. In *Lecture Notes in Computer Science*, pages 14–32. Springer-Verlag Berlin Heidelberg.

Focacci, F., Laburthe, F., and Lodi, A. (2003). *Handbook of Metaheuristics*, Chapter Local Search and Constraint Programming. Kluwer Academic Publishers.

Fonseca, C. M. and Fleming, P. J. (1993). Genetic algorithms for multiobjective optimization: Formulation, discussion and generalization. In *Proceedings of the 5th Genetic Algorithms Conference*, pages 416–423.

Fonseca, C. M. and Fleming, P. J. (1995). An overview of evolutionary algorithms in multiobjective optimization. *Evolutionary Computation*, 3(1):1–16.

Fonseca, C. and Fleming, P. (1998). Multiobjective optimization and multiple constraint handling with evolutionary algorithms. *IEEE Transactions on Systems, Man, and Cybernetics.-Part A: Systems and Humans*, 28(1):26–37.

Fraser, A. S. (1957). Simulation of genetic systems by automatic digital computers II: Effect of linkage on rates under selection. *Australian Journal of Biological Sciences*. 10:492-499.

Furtuna, R., Curteanu, S., and Leon, F. (2012). Multi-objective optimization of a stacked neural network using an evolutionary hyper-heuristic. *Applied Soft Computing*, 12(1).

Gandibleux, X. and Ehrgott, M. (2005). *Evolutionary Multi-Criterion Optimization*, chapter 1984- 2004:20Years of Multiobjective Metaheuristics. But What About the Solution of Combinatorial Problems with Multiple Objective. Springer.

Gandibleux, X. and Fréville, A. (2000). Tabu search based procedure for solving the 0-1 multiobjective knapsack problem: The two objectives case. *Journal of Heuristics*, 6(3):361–383.

Gandibleux, X., Mezdaoui, N., and Fréville, A. (1997). A tabu search procedure to solve multiobjective combinatorial optimization problems. In Malyshkin, V., editor, *Advances in Multiple Objective and Goal Programming*, volume 455 of *Lecture Notes in Economics and Mathematical Systems*, pages 291–300. Springer-Verlag.

Garey, M. and Johnson, D. (1979). *Computers and Intractability* - *A Guide to the Theory of NPCompleteness*. W.H. Freeman.

Gendreau, M. (2003). An Introduction to Tabu Search, chapter *Handbook of Metaheuristics*. Kluwer Academic Publishers.

Giagkiozis, I., Purshouse, R., and Fleming, P. (2013). An overview of population-based algorithms for multi-objective optimisation. *International Journal of Systems Science*. DOI:10.1080/00207721.2013.823526.

Gibbs, J., Kendall, G., and Özcan, E. (2011). Scheduling English football fixtures over the holiday period using hyper-heuristics. In *Proceedings of the 11th International Conference on Parallel Problem Solving From Nature*, volume 6238, pages 496–505.

Glover, F. (1977). Heuristics for integer programming using surrogate constraints. *Decision Sciences*, 8(1):156–166.

Glover, F. (1986). Future paths for integer programming and links to artificial intelligence. *Computer Operations Research*, 13(5):533–549.

Glover, F. and Laguna, M. (1995). *Modern heuristic techniques for combinatorial problems*, chapter Tabu search, pages 70–150.

Glover, F. and Laguna, M. (1997). *Evolutionary Algorithms for Solving Multi-Objective Problems*. Kluwer Academic Publishers.

Glover, F. and Kochenberger, G. (2003). *Handbook of Metaheuristics*. Kluwer Academic Publishers.

Glover, F., Laguna, M., and Martí, R. (2000). Fundamentals of scatter search and path relinking. *Control and Cybernetics*, 39(3):653–68.

Glover, F. and Laguna, M., and Martí, R. (2003). *Handbook of Metaheuristics*, chapter Scatter Search and Path Relinking: Advances and Applications, Kluwer Academic Publishers.

Goldberg, D. E. (1987). *Genetic Algorithms and Simulated Annealing*, chapter Simple genetic algorithms and the minimal, deceptive problem, pages 74–88. Morgan Kaufmann.

Goldberg, D. E. (1989). *Genetic Algorithms in Search Optimization and Machine Learning*. Addison-Wesley.

Goldberg, D. E. (1999), Using time efficiently: Genetic-evolutionary algorithms and the continuation problem. In *Proceedings of Genetic and Evolutionary Computation Conference*, pages 212-219.

Goldberg, D. E., and Rudnick, M. (1991). Genetic algorithms and the variance of fitness. *Complex Systems,* 5:265-278.

Gomez, J., and Terashima-Marín, H. (2010). Approximating multi-objective hyper-heuristics for solving 2d irregular cutting stock problems. In *Advances in Soft Computing*, Lecture Notes in Computer Science, pages 349–360.

Grobler, J., Engelbrecht, A., Kendall, G. and Yadavalli, V. (2012), Investigating the use of local search for improving meta-hyper-heuristic performance. In *Proceedings of IEEE* Congress on Evolutionary Computation, pages 1-8.

Grosan, C., Oltean, M., and Dumitrescu, D. (2003). Performance metrics for multiobjective optimization evolutionary algorithms. In *Proceedings of Conference on Applied and Industrial Mathematics*.

Haario, H., Saksman, E., and Tamminen, J. (2001). *An adaptive metropolis algorithm*. Bernoulli, 7:223–242.

Hajela, P. and Lin, C. (1992). Genetic search strategies in multicriterion optimal design. *Journal of Structural Optimization*. 4:99–107.

Hansen, P. and Mladenovic, N. (1999). *Meta-Heuristics: Advances and Trends in Local Search Paradigms for Optimization*, chapter An introduction to variable neighborhood search, pages 433-458, Kluwer Academic Publishers.

Hart, E., Ross, P. and Nelson, J. (1998). Solving a Real-World Problem Using an Evolving Heuristically Driven Schedule Builder, *Evolutionary Computation*. 6(1):61–80.

Henderson, D. (2003). *The Theory and Practice of Simulated Annealing. Handbook of Metaheuristics*, Kluwer Academic Publishers.

Holland, J. (1975). *Adaptation in Natural and Artificial Systems*, University of Michigan Press, Ann Harbor.

Horn, J. and Goldberg, D. E. (1995). Genetic algorithm difficulty and the modality of fitness landscape. In *Proceedings of the 3rd Workshop on Foundation of Genetic Algorithms*, pages 243–270.

Horn, J., Nafpliotis, N., and Goldberg, D. (1994). A Niched Pareto Genetic Algorithm for Multiobjective Optimization. In *Proceedings of the First IEEE Conference on Evolutionary Computation*, pages 82–87.

Huang, V. L., Qin, A. K., Deb, K., Zitzler, E., Suganthan, P. N., Liang, J. J., Preuss, M. and Huband, S. (2007). Problem Definitions for Performance Assessment of Multi-objective Optimization Algorithms. Technical Report. Nanyang Technological University, Singapore.

Huband, S., Hingston, P., While, L., and Barone, L. (2003). An evolution strategy with probabilistic mutation for multi-objective optimisation. In *Proceedings of Congress on Evolutionary Computation,* pages. 2284–2291.

Huband, S., Hingston, P., Barone, L., and While, L. (2006). A review of multiobjective test problems and a scalable test problem toolkit. *IEEE Transactions on Evolutionary Computation*, 10:477–506.

Hussin, N. (2005). *Tabu Search Based Hyper-heuristic Approaches for Examination Timetabling*. PhD thesis, The University of Nottingham, Nottingham, UK.

Ishibuchi H., Yoshida T. and Murata T. (2002). Selection of Initial Solutions for Local Search in Multiobjective Genetic Local Search. In *Proceedings of the 2002 Congress on Evolutionary Computation (CEC 2002)*, pages 950-955.

Jakobovic, D., Jelenkovic, L. and Budin, L. (2007) Genetic programming heuristics for multiple machine scheduling. In *Proceedings of the European Conference on Genetic Programming (EUROGP'07)*, pages 321–330.

Jaszkiewicz, A. (2001a). Comparison of local search-based metaheuristics on the multiple objective knapsack problem. *Foundations of Computing and Decision Sciences*, 26(1):99–120.

Jaszkiewicz, A. (2001b). Multiple objective genetic local search algorithm. In *Multiple Criteria Decision Making in the New Millennium*, volume 507 of *Lecture Notes in Economics and Mathematical Systems*, pages 231–240, Springer-Verlag.

Joslin, D. and Clements, D. (1999). Squeaky wheel optimization. *Journal of Artificial Intelligence Research,* 10:353-373.

Kargupta, H. (1995). Signal-to-noise, crosstalk, and long range problem difficulty in genetic algorithms. In *Proceedings of the 6th International Conference on Genetic Algorithms*, pages 193–200.

Keller, R. and Poli, R. (2007). Cost-benefit investigation of a genetic-programming hyperheuristic. *In Proceedings of Artificial Evolution (EA'07)*, pages 13–24.

Kendall, G., Cowling, P., and Soubeiga, E. (2002). Choice function and random hyperheuristics. In *Proceedings of the 4th Asia Pacific Conference on Simulated Evolution And Learning*, pages 667–671.

Kendall, G. and Mohamad, M. (2004). Channel assignment in cellular communication using a great deluge hyperheuristic. In *IEEE International Conference on Network*, pages 769–773.

Kennedy, J., Eberhart, R., and Shi, Y. (2001). *Swarm Intelligence*. Morgan Kaufmann.

Khare, V., Yao, X., and Deb, K. (2003). Performance scaling of multi-objective evolutionary algorithms. In *Proceedings of 2nd International Conference on Evolutionary Multi-Criterion Optimization*, pages 376–390.

Khor, E., Tan, K., Wang, M. and Lee, T. (2000). Evolutionary algorithm with dynamic population size formulti-objective optimization. In *Proceedings Conference on Simulated Evolution and Learning 2000 (SEAL'2000)*, pages 2768–2773.

Khor, E., Tan, K., and Lee, T. (2001). Tabu-based exploratory evolutionary algorithm for effective multi-objective optimization. In *Proceedings of The First International Conference on Evolutionary Multi-Criterion Optimization (EMO'01)*, pages 344–358.

Kiraz, B., Sima Uyar, A., and Özcan, E. (2011). An investigation of selection hyper-heuristics in dynamic environments. In *Applications of Evolutionary Computation*, volume 6624 of *Lecture Notes in Computer Science*, pages 314–323. Springer.

Kirkpatrick, S., Gelatt, C., and Vecchi, M. (1983). Optimization by simulated annealing. *Science*, 220:671–680.

Knowles, J. and Corne, D. (2000). Approximating the nondominated front using the pareto archived evolution strategy. *Evolutionary Computation*, 8(2):149–172.

Knowles, J. and Corne, D. (2002). One metrics for computing non-dominated set. In *Proceedings of the World Congress on Computational Intelligence*, pages 711–716.

Kokolo, I., Kita, H., and Kobayashi, S. (2001). Failure of pareto-based moeas: Does nondominated really mean near to optimal? In *Proceedings of the Congress on Evolutionary Computation*, pages 957–962.

Koza, J. (1992). *Solving multiobjective optimization problems using an artificial immune system*. MIT Press.

Krasnogor, N. (2002). *Studies on Theory and Design Space of Memtic Algorithms*. PhD thesis, University of the west of England, Bristol, UK.

Krasnogor, N. and Gustafson, S. (2004). A study on the use of self-generation in memetic algorithms. *Natural Computing*, 3(1):54–76.

Krasnogor, N. and Smith, J. E. (2002). Multimeme algorithms for the structure prediction and structure comparison of proteins. In *Proceedings of the Bird of a Feather Workshops, Genetic and Evolutionary Computation Conference (GECCO02)*, pages 42–44.

Krishniah, P. (1982). Selection of variables under univariate regression models. *Handbook of statistics*, 2:805–820.

Kumar, R., Bal, B., and Rockett, P. (2009). Multiobjective genetic programming approach to evolving heuristics for the bounded diameter minimum spanning tree problem. In *Proceedings of the ACM Genetic and Evolutionary Computation Conference (GECCO09)*, pages 309–316.

Kumari, A., Srinivas, K., and Gupta, M. (2013). Software module clustering using a hyperheuristic based multi-objective genetic algorithm. In *Advance Computing Conference (IACC), 2013 IEEE 3rd International*, pages 813–818.

Kursawe, F. (1990). A variant of evolution strategies for vector optimization. In *Parallel Problem Solving from Nature I (PPSN-I)*, pages 193–197.

Landa-Silva, J. D. (2003). *Metaheuristic and Multiobjective Approaches for Space Allocation*. PhD thesis, University of Nottingham, UK.

Landa-Silva, D. And Obit, J. (2009), Evolutionary Non-linear Great Deluge for University Course Timetabling. In *HAIS*, volume 5572 of *Lecture Notes in Computer Science*, page 269-276. Springer.

Landa-Silva, D., Burke, E., and Petrovic, S. (2004). An introduction to multiobjective metaheuristics for scheduling and timetabling. In *Lecture Notes in Economics and Mathematical Systems*, pages 91–129. Springer.

Lanzi, L., Castelletti, L., and Anghileri, M. (2004). Multi-objective optimisation of composite absorber shape under crashworthiness requirements. *Computer and Structures*, 65(3-4):433–441.

Len, C., Miranda, G., and Segura, C. (2009). Hyperheuristics for a dynamic-mapped multiobjective island-based model. In *Distributed Computing, Artificial Intelligence, Bioinformatics, Soft Computing, and Ambient Assisted*

Living, volume 5518 of *Lecture Notes in Computer Science*, pages 41–49. Springer Berlin Heidelberg.

Li, H. and Landa-Silva, D. (2011). An adaptive evolutionary multi-objective approach based on simulated annealing. *Evolutionary Computation*, 19(4):561–595.

Li, H. and Zhang, Q. (2009). Multiobjective optimization problems with complicated pareto sets, MOEA/D and NSGA-II. *IEEE Transactions on Evolutionary Computatio*n, 13(2):284–302.

Liao, X., Li, Q., Yang, X., Zhang, W., and Li, W. (2008). Multiobjective optimization for crash safety design of vehicles using stepwise regression model. *Structural and Multidisciplinary Optimization*, 35:561569.

Liu, D., Tan, K., Goh, C., and Ho, W. (2007). A multiobjective memetic algorithm based on particle swarm optimization. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, 37(1):42–50.

Loshchilov, I., Schoenauer, M., and Sebag, M. (2011). Not all parents are equal for mo-cmaes. In *Proceedings of the 6th International Conference on Evolutionary Multi-criterion Optimization EMO11*, pages 31–45.

Lourenco, H., Martin, O., and Stutzle, T. (2003). *Handbook of Metaheuristics*, chapter Iterated local search, pages 320–353. Springer.

Lu, H. and Yen, G. (2002). Rank-density based multiobjective genetic algorithm. In *Proceedings of the 2002 Congress on Evolutionary Computation (CEC 2002)*, pages 944–949.

Marklund, P. and Nilsson, L. (2001). Optimization of a car body component subjected to side impact. *Structural and Multidisciplinary Optimization*, 21(5):383–392.

Martin, O., Otto, S., and Felten, E. (1991). Large-step markov chains for the traveling salesman problem. *Complex Systems*, 5(3):299–326.

McClymont, K., Keedwell, E., and Savic, D. (2013). A general multi-objective hyper-heuristic for water distribution network design with discolouration risk. *to appear in Journal of Hydroinformatics*.

McClymont, K. and Keedwell, E. C. (2011). Markov chain hyperheuristic (mchh): an online selective hyper-heuristic for multiobjective continuous problems. In *Proceedings of Genetic and Evolutionary Computation Conference (GECCO11)*, pages 2003–2010.

McMullan, P. and McCollum, B. (2007). Dynamic job scheduling on the grid environment using the great deluge algorithm. In *Parallel Computing Technologies*, volume 4671 of *Lecture Notes in Computer Science*, pages 283–292. Springer Berlin/ Heidelberg.

Metropolis, N., Rosenbluth, A., Rosenbluth, M., Teller, A., and Teller, E. (1953). Equation of state calculations by fast computing machines. *Journal of Chemistry and Physics*, 21:1087–1092.

Miranda, G., Armas, J., Segura, C., and León, C. (2010). Hyperheuristic codification for the multi-objective 2d guillotine strip packing problem. In *Proceedings of IEEE Congress on Evolutionary Computation*, pages 1–8.

Misir, M., Vancroonenburg,W., Verbeeck, K., and Berghe, G. (2011). Aselection hyperheuristic for scheduling deliveries of ready-mixed concrete. In *Proceedings of the Metaheuristics International Conference*, pages 289–298.

Misir, M., Verbeeck, K., De Causmaecker, P., and Vanden Berghe, G. (2012). An intelligent hyperheuristic framework for chesc. In *International Conference on Learning and Intelligent Optimization (LION 6)*, volume 7219 of *Lecture Notes in Computer Science*, pages 461–466. Springer.

Moscato, P. (1999). *New Ideas in Optimisation, chapter Memetic algorithms: A short introduction*. McGraw Hill.

Murata, T. and Ishibuchi, H. (1995). Moga: Multi-objective genetic algorithms. In *Proceedings of the 1IEEE International Conference on Evolutionary Computation*, volume 1, pages 289–294.

Nahas, N., Nourelfath, M., and Ait-Kadi, D. (2010). Iterated great deluge for the dynamic facility layout problem. Technical report, CIRRELT.

Nam, D. and Park, C. (2000). Multiobjective simulated annealing: A comparative study to evolutionary algorithms. *International Journal of Fuzzy Systems*, 2(2):87–97.

Nareyek, A. (2003). *Metaheuristics: Computer Decision-Making*, chapter Choosing search heuristics by non-stationary reinforcement learning, pages 523–544. Kluwer.

Nebro, A. J. and Durillo, J. (2010). Study of the parallelization of the multi-objective metaheuristics MOEA/D. *Proceedings of 4th International Conference of Learning and Intelligent Optimization (LION 4),* 6073:303–317.

Nourelfath, M., Nahas, N., and Montreuil, B. (2007). Coupling ant colony optimization and the extended great deluge algorithm for the discrete facility layout problem. *Engineering Optimization*, 39(8):953–968.

Obit, J., Landa-Silva, D., Ouelhadj, D., and Sevaux, M. (2009). Non-linear great deluge with learning mechanism for solving the course timetabling problem. In *8th Metaheuristics International Conference (MIC 2009)*, pages 2010–2011.

Osman, H. (1993). Metastrategy simulated annealing and tabu search algorithms for the vehicle routing problem. *Annals of Operations Research*, 41:421–451.

Özcan, E., Bilgin, B., and Korkmaz, E. (2008). A comprehensive analysis of hyper-heuristics. *Intelligent Data Analysis*, 12(1):3–23.

Özcan, E., Bykov, Y., Birben, M., and Burke, E. K. (2009). Examination timetabling using late acceptance hyper-heuristics. In *the IEEE Congress on Evolutionary Computation*, pages 997–1004.

Özcan, E., Misir, M., Ochoa, G., and Burke, E. K. (2010). A reinforcement learning - great-deluge hyper-heuristic for examination timetabling. *International Journal of Applied Metaheuristic Computing*, 1(1):39–59.

Özcan, E. and Kheiri, A. (2011). A hyper-heuristic based on random gradient, greedy and dominance. In *Proceedings of Computer and Information Sciences II: 26th International Symposium on Computer and Information Science*s, pages 404–409.

Özcan, E. and Parkes, A. (2011). Policy matrix evolution for generation of heuristics. In *Proceedings of the 13th Annual Conference on Genetic and Evolutionary Computation (GECCO11)*, pages 2011–2018.

Petrovic, S. and Bykov, Y. (2003). A multiobjective optimisation technique for exam timetabling based on trajectories. In *Practice and Theory of Automated Timetabling IV*, volume 2740 of Lecture Notes in Computer Science, pages 181–194. Springer Berlin/Heidelberg.

Petrovic, S., Yang, Y., and Dror, M. (2000). Case-based selection of initialisation heuristics for metaheuristic examination timetabling. *Evolutionary Computation*, 8(2):173–195.

Petrovic, S., Yang, Y., and Dror, M. (2007). Case-based selection of initialisation heuristics for metaheuristic examination timetabling. *Expert Systems with Applications*, 33(3):772–785.

Pillay, N. (2008). An analysis of representations for hyper-heuristics for the uncapacitated examination timetabling problem in a genetic programming system. In *Proceedings of the 2008 Annual Conference of the South African Institute of Computer Scientists and Information Technologists on IT Research in Developing Countrie*s, SAICSIT Conference, pages 188–192.

Pisinger, D. and Ropke, S. (2007). A general heuristic for vehicle routing problems. *Computers and operations research*. 34:2403–2435.

Poloni, C., Giurgevich, A., Onesti, L., and Pediroda, V. (2000). Hybridization of a multiobjective genetic algorithm, a neural network and a classical optimizer for complex design problem in fluid dynamics. *Computer Methods in Applied Mechanics and Engineering*, 186(2-4):403–420.

Pramodh, C. and Ravi, V. (2007). Modified great deluge algorithm based auto associated neural network for bankruptcy prediction in banks. *International Journal of Computational Intelligence Research*, 3(4):363–370.

Pulido, G. and Coello Coello, C. (2003). The micro genetic algorithm 2:towards online adaptation in evolutionary multiobjective optimization. In Proceedings of *the 2nd International Conference on Evolutionary Multi-Criterion Optimization (EMO 2003)*, volume 2632 of Lecture Notes in Computer Science, pages 252–266. Springer-Verlag Berlin Heidelberg.

Purshouse, R. and Fleming, P. (2003a). Conflict, harmony, and independence: Relationships in evolutionary multicriterion optimisation. In Proceedings of *the 2nd International Conference on Evolutionary Multi-Criterion Optimization (EMO 2003),* volume 2632 of Lecture Notes in Computer Science, pages 16–30. Springer-Verlag Berlin Heidelberg.

Purshouse, R. and Fleming, P. (2003b). Evolutionary many-objective optimisation: an exploratory analysis. In Proceedings *of Evolutionary Computation* (CEC '03), pages 2066-2073.

Purshouse, R. and Fleming, P. (2007). On the evolutionary optimization of many conflicting objectives. *IEEE Transaction on Evolutionary Computation*, 11(6):770–784.

Qu, R. and Burke, E. (2009). Hybridisations within a graph based hyper-heuristic framework for university timetabling problems. Journal of the *Operational Research Society*, 60:1273–1285.

Raad, D., Sinkse, A., and Vuuren, J. (2010). Multiobjective optimization for water distribution systemdesign using a hyperheuristic. *Journal of Water Resources Management*, 136(5):592–596.

Rafique, A. F. (2012). Multiobjective hyper-heuristic scheme for system design and optimization. In *Proceedings of 9th International Conference on Mathematical Problems in Engineering, Aerospace and Science, AIP Conference* 1493, volume 764.

Rayward-Smith, V. (1986). *A First Course in Computability. Blackwell.*

Rechenberg, I. (1965). *Cybernetic solution path of an experimental problem*. Library Translation 1122, Royal Aircraft Establishment.

Redhe, M., Giger, M., and Nilsson, L. (2004). An investigation of structural optimization in crashworthiness design using a stochastic approach - a comparison of stochastic optimization and the response surface methodology. *Structural and Multidisciplinary Optimization*, 27(6):446–459.

Reeves, C. (2003). *Handbook of Metaheuristics*, chapter Genetic Algorithms. Kluwer Academic Publishers.

Ross, P. (2005). *Search Methodologies: Introductory Tutorials in Optimization and Decision Support Methodologies*, Chapter Hyper-heuristics, pages 529–556. Springer.

Ross, P. and Marn-Blazquez, J. G. (2005). Constructive hyper-heuristics in class timetabling. In *Proceedings of IEEE Congress on Evolutionary Computation*, pages 1493–1500.

Sastry, K., Goldbreg, D., and Kendall, G. (2005). *Search Methodolgies: Introductory Tutorials in Optimization and Decision Support Techniques*, chapter Genetic Algorithms. Springer.

Schaffer, J. (1985). Multiple objective optimization with vector evaluated genetic algorithms. In *Proceedings of the 11th Annual Conference on Genetic and Evolutionary Computation*, pages 93–100.

Scott, E. and Geldenhuysys, G. (2000). A comparison of the tabu search and great deluge methods for graph colouring. In *Proceedings of the 16th IMACS World Cong. on Scientific Computing Applied Mathematics and Simulation (IMACS 2000).*

Sin, E. S. and Kham, N. S. M. (2012). Hyper heuristic based on great deluge and its variants for exam timetabling problem. CoRR, 12021891.

Socha, K. and Kisiel-Dorohinicki, M. (2002). Agent-based evolutionary multiobjective optimization. In *Proceedings of the 2002 Congress on Evolutionary Computation (CEC 2002)*, pages 109–114.

Sörensen, K. and Glover, F. (2013). *Encyclopedia of Operations Research and Management Science*, chapter Metaheuristics. Springer.

Soubeiga, E. (2003). *Development and Application of Hyperheuristics to Personnel Scheduling*. PhD thesis, University of Nottingham, UK.

Srinivas, N. and Deb, K. (1994). Multiobjective optimization using nondominated sorting in genetic algorithms. *Evolutionary Computation*, 2(3):221–248.

Storn, R. and Price, K. (1997). Differential evolution: A simple and efficient heuristic for global optimization over continuous. *Journal of Global Optimization*, 11:341–359.

Stützle, T. (1999). *Local Search Algorithms for Combinatorial Problems, Analysis, Algorithms and New Applications*. DISKI.

Sutton, R. and Barto, A. (1998). *Reinforcement Learning: An Introduction*. MIT Press.

Taillard, E. (1991). Robust taboo search for the quadratic assignment problem. *Parallel Computing*, 17:443–455.

Tan, K., Lee, T., and Khor, E. (1999). Evolutionary algorithms with goal and priority information for multi-objective optimization. In *Proceedings of the 11th Annual Conference on Genetic and Evolutionary Computation*, pages 106–113.

Tan, K. C., Lee, T. H., and Khor, E. F. (2002). Evolutionary algorithms for multi-objective optimization: Performance assessments and comparisons. *Artificial Intelligence Review*, 17:253–290.

Tay, J. and Ho, B. (2008). Evolving dispatching rules using genetic programming for solving multi-objective flexible job-shop problems. *Computers and Industrial Engineering*, 54:453–473.

Telfar, G. (1995). *Generally applicable heuristicsfor global optimisation: An investigation of algorithm performance for the euclidean traveling salesman problem*. Master's thesis, Institute of Statistics and Operations Research, Victoria University of Wellington.

Terashima-Marín, H., Ortiz-Bayliss, J., Ross, P., and Valenzuela-Rendon, M. (2008). Hyperheuristics for the dynamic variable ordering in constraint satisfaction problems. In *Proceedings of Genetic and Evolutionary Computation Conference (GECCO08)*, pages 571–578.

Thanh, N. and Anh, D. (2009). Comparing three improved variants of simulated annealing for optimizing dorm room assignments. In *Proceeding of Computing and Communication Technologies International Conference RIVF09*.

Thompson, J. and Dowsland, K. (1996). Variants of simulated annealing for the examination timetabling problem. *Annals of Operations Research*, 63:105–128.

Tierney, K. (2013). Late acceptance hill climbing for the liner shipping fleet repositioning problem. In *Proceedings of the 14th EU/ME Workshop. Helmut-Schmidt-Universit¨at Hamburg*.

Ulungu, E. (1993). *Optimisation combinatoire multicrit'ere: Determination de l'ensemble des solutions efficaces et methodes interactives*. PhD thesis, Facult'e des Sciences, Universit'e de Mons-Hainaut. Mons, Belgium.

Van Veldhuizen, D. (1999). *Multiobjective Evolutionary Algorithms: Classifications, Analyses, and New Innovations*. PhD thesis, Air Force Institute of Technology, Wright-Patterson AFB. Ohio.

Van Veldhuizen, D. and Lamont, G. (1998a). Multiobjective evolutionary algorithm research: A history and analysis. Technical Report TR-98-03, Department of Electrical and Computer Engineering, Graduate School of Engineering.

Van Veldhuizen, D. A., and Lamont, G.. (1998b). Evolutionary computation and convergence to a pareto front. In *Proceedings of Late Breaking Papers at the Genetic Programming 1998 Conference*, pages 221–228.

Van Veldhuizen, D. V. and Lamont, G. (2000). Multiobjective evolutionary algorithms: Analyzing the state-of-the-art. *Evolutionary Computation*, 8(2):125–147.

Vázquez-Rodríguez, J. and Petrovic, S. (2012). Calibrating continuous multi-objective heuristics using mixture experiments. *The Journal of Heuristics*, 18:699–726.

Veerapen, N., Landa-Silva, D., and Gandibleux, X. (2009). Hyperheuristic as component of a multi-objective metaheuristic. In *Proceedings of the Doctoral Symposium on Engineering Stochastic Local Search Algorithms (SLS-DS 2009).*

Verstichel, J. and Berghe, G. V. (2009). A late acceptance algorithm for the lock scheduling problem. *Logistic Management*, 5:457–478.

Viennet, R. (1996). Multicriteria optimization using a genetic algorithm for determining the pareto set. *International Journal of System Science*, 27(2):255–260.

Vinkó, T. and Izzo, D. (2007). Learning the best combination of solvers in a distributed global optimization environment. In *Advances in Global Optimization: Methods and Applications*.

Voß, S., Martello, R., Osman, I., and Roucairol, C. (1999). *Meta-Heuristics: Advances and Trends in Local Search Paradigms for Optimization*. Kluwer Academic Publishers.

Voudouris, C. and Tsang, E. (1999). Guided local search and its application to the travelling salesman problem. *European Journal of Operational Research*, 113(2):469–499.

Voutchkov, I. and Keane, A. (2010). *Computational Intelligence in Optimization: Adaptation, Learning, and Optimization*, chapter Multi-objective optimization using surrogates, pages 155–175.

Vrugt, J. and Robinson, B. (2007). Improved evolutionary optimization from genetically adaptive multimethod search. *Proceedings of the National Academy of Sciences*, 104(3):708–711.

Vrugt, J., Robinson, B., and Hyman, J. (2010). Comment on paper ”multi-strategy ensemble evolutionary algorithm for dynamic multi-objective optimization” by wang and li. *Memetic Computing*, 2(2):161–162.

Wang, R., Purshouse, R., and Fleming, P. (2013). Preference-inspired coevolutionary algorithms for many-objective optimization. *IEEE Transaction on Evolutionary Computation*, 17(4):474–494.

Wang, Y. and Li, B. (2010). Multi-strategy ensemble evolutionary optimization for dynamic multiobjective optimization. *Memetic Computing*, 2:3–24.

Watanabe, S., Hiroyasu, T., and Miki, M. (2002). Lcga: Local cultivation genetic algorithm for multi-objective optimization problem. In *Proceedings of Genetic and Evolutionary Computation Conference (GECC2002)*.

Whitley, L. D. (1991). *Foundations of Genetic Algorithms, chapter Fundamental principles of deception in genetic search*, pages 221–241. Morgan Kaufmann.

Wolpert, D. and Macready,W. (1997). No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation*, 1(1):67–8.

Yuan, B., Zhang, C., and Shao, X. (2013). A late acceptance hill-climbing algorithm for balancing two-sided assembly lines with multiple constraints. *Journal of Intelligent Manufacturing*. Publish online.

Zhang, Q. and Li, H. (2007). Evolutionary algorithms for multi-objective optimization: Performance assessments and comparisons. *IEEE Transactions on Evolutionary Computation*, 11(6):712–731.

Zhang, X., Srinivasan, R., and Liew, M. V. (2010). On the use of multi-algorithm, genetically adaptive multi-objective method for multi-site calibration of the swat model. *Hydrological Processes*, 24(8):955–1094.

Zhou, A., Qu, B. Y., Li, H., Zhao, S. Z., Suganthan, P. N., and Zhang, Q. (2011). Multiobjective evolutionary algorithms: A survey of the state of the art. *Swarm and Evolutionary Computation*, 1(1):32–49.

Zielinski,K., and Laur, R. (2007). Variants of Differential Evolution for Multi-Objective Optimization. In *Proceedings of the 2007 IEEE Symposium on Computational Intelligence in Multicriteria Decision Making (MCDM2007),* pages 91-99.

Zitzler, E. (1999). *Evolutionary Algorithms for Multiobjective Optimization: Methods and Applications*. PhD thesis, ETH Zurich, Switzerland.

Zitzler, E., Deb, K., and Thiele, L. (2000). Comparison of multiobjective evolutionary algorithms: Empirical results. *Evolutionary Computation*, 8(2):173–195.

Zitzler, E., Laumanns, M., and Thiele, L. (2001). Spea2: Improving the strength Pareto evolutionary algorithm for multiobjective optimization. In *EUROGEN 2001-Evolutionary Methods for Design, Optimization and Control with Applications to Industrial Problem*, pages 95–100.

Zitzler, E. and Thiele, L. (1999). Multiobjective evolutionary algorithms: A comparative case study and the strength Pareto approach. *IEEE Transactions on Evolutionary Computation*, 3(4):253–290.

Zitzler, E. and K¨ unzli, S. (2004). Indicator-based selection in multiobjective search. In *Lecture Notes in Computer Science*, Parallel Problem Solving from Nature (PPSN VIII), page 832-842.Springer.

Zydallis, J., Van Veldhuizen, D., and Lamont, G. (2001). A statistical comparison of multiobjective evolutionary algorithms including the MOMGA-II. In *Proceedings of the First International Conference on Evolutionary Multi-Criterion Optimization*, pages 226–240.