Move Acceptance in Local Search Metaheuristics for Cross-domain Search

Warren G. Jackson[a], Ender Özcan[a], Robert I. John[a]

[a] *ASAP Research Group*
School of Computer Science
University of Nottingham
NG8 1BB, Nottingham, UK

Abstract

Metaheuristics provide high-level instructions for designing heuristic optimisation algorithms and have been successfully applied to a range of computationally hard real-world problems. Local search metaheuristics operate under a single-point based search framework with the goal of iteratively improving a solution in hand over time with respect to a single objective using certain solution perturbation strategies, known as move operators, and move acceptance methods starting from an initially generated solution. Performance of a local search method varies from one domain to another, even from one instance to another in the same domain. There is a growing number of studies on 'more general' search methods referred to as cross-domain search methods, or hyper-heuristics, that operate at a high-level solving characteristically different problems, preferably without expert intervention. This paper provides a taxonomy and overview of existing local search metaheuristics along with an empirical study into the effects that move acceptance methods, as components of single-point based local search metaheuristics, have on the cross-domain performance of such algorithms for solving multiple combinatorial optimisation problems. The experimental results across a benchmark of nine different computationally hard problems highlight the shortcomings of existing and well-known methods for use as components of cross-domain search methods, despite being re-tuned for solving each domain.

Keywords: Combinatorial Optimization, Parameter Control, Stochastic Local Search, Trajectory Methods, Search Algorithms

1. Introduction

The search methodologies used for tackling real-world combinatorial optimisation problems (Papadimitriou and Steiglitz, 1982), such as Examination Timetabling (Petrovic et al., 2007), High School Timetabling (Ahmed et al., 2015), and Vehicle Routing (Tarantilis et al., 2004) have always been of interest to researchers and practitioners. *Metaheuristics* imposing 'a set of guidelines or

strategies' based on a heuristic search framework (Sörensen and Glover, 2013) can be preferred over exact methods; while *exact methods* can find optimal solutions, their computational efforts (memory usage, flops, etc) can sometimes be orders of magnitude higher than *heuristic methods*. This is mainly due to the fact that many real-world problems are computationally hard to solve (Garey, 1990; Ausiello et al., 1995) and exact methods can fail to produce acceptable solutions in a reasonable time frame. For example, a review of heuristic and exact algorithms (Drezner et al., 2005) for solving the Quadratic Assignment Problem emphasises this, stating that QAP instances most often discussed in the literature cannot be solved to optimality by exact methods when the problem sizes approach 30-40, whereas the heuristic search techniques can rapidly find high quality solutions.

Metaheuristics are classified by Sörensen and Glover (2013) into three categories; *local search*, *constructive*, and *population-based* (Beheshti and Shamsuddin, 2013). Birattari et al. (2001) categorised metaheuristics based on several distinctions; trajectory methods versus discontinuous methods, population-based versus single-point search, memory usage versus memoryless methods, one versus multiple neighbourhood structures, dynamic versus static objective function, and the difference between nature-inspired and non-nature inspired methods. A very similar categorisation is provided by Blum and Roli (2003). This study focuses on a subset of single objective local search metaheuristics embedding single-stage *move acceptance methods* under a single point based search framework, an outline of which is given in Algorithm 1. In the scientific literature, some of such local search metaheuristics can also be referred to as *stochastic local search methods* (Hoos and Stützle, 2004), *reactive search methods* (Battiti et al., 2008), *trajectory methods* (Blum and Roli, 2003) or *hyper-heuristics* (Burke et al., 2013). From this point onwards, we will refer to them solely as local search metaheuristics for consistency.

Algorithm 1: Outline of a Local Search Metaheuristic.

```
1  s ← generateInitialSolution();
2  s_best ← s;
3  while termination criteria not met do
4  |   s' ← apply(h, s);
5  |   process_1();
6  |   s ← acceptRejectDecision(s, s');
7  |   if f(s').isBetterThan(f(s_best)) then
8  |   |   s_best ← s';
9  |   end
10 |   process_2();
11 end
12 return s_best;
```

A local search metaheuristic iteratively makes changes (perturbations) to a complete solution (s_i) at each iteration of the search (i) to produce a sin-

gle neighbouring solution ($s_i^{'}$) by applying a *move operator*, also known as a heuristic, (h) to it (Line 4 of Algorithm 1). The resulting solution is then accepted or rejected for use as the solution in the next iteration (s_{i+1}) based on the decision of the move acceptance method shown in Line 6 of Algorithm 1. Finally, when the algorithm terminates, the best solution found so far (s_{best}) is returned. The *objective* (evaluation, fitness, cost) *function* ($f(.)$) measures the quality of a solution and guides the search. The processes to maintain and update relevant data structures in between the main steps of the local search metaheuristics shown in lines 5 and 10 of Algorithm 1 vary depending on the particular approach. For example, if no operation is involved in either process, and a move is accepted if and only if there is an improvement, then the algorithm becomes a local search (hill climbing) algorithm accepting only equal or improving moves. The behaviour of a local search metaheuristic, and hence its ability to find good quality solutions, is determined by its neighbourhood and solution acceptance strategy, as defined by the move operators and its move acceptance method respectively.

A well-known weakness of local search metaheuristics is their tendency to get stuck in local optima. They therefore require some mechanism to counteract such situations during the search process, enabling the algorithm to explore other regions of the search landscape, potentially leading to solutions with better quality (Hoos and Stützle, 2004). Hence, the use of a method accepting worse quality solutions as such a mechanism is a common strategy. There are many different previously proposed local search metaheuristics making use of a variety of move acceptance methods, such as Simulated Annealing (Kirkpatrick et al., 1983), and Great Deluge (Dueck, 1993).

The motivation of this study arises from the problem of selecting a move acceptance method for solving the cross-domain search problem. Traditionally when practitioners are faced with solving a problem (even those from within the same domain), in addition to choosing suitable move operators and objective functions, they also have to choose a suitable move acceptance method to use alongside them in order for the overall algorithm to perform well. For existing problems, the move acceptance method is usually chosen based on whichever worked well in the past, or which has received the most attention in the respective field. The choice of move acceptance method is, however, most often left unjustified. Moreover, this becomes an even greater issue when the problem to be solved is *new* or *unknown* because the practitioner is then left with a dilemma for choosing the best move acceptance method for solving that problem without previous experience, knowledge, or guidance. An emerging area of research concerns the development of high-level, general-purpose search methodologies known as cross-domain search methods. Cross-domain search is a term used to describe the problem of devising a single search method which can perform well across multiple characteristically different problems, preferably *without* expert intervention or modification (Ochoa et al., 2012). The problems traditionally used when benchmarking cross-domain search methods are those where the computational expense of their objective functions are not much of a concern. Another line of optimisation, called as inverse combinatorial optimi-

sation, includes such problems where the cost of evaluating the objective value of a solution is significantly greater than the time taken by the move operators (Marc and Jérôme, 2014; Heuberger, 2004; Nino-Ruiz et al., 2017; Malcolm and Klaus, 2002). The problem domains used in this study do not include such problems, and while it would be interesting to devise a general-purpose search method with the ability to effectively solve both traditional and inverse CO problems, the paper is focused on solving "traditional" CO problems and we leave such work for the concerns of researchers in those related fields.

This poses an even greater problem for the selection of move acceptance methods as a single high-level strategy must perform well across a diverse set of problems. A move acceptance method which has good cross-domain performance can therefore be reasonably expected to perform well for solving those *new* or *unknown* problems. In this paper, we investigate the effects that the choice of move acceptance method has on the cross-domain performance of such general-purpose search methods, based on a classification, by using a subset of metaheuristics known as local search metaheuristics. Some researchers might not agree with the given methodology used in this paper, however we believe that the advancements that are presented are still of value to the wider community since this experimentation can be generalised and extended to other local search frameworks and their hybrids (Talbi, 2013) such as Tabu Search (Glover and Laguna, 2013) where the inclusion of a memory for guiding the search on top of the different move acceptance methods can be compared. Similarly, an Iterated Local Search framework (Lourenço et al., 2003), which perturbs and then applies hill climbing on an incumbent solution, can be investigated using various move acceptance methods. In their paper, Lourenço et al. (2003) discuss the concept of applying such acceptance criteria, referring to *Better*, *RW*, and *LMSC* as such strategies. Restricting the scope of the study to investigate the effects of move acceptance on only local search metaheuristics has the benefit of eliminating other potential confounding factors from related areas including, but not limited to, for example, population size and the strategy to maintain population diversity in population-based metaheuristics, and the strategies to prohibit moves and tabu list length in Tabu Search.

The previous reviews on metaheuristics cover a variety of approaches, either conceptually, or in the context of specific problems (Blum and Roli, 2003; Bianchi et al., 2009; Blum et al., 2011; Beheshti and Shamsuddin, 2013; Hopper and Turton, 2001; Lewis, 2008; El-Sherbeny, 2010; Liao et al., 2011). In this study, we provide a taxonomy for local search metaheuristics embedding single-stage move acceptance methods. The influence of move acceptance on the performance of local search metaheuristics in the context of cross-domain search has not been done before. This is despite Özcan et al. (2008) suggesting that the choice of move acceptance method in a selection hyper-heuristic (as one classification of a cross-domain search method) has more effect on the (cross-domain) performance of hyper-heuristics compared to their embedded heuristic selection method. Hence, in this study, we perform a substantial number of experiments using a selected move acceptance method from each classification in our taxonomy to assess their cross-domain performance on 45 problem instances

from a total of 9 different problem domains (see Section 3 for more details).

A classification for local search metaheuristics based on their move acceptance method along with a summarised survey is given in Section 2. The experimental design and the details and configurations of each local search metaheuristic are given in Section 3. The results are discussed in Section 4, and the study is concluded in Section 5.

2. A Classification for Local Search Metaheuristics

In this section, a taxonomy is given for classifying local search metaheuristics based on the characteristics of their move acceptance methods. Local search metaheuristics are composed of two fundamental components; a *predefined* solution neighbourhood, and a move acceptance method. Here, an emphasis is put on predefined, since the strategy for applying move operator(s) is determined in advance and there is no discrimination between multiple operators if there is any. This contrasts with methods such as hyper-heuristics where (machine) learning techniques could be used for *selecting* the most appropriate move operator given the current search state. Similarly, Tabu Search is not considered in this study as a local search metaheuristic embedding a move acceptance strategy; Glover and Laguna (2013) have previously contrasted Tabu Search to single-point based search methods utilising a (move) acceptance criterion. A classification for local search metaheuristics is shown in Figure 1 and distinguishes them based on two features of their move acceptance methods as indicated by the dashed arrows. Firstly (left), the nature of the accept/reject decision, as indicated by the *acceptRejectDecision()* procedure in Line 6 of Algorithm 1, is considered which takes into account what the objective value of the candidate solution is compared to, and the resulting probability of acceptance that is returned. The second part of the classification (right) considers the nature of how the algorithmic parameters are set within the move acceptance method. The mechanism(s), if any, to update the settings of the parameters are usually employed during the procedures $process_1()$ and/or $process_2()$ as shown on Lines 5 and 10 of Algorithm 1 respectively.

2.1. Classification of the Accept/Reject Decision

The first part of the classification concerns whether the acceptance of a solution in any given move is deterministic or not. There are two fundamental mechanisms for accepting or rejecting non-improving solutions. On one hand, the acceptance mechanism accepts or rejects a solution outright by using a binary operator to compare the objective value of the candidate solution to some acceptance threshold value. These methods are what we call as *non-stochastic move acceptance methods*. On the other hand, the acceptance mechanism can accept a solution probabilistically, either directly by assigning a probability that the move acceptance should accept a solution, or indirectly by subjecting the parameter(s) affecting the value of the acceptance threshold to randomness. These methods are what we call as *stochastic move acceptance methods*.

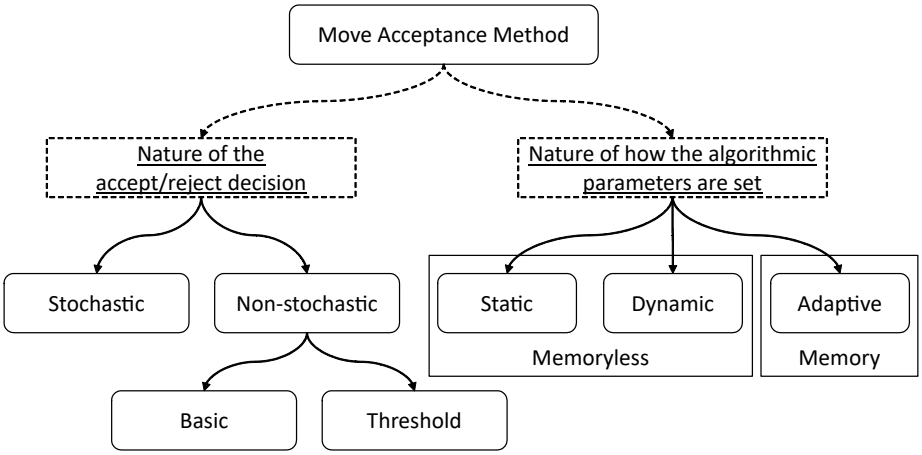


Figure 1: A taxonomy for single-stage local search metaheuristics based on the natures of the accept/reject mechanism and how the algorithmic parameters are set within their move acceptance method.

There are two strategies used to calculate the values used for the acceptance threshold in non-stochastic basic move acceptance methods. The taxonomy therefore includes a secondary level to further classify such methods. The first strategy, called as *non-stochastic basic*, reuses the objective values of previously encountered solutions for the accept/reject decisions. These either use the objective value of the current solution, or maintain a memory structure to decide on an appropriate value for the acceptance threshold at a given point during the search. The second strategy on the other hand, called as *non-stochastic threshold*, uses any real value as the value of the acceptance threshold. The values of the acceptance threshold are therefore not guaranteed to correspond to the objective value of any solution to the problem being solved.

The overall classification of a move acceptance method based on the nature of the accept/reject decision follows a trivial precedence. If the mechanism deciding whether to accept or reject a solution uses, or is influenced by parameters utilising, stochastic nature then it is *stochastic*, irrespective of the acceptance threshold. If the acceptance mechanism is non-stochastic but uses acceptance thresholds during any point of the search which are not objective values of previous solutions, then it is *non-stochastic threshold*. Otherwise, if the acceptance mechanism only uses objective values of previous solutions for the acceptance threshold, then it is *non-stochastic basic*. This can be summarised by the following precedence order relationship ($x \prec y : x$ precedes y).

$$\text{stochastic} \prec \text{non-stochastic threshold} \prec \text{non-stochastic basic}$$

2.2. Classification of the Algorithmic Parameter Setting

The second part to the classification concerns the nature of the parameter settings and the mechanisms used to control them. These are broken down into

three classifications, *static*, *dynamic*, and *adaptive*. A similar classification exists for parameter control in the context of multi-point search methods, namely evolutionary algorithms (EAs) such as those put forward by Eiben et al. (1999); Eiben and Smit (2011). Our taxonomy abstracts upon those classifications by only considering whether parameter tuning is based on what they refer to as a non-iterative control method (static) or an iterative control method (adaptive). Moreover, ours extends their view on iterative control methods by considering whether an algorithmic parameter setting mechanism relies on the overall number of iterations/time budget, along with the current step/elapsed time (dynamic), and/or acts upon search history (adaptive).

If a move acceptance method has multiple algorithmic parameter settings, then the final classification of the move acceptance method based on the nature of the parameter settings and the mechanisms used to control them is determined by the following precedence relation:

$$\text{adaptive} \prec \text{dynamic} \prec \text{static}$$

The overall classification based on the algorithmic parameter setting mechanisms can be determined using the descriptions below. This can be encapsulated as a Venn diagram as shown in Figure 2.

Static algorithmic parameter setting mechanisms. A move acceptance method is classified as having a *static* nature of how the algorithmic parameters are set if given the same candidate and current solutions, the acceptance threshold or acceptance probability would be the same irrespective of the current elapsed time or iteration count and search history.

Dynamic algorithmic parameter setting mechanisms. A move acceptance method is classified as having a *dynamic* nature of how the algorithmic parameters are set if given the same candidate and current solutions at the same current elapsed time or iteration count, the acceptance threshold or acceptance probability would be the same irrespective of search history.

Adaptive algorithmic parameter setting mechanisms. A move acceptance method is classified as having an *adaptive* nature of how the algorithmic parameters are set if given the same candidate and current solutions at the same current elapsed time or iteration count, the acceptance threshold or acceptance probability is not guaranteed to be the same as one or more components depend on search history.

2.3. Example Classification using Simulated Annealing

Here we provide a breakdown of the classification of Simulated Annealing (SA), a very popular local search metaheuristic. Starting with the nature of the accept/reject mechanism, Simulated Annealing would be classified as having a *stochastic* accept/reject mechanism. Despite accepting non-worsening moves outright, worse quality moves are accepted based on an acceptance probability

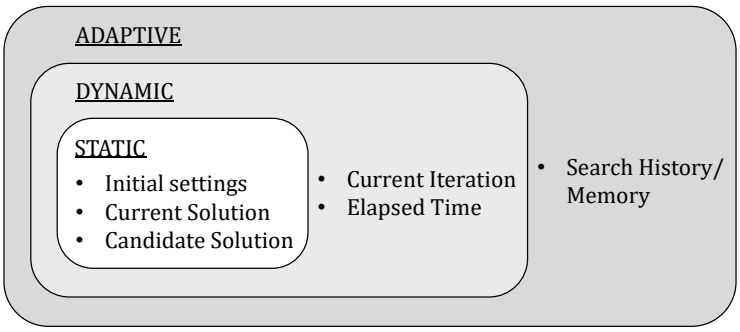


Figure 2: Venn diagram illustrating the information used by each algorithmic parameter setting mechanism to decide on the settings of the parameters of the move acceptance method.

determined by the Metropolis criterion. All non-reheating cooling schedules, such as geometric cooling and Lundy and Mees's cooling, contain mechanisms to reduce an internal temperature setting over *time*. These mechanisms are not effected by search history and depend only on the elapsed duration of the search. Such mechanisms are therefore *dynamic*. All other parameters, such as α for geometric cooling, and β for Lundy and Mees's cooling, remain fixed, hence these are *static*. The acceptance probability, given an elapsed duration, will therefore always be the same given the same current and candidate solution objective values. Hence, the classification for the nature of Simulated Annealing's algorithmic parameter setting mechanisms is *dynamic*. In conclusion, Simulated Annealing has a *dynamic stochastic* move acceptance method.

2.4. An Overview and Classification of Existing Methods

Local search metaheuristics are used as heuristic optimisation methods for solving a variety of computationally hard problems. Table 1 forms a succinct overview of the existing local search metaheuristics from the literature and classifies them based on the above taxonomy. The abbreviations used for each local search metaheuristic can be found in Table 2 of Appendix B.

As can be seen by classifying the existing methods, there are no local search metaheuristics which utilise dynamic algorithmic parameter setting mechanisms in combination with a non-stochastic basic acceptance strategy. The majority of the approaches tend towards those with adaptive algorithmic parameter setting mechanisms. This is unsurprising since it is known that parameter adaptation is one key prerequisite for improving the performance of metaheuristics. Selection hyper-heuristics are mostly made up of two key methods which are invoked successively: heuristic selection and move acceptance. Despite the emergence of hyper-heuristics as methods for solving the cross-domain search problem (being reusable and effectively applicable to multiple domains), there are no local search metaheuristics in the literature that have been developed specifically for solving this problem. Local search metaheuristics seem to have only been developed to complement the heuristic selection process in selection hyper-heuristics. This

Table 1: Succinct survey and classification of local search metaheuristics as used in the scientific literature. If a local search metaheuristic has been used with a different algorithmic parameter setting nature compared to their original descriptions, references of the earliest occurring paper using those alternative mechanisms are given in their respective column(s) and in parentheses. The bold entries indicate the approaches selected from each local search metaheuristic class used in our experiments. A key can be found in Table 2 of appendix B mapping the abbreviations of each local search metaheuristic to their given names.

	Static	Dynamic	Adaptive
Non-stochastic Basic	AM (Cowling et al., 2001) **IE** (Bilgin et al., 2007) OI (Cowling et al., 2001)	-	LA (Burke and Bykov, 2008) AILLA-F (Mısır et al., 2011b) **AILLA** (Mısır et al., 2012) SCHC (Bykov and Petrovic, 2016)
Non-stochastic Threshold	(**TA** (Sabar et al., 2013))	TA (Dueck and Scheuer, 1990) **GD** (Dueck, 1993) (FD (Sin and Kham, 2012))	(TA (Tarantilis et al., 2004)) (GD (Sinclair, 1993)) FD (Burke and Bykov, 2006) EGD (McCollum et al., 2009) LBTA (Tarantilis and Kiranoudis, 2002) BATA (Tarantilis et al., 2004) RRT (Dueck, 1993) ILTA (Mısır et al., 2009) **AILTA** (Mısır et al., 2010)
Stochastic	EMC (Ayob and Kendall, 2003) LMC (Ayob and Kendall, 2003) **NA** (Burke et al., 2010)	(EMC (Özcan et al., 2008)) **SA** (Kirkpatrick et al., 1983)	EMCQ (Ayob and Kendall, 2003) (SA (Ingber, 1989)) **SARH** (Connolly, 1992)[1] NLGD (Landa-Silva and Obit, 2008) AA (Burke et al., 2010)

finding is interesting as it is suggested in Özcan et al. (2008) that the choice of move acceptance method in a selection hyper-heuristic, which commonly takes the form of a local search metaheuristic, has more effect on the (cross-domain) performance of hyper-heuristics compared to the embedded heuristic selection method.

There are a variety of local search metaheuristics in the literature, almost evenly spanning the different natures of the accept/reject decision. The traditional methods, such as, Simulated Annealing (in 1983), Great Deluge (in 1993), and Record to Record Travel (also in 1993) are classified as either stochastic or non-stochastic threshold. Non-stochastic basic methods have more recently emerged with the earliest of such, excluding those simplistic local search metaheuristics, being Late Acceptance in 2008, and Step Counting Hill Climbing being introduced as recent as 2016.

3. Experimental Design

The performance and behaviours of eight local search metaheuristics are compared and contrasted with the aim of observing both the intra-domain and cross-domain effectiveness of different local search metaheuristics classified by their move acceptance methods based on the taxonomy given in the previous section. The taxonomy gives a total of 9 distinct classifications for local search metaheuristics however to date, those from the literature fall into only 8 of these. A single local search metaheuristic is thus chosen from each of these by choosing one whose move acceptance method has previously been *used* for solving the cross-domain search problem where appropriate, otherwise a well known method is chosen as illustrated in Table 1 in bold.

Each local search metaheuristic is evaluated across a total of nine minimisation problems, which to date are still or scientific interest, using the Hyper-heuristics Flexible Framework (HyFlex) (Ochoa et al., 2012). Six of these domains were used within the Cross-domain Heuristic Search Challenge (CHeSC) 2011[2]. These are the One-Dimensional Bin Packing Problem (BP) (Hyde et al., 2010b; Sridhar et al., 2017; Scheithauer, 2018), Permutation Flow Shop Problem (FS) (Vazquez-Rodriguez et al., 2010; Fernandez-Viagas et al., 2017, 2018), Personnel Scheduling Problem (PS) (Curtois et al., 2010; Ernst et al., 2004; Asta et al., 2016; Santos et al., 2016), Maximum Satisfiability Problem (SAT) (Hyde et al., 2010a; Morgado et al., 2013; Anstegui et al., 2016), Travelling Salesman Problem (TSP) (Applegate et al., 2011; Escario et al., 2015; Ezugwu et al., 2017), and the Vehicle Routing Problem (VRP) (Walker et al., 2012; Lin et al.,

[1]Note that SARH is a variant of SA where the temperature setting is directly increased based on a reheating mechanism. This is in contrast to other adaptive variants of SA such as VFR where the algorithmic parameters affecting the temperature setting are controlled to allow the temperature to increase/decrease continually over time, hence this distinction is made.

[2]The Cross-domain Heuristic Search Challenge 2011 `http://www.asap.cs.nott.ac.uk/external/chesc2011`

2014; Montoya-Torres et al., 2015). The remaining three domains were recently introduced (Adriaensen et al., 2015) to the HyFlex Framework as a HyFlex Extension (HyFlext). These are the 0-1 Knapsack Problem (KP) (Adriaensen and Ochoa, 2015a; Frville, 2004; Zhou et al., 2016; Lim et al., 2016), Max Cut Problem (MAC) (Etscheid and Rglin, 2014; Ben-Ameur et al., 2014; Zhou et al., 2015), and Quadratic Assignment Problem (QAP) (Adriaensen and Ochoa, 2015c; Dokeroglu and Cosar, 2016; Hafiz and Abdennour, 2016). The HyFlex framework is designed such that the search is performed over the search space of complete and feasible solutions. For each problem domain, 5 problem instances were chosen as those used in the CHeSC 2011 competition, and 5 characteristically differing instances each from the 3 HyFlext domains as summarised in Table 2 along with the respective move operator(s) and solution initialisation procedures used for each domain. No official documentation exists for the TSP problem domain. For information on the TSP problem, the reader is referred to (Bellmore and Nemhauser, 1968). The instances used by HyFlex are taken from TSPLIB. The objective function embedded in the HyFlex implementation is different from that used in TSPLIB and calculates the cost of a solution as the sum of the non-rounded Euclidean distances (rather than rounded integer values) between adjacent cities, including that between the first and last cities. Hence, solutions exist with better than optimal tour lengths compared to the results from TSPLIB. Each local search metaheuristic is evaluated 31 times for each problem instance where the termination criteria for each run (evaluation) is equal to 10 CHeSC 2011 competition minutes. This is equivalent to 415 seconds on our machine, as determined by the official benchmark tool[3], using an Intel Core i7-3820 processor at 3.60GHz with 16GB of memory running Windows 7 SP1 and Java 1.8.0_40-b26. The move operators and their settings were chosen such that each application made a *single* perturbation to the solution (with the exception of the PS problem domain due to its specialised problem representation meaning that the search space cannot be explored sufficiently using single perturbations).

3.1. Method of Analysis

In this study, we want to observe both the intra-domain (performance across multiple instances from the same domain) and cross-domain performance of the local search methods embedding the characteristically different (based on our taxonomy) move acceptance methods. The different objective functions used by each domain, and the variable objective value ranges of different problem instances poses an issue when comparing general-purpose search methods over multiple domains and even problem instances. In order to be able to compare the performances of the move acceptance methods, the results are normalised following the scheme used in Di Gaspero and Urli (2012), and as shown in Equation (1) where $f(s)$ is the result being normalised, and $f(s_{best})$ ($f(s_{worst})$)

[3]HyFlex benchmarking tool available online: http://www.asap.cs.nott.ac.uk/external/chesc2011/benchmarking.html

Table 2: Initialisation procedure, move operator(s), and problem instances used in each problem domain.

Domain	Initialisation	Operator(s)	(HyFlex ID) Instance(s)
Bin Packing (BP) (Hyde et al., 2010b)	Randomised first fit (Johnson et al., 1974)	Swap, Split, Destroy	(1) falkenauer/u1000-01 (9) testdual7/binpack0 (11) testdual10/binpack0 (ESICUP, 2011) (7) triples2004/instance1 (10) 50-90/instance1 (Hyde, 2011)
Flow Shop (FS) (Vazquez-Rodriguez et al., 2010)	Randomised NEH (Nawaz et al., 1983)	Swap	(1) 20x5/2, (3) 100x20/4, (8) 500x20/2, (10) 200x20/1, (11) 500x20/3 (Taillard, 1993)
Personnel Scheduling (PS) (Curtois et al., 2010)	Randomised Hill Climbing	New swap, Vertical swap, Random unassign	(5) Ikegami-3Shift-DATA1.2 (Ikegami and Niwa, 2003) (8) ERRVH-B, (9) MER-A, (10) BCV-A.12.1, (11) ORTEC01 (Curtois, 2009)
Maximum Satisfiability (SAT) (Hyde et al., 2010a)	Random binary string	Flip	(3) parity-games/instance-n3-i3-pp, (4) parity-games/instance-n3-i3-pp-ci-ce, (5) parity-games/instance-n3-i4-pp-ci-ce (CRIL, 2009) (10) jarvisalo/eq.atree.braun.8.unsat (CRIL, 2007) (11) highgirth/3sat/hg-3sat-v300-c1200-4 (Argelich et al., 2009)
Travelling Salesman Problem (TSP)	Random permutation	Swap	(0) pr299, (2) rat575, (6) d1291, (7) u2152, (8) usa13509 (Reinelt, 2008)
Vehicle Routing Problem (VRP) (Walker et al., 2012)	Randomised constructive heuristic	2-opt (mutation), Shift (mutation)	(1) Solomon/RC/RC207, (2) Solomon/RC/RC103, (5) Homberger/RC/RC2-10-1, (6) Homberger/R/R1-10-1, (9) Homberger/C/RC1-10-8 (SINTEF, 2011)
0-1 Knapsack Problem (KP) Adriaensen and Ochoa (2015a)	Most valuable item first	Pack random, Remove random	(0) 1K-15-12, (1) 2K-1-13, (3) 2K-5-17, (5) 5K-1-24, (8) 5K-5-28 Pisinger (2015)
Max Cut Problem (MAC) Adriaensen and Ochoa (2015b)	Greedy Initialisation	Swap random	(0) g3-8, (9) pm3-15-50 DIMACS (2015) (2) g14, (5) g22, (7) g55 Rinaldi (2015)
Quadratic Assignment Problem (QAP) Adriaensen and Ochoa (2015c)	Random	Swap random	(0) sko100a, (6) tai150b, (7) tai256c, (8) tho150, (9) wil100 Burkard et al. (1997)

is the best (worst) solution obtained by all algorithms over the same problem instance. The normalised results are thereby linearly scaled between 0 (best result) and 1 (worst result) for each problem instance.

$$f_{norm}(s) = \frac{f(s) - f(s_{best})}{f(s_{worst}) - f(s_{best})} \quad (1)$$

Lilliefors test was performed on the computational results to test for normality and showed that they do not always come from a normal distribution. The move acceptance methods in the results section are therefore compared using non-parametric statistical tests with Wilcoxon signed rank test being used to compare the performances of a pair of algorithms, and the Kruskal-Wallis one-way ANOVA test being used to compare the performance of more than two algorithms.

The local search metaheuristic with the best intra-domain performance was chosen as the one which obtained the lowest sum of normalised scores over all 5 instances from each respective domain. The results of each local search metaheuristic were compared within each domain using the aforementioned ANOVA test with Tukey's Honestly Significant Difference Procedure post-hoc analysis to find which algorithm (or group of algorithms) have the best intra-domain performance for each domain.

A general performance score is calculated as the sum of averaged normalised results, also referred to as a μ_{norm} score (Adriaensen et al., 2015), over all 45 problem instances to compare, at a high-level, the cross-domain performance of each local search metaheuristic across the 9 different problems.

For each domain, we discuss the behaviours and characteristics of the local search metaheuristics and the effects that these have for solving them. The diverse characteristics of the local search metaheuristics embedding the best performing move acceptance method for solving each domain are illustrated using progress plots recorded for the problem instance where the difference in performance of the respective move acceptance method is maximised when compared to the remaining methods. The intra-domain scores and the results of an ANOVA test performed for each problem instance are used to gain an insight into the features of the move acceptance methods which likely contribute to their cross-domain effectiveness.

3.2. Local Search Metaheuristic Configurations

The designs and parameter setting mechanisms of the local search metaheuristics are taken from the scientific literature where appropriate. The parameter settings of each local search metaheuristic, apart from those which have no parameters or are designed to be parameter-less (AILLA and AILTA) and therefore utilise their default settings, are tuned using a full factorial tuning process for move acceptance methods with 1 or 2 parameters, or by using the Taguchi orthogonal array design of experiments method (Roy, 2010) for 3 or more parameters. Parameter tuning is performed on each domain by arbitrarily choosing 2 instances from each domain. The final parameter settings used for each problem domain in each case are given in a table following each description.

3.2.1. Improving or Equal (IE)

IE uses a static non-stochastic basic move acceptance method. IE accepts a candidate solution as the current solution in the next stage if and only if the objective function value of the candidate solution is not worse than the current solution as shown in Equation (2).

$$s_{i+1} \leftarrow \begin{cases} s_i^{'} & f(s_i^{'}) \leq f(s_i) \\ s_i & \texttt{otherwise} \end{cases} \tag{2}$$

3.2.2. Adaptive Iteration Limited List-based Threshold Accepting (AILLA)

AILLA (Mısır et al., 2012) uses an adaptive non-stochastic basic move acceptance method. AILLA maintains a list of length L recording the objective values of the L best solutions found. Whenever a new best solution is found, the worst objective value is removed from the list and replaced with that of the new best solution. AILLA also incorporates a restart mechanism to reinitialise the current solution depending on several factors of the current search state and is explained in detail in (Mısır et al., 2012). AILLA accepts a candidate solution as the current solution in the next iteration, as shown in Equation (3), if its objective value is not worse than the current solution, or if it is not worse than the ith objective value in the list, as detailed in (Mısır et al., 2012), since the last re-initialisation, and a number of non-improving iterations have passed which is dynamically reduced from 10 to 5 as the search progresses. Note that by $s_{i^{th}best}$, we are referring to the parameter i in the paper for AILLA and not the current iteration. The implementation and parameter settings for AILLA were taken from (Mısır et al., 2012). All parameters and adaptation methods used are set to their default settings since AILLA was designed as a parameter-less move acceptance method for solving the cross-domain search problem. Moreover, it was used as the move acceptance component of the winning hyper-heuristic from the CHeSC 2011 competition which to date remains one of the best cross-domain search procedures.

$$s_{i+1} \leftarrow \begin{cases} s_i^{'} & f(s_i^{'}) \leq f(s_i) \vee (f(s_{i^{th}best}) \wedge w\_iterations \geq k) \\ s_{re\text{-}init} & \text{(see Mısır et al. (2012))} \\ s_i & \texttt{otherwise} \end{cases} \tag{3}$$

3.2.3. Threshold Accepting (TA)

TA (Dueck and Scheuer, 1990) uses a static non-stochastic threshold move acceptance method. TA accepts a candidate solution as the current solution in the next iteration if and only if the objective function value of the candidate solution is not worse than an acceptance threshold calculated as the sum of the objective function value of the *current* solution and a threshold parameter, T, as shown in Equation (4).

$$s_{i+1} \leftarrow \begin{cases} s_i^{'} & f(s_i^{'}) \leq f(s_i) + T \\ s_i & \texttt{otherwise} \end{cases} \tag{4}$$

TA has a single parameter, T, which defines how much worse than the current solution a candidate solution is allowed to be before being rejected. A static definition of TA from (Sabar et al., 2013) is used in this study, which they incorrectly identify as record-to-record travel. This variant was used for solving multiple problems including examination timetabling and the capacitated vehicle routing problem. As a general-purpose search method however, the setting for T must be set appropriately for the problem being solved. The objective function values of different problem domains, and even problem instances, return a different range of values meaning that a single setting would not be effective. The threshold parameter is therefore calculated as a factor (k) of the cost of the initial solution as shown in Equation (5). This approach for calculating a threshold setting has been seen in related local search metaheuristics such as AILTA (see below) albeit their settings are chosen adaptively.

The settings of k used in the experimentation, as derived from an exhaustive tuning process, are given in Table 3. Note that in some cases, a setting of $k = 0$ could have yielded improved results by reducing TA to IE. This however would have detracted from the nature of the TA move acceptance method and a minimum bound of 0.0005 was therefore imposed on k in the tuning process.

$$T = k \times f(s_0) \tag{5}$$

Table 3: Values of k that were used to calculate the threshold parameter T for each problem domain for the Threshold Accepting move acceptance method.

Parameter	BP	FS	PS	SAT	TSP	VRP	KP	MAC	QAP
k	0.0005	0.0005	2.0000	0.0010	0.0010	0.0500	0.1500	0.1000	0.0005

3.2.4. Great Deluge (GD)

GD (Dueck, 1993) uses a dynamic non-stochastic threshold move acceptance method. A dynamic time-based version of GD from (Özcan et al., 2010) is used in this study. GD accepts a candidate solution as the current solution if the objective function value of the candidate solution ($f(s_i')$) is not worse than the current solution ($f(s_i)$), or if it is not worse than a threshold value (τ_i) which is linearly decreased in time between the objective function value of the initial solution ($f(s_0)$) and some target value ($qualityLB$), as shown in Equation (6) where the acceptance threshold is calculated as in (Özcan et al., 2010) shown in Equation (7).

$$s_{i+1} \leftarrow \begin{cases} s_i' & f(s_i') \leq \max(f(s_i), \tau_i) \\ s_i & \texttt{otherwise} \end{cases} \tag{6}$$

$$\tau_i = qualityLB + (f(s_0) + qualityLB) \times \left(1 - \frac{T_{elapsed}}{T_{total}}\right) \tag{7}$$

A target value of 0 does not make sense in the context of a general-purpose search method since the objective function value ranges vary between different problem domains. Moreover, some problem domains may have non-positive

objective value ranges causing the threshold to *increase* over time. The objective values of the optimal, where known, are therefore used and the respective settings are summarised in Table 4.

Table 4: Target values (*qualityLB*) used by Great Deluge for solving each problem instance. Bold styling is used to show where known optimal values are known and hence used as the target value. Bin Packing problem instances use a target value of 0 equal to the lower bound of its objective function. Flow Shop instances use the lower bound of the problems' makespan. All other values are best known solution fitnesses.

Domain	(Instance, Target Value) Pairs				
BP	(1, 0.00)	(7, 0.00)	(9, 0.00)	(10, 0.00)	(11, 0.00)
FS	(1, 6099.00)	(3, 5979.00)	(8, 26353.00)	(10, 10979.00)	(11, 26320.00)
PS	(5, **3.00**)	(8, 3121.00)	(9, 8814.00)	(10, 1294.00)	(11, **270.00**)
SAT	(3, **0.00**)	(4, **0.00**)	(5, **1.00**)	(10, **1.00**)	(11, **7.00**)
TSP	(0, **48191.00**)	(2, **6773.00**)	(6, **50801.00**)	(7, **64253.00**)	(8, **19982859.00**)
VRP	(1, **8617.10**)	(2, **3332.80**)	(5, 142478.95)	(6, 50278.50)	(9, 135540.07)
KP	(0, **-104046.00**)	(1, **-1263861.00**)	(3, **-431363.00**)	(5, **-4417737.00**)	(8, **-1530536.00**)
MAC	(0, -41684814.00)	(2, -3064.00)	(5, -13359.00)	(7, -10299.00)	(9, -3014.00)
QAP	(0,0 152002.00)	(6, 441786736.00)	(7, 43849646.00)	(8, 7620628.00)	(9, 273038.00)

3.2.5. Adaptive Iteration Limited Threshold Accepting (AILTA)

AILTA (Mısır et al., 2010) uses an adaptive non-stochastic threshold move acceptance method. It is similar to AILLA but with two key differences. One being that it calculates a threshold value rather than reusing objective values of previously visited solutions, and the other that it considers only the current best solution's objective value rather than maintaining a list of L best solution values, instead opting to adapt an acceptance threshold factor, ϵ, coming from a predefined list of settings. AILTA accepts a candidate solution as the current solution in the next iteration if the objective value of the candidate solution is not worse than the current solution, or if a certain number of consecutive rejected moves have occurred and the objective value of the candidate solution is less than an acceptance threshold calculated as a factor of the cost of the best solution found so far as shown in Equation (8).

$$s_{i+1} \leftarrow \begin{cases} s_i^{'} & f(s_i^{'}) \leq f(s_i) \vee \\ & w\_iterations \geq k \wedge f(s_i^{'}) \leq f(s_{best}) + |f(s_{best}) \times \epsilon| \\ s_i & \texttt{otherwise} \end{cases} \tag{8}$$

The original equation for calculating the threshold from (Mısır et al., 2010) works only for problems whose objective functions return non-negative values. For this study, the equation was therefore modified from $f(s_{best}) + (1 \times \epsilon)$ to $f(s_{best}) + |f(s_{best}) \times \epsilon|$ to allow it to work over all objective value ranges.

The ϵ parameter controls how much worse a solution is allowed to be before it is rejected, and k determines how many continuous rejected moves should elapse before switching from the improving or equal acceptance strategy to the threshold accepting strategy. In this study, the setting for k was the same as used in (Mısır et al., 2010) which was 100. The ϵ parameter is incremented

according to a number of consecutive non-improving moves exceeding a parameter *max_iter* and reset to its initial value upon accepting an improving move. In that paper, this was set to 5000 however it was used exclusively for the homecare scheduling problem taking approximately 62500 iterations. Due to the variable iteration count encountered in cross-domain search, if the total number of iterations does not exceed 62500, then *max_iter* is set proportional to an estimated number of iterations for each problem instance based on pre-experimental analysis as shown in Equation (9) rounded to the nearest integer value. Upon exceeding this limit, ϵ is incremented from its initial setting of 0.003 by 0.001 up to a maximum setting of 0.010.

$$max\_iter = min\left(\frac{5000 \times total\_iterations}{62500}, 5000\right) \tag{9}$$

3.2.6. Naïve Acceptance (NA)

NA (Burke et al., 2010) uses a static stochastic move acceptance method. NA accepts a candidate solution as the current solution in the next iteration if the objective function value of the candidate solution is strictly better than the current solution, else it accepts the candidate solution with a fixed probability as shown in Equation (10). The only parameter in NA is the probability to accept equal or worse quality solutions. Previous studies mostly fixed this value to be 0.5 as is the case in (Burke et al., 2010) which evaluated NA for cross-domain search, hence we use this value in this study.

$$s_{i+1} \leftarrow \begin{cases} s_i^{'} & f(s_i^{'}) < f(s_i) \vee random \in [0,1] < 0.5 \\ s_{i+1} & \texttt{otherwise} \end{cases} \tag{10}$$

3.2.7. Simulated Annealing (SA)

SA (Kirkpatrick et al., 1983) uses a dynamic stochastic move acceptance method. SA accepts a candidate solution if its quality is better than or equal to the cost of the current solution, or if a random number in the range $[0, 1]$ is less than some probability P determined by the metropolis criterion (Metropolis et al., 1953) as shown in Equation (11). The metropolis criterion has two parameters, one being the signed difference between the current and candidate solution, $\delta = f(s') - f(s)$, and the other being a system temperature, T, determined by an accompanying annealing schedule. Annealing schedules from the literature include linear cooling, geometric cooling and Lundy and Mees cooling (Lundy and Mees, 1986). The components and settings of SA used in this paper, as well as the procedure for calculating the system temperature, are those from (Jackson et al., 2017) using the θ_{DL} settings for the initial and final temperatures. In this study, an extended set of problems are used and hence the settings are given in Table 5.

$$s_{i+1} \leftarrow \begin{cases} s_i^{'} & f(s_i^{'}) \leq f(s_i) \vee random \in [0,1] < e^{-\delta/T} \\ s_i & \texttt{otherwise} \end{cases} \tag{11}$$

Table 5: Values of χ_0 and t_{final} that were used for each problem domain in the Simulated Annealing local search metaheuristic.

Parameter	BP	FS	PS	SAT	TSP	VRP	KP	MAC	QAP
χ_0	5%	10%	90%	5%	10%	70%	30%	90%	70%
t_{final}	10^{-5}	10^{-3}	10^{4}	10^{-1}	10^{0}	10^{-3}	10^{3}	10^{-1}	10^{0}

3.2.8. Simulated Annealing with Reheating (SARH)

SARH (Connolly, 1992) uses an adaptive stochastic move acceptance method. SARH extends upon the traditional definition of Simulated Annealing by periodically increasing the temperature of the system during the search to either prevent it from becoming stuck or to escape from local optima. There are multiple approaches in the literature for deciding the value of the reheated temperature setting, how the cooling schedule operates after reheating, and when a reheat should occur. Such ways in which the reheated temperature is set includes a function of the temperature when the maximum specific heat occurs (Abramson et al., 1999), the initial temperature setting (Sim, 2011), and some factor of the temperature setting when the best solution was found, for example $1.0 \times T_{best}$ or $2.0 \times T_{best}$ in (Connolly, 1992) and (Anagnostopoulos et al., 2006) respectively. After performing a reheat, the cooling schedule can either be left to operate as is, decreasing the temperature over time (Abramson et al., 1999), or disabled such that the temperature is fixed at the reheat temperature for the remainder of the search (Connolly, 1992). Note that reheating is not the same as re-annealing (Ingber, 1989) where the parameters affecting the temperature setting are controlled to gradually decrease and increase the temperature over time depending on the search features. Strategies for deciding when to perform the reheat include doing so when when the maximum specific heat occurs in the system (Abramson et al., 1999), by keeping track of when the last improvement was made, reheating the temperature after a predefined number of continuous non-improving moves have occurred (Connolly, 1992), and after a predefined number of moves since the best solution was last improved (Anagnostopoulos et al., 2006).

In this study, we use the earliest example of such mechanism for deciding when to perform reheating as it can be seen in other adaptive approaches within the literature to signal an adaptation event such as in AILLA and AILTA. For the reheat temperature, we chose to use the approach used in (Anagnostopoulos et al., 2006) where the temperature is set to twice that when the best solution was found as preliminary testing showed this to be more effective. After performing a reheat, the cooling schedule is allowed to continue for the remainder of the search. SARH as used in this study contains three parameters, k, $wait\_time$, and T_{final}. The initial temperature is calculated as $f(s_0) \times k$, α is calculated initially and upon each reheat as T_{final}/T_0 and $T_{final}/(2 \times T_{best})$ respectively. The $wait\_duration$ is calculated as a product of the computational time budget and the $wait\_time$ parameter where $wait\_time \in [0, 1]$, exclusive and inclusive. The settings for each domain are given in Table 6.

Table 6: Values of k, $wait\_time$ and t_{final} that were used for each problem domain in the Simulated Annealing with Reheating local search metaheuristic.

Parameter	BP	FS	PS	SAT	TSP	VRP	KP	MAC	QAP
k	0.8	0.6	1.0	0.2	0.2	0.2	0.2	0.2	0.2
$wait\_time$	0.2	0.1	0.1	0.1	1.0	0.2	0.1	0.1	0.1
t_{final}	10^{-5}	10^{-4}	10^{4}	10^{-1}	10^{0}	10^{-3}	10^{3}	10^{-3}	10^{-1}

The mechanism for move acceptance is the same as that used by SA in Equation (11). The adaptation procedure for dealing with reheating of the system temperature is shown in Algorithm 2 and an updated version of the cooling schedule to deal with the reheating capability and time-based termination criterion is shown in Equation (12).

Algorithm 2: $process_2()$ for Simulated Annealing with Reheating.

```
if move was accepted then
    if f(s').isBetterThan(f(s)) then
        time_reheat ← wait_duration + time_elapsed;
    end
    if f(s').isBetterThan(f(s_best)) then
        t_best ← t_i;
    end
else if time_elapsed > time_reheat then
    t_0 ← t_best × 2;
    time_reheat ← wait_duration + time_elapsed;
    time_previous_reheat ← time_elapsed;
    α ← t_final/t_0;
return s_best;
```

$$T = T_0 \times \alpha^{\frac{time\_elapsed - time\_previous\_reheat}{time\_total}} \tag{12}$$

4. Results

This section, covering the results and analysis of the local search metaheuristics embedding different move acceptance methods, is structured as follows. Firstly, the intra-domain results are discussed and the progress plots of the best general move acceptance method are examined for each of the nine problem domains forming Sections 4.1 to 4.9. A summary of the results highlighting the best move acceptance methods to use for solving each problem under the stochastic local search framework is given in Section 4.10, and the cross-domain results are discussed in Section 4.11. The intra-domain and cross-domain scores are given in Table 7 and boxplots of the normalised results from all 5 instances

per domain are given in Figure 3 and Figure 4. Results from the Kruskal-Wallis one-way ANOVA tests performed for each problem domain can be found in Table 8. The non-normalised results for each problem instance from the experimentation can be found in Table 1 of Appendix A. Finally, observations are made in relation to the progress plots and acceptance stats for the best move acceptance methods for solving each problem in Section 4.12. The progress plots and results from this analysis can be found in Figure 5, and in both Table 9 and Table 10 respectively.

Table 7: Intra-domain scores for each local search metaheuristic over the 9 problem domains, calculated as explained in Section 3, with the best general-purpose method for each problem domain stylised bold. The final column shows the cross-domain score for each move acceptance method, calculated as the sum of intra-domain scores.

	BP	FS	PS	SAT	TSP	VRP	KP	MAC	QAP	Cross-domain
IE	**0.13**	**0.52**	3.62	0.74	2.05	3.60	5.00	1.38	0.64	17.67
AILLA	1.26	0.56	2.53	0.10	2.25	3.47	5.00	0.45	0.30	15.91
TA	3.30	2.15	**0.55**	0.74	2.69	**1.94**	3.09	4.47	1.13	20.04
GD	0.56	0.71	1.67	0.12	**1.58**	3.38	5.00	**0.06**	0.07	13.15
AILTA	1.16	2.52	3.61	0.74	2.82	2.74	4.85	0.22	0.40	19.05
NA	4.60	3.88	0.58	4.83	3.17	2.00	1.23	4.49	4.83	29.61
SA	3.28	0.63	0.60	0.14	1.91	2.09	0.92	0.30	0.23	**10.10**
SARH	3.25	0.74	0.65	**0.09**	2.72	1.97	**0.69**	0.07	**0.04**	10.22

Table 8: Kruskal-Wallis One-way ANOVA comparing the performance of all local search metaheuristics for each *problem domain* with n_0 that all results are from the same distribution at CI = 95%. The values are the mean ranks (lower is better) of the aforementioned test with the best local search metaheuristic, and those which do not statistically significantly differ from the best, for each domain being stylised bold.

Problem	IE	AILLA	TA	GD	AILTA	NA	SA	SARH	$\chi^2(7)$	p
BP	**96.9**	478.2	868.7	257.9	441.1	1109.4	853.8	858.1	1051.1	1.07×10^{-222}
FS	**348.5**	**379.3**	759.9	488.6	951.7	1102.1	**430.2**	503.7	677.9	4.05×10^{-142}
PS	998.4	809.5	**380.5**	603.6	997.4	**374.8**	**394.2**	**405.8**	648.0	1.13×10^{-135}
SAT	848.3	**282.4**	848.3	**331.0**	848.3	1163.0	386.7	**256.0**	1010.4	6.89×10^{-214}
TSP	505.7	574.3	712.3	**355.0**	759.6	873.1	**455.9**	728.0	261.1	1.19×10^{-52}
VRP	859.7	824.8	**445.8**	800.6	644.2	**457.4**	**481.7**	**450.0**	286.9	3.76×10^{-58}
KP	953.5	953.5	506.65	953.5	858.1	**285.8**	**275.6**	**177.3**	1113.0	4.65×10^{-236}
MAC	823.1	614.1	1081.1	**172.7**	456.1	1089.9	541.1	**186.1**	1083.3	1.23×10^{-229}
QAP	837.1	549.5	865.1	**213.9**	684.7	1163.0	509.0	**141.7**	987.8	5.18×10^{-209}

4.1. Bin Packing

The best move acceptance method for solving Bin Packing (BP) problems according to the intra-domain scores was IE. The results of performing an ANOVA test on the normalised results of all BP instances shows that IE is the best general method with a mean rank of 96.9 and that the results of the move acceptance methods do not come from the same distribution. This means that IE must perform significantly better than at least one other move acceptance method. Post-hoc analysis shows that IE performed significantly better than all other methods with the least significant difference being between IE and

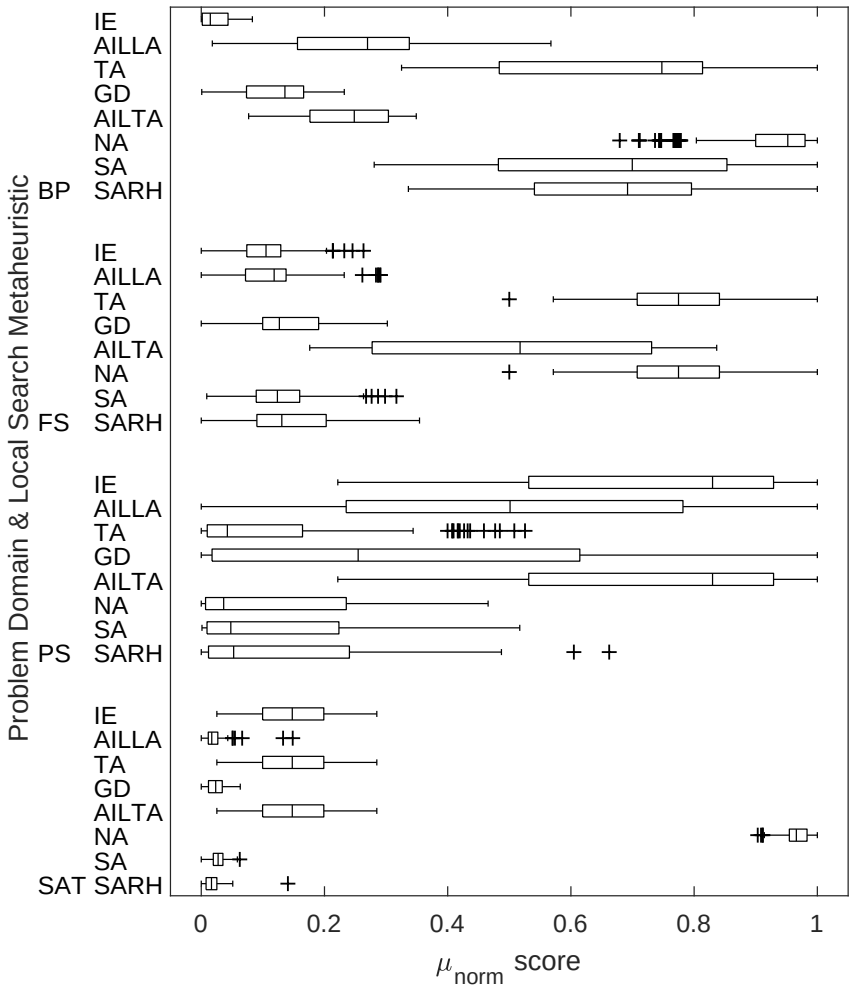


Figure 3: μ_{norm} scores obtained by each local search metaheuristic over all 31 trials for all 5 instances for each of the Bin Packing, Flow Shop, Personnel Scheduling, and Maximum Satisfiability problem domains.

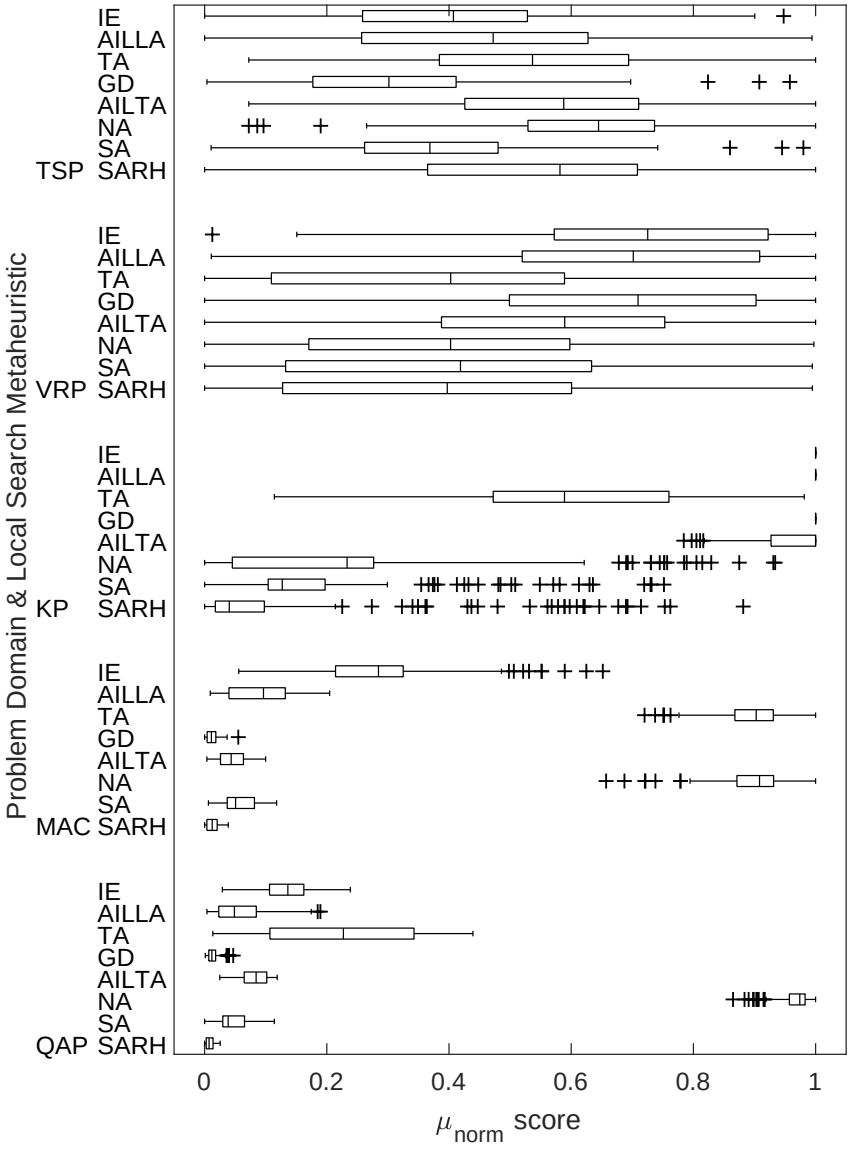


Figure 4: μ_{norm} scores obtained by each local search metaheuristic over all 31 trials for all 5 instances for each of the Travelling Salesman, Vehicle Routing, Knapsack, Max Cut, and Quadratic Assignment problem domains.

GD with a p-value of 1.96×10^{-3}. The results show that move acceptance methods with a stochastic accept/reject nature perform inferior to those with non-stochastic mechanisms. The TA method is however an exception to this rule due to the high setting of the threshold parameter. Moreover, the nature of the algorithmic parameter setting does not influence the overall performance of the local search metaheuristics. The Improving or Equals move acceptance method in this case performs better on average than all other move acceptance methods over all the 5 instances. The progress plot for Bin Packing, shown in Figure 5a, shows IE continually improving upon the current solution and finding the best solution at the last step. IE and GD have similar intra-domain performance due to the accept equal or improving moves mechanism utilised by GD when the acceptance threshold falls below the cost of the current solution. In conclusion, for solving BP problems under the given framework, it does not appear to be necessary to employ a strategy accepting moves of worsening quality.

4.2. Flow Shop

The best move acceptance method for solving Flow Shop (FS) problems according to the intra-domain scores was IE. The results of performing an ANOVA test on the normalised results of all FS instances shows that IE is the best general method and that AILLA and SA do not perform significantly different from IE. Hence, IE, AILLA, and SA are all able to solve FS problems equally as well. The results show that the effects of the accept/reject nature on the performance of the local search metaheuristic for solving Flow Shop problems is not clear cut. In general, non-stochastic basic move acceptance methods perform slightly better on average across all 5 instances with intra-domain scores of 0.52 and 0.56, compared to the next best method with a score of 0.63. Aside from the non-stochastic basic methods, those with static algorithmic parameter setting natures perform much worse than their dynamic and adaptive counterparts. The fixed set of threshold factors (ϵ), whose settings are too high, chosen from by AILTAs adaptation mechanism leads to its inferior performance. Figure 5b shows the progress plot for IE solving the Flow Shop problem. IE accepts improving moves throughout the search; however, the search landscape has many shoulder/plateau regions. Since AILLA does not accept worse moves until after a period of non-improving moves are accepted, its characteristics and performance is similar to IE. Furthermore, the settings for SA promote a rapid decrease in its temperature parameter meaning that few to no worse moves are accepted after the initial stages of the search. The progress plot also shows that while the best solution is found near to the end of the search, there is a large proportion of the search after this where the solution is not improved. IE could solve some instances the best whilst AILLA was the best at solving other instances. This is due to the presence of large shoulder regions in the search landscape allowing AILLA and SA to escape those regions. In summary, local search metaheuristics solving different instances of FS problems benefit from either accepting no to very few worse moves. The plateau regions present suggest that the inclusion of an accept only improving moves mechanism or tabu list strategy may be helpful in avoiding becoming stuck on such landscape features.

4.3. Personnel Scheduling

The best move acceptance method for solving Personnel Scheduling (PS) problems according to the intra-domain scores was TA. The results of performing an ANOVA test on the normalised results of all PS instances would suggest that NA is the best general method, performing slightly better than TA, and that TA, SA, and SARH do not perform significantly different from NA. The results would suggest that a local search metaheuristic needs to use a move acceptance method with a stochastic accept/reject nature; however, TA performs equally as well. The non-stochastic basic and non-stochastic threshold move acceptance methods did not perform well, apart from TA which has a high τ setting. The progress plot for Personnel Scheduling, shown in Figure 5h, shows TA accepting most, if not all, worse moves. In addition to TA, the stochastic methods also use parameter settings which cause them to accept most, if not all, worse moves; hence, the intra-domain performances of all four methods are similar. Furthermore, the nature of the algorithmic parameter setting mechanisms did not have much effect on the local search metaheuristic's intra-domain performance. The Personnel Scheduling problem is the only problem that needs to utilise specialised move operators. These operators appear to guide the search finding the best solution close to the end of the search, despite no or little guidance form the move acceptance methods which perform the best. In conclusion, a local search metaheuristic simply needs a naïve move acceptance strategy accepting most or all worse moves for solving such problems under the given framework.

4.4. Maximum Satisfiability

The best move acceptance method for solving Maximum Satisfiability (SAT) problems according to the intra-domain scores was SARH. The results of performing an ANOVA test on the normalised results of all SAT instances shows that SARH is the best general method, and that AILLA and GD are equally as good at solving SAT problems. The results show that the nature of the algorithmic parameter setting mechanism has a strong effect on the effectiveness of the move acceptance method to solve MAX-SAT problem instances with the move acceptance methods utilising a static algorithmic parameter setting mechanism performing poorly on this problem. The nature of the accept/reject decision did not have much of an effect on the intra-domain performance of algorithms to solve SAT problems. AILTA is the only method to break this pattern for the reasons discussed in previous problem domain results. The settings of τ in this case were too small. AILTA could therefore not accept worse moves, leading to its inferior performance and similarity to IE. Neither IE, TA, nor AILTA could accept worse moves, and NA accepted far too many worse moves, in all cases resulting in poor overall performance. The Wilcoxon Signed Rank test results shows that depending on the instance they are solving, SARH and AILLA interchangeably perform significantly better than each other. The progress plot, shown in Figure 5c, shows SARH solving a Maximum Satisfiability problem. The behaviour shown is characteristic of Simulated Annealing algorithms, initially accepting many worse moves, and reducing them over time to converge on

a good quality solution towards the end of the search. In this scenario, SARH behaves the same as SA (without reheating). Moreover, the setting of 1.0 for the *wait_time* parameter used for solving this problem effectively disabled the reheat mechanism. The lowest initial temperature setting is preferred by both SA and SARH reflecting the advantages of both temperature setting strategies, and suggesting that the different initial temperature setting mechanism is the cause of the improved performance of SARH over SA for solving these problems. In conclusion, a strategy to accept worse moves which reduces over time is needed for solving Maximum Satisfiability problems, and this is characteristic of both SA methods, GD, and AILLA. Furthermore, it is evident that Simulated Annealing algorithms are extremely sensitive to initial temperature settings and different strategies to choose them can significantly affect the algorithms overall performance.

4.5. Travelling Salesman Problem

The best move acceptance method for solving Travelling Salesman (TSP) problems according to the intra-domain scores was GD. The results of performing an ANOVA test on the normalised results of all TSP instances shows that GD is the best general method, with SA performing equally as good at solving TSP problems. The results also show that there is no discernible difference in performance between the move acceptance methods utilising different natures of the accept/reject decision. Furthermore, the nature of the algorithmic parameter setting mechanisms neither had much effect on the overall performance of the move acceptance method to solve the TSP problem instances. TA, AILTA, NA, and SARH performed worse in general than other methods due to the acceptance of too many worse moves. With the same reasons as the Maximum Satisfiability problem, SARH used the lowest setting for k, and 1.0 was used for the *wait_time*. The settings for TA and AILTA were also too high with the same effect. The target threshold parameter for GD was set based on the known optimal solution costs for each instance. Figure 5i provides the progress plot for GD solving a TSP problem. It shows the current solution cost closely following the current threshold setting up to about 1.1×10^7 iterations where the solution no longer improves at the same rate as the acceptance threshold. At this point, GD descends into IE acceptance. During this stage, GD accepting small and frequent worse moves, improving the solution over time at the same rate as the acceptance threshold decreases. Furthermore, the initial temperature setting for SA was small at $\chi_0 = 10\%$ and descending to a final temperature much smaller than the optimal solution values. The results of performing Wilcoxon Signed Rank test shows the best method(s) for solving each instance differed between IE, AILLA, GD, and SARH; however, combined with the results of the Kruskal-Wallis one-way ANOVA test, it is evident that GD and SA are able to perform consistently well across all the instances. In conclusion, an acceptance strategy needs to accept worse moves with small move deltas in a controlled manor, gradually improving the solution over time to best solve TSP problems. Acceptance of too many worse moves proves detrimental to performance and

not accepting worse moves, as is the case with IE, leads to an acceptable but inferior intra-domain performance.

4.6. Vehicle Routing Problem

The best move acceptance method for solving Vehicle Routing (VRP) problems according to the intra-domain scores was TA. The results of performing an ANOVA test on the normalised results of all VRP instances shows that TA is the best general method, with all stochastic methods performing equally as well. The results show that there is no that there is no discernible difference between the performances of each move acceptance method using different algorithmic parameter setting mechanism natures. The nature of the accept/reject decision has the most significant effect on the algorithm's performance with stochastic methods outperforming both non-stochastic variants. In this case, TA, performing the best, was tuned to utilise a threshold setting high enough to allow all worsening moves to be accepted which appeared to benefit this domain. The progress plot, shown in Figure 5f and enhanced in Figure 5g, illustrates the acceptance of all improving, equal, and worse moves by utilising a comparatively high threshold setting, complementing an effective search initially. Soon after finding a reasonable best solution, the search stagnates due to the lack of intensification in the TA move acceptance method. In conclusion, an acceptance strategy needs to accept most worse moves to effectively solve VRP problems, and the introduction of dynamic or adaptive parameter setting mechanisms has minimal effect on its performance.

4.7. Knapsack Problem

The best move acceptance method for solving 0-1 Knapsack problems (KP) according to the intra-domain scores was SARH. The results of performing an ANOVA test on the normalised results of all KP instances shows that SARH is the best general method, with TA and SA performing equally as well. The nature of the accept/reject decision clearly has a significant effect on the effectiveness of a local search metaheuristic with all stochastic methods forming equally the best and significantly better than non-stochastic methods. In general, the nature of the algorithmic parameter settings slightly effected the performance, albeit insignificant. Figure 5d provides the progress plot for SARH solving an instance of the KP problem. In this case, the effectiveness of SARH comes from the initial temperature setting mechanism rather than the reheat mechanism itself, illustrating the sensitivity of this parameter for SA based move acceptance methods. Initially, many worse moves are accepted and the current best solution slowly improves over time for the first 2/3rds of the search. Thereafter, the solution rapidly improves to find the best solution at the end of the search. Evidently, acceptance of many worse moves enhances the effectiveness of the search process. It should be noted that the greedy initialisation procedure used in this domain means that it is an absolute requirement that moves worse than the initial solution are accepted under such framework as the search starts from a local optimum. In conclusion, an acceptance strategy needs to accept most worse moves to effectively solve KP problems under the current setup.

4.8. Max Cut Problem

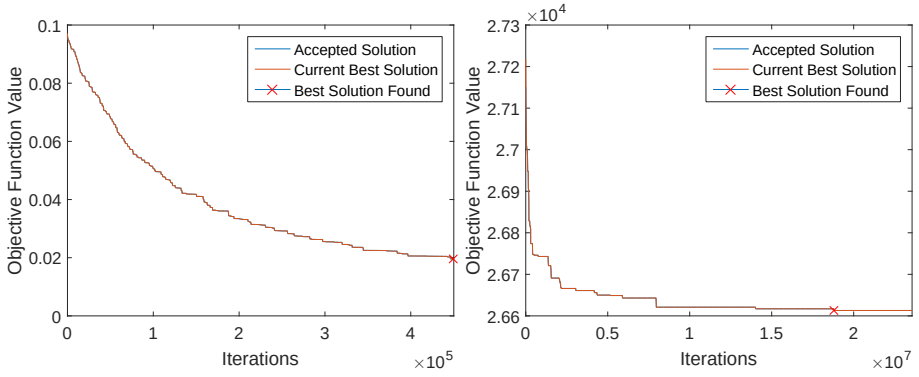
The best move acceptance method for solving Max Cut problems (MAC) according to the intra-domain scores was GD. The results of performing an ANOVA test on the normalised results of all MAC instances shows that GD is the best general method, with SARH performing equally as well. The nature of the algorithmic parameter settings had a marked effect on the effectiveness of local search metaheuristics for solving this problem. The nature of the accept/reject decision did not have as much of an effect, and in fact, the success of a move acceptance method was down to the acceptance of few, but some, worse moves. IE, accepting no worse moves, did not perform well as well as NA which accepts 50% of worse moves. GD in comparison accepted just 5.7% of worse moves. The progress plot, shown in Figure 5j, shows the current best solution improving at the same rate as the acceptance threshold and finding the best solution at the end of the search. Infrequent acceptance of worse moves also aids the search to prevent it from becoming stuck in local optima. In conclusion, an acceptance strategy needs to accept some worse moves to efficiently solve MAC problems but a careful balance needs to be achieved to prevent becoming stuck in local optima, or from performing a random walk over the search space.

4.9. Quadratic Assignment Problem

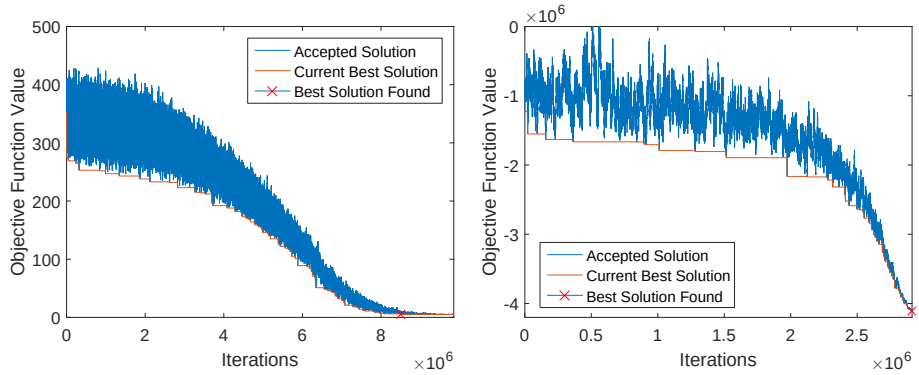
The best move acceptance method for solving Quadratic Assignment (QAP) problems according to the intra-domain scores was SARH. The results of performing an ANOVA test on the normalised results of all QAP instances shows that SARH is the best general method, with GD performing equally as well. Similarly with MAC problems, the nature of the algorithmic parameter settings had a marked effect on performance whereas the nature of the accept/reject decision did not. Again, success of the search relied on acceptance of some, but not too many, worse moves. Figure 5e provides the progress plot for a characteristic trace of a Simulated Annealing algorithm. Towards the end of the search, a reheat phase is activated allowing an improved solution to be found. In conclusion, similarly with solving MAC problems, an acceptance strategy needs to accept some worse moves to efficiently solve QAP problems but a careful balance needs to be achieved to prevent becoming stuck in local optima, or from performing a random walk over the search space.

4.10. Summary of Intra-domain Results

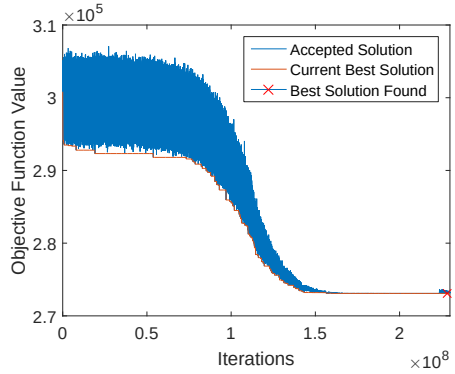
To observe the best move acceptance method(s) for solving each type of problem, a Wilcoxon Signed Rank test is performed between the best general move acceptance method, as determined by the best intra-domain score, and the remaining methods for each problem using the normalised set of results from all 5 instances. There are some domains where a single move acceptance method performs statistically significantly better than all other methods, and some domains where multiple move acceptance methods perform not significantly different from the best general method, hence a set of such methods are said to perform the equally as well. Bin Packing (IE), Travelling Salesman



(a) (b)



(c) (d)



(e)

Figure 5: Objective function value traces of accepted solutions for: IE solving (a) instance 11 of the Bin Packing problem and (b) instance 11 of the Flow Shop problem, and SARH solving (c) instance 4 of the MAX-SAT problem, and (d) instance 5 of the 0-1 Knapsack Problem, and (e) instance 9 of the Quadratic Assignment problem.

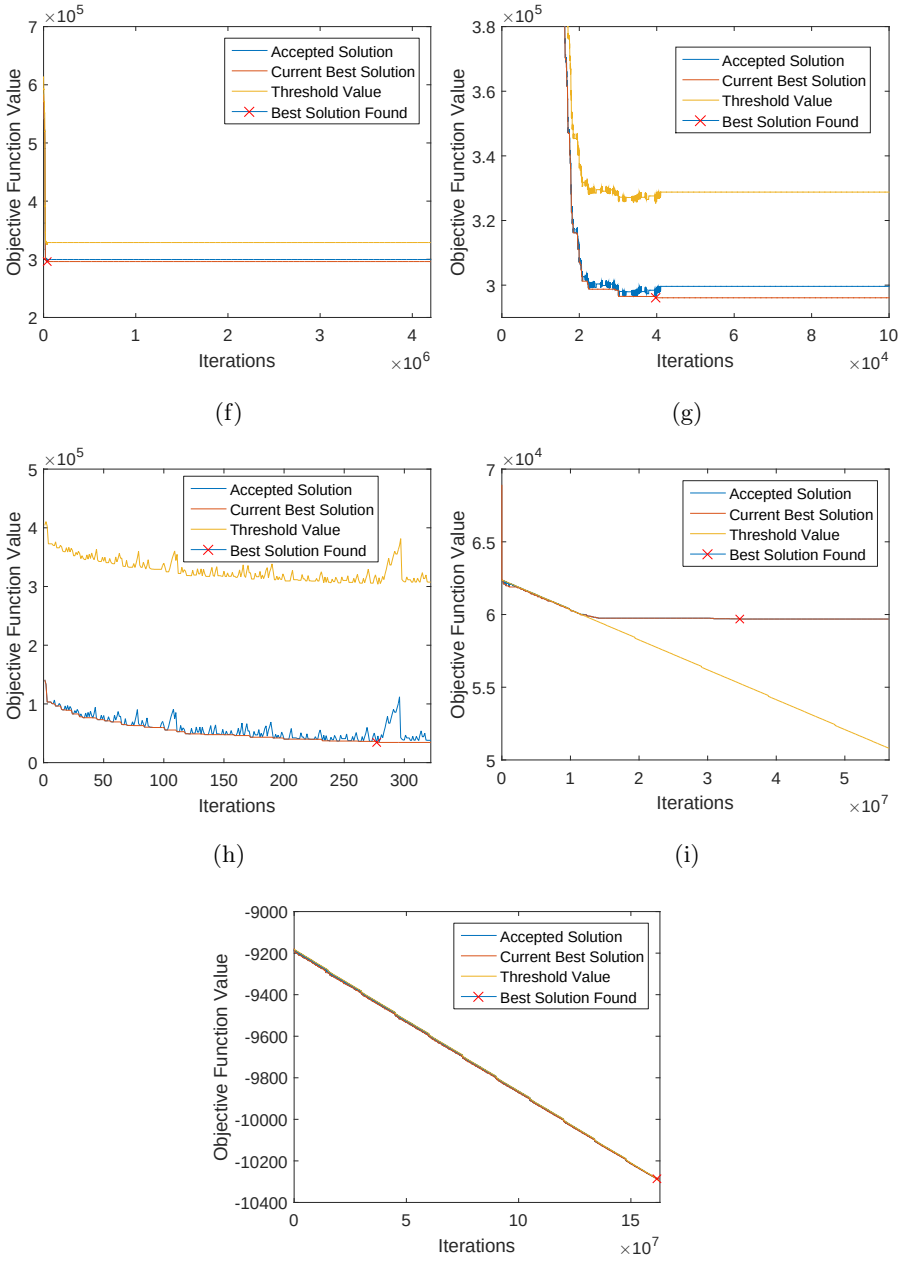


Figure 5: Objective function value traces of accepted solutions and acceptance thresholds for: TA solving (f) instance 5 of the Vehicle Routing problem with (g) focusing on the area where the best solution is found, and (h) instance 9 of the Personnel Scheduling problem, and GD solving (i) instance 6 of the Travelling Salesman problem, and (j) instance 7 of the Max Cut problem.

(GD), and Quadratic Assignment (SARH) are all solved best by a single move acceptance method as indicated in parenthesis. On a side note, due to the non-normal distribution of the results, the Wilcoxon Signed Rank test shows that SARH performs significantly better than the best general method using the cross-domain scores (TA) for the VRP domain. For these cases, it is interesting to see that the different move acceptance methods come from each classification of the accept/reject nature from the taxonomy. Moreover, they all come from different algorithmic parameter setting natures. This however could be due to the fixed internal settings used by the adaptive non-stochastic threshold classification (AILTA) meaning that it could have performed better than GD for solving Travelling Salesman problems if a better mechanism was used. Flow Shop (IE, AILLA), Personnel Scheduling (TA, NA, SA), MAX-SAT (SARH, AILLA), 0-1 Knapsack (SARH, NA) and Max Cut (GD, SARH) are all best solved by a group of move acceptance methods given in parenthesis proceeding the respective domains. With the exception of the Personnel Scheduling problem, the set of move acceptance methods which best solves each problem contains an adaptive nature of the algorithmic parameter setting mechanism. Flow Shop favours being solved by non-stochastic basic methods whereas Personnel Scheduling and 0-1 Knapsack both favour stochastic methods. MAX-SAT and MAX Cut do not favour any particular nature of the accept/reject decision with a stochastic method and a non-stochastic (basic/threshold respectively) method solving them the best. Clearly then, for some problems a particular accept/reject nature is required to solve them well and the nature of the algorithmic parameter settings does not significantly effect their effectiveness. For some problems however, a particular accept/reject nature is not favoured but an adaptive method is always present. If a local search metaheuristic is sought for solving problems from a particular domain, then the respective move acceptance method is recommended to be used as its move acceptance component. If a *new* or *unknown* problem is to be solved, or a *general purpose* search method is sought, we showed that out of those tested, Simulated Annealing was the most effective method *in general* and is therefore recommended for such scenarios.

4.11. Cross-domain Results

The cross-domain scores for the local search metaheuristics as the sum of intra-domain scores over nine problem domains are shown in the final column of Table 7. With a minimum and maximum possible score of 0 and 45 respectively (lower is better) we can see from the results that most move acceptance methods, with NA being the exception, perform better than average obtaining scores less than 22.5. IE as a move acceptance method represents a standard hill climbing local search approach with no mechanism for accepting worse moves. IE should therefore be considered as the baseline result for cross-domain search with methods failing to surpass its performance being deemed to have poor cross-domain performance. Moreover, IE is parameter free meaning that no expert intervention is required to reconfigure it for solving different problems. AILLA, GD, SA, and SARH all improved over IE's cross-domain performance

whereas TA, AILTA, and NA performed worse under the given local search metaheuristic framework.

No move acceptance method could consistently perform the best across all problems, despite their parameters being re-tuned for each problem domain where necessary. Moreover, different move acceptance methods were able to outperform each other when solving different problem instances for some of the domains. The move acceptance method with the best cross-domain performance was Simulated Annealing which never actually performed the best for a given problem domain. Moreover, SA was only able to outperform the remaining move acceptance methods, on average, for 3 out of 45 problem instances. This result highlights the trade-off between requiring and selecting from multiple algorithms which perform exceptionally for solving a few problems each, and having a single algorithm which performs sufficiently well for solving all problems.

The cross-domain performance of SARH closely followed that of SA with scores of 10.10 and 10.22 respectively. An important factor in the design of general-purpose search methods is the ideological goal of eliminating the need for expert intervention. In this study, the parameters of the move acceptance methods were re-tuned for each problem domain to show their potential as a general-purpose search method if a parameter configuration oracle existed. Furthermore, SA had a slight advantage over SARH in that the initial temperatures were determined using an additional procedure which takes place prior to parameter tuning whereas for SARH, the initial temperate was set as a factor of the initial solution cost. On the other hand, this makes SARH more general in that no expert intervention is required to determine a suitable value for the initial temperature beforehand.

Generally, it can be observed that using algorithmic parameter setting mechanisms which are not static improves the cross-domain performance of the algorithms at the same accept/reject nature level. For static methods, use of a non-stochastic basic algorithmic parameter setting mechanism has a clear advantage over a non-stochastic threshold mechanism which in turn outperforms a stochastic mechanism. This is no surprise since static parameter setting mechanisms have no way to reduce the number of accepted worse moves if too many are being accepted. In contrast, this is now possible given dynamic or adaptive parameter setting mechanisms. Hence the reason basic is better than threshold and stochastic for static parameter setting mechanisms, but the opposite is true when using dynamic/adaptive mechanisms.

SA, as well as many of the existing move acceptance methods, contains multiple parameters. These parameters however require re-tuning when the task at hand concerns solving problems across different domains to achieve good performance. Move acceptance methods which are parameter-free (contain no parameters) or parameter-less (designed in such a way that their settings should not need to be changed) can be regarded as more general since they do not require expert intervention when tasked with solving different problems. The parameter-free/parameter-less move acceptance method with the best cross-domain performance was AILLA.

It is extremely important to carefully consider the design of adaptive algo-

rithmic parameter setting mechanisms. Both AILTA and AILLA are classified as adaptive and their adaptation mechanisms are quite similar acting upon a set of values which are used directly to determine the acceptance threshold. AILTA acts upon a set of threshold values (ϵ) which are multiplied with the cost of the best solution found so far whereas AILLA acts upon a set of previous best solution values. The significant difference between AILTA and AILLA is that the set of threshold values used by AILTA are themselves fixed whereas the set of previous best solution values depend on maintaining a memory of the search history and hence are adaptive. That is, AILTA contains an adaptive mechanism to choose from a set of fixed/static settings, whereas AILLA contains an adaptive mechanism to choose from a set of values which are adapted. For this reason, AILTA has a poor cross-domain performance compared to AILLA.

4.12. Progress Plot and Acceptance Observations

Progress plots including the current and best solution values, along with the threshold values where applicable, were recorded for the move acceptance method which performed the best for solving each problem domain. These plots, shown in Figure 5, are presented to gain insight into the behaviours of each move acceptance method which allows them to solve each problem well. Moreover, acceptance statistics showing the ratio of accepted and rejected improving, equal, and worse moves are given in Table 9 and Table 10 to emphasise the variety of behaviours exhibited by the different best performing move acceptance methods over the nine problem domains in this study.

Table 9: Percentage of each type of move based on the acceptance decision and move delta, $\delta = f(s') - f(s)$, as improving, equal, and worsening when using the best general local search metaheuristic for solving an instance, as used in the objective function value traces, of the respective problem domain. Note that no move acceptance mechanism has the ability to reject improving moves.

Domain	BP	FS	PS	SAT	TSP	VRP	KP	MAC	QAP
Best Method	IE	IE	TA	SARH	GD	TA	SARH	GD	SARH
Accept $\delta < 0$	0.6%	0.0%	40.8%	19.7%	0.0%	0.1%	43.8%	5.0%	20.3%
Reject $\delta < 0$	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
Accept $\delta = 0$	22.9%	3.5%	26.7%	16.8%	0.1%	99.9%	6.2%	8.3%	0.2%
Reject $\delta = 0$	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
Accept $\delta > 0$	0.0%	0.0%	32.5%	19.7%	0.0%	0.1%	43.8%	5.0%	20.3%
Reject $\delta > 0$	76.5%	96.5%	0.0%	43.8%	99.9%	0.0%	6.2%	81.7%	59.1%

For the Bin Packing, Figure 5a, and Flow Shop, Figure 5b, problems, the IE move acceptance method performed the best, continuously improving the solution over time and finding the best solution towards the end of the search. Trivially, IE accepts all improving and equal moves while rejecting all worse moves. Observing the ratio of accepted and rejected improving, equal, and worse moves, there is a very low frequency of improving moves in both instances with BP having 22.9% and 76.5% of equal and worse attempted moves, and FS having 3.5% and 96.5% of those move attempts respectively. There are three other domains which have the same characteristic, these are TSP, VRP, and MAC.

Table 10: Percentage of accepted and rejected moves based on improving, equal, and worsening move deltas when using the best general local search metaheuristic for solving an instance, as used in the objective function value traces, of the respective problem domain. Note that no move acceptance mechanism has the ability to reject improving moves.

Domain	BP	FS	PS	SAT	TSP	VRP	KP	MAC	QAP
Best Method	IE	IE	TA	SARH	GD	TA	SARH	GD	SARH
Accept $\delta < 0$	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%
Reject $\delta < 0$	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
Accept $\delta = 0$	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%
Reject $\delta = 0$	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%	0.0%
Accept $\delta > 0$	0.0%	0.0%	100.0%	31.0%	0.0%	100.0%	87.6%	5.7%	25.6%
Reject $\delta > 0$	100.0%	100.0%	0.0%	69.0%	100.0%	0.0%	12.4%	94.3%	74.4%

It is interesting to see that these domains are also best solved by other non-stochastic move acceptance methods. Specifically, GD, TA, and GD for solving TSP, VRP, and MAC problems respectively. The GD move acceptance method was the best for solving the TSP, Figure 5i, and MAC, Figure 5j, problems. In both problems, there is a large ratio of worse moves and GD accepts only a small number of them. In both cases, GD improves the solution over time with the current solution cost closely following the threshold value. For solving the MAC problem, GD facilitates gradual improvement of the solution over time for the whole duration of the search, finding the best solution at the end of the search. This contrasts with when GD is used to solve TSP problems where part way though the search, the rate at which the threshold value decreases overcomes the improvement achieved by the move operators causing GD to reduce to the IE move acceptance method. Further improvement of the solution after this point then relies solely on the move operators being able to improve the current solution and the best solution is found part way through this secondary behavioural stage. Using TA to solve VRP, Figure 5f, problems is interesting because the high threshold value meant that it accepted all moves, as can be seen in Figure 5g, leaving the progress of the search in the hands of the move operators. Moreover, TA found the best solution towards the start of the search where the search then worsens slightly before stagnating. This can be seen by the 99.9% of equal quality moves produced and is down to the move operator itself.

For the remaining four domains, the ratios between either all improving, equal, and worse moves or just improving and worse moves were reasonably balanced, and, with the exception of PS, were solved best by a stochastic move acceptance methods. Another case where TA had the best performance was for solving PS problems. Similarly with solving VRP problems, TA accepted all improving, equal, and worse moves. In contrast however, the ratio between these moves was balanced at 40.8%, 26.7%, and 32.5% respectively, and as can be seen from the progress plot in Figure 5h, the threshold value is set high enough such that no worse moves are rejected. This also points towards the fact that the choice of move operator(s) was beneficial to solving this problem. SAT, KP, and QAP were all solved best using the SARH move acceptance method. In each of these domains, the ratio of improving, equal, and worse attempted moves

were more reasonably balanced compared to those covered previously where the attempted worsening moves clearly dominated attempted improving moves. Perhaps most interestingly, the percentage of accepted improving and worse moves, as shown in Table 9, were the same in each domain for all three domains where SARH outperformed the remaining move acceptance methods. Looking at the progress plots for SAT, KP, and QAP, we can observe the characteristic plots for Simulated Annealing based algorithms. For QAP, in Figure 5e, about 5% from the end of the search, we can see a reheat occur which consequently enables the best solution to be improved further than what would have been found without a reheat mechanism. For solving SAT and KP however, the best solutions are found close to the end of the search, thus no reheat mechanism is activated during the search. This means that SARH consequently behaved the same as SA and highlights the sensitivity of the initial parameter settings for SA based algorithms since different strategies were used for calculating the initial temperature as discussed in Section 3.2.7 and Section 3.2.8.

5. Conclusions and Future Work

In this study, we proposed a taxonomy for classifying local search metaheuristics based on their move acceptance method and performed a survey of these algorithms spanning the different components in the taxonomy. In addition, an empirical study comparing the performance of said move acceptance methods, one from each of the possible classifications, was carried out comparing their performance across 9 characteristically different problems under a single point-based search framework, representing the cross-domain search problem.

If a researcher or practitioner is seeking guidelines for which move acceptance method to use for solving a *new*, *unknown*, or *existing* problem that is not covered in this study, then Simulated Annealing is recommended since it has the best performance over our cross-domain benchmark. If time is constrained, and/or sufficient resources are not available for conducting parameter tuning experiments, then AILLA, as a parameter-less method, should be used to gain reasonable results. If the problem to be solved however is *known* and covered in this study, then the move acceptance method recommended for use is that which has the best intra-domain performance, as stylised bold in Table 7, for that problem domain.

The level of generality at which search methods can act have been raised through the use of hyper-heuristics in the past, allowing them to solve problems from different domains without modification. Realistically however, their effectiveness for solving problems well from different domains is not as good as what they should be, and remains in the focus of current research. Since the design of AdapHH in 2011 (Mısır et al., 2011a), only a few methods have been able to improve upon its cross-domain performance. This was either achieved using machine learning techniques for improving heuristic selection (Kheiri and Özcan, 2016), or by using accidental complexity analysis to both simplify and optimise the existing design of AdapHH (Adriaensen and Nowé, 2016). In this study, we have shown that even under a simple local search metaheuristic framework,

the choice of move acceptance method can have a significant effect on the cross-domain performance of such search methods; the variation in performance of each of the move acceptance methods can even be seen across each of the problem domains despite performing parameter tuning for solving each problem. Considering these outcomes, a potential future research direction to improve the effectiveness of general purpose search methods should focus on improving the move acceptance components by designing them specifically for solving the cross-domain search problem.

Traditionally, problems are solved using a single move acceptance method and their parameters are tuned and/or controlled for the given problem. The results show that different problems are better solved by move acceptance methods with different classifications of the accept/reject decision nature. Based on this, there are at least two ways in which the effectiveness of general purpose search methods may be improved. One strategy could involve the use of multi-stage algorithms which utilise different move acceptance methods throughout the search. This has the advantage of being able to mix such methods which are, with respect to our taxonomy, characteristically different. In theory, such search methods could then use search history to adapt the search process by changing between each of them as required to effectively solve the problem in hand. A multi-stage approach was used in (Kheiri and Özcan, 2016) employing two move acceptance strategies; greedy and threshold acceptance. At each stage, the greedy strategy is invoked stochastically, otherwise the threshold accepting strategy is employed. Another possibility is to select a single move acceptance method, or use the decisions, from a set of move acceptance methods. Previous work has involved the mixture of move acceptance criteria. Tensor analysis, as an advanced machine learning technique, was used in (Asta and Özcan, 2015) to associate a set of move operators with two move acceptance methods within a hyper-heuristic. Each move acceptance method was used in a phase based approach where each subsequent phase employed the opposite move acceptance method to the one currently used. Group decision making was used in (Kheiri et al., 2016), and is a technique used to collaboratively arrive at a decision to accept or reject a move based on several independent move acceptance methods. Each move acceptance method within a group decision making strategy can be assigned a weight to change the influence that each move acceptance method has in the overall decision. Furthermore, adaptation of these weights based on the nature of the accept/reject decision of the move acceptance methods can be used to increase the influence that each has based on the state of the search. These are just some of the ways in which move acceptance methods using different natures of accept/reject decisions can be used together under the same search method.

In this study, we used a single point-based search framework as provided in Algorithm 1, perturbing a solution and then making an accept/reject decision for the new solution. While not all researchers may agree with the methodology used in this paper, the work that is presented is still of value to the wider community since this experimentation can be generalised and extended to other local search frameworks and their hybrids (Talbi, 2013) such as Tabu Search (Glover

and Laguna, 2013) where the inclusion of a memory for guiding the search on top of the different move acceptance methods can be compared. Similarly, an Iterated Local Search framework (Lourenço et al., 2003), which perturbs and then applies hill climbing on an incumbent solution, can be investigated using various move acceptance methods. In their paper, Lourenço et al. (2003) discuss the concept of applying such acceptance criteria, referring to *Better*, *RW*, and *LMSC* as such strategies.

In addition to developing new move acceptance methods, another future research direction for improving the effectiveness of general-purpose search methods is to further extend the definition of a hyper-heuristic to be a *hyper*-hyper-heuristic (hyper^{2}heuristic). The definition of a hyper-heuristic is *"a search method or learning mechanism for selecting or generating heuristics to solve computationally difficult problems"*. Hyper-heuristics contain a single heuristic selection/generation strategy, and a single move acceptance method. A hyper^{2}heuristic would be a "search method or learning mechanism for selecting or generating the *heuristic selection and move acceptance components* of a hyper-heuristic to solve computationally hard problems". Based on our observations in this study, a search method which can select a move acceptance method whose characteristics are appropriate for the current problem being solved could potentially overcome the shortcomings of using a single move acceptance method within a hyper-heuristic. As with conventional hyper-heuristics, there are also two types of hyper^{2}heuristic; generation hyper^{2}heuristics, and selection hyper^{2}heuristics. Since hyper^{2}heuristics can generate or select from hyper-heuristic components, they themselves can simultaneously embed both generation and selection hyper-heuristics. A generation hyper^{2}heuristic is a heuristic that generates a heuristic selection or move acceptance component of a hyper-heuristic. Previous work includes Hyde et al. (2009) where genetic programming was used to evolve move acceptance methods combining accept all moves and great deluge within hyper-heuristics. A similar method was used in Sabar et al. (2015) which used gene expression programming. Machine learning was used in Asta and Özcan (2014) to generate different move acceptance methods to be used with each move operator under a hyper-heuristic framework. A selection hyper^{2}heuristic on the other hand is a heuristic that selects from a set of heuristic selection and move acceptance methods. To our knowledge, such selection hyper^{2}heuristics have not been explored and is proposed as a new and interesting area of research with the potential to increase the effectiveness of present generation general-purpose search methods.

References

Abramson, D., Krishnamoorthy, M., Dang, H., 1999. Simulated annealing cooling schedules for the school timetabling problem.

Adriaensen, S., Nowé, A., 2016. Case study: An analysis of accidental complexity in a state-of-the-art hyper-heuristic for hyflex, in: IEEE Congress on

Evolutionary Computation, CEC 2016, Vancouver, BC, Canada, July 24-29, 2016, pp. 1485–1492.

Adriaensen, S., Ochoa, G., 2015a. A hyflex module for the 0-1 knapsack problem.

Adriaensen, S., Ochoa, G., 2015b. A hyflex module for the max cut problem.

Adriaensen, S., Ochoa, G., 2015c. A hyflex module for the quadratic assignment problem.

Adriaensen, S., Ochoa, G., Nowé, A., 2015. A benchmark set extension and comparative study for the hyflex framework, in: Evolutionary Computation (CEC), 2015 IEEE Congress on, IEEE. pp. 784–791.

Ahmed, L.N., Özcan, E., Kheiri, A., 2015. Solving high school timetabling problems worldwide using selection hyper-heuristics. Expert Syst. Appl. 42, 5463–5471.

Anagnostopoulos, A., Michel, L., Hentenryck, P.V., Vergados, Y., 2006. A simulated annealing approach to the traveling tournament problem. J. of Scheduling 9, 177–193.

Anstegui, C., Gabs, J., Malitsky, Y., Sellmann, M., 2016. Maxsat by improved instance-specific algorithm configuration. Artificial Intelligence 235, 26–39.

Applegate, D., Bixby, R., Chvtal, V., Cook, W., 2011. The traveling salesman problem: A computational study.

Argelich, J., Li, C.M., Manya, F., Planes, J., 2009. Maxsat evalulation 2009 benchmark data sets.

Asta, S., Özcan, E., 2014. An apprenticeship learning hyper-heuristic for vehicle routing in hyflex, in: Evolving and Autonomous Learning Systems (EALS), 2014 IEEE Symposium on, pp. 65–72.

Asta, S., Özcan, E., 2015. A tensor-based selection hyper-heuristic for cross-domain heuristic search. Information Sciences 299, 412–432.

Asta, S., zcan, E., Curtois, T., 2016. A tensor based hyper-heuristic for nurse rostering. Knowledge-Based Systems 98, 185–199.

Ausiello, G., Crescenzi, P., Protasi, M., 1995. Approximate solution of {NP} optimization problems. Theoretical Computer Science 150, 1–55.

Ayob, M., Kendall, G., 2003. A monte carlo hyper-heuristic to optimise component placement sequencing for multi head placement machine, in: PLACEMENT MACHINE, INTECH'03 THAILAND, pp. 132–141.

Battiti, R., Brunato, M., Mascia, F., 2008. Reactive Search and Intelligent Optimization. volume 45 of *Operations research/Computer Science Interfaces*. Springer Verlag.

Beheshti, Z., Shamsuddin, S.M., 2013. A review of population-based meta-heuristic algorithm. International Journal of Advances in Soft Computing and its Applications 5.

Bellmore, M., Nemhauser, G.L., 1968. The traveling salesman problem: A survey. Operations Research 16, 538–558.

Ben-Ameur, W., Mahjoub, A., Neto, J., 2014. The Maximum Cut Problem.

Bianchi, L., Dorigo, M., Gambardella, L.M., Gutjahr, W.J., 2009. A survey on metaheuristics for stochastic combinatorial optimization. Natural Computing 8, 239–287.

Bilgin, B., Özcan, E., Korkmaz, E.E., 2007. An Experimental Study on Hyper-heuristics and Exam Timetabling. Springer Berlin Heidelberg, Berlin, Heidelberg. pp. 394–412.

Birattari, M., Paquete, L., Stützle, T., Varrentrapp, K., 2001. Classification of Metaheuristics and Design of Experiments for the Analysis of Components. Technical Report. Intellektik Darmstadt University of Technology, Darmstadt, Germany.

Blum, C., Puchinger, J., Raidl, G.R., Roli, A., 2011. Hybrid metaheuristics in combinatorial optimization: A survey. Applied Soft Computing 11, 4135–4151.

Blum, C., Roli, A., 2003. Metaheuristics in combinatorial optimization: Overview and conceptual comparison. ACM Comput. Surv. 35, 268–308.

Burkard, R.E., Karisch, S.E., Rendl, F., 1997. Qaplib - a quadratic assignment problem library. Journal of Global Optimization 10, 391–403.

Burke, E.K., Bykov, Y., 2006. Solving exam timetabling problems with the flex-deluge algorithm, in: Proceedings of the International Conference on the Practice and Theory of Automated Timetabling (PATAT 2006), pp. 370–372.

Burke, E.K., Bykov, Y., 2008. A late acceptance strategy in hill-climbing for exam timetabling problems, in: Proceedings of the International Conference on the Practice and Theory of Automated Timetabling (PATAT 2008), Montreal, Canada. p. Extended Abstract.

Burke, E.K., Curtois, T., Hyde, M., Kendall, G., Ochoa, G., Petrovic, S., Vázquez-Rodríguez, J., Gendreau, M., 2010. Iterated local search vs. hyper-heuristics: Towards general-purpose search algorithms, in: Evolutionary Computation (CEC), 2010 IEEE Congress on, pp. 1–8.

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

Bykov, Y., Petrovic, S., 2016. A step counting hill climbing algorithm applied to university examination timetabling. Journal of Scheduling 19, 479–492.

Connolly, D., 1992. General purpose simulated annealing. The Journal of the Operational Research Society 43, 495–505.

Cowling, P., Kendall, G., Soubeiga, E., 2001. A Hyperheuristic Approach to Scheduling a Sales Summit. Springer Berlin Heidelberg, Berlin, Heidelberg. pp. 176–190.

CRIL, 2007. Sat competition 2007 benchmark data sets.

CRIL, 2009. Sat competition 2009 benchmark data sets.

Curtois, T., 2009. Staff rostering benchmark data sets.

Curtois, T., Ochoa, G., Hyde, M., Vazquez-Rodriguez, J.A., 2010. A HyFlex Module for the Personnel Scheduling Problem. Technical Report. School of Computer Science, University of Nottingham.

Di Gaspero, L., Urli, T., 2012. Evaluation of a family of reinforcement learning cross-domain optimization heuristics, in: Learning and Intelligent Optimization. Springer Berlin Heidelberg. Lecture Notes in Computer Science, pp. 384–389.

DIMACS, 2015. 7th dimacs implementation challenge.

Dokeroglu, T., Cosar, A., 2016. A novel multistart hyper-heuristic algorithm on the grid for the quadratic assignment problem. Engineering Applications of Artificial Intelligence 52, 10–25.

Drezner, Z., Hahn, P.M., Taillard, É.D., 2005. Recent advances for the quadratic assignment problem with special emphasis on instances that are difficult for meta-heuristic methods. Annals of Operations Research 139, 65–94.

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

Dueck, G., Scheuer, T., 1990. Threshold accepting: A general purpose optimization algorithm appearing superior to simulated annealing. Journal of Computational Physics 90, 161–175.

Eiben, A.E., Hinterding, R., Michalewicz, Z., 1999. Parameter control in evolutionary algorithms. Evolutionary Computation, IEEE Transactions on 3, 124–141.

Eiben, A.E., Smit, S.K., 2011. Parameter tuning for configuring and analyzing evolutionary algorithms. Swarm and Evolutionary Computation 1, 19–31.

El-Sherbeny, N.A., 2010. Vehicle routing with time windows: An overview of exact, heuristic and metaheuristic methods. Journal of King Saud University - Science 22, 123–131.

Ernst, A., Jiang, H., Krishnamoorthy, M., Sier, D., 2004. Staff scheduling and rostering: A review of applications, methods and models. European Journal of Operational Research 153, 3–27.

Escario, J., Jimenez, J., Giron-Sierra, J., 2015. Ant colony extended: Experiments on the travelling salesman problem. Expert Systems with Applications 42, 390–410.

ESICUP, 2011. European special interest group on cutting and packing benchmark data sets.

Etscheid, M., Rglin, H., 2014. Smoothed analysis of local search for the maximum-cut problem, pp. 882–889.

Ezugwu, A.S., Adewumi, A., Frncu, M., 2017. Simulated annealing based symbiotic organisms search optimization algorithm for traveling salesman problem. Expert Systems with Applications 77, 189–210.

Fernandez-Viagas, V., Ruiz, R., Framinan, J., 2017. A new vision of approximate methods for the permutation flowshop to minimise makespan: State-of-the-art and computational evaluation. European Journal of Operational Research 257, 707–721.

Fernandez-Viagas, V., Valente, J., Framinan, J., 2018. Iterated-greedy-based algorithms with beam search initialization for the permutation flowshop to minimise total tardiness. Expert Systems with Applications 94, 58–69.

Frville, A., 2004. The multidimensional 0-1 knapsack problem: An overview. European Journal of Operational Research 155, 1–21.

Garey, Michael R.; Johnson, D.S., 1990. Computers and Intractability: A Guide to the Theory of NP-Completeness. W. H. Freeman & Co.

Glover, F., Laguna, M., 2013. Tabu search. volume 5-5.

Hafiz, F., Abdennour, A., 2016. Particle swarm algorithm variants for the quadratic assignment problems - a probabilistic learning approach. Expert Systems with Applications 44, 413–431.

Heuberger, C., 2004. Inverse combinatorial optimization: A survey on problems, methods, and results. Journal of Combinatorial Optimization 8, 329–361.

Hoos, H., Stützle, T., 2004. Stochastic Local Search: Foundations & Applications. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

Hopper, E., Turton, B.C.H., 2001. A review of the application of meta-heuristic algorithms to 2d strip packing problems. Artificial Intelligence Review 16, 257–300.

Hyde, M., Ochoa, G., Curtois, T., Vazquez-Rodriguez, J.A., 2010a. A HyFlex Module for the Maximum Satisfiability (MAX-SAT) Problem. Technical Report. School of Computer Science, University of Nottingham.

Hyde, M., Ochoa, G., Curtois, T., Vazquez-Rodriguez, J.A., 2010b. A HyFlex Module for the One Dimensional Bin Packing Problem. Technical Report. School of Computer Science, University of Nottingham.

Hyde, M., Özcan, E., Burke, E.K., 2009. Multilevel search for evolving the acceptance criteria of a hyper-heuristic, in: Proceedings of the the 4th Multidisciplinary Int. conf. on Scheduling: Theory and Applications, pp. 798–801.

Hyde, M.R., 2011. One dimensional packing benchmark data sets.

Ikegami, A., Niwa, A., 2003. A subproblem-centric model and approach to the nurse scheduling problem. Mathematical Programming 97, 517–541.

Ingber, L., 1989. Very fast simulated re-annealing. Mathematical and Computer Modelling 12, 967–973.

Jackson, W.G., Özcan, E., John, R.I., 2017. Tuning a simulated annealing metaheuristic for cross-domain search, in: 2017 IEEE Congress on Evolutionary Computation (CEC), pp. 1055–1062.

Johnson, D.S., Demers, A., Ullman, J.D., Garey, M.R., Graham, R.L., 1974. Worst-case performance bounds for simple one-dimensional packing algorithms. SIAM Journal on Computing 3, 299–325.

Kheiri, A., Mısır, M., Özcan, E., 2016. Ensemble Move Acceptance in Selection Hyper-heuristics. Springer International Publishing. pp. 21–29.

Kheiri, A., Özcan, E., 2016. An iterated multi-stage selection hyper-heuristic. European Journal of Operational Research 250, 77–90.

Kirkpatrick, S., Gelatt, C.D., Vecchi, M.P., 1983. Optimization by simulated annealing. Science 220, 671–680.

Landa-Silva, D., Obit, J.H., 2008. Great deluge with non-linear decay rate for solving course timetabling problems, in: Intelligent Systems, 2008. IS '08. 4th International IEEE Conference, pp. 8–11.

Lewis, R., 2008. A survey of metaheuristic-based techniques for university timetabling problems. OR Spectrum 30, 167–190.

Liao, T.W., Egbelu, P.J., Sarker, B.R., Leu, S.S., 2011. Metaheuristics for project and construction management - a state-of-the-art review. Automation in Construction 20, 491–505.

Lim, T., Al-Betar, M., Khader, A., 2016. Taming the 0/1 knapsack problem with monogamous pairs genetic algorithm. Expert Systems with Applications 54, 241–250.

Lin, C., Choy, K., Ho, G., Chung, S., Lam, H., 2014. Survey of green vehicle routing problem: Past and future trends. Expert Systems with Applications 41, 1118–1138.

Lourenço, H.R., Martin, O.C., Stutzle, T., 2003. Iterated local search. International series in operations research and management science , 321–354.

Lundy, M., Mees, A., 1986. Convergence of an annealing algorithm. Mathematical Programming 34, 111–124.

Malcolm, S., Klaus, M., 2002. Monte carlo methods in geophysical inverse problems. Reviews of Geophysics 40, 3–1–3–29.

Marc, D., Jérôme, M., 2014. An Introduction to Inverse Combinatorial Problems. Wiley-Blackwell. chapter 17. pp. 547–586. https://onlinelibrary.wiley.com/doi/pdf/10.1002/9781119005353.ch17.

McCollum, B., McMullan, P., Parkes, A.J., Burke, E.K., Abdullah, S., 2009. An extended great deluge approach to the examination timetabling problem, in: 4th Multidisciplinary International Conference on Scheduling: Theory and Applications, 2009, pp. 424–434.

Metropolis, N., Rosenbluth, A.W., Rosenbluth, M.N., Teller, A.H., Teller, E., 1953. Equation of state calculations by fast computing machines. The Journal of Chemical Physics 21, 1087–1092.

Mısır, M., De Causmaecker, P., Vanden Berghe, G., Verbeeck, K., 2011a. An adaptive hyper-heuristic for chesc 2011 .

Mısır, M., Vancroonenburg, W., Verbeeck, K., Berghe, G.V., 2011b. A selection hyper-heuristic for scheduling deliveries of ready-mixed concrete, in: Proceedings of the Metaheuristics International Conference (MIC 2011), pp. 289–298.

Mısır, M., Verbeeck, K., De Causmaecker, P., Berghe, G.V., 2010. Hyper-heuristics with a dynamic heuristic set for the home care scheduling problem, in: IEEE Congress on Evolutionary Computation, pp. 1–8.

Mısır, M., Verbeeck, K., De Causmaecker, P., Vanden Berghe, G., 2012. An Intelligent Hyper-Heuristic Framework for CHeSC 2011. Springer Berlin Heidelberg. pp. 461–466.

Mısır, M., Wauters, T., Verbeeck, K., Vanden Berghe, G., 2009. A new learning hyper-heuristic for the traveling tournament problem, in: Proceedings of the 8th Metaheuristic International Conference (MIC09).

Montoya-Torres, J., Lpez Franco, J., Nieto Isaza, S., Felizzola Jimnez, H., Herazo-Padilla, N., 2015. A literature review on the vehicle routing problem with multiple depots. Computers and Industrial Engineering 79, 115–129.

Morgado, A., Heras, F., Liffiton, M., Planes, J., Marques-Silva, J., 2013. Iterative and core-guided maxsat solving: A survey and assessment. Constraints 18, 478–534.

Nawaz, M., Emory Enscore Jr., E., Ham, I., 1983. A heuristic algorithm for the m-machine, n-job flow-shop sequencing problem. Omega 11, 91–95.

Nino-Ruiz, E.D., Ardila, C., Capacho, R., 2017. Local search methods for the solution of implicit inverse problems. Soft Computing , 1–14.

Ochoa, G., Hyde, M., Curtois, T., Vazquez-Rodriguez, J.A., Walker, J., Gendreau, M., Kendall, G., McCollum, B., Parkes, A., Petrovic, S., Burke, E.K., 2012. HyFlex: A Benchmark Framework for Cross-Domain Heuristic Search. Springer Berlin Heidelberg, Berlin, Heidelberg. pp. 136–147.

Özcan, E., Bilgin, B., Korkmaz, E.E., 2008. A comprehensive analysis of hyper-heuristics. Intell. Data Anal. 12, 3–23.

Özcan, E., Mısır, M., Ochoa, G., Burke, E.K., 2010. A reinforcement learning - great-deluge hyper-heuristic for examination timetabling. International Journal of Applied Metaheuristic Computing 1, 39–59.

Papadimitriou, C.H., Steiglitz, K., 1982. Combinatorial Optimization: Algorithms and Complexity. Prentice-Hall, Inc., Upper Saddle River, NJ, USA.

Petrovic, S., Yang, Y., Dror, M., 2007. Case-based selection of initialisation heuristics for metaheuristic examination timetabling. Expert Systems with Applications 33, 772–785.

Pisinger, D., 2015. A more advanced generator for 0-1 knapsack problems.

Reinelt, G., 2008. Tsplib, a library of sample instances for the tsp.

Rinaldi, G., 2015. Rudy graph generator.

Roy, R.K., 2010. A primer on the Taguchi method. Society of Manufacturing Engineers.

Sabar, N.R., Ayob, M., Kendall, G., Qu, R., 2013. Grammatical evolution hyper-heuristic for combinatorial optimization problems. Evolutionary Computation, IEEE Transactions on 17, 840–861.

Sabar, N.R., Ayob, M., Kendall, G., Qu, R., 2015. Automatic design of a hyper-heuristic framework with gene expression programming for combinatorial optimization problems. Evolutionary Computation, IEEE Transactions on 19, 309–325.

Santos, H., Toffolo, T., Gomes, R., Ribas, S., 2016. Integer programming techniques for the nurse rostering problem. Annals of Operations Research 239, 225–251.

Scheithauer, G., 2018. One-dimensional bin packing. International Series in Operations Research and Management Science 263, 47–72.

Sim, K., 2011. Ksats-hh: A simulated annealing hyper-heuristic with reinforcement learning and tabu-search. Entry in the Cross-domain Heuristic Search Challenge available from http://www.asap.cs.nott.ac.uk/external/chesc2011/index.html, June .

Sin, E.S., Kham, N.S.M., 2012. Hyper heuristic based on great deluge and its variants for exam timetabling problem. CoRR abs/1202.1891.

Sinclair, M., 1993. Comparison of the performance of modern heuristics for combinatorial optimization on real data. Computers & Operations Research 20, 687–695.

SINTEF, 2011. Vrptw benchmark problems, on the sintef transport optimisation portal.

Sörensen, K., Glover, F.W., 2013. Metaheuristics, in: Encyclopedia of Operations Research and Management Science. Springer US, pp. 960–970.

Sridhar, R., Chandrasekaran, M., Sriramya, P., Raja, S., 2017. A review on application of 1d, 2d and 3d bin packing techniquies. Journal of Advanced Research in Dynamical and Control Systems 2017, 165–169.

Taillard, E., 1993. Benchmarks for basic scheduling problems. European Journal of Operational Research 64, 278–285.

Talbi, E.G., 2013. A Unified Taxonomy of Hybrid Metaheuristics with Mathematical Programming, Constraint Programming and Machine Learning. Springer Berlin Heidelberg, Berlin, Heidelberg. pp. 3–76.

Tarantilis, C.D., Kiranoudis, C.T., 2002. A list-based threshold accepting method for job shop scheduling problems. International Journal of Production Economics 77, 159–171.

Tarantilis, C.D., Kiranoudis, C.T., Vassiliadis, V.S., 2004. A threshold accepting metaheuristic for the heterogeneous fixed fleet vehicle routing problem. European Journal of Operational Research 152, 148–158.

Vazquez-Rodriguez, J.A., Ochoa, G., Curtois, T., Hyde, M., 2010. A HyFlex Module for the Permutation Flow Shop Problem. Technical Report. School of Computer Science, University of Nottingham.

Walker, J., Ochoa, G., Gendreau, M., Burke, E.K., 2012. A vehicle routing domain for the hyflex hyper-heuristics framework, in: Proceedings of Learning and Intelligent Optimization (LION 2012), pp. 265–276.

Zhou, Y., Chen, X., Zhou, G., 2016. An improved monkey algorithm for a 0-1 knapsack problem. Applied Soft Computing Journal 38, 817–830.

Zhou, Y., Lai, X., Li, K., 2015. Approximation and parameterized runtime analysis of evolutionary algorithms for the maximum cut problem. IEEE Transactions on Cybernetics 45, 1491–1498.

Appendices

A. Non-normalised Results for all Problem Instances

Table 1: Mean averages and statistical significance between each local search metaheuristic and the best general method over 31 runs for each problem domain where the local search metaheuristic with the best intra-domain score is stylised bold for each problem, and $<$ ($>$) is used to show that the respective method performs statistically significantly better (worse) than the best general method using a left (right) tailed Wilcoxon signed rank test with CI = 95%, and $\leq$ ($\geq$) is used when there is no statistically significant difference but shows that the respective method performs better (worse) compared to their mean results. $\equiv$ is used when the mean of both algorithms are exactly the same.

	Bin Packing (BP)									
	Instance 1		Instance 7		Instance 9		Instance 10		Instance 11	
IE	**0.0044**	-	**0.0080**	-	**0.0137**	-	**0.1088**	-	**0.0244**	-
AILLA	0.0175	$>$	0.0304	$>$	0.0234	$>$	0.1126	$>$	0.0374	$>$
TA	0.0324	$>$	0.0680	$>$	0.0368	$>$	0.1258	$>$	0.0584	$>$
GD	0.0104	$>$	0.0192	$>$	0.0178	$>$	0.1089	$>$	0.0303	$>$
AILTA	0.0178	$>$	0.0215	$>$	0.0219	$>$	0.1128	$>$	0.0362	$>$
NA	0.0577	$>$	0.1820	$>$	0.0368	$>$	0.1272	$>$	0.0656	$>$
SA	0.0328	$>$	0.0597	$>$	0.0419	$>$	0.1221	$>$	0.0610	$>$
SARH	0.0349	$>$	0.0687	$>$	0.0352	$>$	0.1235	$>$	0.0619	$>$
	Flow Shop (FS)									
	Instance 1		Instance 3		Instance 8		Instance 10		Instance 11	
IE	6279.00	-	**6345.90**	-	26804.32	-	**11385.68**	-	**26638.87**	-
AILLA	**6271.19**	$\leq$	6356.48	$>$	**26800.03**	$\leq$	11400.23	$\geq$	26646.26	$\geq$
TA	6276.71	$\leq$	6355.81	$>$	27302.45	$>$	11519.26	$>$	27075.61	$>$
GD	6277.90	$\leq$	6355.42	$>$	26833.45	$>$	11411.94	$>$	26690.74	$>$
AILTA	6365.26	$>$	6406.00	$>$	27273.94	$>$	11595.32	$>$	27045.90	$>$
NA	6576.29	$>$	6611.55	$>$	27303.77	$>$	11768.61	$>$	27076.84	$>$
SA	6282.61	$\geq$	6347.32	$\geq$	26815.16	$>$	11406.13	$>$	26666.61	$>$
SARH	6274.97	$\leq$	6354.87	$\geq$	26858.87	$>$	11418.10	$>$	26682.16	$>$
	Personnel Scheduling (PS)									
	Instance 5		Instance 8		Instance 9		Instance 10		Instance 11	
IE	1223.42	$>$	60041.94	$>$	99145.97	$>$	3036.16	$>$	4550.13	$>$
AILLA	928.87	$>$	56430.29	$>$	97238.45	$>$	2186.68	$>$	1884.45	$>$
TA	49.71	-	49444.94	-	**45444.84**	-	1627.03	-	494.65	-
GD	577.74	$>$	56578.71	$>$	69897.03	$>$	2019.19	$>$	473.61	$\leq$
AILTA	1207.00	$>$	60041.94	$>$	99145.97	$>$	3036.16	$>$	4550.13	$>$
NA	**47.77**	$<$	48672.52	$<$	51132.55	$>$	**1612.23**	$\leq$	**455.26**	$<$
SA	49.71	$\equiv$	**48566.10**	$<$	51855.35	$>$	1655.48	$>$	474.61	$\leq$
SARH	50.42	$\geq$	49376.87	$\leq$	51798.97	$>$	1684.03	$>$	472.61	$<$
	Maximum Satisfiability (SAT)									
	Instance 3		Instance 4		Instance 5		Instance 10		Instance 11	
IE	30.10	$>$	40.42	$>$	54.94	$>$	27.65	$>$	12.81	$>$
AILLA	6.77	$>$	6.23	$>$	14.42	$>$	**4.87**	$<$	**7.77**	$<$
TA	30.10	$>$	40.42	$>$	54.94	$>$	27.65	$>$	12.81	$>$
GD	7.45	$>$	5.45	$>$	**8.19**	$\leq$	10.03	$>$	9.23	$>$
AILTA	30.10	$>$	40.42	$>$	54.94	$>$	27.65	$>$	12.81	$>$
NA	171.74	$>$	180.61	$>$	255.06	$>$	237.03	$>$	82.90	$>$
SA	7.81	$>$	7.65	$>$	13.74	$>$	8.87	$\geq$	8.26	$\leq$
SARH	**5.87**	-	**3.97**	-	9.61	-	8.68	-	8.35	-
	TSP									
	Instance 0		Instance 2		Instance 6		Instance 7		Instance 8	
IE	61397.45	$>$	8015.58	$>$	59660.40	$>$	**76636.78**	$<$	**2.503**$\times 10^7$	$<$
AILLA	57731.16	$>$	**7765.92**	$<$	61460.80	$>$	78715.32	$>$	2.506×10^7	$>$
TA	59083.71	$>$	8014.17	$>$	61878.83	$>$	79086.41	$>$	2.507×10^7	$>$
GD	56722.47	-	7810.23	-	**59534.18**	-	76675.67	-	2.504×10^7	-
AILTA	57590.58	$>$	8274.93	$>$	61871.75	$>$	79086.82	$>$	2.507×10^7	$>$
NA	63039.82	$>$	8279.75	$>$	61888.08	$>$	79086.89	$>$	2.507×10^7	$>$
SA	58777.11	$>$	7936.98	$>$	59621.38	$>$	76982.33	$>$	2.506×10^7	$>$
SARH	**56162.12**	$\leq$	8253.69	$>$	61888.08	$>$	79086.89	$>$	2.507×10^7	$>$

	VRP									
	Instance 1		Instance 2		Instance 5		Instance 6		Instance 9	
IE	44486.12	>	31117.25	>	580546.60	>	326023.29	>	390515.55	>
AILLA	44484.07	>	29697.39	≥	580543.39	>	325751.88	≥	390286.71	>
TA	30602.21	-	28943.13	-	**316610.64**	-	324344.30	-	389129.20	-
GD	44484.32	>	29046.09	≥	580544.25	>	325173.72	≥	389654.16	>
AILTA	44448.59	>	29601.24	≥	382948.27	>	324492.35	<	389052.44	<
NA	30941.72	≥	**28782.94**	≤	341398.15	>	324202.99	<	388681.77	<
SA	30565.62	≤	29523.78	≥	347218.93	>	324710.35	≥	389054.67	<
SARH	**30531.03**	≤	29117.72	≥	335212.79	≥	**323792.05**	<	**388615.01**	<
	KP									
	Instance 0		Instance 1		Instance 3		Instance 5		Instance 8	
IE	-96693.00	>	-298587.00	>	-192251.00	>	-1225473.00	>	-770658.00	>
AILLA	-96693.00	>	-298587.00	>	-192251.00	>	-1225473.00	>	-770658.00	>
TA	-97588.19	>	-641594.55	>	-315430.13	>	-1650922.19	>	-1051084.32	>
GD	-96693.00	>	-298587.00	>	-192251.00	>	-1225473.00	>	-770658.00	>
AILTA	-96693.00	>	-298587.00	>	-192251.00	>	-1446536.94	>	-817592.81	>
NA	-97487.03	>	-1002562.97	>	-401438.87	>	-3260820.55	>	**-1400554.52**	<
SA	**-97987.32**	<	-1162161.68	>	-385713.74	>	-3682250.84	>	-1291521.26	>
SARH	-97771.35	-	**-1196733.74**	-	**-403552.81**	-	**-3971527.97**	-	-1357087.48	-
	MAC									
	Instance 0		Instance 2		Instance 5		Instance 7		Instance 9	
IE	-3.524×10^{7}	>	-3009.71	>	-13039.13	>	-9916.26	>	-2932.06	>
AILLA	-3.987×10^{7}	>	-3052.77	>	-13227.35	>	-10135.10	>	-2980.84	>
TA	-3.012×10^{7}	>	-2873.42	>	-12347.03	>	-9141.45	>	-2105.68	>
GD	-4.144×10^{7}	-	-3059.23	-	**-13356.32**	-	**-10276.58**	-	**-2996.65**	-
AILTA	-4.113×10^{7}	>	-3059.16	≥	-13301.52	>	-10205.10	>	-2965.48	>
NA	-3.007×10^{7}	>	-2868.87	>	-12349.77	>	-9142.23	>	-2103.55	>
SA	-4.115×10^{7}	>	-3053.90	>	-13279.52	>	-10164.16	>	-2964.77	>
SARH	$\mathbf{-4.154}\times10^{7}$	<	**-3059.90**	<	-13355.03	<	-10261.48	>	-2994.13	≥
	QAP									
	Instance 0		Instance 6		Instance 7		Instance 8		Instance 9	
IE	154560.39	>	5.146×10^{8}	>	4.495×10^{7}	>	8314086.58	>	275459.68	>
AILLA	152861.16	>	5.113×10^{8}	>	4.486×10^{7}	>	8207670.32	>	273895.03	>
TA	155877.23	>	5.108×10^{8}	>	4.489×10^{7}	>	8532912.84	>	280221.16	>
GD	152202.39	>	5.009×10^{8}	≥	4.489×10^{7}	>	8147444.32	>	273221.61	>
AILTA	153472.71	>	5.034×10^{8}	>	4.503×10^{7}	>	8259894.32	>	274763.87	>
NA	168563.55	>	5.983×10^{8}	>	4.793×10^{7}	>	9280370.39	>	289552.26	>
SA	152504.19	>	5.076×10^{8}	>	4.490×10^{7}	>	8203339.48	>	273730.26	>
SARH	**152162.97**	-	$\mathbf{5.007}\times10^{8}$	-	$\mathbf{4.485}\times10^{7}$	-	**8141744.77**	-	**273115.23**	-

B. Local Search Metaheuristic Abbreviations

Table 2: Key of local search metaheuristic abbreviations

Abbreviation	Name
AA	Adaptive Acceptance
AILLA	Adaptive Iteration Limited List-based Threshold Accepting
AILLA-F	Adaptive Iteration Limited List-based Threshold Accepting with a Fixed Limit
AILTA	Adaptive Iteration Limited Threshold Accepting
AM	All Moves
BATA	Backtracking Adaptive Threshold Accepting
EGD	Extended Great Deluge
EMC	Exponential Monte Carlo
EMCQ	Exponential Monte Carlo with Counter
FD	Flex Deluge
GD	Great Deluge
IE	Improving or Equals
ILTA	Iteration Limited Threshold Accepting
LA	Late Acceptance
LBTA	List-based Threshold Accepting
LMC	Linear Monte Carlo
NA	Naïve Acceptance
NLGD	Non-linear Great Deluge
OI	Only Improving (sometimes referred to as Improving Only)
RRT	Record-to-record Travel (sometimes stylised RTR)
SA	Simulated Annealing
SARH	Simulated Annealing with Reheating
SCHC	Step Counting Hill Climbing
TA	Threshold Accepting
VFR	Very Fast Simulated Re-annealing