Adaptive Iterated Local Search for Cross-domain Optimisation

Edmund K. Burke
ASAP Group
School of Computer Science
University of Nottingham, UK
ekb@cs.nott.ac.uk

Michel Gendreau
CIRRELT Center
University of Montreal,
Canada
michelg@crt.umontreal.ca

Gabriela Ochoa
ASAP Group
School of Computer Science
University of Nottingham, UK
gxo@cs.nott.ac.uk

James D. Walker
ASAP Group
School of Computer Science
University of Nottingham, UK
jdw@cs.nott.ac.uk

ABSTRACT

We propose two adaptive variants of a multiple neighborhood iterated local search algorithm. These variants employ online learning techniques, also called adaptive operation selection, in order to select which perturbation to apply at each iteration step from a set of available move operators. Using a common software interface (the *HyFlex* framework), the proposed algorithms are tested across four hard combinatorial optimisation problems: permutation flow shop, 1D bin packing, maximum satisfiability, and personnel scheduling (including instance data from real-world industrial applications). Using the HyFlex framework, exactly the same high level search strategy can be applied to all the domains and instances. Our results confirm that the adaptive variants outperform a baseline iterated local search with uniform random selection of the move operators. We argue that the adaptive algorithms proposed are general yet powerful, and contribute to the goal of increasing the generality and applicability of heuristic search.

Categories and Subject Descriptors

I.2.8 [**Artificial Intelligence**]: Problem Solving, Control Methods, and Search—*Heuristic methods*

General Terms

Algorithms, Design.

Keywords

Combinatorial optimization, iterated local search, hyper-heuristics, meta-heuristics, adaptive.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
GECCO'11, July 12–16, 2011, Dublin, Ireland.
Copyright 2011 ACM 978-1-4503-0557-0/11/07 ...$5.00.

1. INTRODUCTION

The idea of increasing the generality of heuristic search algorithms is appealing, as their potential applicability would increase. Researchers pursuing this direction, are sometimes limited on the number and complexity of problem domains for testing their general purpose methods. This can be explained by the substantial effort required for implementing the required problem-specific data structures, search operators, constraints, and objective functions.

Hyper-heuristics [11, 12, 28] are search methodologies that are motivated by the goal of producing more generally applicable, automated methods. Selective hyper-heuristics are a widely studied method [12]. They attempt to automatically select which heuristic (from a given set) to apply at each decision point during the search process. A hyper-heuristic framework has access to a set of such problem specific low level heuristics (or search operators), and the goal is to combine them to produce an effective self-configured algorithm. In order to facilitate the research and further develop generally applicable cross domain heuristic search methods, a software benchmark framework (*HyFlex*) has recently been developed [5]. This framework is supporting an exciting international research competition: The First Cross-Domain Heuristic Search Challenge [10]. HyFlex is a modular Java class library for supporting the design of cross-domain heuristic search methods. The library provides a number of problem domain modules (for selected hard combinatorial problems), which encapsulate, using a common software interface, all the algorithm components that are problem-specific: namely, the data structure for representing candidate solutions, the objective function, and a repository of associated problem specific low level heuristics of different types. Therefore, for designing a complete and general search heuristic, the user only needs to provide the high-level control strategy that will manage and configure the provided algorithmic building blocks. Notice that the framework allows the interesting possibility of implementing a high-level search controller that can successfully solve instances of hidden/unseen problem domains.

This paper uses HyFlex in order to test the generality and cross-domain abilities of two adaptive variants of iterated local search. The proposed variants incorporate adaptive mechanisms for selecting among a set of available neighborhoods. They improve the best performing method presented in [6], which can be considered as an *iterated local search* (ILS) algorithm with multiple neighborhoods. Iterated local search is a relatively simple but successful algorithm.

It operates by iteratively alternating between applying a move operator to the incumbent solution and restarting local search from the perturbed solution. This search principle has been rediscovered multiple times, within different research communities and different names [1, 26]. The term *iterated local search* was proposed in [25]. In [36], an adaptive operator selection method, *adaptive pursuit* [35], is successfully applied to automatically select the ILS perturbation step size of a single move operator when applied to a single instance of the single-constraint knapsack problem. The variants presented in this article, instead, adapt the choice among different neighborhood structures. Furthermore, their consistency and robustness is tested across several domains and a number of instances for each domain.

2. THE HYFLEX FRAMEWORK

HyFlex (Hyper-heuristics Flexible framework) [5] is a Java object oriented framework for the implementation and comparison of different iterative general-purpose heuristic search algorithms (also called hyper-heuristics). The framework appeals to modularity and is inspired by the notion of a domain barrier between the low-level heuristics and the hyper-heuristic [14, 11] . HyFlex provides a controlled communication protocol (a software interface) between the problem specific components and the domain independent algorithm components. The purpose is to support researchers in their efforts to develop generally applicable search heuristics. The creative efforts can focus in the design of intelligent and adaptable cross-domain search controllers.

HyFlex extends the conceptual hyper-heuristic framework discussed in [14, 11] by maintaining a population (instead of a single incumbent solution) in the problem domain layer. Moreover, a richer variety of low-level heuristics is provided. A relevant antecedent to HyFlex is PISA [3], a text-based software interface for multi-objective evolutionary algorithms. PISA provides a division between the application-specific and an the algorithm-specific parts of a multi-objective evolutionary algorithm. In HyFlex the interface is not text-based. Instead, it is given by an abstract Java class. This allows a more tight coupling between the modules and overcomes some of the speed limitations encountered in PISA. While PISA is designed to implement evolutionary algorithms, HyFlex can be used to implement both population-based and single point metaheuristics and hyper-heuristics. Moreover, it provides a rich variety of combinatorial optimisation problems including real-world instance data. Each HyFlex domain module encapsulates the problem model and the data structure for encoding a candidate solution. It also provides: (i) a routine to initialise randomised solutions, (ii) an objective function for evaluating the quality of solutions, and (iii) a set of varied instance data sets from different sources that can be easily loaded. More importantly, HyFlex provides a rich set of useful low level heuristics, which can be classified into the following types:

mutational heuristics: these are perturbation or mutation heuristics which induce a small modification to the current solution.

ruin-recreate heuristics: these are large neighborhoods or construction and destruction heuristics. They operate by randomly destroying part of the solution and then rebuilding it using a greedy or constructive procedure. They differ from mutational heuristics not only on the extent of the change, but also because problem specific constructive heuristics are used to recreate the partially destroyed solutions.

hill-climbing heuristics: operate by iteratively perturbing an incumbent solution, accepting improving or non-deteriorating solutions, until a local optimum is found or a stopping condition is met. They differ from mutational heuristics in that they incorporate an iterative improvement process; therefore there is a guarantee that a non-deteriorating solution will be produced.

crossover heuristics: widely used in evolutionary approaches, crossover or recombination heuristics take two solutions and combine them to produce an offspring solution.

Finally, HyFlex provides two control parameters (*intensity-of-mutation* and *depth-of-search*), which can be used to control the behavior of some of the provided heuristics. The precise functioning of the parameters would depend on the specific heuristic and problem domain. This information is available on the technical reports available for each problem domain [22, 23, 17, 37].

Currently, four problem domain modules are implemented (which can be downloaded from[1]): permutation flow shop, one-dimensional bin packing, maximum satisfiability (MAX-SAT) and personnel scheduling.

2.1 Permutation flow shop

In the permutation flow shop problem, a set of n jobs are to be processed in m machines. Jobs are processed first in $machine_1$, then in $machine_2$, and so on. The constraint is that the initial sequence of the jobs at $machine_1$ must be kept in all machines, i.e. jobs cannot jump other jobs. Additionally, no machine is allowed to remain idle when a job is ready for processing. All jobs and machines are available at time 0, and each job i requires a processing time on machine j denoted by p_{ij}.The most widely considered objective function is to minimise the length of the schedule, i.e. the makespan.

Initialisation: Solutions are created with a randomised version of the widely used NEH algorithm [27], which works as follows. First a random permutation of the jobs is generated. Second, a schedule is constructed from scratch by assigning the first job in the permutation to an empty schedule; the second job is then assigned to places 1 and 2 and fixed where the partial schedule has the smallest makespan; the third job is assigned to places 1, 2 and 3 and fixed to the place where the partial schedule has the smallest makespan, and so on.

Low-level heuristics: 14 low level heuristics are provided for this domain: specifically, 5 mutational, 2 ruin-recreate, 4 local search, and 3 crossover heuristics. The mutational and local search heuristics are inspired by those proposed in [30, 29]. The crossover heuristics are widely known recombination operators for the permutation representation. The ruin-recreate heuristics incorporate the successful NEH procedure in the construction process. For more details, see [37].

Objective function: The objective function to be minimised corresponds to the *makespan*, i.e., the overall completion time for all the jobs.

Instance data: The five instances used are taken from the widely known Taillard set [33]. We selected one instance of 100 jobs and 20 machines, and four instances of 500 jobs and 20 machines. The job processing times, p_{ij}, in all these instances are randomly generated integer numbers that follow a Uniform distribution in the range $1, \ldots, 99$.

[1] `http://www.asap.cs.nott.ac.uk/chesc2011/`

Table 1: Bin packing instances

instance	name and source	capacity	no. pieces
1	falkenauer/falk500-1 [18]	150	500
2	falkenauer/bpt501-1 [18]	100	501
3	schoenfield/schoenfieldhard1 [2]	1000	160
4	1000/10-30/instance1 [13]	150	1000
5	2000/10-50/instance1 [13]	150	2000

2.2 One dimensional bin packing

In the one-dimensional bin packing problem, a set of integer-size pieces, L, must be packed into bins of a given capacity C. The objective is to the minimise the number of bins used to pack the pieces. This can also be formulated as follows. A set of integers must be divided into the smallest possible number of subsets, so that the sum of the values in a subset does not exceed a given value C.

Initialisation: Solutions are initialised by a randomised version of the the widely known 'first-fit' heuristic [24]. First-fit packs the pieces one at a time, each into the first bin that they will fit into, opening a new bin when necessary.

Low-level heuristics: 7 low-level heuristics are available for this domain: specifically, 2 mutational, 2 ruin and recreate, repacked with best-fit, 1 crossover and 2 local search heuristics. For more details see [23].

Objective function: The objective function to be minimised favours bins that are filled completely, or nearly so (i.e. which have the least wasted space). It returns a value between zero and one, where lower is better, and a set of completely full bins would return a value of zero. For more details, see [23].

Instance data: The five problems used are summarised in Table 1. The first 3 instances can be downloaded from the 'EURO Special Interest Group on Cutting and Packing' website [2])

2.3 Maximum satisfiability (MAX-SAT)

The *propositional satisfiability problem* (SAT) can be formulated as follows. Given a formula in Boolean logic, decide whether there is an assignment of truth values to the variables in the formula under which the formula evaluates to 'true'. SAT is a decision problem. We consider here one of its related optimisation problems, the maximum satisfiability problem (MAX-SAT), in which the objective is to find the maximum number of clauses of a given Boolean formula, that can be satisfied by some assignment. The problem can also be formulated as a minimisation problem, where the objective is to minimise the number of unsatisfied clauses. This last formulation is the one considered in the HyFlex framework.

Initialisation: Solutions are initialised uniformly at random assigning a true or false value to each variable in the formula.

Low-level heuristics: 9 low-level heuristics are available for this domain: specifically: 2 mutational, 1 ruin and recreate, 2 crossovers, and 4 local search heuristics. The mutational and ruin-recreated heuristics correspond to simply flipping a number (which could be one) of randomly selected variables. The crossover operators are the standard one and two-point crossovers for binary representation. The local search heuristics are based on efficient algorithms from the literature: specifically, GSAT [32], HSAT [20], and WalkSAT [31]. See [22] for more details.

Objective function: The objective function to be minimised corresponds to the number of unsatisfied clauses.

[2] `http://paginas.fe.up.pt/~esicup/tiki-list_file_gallery.php?galleryId=1`

Table 2: MAX-SAT instances

instance	name	no. variables	no. clauses
1	uf250-01	250	1065
2	sat05-486.reshuffled-07	700	3500
3	blocksworld/huge	459	7054
4	flat200-1	600	2237
5	s2w100-2	500	3100

Table 3: Personnel scheduling instances

instance	name	staff	shift types	length (days)
1	BCV-1.8.2	8	5	28
2	BCV-3.46.1	46	3	26
3	BCV-A.12.2	12	5	31
4	ERRVH-B	51	8	48
5	MER-A	54	12	48

Instance data: The five problem instances were taken from 'SATLIB' [21]. They are summarised in table 2.

2.4 Personnel scheduling

In a personnel scheduling problem, decisions should be made about which times and on which days (i.e. which shifts) each employee should work over a specific planning period. Most of the personnel scheduling instances can be considered as a new and different problem rather than just a different instance. This is because most instances contain unique constraints and objectives, characteristics of the given organisation or workplace, and not just different instance parameters (such as the number of employees, shift types, planning period length, constraint priorities, etc). In consequence, implementing a problem domain module for personnel scheduling brings additional challenges. In HyFlex this was handled with a specially designed data file format with which each instance can select a combination of objectives and constraints from a wide choice.

Initialisation: Solutions are initialised using a local search heuristic with a neighbourhood operator that adds new shifts to the roster.

Low-level heuristics: 12 low level heuristics are provided for this domain: specifically, 1 mutational, 5 local search, 3 ruin and recreate, and 3 crossover heuristics. These heuristics are incorporated from previously proposed successful metaheuristic approaches to nurse rostering problems [4, 7, 8, 9]. For more details, see [17].

Objective function: The constraints are transformed to objectives with very high weights. As the weight is very high it is simple to tell if a solution is feasible or not just by examining the objective function value. The overall objective function (to be minimised) is a weighted sum of all the sub-objectives.

Instance data: The five problem instances were taken from the 'Staff Rostering Benchmark Data Sets' [16], which is a repository of diverse and challenging benchmark test instances from various sources including industrial collaborators and scientific publications. This is an interesting domain as most of the of the instances correspond to real world scenarios with different constraints and planning horizons. The five problem instances selected are summarised in Table 3. More details can be found in [17].

3. THE PROPOSED ALGORITHMS

Two adaptive algorithms are proposed. They extend, by incorporating an adaptive layer, the best performing heuristic presented in [6], which can be considered as an ILS algorithm with both multiple perturbation operators, and multiple hill-climbing (local search)

heuristics. Before discussing the adaptive variants, we describe below the baseline ILS algorithm.

3.1 The baseline ILS algorithm

The ILS implementation proposed in [6] contains a perturbation stage during which a neighborhood move is selected uniformly at random (from the available pool) and applied to the incumbent solution. This perturbation phase is then followed by an improvement phase, in which all local search heuristics are tested and the one producing the best improvement is used. If the resulting new solution is better than the original solution then it replaces the original solution, otherwise the new solution is simply discarded. This last stage corresponds to a greedy (only improvements) acceptance criterion. The pseudo-code of this iterated local search algorithm is shown below (Algorithm 1). It is worth mentioning that, during our implementation, alternative more sophisticated acceptance criteria were tested, such as simulated annealing and great deluge. However, the incorporation of these acceptance mechanisms did not improve the overall results, and they added additional control parameters that require tuning. Therefore, we opted for using the simple greedy acceptance mechanism.

Algorithm 1 *Iterated Local Search.*

s_0 = GenerateInitialSolution
s^* = LocalSearch(s_0)
repeat
  $s^{'}$ = Perturbation (s^*)
  $s^{*^{'}}$ = LocalSearch(s')
  if $f(s^{*^{'}}) < f(s*)$ **then**
    $s^* = s^{*^{'}}$
  end if
until time limit is reached

3.2 The adaptive ILS algorithms

The adaptive ILS algorithms proposed in this article substitute the uniform random selection of neighbourhoods in the perturbation stage, by online learning strategies. Specifically, we implemented the following strategies: *choice function* [14] (from the hyper-heuristics literature), and *extreme value based adaptive operator selection* [19] (from the evolutionary computation literature).

3.2.1 Choice function

The choice function was proposed in [15], as a method for selecting low-level heuristics in a hyper-heuristic. The method gathers and uses information about the performance of individual heuristics, the performance of pairs of heuristics when performed in succession, and on the amount of time that has elapsed since a heuristic has been called. The final selection mechanism uses these three measures, in a weighted sum, to determine the heuristic to choose.

The first, f_1, measures the recent success of a single low-level heuristic. This is measured by recording the change in objective function value. The immediate information from the last application of the heuristic is used in conjunction with information from further back to form the final value for f_1. The second, f_2, measures the effect of applying one heuristic after another. This captures greater information about the relationship between heuristics and allows for more intelligent heuristic selection. The final measure, f_3, is a simple measurement of the amount of time that has elapsed since a certain heuristic was last called.

These are brought together by means of a function, *F*, to give a final score. With this, the best performing heuristic will most likely be chosen. However, there is still a chance for worse heuristics to be chosen to allow for diversification. Once these values have been determined for each heuristic, it must be decided how to select a heuristic using the values. In [15], three methods are proposed. For our implementation, a roulette wheel choice was used. This works by effectively allocating a chunk of a 'roulette wheel' to each heuristic, with size proportional to that heuristic's value of *F*.

3.2.2 Extreme value based adaptive operator selection

As discussed in [19] an *adaptive operator selection* scheme consists of two components, described as *credit assignment* and *selection mechanism*. Credit assignment involves the attribution of credit (or reward) to variation operators, to be determined by its performance during the search process. Here, we used the scheme proposed in [19]: *extreme value* credit assignment, which is based on the principle that infrequent, yet large, improvements in the objective score are likely to be more effective than frequent, small improvements. Therefore, it rewards operators which have had a recent large positive impact on the objective score, while consistent operators that only yield small improvements receive less credit, and ultimately have less chance of being chosen. Following the application of an operator to the problem, the change in objective score is added to a window of size *W*, which works on a FIFO mechanism. The credit for any operator is the maximum score within the window. Window size plays an important part in the mechanism. If it is too small then the range of information on offer is narrowed, meaning that useful operators are missed. If it is too large then information is considered from many iterations ago, when the position in the search space might have meant that the operator performed differently to how it would at the latest iteration. However, the window size is the only parameter that needs to be tuned, which is a desirable property when the goal is to achieve robust and general algorithms. After testing several values of (W), we decided upon a value of 25.

The credit assignment mechanism is combined with a selection strategy that uses the accumulated credits to select the operator to apply in the current iteration. Operator selection strategies in the literature, generally assign a probability to each operator and use a roulette wheel-like process to select the operator according to them. We use here one of these rules, namely, *adaptive pursuit*, originally proposed for learning automata and adapted to the context of operator selection in [35]. With this method, at each time step, the operator with maximal reward is selected and its selection probability is increased (follows a winner-take-all strategy.), while the other operators have their selection probability decreased.

4. EXPERIMENTS AND RESULTS

Five problem instances were selected for each domain, as described in section 2 (Tables 1, 2, and 3). For each instance and algorithm variant, 10 runs were conducted, each lasting 10 CPU minutes. The experiments were conducted on a PC (running Windows XP) with a 2.33GHz Intel(R) Core(TM)2 Duo CPU and 2GB of RAM.

Three algorithm variants are compared: the baseline iterated local search implementation with uniformly at random selection of neighborhood operators: $Uniform$ (described in section 3.1), and two adaptive variants incorporating online learning mechanisms for operator selection, namely, choice function:*Adap-CF*, and extreme value based: *Adap-EV* (described in section 3.2). The exact same algorithms were used for each domain and instance, no domain-specific (or instance-specific) tuning process was applied.

The following subsections present our results from three different perspectives: (i) ordinal data analysis, (ii) distribution of best

objective function values, and (iii) percentage deviation of best-known solutions on the personnel scheduling domain.

4.1 Borda count

Since our study involves different domains and instances with varied magnitudes and ranges of the objective values, we selected ordinal data analysis to compare the alternative algorithms. If m is the number of instances (considering all the domains) and n the number of competing algorithms. For each experiment (instance) an ordinal value o_k is given representing the rank of the algorithm ($1 \leq o_k \leq n$). These methods can be used to compare the performance of competing search heuristics, as discussed in [34]. They aggregate and summarise m linear orders o_k into a single linear order O. We selected a straight forward ordinal aggregation method: the *Borda count* voting method, first purposed by Jean-Charles de Borda in 1770. An algorithm having a rank o_k in a given instance is simply given o_k points, and the total score of an algorithm is the sum of its ranks o_k across the m instances. The methods are, therefore, compared according to their total score, with the smallest score representing the best performing algorithm. In our comparative study, the number of instances, m, is 20 (5 for each domain). Therefore, for a given domain the best possible score is 5, while the best possible total score (considering all the domains) is 20. The ranks were calculated using as a metric the average best objective function (at the end of the run) obtained across the 10 runs per instance.

Figure 1 shows the total Borda scores for the three competing algorithms, including the total score per domain. The first point to note is that the two adaptive algorithms outperform the baseline ILS with uniformly at random operator selection. Secondly, the extreme value based mechanism produced a better total Borda score than the choice function method. It is worth mentioning that the extreme valued based method is a more recent approach, it has a single control parameter which may be an advantage over the choice function method that has three parameters that need to be tuned. However, as shown in Figure 1, the choice function produced the best results for the flow shop domain. We need to explore further why this is the case. One possible explanation is the fact that the flow shop instances all come from the same source (the Taillard set [33]) and they are generated using a similar procedure. It may be the case that the choice function parametrisation is well suited for this set of instances.

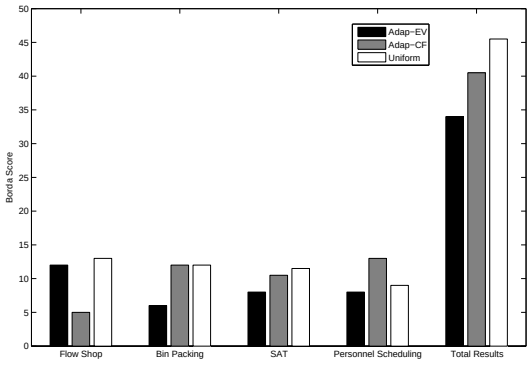

Figure 1: Borda results for all domains

Tables 4, 5, 6, and 7, show the Borda count (ranks) for each instance on the four domains, respectively. These results are useful to assess how homogeneous the results are for the five instances on each domain. For example, it can be noticed that for flow shop (Table 4), *Adap-CF* consistently ranks first on all the instances. We believe that this is due in part to the lack of diversity in the instance set for this domain. In comparison, the personnel scheduling domain, which provides instance data from a variety of sources, shows a less homogeneous distribution of ranks.

Table 4: Borda count results for flow shop

Instance	*Adap-EV*	*Adap-CF*	$Uniform$
1	3	1	2
2	2	1	3
3	2	1	3
4	2	1	3
5	3	1	2
Total	12	5	13

Table 5: Borda count results for bin packing

Instance	*Adap-EV*	*Adap-CF*	$Uniform$
1	1	2	3
2	2	3	1
3	1	2	3
4	1	3	2
5	1	2	3
Total	6	12	12

4.2 Distribution of the best objective function values

In addition to the Borda aggregation method presented in the previous section, boxplots 2, 3, 4, and 5, illustrate the magnitude and distribution of the best objective values (at the end of the run) for a selected instance of each domain. We arbitrarily took instance number 2 of each domain, but similar trends can be observed in the other instances.

Figures 3, 4, and 5 suggest that *Adap-EV* produces significantly better results on three of the tested domains, namely, bin packing, MAX-SAT, and personnel scheduling. With the performance differences being more marked on bin packing and personnel scheduling. The adaptive algorithm using the choice function mechanism, *Adap-CF*, was better for the flow shop domain (Figure 2). This deserves future investigation. In particular, we should explore whether increasing the variety of the instance data set would have an impact on the results.

4.3 Comparison against best-known solutions: personnel scheduling domain

Finally, in order to assess the performance of our best adaptive algorithm, Table 8 compares the best solution obtained by *Adap-EV* against the best-known solutions of the five instances of the personnel scheduling domain. We selected this domain as it contains real-

Table 6: Borda count results for MAX-SAT

Instance	*Adap-EV*	*Adap-CF*	$Uniform$
1	1	2	2
2	1	3	2
3	2	1	3
4	2	3	1
5	2	1	3
Total	8	10	11

Table 7: Borda count results for personnel scheduling

Instance	*Adap-EV*	*Adap-CF*	*Uniform*
1	1	3	2
2	1	3	2
3	2	3	1
4	1	2	3
5	3	2	1
Total	8	13	9

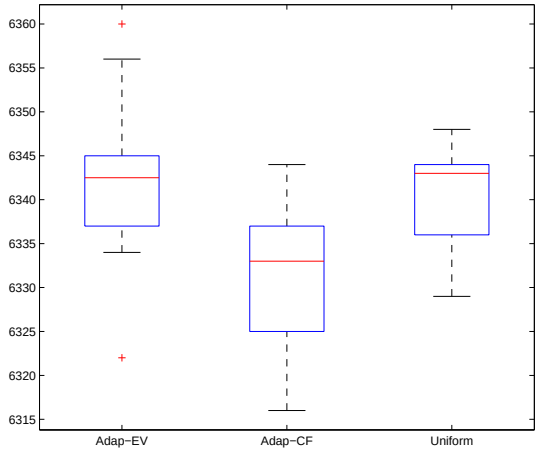

Figure 2: Distribution of the objective function values for the flow shop instance 2 (500 jobs and 20 machines).

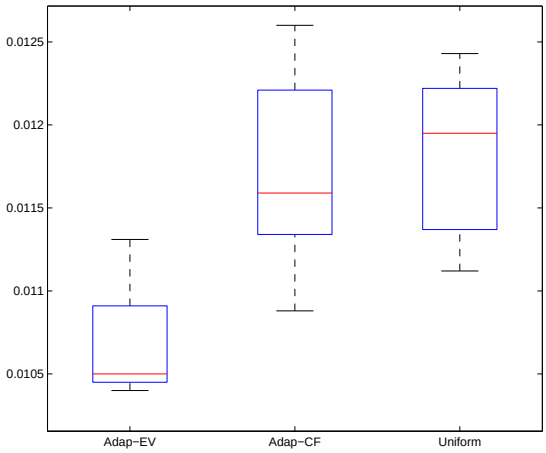

Figure 3: Distribution of objective function values for the bin packing instance 2: falkenauer/bpt501-1

world instances, and it is the least studied of the HyFlex domains. Moreover, with the exception of instance ERRVH-B, which is a relatively new addition to the the benchmark, the best-known solutions are available in the 'Staff Rostering Benchmark Data Sets' [16].

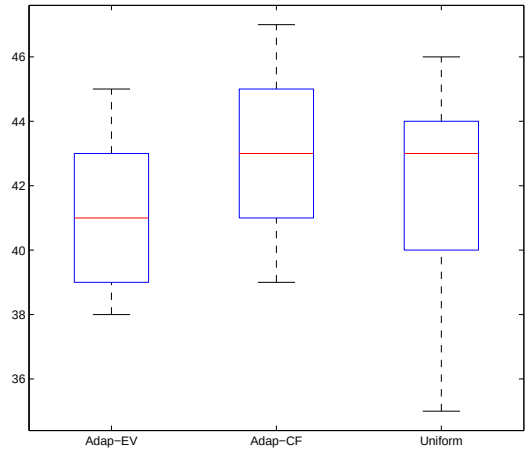

Figure 4: Distribution of objective function values for the MAX-SAT instance 2: sat05-486.reshuffled-07.

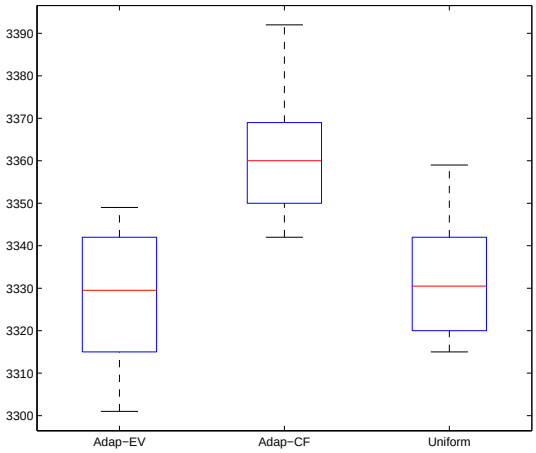

Figure 5: Distribution of objective function values for the personnel scheduling instance 2: BCV-3.46.1.

For the new instance ERRVH-B our solution represents the current best-known.

We were surprised to realise that our *Adap-EV* approach obtained a new best-known solution for instance MER-A (instance 4 in Table 8), it also matched a previous best-known result (instance 1). This is unexpected as our approach was designed as a general-purpose method. The favourable results can be explained as follows. First, the personnel scheduling domain module contains powerful move operators taken from state-of-the art metaheuristic approaches to nurse rostering problems [4, 7, 8, 9]. Second, the adaptive approaches proposed are able to learn on the fly and select an adequate operator at each decision point.

Table 8: Personnel scheduling best-known solutions

Inst.	Name	*Adap-EV*	*Best-known*	$\%deviation$
1	BCV-1.8.2	**853**	**853**	0
2	BCV-3.46.1	3301	**3280**	0.64
3	BCV-A.12.2	2003	**1953**	2.56
4	ERRVH-B	**3177**	**3177**	0
5	MER-A	**9888**	9915	-0.27

5. CONCLUSIONS

We have incorporated online adaptive operator selection techniques into a multiple neighborhood iterated local search algorithm. Using the HyFlex software framework, these adaptive algorithms were tested across four distinct hard combinatorial optimisation problems (permutation flows hop, 1D bin-packing, MAX-SAT, and personnel scheduling), with some of these domains including real-world instance data. Two online adaptive mechanisms, coming from distinct research communities, were implemented and compared. These were, the extreme value based adaptive operator selection technique from evolutionary algorithms, and the choice function from hyper-heuristics. Both of these techniques outperformed a baseline ILS implementation that selects uniformly at random among the available neighbourhood operators. The extreme value based mechanism produced the best overall results, which makes this algorithm the currently best performing hyper-heuristic implemented with the HyFlex framework.

We can see two clear possible extensions to this work. First, more sophisticated online learning mechanisms can be tested for selecting not only the perturbation operator, but also the local search heuristic in the improvement phase of the ILS algorithm. The HyFlex framework allows the implementation of population based and memetic algorithms. Therefore, an interesting direction would be to compare adaptive population based algorithms against single-point based ones. Second, the HyFlex framework can be extended in many ways to include new domains, additional instances and operators in existing domains, and multi-objective and dynamic problems. In particular, our study suggests that the instance data set for the flow shop domain should be extended to include more variety.

It is our vision that the HyFlex framework can be used as a benchmark for testing the robustness and generality of heuristic search methods, and thus can serve as a tool to promote further research in this area.

6. REFERENCES

[1] J. Baxter. Local optima avoidance in depot location. *Journal of the Operational Research Society*, 32:815–819, 1981.

[2] G. Belov and Guntram Scheithauer. A cutting plane algorithm for the one-dimensional cutting stock problem with multiple stock lengths. *European Journal of Operational Research*, 141(2):274–294, 2002.

[3] S. Bleuler, M. Laumanns, L. Thiele, and E. Zitzler. PISA—A Platform and Programming Language Independent Interface for Search Algorithms. In *Conference on Evolutionary Multi-Criterion Optimization (EMO 2003)*, volume 2632 of *LNCS*, pages 494–508, Berlin, 2003. Springer.

[4] E. K. Burke, P. Cowling, P. De Causmaecker, , and G. Vanden Berghe. A memetic approach to the nurse rostering problem. *Applied Intelligence*, 15(3):199–214, 2001.

[5] E. K. Burke, T. Curtois, M. Hyde, G. Kendall, G. Ochoa, S. Petrovic, and J. A. Vazquez-Rodriguez. HyFlex: A flexible framework for the design and analysis of hyper-heuristics. In *Multidisciplinary International Scheduling Conference (MISTA 2009)*, pages 790–797, Dublin, Ireland, August 2009.

[6] E. K. Burke, T. Curtois, M. Hyde, G. Kendall, G. Ochoa, S. Petrovic, J. A. Vazquez-Rodriguez, and M. Gendreau. Iterated local search vs. hyper-heuristics: Towards general-purpose search algorithms. In *IEEE Congress on Evolutionary Computation (CEC 2010)*, pages 3073–3080, Barcelona, Spain, July 2010.

[7] E. K. Burke, T. Curtois, G. Post, R. Qu, and B. Veltman. A hybrid heuristic ordering and variable neighbourhood search for the nurse rostering problem. *European Journal of Operational Research*, 188(2):330–341, 2008.

[8] E. K. Burke, T. Curtois, R. Qu, and G. Vanden Berghe. A time predefined variable depth search for nurse rostering. Technical report, School of Computer Science, University of Nottingham, 2007.

[9] E. K. Burke, T. Curtois, R. Qu, and G. Vanden Berghe. A scatter search methodology for the nurse rostering problem. *Journal of the Operational Research Society*, 61:1667–1679, 2010.

[10] E. K. Burke, M. Gendreau, M. Hyde, G. Kendall, B. McCollum, G. Ochoa, A. J. Parkes, and S. Petrovic. The cross-domain heuristic search challenge – an international research competition. In *International Conference on Learning and Intelligent Optimization (LION5)*, Lecture Notes in Computer Science. Springer, 2011. (to appear).

[11] E. K. Burke, E. Hart, G. Kendall, J. Newall, P. Ross, and S. Schulenburg. Hyper-heuristics: An emerging direction in modern search technology. In F. Glover and G. Kochenberger, editors, *Handbook of Metaheuristics*, pages 457–474. Kluwer, 2003.

[12] E. K. Burke, M. Hyde, G. Kendall, G. Ochoa, E. Ozcan, and J. Woodward. *Handbook of Metaheuristics*, volume 146 of *International Series in Operations Research & Management Science*, chapter A Classification of Hyper-heuristic Approaches, pages 449–468. Springer, 2010. Chapter 15.

[13] E. K. Burke, M. R. Hyde, G. Kendall, and J. Woodward. The scalability of evolved on line bin packing heuristics. In *Proceedings of the IEEE Congress on Evolutionary Computation (CEC 2007)*, pages 2530–2537, Singapore, September 2007.

[14] P. Cowling, G. Kendall, and E. Soubeiga. A hyperheuristic approach for scheduling a sales summit. In *Selected Papers of the Third International Conference on the Practice And Theory of Automated Timetabling, PATAT 2000*, LNCS, pages 176–190, Konstanz, Germany, 2000. Springer.

[15] P. Cowling, G. Kendall, and E. Soubeiga. A hyperheuristic approach for scheduling a sales summit. In *Selected Papers of the Third International Conference on the Practice And Theory of Automated Timetabling, PATAT 2000*, LNCS, pages 176–190. Springer, 2000.

[16] T. Curtois. Staff rostering benchmark data sets. Website, 2009. `http:///www.cs.nott.ac.uk/~tec/NRP/`.

[17] T. Curtois, G. Ochoa, M. Hyde, and J. A. Vázquez-Rodríguez. A hyflex module for the personnel scheduling problem. Technical report, School of Computer Science, University of Nottingham, 2011.

[18] E. Falkenauer. A hybrid grouping genetic algorithm for bin packing. *Journal of Heuristics*, 2:5–30, 1996.

[19] A. Fialho, L. Da Costa, M. Schoenauer, and M. Sebag. Extreme value based adaptive operator selection. In *Parallel Problem Solving from Nature (PPSN X)*, volume 5199 of *Lecture Notes in Computer Science*, pages 175–184. Springer, 2008.

[20] I. Gent and T. Walsh. Towards an understanding of hill-climbing procedures for sat. In *Proceedings of the 11th National Conference on Artificial Intelligence (AAAI'93)*, pages 28–33, Washington D.C., USA, July 1993.

[21] H. H. Hoos and T. Stützle. Satlib: An online resource for research on sat. In I. P. Gent, H. V. Maaren, and T. Walsh, editors, *SAT 2000*, pages 283–292. IOS Press, 2000. SATLIB is available online at www.satlib.org.

[22] M. Hyde, G. Ochoa, T. Curtois, and J. A. Vázquez-Rodríguez. A hyflex module for the boolean satisfiability problem. Technical report, School of Computer Science, University of Nottingham, 2011.

[23] M. Hyde, G. Ochoa, T. Curtois, and J. A. Vázquez-Rodríguez. A hyflex module for the one dimensional bin-packing problem. Technical report, School of Computer Science, University of Nottingham, 2011.

[24] D. Johnson, A. Demers, J. Ullman, M. Garey, and R. Graham. Worst-case performance bounds for simple one-dimensional packaging algorithms. *SIAM Journal on Computing*, 3(4):299–325, December 1974.

[25] H. R. Lourenco, O. Martin, and T. Stutzle. *Iterated Local Search*, pages 321–353. Kluwer Academic Publishers,, Norwell, MA, 2002.

[26] O. Martin, S. W. Otto, and E. W. Felten. Large-step Markov chains for the TSP incorporating local search heuristics. *Operations Research Letters*, 11(4):219–224, 1992.

[27] M. Nawaz, E. Enscore Jr., and I. Ham. A heuristic algorithm for the m-machine, n-job flow-shop sequencing problem. *OMEGA-International Journal of Management Science*, 11(1):91–95, 1983.

[28] P. Ross. Hyper-heuristics. In E. K. Burke and G. Kendall, editors, *Search Methodologies: Introductory Tutorials in Optimization and Decision Support Techniques*, chapter 17, pages 529–556. Springer, 2005.

[29] R. Ruiz and T. G. Stützle. An iterated greedy heuristic for the sequence dependent setup times flowshop problem with makespan and weighted tardiness objectives. *European Journal of Operational Research*, 187(10):1143–1159, 2007.

[30] R. Ruiz and T. G. Stützle. A simple and effective iterated greedy algorithm for the permutation flowshop scheduling problem. *European Journal of Operational Research*, 177:2033–2049, 2007.

[31] B. Selman, H. Kautz, , and B. Cohen. Noise strategies for improving local search. In *Proceedings of the 11th National Conference on Artificial Intelligence (AAAI'94)*, pages 337–343, Seattle, WA, USA, July 1994.

[32] B. Selman, H. Levesque, and D. Mitchell. A new method for solving hard satisfiability problems. In *Proceedings of the 10th National Conference on Artificial Intelligence (AAAI'92)*, pages 440–446, San Jose, CA, USA, July 1992.

[33] E. Taillard. Benchmarks for basic scheduling problems. *European Journal of Operational Research*, 64(2):278–285, 1993.

[34] E. G. Talbi. *Metaheuristics: from design to implementation*. Wiley, 2009.

[35] Dirk Thierens. An adaptive pursuit strategy for allocating operator probabilities. In *Proceedings of the 2005 conference on Genetic and evolutionary computation*, GECCO '05, pages 1539–1546, New York, NY, USA, 2005. ACM.

[36] Dirk Thierens. Adaptive operator selection for iterated local search. In *Engineering Stochastic Local Search Algorithms. Designing, Implementing and Analyzing Effective Heuristics*, volume 5752 of *Lecture Notes in Computer Science*, pages 140–144. Springer Berlin / Heidelberg, 2009.

[37] J. A. Vázquez-Rodríguez, G. Ochoa, T. Curtois, and M. Hyde. A hyflex module for the permutation flow shop problem. Technical report, School of Computer Science, University of Nottingham, 2011.