A re-characterization of hyper-heuristics

Jerry Swan, Patrick De Causmaecker, Simon Martin, Ender Özcan

Abstract Hyper-heuristics are an optimization methodology which 'search the space of heuristics' rather than directly searching the space of the underlying candidate-solution representation. Hyper-heuristic search has traditionally been divided into two layers: a lower problem-domain layer (where domain-specific heuristics are applied) and an upper hyper-heuristic layer, where heuristics are selected or generated. The interface between the two layers is commonly termed the "domain barrier". Historically this interface has been defined to be highly restrictive, in the belief that this is required for generality. We argue that this prevailing conception of domain barrier is so limiting as to defeat the original motivation for hyper-heuristics. We show how it is possible to make use of domain knowledge without loss of generality and describe generalized hyper-heuristics which can incorporate arbitrary domain knowledge.

This is a preprint — please cite as:

```
@incollection{recharacterizationhh,
    author = {Swan, Jerry and De Causmaecker, Patrick
        and Martin, Simon and \"{O}zcan, Ender},
    title = {A re-characterization of hyper-heuristics},
    editor = {L. Amodeo, E-G. Talbi, F. Yalaoui},
    booktitle = {Recent Developments of Metaheuristics},
    publisher = {Springer},
    year = {2016 (to appear)}
}
```

1 Introduction

Sörensen and Glover [1] define a *metaheuristic* as "a high-level problem-independent algorithmic framework that provides a set of guidelines or strategies to develop heuristic optimization algorithms". The goal of *hyper-heuristics* is to act as effective cross-domain search methodologies, i.e. to be applicable not only to

Jerry Swan
Computer Science, University of York, UK. e-mail: jerry.swan@york.ac.uk

Patrick De Causmaecker
Computer Science, KU Leuven, NL. e-mail: patrick.decausmaecker@kuleuven.be

Simon Martin
Computer Science, University of Stirling, UK. e-mail: spm@cs.stir.ac.uk

Ender Özcan
Computer Science, University of Nottingham, UK. e-mail: ender.ozcan@cs.nott.ac.uk

instances with different characteristics from a single domain, but also across multiple problem domains. The definition of a hyper-heuristic varies considerably in the literature: indeed, interpreting the notion of 'searching the space of heuristics' in full generality allows application to *any* heuristically-informed solution mechanism (e.g. the choice of pivot function used in Quicksort [2]). In this article, we concentrate on the application of hyper-heuristics to metaheuristic search. In the following sections, we describe how the definition of hyper-heuristics has evolved over time. We re-visit the underlying motivation in order to highlight some popular misconceptions and the attendant need for re-characterization.

1.1 Historical development of hyper-heuristics

One of the earliest studies in this area was an application to a job shop scheduling problem due to Fisher and Thompson [3]. The use of scheduling (dispatching) rules as heuristics is common in this area and the study was motivated by the idea of "a combination of the two rules being superior to either one separately". In the early 1990s, Storer et al [4, 5] proposed a general approach combining heuristic- and solution- space methods for solving sequencing problems. The authors used job shop scheduling as a case study and argued that the proposed approach can be "easily" applied to any scheduling objective. Fang et al. [6, 7] subsequently evolved sequences of heuristics for constructing schedules, explaining how the proposed approach can be "simply amended" to deal with more complex industrial open shop scheduling problems.

The term 'hyperheuristics' (in unhyphenated form) was first introduced by Cowling et al. [8] as a means of deciding which low level heuristic to apply during the search process, depending on the nature of the region being explored. This initial definition referred only to (what has become known as) 'selective' hyper-heuristics, with generative hyper-heuristics being a later development [9]. The motivation for the use of the term 'hyper' comes from hypergraphs, where an edge is an n-ary relation on vertices, the analogy being that hyper-heuristic selection is performed on a *collection* of operators (i.e. functions with signature $Op : S \to S$, for some candidate solution representation S). Hyper-heuristic selection thus takes a list of operators, together with a function for choosing an operator from this list and applies the selected operator to an incumbent state. Mathematically, we can represent this as:

$$\text{select} : [Op] \times ([Op] \to Op) \times S \to S$$
$$\text{select} : (operators, choose, incumbent) \mapsto choose(operators)(incumbent)$$

If, as has invariably been the case, the list of operators and the function for choosing from them are known in advance, then the signature for select can be considered to be $S \to S$, precisely that of an operator. This notion of 'recursive composition via selection' [10] could equally be applied to other metaheuristic components (i.e. acceptance, termination etc), though the authors are not aware of any such approaches

(e.g. the evolution of acceptance criteria by Hyde et al [11] was obtained via a *generative* rather than selective approach).

Cowling et al. [8] stated that a hyper-heuristic approach operates at 'a higher abstraction level' than a metaheuristic and in practice this has been translated as 'operating independently of the underlying problem domain'. To this end, Cowling et al. [8] introduced the notion of a *domain barrier* between the layers of hyper-heuristic framework and problem-domain implementation. As we explicitly demonstrate in Section 2, this notion of domain independence can be described purely in terms of *generic metaheuristics*, avoiding the rather mixed collection of concepts that has become associated with hyper-heuristics (see also Figure 1).

The widely-cited definition due to Burke et al. [12] of "(meta-)heuristics to choose (meta-)heuristics" has the stated motivation of "raising the level of generality at which optimisation systems can operate". The authors contemplate that many businesses, particularly small ones, are interested in "good enough, soon enough, cheap enough" solutions to their problems. Given the high cost of developing problem-specific methods, this highlights the need for a general, easy-to-use, yet robust approach for 'providing near optimal solutions'. The intention was that the domain barrier represents the separation between different levels of expertise, i.e. practitioners would be responsible for implementing only solution representations and naïve 'knowledge-poor' (and hence presumably often randomized) heuristics for each new problem domain, with researchers tasked with devising hyper-heuristics which work well across domains.

Ross et al. [13, 14] defined a hyper-heuristic as a search method which combines simple heuristics to solve a range of problems satisfactorily. They evolved bin-packing rules using a Learning Classifier System [13] which learns which low level heuristic to use at a given decision point. Ross [9] provided a similar definition as Soubeiga [15] introducing hyper-heuristics as "heuristics to choose heuristics" and combining multiple heuristics to compensate for individual weaknesses. In a foundational paper on generative hyper-heuristics, Ross [9] further proposed hyper-heuristics as a special form of genetic programming, with a function set consisting of existing heuristics. In this study, the aim in hyper-heuristic design is presented as finding a "fast, reasonably comprehensible" approach, repeatedly able to produce high quality solutions. Qu et al. [16] proposed a tabu search approach to examination timetabling and graph colouring problems, indirectly acting on the candidate solutions via a mixture of graph-colouring heuristics. Two cross-domain heuristic search competitions, CHeSC 2011 and 2014 were performed using the HyFlex selective hyper-heuristics framework [17], which provided an implementation of six problem domains.

A recent definition of hyper-heuristic which is probably the most commonly-used is provided by Burke et al. [18] as "a search method or learning mechanism for selecting or generating heuristics to solve computational search problems". A more concrete definition adopts the terminology of the 'Algorithm Selection Problem' (ASP) [19] to describe hyper-heuristics as 'a mapping from features to algorithms'. A rich research area that has historically been more overtly influenced by the ASP than hyper-heuristics is the field of algorithm portfolios [20, 21]. Recent work in this

field includes 'Dynamic Algorithm Portfolios' [22] which chooses from a subset of available algorithms, applying them simultaneously to a problem instance until the fastest algorithm solves it.

In principle, the adoption of the ASP would allow the gamut of machine learning techniques to be applied to hyper-heuristics, but in practice the features made available for learning by selective hyper-heuristics have been limited. In contrast, the input to *generative* hyper-heuristics is (necessarily) domain-specific, and the only general framework supporting generative hyper-heuristics which we are aware of is TEMPLAR [23]. Chakhlevitch and Cowling [24] specifically argue the importance of limited problem domain information in achieving cross-domain generality for selective hyper-heuristics. Moreover, they further state that a hyper-heuristic would ideally be informed only of the number of low level heuristics for a given problem domain and objective value of a given solution. A variant of the strict notion of domain barrier due to Woodward et al [25] has been perpetuated via HyFlex as a *de facto* standard.

As can be seen from the above, the definition of hyper-heuristic has evolved considerably over time. As a result, there is relatively little clear consensus on what the essential mechanisms of a hyper-heuristic actually are. Figure 1 is a feature diagram of various concepts historically associated with (selective) hyper-heuristics. The concepts which are non-obvious (or otherwise not covered above) are:

- Heterogeneous operators: The ability to treat different operators in a uniform manner in the hyper-heuristic layer. For example, with a permutation representation, the ability to mix e.g. 2-opt with transpositions.
- Selection *a posterioi* versus *a priori*: whether or not an operator must be applied (to the current incumbent solution) before it can be chosen. Metaheuristics are traditionally, 'apply then choose', e.g. choose the first- or best- improving. The *a priori* case is when an operator is chosen via some mapping based on its features and the search trajectory.

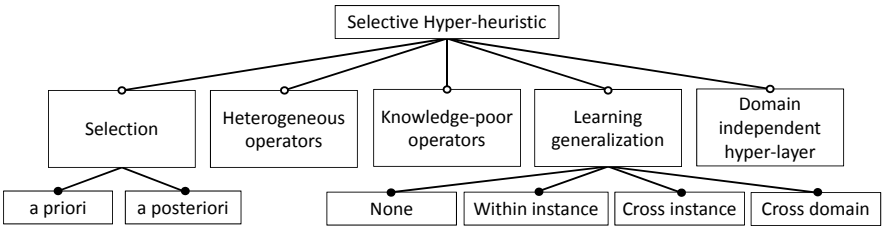

Fig. 1: Concepts historically associated with selective hyper-heuristics

Despite the diversity of concepts associated with hyper-heuristics, the authors are aware of only a few attempts to consolidate them. It is also interesting to note that some conceptual and formal approaches that might reasonably be included under the

wider notion of 'heuristics to select or generate heuristics' (e.g. [26, 27]) have not historically been considered to be part of the literature. As discussed above, selective hyper-heuristics can be shown to be an instance of the well-known 'Composite' design pattern [10], a mechanism used by the HYPERION framework [28, 29] to allow the same source code to express both metaheuristics and hyper-heuristics. The widely-cited classification scheme due to Burke et al [30] is generalized by Swan et al [31] to allow any combination of selective/generative and online/offline to co-exist and interoperate at runtime within the same architecture.

1.2 Effectiveness in new domains

It has been observed that not only the design of a selective hyper-heuristic but also the choice of predefined low level heuristics influences its performance [32]. To the best of the authors' knowledge, there are no applications of selective hyper-heuristic for which the use of only 'knowledge poor' low-level heuristics is competitive with the state-of-the-art. In practice, state-of-the-art low level heuristics have therefore made their way into the domain implementations for improved performance, e.g. in several of the problem domains implemented by HyFlex [17].

This indicates that (selective) hyper-heuristic research has become disconnected from the original motivation, failing to provide solutions which are 'good enough, cheap enough' (and in general certainly not the 'near optimal' solutions which were originally hoped for). Due to the artificially-restricted notion of the domain barrier, devising and using selective hyper-heuristics is currently no less labour-intensive than simply using some generic metaheuristic framework (detailed reasons for this are given in Section 2). For researchers, it surely is clear that the application of machine learning (e.g. [33]) is necessary to avoid cross-domain generalization being obtained via laborious manual 'generate and test'. However, the *de facto* domain barrier restrictions mean that even the elaborate machine learning techniques that have been employed in selective hyper-heuristics tend to make use of limited information. There is therefore a need to move from this restrictive interface to one which:

- Enables more expressive (i.e. feature-rich) hyper-heuristics.
- Allows state-of-the art knowledge to be easily incorporated into a new problem domain model by less-experienced practitioners.

To achieve this, it is necessary to disentangle approaches which have become prevalent from the goals they sought to achieve. This is particularly important since none of the concepts of Figure 1 suffice to fully-characterize the many publications with 'hyper-heuristic' in the title. To reiterate: applying hyper-heuristics would be of interest to practitioners (e.g. in industry) if this avoids the need for labour-intensive modelling of a specific problem domain. However, metaheuristic search is *already* a computational intelligence success story in this respect: approaches such as simulated annealing, tabu search, genetic algorithms and swarm optimization yield good

(and often state-of-the-art) results in a wide range of problem domains. The minimal requirement for domain modelling using these techniques is very small, needing only a choice of solution representation, solution quality measure and one or more operators for perturbing/recombining solutions.

Depending on the available modelling budget, the sophistication of domain knowledge can range from the very naïve (e.g. potentially infeasible solutions; randomized operators, quality measure that does not yield a search gradient) through to a highly-informed combination of state-of-the-art techniques. The important point relative to selective hyper-heuristics is that, in this case, practitioners retain the option to increase the competence of the framework layer as required. In the next section, we show how the popular conception of selective hyper-heuristics can be viewed as a (somewhat uninformed) special case of *generic metaheuristics*.

2 Popular notion of the domain barrier

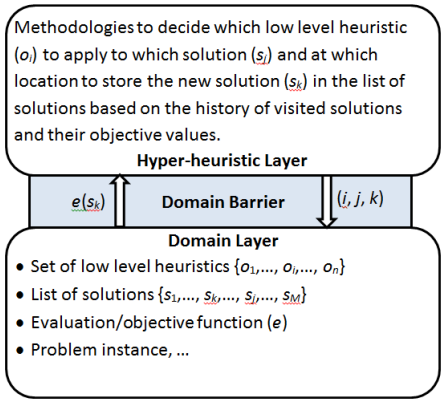

Fig. 2: A popular conception of selective hyper-heuristics

The *de facto* conception of selective hyper-heuristics (e.g. as exemplified by Hyflex [17]) is shown in Figure 2. Here the notion of 'heuristic' is restricted to that of 'operator', i.e. a perturbation of a candidate solution. Hyflex operates as follows: the hyper-heuristic solver maintains a list of heuristics $[o_1, \ldots, o_n]$ and a list of solutions $[s_1, \ldots, s_m]$. The heuristic value of a solution s_k is given by $e(s_k)$. At each iteration, the solver chooses three integers i, j, k such that the solution in slot k is replaced by the result of applying operator i to solution j. This is the characterisation given by Woodward et al [25] as:

$$Sel : \mathbb{N} \times \mathbb{N} \times \mathbb{N} \to \mathbb{R} \tag{1a}$$
$$Sel : (i, j, k) \mapsto e(o_i(s_k)) \tag{1b}$$

with solution k having first being replaced by $o_i(s_j)$ as a side-effect. The only information available to the solver in making this choice is the fitness value/execution time resulting from operator application (and any memoisation of such information from previous iterations). Problem-specific information is hidden in the belief that the generality of the hyper-heuristic will be lost. This can be considered as a 'lowest-common-denominator' approach to generality.

It should be immediately clear that this formulation is too restrictive to allow many popular metaheuristics to operate hyper-heuristically (not least) because of the following common requirements:

1. The ability to compare solutions for domain-specific equality. This arises since solution representations are only visible to the hyper-layer as integer indices. The result is that even such elementary techniques as 'breadth-first search' cannot be expressed.
2. The ability to determine those parts of a solution that have been changed by a heuristic. This is a common requirement in tabu-style approaches (e.g. making recently-perturbed permutation indices tabu in the TSP).
3. The ability to detect and react to constraint violations (e.g. infeasible solutions) at the framework level.

The inability to test for equality also precludes approaches such as reactive tabu or breakout local search [34, 35], which explicitly maintain equality-based solution histograms in order to determine when a diversification strategy should be triggered.

For clarity, we now make explicit the difference between metaheuristics and selective hyper-heuristics (as defined by the 'traditional' Hyflex-style domain barrier). Consider a generic framework for a local search metaheuristic shown in Listing 1. The framework is parameterized by the type S of solution representation and the type F denoting the features to be memoized in the search history. $[T]$ denotes a list of elements of type T and operators Op are functions $S \to S$. The memoized features are used for decision-making during the selection process and are obtained via a mapping $features : Op \times S \times S \to F$.

```
S search(incumbent: S, operators: [Op], history: [F]) {
    while(not finished(incumbent, history)) {
        Op op = selectOp(incumbent, operators, history);
        S incoming = op(incumbent);
        incumbent = accept(incumbent,incoming)
        history.update( features(op, incumbent, incoming) );
    }
    return incumbent;
}
```

Listing 1: Generic local search meta- or hyper- heuristic

To instantiate this framework as a metaheuristic (e.g. simulated annealing for the TSP), we put:

- S as a permutation.
- $[Op]$ as any desired perturbations e.g. transpositions, 2-opt etc.
- *selectOp* as uniform random selection.
- *accept* as Metropolis-Hastings.
- Feature type F to be the empty set (in this particular case).

To instantiate this framework as a Hyflex-style hyper-heuristic, we put:

- S as an integer in the range $[0,m)$.
- Op as an integer in the range $[0,n)$.
- *selectOp* as e.g. choice function [36].
- Feature type F to be $(Op,S,S,\mathbb{R})$, given by $(i,j,k,e(s_k))$ from Eq. (1a), above.

It should be clear that the same arguments as given above apply to any other metaheuristic, population-based or otherwise. Hyflex-style selective hyper-heuristics can therefore be seen as a special case of a metaheuristic in which solutions and operators are mapped onto opaque integer indices at the framework level. It should be clear from the fact that the framework level is *already generic in terms of solution representation*, that there is absolutely no requirement for this degree of opacity: we can instantiate the hyper-heuristic framework with arbitrary solution and operator types, thereby eliminating the above issues (equality of states etc) associated with the opaque handles of Hyflex. In addition, such types can provide much greater utility without loss of generality, e.g. the ability to decompose a solution into parts to act as finer-grained tabu attributes.

3 The need for 'domain-independent domain knowledge'

As explained in the previous section, if one adopts the prevailing notion of the domain barrier, then the minimal responsibilities of a practitioner in implementing some new domain are precisely the same *irrespective of whether they wish to use metaheuristics or hyper-heuristics*. In fact, they are considerably worse off with the latter approach: if search quality is unsatisfactory, then there is no means of 'injecting further domain knowledge' at the framework level as can be done with a metaheuristic. What is therefore needed is a hyper-heuristic approach which can operate on much richer domain knowledge.

To illustrate the extent to which this is possible, it is useful to consider the distinction between 'analytic' and 'empirical' knowledge. We define the former to be information which is given *a priori* (or otherwise formally derived from) the problem description and the latter to be that derived from the solution trajectory. The *de facto* conception of hyper-heuristics is that they can only make use of empirical knowledge at the framework level. The empirical features exposed by HyFlex are:

- Objective value arising from applying $o_i(s_k)$.

- Execution time for operator application.
- Integer handle of an operator.

Given this limited set of features, it is difficult to learn useful information even *within a domain*. Handles denoting operators have no persistent meaning across problem domains and the possibilities for cross-domain learning are therefore even more limited. While it is encouraging to see that extensions to the *de facto* conception of the domain barrier have recently been proposed [37], it is possible to go much further than this, as we now demonstrate.

A recent machine learning approach of Asta and Özcan [33] (in which linkage between operators is estimated via tensor factorization) is perhaps representative of the limits of what might be learned from the kind of empirical information described above. In contrast, in many cases we do not need to mine information of this form from the solution trajectory: we already have it as prior knowledge. Consider the ability to reason algebraically about operators. For example, it is well known that transpositions of permutations are self-inverse, so it is wasteful to apply the same transposition in succession. This is of course an extremely simple example: there is no limit to the kinds of exploitable analytic information we might devise. As a more general example, *any* sequence of operators (of any sort, as long as they have signature $S \rightarrow S$) forms an algebraic structure known as a *monoid* under concatenation. This monoid structure can be represented by equations between operators that describe which sequences always lead back to their starting state (irrespective of the specific start state). As shown by Swan et al [38], it is often possible to use this algebraic relationship between operators to derive a set of *rewrite rules*. These rewrite rules allow any sequence of operators to be reduced *a priori* to its minimal-length equivalent, thereby eliminating redundancy (i.e. cycles) in the state-space graph. In particular, this is an example of cross-domain analytic knowledge: Swan et al apply this to the Quadratic Assignment Problem, but the equational description of the associated monoid structure could be used in any problem which operates on permutations.

While this cannot be achieved hyper-heuristically with a HYFLEX-style formulation, making the additional information (in this case the monoid equations) available to a hyper-heuristic solver has absolutely no cost in terms of generality: a solver which is incapable of acting on such information can simply ignore it. The challenge is therefore to exploit such information without loss of generality as 'domain-independent domain knowledge'. Although the notion of being able to operate on arbitrary features has always been implicit in the ASP[1], the vast majority of work in selective hyper-heuristics has been concerned with the limited feature set described above.

The need to move away from such unnecessary restrictions then leads immediately to considerations of knowledge-representation, which have fortunately been well-studied in the wider AI community for many years. Two examples of existing hyper-heuristic systems which can (in terms of their architectural principles) incorporate arbitrary domain knowledge are the blackboard system of Swan et al [31]

[1] and to a less overt degree implied in early work on hyper-heuristics (e.g. [12])

and the multi-agent system of Martin et al [39]. Although aspects of their architectures differ, they both have the ability to associate competent, representation-aware algorithms (known as 'knowledge sources' or 'agents', respectively) with heterogeneous sources of information, and it is this which allows these frameworks to operate across domains. The actual association process (which is precisely a mapping from features to algorithms) might range from the relatively trivial (e.g. a dictionary of key-value pairs indicating that a particular algorithm is competent to operate on permutation representations) to more specialized condition-action patterns induced from any source of information (analytic or empirical) that the agent is able to recognize. In the next section, we discuss the use of constraint satisfaction as a generic vocabulary for expressing domain-independent domain knowledge.

4 Cross-domain knowledge representation

We now elaborate on the desired nature of knowledge representation for use in a hyper-heuristic framework capable of generalized cross-domain learning. Such a generalized representation should ideally allow for the expression of problem specifications and the description of low-level heuristics, together with formal properties of those problems and heuristics. What we therefore require is a description in a problem-independent vocabulary: such a representation would explicitly support cross domain learning. We may then rely on analytic knowledge of widely-used problem representations such as graphs and tensors, or reductions to well-known problems. In principle the hyper-heuristic could access such a specification in any detail. In this respect, a hyper-heuristic need not differ from a metaheuristic specialised to some problem domain. As discussed above, the main goal of hyper-heuristic research is to act effectively in a newly-specified domain without relying on the presence of optimization experts. Analytic knowledge allows us to better achieve this by injecting richer domain knowledge into the search process. In particular, it can contain rules about operator applications which are known *a priori* to result in improvement. An well-known example is the uncrossing of edges in the TSP:

detect : $findcross$

$\exists$ segment (A_p, A_{p+1}) and (A_{p+i}, A_{p+i+1}) in cycle $\{A_i | i = 1 \dots n\}$

$s.t. |A_pA_{p+1}| + |A_{p+i}A_{p+i+1}| > |A_pA_{p+i}| + |A_{p+1}A_{p+i+1}|$

action : *uncross*

replace $(A_p, A_{p+1}) \rightarrow (A_p, A_{p+i}), (A_{p+i}, A_{p+i+1}) \rightarrow (A_{p+1}, A_{p+i+1})$

and reverse $\{A_{p+1} \dots A_{p+i}\}$.

Such representations could also contain statements about certain relationships or constraints which are (nearly) always satisfied. A domain expert may wish to express his experience that certain patterns never contribute to good solutions. A suit-

able knowledge representation formalism will permit efficient handling of such expressions. One possible means of expressing cross-domain knowledge is constraint programming, which provides a well-established means for describing such generalised problems. Constraint satisfaction problems have the advantage of being declarative, i.e. allow the statement of constraints in an implementation-independent manner. Specifying patterns (in various forms) is common to many constraint languages and properties of low level operators and the relationships between them can readily be formulated in this manner. The detailed description of the workings of those operators may still proceed in any language. This implementation may moreover differ from one domain to another. Given an ontology of domain-independent concepts, it is very natural to express the transferable knowledge in terms of constraints.

To make these issues concrete, we proceed to discuss the features of a specific knowledge representation format. XCSP [40, 41, 42] allows expression of constraint satisfaction problems (CSP), weighted constraint satisfaction problems (WCSP) and quantified constraint satisfaction problems (QCSP). It allows the description of instances according to characteristics such as real-world; patterned; random instances with/without a structure or involving only boolean variables. The authors furthermore distinguish instances based on whether they are defining all constraints in extension, partly in intension or use global constraints. Designed for expression of general decision and optimization problems, it supports two notations: a full XML notation compromising between human and machine readability and an abridged notation which is much more human readable. Both notations are equivalent and translation in both directions is possible.

The XCSP specification of a problem domain for a hyper-heuristic can therefore contain information about problem properties and low level heuristics. Information on low level heuristics in present hyper-heuristic frameworks include categorizations such as 'hill climber' and 'population based' [37]. As argued elsewhere in this paper, it is possible to extend this much further: since XCSP allows for the introduction of arbitrary problem and data descriptions, then in principle any information about a low level heuristic that has been acquired should be expressible, in principle to any detail. For example, current hyper-heuristic practice requires that determining when it might be appropriate to call one heuristic immediately following another is achieved empirically. In many cases, this information is already available analytically, e.g. in the manner which is exploited in the 'Reverse Elimination Method' of tabu search [43].

Presently, tools are available for solving, parsing, checking of instances and solutions and shuffling variables (to check robustness of solvers). Since the format is open and has systematic parsers available, other tools may be conceived: one might think of discovering patterns in instances or history, item set mining and constraint learning, which could serve to summarize XCSP descriptions and keep them fit for use by on-line hyper-heuristics. Links between data mining and constraint programming have been suggested before [44], and ongoing integration between the two could support a fundamental move from pure model-based problem solving algorithms to an integrated, data-driven approach. In this extended representation,

instances and algorithm components (low level heuristics) would be described together with the conventional model. In a sense, this is a natural evolution, given the decades of algorithm design guided by tests on benchmark instances. This combined representation would describe problem, instances and history in one comprehensive format.

All this may change our vision of how combinatorial problems are described, with constraint languages providing models for the problem in the conventional sense, together with information on how such problems could be solved. This information, in the conventional hyper-heuristics paradigm, is specified in terms of low level heuristics, but predominantly provided declaratively by the research community.

5 Future Directions - The role of ontologies

Ontologies codify knowledge in order to drive (traditionally formalized) reasoning processes. Their first recorded use dates back to Aristotle [45]. In computer science, the graph-based semantic nets of Quillan and Simmons [46] and Minsky's frame systems provided a foundational definition of entities in terms of specialization and part/whole relations [47]. These approaches subsequently spawned a multiplicity of variants (e.g. [48]). The previous section framed knowledge representation for hyper-heuristics in the vocabulary of constraint satisfaction. By expressing constraints as relations, such representations clearly have an equivalent representation as graphs, hypergraphs or RDF-triples. The use of an ontology for scheduling and routing problems can be seen in Martin et al [39, 49]. Recently, the Resource Description Framework (RDF) of the Semantic Web [50] has emerged as a common ontological basis for knowledge exchange. One important feature of web ontologies is that knowledge can be hierarchically constructed by referencing (the definitions for) other knowledge elements through a web-hosted URI.

What makes this relevant for hyper-heuristics is the associated support for the discovery, aggregation and substitution of uniquely-identifiable knowledge elements in the form of problem and algorithm descriptions. The goal then is that practitioner activity moves from 'under-the-hood' software development to the use of tools to hybridize pre-existing declarative specifications or else tweak constraints. Ontologies provide further support for the large-scale vision of hyper-heuristics, consisting of online data repositories containing discovered rules, patterns and constraints which describe good solution approaches. A suitable choice of knowledge representation elements (e.g. based on XCSP as above and/or other interoperability standards such as OpenMath [51]) can form the basis of community investment in such cross-domain learning tools. This is to be contrasted with the more isolated and domain-specific development that typically takes place in today's research settings.

6 Conclusion

We have traced the historical development of hyper-heuristics and highlighted the motivating division of responsibility: hyper-heuristic researchers are responsible for devising methods which work well across domains, with the goal of allowing practitioners to invest minimal effort in modelling a new domain. A requirement for 'lifelong, cross-domain learning' is strongly implied by this division of responsibility: when provided with the definition of a new domain, a hyper-heuristic must be able to produce effective solutions in this domain without significant practitioner expertise or intervention.

As part of a wider community initiative, we therefore argue for a polar stance to that of the prevailing view of hyper-heuristics: instead of imposing a 'maximally restrictive' interface between problem-domain and hyper-heuristic solver, we propose that it is vital to make problem domain ('analytic') and solution trajectory ('empirical') information available to the solver via some 'universal' knowledge exchange format. The wider possibilities then include:

- The extension of the algorithm selection problem to include 'analytic' information as part of the mapping process.
- The availability of arbitrarily rich features for machine learning approaches.
- The creation of a library of declarative descriptions of domains via a constraint language, more easily customized for a new domain than program code.

To coordinate these varied activites, a wider community initiative is in progress to promote an architectural vision of 'Metaheuristics in the Large' [52].

References

1. Kenneth Sörensen and Fred W Glover. Metaheuristics. In *Encyclopedia of operations research and management science*, pages 960–970. Springer US, 2013.
2. Jerry Swan and Nathan Burles. Templar - A framework for template-method hyper-heuristics. In *Genetic Programming - 18th European Conference, EuroGP 2015, Copenhagen, Denmark, April 8-10, 2015, Proceedings*, pages 205–216, 2015.
3. H. Fisher and G. L. Thompson. Probabilistic learning combinations of local job-shop scheduling rules. In J. F. Muth and G. L. Thompson, editors, *Industrial Scheduling*, pages 225–251, New Jersey, 1963. Prentice-Hall, Inc.
4. Robert H. Storer, S. David Wu, and Renzo Vaccari. New search spaces for sequencing problems with application to job shop scheduling. *Manage. Sci.*, 38(10):1495–1509, October 1992.
5. Robert H. Storer, S. David Wu, and Renzo Vaccari. Problem and heuristic space search strategies for job shop scheduling. *ORSA Journal on Computing*, 7(4):453–467, 1995.
6. Hsiao lan Fang, Hsiao lan Fang, Peter Ross, Peter Ross, Dave Corne, and Dave Corne. A promising Genetic Algorithm approach to job-shop scheduling, rescheduling, and open-shop scheduling problems. In *Proceedings of the Fifth International Conference on Genetic Algorithms*, pages 375–382. Morgan Kaufmann, 1993.
7. Hsiao lan Fang, Peter Ross, and Dave Corne. A promising hybrid GA/heuristic approach for open-shop scheduling problems. In *Proceedings of the 11th Conference on Artificial Intelligence*, pages 590–594, 1994.

8. Peter Cowling, Graham Kendall, and Eric Soubeiga. A hyperheuristic approach to scheduling a sales summit. In Edmund Burke and Wilhelm Erben, editors, *Practice and Theory of Automated Timetabling III*, volume 2079 of *Lecture Notes in Computer Science*, pages 176–190. Springer Berlin Heidelberg, 2001.
9. Peter Ross. Hyper-heuristics. In Edmund K. Burke and Graham Kendall, editors, *Search Methodologies*, pages 529–556. Springer US, 2005.
10. John Woodward, Jerry Swan, and Simon Martin. The 'Composite' design pattern in metaheuristics. In *Proceedings of the 2014 Conference Companion on Genetic and Evolutionary Computation Companion*, GECCO Comp '14, pages 1439–1444, New York, NY, USA, 2014.
11. M. Hyde, E. Özcan, and Edmund K. Burke. 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*, pages 798–801, 2009.
12. Edmund Burke, Graham Kendall, Jim Newall, Emma Hart, Peter Ross, and Sonia Schulenburg. Hyper-heuristics: An emerging direction in modern search technology. In *Handbook of metaheuristics*, pages 457–474. Springer, 2003.
13. Peter Ross, Sonia Schulenburg, Javier G. Marín-Blázquez, and Emma Hart. Hyper-heuristics: Learning to combine simple heuristics in bin-packing problems. In *Proceedings of the Genetic and Evolutionary Computation Conference*, GECCO '02, pages 942–948, San Francisco, CA, USA, 2002. Morgan Kaufmann Publishers Inc.
14. Peter Ross, Javier G. Marín-Blázquez, Sonia Schulenburg, and Emma Hart. Learning a procedure that can solve hard bin-packing problems: A new GA-based approach to hyper-heuristics. In *Proceedings of the 2003 International Conference on Genetic and Evolutionary Computation: Part II*, GECCO'03, pages 1295–1306, Berlin, Heidelberg, 2003. Springer-Verlag.
15. Eric Soubeiga. *Development and application of hyperheuristics to personnel scheduling*. PhD thesis, School of Computer Science,University of Nottingham, UK, 2003.
16. Rong Qu, Edmund K. Burke, and Barry McCollum. Adaptive automated construction of hybrid heuristics for exam timetabling and graph colouring problems. *European Journal of Operational Research*, 198(2):392 – 404, 2009.
17. Gabriela Ochoa, Matthew Hyde, Tim Curtois, A Vazquez-Rodriguez, James Walker, Michel Gendreau, Barry Kendall, Graham McCollum, Andrew J Parkes, Sanja Petrovic, et al. Hyflex: A benchmark framework for cross-domain heuristic search. In *Evolutionary Computation in Combinatorial Optimization*, pages 136–147. Springer, 2012.
18. Edmund K Burke, Michel Gendreau, Matthew Hyde, Graham Kendall, Gabriela Ochoa, Ender Özcan, and Rong Qu. Hyper-heuristics: A survey of the state of the art. *Journal of the Operational Research Society*, 64(12):1695–1724, 2013.
19. John R. Rice. The algorithm selection problem. volume 15 of *Advances in Computers*, pages 65 – 118. Elsevier, 1976.
20. Carla P. Gomes and Bart Selman. Algorithm portfolios. *Artificial Intelligence*, 126(1-2):43 – 62, 2001.
21. Bernardo A. Huberman, Rajan M. Lukose, and Tad Hogg. An Economics Approach to Hard Computational Problems. *Science*, 275(5296):51–54, 1997.
22. Matteo Gagliolo and Jürgen Schmidhuber. Learning dynamic algorithm portfolios. *Annals of Mathematics and Artificial Intelligence*, 47(3-4):295–328, 2006.
23. Jerry Swan and Nathan Burles. Templar - a framework for template-method hyper-heuristics. In Penousal Machado et al., editors, *Genetic Programming*, volume 9025 of *Lecture Notes in Computer Science*, pages 205–216. Springer, 2015.
24. Konstantin Chakhlevitch and Peter I. Cowling. Hyperheuristics: Recent developments. In Carlos Cotta, Marc Sevaux, and Kenneth Sörensen, editors, *Adaptive and Multilevel Metaheuristics*, volume 136 of *Studies in Computational Intelligence*, pages 3–29. Springer, 2008.
25. J. Woodward, A. Parkes, and G. Ochoa. A mathematical formalization of hyper-heuristics. Workshop on Hyper-Heuristics - Automating the Heuristic Design Process, September 2008. `http://www.cs.stir.ac.uk/~jrw/publications/defHH.pdf`. Online; accessed 21st October 2015.
26. Douglas B. Lenat. EURISKO: A program that learns new heuristics and domain concepts. *Artif. Intell.*, 21(1-2):61–98, 1983.

27. Jack Mostow and Armand E. Prieditis. Discovering admissible heuristics by abstracting and optimizing: A transformational approach. In *Proceedings of the 11th International Joint Conference on Artificial Intelligence - Volume 1*, IJCAI'89, pages 701–707, San Francisco, CA, USA, 1989. Morgan Kaufmann Publishers Inc.
28. Jerry Swan, Ender Özcan, and Graham Kendall. Hyperion - a recursive hyper-heuristic framework. In Carlos Coello, editor, *Learning and Intelligent Optimization*, volume 6683 of *Lecture Notes in Computer Science*, pages 616–630. Springer Berlin / Heidelberg, 2011.
29. Alexander E.I. Brownlee, Jerry Swan, Ender Özcan, and Andrew J. Parkes. Hyperion^2: A toolkit for {Meta-, Hyper-} heuristic research. In *Proceedings of the 2014 Conference Companion on Genetic and Evolutionary Computation Companion*, GECCO Comp '14, pages 1133–1140, New York, NY, USA, 2014. ACM.
30. Edmund K Burke, Matthew Hyde, Graham Kendall, Gabriela Ochoa, Ender Özcan, and John R Woodward. A classification of hyper-heuristic approaches. In *Handbook of metaheuristics*, pages 449–468. Springer, 2010.
31. Jerry Swan, John. R. Woodward, Ender Özcan, Graham Kendall, and Edmund. K. Burke. Searching the Hyper-heuristic Design Space. *Cognitive Computation*, 6(1):66–73, 2014.
32. Ender Özcan, Burak Bilgin, and Emin Erkan Korkmaz. Hill climbers and mutational heuristics in hyperheuristics. In *Parallel Problem Solving from Nature - PPSN IX*, volume 4193 of *Lecture Notes in Computer Science*, pages 202–211. Springer Berlin Heidelberg, 2006.
33. Shahriar Asta and Ender Özcan. A tensor-based selection hyper-heuristic for cross-domain heuristic search. *Information Sciences*, 299:412 – 432, 2015.
34. Roberto Battiti. Reactive search: Toward self–tuning heuristics. In V. J. Rayward-Smith, I. H. Osman, C. R. Reeves, and G. D. Smith, editors, *Modern Heuristic Search Methods*, pages 61–83. John Wiley & Sons Ltd., Chichester, 1996.
35. Una Benlic and Jin-Kao Hao. A study of adaptive perturbation strategy for iterated local search. In *Evolutionary Computation in Combinatorial Optimization - 13th European Conference, EvoCOP 2013, Vienna, Austria, April 3-5, 2013. Proceedings*, pages 61–72, 2013.
36. Graham Kendall, Eric Soubeiga, and Peter Cowling. Choice function and random hyperheuristics. In *Proceedings of the fourth Asia-Pacific Conference on Simulated Evolution And Learning, SEAL*, pages 667–671. Springer, 2002.
37. Andrew J. Parkes, Ender Özcan, and Daniel Karapetyan. A software interface for supporting the application of data science to optimisation. In *Learning and Intelligent Optimization*, volume 8994 of *Lecture Notes in Computer Science*, pages 306–311. Springer, 2015.
38. Jerry Swan, Martin Edjvet, and Ender Özcan. Augmenting metaheuristics with rewriting systems. Technical Report CSM-197, Computing Science and Mathematics, University of Stirling, Stirling FK9 4LA, Scotland, January 2014.
39. Simon Martin, Djamila Ouelhadj, Pieter Smet, Greet Vanden Berghe, and Ender Özcan. Cooperative search for fair nurse rosters. *Expert Systems with Applications*, 40(16):6674–6683, 2013.
40. Abscon, an XCSP v2.0 solver. `http://www.cril.univ-artois.fr/~lecoutre/software.html`. Online; accessed Oct 2015.
41. Olivier Roussel and Christophe Lecoutre. XML representation of constraint networks: Format XCSP 2.1. *CoRR*, abs/0902.2362, 2009.
42. XCSP v3.0. `http://www.xcsp.org/series.html`. Online; accessed Oct 2015.
43. Fred Glover and Manuel Laguna. *Tabu Search*. Norwell, MA, USA, 1997.
44. Luc De Raedt, Tias Guns, and Siegfried Nijssen. Constraint programming for data mining and machine learning. *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence (AAAI-10)*, 2010.
45. John F. Sowa. *Knowledge Representation: Logical, Philosophical and Computational Foundations*. Brooks/Cole Publishing Co., Pacific Grove, CA, USA, 2000.
46. R. Quillan. *A notation for representing conceptual information: an application to semantics and mechanical English paraphrasing*. Systems Development Corp., 1963.
47. Marvin Minsky. A framework for representing knowledge. Technical report, Cambridge, MA, USA, 1974.

48. Thomas R Gruber. Toward principles for the design of ontologies used for knowledge sharing? *International journal of human-computer studies*, 43(5):907–928, 1995.
49. S. Asta, S. Martin, E. Özcan, and E. Burke. A multi-agent system embedding online tensor learning for flow shop scheduling. 2015.
50. Grigoris Antoniou and Frank Van Harmelen. *A semantic web primer*. MIT press, 2004.
51. `http://www.openmath.org`. Online; accessed Oct 2015.
52. Jerry Swan, Steven Adriaensen, Mohamed Bishr, Edmund K. Burke, John A. Clark, Patrick De Causmaecker, Juanjo Durillo, Kevin Hammond, Emma Hart, Colin G. Johnson, Zoltan A. Kocsis, Ben Kovitz, Krzysztof Krawiec, Simon Martin, J. J. Merelo, Leandro L. Minku, Ender Özcan, Gisele L. Pappa, Erwin Pesch, Pablo Garcia-Sànchez, Andrea Schaerf, Kevin Sim, Jim Smith, Thomas Stützle, Stefan Voß, Stefan Wagner, and Xin Yao. A research agenda for metaheuristic standardization. In *MIC 2015: The XI Metaheuristics International Conference*, June 2015.