Parallel Problem SolvingusingHyper-heuristics and Hyflex Framework

Manoj Bahel[1], Dr. Ani Thomas[2],ShrishtiAdil[3], *PranjaliSarangpure[3]*

[1] Research Scholar Bhilai Institute of Technology, Durg

[2]H.O.D. (I.T.) Bhilai Institute of Technology, Durg

[3]Student B.E. (IT) Bhilai Institute of Technology, Durg

bahel.manoj@gmail.com,ani.thomas@bitdurg.ac.in

Abstract- *Hyper-heuristics comprises a set of approaches with the common goal of automating the design and adaptation of Meta-heuristic methods to solve problems. The primary objective is to produce more generally applicable search methodologies over a given problem domain. The hyper-heuristics will lead to the development of more general solutions that are able to handle a wide range of problemdomains on the other hand meta-heuristic approaches are more suitable to a particular problem or a narrow class of problems. Hyper-heuristics are broadly concerned with finding better sequence of heuristicor algorithmapplicable in a given situation. Evolutionary algorithms and meta-heuristics approaches have been used to solve a number of optimization problems bur their design, however, has become increasingly complex with real world problems, because there are significant number of parameter or algorithm choices involved, and the lack of guidance as to how to proceed when selecting them. To overcome these difficulties a HyFlex framework is used which is a modular and flexible Java class library. It provides a numberof problem domain modules, each of which consists of the problem specific algorithm components. In Flex framework only high-level control strategy needs to be implemented and it provides easy-to-use interface used to link problem domain modules.*

Keywords:*Hyper-heuristics,HyFlex framework,Construction and Perturbation heuristics,Generation hyper-heuristics.*

1. Introduction:

Hyper-heuristic research aims to achieve the level of generality for the given problem domain. Hyper-heuristics concentrates at reducing the role of human expert in the process of designing computational searchmethodologies, and thus raise the level of generality in which thesemethodologies operate(Edmund Burke et al., 2003). Heuristics, meta-heuristics approach to solve computational search problems are becoming complex day by day as these involves number of parameters which are problem dependent and availability of different algorithms available to solve it. Hyper-heuristics tries to overcome these by automating the design and adaptation of heuristic methods, which further results in getting generality of solution in a given problem domain.In 1997 the term hyper-heuristic was first used and a protocol is described that combines several artificial intelligence methods in the context of automated theorem proving(Denzinger, Fuchs, & Fuchs, 1997).In 2000 a term 'heuristics to choose heuristics' was used in the context of combinatorial optimization

A hyper-heuristic is a high-level methodology that operates upon a given a probleminstance and a number of low-level heuristics. It then selects and apply an appropriatelow-level heuristic to get generalized solution(Edmund Burke et al., 2003).A number of hyper-heuristic approaches using high-level methodologies, in conjunction with a set of low-level heuristics, applied to different combinatorial problems,have been proposed in the literatures.

As the research in the field of heuristic search automation advances there is a need of reducing the involvement of human expert because of increased complexity of the whole system due number of parameters to set,availability of different algorithms etc. Hyflex is a framework for hyper-heuristics which is a flexible Java class library fordesigning and testing. HyFlex isa software framework for the development of cross-domain search methodologies. The framework features a common software interface for dealing with different combinatorial optimization problems(Edmund K Burke, Curtois, Hyde, Ochoa, & Vazquez-Rodriguez, 2011).A detailed knowledge of the problem domains is not required for algorithm designer. He/She can concentrate on developing more general purpose solutions for the problem domain.It provides a number of problem domain modules, each of which encapsulates the problem-specific algorithm components.(Cowling, Kendall, & Soubeiga, 2001). Six hard combinatorial problems are fully implemented: maximum satisfiability, one dimensional bin packing, permutation flow shop, personnel scheduling, traveling salesman and vehicle routing(Edmund K Burke et al., 2011).HyFlex provides an interface to link the problem domain modules in an easy manner.

2. Hyper-heuristics

2. Hyper-heuristics: The term hyper-heuristics has been broadly defined as "heuristics to select heuristics"(Edmund K Burke et al., 2011). Hyper-heuristics operate on a space of actions that modify a solution of a given problem rather than on space of solutions themselves, as is the case of meta-heuristics.Need for new approach arises because meta-heuristics were usually problem specific and required fine tuning of their parameters. So it is necessary to have domain knowledge as well as good knowledge of the applied meta-heuristic(Bilgin, Özcan, & Korkmaz, 2006)(Özcan, Bilgin, & Korkmaz, 2008).Therefore, the reusability of such sophisticated algorithms was often limited. Domain specific control parameters required meant that these algorithms could not have been used on other domain than they were designed for and even different set of constraints in the same domain would mean suboptimal performance and would require very different parameter settings (Edmund Burke et al., 2003).

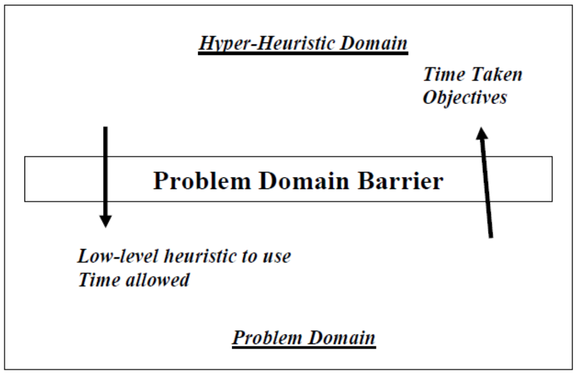


Figure 1 A schematic of separation of high level HH and LLHs

Hyper-heuristic were introduced as a high level approach to select a low level heuristics or combination of low level heuristics that would be most suitable for the problem at hand. The Figure 1 shows the communication of hyper-heuristic high level through the domain barrier by using domain specific evaluation metrics.

Various different classifications of hyper-heuristics have been presented in literature.majority of these classifications were summarized in (Edmund K Burke et al., n.d.). In it hyper-heuristics are classified on the basis of their learning mechanisms and by the nature of the heuristics search space.

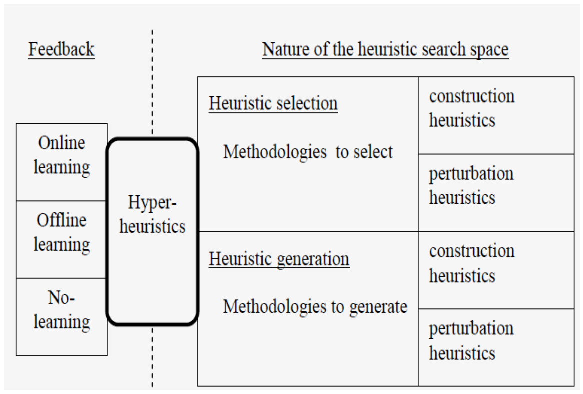


Figure 2 A classification of hyper-heuristic approaches, according to two dimensions (I) the natureof the heuristic search space, and (ii) the source of feedback during learning.

From the learning point of view, three categories were introduced:

- No learning - these algorithms without learning simply follow predetermined sequence of actions that do not reflect the performance on the current problem nor any preceding problems. An example of such technique would be VNS as described in (Hansen & Mladenović, 2001).
- Offline learning - these algorithms use information gathered through training on few problem instances to select the most suitable heuristics for the given problem domain as a whole. Example of this approach is the XCS framework described in (Javier G. Marín-Blázquez, 1998).
- Online learning - these methods use feedback obtained during the execution on a given problem to alter the heuristic selection on the run. Algorithms that use reinforced learning for heuristic selection like (Sim, n.d.) or (E. K. Burke, Kendall, & Soubeiga, 2003)are a good example of this group as well as those using EA to select heuristics like (Kendall, 2003)

Classification by the nature of the heuristic search space introduces two categories:

- Heuristicselection methodologies:

  Produce combinations of pre-existing:
  - Construction heuristics
  - Perturbation heuristics
- Heuristic generation methodologies:

Generate new heuristic methods using basic components (building-blocks) of:

- Construction heuristics
- Perturbation heuristics

2.1 Construction and Perturbation heuristics

- Construction heuristics - they start with empty solution and they gradually build it until complete solution is found.
- Perturbation heuristics - they can only modify existing solution and include mutational heuristics and hill-climbers or local searchers. These will typically use a randomly generated solution or solution created by construction heuristic as a starting point. Construction heuristics can be only applied when the solution is incomplete, whilst the perturbation heuristics can be applied any time.

2.1.1Selection constructive hyper-heuristics

In these approaches solutions are built incrementally. It startswith an empty solution, and then gradually constructive heuristics are selected andused to build a completesolution. The hyper-heuristic framework is provided witha set of pre-existing constructiveheuristics. The main task is to select the heuristic that is somehow the most suitable for the current problem state. This process continues until the final state has been reached. As there is a natural ending to the construction process when a complete solution is reached.

2.1.2 Selection perturbative hyper-heuristics

These approaches aimthrough a process of automatically selecting and applying aheuristic. In these approaches heuristics are automatically selected and applied to improve a candidate solution. Online and offline machine learning techniquesare useful to the heuristic selection strategies in order tomake informed decisions regarding which heuristic toemploy at a given step. Wide variety of combinatorial optimization problems have been solved using hyper-heuristic methodologiesbased on perturbative heuristics.

There are a few studies on the hyper-heuristic methodologiesto select perturbative heuristics that performmulti-point search. Themajority of proposed approaches conduct asingle point search. In a single point search based hyper-heuristicframework, an initial candidate solution goesthrough a set of successive stages repeatedly until termination.First, a heuristic (or a subset of heuristics) is selectedfrom a set of low-level perturbative heuristics and thenapplied to a single candidate solution.

Finally, a decision ismade about whether to accept or reject the new solution.A hyper-heuristic to select perturbative heuristicsperforming single-point search combines two separatecomponents: (I) heuristic selection method and (ii) move acceptance method.

2.1.3 Generation hyper-heuristics

The main feature of these type of hyper-heuristic searches is that a space of heuristics constructed from components rather than a space of complete, pre-defined, heuristics. Heuristic generator may outputs the new heuristic that produced the solution, and this newly generated heuristic can be potentially reused on new problem instances.

Genetic programming is an evolutionary computation technique that evolves a population of computer programs, and is the most common methodology used to automatically generate heuristics. So genetic programming can be viewed as a hyper-heuristic to generate heuristics.

3. Hyflex framework

Real-world complex optimization problems are successfully handled by evolutionary algorithms and meta-heuristics, but their design however has become increasingly complex. To make these methodologies universally applicable, it is important to provide self-managed systems that can manage and reconfigure themselves ‘on the fly’; adapting to the changing problem conditions, based on general rules provided by their users(Edmund K Burke et al., 2011).

To provide such a level of generality, hyper-heuristic frameworks are used. This is a very active area of research where new hyper-heuristics frameworks are emerging. They are classified into different types of hyper-heuristic with the help of shared common features. This field is a sub-field of artificial intelligence , which again is a substantial collection of bio-inspired algorithms (Ryser-welch & Miller, n.d.).

Hyper-heuristics instead of working on a search space of problem solutions, automatically generates an algorithm that solves a problem more efficiently. It provides at least one solution whenever the algorithm stops but a global optima is not guaranteed to be found with heuristics.

HyFlex was used to support an international research competition: the first Cross-Domain Heuristic Search Challenge. HyFlex (Hyper-heuristic Flexible framework) (Edmund K Burke et al., 2011)is a software framework helping in the development of domain independent search heuristics (hyper-heuristics), and testing across multiple problem domains.

Hyper-heuristics Flexible framework enables the development, testing and comparison of hyper-heuristics. To achieve these goals it uses modularity and the concept of decomposing a heuristic search algorithm into two main parts(Edmund K Burke et al., 2011):

1. A general-purpose part: the algorithm or hyper-heuristic.
2. The problem-specific part: provided by the HyFlex framework.

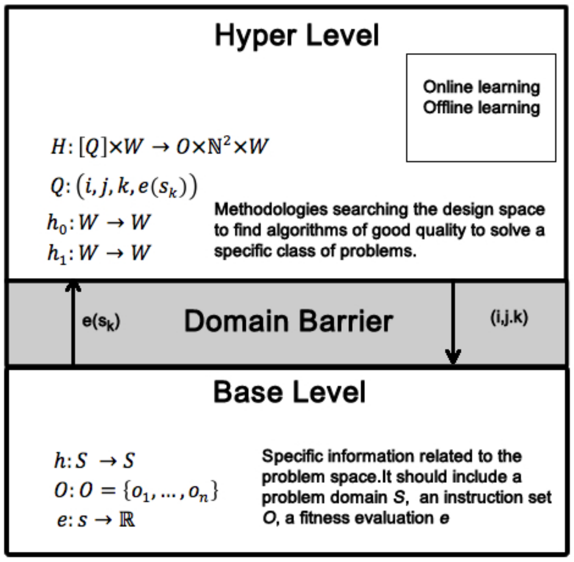

Hyper Level
Online learning
Offline learning
$H: [Q] \times W \to O \times \mathbb{N}^2 \times W$
$Q: (i, j, k, e(s_k))$
$h_0: W \to W$
$h_1: W \to W$
Methodologies searching the design space to find algorithms of good quality to solve a specific class of problems.
$e(s_k)$
Domain Barrier
(i,j,k)
Base Level
$h: S \to S$
$O: O = \{o_1, \ldots, o_n\}$
$e: s \to \mathbb{R}$
Specific information related to the problem space.It should include a problem domain S, an instruction set O, a fitness evaluation e

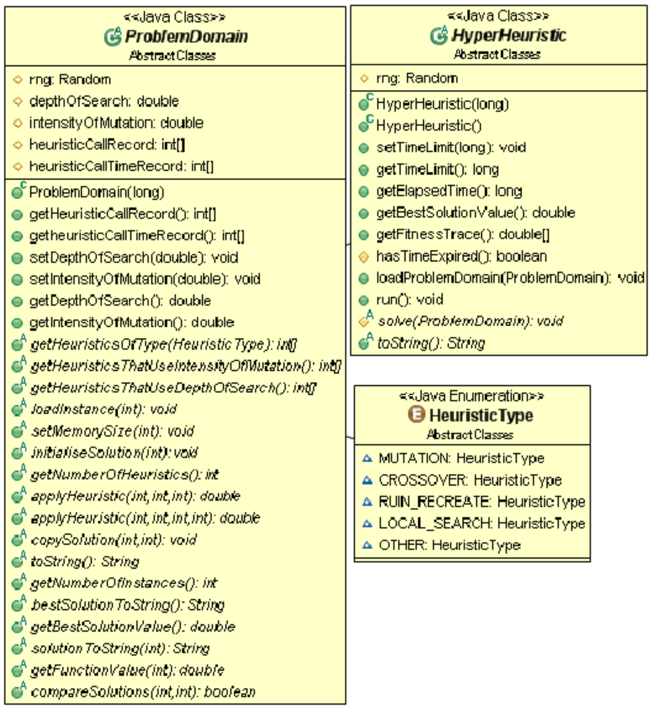
Figure 3 The mathematical model suggested by (Swan, Drake, Ozcan, Goulding, & Woodward, 2013)to represent a hyper-heuristicsystem.

The motivation of Hyflex was inspired by the two-level hyper heuristic model. "*The emphasis of our HyFlex framework lies in providing the algorithm components that are problem specific, thus liberating the algorithm designers needing to know the problem's domain's specific details*"(Ek Burke, Curtois, & Hyde, 2009)

The algorithm designers only devise new hyper level algorithms; the Base level contains a library of well-known combinatorial problem domains with their benchmarks.
In this context, the low-level heuristic is aset of operators that either brings small or large changes in the problem solutions. These perturbations results in expansion of search to a larger neighborhood and then guarantees better solutions are discovered. The framework structure hides strictly within the domain barrierthe problem domain, in order to implement a domain-independentform of hyper-heuristic.

The framework is written in Java which is an object orientation, platform independence and it has automatic memory management. At the highest level, the frameworkconsists of just two abstract classes: **Problem Domain** and **Hyper Heuristic**. The structure of these classes is shown in the class diagram of figure 4.

In the diagram, the signatures adjacent to circles are public methods and fields and the signatures adjacent to diamonds are protected. Abstract methods are denoted by italics and the implementations of these methods are necessarily different for each instantiation of problem domain or hyper-heuristic.


```
<<Java Class>>
ProblemDomain
Abstract Classes

rng: Random
depthOfSearch: double
intensityOfMutation: double
heuristicCallRecord: int[]
heuristicCallTimeRecord: int[]

ProblemDomain(long)
getHeuristicCallRecord(): int[]
getheuristicCallTimeRecord(): int[]
setDepthOfSearch(double): void
setIntensityOfMutation(double): void
getDepthOfSearch(): double
getIntensityOfMutation(): double
getHeuristicsOfType(HeuristicType): int[]
getHeuristicsThatUseIntensityOfMutation(): int[]
getHeuristicsThatUseDepthOfSearch(): int[]
loadInstance(int): void
setMemorySize(int): void
initialiseSolution(int): void
getNumberOfHeuristics(): int
applyHeuristic(int,int,int): double
applyHeuristic(int,int,int,int): double
copySolution(int,int): void
toString(): String
getNumberOfInstances(): int
bestSolutionToString(): String
getBestSolutionValue(): double
solutionToString(int): String
getFunctionValue(int): double
compareSolutions(int,int): boolean

<<Java Class>>
HyperHeuristic
Abstract Classes

rng: Random

HyperHeuristic(long)
HyperHeuristic()
setTimeLimit(long): void
getTimeLimit(): long
getElapsedTime(): long
getBestSolutionValue(): double
getFitnessTrace(): double[]
hasTimeExpired(): boolean
loadProblemDomain(ProblemDomain): void
run(): void
solve(ProblemDomain): void
toString(): String

<<Java Enumeration>>
HeuristicType
Abstract Classes

MUTATION: HeuristicType
CROSSOVER: HeuristicType
RUIN_RECREATE: HeuristicType
LOCAL_SEARCH: HeuristicType
OTHER: HeuristicType
```

Figure 4 Class-diagram-for-the-HyFlex-framework

3.1 The Problem Domain Class

The Problem Domain class provides the following elements.

- A user-configurable memory (a population) of solutions, which can be managed by the hyper-heuristic through methods such as **setMemorySize** and **copySolution**.
- A routine to randomly initialize solutions, **initialiseSolution(I)**, where i is the index of the solution array in the memory.
- A set of problem specific heuristics, which are used to modify solutions. These are called with the **applyHeuristic(i, j, k)** method, where i is the index of the heuristic to call, j is the index of the solution in memory to modify and k is the index in memory where the resulting solution should be placed.
- A varied set of instances that can be easily loaded using the method: **loadInstance(a)** , where a is the index of the instance to be loaded.
- A fitness function, which can be called with the **getFunctionValue(i)** method, where i is the index of the required solution in the memory. HyFlex problem domains are always implemented as minimization problems, so a lower fitness is always better. The fitness of the best solution found so far in the run can be obtained with the **getBestSolutionValue()** method.
- Two parameters: **α** and **β**, (0 <= [α, β] <= 1), which are the 'intensity' of mutation and 'depth of search', respectively, that control the behavior of some search operators.

3.2 The Hyper Heuristic Class

- The **HyperHeuristic** class is designed to allow algorithms which implement this class to be compared and benchmarked across one or more of the problem domains available. Each class must contain a **toString()** method, to give the methodology a name.
- It must also contain a **solve()** method, in which the functionality of the particular methodology is written. The **solve ()** method would normally contain a loop, which continues while the time limit (defined by the user) has not been exceeded.
- In the loop, the code should provide a mechanism for selecting between the available problem-specific heuristics and choose to which solutions in memory to apply the heuristics. This class could choose to work with a memory size of 1 for a single point search, or a large memory could be maintained for a population based approach.
- A hyper-heuristic class automatically records the length of time for which it has been running and this can be monitored through methods such as has **TimeExpired ()** and **getElapsedTime ()**.

4. Limitations of Hyflex framework

- Ability of Hyflex of solving large real-world problems is limited by strict use of templates (Ross, 2014).
- The algorithm designers are required to structure their code with the explicit definitions of the three components.
- The framework seems to only support local search meta-heuristic in the Hyper level, making it very challengingto use Genetic Programming.

5. Discussion and Future work

The hyper-heuristics operates on a search space of heuristics rather than directly on a search space of problem solutions. This feature provides the potential for increasing the level of generality of search methodologies.

Heuristic generation methodologies offer more scope for greater levels of generalization. However, they can be more difficult to implement, when compared with heuristic selection methodologies since they require the decomposition of existing heuristics, and the design of an appropriate framework.

HyFlex is a software framework which enables cross-domain algorithms to be easily developed and reliably compared. It currently provides 6 problem domains, each containing a set of problem instances and search operators to apply. Therefore, it represents a novel extension of the notion of benchmark for combinatorial optimization.When using HyFlex, researchers can concentrate their efforts on designing their adaptive methodologies, rather than implementing the required set of problem domains.

Different algorithm design ideas have been implemented and tested using HyFlex. Some successful design principles start to emerge such as the use of diversification and intensification phases, the use of reinforcement learning for heuristic selection, adaptation of the heuristic parameters and the use of adaptive acceptance criteria.HyFlex can be extended to include new domains, additional instances and Operators in existing domains and multi-objective and dynamic problems. Thecurrent software interface can also be extended to incorporate additional feedbackinformation from the domains to guide the adaptive search controllers.

REFERENCES

Bilgin, B., Özcan, E., & Korkmaz, E. E. (2006). An experimental study on hyper-heuristics and exam timetabling. In *Practice and Theory of Automated Timetabling VI* (pp. 394–412). https://doi.org/10.1007/11593577

Burke, E., Curtois, T., & Hyde, M. (2009). HyFlex: A flexible framework for the design and analysis of hyper-heuristics. *Multidisciplinary International Scheduling Conference (MISTA 2009), …*, (August), 790--797. Retrieved from http://www.mistaconference.org/2009/abstracts/790-797-112-A.pdf

Burke, E. K., Curtois, T., Hyde, M., Ochoa, G., & Vazquez-Rodriguez, J. a. (2011). HyFlex: A Benchmark Framework for Cross-domain Heuristic Search. *Arxiv Preprint arXiv11075462, abs/1107.5*, 28. https://doi.org/10.1007/978-3-642-29124-1_12

Burke, E. K., Hyde, M., Kendall, G., Ochoa, G., Ozcan, E., & Woodward, J. R. (n.d.). A Classification of Hyper-heuristic Approaches, 1–21.

Burke, E. K., Kendall, G., & Soubeiga, E. (2003). A Tabu-Search Hyperheuristic for Timetabling and Rostering. *Journal of Heuristics*, *9*(6), 451–470. https://doi.org/10.1023/B:HEUR.0000012446.94732.b6

Burke, E., Kendall, G., Newall, J., Hart, E., Ross, P., & Schulenburg, S. (2003). Hyper-Heuristics: An Emerging Direction in Modern Search Technology. In *Handbook of Metaheuristics* (pp. 457–474). https://doi.org/10.1007/0-306-48056-5_16

Cowling, P., Kendall, G., & Soubeiga, E. (2001). A Hyperheuristic Approach to Scheduling a Sales Summit. *Third International Conference on the Practice And Theory of Automated Timetabling, PATAT 2000*, 176–190. https://doi.org/10.1007/3-540-44629-X_11

Denzinger, J., Fuchs, M., & Fuchs, M. (1997). High_Performance_ATP_Systems_by_Combinin. In *Fifteenth International Joint Conference on Artificial Intelligence (IJCAI 97)* (pp. 102–107).

Hansen, P., & Mladenović, N. (2001). Variable neighborhood search: Principles and applications. *European Journal of Operational Research*, *130*(3), 449–467. https://doi.org/10.1016/S0377-2217(00)00100-4

Javier G. Marín-Blázquez, S. S. (1998). A hyper-heuristic framework with XCS: learning to create novel problem-solving algorithms constructed from simpler algorithmic ingredients. In *IWLCS'03-05 Proceedings of the 2003-2005 international conference on Learning classifier systems* (Vol. 1529, pp. 193–218). Retrieved from http://portal.acm.org/citation.cfm?id=945844

Kendall, G. (2003). An investigation of a tabu assisted hyper-reuristic genetic algorithm. *The 2003 Congress on Evolutionary Computation, 2003. CEC '03.*, 2230–2237. https://doi.org/10.1109/CEC.2003.1299949

Özcan, E., Bilgin, B., & Korkmaz, E. (2008). A comprehensive analysis of hyper-heuristics. *Intelligent Data Analysis*, *12*(1), 3–23. https://doi.org/10.1038/msb.2008.79

Ross, P. (2014). Hyper-heuristics. *Search Methodologies*, 409–429. https://doi.org/10.1016/B978-1-4160-4649-3.00017-X

Ryser-welch, P., & Miller, J. F. (n.d.). A Review of Hyper-Heuristic Frameworks. York University,England.

Sim, K. (n.d.). KSATS-HH : A Simulated Annealing Hyper-Heuristic with Reinforcement Learning and Tabu-Search, 7–10.

Swan, J., Drake, J., ??zcan, E., Goulding, J., & Woodward, J. (2013). A comparison of acceptance criteria for the daily car-pooling problem. *Computer and Information Sciences III - 27th International Symposium on Computer and Information Sciences, ISCIS 2012*, 477–483. https://doi.org/10.1007/978-1-4471-4594-3-49

Ochoa, G., Hyde, M.: The Cross-domain Heuristic Search Challenge (CHeSC 2011). Website (2011), http://www.asap.cs.nott.ac.uk/chesc2011/