Learning to Solve Bin Packing Problems with an Immune Inspired Hyper-heuristic

Kevin Sim[1] , Emma Hart[1] and Ben Paechter[1]

[1] Institute for Informatics and Digital Innovation,
Edinburgh Napier University,
Merchiston Campus, Edinburgh, EH10 5DT
k.sim@napier.ac.uk

Abstract

Motivated by the natural immune system's ability to defend the body by generating and maintaining a repertoire of antibodies that collectively cover the potential pathogen space, we describe an artificial system that discovers and maintains a repertoire of *heuristics* that collectively provide methods for solving *problems* within a problem space. Using bin-packing as an example domain, the system continuously generates novel heuristics represented using a tree-structure. An novel affinity measure provides stimulation between heuristics that cooperate by solving problems in different parts of the space. Using a test suite comprising of 1370 problem instances, we show that the system self-organises to a minimal repertoire of heuristics that provide equivalent performance on the test set to state-of-the art methods in hyper-heuristics. Moreover, the system is shown to be highly responsive and adaptive: it rapidly incorporates new heuristics both when entirely new sets of problem instances are introduced or when the problems presented change gradually over time.

Introduction

Heuristic search methods have been shown to be successful in solving a wide-range of real-world problems. Typically for a given application domain, a range of heuristics for solving problems will exist; these might range in nature from simple rules encapsulating expert knowledge to complex search algorithms that need to be tuned by experts to work. Commonly, different heuristics will work well on problems in different parts of the *problem space*. By collecting together a set of heuristics, it is hoped that collectively, the weaknesses of individual heuristics can be compensated for by other heuristics in the set (Burke et al., 2003). The goal of the *hyper-heuristics* field is to find automated methods that can both generate appropriate sets of heuristics and provide a means of selecting between heuristics in the set, given either a new problem instance or even a partially solved problem instance.

While such approaches have proved successful in many application areas, most hyper-heuristic approaches fail to *continuously* learn from experience On the one hand, the failure to exploit previous knowledge leads to inefficient hyper-heuristics; on the other, if the characteristics of instances of problems in the domain change over time, a hyper-heuristic may need to be completely re-tuned or in the worst case redesigned periodically. An 'ideal' hyper-heuristic would be able to exploit previous knowledge through access to some kind of memory, rapidly adapt existing knowledge to new circumstances, and additionally, generate new knowledge when previous knowledge is not applicable.

We observe that the immune system fulfils very similar properties in its role as a host maintenance and defence system. The immune system maintains a repertoire of antibodies that has been shown theoretically to cover the space of potential pathogens. Clonal selection mechanisms provide a means of rapidly adapting existing antibodies to new variants of previous pathogens; meta-dynamic processes are able to generate novel antibodies; a memory mechanism enables the immune system to respond rapidly when faced with pathogens it has previously been exposed to.

Using this analogy, we present a system that is shown experimentally to outperform single human-designed heuristics by a significant margin and furthermore, is shown to be more adaptable and responsive than a recent state-of-the-art hyper-heuristic approach when faced with a continually changing problem landscape, thereby addressing the needs of real-world practitioners more fully. The novel system described has the following features:

- it generates novel heuristics from a library of component parts
- it utilises meta-dynamic processes to both add and remove heuristics from the system resulting in a self-organising network of heuristics
- it sustains a network of interacting heuristics of minimal size that collectively solve problems from the whole of the problem space
- it encapsulates memory in the sustained network enabling rapid adaptation to new problems

Background

We briefly cover some background in relation to the tested application domain of bin-packing and outline the immunology that inspired this approach before describing the system in detail.

Immunology

Artificial Immune Systems (AIS) algorithms have been applied in many domains, including solving combinatorial optimisation (CO) problems (Kromer et al., 2012). Unlike other biologically inspired paradigms there is no de-facto model used by AIS practitioners. Of the models used the one most frequently applied to CO problems is *clonal selection theory* (Burnet, 1959), however, many other paradigms exist: we exploit the *idiotypic network* (Jerne, 1974) model in this work, in part due to its plasticity and its ability to describe the memory mechanism exhibited by the immune system.

Idiotypic network inspired models have been developed to address machine learning problems such as clustering (Neal, 2003), however the most relevant line of work in relation the model proposed in this article is in the robotics domain. An idiotypic network model to perform behaviour arbitration in mobile robots was one of the first applications in AIS (Watanabe et al., 1998) and spawned subsequent related work. In these early approaches, antibodies in the network represented behaviours (actions); antibodies were stimulated by environmental conditions and further suppressed or stimulated by interactions with other antibodies in a modified version of Farmer's original equation (Farmer et al., 1986). The immune repertoire consisted of a set of antibodies which collectively covered a space of appropriate actions required to achieve the robot goals in the space defined by environment. In this early work, antibodies were pre-defined; research focused on evolving connections and matching strengths between antibodies. In more recent work, Whitbrook *et al* have extended earlier work by using an evolutionary algorithm in separate learning phase to produce antibodies which are used to seed the network. The benefits of seeding the network with a diverse and novel set of antibodies are described in Whitbrook et al. (2010).

In this paper, we adopt a similiar approach to Whitbrook *et al* in recognising the need to develop a diverse set of antibodies for potential inclusion in the network. In contrast to previous work however, our mechanism is not just used to seed the network in an initial phase but continuously generates a stream of potential antibodies which are either incorporated into the network or rejected according to a meta-dynamic process. This results in a learning scheme in which the network is able to continuously adapt over time.

HyperHeuristics

Originally described as *"heuristics to select heuristics"* (Burke et al., 2003) the field of hyper-heuristics has evolved to also encompass *"heuristics to generate heuristics"* (Burke et al., 2010); both methods have the common goal of searching a landscape defined by the heuristics in order to find a procedure for solving a problem, rather than searching directly over the solution space defined by the problem itself. Hyper-heuristic methods have been widely applied to bin-packing problems. We discuss the most relevant research briefly below.

Heuristic generation Genetic Programming (GP) is typically used a method of generating new heuristics. In Burke et al. (2006, 2012) GP was used to automate the design of heuristics for the bin-packing problem in multiple dimensions. Using a small set of benchmarks, they found the generated heuristics to be competitive with human-designed heuristics. In Sim and Hart (2013), the authors introduced Single Node Genetic Programming as a method to evolve new heuristics for bin-packing. SNGP, introduced in Jackson (2012), differs from the conventional GP model introduced by Koza (1992) in a number of key respects. A single tree structure is used to represent a population of possible trees by allowing any node to be the start node. Only mutation is used to change connections with the tree, alleviating the undesirable affect of *bloat* found in conventional GP and enabling different network structures to emerge in addition to classical tree structures.

Heuristic selection Most hyper-heuristics use a *fixed* set of low-level heuristics (whether generated or hand-built) and define or learn a model to select an appropriate heuristic based on the current state of the problem or a description of the problem characteristics. In contrast, Sim and Hart (2013) introduce an island model based on the concept of *Cooperative Coevolution* (Potter and De Jong, 2000) in which a set of islands each contribute a heuristic to a collaborating set that collectively are able to solve a problem. Crucially, the number of islands is not prefixed but is adaptable. In Sim and Hart (2013), an island contains a population described using SNGP; each island contributes its best heuristic to the collaboration set. Based on a set of 685 test instances, the authors showed that the model was able to solve more instances and reduce the number of extra bins required than an equivalent sized set of man-made heuristics from the literature.

In the current work, a simplified version of SNGP is used to generate novel heuristics and the island model described above is replaced by an AIS algorithm which is shown to be able to find, maintain and adapt a collaborating set of heuristics that equals or outperforms other previous approaches.

1D BPP and Benchmarks

The objective of the one dimensional bin packing problem (BPP) is to find a packing which minimises the number of containers, b, of fixed capacity c required to accommodate a set of n items with weights $\omega_j : j \in \{1 \ldots n\}$ falling in the range $1 \leq \omega_j \leq c, \omega_j \in \mathbb{Z}$ whilst enforcing the constraint that the sum of weights in any bin does not exceed the bin capacity c. The lower and upper bounds on b, (b_l and

b_u) respectively, are given by equation 1 Any heuristic that does not return empty bins will produce, for a given problem instance, p, a solution using b_p bins where $b_l \leq b_p \leq b_u$.

$$b_l = \left\lceil \sum_{j=1}^{n} \omega_j \div c \right\rceil , \; b_u = n \tag{1}$$

Table 1 shows the parameters from which the benchmark problem instances used in this study were generated. Data sets $ds1, ds2$ & $ds3$, introduced by Scholl et al. (1997) all have optimal solutions that vary from the lower bound given by Equation 1. However all are known and have been solved since their introduction (Schwerin and Wäscher, 1997). All of the instances from $FalU$ and $FalT$, introduced by Falkenauer (1996), have optimal solutions at the lower bound except for one (Gent, 1998).

Six deterministic heuristics commonly cited in the literature are used for comparison — FFD, DJD, DJT, ADJD, BFD and SS. Descriptions of each of these heuristics can be found in Sim and Hart (2013). Note that in the implementation used here, each deterministic heuristic is presented with each problem instance's items pre-sorted in descending order of size.

Implementation

The system comprises of three main parts: a database of problem instances, a heuristic generator and the AIS as illustrated by Figure 1.

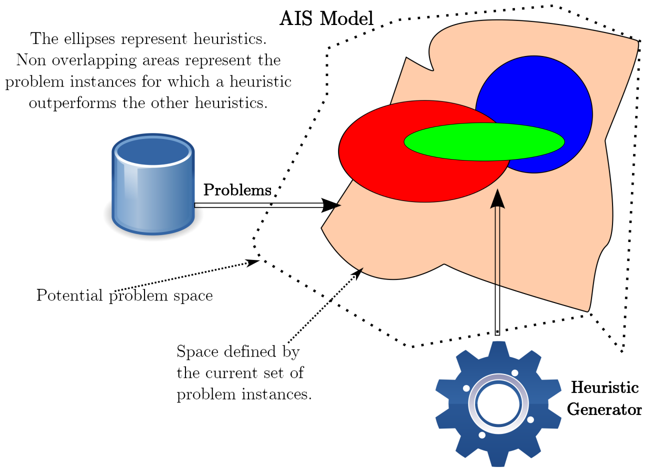

AIS Model
The ellipses represent heuristics. Non overlapping areas represent the problem instances for which a heuristic outperforms the other heuristics.
Problems
Potential problem space
Space defined by the current set of problem instances.
Heuristic Generator

Figure 1: System Model

The system is designed to run continuously; problem instances can be added or removed from the system at any point. A heuristic generator akin to *gene libraries* in the natural immune system provides a continual source of potential heuristics. The AIS itself consists of a set of heuristics (akin to *antibodies in the natural immune system* that interact with each other based on an *affinity metric*. The overall goal of the system is to develop a repertoire of heuristics that can solve the set of problem instances to which they are currently exposed and that can adapt its structure and constituent parts as the problem instances change in nature. The component parts are described in detail below.

Heuristic Generator

Heuristics are represented using a tree structure and are generated using only the *initialisation* process used in SNGP (Jackson, 2012) resulting in structures as shown in the example given in Figure 2 — this tree shows one of the deterministic heuristics from the literature DJD encoded in a tree format. A fixed set of terminal and function nodes are available to the generator and are defined in Table 2 which combines nodes according to the process outlined in Algorithm 1. Further details outlining the justification for the choice of nodes and further details on the SNGP process can be found in Sim and Hart (2013). One heuristic is generated per iteration of the AIS algorithm.

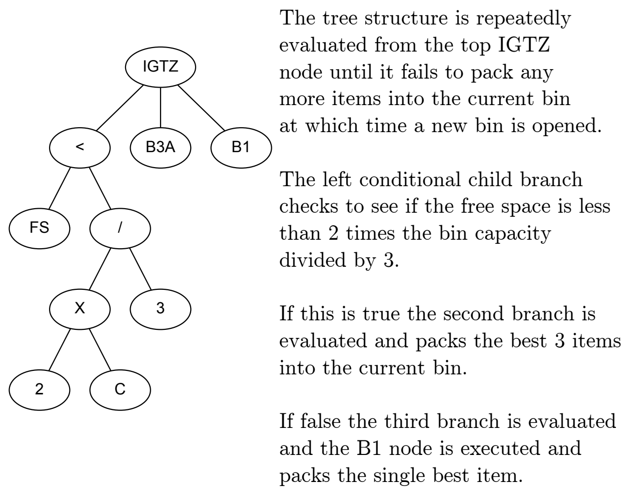

IGTZ
<
B3A
B1
FS
/
X
3
2
C
The tree structure is repeatedly evaluated from the top IGTZ node until it fails to pack any more items into the current bin at which time a new bin is opened.
The left conditional child branch checks to see if the free space is less than 2 times the bin capacity divided by 3.
If this is true the second branch is evaluated and packs the best 3 items into the current bin.
If false the third branch is evaluated and the B1 node is executed and packs the single best item.

Figure 2: DJD Heuristic Expressed as a Tree

Algorithm 1 Heuristic Generation

```
1: Each of the terminal nodes $T \in \{t_1, \ldots t_r\}$ are added exactly once. The terminal nodes are given an integer identification number ranging from $1 \ldots r$.
2: A number, $n$, of function nodes are selected at random from the set of all function nodes $F \in \{f_1, \ldots, f_s\}$ and given an identification number ranging from $r+1, \ldots$ to $r+n$. This allows for the possibility of duplicate function nodes within the population or for SNGP structures with function nodes omitted.
3: The function nodes have all their child nodes assigned at random from nodes with a lower id thus preventing any infinite looping.
4: A single node is chosen at random to be the root node.
```

Table 1: Data sets $ds1$, $ds3$ and $FalU$ were created by generating n items with weights randomly sampled from a uniform distribution between the bounds given by ω. Those in $FalT$ were generated in a way similar to Falkenauer (1996) so that the optimal solution has exactly 3 items in each bin with no free space. Scholl's $ds2$ was created by randomly generating weights from a uniform distribution in the range given by $\varpi \pm \delta$. The final column gives the number of instances generated for each parameter combination.

Data Set	capacity (c)	n	ω	#Problems
$ds1$	100,120,150	50,100,200,500	[1,100],[20,100],[30,100]	$36 \times 20 = 720$
$ds3$	100000	200	[20000,30000]	10
$FalU$	150	120,250,500,1000	[20,100]	$4 \times 20 = 80$
$FalT$	1	60,120,249,501	[0.25,0.5]	$4 \times 20 = 80$

Data Set	c	n	ϖ (avg weight)	$\delta(\%)$	# Problems
$ds2$	1000	50,100,200,500	$\frac{c}{3}, \frac{c}{5}, \frac{c}{7}, \frac{c}{9}$	20,50,90	$48 \times 10 = 480$

Table 2: Nodes Used

Function Nodes

/	Protected divide returns -1 if denominator is 0 otherwise the result of dividing the first operand by the second
>	Returns 1 if the first operand is greater than the second or -1 otherwise
IGTZ	Evaluates the first operand. If it evaluates as greater than zero the result of evaluating the second operand is returned otherwise the result of evaluating the third operand is returned
<	Returns 1 if the first operand is less than the second or -1 otherwise
X	Returns the product of two operands

Terminal Nodes

B1	Packs the single largest item into the current bin returning 1 if successful or -1 otherwise
B2	Packs the largest combination of exactly 2 items into the current bin returning 1 if successful or -1 otherwise
B2A	Packs the largest combination of up to 2 items into the current bin giving preference to sets of lower cardinality. Returns 1 if successful or -1 otherwise
B3A	As for B2A but considers sets of up to 3 items
B5A	As for B2A but considers sets of up to 5 items
C	Returns the bin capacity
FS	Returns the free space in the current bin
INT	returns a random integer value $\in (-1, ..., +5)$
W1	Packs the smallest item into the current bin returning 1 if successful else -1

AIS

The AIS component is responsible for constructing a network of interacting heuristics and for governing the dynamic processes that enable heuristics to be incorporated or rejected from the current network. These two aspects are now described.

Affinity A key aspect of this is the *affinity metric* that defines the manner and the extent to which heuristics can interact. In the natural immune system, affinity is defined by physical and chemical interactions between molecules of different shape, with molecules with *complementary* structures showing highest affinity (the well known lock-and-key analogy). We re-interpret the notion of complementarity in the heuristic space as follows:

Definition Heuristic H_A is complementary to Heuristic H_B if Heuristic H_A uses fewer bins than Heuristic H_B on at least one problem instance

The *affinity* of Heuristic H_A for Heuristic H_B is equal to the number of extra bins Δb_{abp} used by Heuristic H_B summed across the set of l problem instances available to the system, and is defined in equation 2. Note that affinity between two heuristics is asymmetrical.

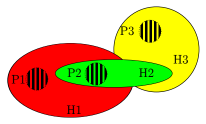
$$\alpha_{ab} = \sum_{p=1}^{l} \Delta b_{abp} \begin{cases} \Delta b_{a_p b_p} = b_{b_p} - b_{a_p} & : \text{ if } b_{a_p} < b_{b_p} \\ \Delta b_{a_p b_p} = 0 & : \text{ otherwise} \end{cases} \tag{2}$$

The total *stimulation* experienced by a heuristic is the sum of the affinities with all other heuristics in the network and is given by Equation 3 where there are m heuristics present in the network.

$$s_x = \sum_{j=1}^{m} \alpha_{xj} \tag{3}$$

An example for a system of $m = 3$ heuristics and $l = 3$ problems is given in figure 3.

Network Dynamics Each iteration, one new heuristic is generated and is made available to the network. The affinity metric described encourages *diversity* between pairs of



H1 is the best heuristic on P1
H1 and H2 are equal winners on P2
H3 is the best heuristic on P3

H1 gets a stimulation value from H2 on P1 of 3 as it solves P1 using 3 less bins.

H2 gets a stimulation value from H1 on P1 of 0 as it uses more bins and is ignored.

This is repeated for each heuristic and each problem instance.

Number of Bins Used

	H1	H2	H3
P1	10	13	12
P2	9	9	10
P3	8	9	7

The total stimulation recieved by each heuristc is:

$$
\begin{array}{lccc}
 & \text{P1} & \text{P2} & \text{P3} \\
\text{H1} = & (3+2) & +\ (0+1) & +\ (1+0) \\
\text{H2} = & (0+0) & +\ (0+1) & +\ (0+0) \\
\text{H3} = & (0+1) & +\ (0+0) & +\ (1+2)
\end{array}
$$

Figure 3: Affinity between Complementary Heuristics

heuristics leading to increased network performance by sustaining those that cover different parts of the problem space. In a practical application however, it is reasonable to assume that in addition to maintaining diversity, important goals of the system should be to find (1) the set of heuristics that most efficiently cover the problem space and (2) the set that collectively minimise the total number of bins used to solve all problems the network is exposed to. While the latter is addressed by sustaining any heuristic with non-zero stimulation, the former goal requires some attention.

Previous AIS models relating to idiotypic networks attempt to use Farmer's original equation (Farmer et al., 1986) to govern the dynamics of addition and removal of nodes from a network. In machine-learning applications such as data-clustering this was quickly found to lead to population explosion e.g. Timmis et al. (2000), later addressed by using resource limiting mechanisms (Timmis and Neal, 2001). In previous robotic applications, the situation is avoided completely by using a network of fixed size and focusing only on evolving connections. In more theoretical models such as Hart (2006) the criteria are not relevant, as the goal is simply to show that a network can be sustained. In this heuristic case, simply sustaining all heuristics that contribute to covering the heuristic space is likely to lead to population explosion in the same manner observed in data-mining applications, as no pressure exists on the system to encourage efficiency.

Therefore, at the end of each iteration, the contribution made by each heuristic to the overall performance is calculated in terms of whether it plays a unique contribution in determining the overall quality of the system. Any heuristic whose contribution is subsumed by one or more other heuristics is removed from the system. Thus, in Figure 3 heuristic $H2$ is subsumed by the combination of $H1$ and $H3$ and is therefore removed from the system. This simple method thus provides pressure on the system to minimise the number of heuristics used. This can be thought of as an artificial form of apoptosis that removes antibodies that cover duplicate parts of the landscape and allows for proliferation of new cells only in the case that they cover an equal or greater area of the search space that an existing heuristic is occupying.

Pseudo-code describing the network dynamics is give in Algorithm 2. $\mathcal{H}$ is the set of heuristics currently present in the network, $\mathcal{E}$ is the set of problems in the current environment. Note that there is a single parameter in the algorithm that defines the maximum concentration a heuristic can reach. This parameter introduces user control over the period of the network *memory* and is discussed in section .

Algorithm 2 AIS Pseudo Code

```
 1: repeat
 2:    Add a randomly generated heuristic
 3:    Optionally change the set of problem instances
 4:    for all heuristics $i \in \mathcal{H}$ do
 5:       Calculate current $stimulation_i$ using Equation 3
          based on $\mathcal{E}$
 6:       if $stimulation_i > 0$ then
 7:          if $concentration_i < concentration_{max}$ then
 8:             $concentration_i \leftarrow concentration_i + 1$
 9:          end if
10:       else
11:          $concentration_i \leftarrow concentration_i - 1$
12:       end if
13:    end for
14:    Remove heuristics with $concentration \leq 0$
15:    Remove all heuristics that give no global improvement (oldest first)
16: until stopping criteria met
```

Experimental Results

A number of experiments were conducted using the model described with the test set of 1370 instances previously described in section .

Baseline comparison on a static dataset

An initial experiment was performed in order to compare the AIS model to previous work in terms of solution quality on a *static* data set. 1370 instances were split into two sets: the *training* set comprised of the first and then every second problem instance from each of the data sets with the remaining problem instances used as a test set. This split ensured an even distribution of problem instances from each of parameter combinations used to generate them in the test and training set. All 685 problems in the training set were placed in the AIS environment, and the AIS algorithm run for 200 iterations. The quality of the network is measured in terms of the number of problem instances for which the known optimal solution was found by at least one heuristic, and additionally in terms of the number of extra bins required to solve the instances compared to the known optimal. Results are recorded for problem instances in the training set, and then on the test set of instances (with no further iteration of the network). The experiments are repeated 30 times, reinitialising the system each time. Results are compared to using each of the single deterministic heuristics and to the island model described in Sim and Hart (2013) and shown in table 3a.

The best results shown for the AIS are identical to the best results presented in Sim and Hart (2013), even though the Island Model in that publication required evolutionary operators in order to find solutions and the AIS relies simply on randomly generated heuristics. Given this performance, we now examine the response of the system to dynamically changing data.

Response to dynamically changing data

The AIS method was inspired by the ability of the natural immune system to rapidly respond to a dynamically changing environment, in which the self-sustaining network of antibodies is postulated to act as a memory of past responses. Four experiments were devised in order to investigate the behaviour of the system under the following dynamically changing conditions.

1. A set of 685 randomly selected problem instances is introduced every 200 iterations. At this point, problems currently in the system are removed, the network cleared, and the system is started from scratch

2. A set of 685 randomly selected problem instances is introduced every 200 iterations. At this point, problems currently in the system are removed, but the existing network is retained

3. Every 200 iterations the problem instances used are toggled between those from Scholl's data sets 1 & 2 (described in Table 1). Antibodies in the existing network are retained.

4. A new problem is introduced every iteration. The existing network is retained.

Experiment 1 was designed to investigate the time taken for the system to reach equilibrium from an initial starting state. The results are shown in Figure 4. Note that in this graph (and in the following ones) only the first 5 cycles are shown. At the end of each 200 iteration cycle, the total number of bins required by the system is *assumed* to be the best that can be achieved and is given a score of 0. The y-axis then shows the number of bins required over and above this score, enabling different sets of problem instances where the optimal number of bins required varies to be plotted on a relative scale to highlight the response to changing conditions.

It is clear that the system performs poorly at the point of restart requiring up to 1200 extra bins than the best result

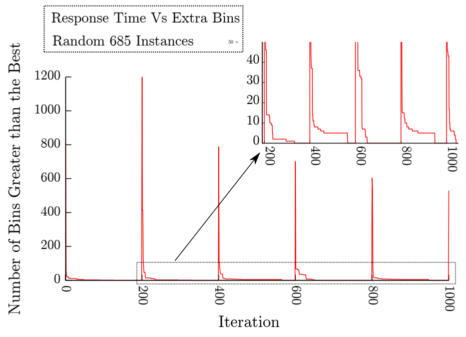


Figure 4: Response when the system is completely restarted every 200 iterations

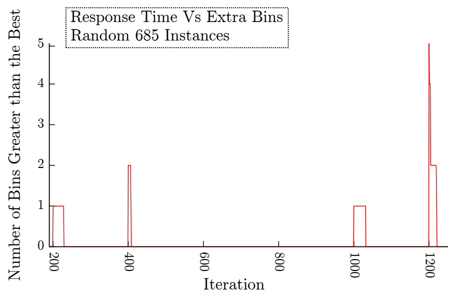


Figure 5: Response When the Problem Instances are Changed every 200 iterations

found during each cycle. The response is still rapid however: the median number of iterations required to reach the best result is 65 (shown in table4).

Experiment 2 examines the role of memory, implicit in the sustained network. The system is effectively trained during a 200 iteration cycle on a set of problem instances representative of the whole set of 1370 problem instances. When the problem instances are changed, the heuristics already in the system have a greater probability of performing well on the new set of problem instances than randomly generated ones due to the shared problem characteristics. Results are shown in Figure 5. The median number of iterations to reach the optimal value is 10 and the total number of bins required is at most 5 greater than that at the end point (Table 4).

Experiment 3 presents a more difficult test for the system; the two alternating problem sets have very different characteristics. The memory encapsulated in the network is therefore expected to be of less relevance. Results are shown in Figure 6 and in Table 4. Performance drops in comparison to Figure 5 where the memory could be exploited; however, there is still an improvement in comparison to the complete re-start applied in Experiment 1.

Finally, **Experiment 4** investigates the system response to slowly changing environmental conditions. Figure 7 shows how the system responds as problem instances are gradually introduced to the system. The system starts with no problem instances and no heuristics. Each iteration one randomly

Single Deterministic Heuristics		
Heuristic	Problems Solved	Extra Bins
FFD	393	1088
DJD	356	1216
DJT	**430**	**451**
ADJD	336	679
BFD	394	1087
SS	383	1112

(a)

	Collaborative Heuristic Models							
	Problems Solved				Extra Bins			
	min	max	mean	sd	min	max	mean	sd
Immune Model	554	**559**	556	1.4	**159**	165	162	1.4
Island Model	552	**559**	557	1.4	**159**	164	162	1.4

(b)

Table 3: A comparison of results obtained on a static dataset of 685 problems using a) single heuristics and b) collaborative methods

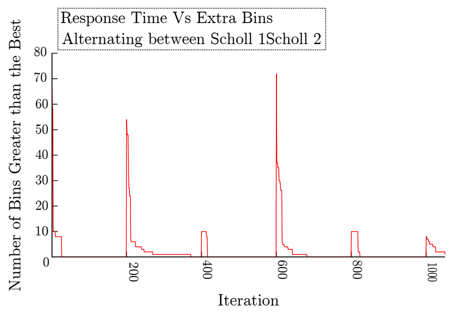


Figure 6: Response When alternating between data sets every 200 iterations

Table 4: System Response Time to Varying Conditions

	Response Time		
	min	**max**	**median**
Exp1	9	186	63
Exp2	1	111	10
Exp3	15	200	64.5

Exp1, 2 & 3 correspond to Figures 4, 5 & 6 respectively. The minimum, maximum, and median number of iterations that were required to reach the optimal value are shown. All results are taken over 30 data points.

generated heuristic and one randomly selected problem instance are introduced. The graph shows the percentage of extra bins over the known optimal that are required using the heuristics present in the network at each iteration. The number of heuristics sustained by the system is also shown.

Most benefit is gained from adding new heuristics when few heuristics are present. As the number of heuristics increases, it becomes harder for a newly added heuristic to

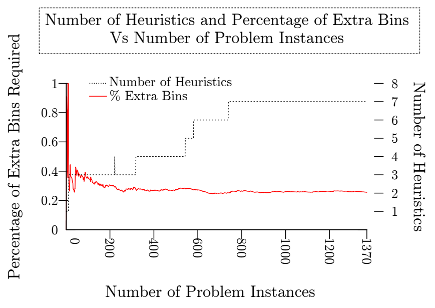


Figure 7: Number of Heuristics Sustained and the Percentage of Extra Bins Required than Optimal as the Number of Problem Instances is Increased

find a new niche. Furthermore, the total number of potential heuristics is limited by the generation method currently used (although this could easily be extended by adding new nodes).

Conclusion

We have introduced an AIS model for generating and maintaining a repertoire of heuristics that solve bin-packing problems. Results show that the model achieves equal results to state-of-the-art methods on static data sets, and is extremely responsive to dynamically changing data-sets, making it suitable for use as a continuous learning system. The memory encapsulated in the network can be exploited to provide a rapid response to new problems that share characteristics with those previously seen by the network.

The model can be generalised to other domains by replacing the component parts of the heuristic generator. Currently, the generation model does not make use of any evolutionary or cloning operators to improve randomly generated solutions. By adding this feature in the future, it is hoped to speed up the response even further.

Acknowledgements

This work was funded by EPSRC grant P/J1021628/1 Real World Optimisation with Life-Long Learning.

References

Burke, E., Hyde, M., and Kendall, G. (2006). Evolving bin packing heuristics with genetic programming. In *Parallel Problem Solving from Nature - PPSN IX*, volume 4193 of *Lecture Notes in Computer Science*, pages 860–869. Springer Berlin / Heidelberg.

Burke, E., Kendall, G., Newall, J., Hart, E., Ross, P., and Schulenburg, S. (2003). Hyper-heuristics: An emerging direction in modern search technology. In *Handbook of Metaheuristics*, International Series in Operations Research & Management Science, chapter 16, pages 457–474. Kluwer.

Burke, E. K., Hyde, M., Kendall, G., Ochoa, G., Özcan, E., and Woodward, J. R. (2010). A classification of hyper-heuristic approaches. In Gendreau, M. and Potvin, J.-Y., editors, *Handbook of Metaheuristics*, volume 146 of *International Series in Operations Research & Management Science*, pages 449–468. Springer US.

Burke, E. K., Hyde, M. R., Kendall, G., and Woodward, J. (2012). Automating the packing heuristic design process with genetic programming. *Evol. Comput.*, 20(1):63–89.

Burnet, F. M. (1959). *The clonal selection theory of acquired immunity*. Cambridge University Press, Cambridge, UK.

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

Farmer, J. D., Packard, N. H., and Perelson, A. S. (1986). The immune system, adaptation, and machine learning. *Phys. D*, 2(1-3):187–204.

Gent, I. P. (1998). Heuristic solution of open bin packing problems. *Journal of Heuristics*, 3(4):299–304.

Hart, E. (2006). Analysis of a growth model for idiotypic networks. In Bersini, H. and Carneiro, J., editors, *Artificial Immune Systems*, volume 4163 of *Lecture Notes in Computer Science*, pages 66–80. Springer Berlin / Heidelberg.

Jackson, D. (2012). Single node genetic programming on problems with side effects. In Coello, C., Cutello, V., Deb, K., Forrest, S., Nicosia, G., and Pavone, M., editors, *Parallel Problem Solving from Nature - PPSN XII*, volume 7491 of *Lecture Notes in Computer Science*, pages 327–336. Springer Berlin Heidelberg.

Jerne, N. K. (1974). Towards a network theory of the immune system. *Ann Immunol (Paris)*, 125C(1-2):373–89.

Koza, J. R. (1992). *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. MIT Press, Cambridge, MA, USA.

Kromer, P., Platos, J., and Snasel, V. (2012). Practical results of artificial immune systems for combinatorial optimization problems. In *Nature and Biologically Inspired Computing (NaBIC), 2012 Fourth World Congress on*, pages 194–199.

Neal, M. (2003). Meta-stable memory in an artificial immune network. In Timmis, J., Bentley, P., and Hart, E., editors, *Artificial Immune Systems*, volume 2787 of *Lecture Notes in Computer Science*, pages 168–180. Springer Berlin Heidelberg.

Potter, M. A. and De Jong, K. A. (2000). Cooperative coevolution: An architecture for evolving coadapted subcomponents. *Evol. Comput.*, 8:1–29.

Scholl, A., Klein, R., and Jürgens, C. (1997). Bison: a fast hybrid procedure for exactly solving the one-dimensional bin packing problem. *Comput. Oper. Res.*, 24(7):627–645.

Schwerin, P. and Wäscher, G. (1997). The bin-packing problem: A problem generator and some numerical experiments with ffd packing and mtp. *International Transactions in Operational Research*, 4(5-6):377–389.

Sim, K. and Hart, E. (2013). Generating single and multiple cooperative heuristics for the one dimensional bin packing problem using a single node genetic programming island model. In *Proceedings of GECCO 2013*, New York, NY, USA (to appear). ACM.

Timmis, J. and Neal, M. (2001). A resource limited artificial immune system for data analysis. *Knowledge-Based Systems*, 14(34):121 – 130.

Timmis, J., Neal, M., and Hunt, J. (2000). An artificial immune system for data analysis. *Biosystems*, 55(13):143 – 150.

Watanabe, Y., Ishiguro, A., and Uchikawa, Y. (1998). Decentralized behavior arbitration mechanism for autonomous mobile robot using immune network. In DasGupta, D., editor, *Artficial Immune Systems and Their Applications*, pages 187 – 209. Springer-Verlag New York, Inc.

Whitbrook, A. M., Aickelin, U., and Garibaldi, J. M. (2010). Two-timescale learning using idiotypic behaviour mediation for a navigating mobile robot. *Appl. Soft Comput.*, 10(3):876–887.