Markov Chain Selection Hyper-heuristic for the Optimisation of Constrained Magic Squares

Ahmed Kheiri and Ed Keedwell
University of Exeter
College of Engineering, Mathematics and Physical Sciences
Streatham Campus, Harrison Building, Exeter EX4 4QF, UK
Email: {a.kheiri, e.c.keedwell}@exeter.ac.uk

***Abstract*—A square matrix of size $n \times n$, containing each of the numbers $(1, \ldots, n^2)$ in which every row, column and both diagonals has the same total is referred to as a magic square. The problem can be formulated as an optimisation problem where the task is to minimise the deviation from the magic square constraints and is tackled here by using hyper-heuristics. Hyper-heuristics have recently attracted the attention of the artificial intelligence, operations research, engineering and computer science communities where the aim is to design and develop high-level strategies as general solvers which are applicable to a range of different problem domains. There are two main types of hyper-heuristics in the literature: methodologies to select and to generate heuristics and both types of approaches search the space of heuristics rather than solutions. In this study, we describe a Markov chain selection hyper-heuristic as an effective solution methodology for optimising constrained magic squares. The empirical results show that the proposed hyper-heuristic is able to outperform the current state-of-the-art method.**

I. Introduction

Search methodologies are at the core of almost all decision support systems, particularly while dealing with combinatorial optimisation problems. The state-of-the-art methods are often tailored for a particular problem by the experts in the area. Such systems are generally costly to build. Since they are custom-made, it is almost impossible to reuse them in another problem domain. Even a slight change in the problem definition may require expert intervention. Whenever exact methods cannot be implemented, researchers and practitioners resort to heuristics which are 'rule of thumb' methods for solving a given problem. There is a growing interest towards more general, cheaper and intelligent systems that can automate the heuristic design process. Humans design and provide the components of such systems while computers either run those components or use them to build new components while solving a given problem. Hyper-heuristics are such automated search methodologies that explore the space of heuristics for solving computationally difficult optimisation problems in decision support [1]. Hyper-heuristic research has been growing since the initial ideas emerged in 1960s [2], [3]. This work focuses on selection hyper-heuristics, which were initially defined as 'heuristics to choose heuristics' [4]. Table I provides some selected problem domains in which hyper-heuristics were used as solution methodologies.

TABLE I. Some selected problem domains in which hyper-heuristics were used as solution methodologies.

Problem Domain [Reference]	Problem Domain [Reference]
Channel assignment [5]	Job shop scheduling [2]
Component placement sequencing [6]	Sales summit scheduling [4]
University course timetabling [7]	Space allocation [8]
Packing [13]	High school timetabling [10]
Orc quest, logistics domain [11]	Vehicle routing problems [12]
Production scheduling [14]	

The use of a logical interface between the high level hyper-heuristic and problem domain, referred to as *domain barrier*, makes selection hyper-heuristics more general search methodologies than the current techniques tailored for a particular domain. This barrier does not allow a hyper-heuristic to retrieve any problem domain specific information. Hence, any selection hyper-heuristic (or its components) can be reused while solving any given problem, assuming that problem domain components have already been implemented. The main components of selection hyper-heuristics are *heuristic selection* and *move acceptance* methods as identified in [15]. Traditionally, heuristic selection selects and applies a heuristic from a set of low level heuristics to the candidate solution generating a new solution and a move acceptance method decides whether to continue with the new solution or the old solution. Recently, a new field of hyper-heuristic methods embedding sequence-analysis techniques has been developed [16], [17]. Experiments on six optimisation problems [16] have shown that selecting and applying a sequence of heuristics can potentially improve the quality of solutions more than those that simply select a single heuristic. The work in [16], [17] utilises a hidden Markov model at which low level heuristics are represented as hidden states. Another hyper-heuristic method employing hidden Markov model exists [18], it represents solutions as hidden states as opposed to the work proposed in [16]. Similar to the work presented in [19], this study uses a Markov chain method as a selection hyper-heuristic aiming to learn good transitions between low level heuristics.

A magic square of size $n \times n$, containing each of the numbers $(1, \ldots, n^2)$, is a two-dimensional array at which every row, column and diagonal has a total value of $n(n^2+1)/2$. Constructing a magic square of a given order is considered as a computationally difficult permutation problem, particularly when additional constraints are imposed [20], [21]. In this study, we investigate the performance of a Markov chain selection hyper-heuristic (MCHH) for optimising constrained magic squares.

The paper is structured as follows. Section II provides the background of magic square problem. Section III presents the description of the problem. Section IV describes the

method components. Section V presents the empirical results. Section VI provides the conclusions of the study.

II. Background

The history of magic squares dates back to 2200 B.C. [20]. An unusual numerical pattern found by Emperor Yu on a tortoise's shell was the oldest known magic square. The Emperor decided to call this unique diagram "Lo-Shu". The Chinese have used the magic squares in the interpretation of philosophy, human behaviour, natural phenomena and other areas of study; and interestingly, some of the porcelain plates in some private collections and museums in China were decorated with magic squares. It is thought that the magic squares were transmitted to the Arabs from the Chinese, probably through India. Magic squares were then introduced to Europe, then journeyed to Japan. Magic squares in India were used in applications other than only in the traditional mathematical context. A sequence of naïve rules to construct magic squares were made by Islamic mathematicians. The seventeenth century witnessed a serious consideration to the study of magic squares when Antoine de la Loubere, a French aristocrat, studied the theory behind the construction of magic squares. The extension of magic squares to 3-dimensions was brought by Adamas Kochansky in 1686. Recently, the magic squares attracted researchers and were applied in statistics, combinatorial mathematics, artificial intelligence, graph theory, industrial arts, experiment designs, location analytics, electronic circuits among others [22], [20].

An exact solver to construct the magic squares is provided in [23]. A magic square of an odd order can be generated using the Siamese method (also known as De la Loubére's method). An odd order magic square is of the form $n = 2k+1$, where k is an integer greater than 0. In the Siamese method, the number 1 is written in the middle of the first row. The remaining numbers are placed in ascending order as an upward diagonal to the empty right square cells. In case the cell is already filled, then the cell below the previous number is used to place the number. A magic square of a doubly even order can be generated using a cross method. A doubly even order magic square is of the form $n = 4k$, where k is an integer greater than 0. The idea is to draw a cross through every 4x4 sub-square and then fill out all the square cells with all numbers in ascending order from the top left of the square to the bottom right. Then, each number, m_{ij}, along a diagonal of the cross is replaced by $(n^2 + 1) - m_{ij}$. Finally, a magic square of a singly even order can be generated using the "LUX" method which has been proposed by J. H. Conway [24]. A singly even order magic square is of the form $n = 4k+2$, where k is an integer greater than 0. The method starts by creating $k + 1$ rows of L, then 1 row of U followed by $k-1$ rows of X. Then replacing the U in the centre with the L above it. The resulted letters form a square of an odd order $2k + 1$. Constructing the singly even order magic square is done by using the Siamese method and filling out each set of square cells surrounding a letter sequentially according to the shape of the letter. Other methods for generating magic squares are reported in [24].

Although there is at most only one distinct magic square of order less than 4, the number of magic squares of order 4 is 880 as has been known since the seventeenth century. The exact number of distinct magic squares of order 5 is 275,305,224 [20]. Researchers claimed that determining the number of distinct magic squares of order 6 and more is a hard unsolved computational problem [25], [26]. A Monte Carlo method is used in [26] to predict the number of magic squares of order 6 and their estimate was $(0.17745 \pm 0.00016) \times 10^{20}$.

The exact deterministic methods for constructing magic squares similar to the ones presented above can only produce a single magic square of a given order. Such methods may fail when some constraints are imposed. A stochastic constructor method based on an improved evolutionary algorithm is proposed in [20]. The magic square problem was then the subject of a competition hosted by SolveIT Software with the goal of finding the *quickest* approach [27]. To make the competition more challenging, a constraint has been imposed such that a solution must have a contiguous sub-square S in a given position.

The winner of the competition developed an approach based on the late acceptance hill climbing algorithm [28]. The organisers of the competition claimed that the approach was able to construct a magic square of size 2600x2600 in one minute. The second and the third approaches were based on the iterative heuristic improvement of rows and columns and the multi-step iterative local search, respectively. More details of these methods are provided in http://www.yuribykov.com/IOC. The framework of the winning approach is extended in [21] to enable the use of selection hyper-heuristics for any given constrained magic square problem. Seven different heuristic selection methods are combined with six move acceptance methods producing a total of 42 selection hyper-heuristics are used for the experiments. The results revealed that the random permutation selection method and naïve move acceptance criterion which accepts a worsening solution with a probability of 0.004% is the fastest in solving the problem and most successful approach beating the winning approach. The hyper-heuristic framework manages 9 mutational heuristics for the small size of the problem and mixes a different two mutational heuristics for larger instances.

III. Problem Description

Given a square matrix M of order n such that:

$$M = \begin{pmatrix} m_{1,1} & m_{1,2} & \cdots & m_{1,n} \\ m_{2,1} & m_{2,2} & \cdots & m_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ m_{n,1} & m_{n,2} & \cdots & m_{n,n} \end{pmatrix}$$

The matrix is considered magic square if:

1) $m_{i,j} \in \{1, 2, ..., n^2\}$
2) $m_{i,j} \neq m_{p,q}$ for all $i \neq p$ and $j \neq q$
3) $\sum_{i=1}^{n} m_{i,j} = n(n^2 + 1)/2, \qquad j = 1, 2, ..., n$
4) $\sum_{j=1}^{n} m_{i,j} = n(n^2 + 1)/2, \qquad i = 1, 2, ..., n$
5) $\sum_{i=1}^{n} m_{i,(n+1-i)} = n(n^2 + 1)/2$
6) $\sum_{i=1}^{n} m_{i,i} = n(n^2 + 1)/2$

The problem can be formulated as minimisation problem at which the goal is to minimise the following equation:

$$\sum_{i=1}^{n}\left|\sum_{j=1}^{n} m_{i,j} - n(n^2+1)/2\right| + \sum_{j=1}^{n}\left|\sum_{i=1}^{n} m_{i,j} - n(n^2+1)/2\right| + \left|\sum_{i=1}^{n} m_{i,(n+1-i)} - n(n^2+1)/2\right| + \left|\sum_{i=1}^{n} m_{i,i} - n(n^2+1)/2\right| \quad (1)$$

In this work, a constraint has been imposed such that a solution must have a contiguous sub-square S in a given position (i, j), where:

$$S = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$$

IV. Methodology

A Markov chain selection hyper-heuristic is used to construct constrained magic squares. The selection method selects a heuristics from a set of low level heuristics and applies to a candidate solution at each decision point. After the chosen heuristic is applied to a candidate solution generating a new solution, this solution is either accepted or rejected depending on the acceptance method. In this work, the proposed selection hyper-heuristic aims to learn good transitions between low level heuristics. To accomplish this, the proposed hyper-heuristic constructs a connected Markov chain in which states correspond to the low level heuristics (LLHs). Each low level heuristic (state) in the Markov chain has a transition probability to move to every other low level heuristic including itself. The hyper-heuristic moves to and applies the next low level heuristic to the candidate solution $S_{current}$, using a roulette wheel selection method. The probability to move from LLH_k to LLH_l is given by: $score_{LLH_{k,l}} / \sum_{\forall j}(score_{LLH_{k,j}})$. Initially, all probabilities to move from one state to another are distributed equally, i.e., $score_{LLH_{i,j}} = 1$ for all i, j. The associated score of moving from LLH_k to LLH_l is increased by one only in the case that applying LLH_l to the candidate solution returned a solution with a quality better than the best recorded solution in hand. The algorithm is outlined in Algorithm 1.

The move acceptance method used in this study is naïve move acceptance criterion as suggested in [21], which accepts a worsening solution with a given probability.

A candidate solution is in the form of a two dimensional array. The objective function is calculated using Equation 1. The constrained magic square is constructed if the imposed constraint is satisfied and the objective function value is 0. We divide the problem into two sub-problems as in [27], [21]. The first sub-problem deals with constructing constrained magic squares of odd order less than or equal 23, or even order less than or equal 18. The second sub-problem deals with constructing constrained magic squares of odd order greater than 23 or with an even order greater than 18. The proposed hyper-heuristic is employed to solve the first sub-problem. In the second sub-problem, the hyper-heuristic and the exact solvers presented in Section II are both used to solve the second sub-problem. The probability of accepting worsening solutions in the first sub-problem is set to 0.004% as suggested in [21], while for the second sub-problem the parameter is fixed with respect to the matrix size $10/n\%$.

Algorithm 1: Markov Chain Hyper-heuristic

```
Let LLH = {LLH_1, LLH_2, ..., LLH_n} represent set of all low level heuristics;
for i ← 1, 2, ..., n do
    for j ← 1, 2, ..., n do
        score_{LLH_{i,j}} = 1;
    end
end
S_current ← S_initial;
S_best ← S_current;
LLH_current ← SelectRandom(LLH);
repeat
    LLH_previous ← LLH_current;
    LLH_current ← RouletteWheel(score, LLH_previous);
    S_new ← Apply(LLH_current, S_current);
    S_current ← MoveAcceptance(S_current, S_new);
    if S_current isBetterThan S_best then
        S_best ← S_current;
        score_{LLH_{previous,current}} ++;
    end
until MagicSquareIsConstructed();
```

A. First Sub-problem

The initial solution is generated by firstly fixing the contiguous sub-square S in the given position (i, j) of the square matrix. The remaining cells of the matrix are then randomly filled with the remaining numbers. Nine low level heuristics, as in [21], that perturb a given candidate solution in different ways are used under the problem domain implementation of the hyper-heuristic framework. All these low level heuristics are designed such that the imposed constraint is never violated.

- **LLH1:** Single swap that attempts to satisfy the magic rule on a selected row, column or diagonal.
- **LLH2:** Select row, column or diagonal and swap it with another row, column or diagonal.
- **LLH3:** Swap the largest element from the largest (or close to largest) row, column or diagonal (i.e. the selection of the largest row, column or diagonal has more probability than the 2nd largest; and the second largest has more probability to be selected than the 3rd, and so on) with the smallest element from the smallest (or close to smallest) row, column or diagonal.
- **LLH4:** Several swaps that attempt to satisfy the magic rule on a selected row, column or diagonal. The heuristic terminates as long as the magic rule is satisfied on the selected row, column or diagonal; or n^2 swaps have been performed.

- **LLH5:** Rectify swap as in [20] by selecting two rows k and l, then swap two elements on column s if:

$$\sum_{j=1}^{n} m_{k,j} - n(n^2+1)/2 = m_{k,s} - m_{l,s} \qquad k \neq l$$

$$n(n^2+1)/2 - \sum_{j=1}^{n} m_{l,j} = m_{k,s} - m_{l,s} \qquad k \neq l$$

Then, apply the same process for elements on row s of columns k and l, if:

$$\sum_{i=1}^{n} m_{i,k} - n(n^2+1)/2 = m_{s,k} - m_{s,l} \qquad k \neq l$$

$$n(n^2+1)/2 - \sum_{i=1}^{n} m_{i,l} = m_{s,k} - m_{s,l} \qquad k \neq l$$

- **LLH6:** Select two random elements and swap them only if they are not on row, column or diagonal that satisfy the magic rule.

- **LLH7:** Rectify swaps as in [20] by selecting two rows k and l, then swap two elements on column s and another two elements on column t where $k \neq l$ and $s \neq t$, if:

$$\sum_{j=1}^{n} m_{k,j} - n(n^2+1)/2 = m_{k,s} - m_{l,s} + m_{k,t} - m_{l,t}$$

$$n(n^2+1)/2 - \sum_{j=1}^{n} m_{l,j} = m_{k,s} - m_{l,s} + m_{k,t} - m_{l,t}$$

Then, apply the same process for elements on rows s and t of columns k and l where $k \neq l$ and $s \neq t$, if:

$$\sum_{i=1}^{n} m_{i,k} - n(n^2+1)/2 = m_{s,k} - m_{s,l} + m_{t,k} - m_{t,l}$$

$$n(n^2+1)/2 - \sum_{i=1}^{n} m_{i,l} = m_{s,k} - m_{s,l} + m_{t,k} - m_{t,l}$$

- **LLH8:** Rectify swaps on diagonals as in [20]:
for $i = 1, 2, ..., n$, $j = 1, 2, ..., n$ and $i \neq j$:
swap $m_{i,i}$ with $m_{j,i}$ and $m_{i,j}$ with $m_{j,j}$ if:

$$m_{i,i} + m_{i,j} = m_{j,i} + m_{j,j}$$

and

$$(m_{i,i}+m_{j,j}) - (m_{i,j}+m_{j,i}) = \sum_{i=1}^{n} m_{i,i} - n(n^2+1)/2$$

swap $m_{i,j}$ with $m_{(n+1-j),j}$ and $m_{i,(n+1-i)}$ with $m_{(n+1-j),(n+1-i)}$ if:

$$m_{i,j} + m_{i,(n+1-i)} = m_{(n+1-j),j} + m_{(n+1-j),(n+1-i)}$$

and

$$(m_{i,(n+1-i)}+m_{(n+1-j),j}) - (m_{i,j}+m_{(n+1-j),(n+1-i)}) = \sum_{i=1}^{n} m_{(n+1-i),i} - n(n^2+1)/2$$

swap row i with row j if:

$$(m_{i,i}+m_{j,j}) - (m_{i,j}+m_{j,i}) = \sum_{i=1}^{n} m_{i,i} - n(n^2+1)/2$$

and

$$(m_{i,(n+1-i)}+m_{j,(n+1-j)}) - (m_{i,(n+1-j)}+m_{j,(n+1-i)}) = \sum_{i=1}^{n} m_{(n+1-i),i} - n(n^2+1)/2$$

swap column i with column j if:

$$(m_{i,i}+m_{j,j}) - (m_{i,j}+m_{j,i}) = \sum_{i=1}^{n} m_{i,i} - n(n^2+1)/2$$

and

$$(m_{(n+1-i),i}+m_{(n+1-j),j}) - (m_{(n+1-j),i}+m_{(n+1-i),j}) = \sum_{i=1}^{n} m_{(n+1-i),i} - n(n^2+1)/2$$

swap row i with row $(n+1-i)$ if:

$$(m_{i,i}+m_{(n+1-i),(n+1-i)}) - (m_{i,(n+1-i)}+m_{(n+1-i),i}) = \sum_{i=1}^{n} m_{i,i} - n(n^2+1)/2$$

$$= n(n^2+1)/2 - \sum_{i=1}^{n} m_{(n+1-i),i}$$

- **LLH9:** Select the largest (or close to largest) row, and the smallest (or close to smallest) row; then iterate from the first column to the last and exchange with a probability of 0.5. Then, do the same operator for columns.

B. Second Sub-problem

We employ the same strategy used in [21] which extends the work in [27]. The matrix of size $n \times n$ is recursively divided into sub-matrices, referred to as magic frames, each with a size of $l \times l$ such that $l \leq n$. The non-border elements of the sub-matrices are filled with zeroes. The sum of numbers in each border row and border column is $l(l^2+1)/2$. The sum of numbers in each non-border row, column and diagonal is $l \times l + 1$. Each element $x_i \leq l * l/2$ has a counterpart $y_i = l * l + 1 - x_i$ that is symmetrically placed in the sub-matrix. Example of sub-matrix of order $l = 4$:

$$\begin{bmatrix} 10 & 15 & 3 & 6 \\ 1 & 0 & 0 & 16 \\ 12 & 0 & 0 & 5 \\ 11 & 2 & 14 & 7 \end{bmatrix}$$

The magic square is composed by recursively constructing the magic frames or by embedding a smaller magic square inside the magic frame. Initially, the magic frame is constructed by placing the necessary set of numbers and their counterparts randomly. The numbers of the imposed contiguous sub-square S are fixed in their right locations, if they exist in the frame. The objective function is the absolute distance of the sum of

the first row to $l(l^2+1)/2$, plus the absolute distance of the sum of the first column to $l(l^2+1)/2$. The proposed hyper-heuristic is employed to construct the magic frames until the objective function reaches zero.

If the contiguous sub-square S is placed in a position close to the border of the matrix, then the magic frames will be constructed from the outer of the overall matrix until they cover the contiguous sub-square. The inner matrix will be constructed using the well-known exact methods. If the contiguous sub-square S is placed in a position too far from the border of the matrix, then the following strategy can be employed. Given a rectangle in the matrix with vertices of V1, V2, V3 and V4, and they are not in the diagonals and V1+V2=V3+V4, then V1 can be safely swapped with V3 and V2 can be safely swapped with V4 without violating the magic rule of the matrix. By applying this property, the constraints can be moved close to the border and then safely assign them back into the right locations.

Two low level heuristics are employed for constructing the magic frames and they are designed such that the imposed constraint is never violated:

- **LLH1:** With $p = 10\%$ swap randomly selected element x_i with its counterpart $l * l + 1 - x_i$; otherwise swap randomly selected element x_i with another randomly selected element y_i and then swap their counterparts $l * l + 1 - x_i$ and $l * l + 1 - y_i$.
- **LLH2:** With $p = 5\%$ swap randomly selected element x_i with its counterpart $l * l + 1 - x_i$; otherwise swap randomly selected element x_i with another randomly selected element y_i and then swap their counterparts $l * l + 1 - x_i$ and $l * l + 1 - y_i$.

V. Results

Since the specification of the competition machine is not known, the experiments have been performed on an i3 CPU M330 at 2.13GHz having 4GB RAM. The experiments are performed aiming to detect the running time for constructing constrained magic square with respect to its order n. For a fair comparison, the same machine has been used to test the performances of MCHH and the previously known approaches. The problem-specific information such as the heuristics and construction of initial solutions methods, were already implemented in [27], [21]. The differences in the computing time are, therefore, caused by the differences in the optimisation methods. Each experiment on a given instance is repeated for fifty trials. A trial is terminated if the expected solution is achieved. The solvers are tested with values of $n = 10, 11, 13, \ldots, 23$ for the first-sub problem; and $n = 25, 50, \ldots, 400$ then progressively larger numbers up to 2600 for the second sub-problem. The value of $n = 2600$ is the largest order solved by the winning approach of the magic square competition under a minute as reported by the organisers. The final experiments are concerned with the placement of the top-left corner of the contiguous sub-square S within the complete square. We initially performed the experimentation by embedding S at the location (1,4). Later on, a set of experiments are performed by randomly varying the placement of S.

Table II summarises the performance comparison of MCHH to the best previously proposed solution methodologies, late acceptance hill climbing (LAHC) and random permutation hyper-heuristic (RPHH), which are the winner of the magic square competition and the quickest-known approach, respectively. The table provides the average execution time (in milliseconds) to construct the constrained magic square over 50 trials of each instance, the standard deviation and the pairwise performance comparison based on Mann-Whitney-Wilcoxon test at a 95% confidence level. The notation A $>$ ($<$) B means that A (B) is better than B (A) and this performance is statistically significant. The notation A $\geq$ ($\leq$) B means that A (B) performs slightly better than B (A) on average and this performance is not statistically significant. MCHH performs better than LAHC in all instances of both sub-problems and this performance is statistically significant. In the first sub-problem, MCHH and RPHH are comparable with the exception of only one instance ($n = 14$) where MCHH approach performs much better with statistically significant performance. In the second sub-problem, MCHH performs better than RPHH in all instances and this performance is statistically significant in most of the instances.

Due to the stochastic nature of the low level heuristics makes it extremely difficult to fully assess the complexity of the overall algorithm. Here, we form a regression model to estimate the running time complexity considering various orders of large instances (n=1000 to 2900 with incremental steps of 100) over 50 trials. Table III provides the expected execution time (in milliseconds) of the MCHH, LAHC and RPHH approaches each with an associated Root Mean Square Error RMSE value to give indication on the goodness of the fit. All the three methods run in $a \cdot n \approx \mathrm{O}(n)$. However, MCHH has the smallest constant multiplier (a) and RMSE values. This indicates that MCHH runs predictably faster than RPHH and LAHC on constructing constrained magic squares.

TABLE III. Expected running time complexity of MCHH, LAHC and RPHH.

Method	Model	Multiplier	RMSE
MCHH	$a \cdot n$	$a = 1.063$	1326
LAHC	$a \cdot n$	$a = 3.311$	4745
RPHH	$a \cdot n$	$a = 1.359$	1523

To analyse the impact of changing the placement of the top-left corner of the contiguous sub-square S within the magic square, we generated 100 different randomly selected locations for n=10, 23 and 25, and 2000 different randomly selected locations for n=2600. The selected instances are the smallest and the largest orders of both sub-problems. Figures 1 and 2 provide box plots of execution time and compare the performance of MCHH, LAHC and RPHH approaches for given instances for the first sub-problem and second sub-problem, respectively. The figures show that MCHH and RPHH clearly outperform LAHC method on the selected instances. The performance of MCHH and RPHH is very similar with the exception of $n = 25$ where MCHH appears to perform better.

Figures 3 and 4 provide the average utilisation rate over 50 trials of each low level heuristic considering only the invocations that improve on the best recorded solution in hand while constructing constrained magic squares of selected instances from the first sub-problem and the second sub-problem, respectively. The figures also provide the average

TABLE II. AVERAGE EXECUTION TIME (AVR.) IN MILLISECOND, STANDARD DEVIATION (S.D.) AND THE PAIRWISE PERFORMANCE COMPARISON (A VS B) OVER 50 TRIALS. THE BEST AVERAGE VALUES ARE HIGHLIGHTED IN BOLD.

		MCHH		LAHC		RPHH		A	MCHH	MCHH	LAHC
	n	avg.	std.	avg.	std.	avg.	std.	B	LAHC	RPHH	RPHH
First sub-problem	10	**181.00**	237.00	3825.00	3221.00	250.00	456.00		$>$	$\geq$	$<$
	11	205.00	199.00	3409.00	4070.00	**164.00**	142.00		$>$	$\leq$	$<$
	13	**194.00**	162.00	4823.00	4595.00	241.00	160.00		$>$	$\leq$	$<$
	14	**314.00**	376.00	7841.00	8284.00	327.00	250.00		$>$	$>$	$<$
	15	339.00	326.00	7026.00	5603.00	**308.00**	231.00		$>$	$\leq$	$<$
	16	452.00	555.00	8356.00	8106.00	**397.00**	313.00		$>$	$\leq$	$<$
	18	741.00	890.00	8268.00	5905.00	**684.00**	632.00		$>$	$\leq$	$<$
	19	1023.00	1171.00	11325.00	10572.00	**659.00**	461.00		$>$	$\leq$	$<$
	21	1280.00	1752.00	16061.00	12340.00	**819.00**	657.00		$>$	$\leq$	$<$
	23	2264.00	2485.00	27399.00	25735.00	**1446.00**	1256.00		$>$	$\leq$	$<$
Second sub-problem	25	**7.00**	5.00	157.00	26.00	14.00	13.00		$>$	$>$	$<$
	50	**37.00**	22.00	366.00	252.00	39.00	28.00		$>$	$\geq$	$<$
	100	**36.00**	23.00	415.00	351.00	56.00	49.00		$>$	$>$	$<$
	200	**73.00**	34.00	1249.00	1140.00	113.00	85.00		$>$	$>$	$<$
	400	**176.00**	105.00	1790.00	1498.00	260.00	188.00		$>$	$>$	$<$
	800	**433.00**	206.00	3960.00	2722.00	556.00	290.00		$>$	$>$	$<$
	1000	**645.00**	440.00	4620.00	2775.00	692.00	464.00		$>$	$\geq$	$<$
	1500	**984.00**	538.00	5676.00	3957.00	1252.00	649.00		$>$	$>$	$<$
	2000	**1617.00**	987.00	6161.00	3822.00	2036.00	881.00		$>$	$>$	$<$
	2600	**3423.00**	1823.00	8142.00	4971.00	3684.00	1559.00		$>$	$\geq$	$<$

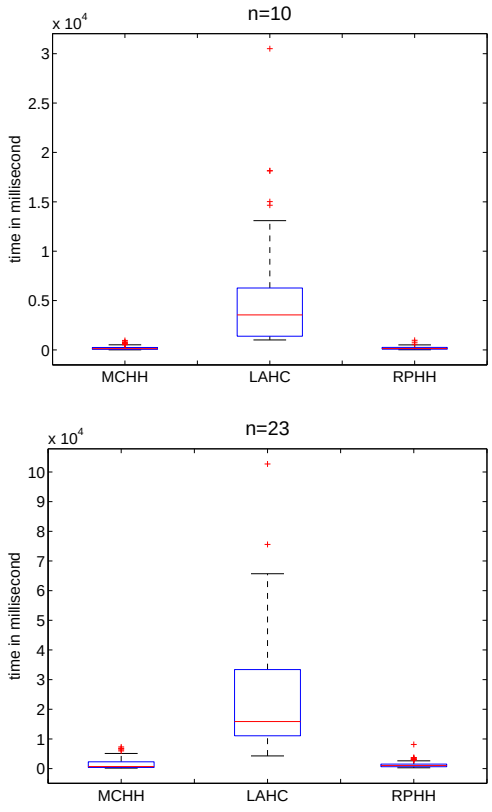


Fig. 1. Box plots of execution time for MCHH, LAHC and RPHH methods for constructing magic squares of the first sub-problem considering different locations of the contiguous sub-square S for n = 10 and 23.

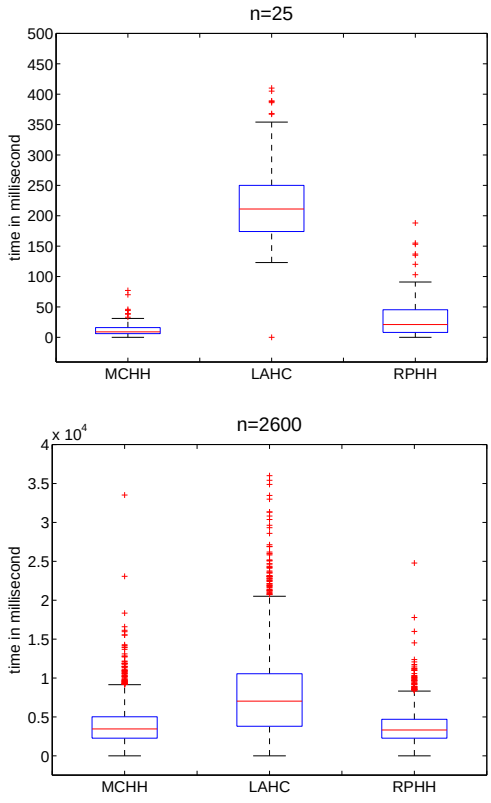
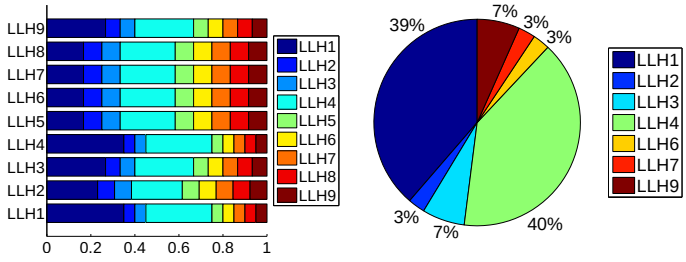


Fig. 2. Box plots of execution time for MCHH, LAHC and RPHH methods for constructing magic squares of the second sub-problem considering different locations of the contiguous sub-square S for n = 25 and 2600.

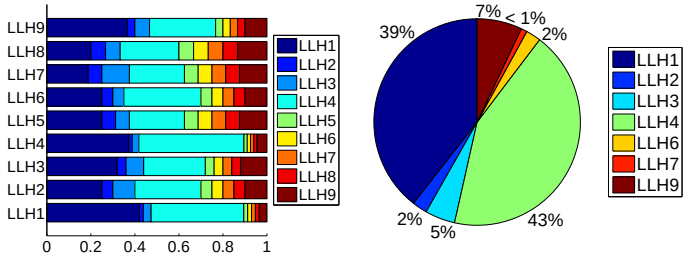
probabilities of the transitions for each low level heuristic over 50 trials. In the first sub-problem, applying LLH1 followed by applying LLH4 seems to generate most of the improvements as compared to other low level heuristics, an interesting finding. In the second sub-problem, both heuristics (LLH1 and LLH2) contribute in improving the best recorded solution in hand while solving the problem. However, LLH1 achieved slightly more improvement than LLH2.

VI. CONCLUSION

A goal in hyper-heuristic research is to raise the level of generality by providing automated hyper-heuristic methodolo-

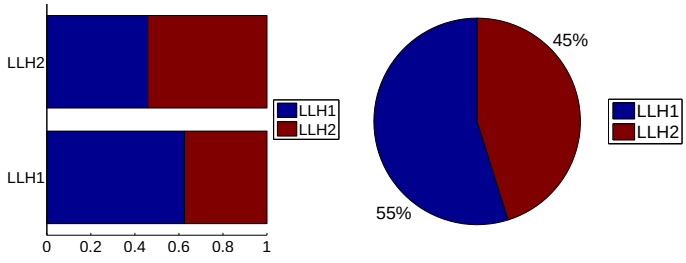


(a) n=10

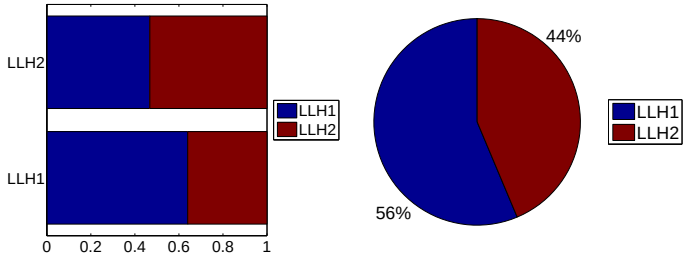


(b) n=23

Fig. 3. Average transition probabilities and utilisation rate of each low level heuristic considering moves that improve the best recorded solution in hand from 50 trials while constructing constrained magic square of the first sub-problem for n = (a) 10 and (b) 23.



(a) n=25



(b) n=2600

Fig. 4. Average transition probabilities and utilisation rate of each low level heuristic considering moves that improve the best recorded solution in hand from 50 trials while constructing constrained magic square of the second sub-problem for n = (a) 25 and (b) 2600.

gies that are able to automatically configure themselves on the fly and applicable to different problem domains without requiring any expert intervention and so additional development cost. In this study, a Markov chain selection hyper-heuristic is implemented to solve the constrained magic square problem. The proposed approach aims to dictate the order in which low level heuristics are applied. The empirical results show that the proposed method is an effective search methodology, running predictably faster than the current state-of-the-art methods in solving the constrained magic square problem. The approach adapts itself to learn good transitions between low level heuristics. As a future work, we plan to apply the method and test its level of generality on other real-world problem domains such as the water distribution problem [29].

REFERENCES

[1] E. K. Burke, M. Gendreau, M. Hyde, G. Kendall, G. Ochoa, E. Özcan, and R. Qu, "Hyper-heuristics: a survey of the state of the art," *Journal of the Operational Research Society*, vol. 64, no. 12, pp. 1695–1724, 2013.

[2] H. Fisher and G. L. Thompson, "Probabilistic learning combinations of local job-shop scheduling rules," in *Industrial Scheduling*, J. F. Muth and G. L. Thompson, Eds. New Jersey: Prentice-Hall, Inc, 1963, pp. 225–251.

[3] W. B. Crowston, F. Glover, G. L. Thompson, and J. D. Trawick, *Probabilistic and parametric learning combinations of local job shop scheduling rules*, ser. 117. Defense Technical Information Center, 1963.

[4] P. Cowling, G. Kendall, and E. Soubeiga, "A hyperheuristic approach to scheduling a sales summit," in *Practice and Theory of Automated Timetabling III*, ser. Lecture Notes in Computer Science, E. Burke and W. Erben, Eds. Springer Berlin Heidelberg, 2001, vol. 2079, pp. 176–190.

[5] G. Kendall and M. Mohamad, "Channel assignment optimisation using a hyper-heuristic," in *Proceedings of the 2004 IEEE Conference on Cybernetic and Intelligent Systems (CIS2004)*, Singapore, 2004, pp. 790–795.

[6] M. Ayob and G. Kendall, "A monte carlo hyper-heuristic to optimise component placement sequencing for multi head placement machine," in *Proceedings of the International Conference on Intelligent Technologies (InTech'03)*, Chiang Mai, Thailand, 2003, pp. 132–141.

[7] M. Kalender, A. Kheiri, E. Özcan, and E. K. Burke, "A greedy gradient-simulated annealing selection hyper-heuristic," *Soft Computing*, vol. 17, no. 12, pp. 2279–2292, 2013.

[8] E. K. Burke, J. D. Landa-Silva, and E. Soubeiga, "Multi-objective hyper-heuristic approaches for space allocation and timetabling," in *Metaheuristics: Progress as Real Problem Solvers*, ser. Operations Research/Computer Science Interfaces Series, T. Ibaraki, K. Nonobe, and M. Yagiura, Eds. Springer US, 2005, vol. 32, pp. 129–158.

[9] E. K. Burke, G. Kendall, and E. Soubeiga, "A tabu-search hyperheuristic for timetabling and rostering," *Journal of Heuristics*, vol. 9, no. 6, pp. 451–470, 2003.

[10] A. Kheiri, E. Ozcan, and A. J. Parkes, "A stochastic local search algorithm with adaptive acceptance for high-school timetabling," *Annals of Operations Research*, 2014.

[11] A. Nareyek, "Choosing search heuristics by non-stationary reinforcement learning," in *Metaheuristics: Computer Decision-Making*, M. G. C. Resende and J. P. de Sousa, Eds. Kluwer, 2003, ch. 9, pp. 523–544.

[12] D. Pisinger and S. Ropke, "A general heuristic for vehicle routing problems," *Computers and Operations Research*, vol. 34, pp. 2403–2435, 2007.

[13] K. A. Dowsland, E. Soubeiga, and E. K. Burke, "A simulated annealing hyper-heuristic for determining shipper sizes," *European Journal of Operational Research*, vol. 179, no. 3, pp. 759–774, 2007.

[14] D. Ouelhadj and S. Petrovic, "A cooperative distributed hyper-heuristic framework for scheduling," in *Proceedings of the IEEE International Conference on Man, Cybernetics, and Systems*, 2008, pp. 1232–1238.

[15] E. Özcan, B. Bilgin, and E. E. Korkmaz, "A comprehensive analysis of hyper-heuristics," *Intelligent Data Analysis*, vol. 12, no. 1, pp. 3–23, 2008.

[16] A. Kheiri and E. Keedwell, "A sequence-based selection hyper-heuristic utilising a hidden Markov model," in *Proceedings of the 2015 on Genetic and Evolutionary Computation Conference*, ser. GECCO '15. New York, NY, USA: ACM, 2015, pp. 417–424.

[17] A. Kheiri, E. Keedwell, M. J. Gibson, and D. Savic, "Sequence analysis-based hyper-heuristics for water distribution network optimisation," *13th Computer and Control for Water Industry (CCWI 2015)*, 2015.

[18] W. Van Onsem, B. Demoen, and P. De Causmaecker, "HHaaHMM: a hyper-heuristic as a hidden Markov model," http://www.hyflex.org/chesc2014/, 2014.

[19] K. McClymont and E. C. Keedwell, "Markov chain hyper-heuristic (MCHH): an online selective hyper-heuristic for multi-objective continuous problems," in *Proceedings of the 13th Annual Conference on Genetic and Evolutionary Computation (GECCO '11)*. New York, NY, USA: ACM, 2011, pp. 2003–2010.

[20] T. Xie and L. Kang, "An evolutionary algorithm for magic squares," in *IEEE Congress on Evolutionary Computation (CEC '03)*, vol. 2, 2003, pp. 906–913.

[21] A. Kheiri and E. Özcan, "Constructing constrained-version of magic squares using selection hyper-heuristics," *The Computer Journal*, vol. 57, no. 3, pp. 469–479, 2014.

[22] D. L. Anderson, "Magic squares," http://illuminations.nctm.org/Lesson.aspx?id=655, 2001.

[23] M. Kraitchik, "Magic squares," in *Mathematical Recreations*. New York: Norton, 1942, ch. 7, pp. 142–192.

[24] E. W. Weisstein, "Magic squares," http://mathworld.wolfram.com/MagicSquare.html, 2003.

[25] G. Abe, "Unsolved problems on magic squares," *Discrete Mathematics*, vol. 127, no. 1-3, pp. 3–13, 1994.

[26] K. Pinn and C. Wieczerkowski, "Number of magic squares from parallel tempering Monte Carlo," *International Journal of Modern Physics C*, vol. 9, no. 4, pp. 541–546, 1998.

[27] Y. Bykov, "The late acceptance hill-climbing algorithm for the magic square problem," http://www.yuribykov.com/IOC/LAHC_MSQ.pdf, 2011.

[28] E. K. Burke and Y. Bykov, "The late acceptance hill-climbing heuristic," Computing Science and Mathematics, University of Stirling, Tech. Rep. Technical Report No. CSM-192, 2012.

[29] K. McClymont, E. Keedwell, D. Savić, and M. Randall-Smith, "A general multi-objective hyper-heuristic for water distribution network design with discolouration risk," *Journal of Hydroinformatics*, vol. 15, no. 3, pp. 700–716, 2013.