J Math Model Algor (2013) 12:27–43
DOI 10.1007/s10852-013-9222-z

A Guide-and-Observe Hyper-Heuristic Approach to the Eternity II Puzzle

Tony Wauters · Wim Vancroonenburg · Greet Vanden Berghe

Received: 28 February 2011 / Accepted: 22 November 2011 / Published online: 1 March 2013

Abstract The present paper considers the optimisation version of the Eternity II puzzle problem and unsigned edge matching puzzles in general. The goal of this optimisation problem is to maximise the number of matching edges in the puzzle. In 2010, the META Eternity II contest awarded the best performing metaheuristic approach to this hard combinatorial optimisation problem. The winning hyper-heuristic of the contest is subject of this paper. Heuristic design decisions are motivated based on the results of extensive experiments. Furthermore, new results for the Eternity II puzzle problem are presented. The main contribution of this paper is the description of a novel guide-and-observe search mechanism combining a set of objectives. The approach significantly outperforms search methods guided by the default objective only.

Keywords Eternity II · Edge matching puzzles · Hyper-heuristics · Guide-and-observe

1 Introduction

The Eternity II (EII) puzzle is an edge matching puzzle consisting of 256 square tiles that need to be placed on a regular 16 by 16 grid. Each tile has four edges featuring a

This paper was originally intended to be published for the META Special issue of Journal of Mathematical Modelling & Algorithms but was published in an earlier issue by mistake in Vol. 11 No. 3 pp. 217–233, DOI 10.1007/s10852-012-9178-4.

T. Wauters (✉) · W. Vancroonenburg · G. Vanden Berghe
CODeS, KAHO Sint-Lieven, Gent, Belgium
e-mail: tony.wauters@kahosl.be

W. Vancroonenburg
e-mail: wim.vancroonenburg@kahosl.be

G. Vanden Berghe
Department of Computer Science, K.U.Leuven, Leuven, Belgium

coloured pattern. The goal of the puzzle is to rotate these tiles and place them on the grid in such a way that the shared edge between any two adjacent tiles is matched, i.e. the patterns of adjacent tiles at the shared edge should be the same. Furthermore, some tiles have *grey* edges that need to be matched to the border of the puzzle.

The Eternity II puzzle was created by Christopher Monckton and released by toy distributor Tomy UK Ltd. in July 2007. The company pledged to offer a cash prize of $2 million to the first person that completely solves the puzzle. However, as of December 31st, 2010 no complete solution has been submitted and the final scrutiny date for the cash prize has passed. In 2010 a competition was organized for the 3rd International Conference on Metaheuristics and Nature Inspired Computing (META'10, Djerba Island, Tunisia). Contestants were required to develop a meta-heuristic or a nature inspired algorithm for solving Eternity II puzzles. However, the focus was not on solving the puzzle completely, but rather on trying to find the best partial solution. Thus, one in which all tiles are assigned a position on the board capable of maximising the number of matched edges. Contestants were asked to test their algorithms on a set of benchmark instances, and to submit their algorithm while reporting on their results. From these submissions, three finalists were selected and invited to present their approach at the META'10 conference. In a final selection round, the finalists' submitted algorithms were tested on the real Eternity II puzzle. The authors presented the best performing algorithm (results averaged over 30 runs of 1 h each) and they were declared the winner. In this paper, we present the components of the winning solution method.

Section 2 gives an overview of the available literature related to optimising the Eternity II puzzle. In Sections 3 and 4 we elaborate on the details, algorithmic components and the parameter settings. In Section 5 we discuss the experiments that eventually have led to the algorithm submitted to the META'10 contest. Finally, we conclude in Section 6 and give directions for future research.

2 Literature Overview

The Eternity II puzzle belongs to a more general class of Edge Matching Puzzles (EMPs). Demaine and Demaine [7] show that for EMPs in general, determining whether a complete solution exists is NP-complete. Antoniadis and Lingas [2] show that the problem is APX-complete, thus proving that Edge Matching Puzzles do not admit polynomial-time approximation schemes unless $P = NP$.

Due to the nature of the problem, several CSP approaches have been developed, in addition to some evolutionary and meta-heuristic methods. Table 1 shows an overview of the literature on optimisation approaches to Eternity II.

Ansótegui et al. [1] studied generic edge matching puzzles with frame (GEMP-F) as SAT/CSP benchmarks. They provide some hardness experiments for both one-set GEMP-Fs and two-set GEMP-Fs. Several SAT and CSP solvers are compared on self generated instances of size $n = \{7, 8\}$ with different numbers of colours for one-set GEMP-Fs and on instances of size $n = \{6, 7\}$ for two-set GEMP-Fs.

Schaus and Deville [15] described a successful two-phase approach to the real EII puzzle. In the first phase an initial solution is constructed by means of a constraint programming approach applied to relaxed problem instances. These relaxed instances are based on the original instances in which some of the edge-matching

Table 1 Literature overview summary for Eternity II

	Methods	Puzzle type	$n_{\max}$
Ansótegui et al. [1]	SAT/CSP	Benchmark	7
Schaus and Deville [15]	CSP, tabu search	Real puzzle	16
Heule [10]	SAT	Clue puzzles, benchmark	14
Muñoz et al. [13]	GA, MOEA	Real puzzle	16
Wang and Chiang [18]	Tabu search	Real puzzle, benchmark	16
Coelho et al. [6]	GVNS	Real puzzle, benchmark	16
Vancroonenburg et al. [17]	Hyper-heuristic	Real puzzle, benchmark	16

constraints of non edge-adjacent pieces are excluded. A second phase is a tabu search method based on a very efficient large neighbourhood that optimally swaps and rotates a set of non edge-adjacent pieces. They are capable of reaching scores up to 458/480 on the real puzzle.

Edge Matching Problems are translated into SAT problems and solved using different SAT solvers in Heule [10]. The instances under consideration are the real clue puzzles and some benchmark puzzles up to size $n = 14$.

Muñoz et al. [13] consider a genetic algorithm and a multi-objective evolutionary algorithm. Some simple crossover and mutation methods, based on the exchange and rotation of regions, are applied. Further on, multiple objectives are used and combined into one weighted objective. Experiments showed that these evolutionary approaches are not able to outrun an exhaustive search. The highest score reached on the real puzzle was 371/480.

At the META'10 contest, several metaheuristic approaches for solving Edge Matching Puzzles were compared. The experiments were performed on the real puzzle and on a number of benchmark problems up to size $n = 16$. The allowed time for solving the puzzles of size $n = 16$ was 60 min. Wang and Chiang [18] described a two-phase approach mainly based on tabu search [9], using simple swap and rotate neighbourhoods. Coelho et al. [6] proposed an approach combining multi-start and general variable neighbourhood search (GVNS), also using simple swap and rotate neighborhoods. Vancroonenburg et al. [17], described a two-phased hyper-heuristic method combined with a backtracking search for generating initial solutions. Several basic neighbourhoods were used, including swap and rotate, in addition to more advanced ones. The approach, that is elaborated on in this paper, was able to reach scores up to 460/480 on the real puzzle and was also the winner of the contest.

3 Problem Description

In the optimisation version of the general unsigned[1] edge matching puzzle problem, a set of $N = n^2$ square tiles need to be rotated and placed on an $n \times n$ grid (Fig. 1).[2] Four coloured patterns (colours for short) are associated with each tile,

[1]The *signed* edge matching puzzle problem considers tiles with edges both having a colour and a *sign* (+/−). The edge colours between any two adjacent tiles should match, and the signs should be opposed.

[2]A more general problem definition could allow $N = n \times m$ tiles to be placed on an $n \times m$ grid. However, without much loss of generality, we will only focus on square grids.

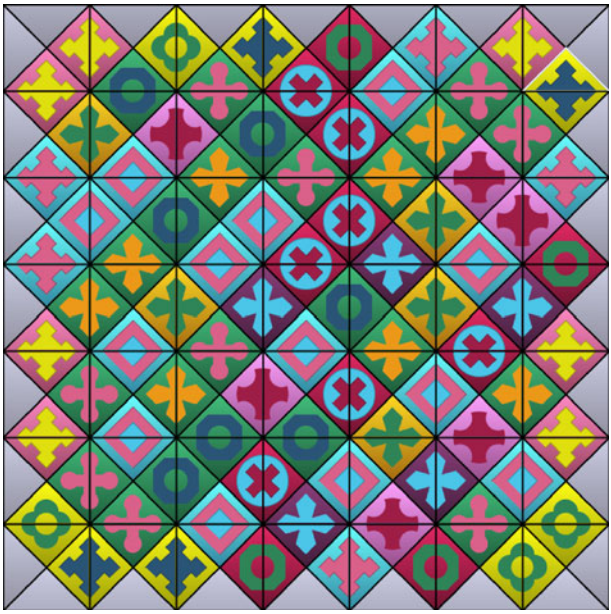

Fig. 1 Edge matching puzzle of size 7×7 (Image created using Eternity II editor, http://eternityii.sourceforge.net/, accessed on February 21st, 2011)

corresponding to the four sides of a tile. Some tiles have one or two sides coloured with a specific colour, *grey*. The grey sides should be matched to the outer edge of the grid. In the present paper, this is considered as a hard constraint. Thus, tiles with grey edges are not allowed to appear on the inner part of the puzzle. In total C coloured patterns (C is part of the input of the problem description) are distributed over all tiles, of which usually C_{inner} only appear on border tiles. The goal of the optimisation problem is to minimise the number of unmatched edges between adjacent tiles on the grid.

The Eternity II puzzle instance is an unsigned edge matching puzzle problem instance of size $n = 16$. It has 22 coloured patterns ($+1$ for the grey frame), of which five only appear on border tiles and not on inner tiles. The instance was designed in such a way that no two tiles are the same. The Eternity II puzzle rules specify that a certain tile (tile 139) belongs at position row 9, column 8. However, we do not consider such *hint* pieces as part of the problem description and thus ignore this.

4 Hyper-Heuristic Approach

The approach presented here is based on a common hyper-heuristic framework [3]. A hyper-heuristic is an iterative search methodology that applies a set of low-level heuristics to solve a given problem. These low-level heuristics can be neighbourhood functions or (meta)heuristics operating on the problem and possibly including domain specific knowledge. Unlike metaheuristics, hyper-heuristics do not search in the space of solutions, but in the space of low-level heuristics. One assumes

that no domain specific knowledge is transferred to the hyper-heuristic. Recently, many different hyper-heuristics have been presented in the literature. In this work the hyper-heuristic uses two decision mechanisms to operate: a heuristic selection method and an acceptance criterion. According to Burke et al. [4] this hyper-heuristic can be classified as 'no-learning' with 'heuristic selection' and 'perturbation heuristics', based on the nature of the heuristic search space and the source of feedback during learning. A pseudocode of the hyper-heuristic framework is given in Algorithm 1. S_0 refers to the initial solution, S is the current solution and S^* refers to the best solution found. At each iteration, the heuristic selection mechanism, denoted as function *SelectHeuristic*(), has to select one of the available low-level heuristics, represented by simple perturbative neighbourhoods. This heuristic H is then applied to the current solution to generate the new solution S'. The acceptance criterion, denoted as function *Accept*(), has to decide whether this newly generated solution can be accepted or not. In what follows we describe the different components of the hyper-heuristic developed for the Eternity II puzzle.

Algorithm 1 Pseudocode for the hyper-heuristic framework

```
S_0 ← GenerateInitialSolution()
S ← S_0
S* ← S_0
while !Terminate() do
    H ← SelectHeuristic()
    S' ← H(S)
    if Accept(f(S), f(S')) then
        S ← S'
        if f(S) < f(S*) then
            S* ← S
        end if
    end if
end while
return S*
```

4.1 Solution Modelling

We use a straightforward solution representation for the Eternity II puzzle. The puzzle board is modelled as a matrix of *tile assignments*, where each element of the matrix corresponds to a position of the puzzle grid. A tile assignment is the assignment of a certain tile t_i ($i = 1 \ldots N$) with a specific rotation to a position on the grid. It is represented as an ordered pair of both the identifier of the tile, and its rotation (0,1,2,3 for resp. 0°, 90°, 180°, 270°) on the board (Fig. 2). We denote a tile assignment on row r, column c as $ta_{r,c}$, and its value is of the form (t_i, rot). Rows are indexed from top to bottom, columns from left to right. For example, the assignment of tile 5 with a rotation 1 (i.e. 90°) to row 1, column 2 can be denoted as $ta_{1,2} = (t_5, 1)$. The entire grid of the puzzle can be denoted as $(ta_{r,c})$ or optionally, when the size of the grid is not clear from the context, as $(ta_{r,c})_{n \times n}$.

Four colours are associated with each tile, corresponding to the four sides. They are indexed starting from the top edge and going clockwise (i.e. 0 = top, 1 = right,

Fig. 2 Visualization of the solution representation. A solution is represented as a matrix of tile assignments, which are ordered pairs of the tile identifier and its rotation. Tiles that contain *grey* edges are only allowed at the border of the matrix

etc.), and they are denoted as $c(t_i, j)$. For example, the left colour of tile 5 is $c(5, 3)$. The special colour *grey*, denoting tiles that need to be matched to the grid edge, is assigned value 0. This notation allows to unambiguously define the colours of a tile assignment $ta_{r,c} = (t_i, rot)$ as:

$$c\left(ta_{r,c},\ j\right) = c\left(t_i,\ j - rot \bmod 4\right) \tag{1}$$

with j the edge index.

In order to reduce the number of possible solutions, tiles that belong at the border (i.e. containing grey edges) are only assigned to tile assignments at the border (and vice versa for non-border tiles). Furthermore, the tile assignment at the top left (row 1, column 1) is fixed to one random corner tile (i.e. a tile with two grey edges) to avoid rotated solutions. The perturbative heuristics described in Section 4.5 maintain these properties.

4.2 Initial Solution Generation

The hyper-heuristic search starts from an initial solution that can be generated either at random or by a constructive method. The random method used in this work, randomly places all corner pieces at the corners, border pieces at the border, and all inner pieces at the inner region of the board. In addition, a backtracking algorithm with different placement strategies is used as constructive method. Examples of placement strategies are: scan-row, spiral, inverse-spiral, mirrored scan-row.

- **Scan-row(SR)** (Fig. 3a): places the pieces from left-to-right and from top-to-bottom.
- **Spiral(SP)** (Fig. 3b): places the pieces in a circular movement from the border to the center.
- **Inverse-spiral(ISP)** (Fig. 3c): places the pieces in a circular movement from the center to the border.

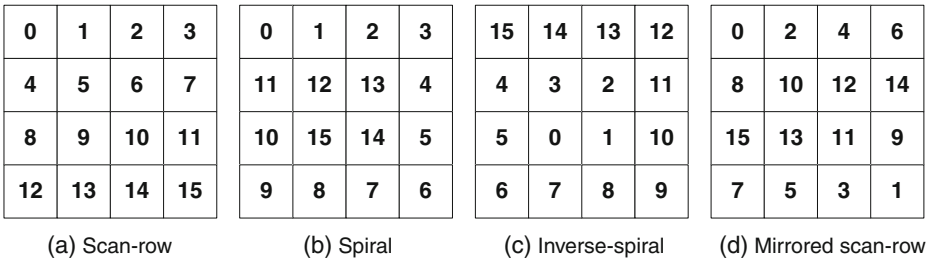

Fig. 3 Placement strategies for backtracking search

- **Mirrored scan-row(MSR)** (Fig. 3d): places the pieces in a way similar to scan-row, but alternates between (left-to-right, top-to-bottom) and a (right-to-left, bottom-to-top) placement.

In the experiments described later in this paper, it is shown that an initial solution generated with backtracking using a scan-row placing strategy yields the best results.

4.3 Heuristic Selection

Many heuristic selection methods exist, some of which use on-line (see e.g. [5, 12, 14]) or off-line learning based on the feedback of the generated solutions. In this work a simple random selection method is applied. A low-level heuristic or neighbourhood is selected at random during each iteration. Given that the number of low-level heuristics (see Section 4.5) is low, the benefit of more complex heuristic selection methods is not expected to outweigh the increase in complexity.

4.4 Acceptance Criteria

For the acceptance of solutions one can use different criteria, including Simulated Annealing [11], Only Improving, Improving or Equal, and Great Deluge [8]. Experiments show (see Section 5.1) that the recently introduced Iteration Limited Threshold Accepting (ILTA) [12] criterion performs slightly better than other methods. ILTA is similar to improving or equal acceptance but it can accept worsening solutions after a number (k) of consecutive worsening solutions. These worsening solutions can only be accepted if their quality is within a certain range R of the current best solution's objective value. ILTA's two parameters (k and R) are more understandable and thus easier to fine-tune compared to, for example, the parameters in Simulated Annealing.

4.5 Low-Level Heuristics

Many low-level heuristics or neighbourhoods can be devised. The most obvious ones are based on simply swapping and rotating pieces on the board. The following three swap and rotate heuristics are applied:

– swap and rotate two corner pieces,
– swap and rotate two border (not corner) pieces,
– swap and rotate two inner pieces.

In each of the above heuristics, a completely random decision is made concerning the selection of tiles to reassign. Furthermore, a *Probabilistic partial optimal placement* heuristic of q pieces is used. The 'partial optimal placement' heuristic was introduced by Schaus and Deville [15], who call it a very large neighbourhood. The heuristic removes q non-adjacent tiles from the board thus creating q holes. The tiles are then optimally reinserted into the holes by solving an assignment problem, for which an efficient algorithm, the Hungarian method, exists. In this work, we use a probabilistic version of the partial optimal placement heuristic. It selects q tiles proportionally to their number of unmatched edges, using a roulette-wheel selection method. We created two versions of this heuristic, one for the inner part and one for the corner/border tiles. For the inner part, q has been set to 24, for the border $q = 16$. As reported by Schaus and Deville [15], increasing q further does not result in significant improvement.

4.6 Objectives

One of the strategies for improving local search and avoiding local optima is to change the fitness landscape of the problem [16]. Different objective functions effectively guide the search through different search landscapes. We present a novel approach that adopts some ideas of existing methods. We consider the following objective functions (Fig. 4):

- **Obj1**, Matched edges: This is the default optimisation objective that counts the number of matched edges in the solution (Fig. 4a).
- **Obj2**, Complete 2×2 squares: This objectives counts the number of completely correct 2×2 regions in the puzzle. A completely correct 2×2 region has all its inner edges matched. Outer edges on the border of the 2×2 region are not checked (Fig. 4b).
- **Obj3**, Perfect tiles: This objective counts the number of perfectly correct tiles. These tiles have all their four edges matched (Fig. 4c).
- **Obj4**, Complete 3×3 squares: This objective counts the number of completely correct 3×3 regions in the puzzle. It is the 3×3 version of $Obj2$ (Fig. 4d).
- **Obj5**, Complete 4×4 squares: It is the 4×4 version of $Obj2$.

Note that a good objective value for one of the new objective functions does not necessarily correspond to a good objective value for the default objective ($Obj1$). This implies that a search method guided by one of the new objective functions may get stuck in a local optimum that is not a local optimum in the default search landscape. That observation is the basic idea behind the new *guide-and-observe* approach. The hyper-heuristic search method consists of two phases. The search is first *guided* by one of the new objective functions ($Obj2 - 5$), while in the second phase it is guided by the default objective function. In order not to miss any good solutions with respect to the default objective, the solution quality in terms of $Obj1$ is continuously being *observed* during the first phase. The search in the second phase starts from the solution with the overall best $Obj1$ value. This two-phase approach is described in Algorithm 2. Do note, however, that the optimal solutions considering the new objective functions correspond to an optimal solution of the default objective ($Obj1$).

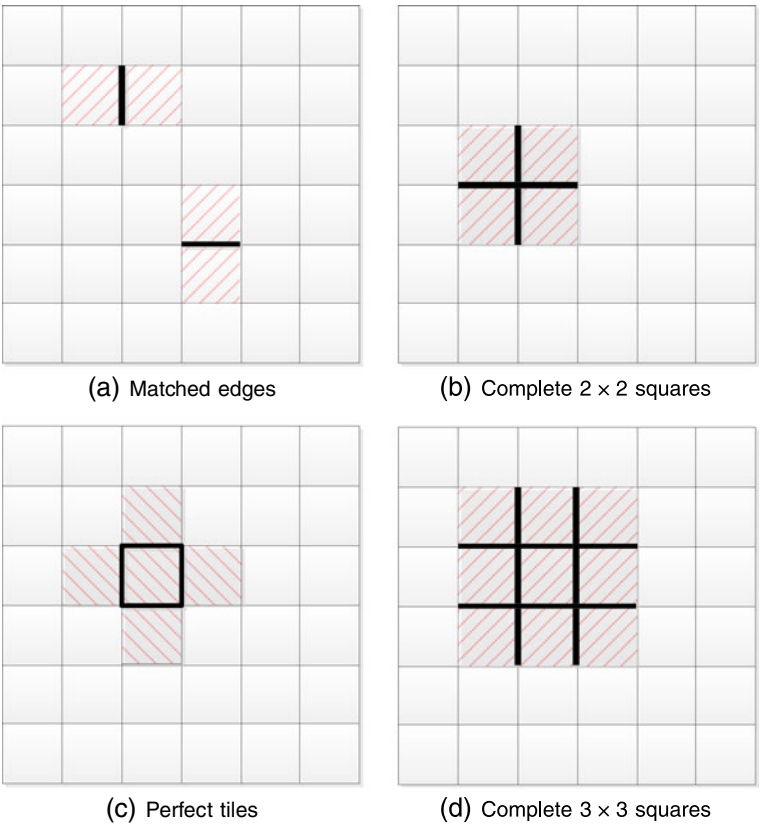

Fig. 4 Objective functions: *hatched areas* indicate the tiles under consideration, *bold edges* indicate the checked edges. **a** is the default objective, **b** counts all completely correct 2×2 square regions, **c** counts all perfectly placed tiles and **d** counts all completely correct 3×3 regions

Algorithm 2 Pseudocode for the two phase guide-and-observe framework.

$S_0 \leftarrow GenerateInitialSolution()$
$S_{bestObj1} = HyperHeuristic(S_0, timeout = t_1, objective = Obj_X)$
$S^* = HyperHeuristic(S_{bestObj1}, timeout = t_2, objective = Obj_1)$
return S^*

Using the notation introduced in Section 4.1 the above objective functions can be further formalized:

– **Obj1**: let $m(ta_{r1,c1}, ta_{r2,c2})$ denote whether the adjacent tile assignments $ta_{r1,c1}, ta_{r2,c2}$ match (value = 1) or not (value = 0). Then:

$$Obj1 = \sum_{r=1}^{n} \sum_{c=1}^{n-1} m(ta_{r,c}, ta_{r,c+1}) + \sum_{c=1}^{n} \sum_{r=1}^{n-1} m(ta_{r,c}, ta_{r+1,c})$$

– **Obj2**: let $sq(ta_{r,c})$ denote whether the square region $[r, r+1] \times [c, c+1]$ is internally completely correct (value = 1) or not (value = 0). Then:

$$Obj2 = \sum_{r=1}^{n-1} \sum_{c=1}^{n-1} sq(ta_{r,c})$$

– **Obj3**: let $perfect(ta_{r,c})$ denote whether the tile assignment $ta_{r,c}$ is *perfectly* placed, i.e. $m(ta_{r-1,c}, ta_{r,c}) = m(ta_{r,c}, ta_{r+1,c}) = m(ta_{r,c-1}, ta_{r,c}) = m(ta_{r,c}, ta_{r,c+1}) = 1$, or not. Then:

$$Obj3 = \sum_{r=1}^{n} \sum_{c=1}^{n} perfect(ta_{r,c})$$

– **Obj4**: let $tr(ta_{r,c})$ denote whether the square region $[r, r+2] \times [c, c+2]$ is internally completely correct (value = 1) or not (value = 0). Then:

$$Obj4 = \sum_{r=1}^{n-2} \sum_{c=1}^{n-2} tr(ta_{r,c})$$

– **Obj5**: let $qu(ta_{r,c})$ denote whether the square region $[r, r+3] \times [c, c+3]$ is internally completely correct (value = 1) or not (value = 0). Then:

$$Obj5 = \sum_{r=1}^{n-3} \sum_{c=1}^{n-3} qu(ta_{r,c})$$

5 Experiments

All experiments apply the following low-level heuristics or neighbourhoods: swap-rotate (corner, border, and inner), probabilistic partial optimal assignment (border+corner and inner). One of them is always selected randomly. All tests are performed 30 times.

In the following experiments, a time limited termination condition leads to the same findings as an iteration limited termination condition. This can be explained as follows: the objectives and the acceptance criteria have a marginal performance impact on the number of iterations that can be performed in a fixed time. In the experiments, the speed of the hyper-heuristic search is about 4000 iterations/second on an Intel Core 2 Duo 2.8 GHz processor.

5.1 Acceptance Criteria

Table 2 shows a comparison of the guide-and-observe hyper-heuristic with different acceptance criteria: Improving or Equal (IE), Accept All (AA), ILTA, Great Deluge (GD) and Simulated Annealing (SA). The settings are as follows: 100, 000 iterations, *Obj*4, scan-row backtracking (10 s) starting solution. For both ILTA and SA, the best parameter settings resulting from the experiments were used. The accept all criterion performs badly because it does not guide the search towards better solutions. The other acceptance criteria, which all accept improving or equal solutions, perform very well. ILTA results in the highest average score, and thus it is used in the final algorithm. This best ILTA had the following parameter settings: $k = 500$, $R = 1.4$.

Table 2 Comparison of the guide-and-observe hyper-heuristic approach with different acceptance criteria

	IE	AA	**ILTA**	GD	SA
Max.	456	426	**458**	455	456
Min.	427	372	**448**	448	450
Average	451.3	403.2	**452.9**	452.1	452.4
Stdev.	5.2	13.9	**2.7**	2.0	1.7

5.2 Objectives

Figure 5 shows the different behaviour of the search guided by the five different objectives. The tests were performed using a random starting solution, an improving or equal acceptance criterion and 100,000 iterations for each phase. An improving or equal acceptance criterion is used to reduce the effects of more advanced acceptance mechanisms like ILTA. The stacked bar chart shows for each objective *x*:

- the average best solution, in terms of *Obj*1, reached while optimising *Objx* (Phase 1),
- the average improvement, in terms of *Obj*1, realized by observing the best solution for *Obj*1 while optimising *Objx* (Phase 1—best observed),
- and the average improvement, in terms of *Obj*1, realized by optimising *Obj*1 in a second phase with the same settings (Phase 2).

The results show the benefit of optimising other objectives than the default one. Moreover, observing the *Obj*1 values while optimising an other objective leads to higher values. A second optimisation phase on *Obj*1 additionally results in

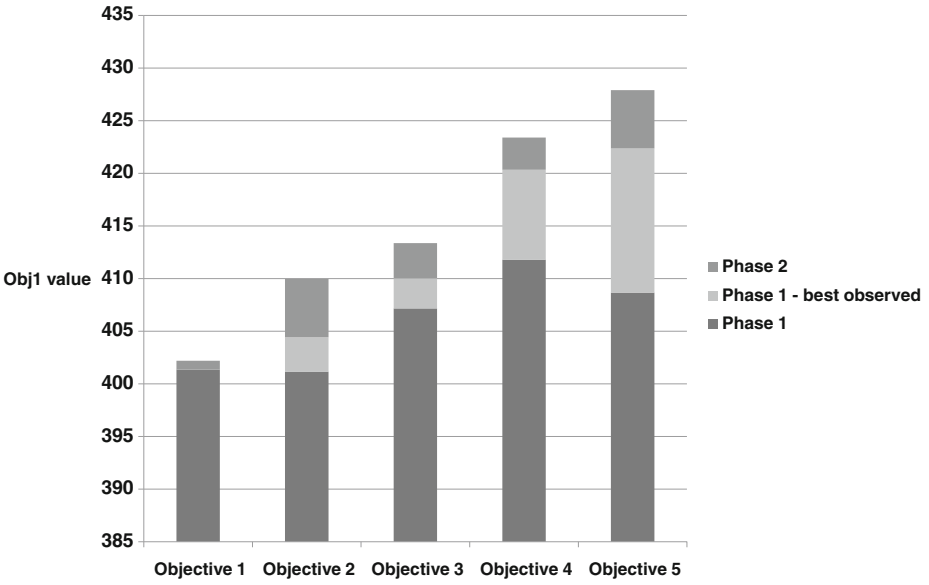

Fig. 5 Comparison of objectives for a two-phase guide-and-observe hyper-heuristic search with 100,000 iterations for each phase, starting from a random solution

an increase of the best observed solution from the first phase. For example, *Obj*4 reaches the highest score, while *Obj*5 reaches the highest observed score in phase 1.

Figure 6 shows the same experiment but now starting from a solution constructed by applying a scan-row backtracking search during 10s. Due to the improved starting solution, the scores are higher than the ones started from a random solution. *Obj*4 and *Obj*5 are still the best objectives, but the difference between the two is very small.

Figures 7 and 8 show a run of a hyper-heuristic search with *Obj*4 for an IE and an ILTA ($k = 500$, $R = 1.4$) acceptance criterion. The searches started from a random solution. The figures present both the *Obj*4 and the *Obj*1 values. The results show that for an equal *Obj*4 value, multiple *Obj*1 values are observed, and thus exploration continues while visiting solutions with equal objective values. This is a result of the equality condition in the acceptance criteria. The ILTA run shows that worsening the solution can help to eventually reach better solutions.

An interesting observation is that *Obj*4 and *Obj*5 combined with the partial optimal assignment move, show an avalanche-like effect. Whereas at the beginning of the search it is difficult to find improvements, this combination continues to generate improvements more quickly after the first one, until it finally converges. This can easily be seen in Fig. 9. It presents the cumulative *Obj*1 improvement found during the search guided by *Obj*5. It is very hard to start from a random solution with *Obj*5 (on a puzzle of size 16×16). However, once started, it results in higher observed *Obj*1 values. For larger puzzles (size $> 16 \times 16$), *Obj*5 potentially has higher benefits. Due to *Obj*5′*s* starting problems, we prefer applying *Obj*4 in further experiments, however.

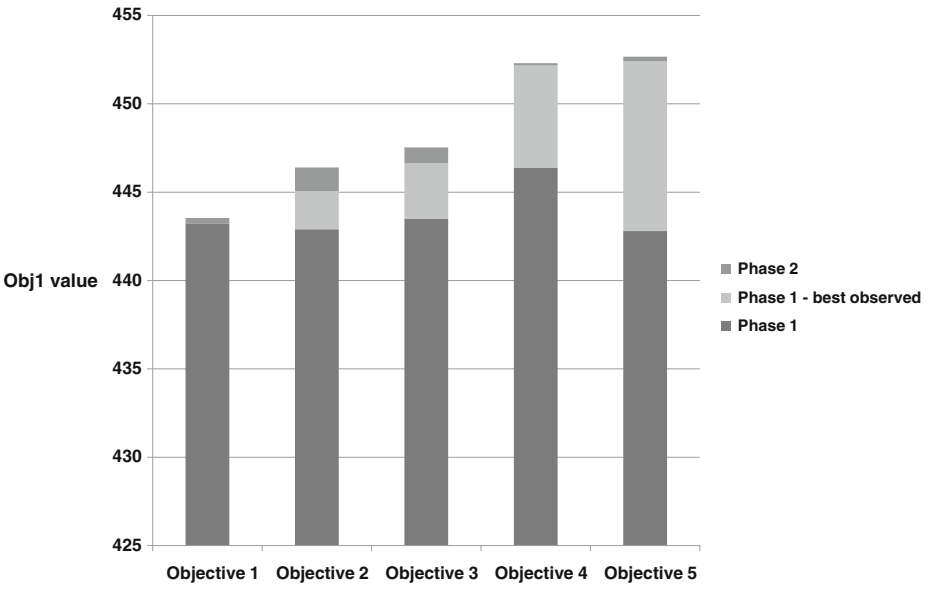

Fig. 6 Comparison of objectives for a two-phase guide-and-observe hyper-heuristic search with 100,000 iterations for each phase, starting from a 10 s scan-row backtracking solution

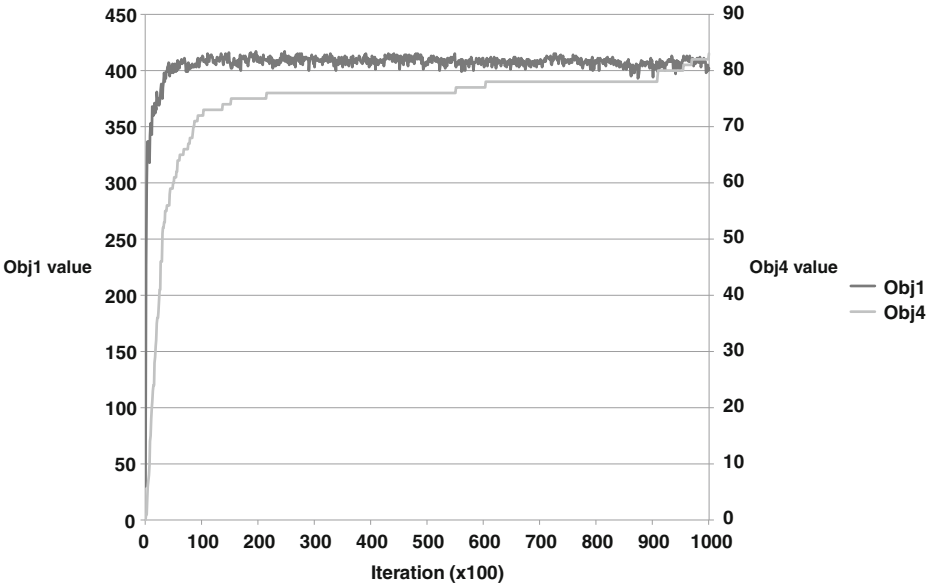

Fig. 7 Guide-and-observe hyper-heuristic run starting from a random solution, optimizing *Obj*4, improving or equal acceptance criterion

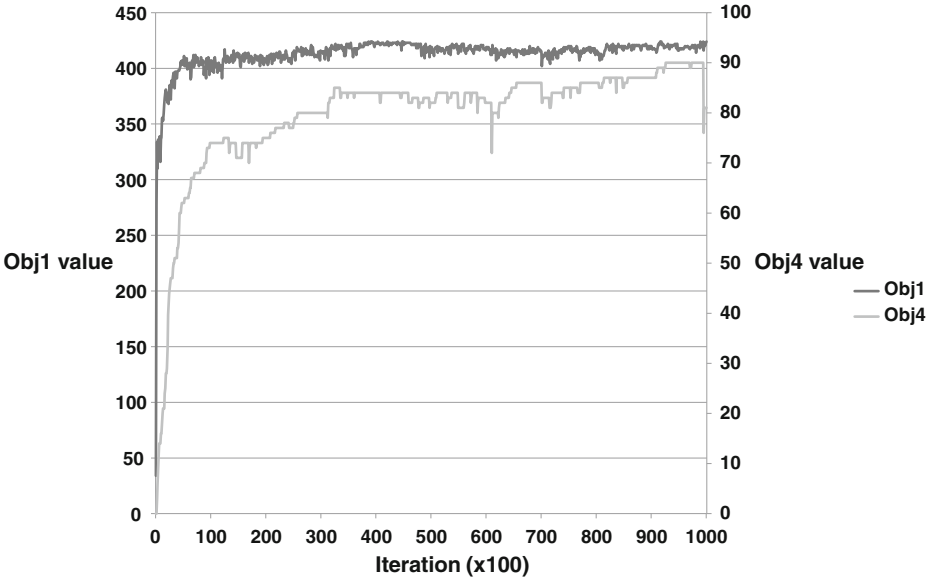

Fig. 8 Guide-and-observe hyper-heuristic run starting from a random solution, optimizing *Obj*4, ILTA ($k = 500$, $R = 1.4$) acceptance criterion

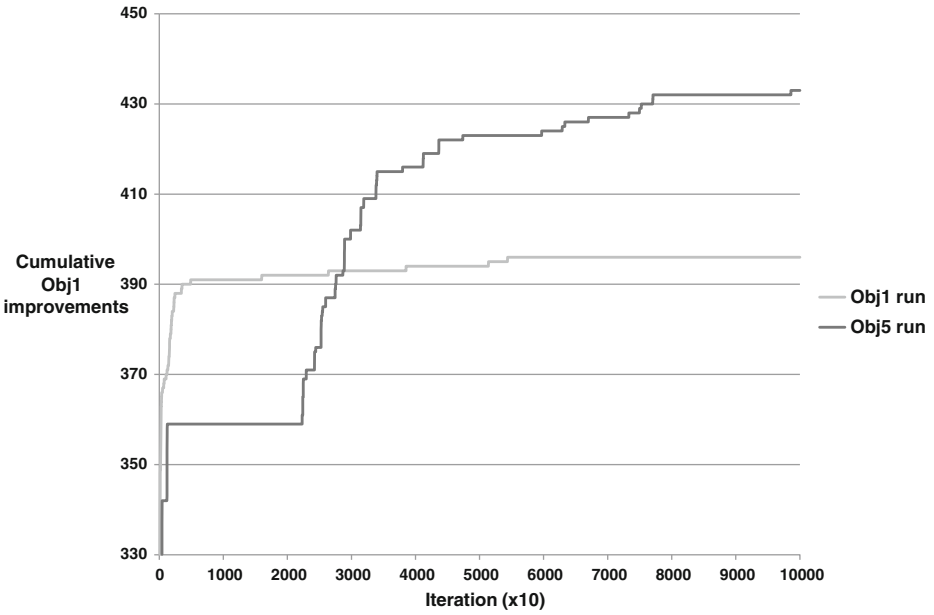

Fig. 9 *Obj*5 avalanche-like effect. Comparison of the cumulative *Obj*1 improvements during a search guided by *Obj*1 or *Obj*5. Single phase hyper-heuristic search for 100,000 iterations, IE acceptance criterion, random starting solution

5.3 Starting Solution

The starting solution can be very important for the final result. Therefore, some experiments were performed to find the best starting solution. Table 3 shows a comparison of the results for an improving or equal hyper-heuristic search, starting from a random solution and four different backtracking solutions. For each method, 30 runs of 100, 000 iterations were carried out, and *Obj*4 was used. The backtracking solutions were constructed by using a backtracking search that was interrupted after 10 s, and that made use of one of the following search orders: scan-row, spiral, inverse-spiral and mirrored scan-row. In Table 3, it can be seen that a backtracking search using a scan-row strategy leads to the best results. A statistical t-test (95% confidence) revealed that it performed significantly better than the other methods. Similar experiments were conducted for the other objectives leading to the same findings.

Table 3 Comparison of results for different starting solution generation methods, followed by a one-phase hyper-heuristic search of 100,000 iterations

	Random	BT SR (10 s)	BT SP (10 s)	BT ISP (10 s)	BT MSR (10 s)
Max.	430	**456**	441	447	448
Min.	413	**438**	413	432	422
Average	421.2	**451.4**	423.5	442.7	438.2
Stdev.	3.7	**3.4**	7.1	3.3	7.9

Table 4 Influence of time given to a scan-row backtracking search, followed by a one-phase hyper-heuristic search of 100,000 iterations

	0.5 s	1 s	3 s	6 s	10 s	30 s	60 s	600 s	3,600 s
Max.	453	455	455	455	458	456	457	458	457
Min.	434	429	436	446	438	448	448	452	451
Average	443.9	447.4	448.6	451.3	451.5	452.3	453.0	454.0	454.2
Stdev.	6.4	5.9	4.8	2.1	4.1	1.8	1.8	1.5	1.6

Table 4 shows the influence of the amount of time given to a scan-row backtracking search for generating the starting solution. The same settings were used as in the previous experiment. The table shows that a time between 6 and 60 s gives a good time/quality trade-off. Giving more time to the scan-row backtracking search does not result in large improvements.

5.4 Global Results

The final algorithm submitted to the META'10 Eternity II competition used the following configuration:

- Initial solution generated using a scan-row backtracking algorithm (time limited to 10 s);
- First phase hyper-heuristic search with an ILTA ($k = 500, R = 1.4$) acceptance criterion for time $t_1 = \frac{3}{4} \times T$, optimising on $Obj4$ and observing the best $Obj1$ solution;
- Second phase hyper-heuristic search starting from the best observed solution in phase 1 for time $t_2 = \frac{1}{4} \times T$, with an ILTA ($k = 500, R = 1.4$) acceptance criterion and optimising on $Obj1$;
- All reported low-level heuristics are used.

The following results were generated on a desktop pc with (CPU: Intel Core 2 Duo 3 Ghz, Windows XP SP3, Java 1.6 JRE). We performed 30 runs with a predetermined time (T), for each instance of a certain size.

- $10 \times 10 \rightarrow 1200$ s
- $12 \times 12 \rightarrow 1800$ s
- $14 \times 14 \rightarrow 2400$ s
- $16 \times 16 \rightarrow 3600$ s

Table 5 shows the results of these runs on four benchmark puzzles (10×10, 12×12, 14×14 and 16×16) and on the real Eternity II puzzle. The benchmark puzzles

Table 5 Results of the two-phase guide-and-observe hyper-heuristic approach on 4 benchmark instances and the real EII puzzle, using a fixed time limit

	10×10	12×12	14×14	16×16	Real EII puzzle
Number of total edges	180	264	364	480	480
Max.	172	254	348	460	461
Min.	168	250	344	453	455
Average	170.4	251.9	345.8	457.3	457.6
Stdev.	0.9	0.9	1.0	1.5	1.5
Time for best over all (s)	46	656.4	762.4	1563.9	2871.97

Table 6 Comparison with META'10 Eternity II contest finalists, average results over 30 runs

	10×10	12×12	14×14	16×16
Number of total edges	180	264	364	480
Present work	**170.4**	**251.9**	**345.8**	**457.3**
Coelho et al. [6]	165.67	238.93	320.77	419.57
Wang and Chiang [18]	159.57	232.33	312.57	408.63

and the execution times are the same as the ones that were provided at the META'10 Eternity II contest. The maximum, minimum, average, and standard deviation of the number of matching edges over 30 runs are reported. We also report the time (in seconds) needed to obtain the best solution over all 30 runs. The results show that the novel approach is capable of matching up to 96% of the edges over all the puzzles. It generated a solution with 461 matching edges on the real Eternity II puzzle, which is a very competitive result. Furthermore, Table 6 shows that the approach compares favourably to the results obtained by the two other finalists of the META'10 Eternity II contest.

6 Conclusion

The contribution of the paper is the introduction of a new guide-and-observe hyper-heuristic approach to optimising Eternity II -like puzzles. The power of the approach can be attributed to the novel parallel observation of objective values, to the set of objectives, and to the efficient hyper-heuristic. We have shown that optimising an objective different from the default objective, allows to reach solutions with a higher number of matching edges. The hyper-heuristic is applied in a two-phase approach. During the initial phase of the search, a new 3×3 squares objective is applied. Optimising the default objective in a second phase, starting from the best observed solution (in terms of the default objective) in the first phase, improves the quality of the solutions even more. The winning approach of the META'10 Eternity II contest is capable of reaching scores up to 461/480 matching edges for the real Eternity II puzzle in less than one hour of computation time. Such a result can be considered very acceptable, taking into account the past research on the topic.

In the future, the applicability of the novel guide-and-observe idea will be further investigated for other combinatorial optimisation problems and in connection with other metaheuristics and hyper-heuristics. In this light, the research question is how to design objective functions that are more appropriate for guiding a local search method, while remaining sufficiently correlated with the real objective of the problem.

With respect to the Eternity puzzle, other objective functions, exploring more advanced heuristic selection mechanisms and new low-level heuristics or neighbourhoods may be interesting directions for future research.

References

1. Ansótegui, C., Béjar, R., Fernàndez, C., Mateu, C.: Edge matching puzzles as hard sat/csp benchmarks. In: CP '08 Proceedings of the 14th International Conference on Principles and Practice of Constraint Programming (2008)

2. Antoniadis, A., Lingas, A.: Approximability of edge matching puzzles. In: Proceedings of the 36th Conference on Current Trends in Theory and Practice of Computer Science, SOFSEM '10, pp. 153–164. Springer, Berlin, Heidelberg (2010)
3. Burke, E., Kendall, G., Newall, J., Hart, E., Ross, P., Schulenburg, S.: Hyper-heuristics: an emerging direction in modern search technology. In: Glover, F., Kochenberger, G. (eds.) Handbook of Metaheuristics. International Series in Operations Research & Management Science, vol. 57, chapter 16, pp. 457–474. Springer, New York (2003)
4. Burke, E.K., Hyde, M., Kendall, G., Ochoa, G., Özcan, E., Woodward, J.R.: A classification of hyper-heuristics approaches. In: Gendreau, M., Potvin, J.-Y. (eds.) Handbook of Metaheuristics, 2nd ed. International Series in Operations Research & Management Science, vol. 57, chapter 15, pp. 449–468. Springer (2010)
5. Burke, E.K., Kendall, G., Soubeiga, E.: A tabu-search hyperheuristic for timetabling and rostering. Journal of Heuristics **9**, 451–470 (2003)
6. Coelho, I., Coelho, B., Coelho, V., Haddad, M., Souza, M., Ochi, L.: A general variable neighborhood search approach for the resolution of the eternity ii puzzle. In: Proceedings of the 3rd International Conference on Metaheuristics and Nature Inspired Computing, META'10 (2010)
7. Demaine, E.D., Demaine, M.L.: Jigsaw puzzles, edge matching, and polyomino packing: connections and complexity. Graphs Comb. **23**(1), 195–208 (2007)
8. Dueck, G.: New optimisation heuristics. the great deluge algorithm and record-to-record travel. J. Comput. Phys. **104**, 86–92 (1993)
9. Glover, F., Laguna, M.: Tabu Search. Kluwer, Norwell, MA (1997)
10. Heule, M.J.H.: Solving edge-matching problems with satisfiability solvers. In: Proceedings of the Second International Workshop on Logic and Search (LaSh 2008) (2008)
11. Kirkpatrick, S., Gelatt, C., Vecchi, M.: Optimization by simulated annealing. Science, New Series **220**(4598), 671–680 (1983)
12. Misir, M., Wauters, T., Verbeeck, K., Vanden Berghe, G.: A new learning hyper-heuristic for the traveling tournament problem. In: Proceedings Of The VIII Metaheuristics International Conference (2009)
13. Muñoz, J., Gutierrez, G., Sanchis, A.: Evolutionary genetic algorithms in a constraint satisfaction problem: Puzzle eternity ii. In: Cabestany, J., Sandoval, F., Prieto, A., Corchado, J. (eds.) Bio-Inspired Systems: Computational and Ambient Intelligence. Lecture Notes in Computer Science, vol. 5517, pp. 720–727. Springer, Berlin, Heidelberg (2009)
14. Ozcan, E., Misir, M., Ochoa, G., Burke, E.K.: A reinforcement learning—great-deluge hyper-heuristic for examination timetabling. IJAMC **1**, 39–59 (2010)
15. Schaus, P., Deville, Y.: Hybridization of cp and vlns for eternity ii. In: JFPC'08 Quatrième Journées Francophones de Programmation par Contraintes (2008)
16. Talbi, E.-G.: Metaheuristics: From Design to Implementation. Wiley (2009)
17. Vancroonenburg, W., Wauters, T., Vanden Berghe, G.: A two phase hyper-heuristic approach for solving the eternity ii puzzle. In: Proceedings of the 3rd International Conference on Metaheuristics and Nature Inspired Computing, META'10 (2010)
18. Wang, W.-S., Chiang, T.-C.: Solving eternity-ii puzzles with a tabu search algorithm. In: Proceedings of the 3rd International Conference on Metaheuristics and Nature Inspired Computing, META'10 (2010)