Solving Eternity-II puzzles with a tabu search algorithm

Wei-Sin Wang and Tsung-Che Chiang

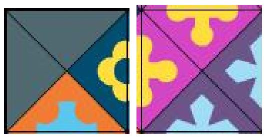
Department of Computer Science and Information Engineering, National Taiwan Normal University, Taiwan, R.O.C.
gnudamgp03@hotmail.com; *tcchiang@ieee.org*

Abstract : In this paper we propose a two-phase approach to solve the Eternity-II puzzles mainly based on the tabu search algorithm. The first phase solves the outside region of the puzzle, and then based on the result obtained in the first phase the entire puzzle is solved in the second phase. The neighborhood functions are based on swap and rotation. Random perturbation and simulated annealing are included to escape from the local optima. The approach is ranked as one of the top three contestants in the Eternity-II contest in the 2010 International Conference on Metaheuristics and Nature Inspired Computing (META 2010).

Keywords : Eternity-II, tabu search, combinatorial optimization

1 Introduction

This research was motivated by the Eternity-II Contest held in International Conference on Metaheuristics and Nature Inspired Computing (META 2010). Eternity-II is a puzzle game. An $n \times n$ puzzle consists of $n \times n$ square pieces that are bordered by colored patterns. The goal is to arrange these pieces so that gray edges are around the outside and colors of adjoining pairs of edges are matched. Figure 1 shows a completed 4×4 puzzle.


(a) Pieces in a puzzle

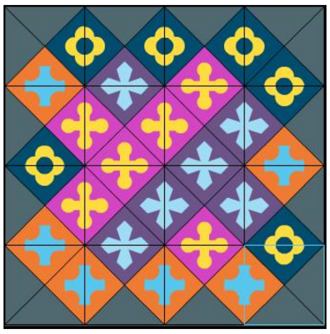

(b) A completed 4×4 Eternity-II puzzle

Figure 1. Eternity-II puzzle [1]

2 Proposed Tabu Search

2.1 Algorithm flow

In this work we propose an approach mainly based on tabu search [2][3] to solve the Eternity-II puzzle. Due to the large search/solution space, we adopt a divide-and-conquer technique. Our algorithm consists of two phases. Phase I focuses on completing the pieces around the outside, namely the pieces with gray color along at least one edge. After the outside pieces are completed or a maximum number of iterations I_1 is reached, all pieces are considered together in phase II. The outside pieces and inside pieces are put randomly in the initialization steps in phase I and II, respectively. Then, a tabu search algorithm tries to adjust their positions to maximize the number of matched pairs of edges. In our observation, the inside part is much harder to complete, and the tabu search algorithm gets trapped in a local optimum easily. We try to help the tabu search algorithm to escape from the local optimum by a simulated annealing (SA) [4] step and a random permutation step in phase II. The flow chart of the entire approach is depicted in Figure 2. Details of each step of our approach are described in the following subsections.

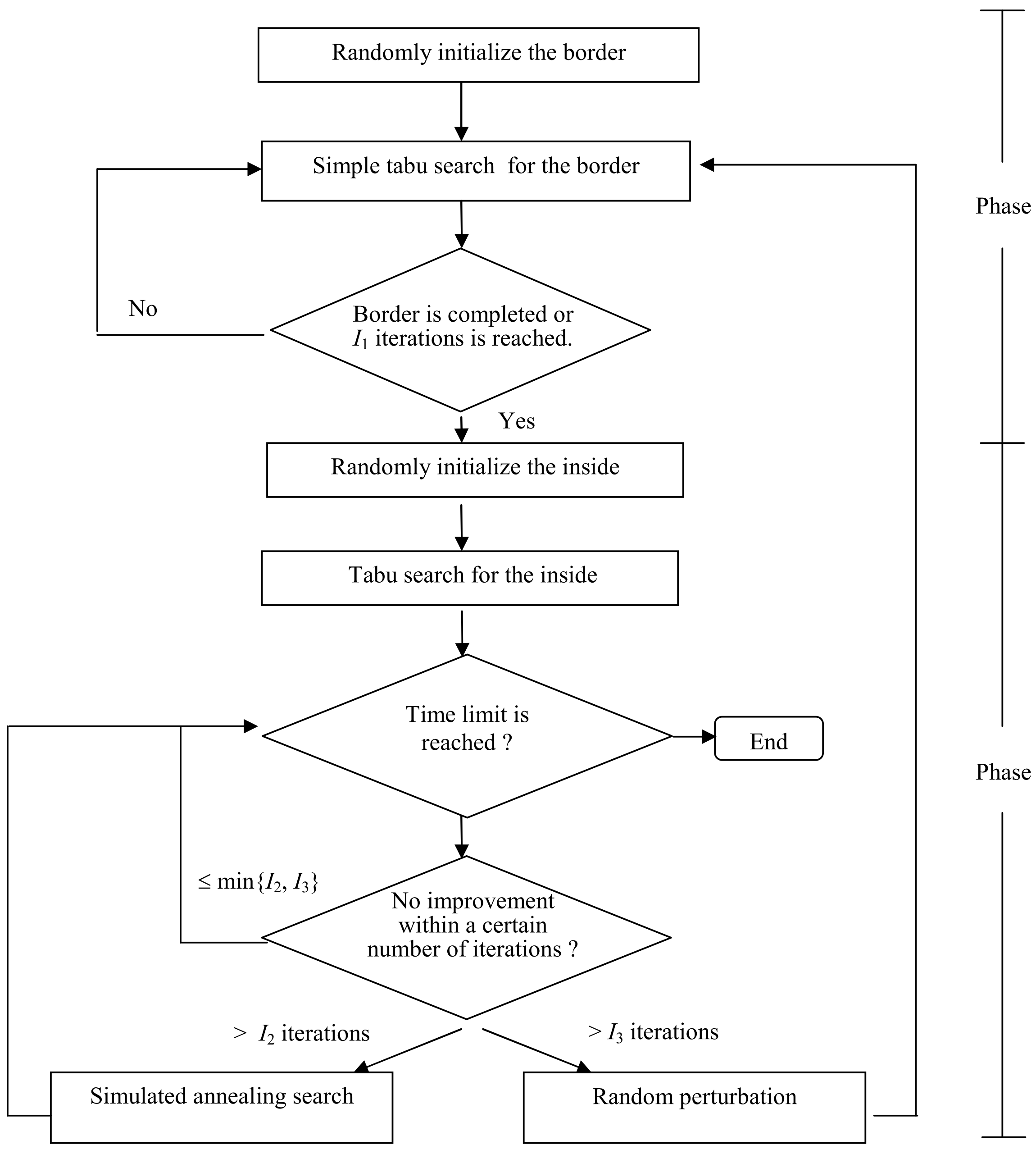

Randomly initialize the border
Simple tabu search for the border
No
Border is completed or I_1 iterations is reached.
Yes
Randomly initialize the inside
Tabu search for the inside
Time limit is reached ?
End
$\leq \min\{I_2, I_3\}$
No improvement within a certain number of iterations ?
$> I_2$ iterations
$> I_3$ iterations
Simulated annealing search
Random perturbation
Phase I
Phase II

Figure 2. Flow chart of the proposed approach

2.2 Solution representation and evaluation

A direct encoding scheme is adopted. The solution is represented by a 2-dimensional array of pieces, where each piece is represented by a four-tuple (color indices of four edges). The fitness or score of a solution is the number of matched pairs of edges.

2.3 Tabu search

2.3.1 Neighbourhood

Generation of the neighbourhood of a solution in phase II is achieved by two types of operations: (1) rotating all inside pieces and (2) randomly swapping two *homogeneous* pieces and rotating them (if possible). Given an $n \times n$ puzzle, the first operation generates $3 \cdot (n-2)^2$ neighbouring solutions since there are $(n-2)^2$ inside pieces and each piece can be rotated for three times. In the second operation, two pieces are homogenous if they have the same number of gray edges. In an $n \times n$ puzzle, there are N_{PHP} = $C_2^4 + C_2^{4(n-2)} + C_2^{(n-2)^2}$ pairs of homogenous pieces. Since N_{PHP} could be very large, only $\max\{\beta_1 \cdot N_{\mathrm{PHP}}, \beta_2\}$ pairs are selected for generating neighbouring solutions. The values of β_1 and β_2 will be given in Section 3.1. Due to the restriction incurred by the gray edges, the outside pieces have fixed orientations. Hence, only randomly swapping two homogeneous pieces is considered when generating neighbourhood in phase I.

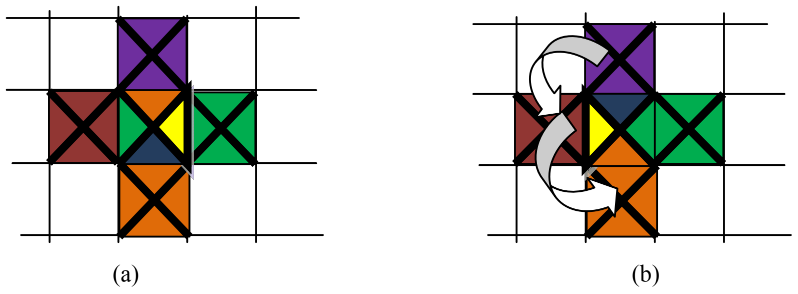


Figure 3. (a) Original solution (b) Neighbouring solution generated by the rotating operation

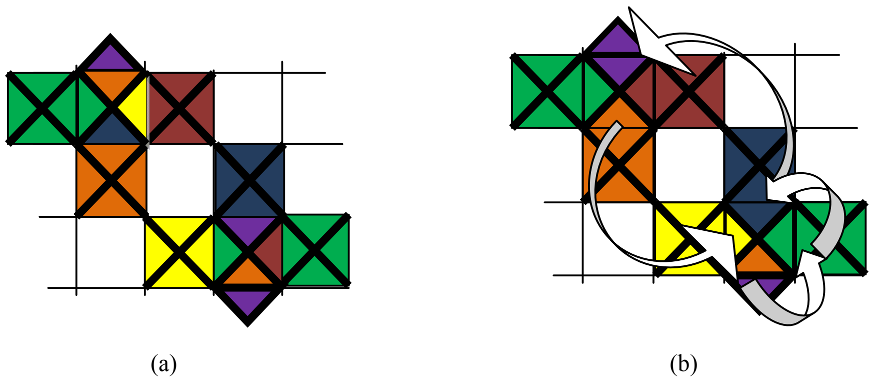


Figure 4. (a) Original solution (b) Neighbouring solution generated by the swapping-then-rotating operation

Once a neighbouring solution is generated, we calculate the local changes of number of matched edges caused by the rotating and/or swapping operations. In this way, we do not need to re-calculate the number of matched edges for the whole puzzle and thus save computation time. Taking Figure 3 as an example, we generate a neighbouring solution in 3(b) by rotating the middle piece in 3(a). We just need to count the number of matched pairs of edges of the rotated piece before and after the rotating operation. In Figure 3 the numbers are 0 and 2, respectively. Thus, the score of the solution in 3(b) can be calculated by that of the solution in 3(a) plus 2 (2–0). Similarly, calculation of the score for a new solution generated by the swapping-then-rotating operation only requires the number of matched pairs of edges of the swapped pieces before and after the swapping-then-rotating operation. In Figure 4 the numbers are 1 (1+0) and 7 (4+3), respectively. The score of the new solution is that of the original solution plus 6 (7–1).

2.3.2 Tabu list

The tabu list defines the neighbouring solutions that are forbidden to be accepted (unless the aspiration criterion, which is described in the next subsection, is satisfied). Since we have two types of operations for generating neighbourhood, there are two types of tabu elements. The first type of tabu element records a position on the puzzle board and the configuration of a piece. The second type of element records a pair of positions on the puzzle board and a pair of configurations of pieces. Here we record the configurations of pieces instead of indices of pieces since there might be multiple pieces with the same configuration. The length of the tabu list will be given in Section 3.1.

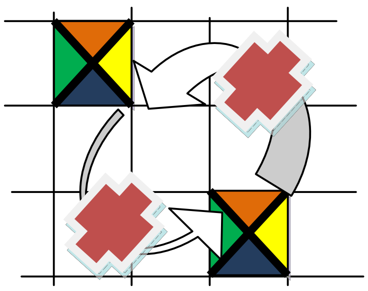


Figure 5. Tabu of a swapping operation

2.3.3 *Aspiration criterion*

If the neighbouring solution has a higher score than that of the best known solution, then the neighbouring solution is acceptable even though it is a tabu solution. Note that the score of the best known solution is reset after the random perturbation step.

2.4 Simulated annealing

2.4.1 *Neighbourhood*

The neighbourhood function of the SA is the same as that of the tabu search.

2.4.2 *Acceptance criterion*

A neighbouring solution S' of an original solution S is accepted in probability

$$Prob(f(S'), f(S), T_i) = \begin{cases} 1 & , if\ f(S') > f(S) \\ e^{\frac{f(S')-f(S)}{T_i}} & , \quad otherwise \end{cases} , \qquad (1)$$

where $f(.)$ is the score and T_i is the temperature of current iteration i.

2.4.3 *Cooling schedule*

The temperature declines by $T_{i+1} = \alpha \cdot T_i$. The values of the initial temperature T_0, final temperature T_E, and α will be given in Section 3.1.

2.5 Perturbation

In addition to the SA step, the perturbation step provides another way to escape from the local optimum by a more random change of the current solution. It selects $n \cdot n \cdot \gamma$ pairs of homogeneous pieces randomly and swap them, where n is the number of rows/columns of the puzzle and γ is a parameter. When n gets large, the number of outside pieces, $4(n-1)$, gets much smaller than the number of inside pieces, $(n-2)^2$. If we just select pairs of homogeneous pieces randomly, outside pieces will be selected in a much lower probability than inside pieces. However, we think that outside pieces could have more restrictions on the solutions to be generated during the search process than inside pieces do. Thus, we control the selection so that pairs of inside and outside pieces are selected with equal probability.

2.6 Termination criterion

The proposed approach terminates if the puzzle is completed or the predefined time limit is reached.

3 Experiments and Results

3.1 Parameter setting

Table 1. Parameter setting in the experiment

Parameter	Value
Length of tabu list	360
Maximum number of iterations of Phase I (I_1)	400
Number of non-improving iterations for simulated annealing (I_2)	400
Number of non-improving iterations for perturbation (I_3)	3500
Ratio of pairs of homogeneous pieces in tabu search (β_1)	0.5
Maximum number of pairs of homogeneous pieces in tabu search (β_2)	6000
Scaling value in the cooling schedule (α)	0.99
Initial temperature in simulated annealing (T_0)	100
Final temperature (T_E)	0
Ratio of pairs of homogeneous pieces in perturbation (γ)	0.5

To make the experimental results reproducible, we set fixed seeds for the random number generator. The seed in run i was set by $10 \cdot i$. There are 30 runs in total.

3.2 Computation environment

The proposed approach was implemented in C++ using Microsoft Visual Studio 2008®. The experiments were conducted using a personal computer with 1.8GHz CPU and 1GB RAM.

3.3 Performance

We tested the proposed approach on the four benchmark instances, 10×10, 12×12, 14×14, and 16×16 puzzles. The computation time limits for them are 1200, 1800, 2400, and 3600 seconds, respectively. The best over all, mean, median, standard deviation of scores, and average number of evaluations over 30 runs are summarized in Table 2. The optimal score of each puzzle is also provided for reference.

Table 2. Summary of experimental results

	Best over all	Mean	Median	Std. deviation	Optimal	Avg. no. of evaluations
10 × 10	163	159.56667	159	1.8740924	180	95,790,842.3
12 × 12	234	232.3333	233	2.385139	264	207,496,197.8
14 × 14	318	312.5667	313	2.679345	364	357,203,966.2
16 × 16	418	408.6333	409	4.757334	480	478,196,791.4

4 Conclusions

In this study, we proposed a metaheuristic algorithm based on tabu search and simulated annealing to solve the Eternity-II puzzle. The proposed approach was tested on four puzzle instances with difference scales. The score of the best solution found by our approach reaches about 87%~90% of that of the optimal solution. There is indeed a large space for further improvement.

We used simple rotating and swapping neighbourhood functions in our approach and observed that the search process got stuck in the local optima easily. It indicates that the target problem has a large number of local optima. Some suggestions for improvement include: (1) We can use domain-specific neighbourhood function to locate the local optima more effectively; (2) We can also apply divide-and-conquer technique plus exact algorithms like branch-and-bound or backtracking search to find the local optima; (3) We can incorporate the concept of population-based metaheuristics like GA [5] and the concept of variable neighbourhood search (VNS) [6] to escape from the local optima.

References

[1] http://us.eternityii.com/try-eternity2-online/

[2] F. Glover, “Tabu Search - Part I,” *ORSA Journal on Computing*, vol. 1, no. 3, pp. 190-206, 1989.

[3] F. Glover, “Tabu Search - Part II,” *ORSA Journal on Computing*, vol 2, no. 1, pp. 4-32, 1990.

[4] S Kirkpatrick, CD Gelatt Jr, MP Vecchi, “Optimization by simulated annealing,” *Science*, vol. 220, pp. 671-680, 1983.

[5] D. E. Goldberg, *Genetic algorithms in search, optimization, and machine learning*, Addison-Wesley, 1989.

[6] N. Mladenovic and P. Hansen, “Variable neighbourhood search,” *Computers & Operations Search*, vol. 24, no. 11, pp. 1098-1100, 1997.