A General Variable Neighborhood Search approach for the resolution of the Eternity II Puzzle

I. M. Coelho[1], B. N. Coelho[2], V. N. Coelho[2], M. N. Haddad[2], M. J. F. Souza[2], and L. S. Ochi[1]

[1] Fluminense Federal University, Niterói, RJ, 24210-240, Brazil
[2] Federal University of Ouro Preto, Ouro Preto, MG, 35400-000, Brazil

imcoelho@ic.uff.br, brunonazario@yahoo.com.br, vncoelho@gmail.com, mathaddad@gmail.com, marcone@iceb.ufop.br, satoru@ic.uff.br

1 Introduction

This work presents a heuristic approach based on the General Variable Neighborhood Search for the Eternity II Puzzle.

A solution is represented by a matrix of pieces, where each piece is represented by: index, number of rotations (0..3), and four numbers representing the colors related to each direction (left, up, right, down).

To explore the solution space of the problem, five movements were developed. Each movement defines a neighborhood $N(.)$, which are presented next.

The Movement Swap Corner ($N^{SC}(s)$) swaps two pieces that must be in the corners of the puzzle. After the swap, it rotates both pieces, separately, in order to put them in their right positions.

Similarly to the Movement Swap Corner, the Movement Swap Side ($N^{SS}(s)$) consists of swapping two pieces that must be in the sides of the puzzle. After that, it rotates both pieces, separately, adjusting their right positions.

The Movement Rotate ($N^{R}(s)$) is characterized by, simply, rotating one piece.

The Movement Swap Center ($N^{SWC}(s)$) swaps two pieces that are located out of the border of the puzzle, in other words, the center of the puzzle.

The idea of the Movement Swap Rotate Center ($N^{SRC}(s)$) is to swap two pieces which are located in the center of the puzzle. Then, it rotates both pieces, separately, leaving them in their best positions. The best position of one piece is the position where the piece matches with more colors, in relation to nearby pieces.

A solution is evaluated by a mono-objective function f, to be maximized, that computes the score, that is, the total number of correctly-matched pairs of edges in a solution. All moves respect the borders of the puzzle, so the method only generates valid solutions.

Two generators for initial solutions were proposed. The first one generates a valid puzzle with random pieces. The second is a greedy-randomized strategy, in which, at each step, one of the best scored pieces is inserted in the game. This algorithm starts with random corners and tries to build good borders by choosing the best side pieces (in case of a tie, it chooses randomly). For the center pieces, at each step, every candidate piece is rotated 4 times in order to define the best insertion in the puzzle.

2 Proposed Algorithm

The proposed algorithm, called `MS-GVNS`, combines ideas from `MultiStart` [1] and *General Variable Neighborhood Search* - `GVNS` [2] procedures. Similar strategies have been successfully applied to other optimization problems by the authors of this work (see [3]). Algorithm 1 outlines the steps.

Algorithm 1: MS-GVNS

Input: *LevelMax*, *IterMax*
Output: Solution s

```
1  while stop criterion not satisfied do
2      s_0 ← BuildGreedyRandomizedSolution()
3      s ← VND(s_0)
4      p ← 0
5      while p < LevelMax and stop criterion not satisfied do
6          iter ← 0
7          while iter < IterMax and stop criterion not satisfied do
8              s' ← s
9              for i = 1 to p + 2 do
10                 k ← SelectNeighborhood()
11                 s' ← Shake(s', k)
12             end
13             s'' ← VND(s')
14             if f(s'') < f(s) then
15                 s ← s''
16                 p ← 0
17                 iter ← 0
18             end
19             iter ← iter + 1
20         end
21         p ← p + 1
22     end
23 end
24 return s
```

Building an initial solution s_0 (line 2 of Algorithm 1) is made by the greedy randomized procedure described previously. The local search (lines 3 and 13 of Algorithm 1), in turn, uses the VND procedure [2], with the movements described previously.

Whenever a given number of iterations without improvement is reached, the MS-GGVNS algorithm applies $p + 2$ times the *Shake* procedure, using a previously selected neighborhood. The procedure *SelectNeighborhood* (line 10 of the Algorithm 1) works as follows. We randomly select a neighborhood k from the list $\{N^{SC}, N^{SS}, N^{R}, N^{SWC}\}$. Each $Shake(s', k)$ call (line 11 of Algorithm 1) performs a random movement from neighborhood k of the shaken solution s'. After *IterMax* iterations without improvement, we increment p in order to generate solutions which become increasingly distant from the current location in the search space.

The local search applied on the solution returned by the *Shake* procedure is based on the VND procedure (line 13 of Algorithm 1). If VND finds a better solution, the variable p returns to the lowest value, that is, $p = 0$.

Only a group of neighborhoods was used in the local search and two different strategies were developed. The **Center-to-Border** strategy consists in exploring the neighborhoods in the following order: N^{SRC}, N^{R}, N^{SS} and N^{SC}. On the other hand, the **Border-to-Center** strategy consists in exploring the neighborhoods in the following order: N^{SC}, N^{SS}, N^{R} and N^{SRC}. Thus, the VND used both strategies **Center-to-Border** and **Border-to-Center**, which can be different at each VND call.

3 Computational Experiments and Conclusions

The proposed algorithm was coded in C++ programming language with the computational framework OptFrame (available at http://sourceforge.net/projects/optframe/ under LGPLv3 license) and compiled with the GNU Compiler Collection version 4.0. The algorithm was tested in a PC Pentium Core 2 Quad (Q6600), 2.4 GHz, with 8 GB of RAM, running Linux Ubuntu 10.04, kernel 2.6.32-24.

As mentioned in [2], one important decision to build an efficient VND procedure is to select an application order of the different neighborhoods. In a preliminary set of experiments (10 runs of 10

minutes for each instance) we tried to discover the optimal sequence of neighborhood application, that is, the one which, in average, produces better solutions in a limited amount of time when running the `MS-GVNS` algorithm. To accomplish this objective, we compared 3 different strategies: **Border-to-Center**, **Center-to-Border** and **Random**. The experiments showed that the best strategies are **Border-to-Center** and **Center-to-Border** and this motivated us to use both strategies for the final tests.

The results in Table 1 were produced using this new strategy. The proposed `MS-GVNS` algorithm was executed with the following parameters: $IterMax = 200$, $LevelMax = N$, where N is the maximum between height and weight of the problem instance.

Results of 30 executions of the problems 10x10, 12x12, 14x14 and 16x16 (with time limits of 1200, 1800, 2400 and 3600 seconds, respectively) appear in Table 1. In this table, column "Best" refers to the best solution cost found in all our experiments. In column "Optimum" we indicate the optimal value for each problem instance. Columns "Average", "Median", "Standard_Deviation", "N_Eval" represent average, median, standard deviation and average number of evaluations, respectively.

Table 1. Experimental results: MS-GVNS algorithm

		MS-GVNS				
Problem	Optimum	Average	Median	Standard_Deviation	N_Eval	Best
10x10	180	165.67	166	0.7581	2362375563	167
12x12	264	238.93	239	1.3628	3408029901	241
14x14	364	320.77	321	2.2234	4438058316	325
16x16	480	419.57	419.5	2.6219	6695171793	425

As can be seen in Table 1, although it was not possible to reach optimal solutions, the algorithm was capable of finding good solutions within the given time limit. The standard deviation was also small, indicating that the proposed method is robust.

Acknowledgements

The authors acknowledge UFOP, FAPEMIG, CAPES and CNPq for supporting the development of this research.

References

1. Martí, R.: Multi-Start Methods, In: Handbook of Metaheuristics, Glover, F. and Kochenberger, G. (Eds), Springer New York, 355–368, 2003.
2. Hansen, P., Mladenovic, N., and Pérez, J.A.M.: Variable neighborhood search: methods and applications, 4OR: Quarterly journal of the Belgian, French and Italian operations research societies, v. 6, 319–360, 2008.
3. Souza, M.J.F., Coelho, I.M., Ribas, S., Santos, H.G., and Merschmann, L.H.C.: A hybrid heuristic algorithm for the open-pit-mining operational planning problem, European Journal of Operational Research, v. 207, 1041–1051, 2010.