Genetic Algorithm and Wisdom of Artificial Crowds Algorithm Applied to Light Up

Leif H. Ashby, *Member, IEEE,* Roman V. Yampolskiy, IEEE member

***Abstract*—We describe an experimental approach to solving instances of an NP-Complete game, Light up, using a genetic algorithm. Subsequently we apply a new postprocessing algorithm known as the Wisdom of Artificial Crowds (WoAC) to improve the quality of achieved solutions. Experimental results are reported and directions for future research are suggested.**

***Index Terms*— Genetic Algorithm, Light Up, Wisdom of Artificial Crowds.**

I. Introduction

A novel approach is proposed to solve NP-Complete problems using a combination of both a Genetic Algorithm and the concept of the Wisdom of Crowds. An NP-Complete problem is one that, given the current state of knowledge in computer science cannot be solved in polynomial time. Thus large instances of NP-Complete problems cannot be precisely solved in reasonable time [5]. We develop an approximation approach which first utilizes a Genetic Algorithm (GA) approach and then performs additional processing via a novel Wisdom of Artificial Crowds (WoAC) algorithm to improve the solution.

II. The Game of Light Up

For this experiment a game known as "Light Up" was the chosen problem. Light Up is a Japanese pen and paper puzzle game. The decision version of Light Up: "Given a Light Up puzzle, does a solution exists?" is known to be NP-Complete [10]. The game board is a simple *NxN* board, with either white squares or black squares. The player moves by placing light bulbs in the white spaces. A light bulb will illuminate its row and column. The black squares block light, and may or may not have a number between 0 and 4, requiring that many light bulbs to touch its four sides. Additionally, light bulbs cannot shine on each other, or rather; no two light bulbs can exist on the same row, or column, unless a black square is between them at some point. The object of the game is to illuminate all the white squares, while satisfying all of the block constraints, without bulbs shining on each other. Light up is a simple to understand and play puzzle, but complex to solve when the board grows larger, which is a trait of NP-Complete problems [17]. Typically a human will only play a 25x25 board at the largest and even this proves challenging for the genetic algorithm approach.

III. Genetic Algorithm

A general Genetic Algorithm is based on taking a difficult problem and encoding a set of potential randomly generated solutions into a chromosome, or an array of values relevant to the problem which represent a possible solution, or state [6]. Then mimicking genetics in nature, chromosomes are combined, or crossed over, to create composite results, which share traits of both 'parents' [7]. A population of these results is maintained, and parents are chosen and combined from the population to form the new population of the next generation. Also like in nature, mutation is applied, typically randomly, to provide diversity, and thus prevent the population from converging too quickly [8].

IV. Wisdom of Artificial Crowds

The Wisdom of Crowds is the concept that states that within a group of people, an aggregation or combination of opinions will result in better answers overall than any individual opinion [1, 3]. Wisdom of Artificial Crowds (WoAC) is a novel swarm-based nature-inspired metaheuristic algorithm for global optimization. WoAC is a post-processing algorithm in which independently-deciding artificial agents aggregate their individual solutions to arrive at an answer which is superior to all solutions present in the population. The algorithm is inspired by the natural phenomenon known as the Wisdom of Crowds [18]. WoAC is designed to serve as a post-processing step for any swarm-based optimization algorithm in which a population of intermediate solutions is produced, for example in this paper we will illustrate how WoAC can be applied to a standard Genetic Algorithm (GA).

The population of intermediate solutions to a problem is treated as a crowd of intelligent agents. For a specific problem an aggregation method is developed which allows individual solutions present in the population to be combined to produce a superior solution. The approach is somewhat related to ensemble learning [15] methods such as boosting or bootstrap aggregation [12, 11] in the context of classifier fusion in which decisions of independent classifiers are combined to produce a superior meta-algorithm. The main difference is that in ensembles "when combing multiple independent and diverse decisions each of which is at least more accurate than random guessing, random errors cancel each other out, correct decisions are reinforced" [13], but in WoAC individual agents are not required to be more accurate than random guessing.

978-1- 4577-1452-8/11/$26.00 ©2011 IEEE

V. Approach

The game Light up appeared to be ideal for solving with a genetic algorithm, however before this could be done, a reliable generator of light up puzzles had to be found. Online sources proved unfruitful, so a method of generating *NxN* puzzles was investigated. Clearly it's simple to create unsolvable puzzles, and this might not matter for the purpose of this experiment. Since goodness of result is to be maximized, an unsolvable puzzle can still have an almost perfect solution, therefore mostly random puzzles can be used, providing enough difficulty to produce less than optimal results from a GA, in order to see the WoAC approach prove effective. Obviously, just by chance perfectly solvable instances of puzzles will also be produced.

The approach to this was straightforward: an *NxN* array was created, and populated at random. 75% of blocks would be white, or empty. 15% are black blocks without constraint, and the remaining 10% would be a randomly selected constrained block, between 0 and 4. These percentages could be tweaked to produce different difficulties of puzzles, in terms of the number of possible solutions, but for the GA tests these values were used.

If only perfectly solvable solutions were necessary, an approach could be taken to identify specific traits in a puzzle that makes it so, and eliminate them, either by changing a cell's value, or removing a cell. This was not pursued however, due to the tangential nature of the problem [16].

The first step in designing the genetic algorithm was to determine the best way to formulate the fitness function. Certain states are defined as illegal, such as two light bulbs shining on each other, as well as having more or less than the block's required number of adjacent light bulbs. Additionally all white squares must be illuminated. So the fitness function must incorporate all of these factors in determining how close a given chromosome is to a solution.

Chromosomal representation of the solution is in the form of a 1-dimensional array, with the size *NxN*, each row wrapping around left to right in order from top down. Similar to other puzzles being solved with a genetic algorithm, it seems undesirable to introduce too many changes to children within any given generation [19], since it's difficult to isolate only positive changes between generations. However, because the board state doesn't rely on ordering, but is rather represented by on/off states, pivot crossovers can be effectively used without introducing any new states. Both single-point and two-point crossovers were implemented by simply exchanging bulb states from parents to produce offspring.

For mutation, toggling bulbs on and off was focused on, allowing a mutation to flip a single bulb, or less frequently toggling multiple random bulbs on or off. Occasionally it will also shift a bulb forwards or backwards one index in the solution array. Mutation rates ranging from 40% to 100% were used, meaning that with a 100% mutation rate, after every crossover both resulting children would be mutated in some way. 10% of the time, a chromosome would have multiple bulbs randomly toggle with between 0% and 2% of the total bulbs affected. 50% of the time a single bulb would be toggled, and the other 40% of the time a bulb would be shifted left or right in the array.

The population size remained constant throughout the experiments, at 500 individuals, with the mating algorithm selecting the 75 best solutions (according to the fitness function), plus an additional 25 random solutions which were then crossed over at random until another population of 500 was reached. A given genetic algorithm run would continue to cross over and mutate until improvement was seen to level off, and at least 50 generations passed in which improvement of no more than 0.05% was seen. This typically resulted in runs which have converged to a solution, and did not run excessively long (see Figures 1-4, and Tables 1-4).

To produce a non-homogeneous population to run the aggregation on, results were collected from discrete genetic algorithm runs. Once enough results from the GA are produced, solutions with the lowest number of violations are combined to, ideally, produce a full solution, or at least one with a fewer number of violations than the previous best [2, 14]. In experimentation, between 75 and 100 genetic algorithm solutions were collected to apply Wisdom of Artificial Crowds on.

In order to combine the best solutions generated, every resulting solutions bulb locations were iterated through, and a dictionary was formed with the index of the bulb location (in the solution array) as the key, and the count as the value. So for every solution that had a given index with a bulb turned on, the value for that index was incremented in the dictionary. This produced a mapping of solutions such that the most common bulbs could be combined into a final solution. However, due to the large variety of solutions produced, there would be a large number of collisions in terms of bulb violations if the best bulbs were used. Once the dictionary was converted to an ordered list, preferring popular bulbs, they were added to a result array such that a bulb would only be added if it did not create a bulb violation with a previously placed bulb. Therefore this always created a solution with no bulb solutions, but it could still contain potentially more space violations, or block violations, than the best individual solution.

VI. Results

A. Data

Data was generated by the developed software. Puzzles' sizes for testing included 10x10, 15x15, 25x25, 40x40. Larger puzzles were deemed too computationally difficult for the time span, and proved hard to visualize.

B. Results

Resulting solutions are presented as the puzzle board, with

a description of the violations (see Figures 7, 8, 10, 12, 14, 15). For the WoAC scatter plots as percentage of agreement were calculated (see Figures 5, 6, 9, 11, 13) by counting the number of bulbs lit in any given solution, and comparing it to the number of times the same bulb was lit in the entire population. The agreement between all bulbs is averaged to find the final number.

VII. Discussion

The results were encouraging with the genetic algorithm producing good results for the puzzle, while rarely giving an optimum solution. It should be noted that most puzzles being generated are not fully solvable, typically with unsatisfiable block requirements (such as a 4 block without 4 empty sides), or multiple blocks that conflict with each other's requirements, thus making it impossible to fully solve a puzzle without a better puzzle generator. That being said, the genetic algorithm and wisdom of artificial crowds approach can still be evaluated.

Upon running the genetic algorithm, it was found that it did an excellent job at avoiding conflicting bulbs, and unlit spaces, but left the solutions less than optimal by neglecting to resolve block dependencies, such that the majority of the violations for a given solution were block violations, where blocks did not have the required number of bulbs on its perimeter. It was determined that, due to the way the fitness function was weighting violations, these specific problems with the solution were being neglected.

A single bulb violation produced two actual violations, one for each bulb. A white space violation is just one violation, but if a mutation or crossover produced a solution where a bulb was misplaced, it would often create multiple white space violations, whereas a block violation is a single violation for each missing bulb. So if a mutation were to solve one of these violations, it might be overshadowed by other violations. To test this hypothesis, block violations were weighted heavier, and an improvement was seen with regards to block violations, although bulb violations more than offset it, typically.

The crossover performance was mostly the same for the puzzles tested, which should be expected, since both the single point and the two-point crossovers are very similar, although some advantage should be seen in the two-point operator [4]. Mutation rates did not seem to affect results much over 40%, with higher mutation rates giving similar results.

Interestingly, for the 40x40 board, the resulting solution has a low percentage of agreement with the general population WoAC is aggregating. This is likely due to the way the board was laid out, and the large amount of deviation between other members of the population, such that recombining the bulb locations in a way that eliminates bulb violations produced a solution that did not agree with many other solutions.

To revisit the last scatter plot shown in the Figure 13, a population was first built using a ranking of block violations, as previously referred to, in order to produce results that had very few block violations, but a high number of bulb violations. The WoAC would then eliminate bulb violations through the aggregation process. It was hoped this would provide a superior solution, but in fact produced almost the same result as the original standard weighting. This indicates that, for this puzzle at least, the solution found is close to the best result, at least for the recombination algorithm as it is now. It's likely that with some heuristics to improve aggregation a better result could be found, as well as how the aggregation is performed with regards to weighting the placement against different fitness factors [9].

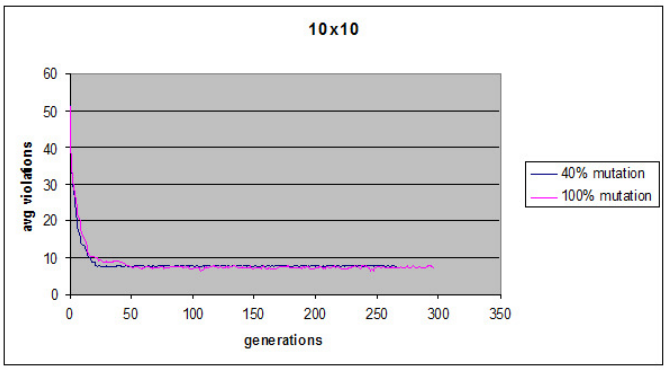

Figure 1. The improvement chart for the 10x10 puzzle.

Table 1: The mutation rate seems to make minor difference, although the average results improved for the 100% mutation rate.

Dimension	Mutation Rate	Avg Violations	Avg Time Taken	Avg Generations
10x10	75%	3 +- 1	9 seconds	320 +- 112
10x10	100%	1.86 += 0.4	10 seconds	375 +- 74

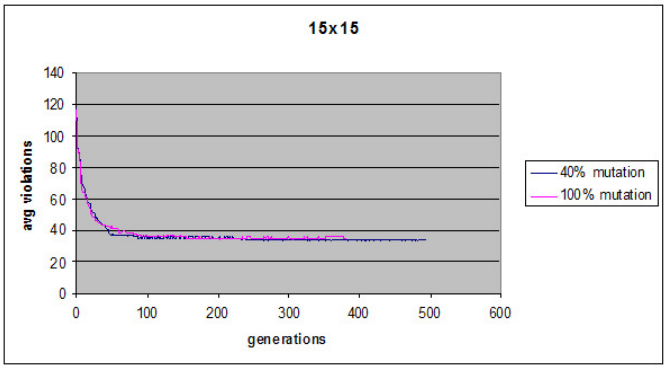

Figure 2. The improvement chart for the 15x15 puzzle.

Table 2: The mutation rate has a marginally positive effect.

Dimension	Mutation Rate	Avg Violations	Avg Time Taken	Avg Generations
15x15	40%	32 +- 3	~20 seconds	491 +- 161
15x15	100%	28 +- 1	~30 seconds	450 +- 125

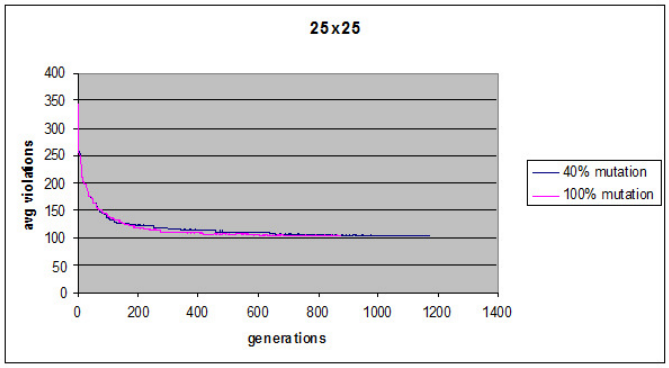

Figure 3. The improvement chart for the 25x25 puzzle.

Table 3: Again, the mutation rate has a marginally positive effect.

Dimension	Mutation Rate	Avg Violations	Avg Time Taken	Avg Generations
25x25	40%	115 +- 9	~2 minutes	652 +- 185
25x25	100%	103 +- 8	~3 minutes	744 +- 328

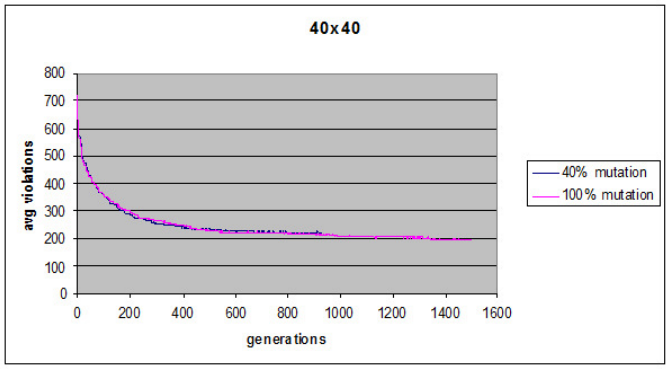

Figure 4. The improvement chart for the 40x40 puzzle.

Table 4: Still, the mutation rate has a marginally positive effect.

Dimension	Mutation Rate	Avg Violations	Avg Time Taken	Avg Generations
40x40	40%	220 +- 9	~6 minutes	889 +- 180
40x40	100%	186 +- 6	~10 minutes	1268 +- 112

For the Wisdom of Artificial Crowds, results improved for more complex puzzles, but WoAC was not intelligent enough to keep from hurting almost correct solutions (see Table 5). This is typical, since when a perfect solution (or near optimal one) is found, it's typically not the majority of the solutions, therefore when aggregation is applied it will not simply duplicate the optimal solution.

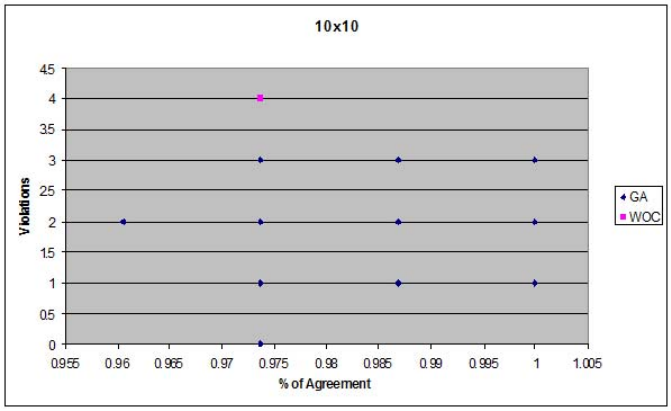

Figure 5. Scatter chart for 10x10 board.

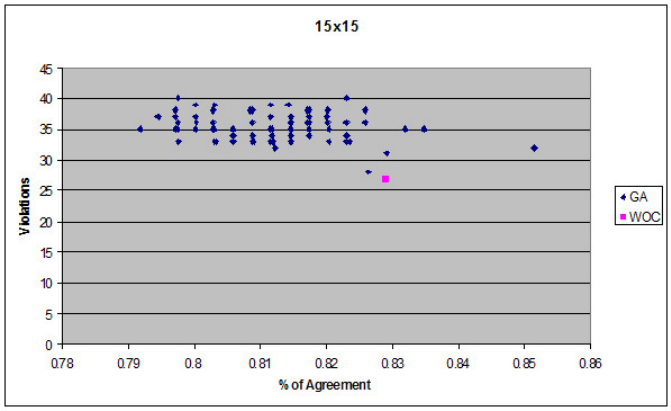

Figure 6. Scatter chart for 15x15 board.

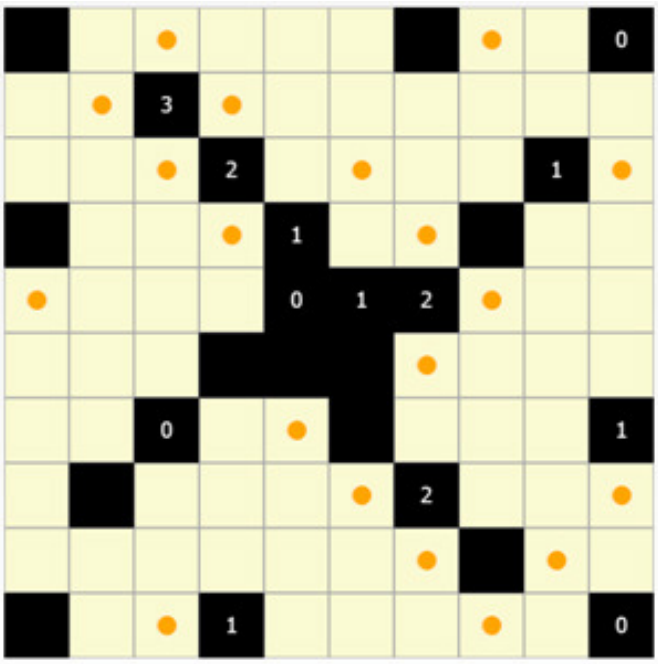

Figure 7. Sample solution to a 10x10 board.

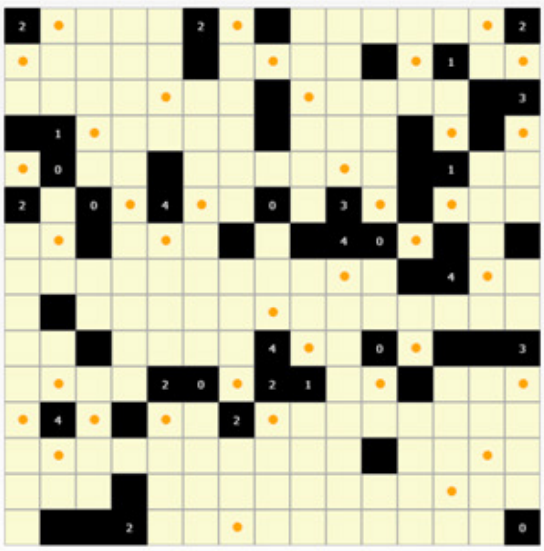

Figure 8. Sample solution to a 15x15 board.

Much better results are seen in larger puzzles, where there may not even be perfect solutions.

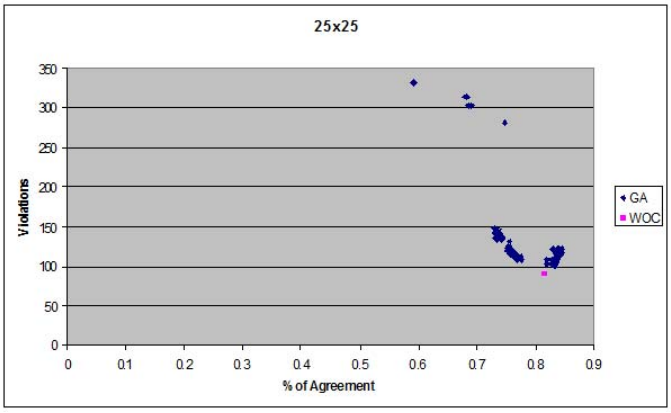

Figure 9. Scatter chart for 25x25 board..

Figure 10. Sample solution to a 25x25 board.

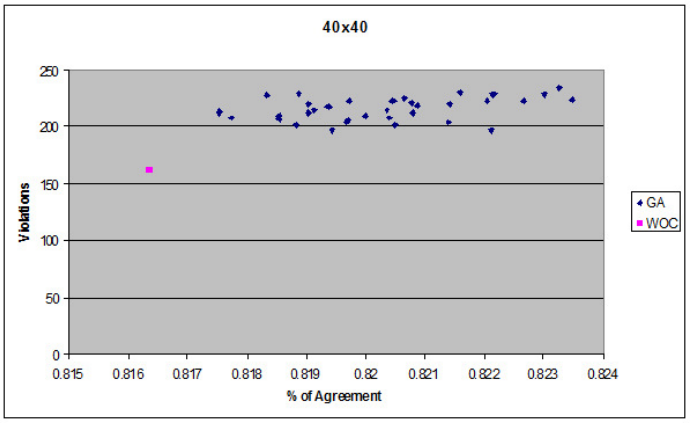

Figure 11. Scatter chart for 40x40 board.

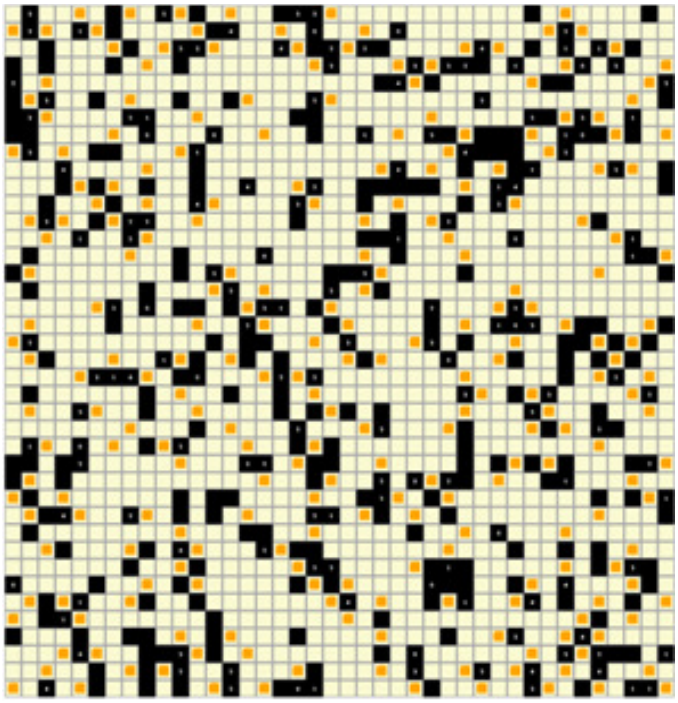

Figure 12. Sample solution to a 40x40 board.

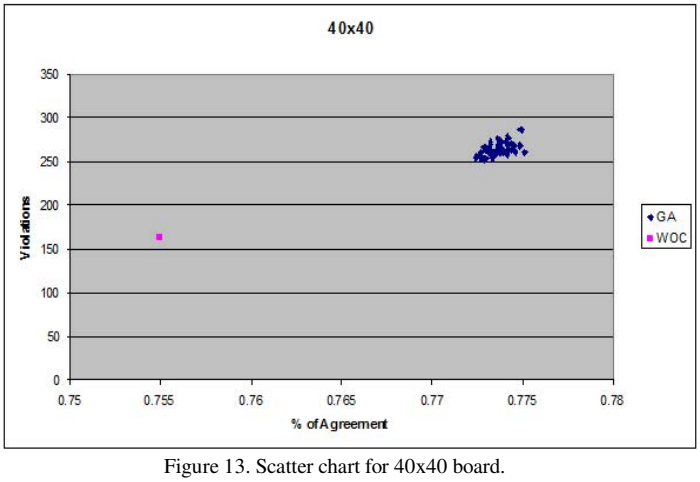

Figure 13. Scatter chart for 40x40 board.

Table 5: Statistics for WoAC - final results.

Dimension	Violations breakdown	Best result before aggregating:	Improvement %
10x10	Block violations: 4 Space violations: 0 Bulb violations: 0	Block violations: 0 Space violations: 0 Bulb violations: 0	0%
15x15	Block violations: 26 Space violations: 1 Bulb violations: 0	Block violations: 26 Space violations: 2 Bulb violations: 0	3.5%
25x25	Block violations: 90 Space violations: 0 Bulb violations: 0	Block violations: 95 Space violations: 1 Bulb violations: 4	10%
40x40	Block violations: 162 Space violations: 0 Bulb violations: 0	Block violations: 187 Space violations: 1 Bulb violations: 9	18%
40x40 (alternate evaluation)	Block violations: 163 Space violations: 0 Bulb violations: 0	Block violations: 37 Space violations: 2 Bulb violations: 213	35%

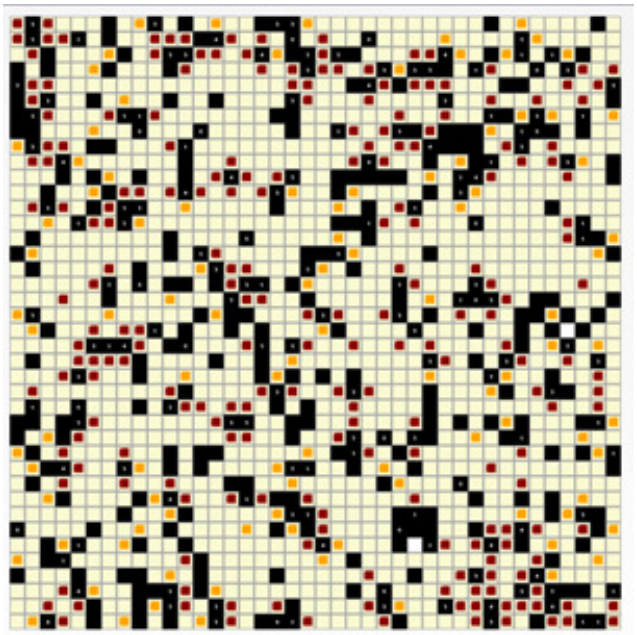

Figure 14. Results before aggregation.

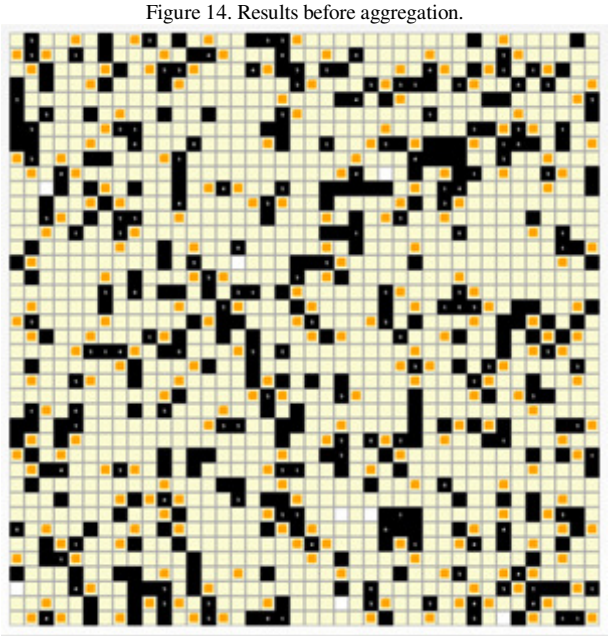

Figure 15. Results after aggregation. This result is from the same 40x40 puzzle as previously seen, but with an alternate evaluation function, such that block violations are more severely counted against a solution's fitness.

REFERENCES

[1] R. Baraglia, J. I. Hidalgo and R. Perego, *A Hybrid Heuristic for the Traveling Salesman Problem*, IEEE Transaction On Evolutionary Computation, 5(6) (December 2001), pp. 613-622.

[2] A. E. Eiben, P.-E. Raué and Z. Ruttkay, *Genetic algorithms with multi-parent recombination*, *The Third Conference on Parallel Problem Solving from Nature*, 1994, pp. 78-87.

[3] B. Fan and B. Krishnamachari, *Exploiting the Wisdom of the Crowd: Localized, Distributed Information-Centric VANETs*, IEEE Communications Magazine, 48(5) (2010), pp. 138-146.

[4] D. B. Fogel, *Evolutionary Computation: Towards a New Philosophy of Machine Intelligence*, IEEE Press, New York, 2000.

[5] L. Fortnow, *The Status of the P versus NP Problem*, Communications of the ACM, 52(9) (2009), pp. 78-86.

[6] D. E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison-Wesley Pub. Co, 1989.

[7] T. D. Gwiazda, *Genetic Algorithms Reference Vol.1 Crossover for single-objective numerical optimization problems*, Lomianki, 2006.

[8] J. Holland, *Adaptation in Natural and Artificial Systems*, University of Michigan Press, Ann Arbor, Michigan, 1975.

[9] J. Howe, *The Wisdom of the Crowd Resides in How the Crowd Is Used*, Nieman Reports, 62(4) (2008), pp. 47-50.

[10] B. McPhail, *Light Up is NP-complete*, Available at: http://www.cs.umass.edu/~mcphailb/papers/2005lightup.pdf, February 28, 2005.

[11] P. Melville and R. J. Mooney, *Constructing Diverse Classifier Ensembles Using Artificial Training Examples*, *18th International Joint Conference on Artificial Intelligence (IJCAI'03)*, Acapulco, Mexico, August 2003, pp. 505-510.

[12] P. Melville and R. J. Mooney, *Diverse Ensembles for Active Learning*, *21st International Conference on Machine Learning (ICML'04)*, Banff, Canada, July 2004, pp. 584-591.

[13] R. J. Mooney, *Machine Learning: Ensembles*, Available at: www.cs.utexas.edu/~mooney/cs391L/slides/ensembles.ppt, Retrieved January 8, 2011.

[14] M. C. Mozer, H. Pashler and H. Homaei, *Optimal Predictions in Everyday Cognition: The Wisdom of Individuals or Crowds?*, Cognitive Science, 32(7) (2008), pp. 1133-1147.

[15] D. Opitz and R. Maclin, *Popular ensemble methods: An empirical study*, Journal of Artificial Intelligence Research, 11 (1999), pp. 169-198.

[16] E. G. Ortiz-Garcıa, S. Salcedo-Sanz, J. M. Leiva-Murillo, A. M. Pe´rez-Bellido and J. A. Portilla-Figueras, *Automated generation and visualization of picture-logic puzzles*, Computers & Graphics, 31(5) (2007), pp. 750-760.

[17] C. H. Papadimitriou, *Computational Complexity*, Addison-Wesley, 1994.

[18] J. Surowiecki, *The Wisdom of Crowds: Why the Many Are Smarter Than the Few and How Collective Wisdom Shapes Business, Economies, Societies and Nations*, Little, Brown, 2004.

[19] D. Whitley, T. Starkweather and D. A. Fuquay, *Scheduling problems and traveling salesman: The genetic edge recombination operator*, *International Conference on Genetic Algorithms*, 1989, pp. 133-140.

Leif H. Ashby Is a senior at the University of Louisville in Computer Engineering and Computer Science department.

Roman V. Yampolskiy (SM'07–M'09) holds a PhD degree from the Department of Computer Science and Engineering at the University at Buffalo. In 2008 Dr. Yampolskiy accepted an assistant professor position at the Speed School of Engineering, University of Louisville, KY.

Dr. Yampolskiy's main areas of interest are behavioral biometrics, digital forensics, pattern recognition, genetic algorithms, neural networks, artificial intelligence and games. Dr. Yampolskiy is an author of over 60 publications including multiple journal articles and books.