A Modified Grid Diversity Operator for Discrete Optimization and its Application to Wind Farm Layout Optimization Problems

Ahmed Salah
Mathematics Department
Faculty of Science, Mansoura University
Mansoura, Egypt
a_salah@mans.edu.eg

Emma Hart
School of Computing
Edinburgh Napier University
Edinburgh, UK
e.hart@napier.ac.uk

ABSTRACT

One of the important factors for any effective metaheuristic optimization algorithms is its ability to maintain a diverse population throughout the generations. Specifically, diversity maintenance can lead to exploration of novel areas of the search space, and hence potentially locate better solutions. Recently, a novel diversity technique called the *Grid Diversity Operator* was introduced, providing improved results on a suite of continuous optimization problems. Here, we extend the grid diversity operator to support discrete optimization problems and validate its performance by experimenting with a range of wind farm layout optimization problem scenarios. The experimental results showed performance improvements for the quality of solutions found especially for the farm scenarios with obstacles.

Keywords

Evolutionary Algorithms; Optimization; Diversity Maintenance; Wind Farms Layout Optimization

1. INTRODUCTION

In population-based metaheuristics for optimization [1], the search for the global optima basically consists of an iterative process that replaces the candidate solutions in the population by newly generated ones. The cumulative effect of these iterations tends to be a continued improvement in the performance of the best candidate solutions. During this process, such new solutions are usually generated from previous ones, i.e. by recombination and mutation. The algorithm must provide mechanisms that allow the greatest possible exploration of the search space. In this context, the capability of maintaining diversity among the individuals in the population is crucial to avoid premature convergence to a specific region of the search space. This allows a better exploration of the domain of the problem and reduces the susceptibility of the algorithm to converging to local optima in multimodal optimization problems. In addition, maintaining diversity may also allow a given algorithm to locate several distinct high-quality solutions in a single run. A user can then choose the one that best fits its needs. This may save time in situations in which unpredicted constraints arise after the execution of the optimization algorithm, as a suitable solution may be present in the pool of solutions returned, thus avoiding the need to reconfigure and to rerun the optimization tool.

Optimization tools have been applied to many real-world optimization problems, but one that has recently gained particular interest is that of wind farm layout optimization. Finding suitable placements of wind turbines inside the wind farm for example to maximize energy production and/or minimize the cost, is a mixed integer-discrete-continuous nonlinear constrained optimization problem without an analytical formulation. It is mathematically complex and can't be solved by using classical analytical optimization techniques. Here, we propose a modification of a continuous optimization diversity operator called *Grid Diversity Operator* (GDO) [2] that can be applied to this discrete optimization problem. Some background to wind optimisation is given, and then we review GDO in section 3 before proposing a modified version for discrete optimization problems in section 4. Section 5 introduces the experimental protocol where we define the procedure to test the proposed GDO variant against a wind farm optimization problem as a case study. Finally, section 6 will provide the experimental results and the discussion.

2. THE WIND FARM LAYOUT PROBLEM

A wind farm is a group of wind turbines located at a site to generate electricity. Modern wind farms typically consist of hundreds of utility-scale wind turbines and with a total capacity of hundreds mega-watts. Design of efficient wind farms that maximize electricity production is a highly complex process with multiple and in many cases conflicting objectives under different constraints. It involves multiple design and engineering tasks, which may come from technical, logistical, environmental, economical, legitimacy and even social considerations [3]. Amongst these tasks, designing the *wind farm layout* is critical, i.e. finding suitable placements of wind turbines inside the wind farm. The wind farm layout optimization problem is concerned with determining the positions of the turbines inside the wind farm to maximize and/or minimize some objective functions. For

GECCO'16 Companion, July 20-24, 2016, Denver, CO, USA

 ISBN 978-1-4503-4323-7/16/07…$15.00

DOI: http://dx.doi.org/10.1145/2908961.2931656

example, this might include maximizing energy production and minimize the cost, while meeting various constraints, which may include wind farm boundary, wind turbines proximity, noise emission level, initial investment limit, and so on. In the most general case, it is necessary to consider multiple factors such as the selection of the number of wind turbines, different wind turbine types, and the discrete hub height.

The layout problem has been specified as a competition at recent GECCO conferences [4]. The competition supplied an API that returned the global energy capture of a given layout, accounting for wind distribution and wake effects caused by other nearby turbines. Most of the published approaches use a grid-based discrete formulation, which simplifies the searching space of the optimization problem from continuous space to discrete space. Full details of the problem, including download of the software and competition results are available from [4].

3. GRID DIVERSITY OPERATOR

The Grid Diversity Operator (GDO) was first described [2, 5] is a novel approach for achieving exploration and exploitation balance through maintaining diversity. It can be defined as a hybrid, non-niching, population-based, genotype diversity maintaining and learning technique. Simply put, GDO is a special infusion technique for initializing new individuals that are inserted into a population after a certain number of generations. Instead of randomly initializing individuals over the whole domain, GDO tries to initialize them in unexplored locations. A memory archive is used to store information collected throughout the run regarding the distribution of the individuals, and is used to infer rarely visited locations.

GDO works by dividing the search space into smaller subspaces using the grid size parameter $G_{sz} \in \Re^n$, which defines the number of intervals per dimension, where n is the number of dimensions for the problem. The GDO will then try to distribute new individuals into the grid slots that have received fewer visits over time, thus increasing the explorative power of the algorithm. The Grid Diversity Operator, *GDO*, is described in listing 1.

Listing 1 Grid Diversity Operator

```
Input: MemoryArchive,N_new,G_sz,P_th
Output: S_new
  S_new ← ∅
  for i = 1 to N_new do
    distributed ← FALSE
    while ¬ distributed do
      Key ← PickSlotAtRandom(G_sz)
      if KeyExist(MemoryArchive,Key) then
        V ← GetValueOfKey(MemoryArchive,Key)
        P ← e^-V
      else
        P ← 1
      end if
      if P > P_th then
        Individual ← CreateNewIndividualInSlot(Key)
        InsertIndividual(S_new,Individual)
        distributed ← TRUE
      end if
    end while
  end for
  Return (S_new)
```

First, a memory archive is initialized as an empty dictionary that has n component *keys*, where each of these keys matches a single value. The *keys* refer to the indices of a slot within the grid, while the *value* represents the number of individuals that have previously been placed in this slot. The memory archive is designed in this manner for efficiency: as most of initialized memory will be sparse, by using the dictionary only visited locations are stored, and therefore the use of memory is efficient regardless of problem dimensionality.

For each iteration in the containing algorithm, every new individual is processed to identify its slot to update the memory archive. For each individual being processed, if there is an entry in the archive with a key that matches the identified slot, the value corresponding to this entry is increased by one. Otherwise, a new entry is added to the archive with a value of 1. The process of updating the memory archive is demonstrated in listing 2.

After processing all individuals, the updated archive is used to initialize new individuals. For each new individual required, we pick a slot $S_{(d_1,d_2,\ldots,d_n)}$ at random and calculate its *distribution probability* P according to equation 1.

$$P = e^{(-N)} \tag{1}$$

where N is the value matching the slot key in the archive or zero if the slot does not yet belong to the archive. Finally, the calculated probability P is compared to the probability threshold parameter of the algorithm, P_{th}, and if $P > P_{th}$ then a new individual is initialized randomly in this specific slot. If not, another slot is picked at random, and the steps are repeated until the individual is initialized successfully.

Listing 2 GDO Update Archive Method

```
Input: MemoryArchive, Individuals, Resolution
Output: MemoryArchive
  for Individual_i ∈ Individuals do
    Key ← FindSlot(Individual_i)
    if KeyExist(MemoryArchive,Key) then
      IncreaseValue(MemoryArchive,Key,1)
    else
      AddKey(MemoryArchive,Key)
      SetValue(MemoryArchive,Key,1)
    end if
  end for
  Return (MemoryArchive)
```

The process of updating the memory archive and distributing new individuals continues until the algorithm terminates, at which point the final population is expected to be more diverse than simply using a random initialization procedure.

The work in[5] validated GDO for a selection of continuous optimization problems from *CEC 2014* benchmark suite[6] using two algorithms (Artificial Immune System Algorithm Opt-aiNET[7] and The SawTooth Genetic Algorithm[8]). The GDO operator was shown to achieve effective exploration through testing on the benchmark problems when incorporated within the two algorithms and it was shown that GDO can significantly help a supported algorithm to achieve better quality solutions in most cases.

4. MODIFIED GRID DIVERSITY OPERATOR FOR DISCRETE OPTIMIZATION

Many of the optimization problems in industry naturally fall into the class of discrete optimization problem. Recalling

that the grid diversity operator (GDO) [2, 5] was initially designed for and only considered continuous optimization, here we propose a modified GDO that supports discrete optimization problems.

The major change concerns the definition of the grid. Similar to continuous optimization problems, the grid is formed by dividing the search space into discrete sub-spaces with one exception: the number of partitions for a decision variable (which is defined by GDO grid size parameter G_{sz}) cannot exceed the number of all possible values for that variable.

An example of how the grid is formed for the modified GDO can be demonstrated by assuming a discrete optimization problem with two decision variables $x_1 = \{1, 2, ..., 9\}$ and $x_2 = \{A, B, ..., L\}$. Setting the GDO grid size parameter $G_{sz} = 3$ for this discrete problem yields that the very first slot will contain all nine possible solutions: $\{1, 2, 3\} \times \{A, B, C\} = \{\{1, A\}, \{1, B\}, ..., \{3, B\}, \{3, C\}\}$. During the runs and while updating GDO's memory archive, the individuals in each slot are counted and the counts are used to calculate the distribution probability which is used to initialize the new individuals. Figure 1 demonstrate this example by assuming some individuals in the slots and the individuals count per slot is shown on the lower-right corner for all slots.

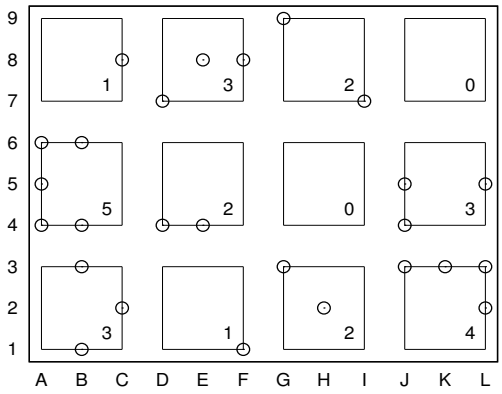

Figure 1: Demonstration of the modified GDO grid for a sample discrete optimization problem with two variables with $G_{sz} = 3$

Both listings (1, 2) are still valid for the modified GDO proposed here as the modifications lies on how the grid works in discrete search space which affect how the memory archive is defined and manipulated. The following points, however, should be noted:

- The memory archive in the modified GDO should not be confused with Tabu search [9] as it is *not* a tabu list and the sub-spaces inside it can still be used to initialized new individuals again. The selection process in GDO favors slots with fewer visits, but does not prohibit the use of any slot.
- GDO should be viewed as an additional layer for some metaheuristic optimization algorithms, but it is not an optimization algorithm by itself. It aims to improve an algorithm's performance by promoting more diversity within a population.

5. EXPERIMENTAL PROTOCOL

To assess the performance of the proposed modified GDO we inject it into the artificial immune algorithm OPT-IA[10]. An integral part of the OPT-IA algorithm is a step which adds new randomly generated solutions during each iteration of the algorithm. Note that there are many other candidate algorithms that support injection that could have been chosen. For example, GDO was previously used with opt-aiNet [7], saw-tooth [8]), with results described in [5]. However, here we choose to use OPT-IA as it has previously been shown to provide superior results to other algorithms on a range of optimization problems [10].

The algorithm is evaluated using GECCO's Wind Farm Layout Optimization Competition framework. This competition has been organized annually from 2014 with the aim of encouraging new approaches to solving the wind optimization problem. The supplied framework for the competition provides layout evaluators, wind field conditions, and baseline performances for the different layout optimization problems. The experiments here are conducted using the latest API and scenarios available and inherits the same rules such that the experimental results can be compared to the competition results. The aim is to optimize the wind farm layout of a set of 25 different scenarios (each represents wind forces, layout shapes, etc.) that consist of 20 demo scenarios (in which 10 of them include obstacles) and the remaining five scenarios are the ones used for the actual competition by minimizing the *Cost of Energy* which is calculated as defined in equation 2:

$$\text{fitness} = \left[\overbrace{\left(c_t * n + c_s * floor(\frac{n}{m})\right)}^{\text{Construction cost}} \overbrace{\left(\frac{2}{3} + \frac{1}{3} * e^{-0.00174n^2}\right)}^{\text{Economy of scale}} + \overbrace{c_{OM} * n}^{\substack{\text{Yearly}\\ \text{operating}\\ \text{costs}}} \right] * \underbrace{\frac{1}{\left(1 - (1+r)^{-y}\right)/r}}_{\text{Interests}} * \underbrace{\frac{1}{8760 * P}}_{\substack{\text{Yearly}\\ \text{power}\\ \text{output}}} + \frac{\overbrace{0.1}^{\substack{\text{Farm size}\\ \text{coefficient}}}}{n} \tag{2}$$

where, $c_t = \$750,000$ is turbine cost, $c_s = \$8,000,000$ is substation cost, $m = 30$ is no. turbines per substation, $r = 0.03$ is the interest rate, $y = 20$ is farm lifetime in years, $c_{OM} = \$20,000$ is operating cost per turbine, n is maximum number of turbines and P is Farm energy output with both n and P are scenario dependent.

5.1 Immune Algorithm OPT-IA

The OPT-IA algorithm introduced in[10] is one of many immune algorithms inspired by the Clonal Selection (*CS*) principle. The one feature that make OPT-IA standout against the other *CS* algorithms is the *Aging Operator* which is special diversity operator designed to control the diversity of the population. For the aging operator to work, all solutions are given additional property, namely *Age*, which records how old that solution is.

First, a population is initialized with d randomly generated solutions each with *Age* set to zero. All the individuals are then evaluated against the objective function and then the population proceeds with the typical *CS* operators; *cloning* and *hypermutation*; where the hypermutation is inversely proportional to the fitness value. Both parent pop-

ulation and successful hypermutated clones (the ones with better fitness than their parents) suffer through aging operator where the solutions with $Age > MaxAge$ are discarded.

The final step of the algorithm is called $(\mu + \lambda)$-*Selection* in which the best d solutions from the survivals of the aging operator have their *Age* property increased by one and added to the next population. If the survivals count is less than d then the next population is completed be creating new randomly generated solutions.

Listing 3 show the main steps of OPT-IA algorithm where: d is the fixed population size, *dup* is the number of duplicates/clones per individual, ρ is the mutation parameter, τ_B is the maximum age, T_{MAX} is the maximum number of function evaluations.

Listing 3 Immune Algorithm OPT-IA

```
Input: d, dup, ρ, τB, TMAX
Output: Population
  FFE ← 0
  Nc ← d × dup
  t ← 0
  P_d^(t) ← PopulationInitialization(d)
  Evaluate(P_d^(t))
  FFE ← FFE + d
  while FFE < TMAX do
    P_Nc^(clo) ← Cloning(P_d^(t), dup)
    P_Nc^(hyp) ← Hypermutation(P_d^(clo), ρ)
    Evaluate(P_Nc^(hyp))
    FFE ← FFE + Nc
    (^a P_d^(t), ^a P_Nc^(hyp)) ← Aging(P_d^(t), P_Nc^(hyp), τB)
    P_d^(t+1) ← (μ + λ)-Selection(^a P_d^(t), ^a P_Nc^(hyp))
    t ← t + 1
  end while
  Population ← P_d^(t)
  Return (Population)
```

5.2 OPT-IA for Wind Farm Layout Optimization Problem and Injection with Modified GDO

Although `opt`-IA algorithm was initially designed for continuous optimization, we were able to implement it for wind farm layout optimization with few modifications. The first is to define what an individual represents. Here, an individual for `opt`-IA defines a solution as usual but here it is coded as a $N_t \times 2$ matrix in which each row is a coordinate for a turbine and N_t is the number of turbines of the solution.

The other change was the hypermutation process which was implemented on two phases. The first phase is to make a random change in the number of turbines in the solution by adding/removing one turbine according to a random number. This step is necessary as the competition instructions were to optimize both the number and the location of the turbines. The second phase is the affinity proportionate mutation in which a number of turbines are relocated to different sites. We first calculate the mutation rate, α, according to equation 3 where ρ is the mutation parameter and f^* is the normalized fitness of the solution.

$$\alpha = (1/\rho) \cdot e^{(-f^*)} \tag{3}$$

The number of turbines to relocate, nm, is then calculated according to equation 4. Finally, the nm turbines are chosen randomly from the solution and are assigned new coordinates.

$$nm = \text{floor}\left[2 \cdot \alpha \cdot N_t\right] \tag{4}$$

As previously stated, the `opt`-IA algorithm feature adding new randomly generated solutions through $(\mu + \lambda)$-Selection step. To inject the algorithm with GDO, we strip the random solutions generator and simply replace it with GDO. The GDO injected $(\mu + \lambda)$-Selection method is shown in listing 4.

Listing 4 $(\mu + \lambda)$-Selection for GDO injected OPT-IA

```
Input: d, Parents, Clones, MemArchive, G_sz, P_th
Output: NextPop
  NextPop ← {Parents ∪ Clones}
  NextPop ← RemoveRedundancy(NextPop)
  if |NextPop| ≥ d then
    NextPop ← SelectBest(NextPop, d)
  else
    N_new ← (d − |NextPop|)
    MemArchive ← GDO–Update(MemArchive, NextPop, G_sz)
    NewIndividuals ← GDO–Create(MemArchive,N_new, G_sz, P_th)
    NextPop ← {NextPop ∪ NewIndividuals}
  end if
  Return (NextPop)
```

6. RESULTS AND DISCUSSION

Two versions of `opt`-IA algorithms are implemented for the experiments. The first is implemented without any modification to the original algorithm according listing 3. The second includes the GDO injection step inside the $(\mu + \lambda)$-*Selection* phase as shown in listing 4. Because the search space is not big per dimension/turbine, smaller values for GDO grid size are better and here we empirically chose $G_{sz} = 1$ for this specific problem. This value for G_{sz} means each slot in GDO's memory archive will refer to a sub-space that only contain a specific solution. In addition, we chose to use a probability threshold for GDO $P_{th} = 0.2$ for all the tests. The rest of `opt`-IA algorithm parameters used for both implementations are as defined in[10]: $d = 20$, $dup = 1$, $\rho = 150$, $\tau_B = 10$. Both versions ran 25 times each for $T_{MAX} = 10,000$ evaluations per scenario as defined by the competition and the best solution of all runs (with respect to cost of energy) is recorded.

In figure 2, where all scenarios are obstacle-free, we can see that the GDO injected version of `opt`-IA achieved noticeable improvement over the clean version for demo scenarios one and two. The other scenarios show no significant difference between the two implementations although GDO version is still better performer except for demo scenario 3.

The results for demo scenarios *with* obstacles show more improvement while using GDO as shown in figure 3. The GDO implementation achieved significantly lower cost of energy in six out of the ten scenarios. The results of the rest of scenarios (13, 18, 19 and 20) where about the same with no significant difference visible.

Finally, and more importantly, figure 4 shows the summarized results for both `opt`-IA implementations over competition scenarios which are more complex and are much challenging. The GDO version dominated the clean `opt`-IA for scenarios two, three and four while falling behind for scenario five and having the same performance for the first scenario.

Combining the results for all the experiments together, we find that GDO significantly helped the containing algorithm (`opt`-IA) for scenarios with obstacles more than those without obstacles. In an attempt to comprehend and explain this behavior, we analyzed the way GDO affect the

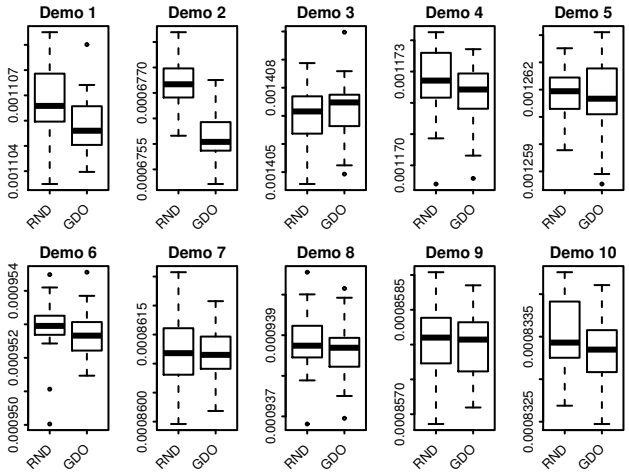

Figure 2: Cost of Energy comparison between clean opt-IA (RND) against injected version (GDO) for the ten demo scenarios *without* obstacles

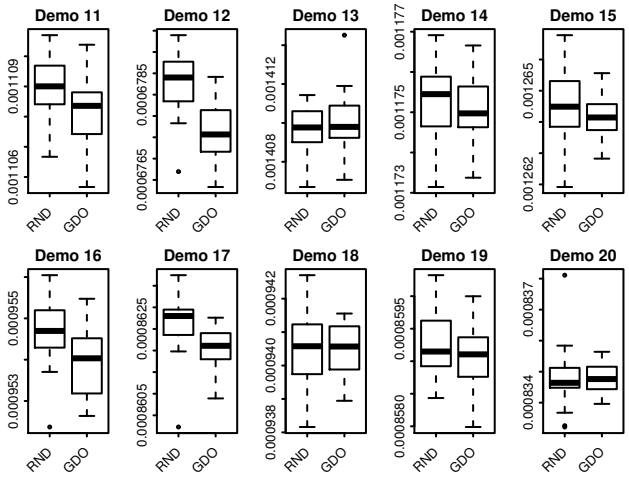

Figure 3: Cost of Energy comparison between clean opt-IA (RND) against injected version (GDO) for the ten demo scenarios *with* obstacles

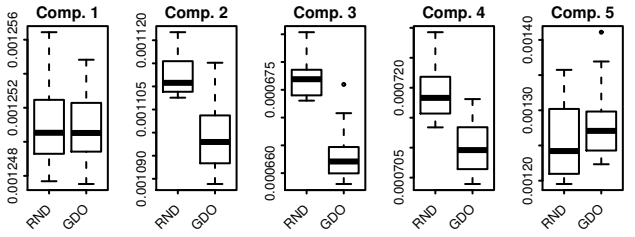

Figure 4: Cost of Energy comparison between clean opt-IA (RND) against injected version (GDO) for the five competition scenarios

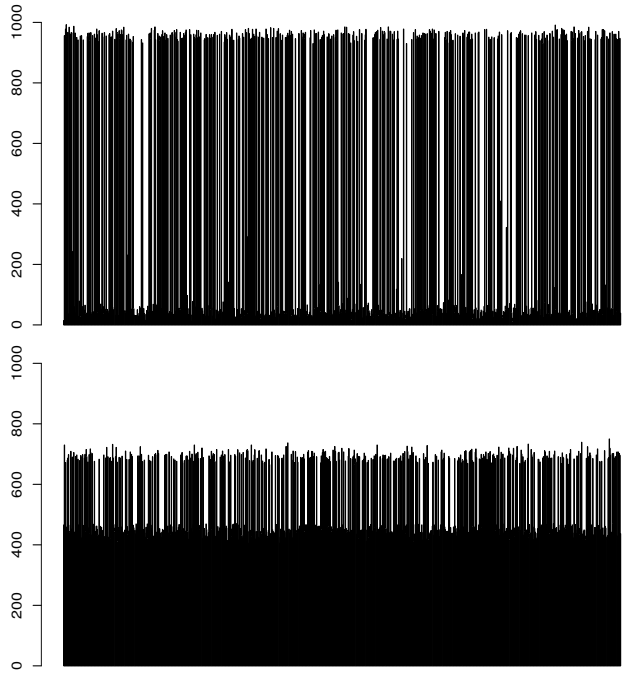

Figure 5: Farm grid locations usage count for demo scenario 12. Top: clean opt-IA (RND), Bottom: injected opt-IA (GDO)

algorithm during the runs. First we computed the number of times a feasible location (within the grid) is used within a layout during the best run for both implementations of opt-IA. Figure 5 shows a comparison of these numbers between the two implementation for demo scenario 12 (which have obstacles and where GDO version dominates). On top it is clear from the distribution of the bars that many locations are either never visited or visited infrequently. On the other hand, using the GDO version, it is clear from the bottom graph in figure 5 that all feasible locations have been visited at least once during the run. This does not come as a surprise since GDO is exactly designed to promote diversity within the populations in an attempt to discover better solutions.

Comparing the performance gain by using GDO with opt-IA with other published work is not trivial as raw results are not available from the second edition of the wind farm layout optimization competition. A slide was released online [4] after the competition was over providing a graph, from which it is difficult to extract much useful information and numbers. Nevertheless, we still attempted to stack the results of both clean and GDO versions of opt-IA to the results table provided in the competition published slides. Figure 6 shows how the results of the two implementations of opt-IA stack against those of the eight competitors.

We did not expect that either of the implementation would outperform any of the competition results simply because all the competition algorithms are customized and geared towards solving this specific problem. Recall that our approach treats the optimization problem simply as a black-box, with no specialized operators and no use of any domain knowledge. Despite this, it appears that the two versions results were at least comparative to the competitions result. Moreover, we found that GDO version of opt-IA achieved slightly better results than both competitors seven and eight for scenarios two and three.

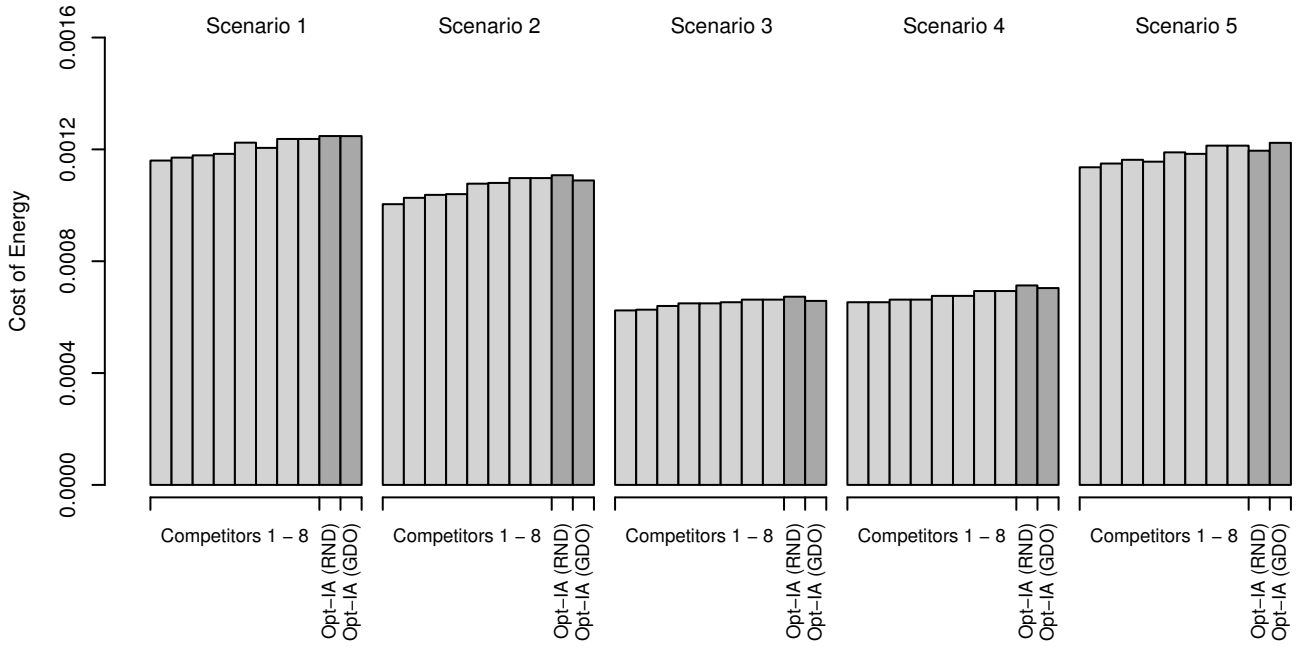

Figure 6: Comparison of *Cost of Energy* between the competition's 8 competitors and the two versions of `opt`-IA algorithm

7. CONCLUSIONS

The work here introduced a modified version of the grid diversity operator (GDO) to handle discrete optimization problems. The main idea was to replace the virtual grid originally used in GDO with a simpler structure in which solutions themselves are added to the memory-archive instead of a slot that does not exist in such type of problems. The new operator has been injected into the immune algorithm `opt`-IA and a series of experiments have been conducted on a set of wind farm optimization problem scenarios that vary in complexity. The results obtained from experimentation prove that the modified GDO technique achieving better results on many of the instances, especially for the scenarios with obstacles. However, there is still a need to validate the modified GDO using different other discrete optimization problems. In addition, a unified grid diversity operator that is suitable for both continuous and discrete optimization should be implemented and validated.

8. REFERENCES

[1] L. N. de Castro. *Fundamentals of Natural Computing: Basic Concepts, Algorithms, and Applications*. Chapman & Hall/CRC Computer & Information Science Series. Chapman & Hall/CRC, 2006.

[2] A. Salah and E. Hart. Grid diversity operator for some population-based optimization algorithms. In *Proceedings of the Companion Publication of the 2015 on Genetic and Evolutionary Computation Conference*, pages 1475–1476. ACM, 2015.

[3] P. Jain. *Wind Energy Engineering*. New York: The McGraw-Hill Companies, Inc., 2011.

[4] D. Wilson, S. Cussat-Blanc, S. Rodriguez, and K. Veeramachaneni. Wind Farm Layout Optimization Competition – 2nd Edition. https://www.irit.fr/wind-competition/2015/slides.pdf, 2015. [Online; accessed 05-April-2016].

[5] A. Salah, E. Hart, and K. Sim. Validating the grid diversity operator: An infusion technique for diversity maintenance in population-based optimisation algorithms. In G. Squillero and P. Burelli, editors, *Applications of Evolutionary Computation: 19th European Conference, EvoApplications 2016, Porto, Portugal, March 30 – April 1, 2016, Proceedings, Part II*, volume 9598 of *Lecture Notes in Computer Science*, pages 11–26, Cham, 2016. Springer International Publishing.

[6] J. J. Liang, B-Y. Qu, and P. N. Suganthan. Problem Definitions and Evaluation Criteria for the CEC 2014 Special Session and Competition on Single Objective Real-Parameter Numerical Optimization. Technical report, Computational Intelligence Laboratory, Zhengzhou University, Zhengzhou, China, December 2013.

[7] L. N. de Castro and J. Timmis. An artificial immune network for multimodal function optimization. *Evolutionary Computation, 2002. CEC'02. Proceedings of the 2002 Congress on*, 1:699–704, 2002.

[8] V. K. Koumousis and C. P. Katsaras. A Saw-Tooth Genetic Algorithm Combining the Effects of Variable Population Size and Reinitialization to Enhance Performance. *IEEE Transactions on Evolutionary Computation*, 10(1):19–28, feb 2006.

[9] F. Glover. Tabu search - part 1. *ORSA Journal on Computing*, 1(3):190–206, 1989.

[10] V. Cutello, G. Narzisi, G. Nicosia, and M. Pavone. An Immunological Algorithm for Global Numerical Optimization. In *Proceedings of the Intl. Conf. on Evolution Artificielle, EA '05*, pages 284–295, 2006.