BlockCopy-based operators for evolving efficient wind farm layouts

Michael Mayo and Chen Zheng
Department of Computer Science
University of Waikato
Hamilton, New Zealand
mmayo@waikato.ac.nz, cz90@students.waikato.ac.nz

***Abstract*—A novel search operator, BlockCopy, is proposed for efficiently solving the wind farm layout optimisation problem. BlockCopy, which can be used either as mutation or a crossover operator, copies patterns of turbines from part of a layout to another part. The target layout may be the same as the source, or a different layout altogether. The rationale behind this is that it is the relative configurations of turbines rather than their individual absolute positions on the layouts that count, and BlockCopy, for the most part, maintains relative configurations. Our evaluation on four benchmark scenarios shows that BlockCopy outperforms two other standard approaches (namely, the turbine displacement algorithm and random perturbation) from the literature. We also evaluate the BlockCopy operator in conjunction with both single-solution and population-based strategies.**

***Index Terms*—wind farm layout optimisation problem, evolutionary strategy, search operator, cost of energy, turbine displacement algorithm**

I. Introduction

Wind is rapidly becoming an important global source of renewable energy. Worldwide, the Global Wind Energy Council predicts that wind energy production could reach as much as 2,000 gigawatts (GW) by 2030 – this would amount to approximately 18% of the world's energy production [5]. Cost reduction in the production of wind energy is therefore crucial. Currently, scientific research from many disciplines plays an important role in achieving cost reductions. In the computational intelligence community, related wind research frequently focusses on the development of new experimental optimisation and prediction techniques. For example, Preen & Bull [17] propose a co-evolutionary method for evolving new wind turbine designs, Razavi-Far & Saif [18] discuss a method of imputing missing data in wind turbine sensors, and Veeramachaneni et al. [20] describe an approach that predicts the wind resource at sites using less data than industry typically collects – therefore saving time during expensive site assessments.

The particular problem we are focussed on in this paper is the wind farm layout optimisation (WFLO) problem [7, 19]. The WFLO problem occurs once a suitable site for a wind farm has been located, and a decision about the geographical placement and other properties of the individual turbines making up the farm has to be made. Overall, wind farm design is a complex decision problem involving many conflicting factors such as the design of the mechanical and electrical infrastructures, the leasing of land from landowners, and the impact of the farm on the environment. The management of the uncertainty of the wind energy itself is also a major issue. The WFLO problem is therefore a small component of the solution to this problem, but it is a significant component because slight efficiencies brought about by careful turbine siting can result in significant cost savings during the twenty year lifetime of a typical farm.

978-1-4799-8697-2/15/$31.00

In this paper, we propose a novel class of operators for a layout optimiser that can be used with an evolutionary strategy or similar metahuristic. The key novel idea behind these new operators is to *reuse* small regions within layouts by copying them.

This approach follows from this intuitive idea: frequently as a layout is optimised, it is not optimised uniformly. Instead, parts of the layout are optimised first (because "good" mutations occur at those points more frequently), and then the less-fit parts of the layout may be improved in due course if the algorithm runs for long enough.

To compensate for this mismatch, therefore, pattern copying can be attempted. If a pattern is copied wholesale from a "good" part of the layout to a "poor" part, then it can be expected that the overall fitness of the layout will dramatically increase. Since selection favours higher-fitness layouts, the result is that higher fitness regions will tend to be copied more often.

Such an approach basically changes the focus of the mutation operator from individual turbines (as has been common in the literature, e.g. the turbine displacement algorithm [21]) to larger patterns of turbines. The particular class of pattern we focus on here is the simplest possible, specifically blocks of size 1km $\times$ 1km, and therefore our operators are named *BlockCopy mutation* and *BlockCopy crossover*.

In an extensive evaluation combining the BlockCopy operators with an evolutionary strategy (ES), we show that compared to the current state-of-the-art, BlockCopy-powered ESes are significantly more effective optimisers.

The paper is organised as follows. In the next section, we cover background about the WFLO problem, and describe in detail the current state-of-the-art. We also briefly mention ESes. In the section after that, we describe our proposed

BlockCopy operators, and then in the subsequently, we perform an extensive evaluation of our approach on four benchmark scenarios from the literature.

II. Background

A. Wind farm layout optimisation (WFLO) problem

The WFLO problem is an abstracted representation of the problem faced by wind farm designers when they must select positions on a site to place individual wind turbines. This process is referred to as *micrositing*.

The main problem faced by wind farm layout designers when micrositing is that of turbulence: if one turbine lies in the wake of another turbine, or in the wake of a large structure such as a building, then it may experience a reduced wind velocity as a result of the increased turbulence and vorticity caused by the obstructions. Consequently the energy that the turbine can extract from the wind is reduced compared to the energy that could be extracted if the turbine was unobstructed.

In a wind farm site with few large fixed obstacles, the main cause of this energy loss is the presence of other turbines – hence the problem becomes one of optimising the turbine positions in order to minimise wake effects.

The primary method of evaluating a wind farm layout is via simulation, i.e. given data about expected wind speeds and directions, the expected wake effects between pairs of turbines for each wind direction can be calculated, and then the overall expected energy harvest or energy cost of the farm can be computed.

Simulation methods for evaluating wind farm configurations range from highly accurate but also highly complex computational fluid dynamics approaches [7] to methods such as that proposed by Kusiak & Song [9] which while simpler and computationally more feasible, are none-the-less accurate enough for industry if certain assumptions hold.

A key point to note is that any algorithm for evaluating one wind farm layouts lies in the class NPO-complete [7]. In other words, the time complexity of evaluating a configuration is a polynomial function of the layout's size.

Furthermore, an added complication is that the evaluation function is also non-parallelizable because wake interactions must be calculated sequentially [7]. For example, suppose that turbine A is upstream of both turbines B and C, but B is also upstream of C. The wake effects on the last turbine C cannot be computed until the wake effect of A on B is known, since B also has an effect on C. Therefore the interactions must be modelled individually and in sequential order.

In terms of the search space, the WFLO problem is continuous, constrained, and non-differentiable, with turbines being able to be placed at any valid location as long as they do not violate the minimum distance constraint between turbines (which is typically a distance of eight times the turbine rotor's radius) or collide with obstacles. Some approaches to the problem (e.g. the seminal work of Mosetti [15]) additionally discretise the search space, thus converting the problem from that of continuous optimization into one of combinatorial optimisation where a position either contains a turbine or does not contain a turbine. In this case, the size of the search space is $\mathcal{O}(2^n)$ where n is the number of grid cell positions [7].

In this paper we focus on the most common 2D planar layout version of the WFLO problem in which all turbines are assumed identical. Each turbine varies only in its (x, y) position on the plane. Furthermore, we also assume that the number of turbines is fixed. Despite these simplifications, the problem is still a high-dimensional, highly multimodal, and complex optimisation. This variant is also the most commonly-tackled version of the problem in the literature.

Less simplified versions of the problem have also been investigated. For example, both Chen et al. [4] and Wang et al. [22] consider variants of the problem where land-owners must be consulted and may charge a fee for participation in the wind farm project. Finding the most crucial and cost-effective plots of land on which to site the turbines is therefore an important part of the problem.

In another version of the problem, Ate et al. [2] and Gonzalez et al. [6] include in their evaluation function considerable detail pertaining to the construction of the wind farm, such as the required electrical infrastructure and roading networks.

Finally, a small current number of works consider the environmental impacts of wind farms. For example, Kwong et al. [10, 11] consider the effects of noise, and attempt to find layouts that minimise noise propagation while maximising energy harvest. Similarly, Neubert et al. [16] and Al-Yahyai et al. [1] both propose methods of generating visually appealing layouts that preserve geometric regularities, whilst also maintaining near-optimal energy efficiency.

B. Turbine displacement algorithm

The current state-of-the-art algorithm in the literature for optimising a wind farm layout is the Turbine Displacement Algorithm (TDA) proposed by Wagner [21]. In two recent extensive evaluations, both Wagner [21] and Dennis et al. [24] found that TDA outperformed all other approaches including genetic algorithms, particle swarm optimisation, and developmental models in terms of finding layouts with the minimum expected wake losses across multiple different benchmark scenarios.

In essence, TDA is a very simple local search algorithm that shifts one random turbine at a time, and then evaluates the modified layout. If the modified layout is at least as good as the original layout, then the algorithm keeps the modified layout and discards the original.

The interestingness of TDA lies in its heuristic for shifting each turbine. Because wake effects are reduced with distance, the algorithm makes the simplifying assumption that only the K nearest neighbouring turbines to the given turbine are important. A *displacement vector* is then calculated, which is a vector pointing in a direction *away* from the K nearest neighbours. The rationale for this is that moving the turbine away from its neighbours is usually more likely to decrease wake interference – although occasionally moving turbines closer together is actually beneficial and therefore the displacement

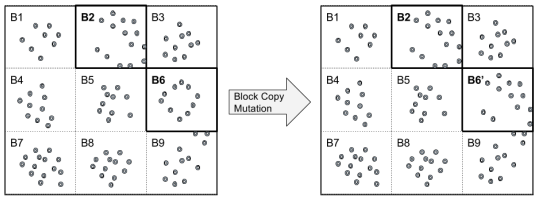


Fig. 1: Illustration of the BlockCopy mutation operator. In this example block B_2 is chosen for mutation and copied to the position occupied by B_6.

vector may be reversed in direction with a small probability p. The displacement vector is also perturbed with angular noise, the magnitude of the noise being determined by a parameter σ_{dir}.

When the displacement vector has been computed, it is added to the current turbine's position to get its new position. If the turbine's new position is invalid (e.g. outside of the layout, or colliding with an obstacle), then the displacement vector is gradually reduced in magnitude until the turbine's new position becomes valid.

C. Evolutionary strategies

In this paper we focus on the well-known evolutionary strategies (ES) [3] family of algorithms. The standard (μ, λ) or $(\mu + \lambda)$ ESes presented by Luke [13] are utilised in our experiments. The use of ESes to solve the WFLO problem is quite common in the recent literature. For example, the work of Lückehe et al. [12] extensively investigates a variety of different ESes for solving this problem on some challenging scenarios. ESes with standard operators, along with TDA, therefore, are useful baselines for evaluating new approaches.

III. BlockCopy Operators

In this section, our proposed BlockCopy operators are described in more detail.

A. BlockCopy Mutation

The basic idea behind the BlockCopy mutation operator is to first of all divide the layout into approximately square blocks, and then to copy the turbines from one randomly chosen block (the "source") to another (the "target"). The turbines within the target block are deleted from the layout before the copy happens so that source pattern is maintained.

This process is illustrated in Figure 1. When a block is copied, the *relative* configuration of the turbines inside the block are maintained. Only the absolute position of the copied turbines change. Therefore this process is essentially a copy/translate of the source block to a new position.

Two problems may arise when blocks are copied in this way across layouts. Firstly, copied turbines may collide either with an obstacle, or with another turbine in a neighbouring block (for example, if two turbines are near the edges of their respective blocks, they may end up too close together thus making the resulting layout invalid). In this case, the copied turbine is simply not placed in order to ensure that all constraints are always satisfied.

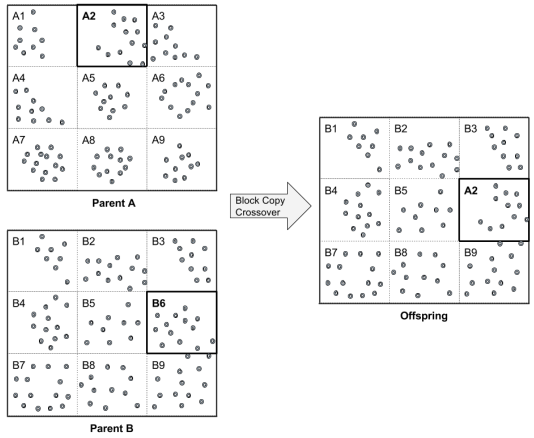


Fig. 2: Illustration of the BlockCopy crossover operator. In this example block an offspring layout is created by the copying of a block from one parent to a copy of another.

Secondly, because blocks may differ in the number of turbines they comprise, the copying may result in the total number of turbines across the layout globally changing. In this case, a global correction is applied after the block copying: turbines are either randomly placed onto the layout, or randomly purged from it, until the total number of turbines is returned to the desired fixed quantity. This step may be omitted if the total number of turbines being optimised is not fixed.

B. BlockCopy Crossover

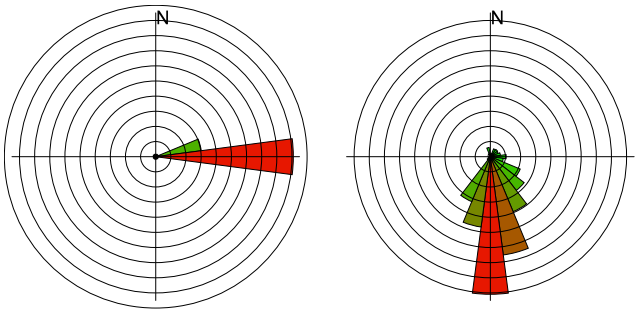
Our proposed BlockCopy Crossover operator works in a similar way to the mutation operator in that a block of a fixed size is copy/translated. However, the major difference with crossover is that not one but two layouts are involved: layouts exchange blocks, and one offspring layout is produced per crossover. This is illustrated in Figure 2. Rather than propagating a pattern within a single layout, therefore, such a crossover strategy allows good patterns to propagate throughout a population of layouts instead.

In this paper, we have limited crossover to the exchange of a single block rather than half of the blocks. Initial tests suggested that exchanging half of the blocks results in an overly disruptive crossover that reduced performance.

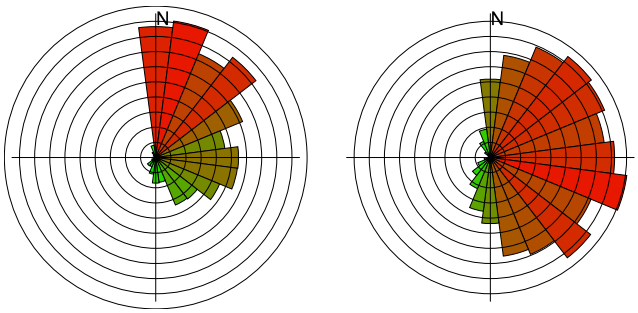
We also note that the current version of BlockCopy proposed here treats the optimisation problem as a "black box". That is, only the final objective cost of each layout is used. In contrast, other approaches (e.g. [14]) treat the optimiser as a "white box" and use intermediate values in the cost computation to provide a search bias. However, such approaches are unfair if compared directly to black box approaches, and so we leave a white box variant of BlockCopy for future work.

IV. Evaluation Scenarios

To evaluate our BlockCopy operators, we selected four benchmark scenarios from the recent literature. The first two scenarios are proposed in Kusiak & Song [9]. Kusiak & Song's first scenario is a very simple test scenario, in which wind blows predominantly in a single direction. This is an artificial



(a) Kusiak & Song 1 [9] (b) Kusiak & Song 2 [9]



(c) 2014 Comp 1 [23] (d) 2014 Comp 3 [23]

Fig. 3: Wind rose used in each scenario. Each wind rose gives the proportionate expected wind speed in each direction. Directions are discretised into 15° bins. m/s in all directions.

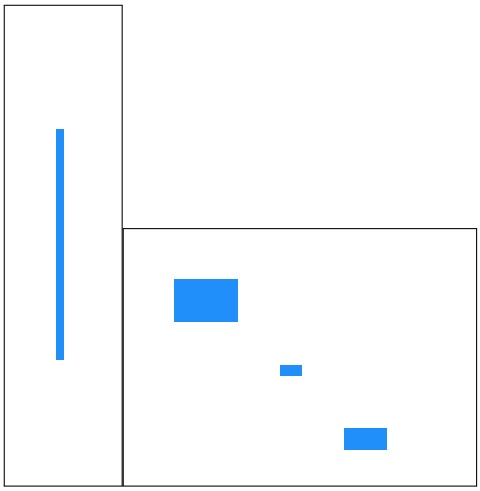
example. The second scenario, however, is more realistic and describes the wind speed/direction distribution at an actual industrial site [9]. In both cases, we have set the size of layouts to 4km × 4km and fixed the number of turbines to 100.

For the second two scenarios, we selected two benchmarks from the 2014 Wind Farm Layout Optimisation competition [23]. These are more challenging scenarios with a larger rectangular area than the first two scenarios, and they also contain obstacles where turbines cannot be placed.

Table I describes the four scenarios. Included in this table are the fixed number of blocks (across and down) that each scenario was divided into for use with the BlockCopy operators. In each case, the quantity of blocks was chosen so that each block was approximately 1km × 1km in area. The Weibull k parameter for estimating the wind speed at turbine height is also included in the table to show how it varies across scenarios.

Figures 3 and 4 graphically depict the data for each scenario. Figure 3 illustrates the wind speed/direction distributions, and Figure 4 shows the layout configuration with obstacles for the latter two scenarios.

The cost function we use was originally proposed and described by Kusiak & Song [9]. The variant of the function that we use here is an extended form used during the 2015 Wind Farm Layout Optimisation Competition [23]. Essentially, cost is defined as the expected cost per kilowatt of energy produced by the farm. It is calculated by taking the total cost of the farm (including construction and yearly operating costs) and dividing that by the total power output of the farm.



(a) 2014 Comp 1 (b) 2014 Comp 3

Fig. 4: Layouts with obstacles used in each scenario. Layouts are not shown to scale.

$$cost = \frac{(c_t \times n + c_s \times \lfloor \frac{n}{m} \rfloor)(\frac{2}{3} + \frac{1}{3} \times e^{-0.00174n^2}) + C_{OM} \times n}{(1-(1+r)^{-y})/r} \times \frac{1}{8760 \times P} + \frac{0.1}{n}$$

More formally, the cost function is defined as shown in the equation where $c_t = 750,000$ is the cost in USD of a turbine; $c_s = 8,000,000$ is the cost in USD of a substation; $m = 30$ is the number of turbines per substation; $r = 0.03$ is the interest rate; $y = 20$ is the lifetime in years of the farm; $C_{OM} = 20,000$ is the cost of operations and maintenance, again in USD; n is the number of turbines (i.e. the farm size); and P is the total energy output of the farm. The interested reader should see [9] and [23] for complete details and explanations of this cost function.

V. Evaluation: BlockCopy Mutation for (1+1)-ES

The BlockCopy mutation operator was evaluated in conjunction with a (1+1)-ES in the first set of experiments. Two main baseline algorithms were chosen for comparison. These were TDA [21], as well as another (1+1)-ES that differed from the experimental one in that it made use of a random perturbation operator instead of BlockCopy. The random perturbation operator simply moves a fixed number of randomly selected turbines to new randomly selected valid positions. Effectively, this approach ignores and likely even disrupts local configurations of turbines. In contrast, TDA moves only one turbine at a time, but the new position for each turbine is more carefully chosen.

We also added a fourth algorithm to the evaluation, specifically an algorithm combining BlockCopy and random perturbation. For this algorithm, whenever a layout is mutated, one of the two above operators (BlockCopy or random perturbation) is chosen randomly with equal probability. The rationale for this was to determine if both operators could enhance each other's

Scenario	Width (km)	Height (km)	# Turbines	Width (blocks)	Height (blocks)	Obstacles?	k
Kusiak & Song 1 [9]	4.0	4.0	100	4	4	No	2.0
Kusiak & Song 2 [9]	4.0	4.0	100	4	4	No	2.0
2014 Comp 1 [23]	3.5	16.1	220	3	16	Yes	2.187–3.624
2014 Comp 3 [23]	15.8	11.3	710	16	11	Yes	2.016–4.473

TABLE I: Wind Scenario dimensions, number of turbines, number of blocks, and k parameter.

Algorithm	Parameter	Value
TDA/all ESes	Max. num. evaluations	2,000
TDA	σ_{dir}	$\frac{\pi}{6}$
TDA	p	0.2
TDA	init. displacement vector size	$1.05 \times 8R$
ES+Perturb/ES+Both	Perturbation size	10 turbines

TABLE II: Parameter settings used in our experiments. (Note: R refers to the turbine rotor radius.)

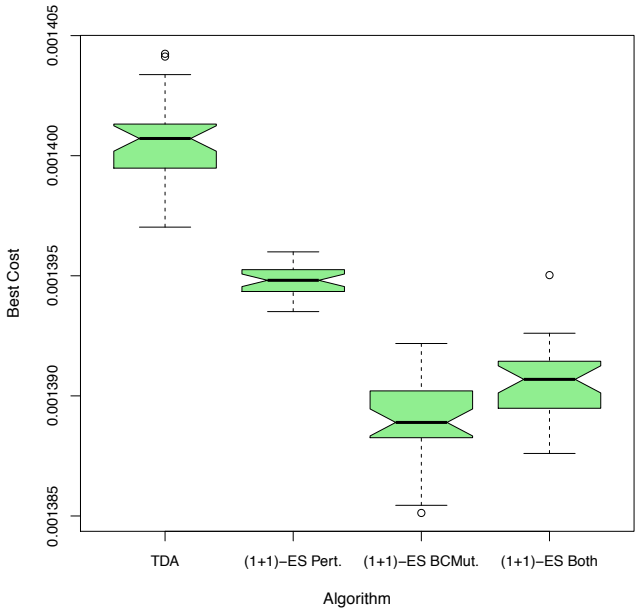


Fig. 5: Cost of best layouts found for the Kusiak & Song 1 [9] wind scenario.

performance (for example, the random perturbation operator may create new useful patterns that could then be copied by BlockCopy).

The parameter settings used for each run of the algorithm are shown in Table II. For TDA, the parameter settings are the same as used in the original publication on TDA [21], except for the initial displacement vector size which is not specified in the paper. Instead we have chosen a sensible value of 105% of the minimum distance between turbines. The termination criteria for each algorithm is the number of evaluations.

Each algorithm was repeated 30 times on each scenario, yielding a total of 4 algorithms $\times$ 4 scenarios $\times$ 30 repetitions = 480 experimental runs.

The final results are shown in Figure 5-8. Each figure depicts a set of box-and-whisker plots showing the distribution of best costs achieved by each algorithm on each scenario.

As can be observed, the distributions are quite different

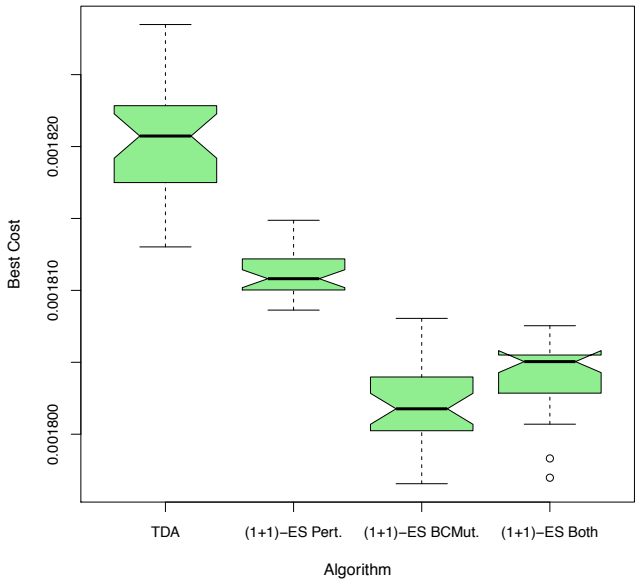


Fig. 6: Cost of best layouts found for the Kusiak & Song 2 [9] wind scenario.

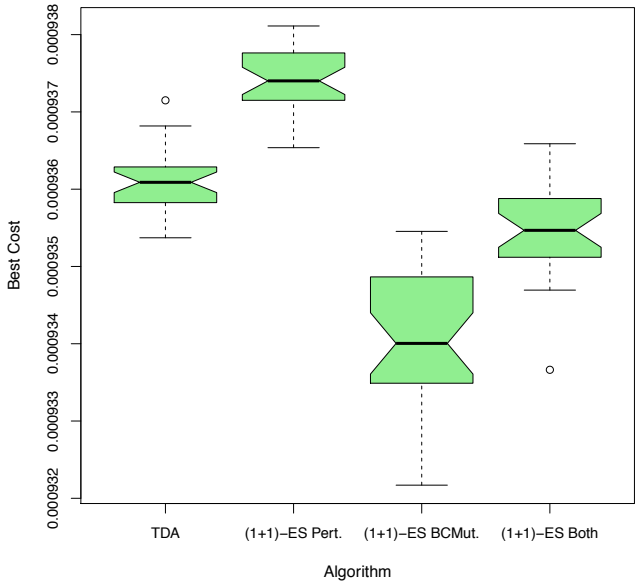


Fig. 7: Cost of best layouts found for the 2014 Comp 1 [23] wind scenario.

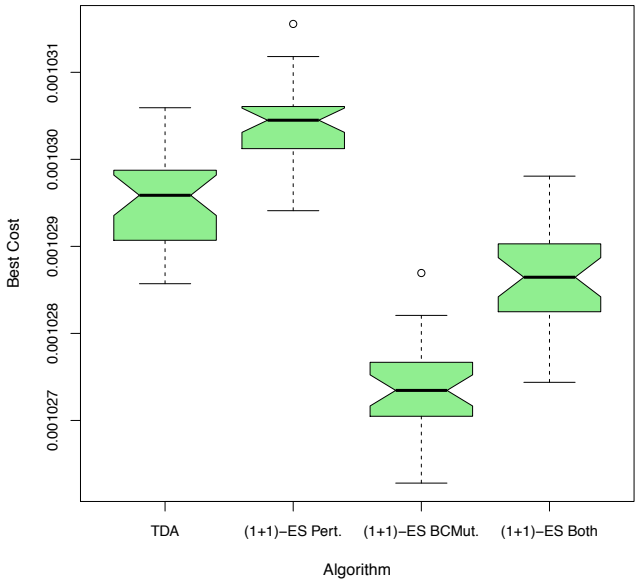


Fig. 8: Cost of best layouts found for the 2014 Comp 3 [23] wind scenario.

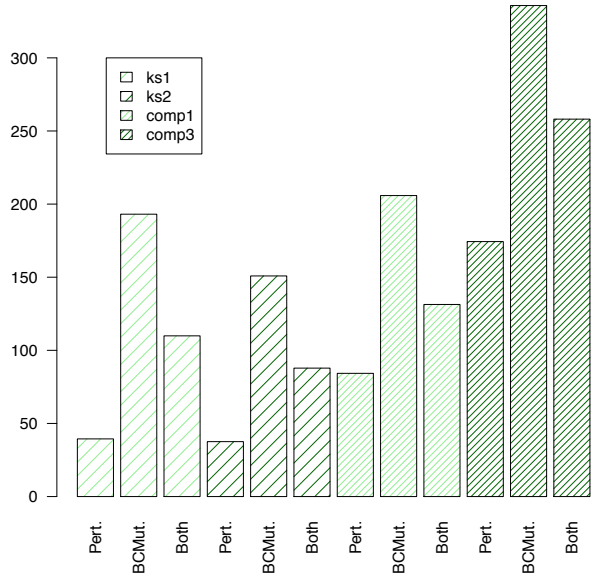


Fig. 9: Number of accepted mutations by scenario and 1+1 ES variant.

	(1+1)-ES BCMut. vs. ...		
Scenario	TDA	(1+1)-ES Pert.	(1+1)-ES Both
ks1	8.5×10^{-18}	8.5×10^{-18}	9.5×10^{-4}
ks2	8.5×10^{-18}	8.5×10^{-18}	4.6×10^{-4}
comp1	5.9×10^{-17}	1.5×10^{-11}	8.0×10^{-8}
comp3	1.7×10^{-17}	8.5×10^{-18}	6.4×10^{-10}

TABLE III: Results of standard Wilcoxon one-sided statistical tests comparing the distribution of results obtained using the (1+1)-ES with BlockCopy against the other algorithms. Shown are the p-values computed by the test.

for each algorithm. The ES with BlockCopy is clearly the superior algorithm in each case, achieving both the single layout with lowest overall cost, and the lowest median cost. Furthermore, the plots suggest that the differences in the costs are statistically significant because the notched regions of the box portions of the plots do not overlap.

To more precisely explore this, we also performed Wilcoxon one-sided tests comparing the performance of the (1+1)-ES using BlockCopy mutation vs. each of the other algorithms in turn. The results are shown in Table III. As can be observed, the p-values computed by all of the tests are well below the 0.01 level required for 99% confidence that the true median performance of the (1+1)-ES with BlockCopy is less than that of the other algorithms.

Next, we examined the performances of the three different ESes in terms of the number of accepted mutations. This is an important statistic to examine because a (1+1)-ES is prone to becoming trapped in local optima, and this may be indicated by a low number of accepted mutations. The result of this analysis is shown in Figure 9 which depicts the average number of accepted mutations by algorithm and scenario. It can be observed from this figure that the ES with the BlockCopy operator is clearly superior, finding significantly more cost-improving mutations than the other ESes. In fact for one of the scenarios (ks1), the number of acceptable mutations found is four times that of the ES with random perturbation.

VI. Evaluation: BlockCopy Mutation and Crossover for (5,10)-ES

Conventional wisdom suggests that a (1+1)-ES is prone to becoming trapped in local optima. Beyond changing the mutation operator, one way to overcome local optima issues is to employ a population-based strategy.

To that end, in addition to a (1+1)-ES we also implemented a (5,10)-ES for solving the wind farm layout optimisation problem. Two variants of the (5,10)-ES were evaluated: in the first variant, the BlockCopy crossover operator was the sole operator. Parents in the intermediate generation (i.e. the 10 selected layouts) were randomly paired up to produce offspring layouts. In the second variant of the (5,10)-ES, only BlockCopy mutation was employed and offspring layouts were generated from a single parent.

The results are depicted in Figure 10-13. Each plot shows the (1+1)-ES results from the first evaluation on the left-hand side, and two sets of results (for BlockCopy crossover and mutation respectively) on the right-hand side respectively.

Unfortunately, the population-based approach fails generally to improve on the (1+1)-ES algorithm. In all cases, the median best costs achieved are either considerably worse than the (1+1)-ES or they overlap and there is no significant difference.

One interesting case is the ks2 scenario (Figure 11) in which the lowest cost obtained by the (5,10)-ES with BlockCopy

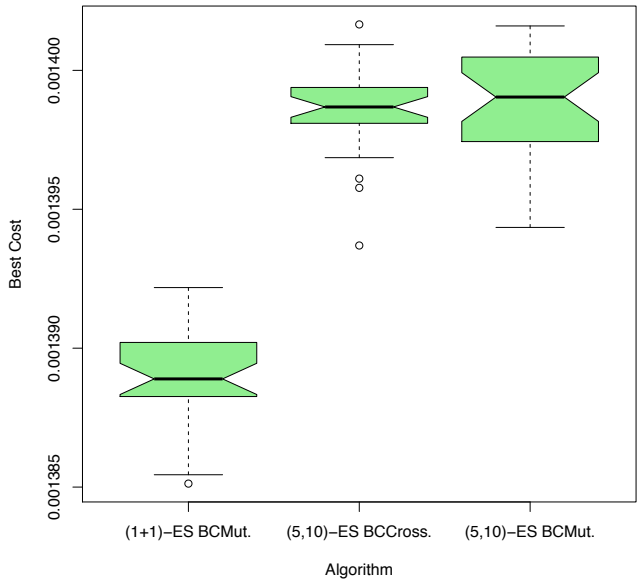


Fig. 10: Comparision of best result in Figure 5 with performance of a (5,10)-ES using either BlockCopy mutation or crossover on the Kusiak & Song 1 [9] wind scenario.

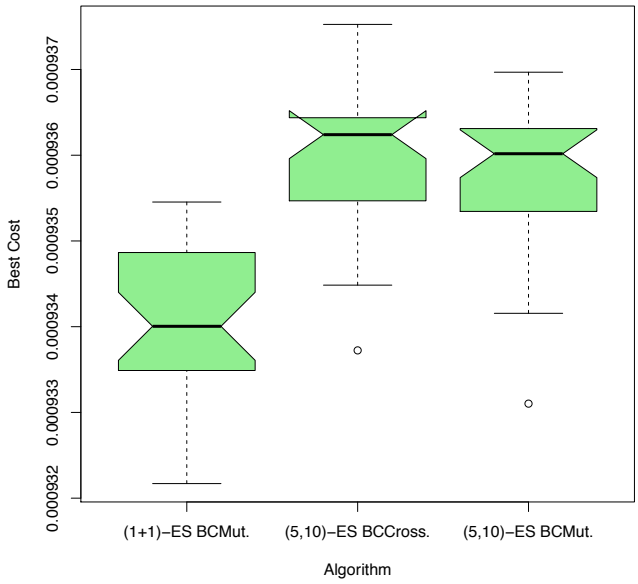


Fig. 12: Comparision of best result in Figure 7 with performance of a (5,10)-ES using either BlockCopy mutation or crossover on the 2014 Comp 1 [23].

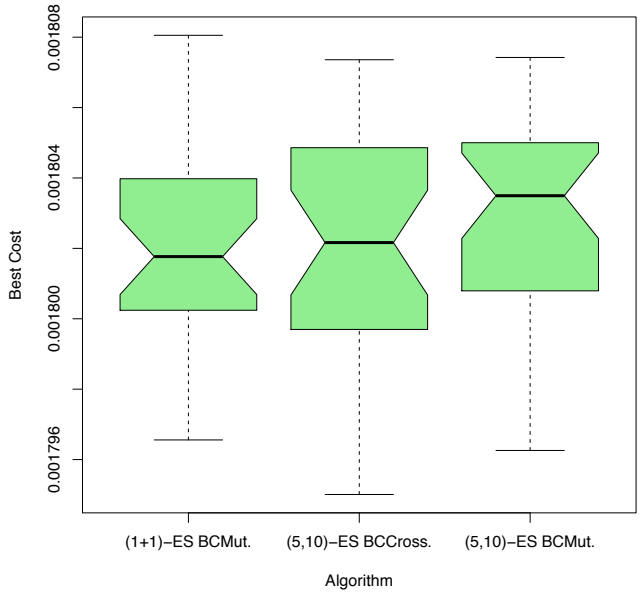


Fig. 11: Comparision of best result in Figure 6 with performance of a (5,10)-ES using either BlockCopy mutation or crossover on the Kusiak & Song 2 [9] wind scenario.

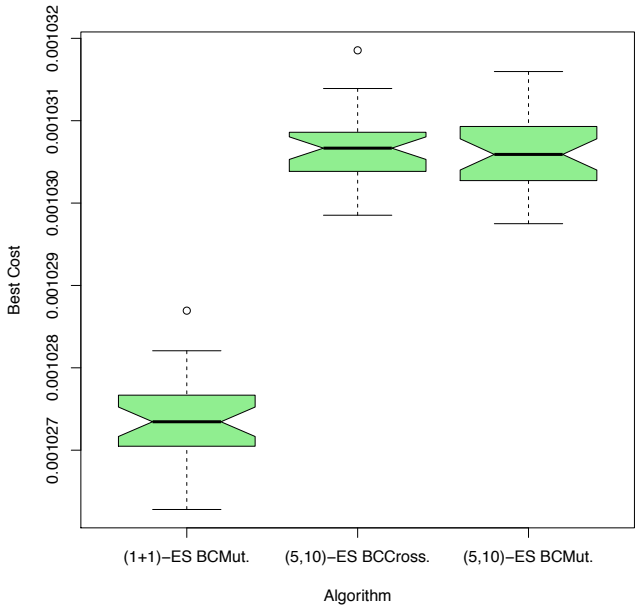


Fig. 13: Comparision of best result in Figure 8 with performance of a (5,10)-ES using either BlockCopy mutation or crossover on the 2014 Comp 3 [23].

Crossover is smaller than any cost achieved by the (1+1)-ES algorithm. However, this is not a general phenomenon and may be due to chance.

The likely reason for the poor performance of the (5,10)-ES is that using a population-based strategy increases exploration at the expense of exploitation. Due to the limited number of evaluations (2,000), the population-based strategies are unable to achieve the same degree of solution quality as the (1+1)-ES does. It may be argued that increasing the number of evaluations would address this problem. However, given the expense of the evaluation function (which could be much greater for a higher-fidelity simulation) and the fact that many other works in the literature use a similar number of evaluations, we did not increase the number of evaluations given to the (5,10)-ES [1].

VII. Conclusion

To conclude, this paper has described a novel search operator for the WFLO problem. It can be used in either a mutation or a crossover context, and have shown in our experiments that the operator is highly effective when compared to other common approaches to solving the same problem.

Our current and ongoing work in this area concerns developing further techniques to improve the performance of BlockCopy-based search. In particular, we would like to increase the number of evaluations without increasing runtimes significantly by employing surrogate models [8]. Such an approach should enable both single solution-based approaches and population-based approaches to improve, if an accurate enough surrogate model for this problem can be identified.

References

[1] S. Al-Yahyai, Y. Charabi, and A. Gastli. Geometrical approach for wind farm symmetrical layout design optimization. In *GCC Conference and Exhibition (GCCCE), 2015 IEEE 8th*, pages 1–6, 2015.

[2] D. Atef, H. Osman, M. Ibrahim, and K. Nassar. A simulation-based planning system for wind turbine construction. In *Proceedings of the Winter Simulation Conference*, WSC '10, pages 3283–3294, 2010.

[3] H. Beyer and H. Schwefel. Evolution strategies: A comprehensive introduction. *Journal Natural Computing*, 1(1):3–52, 2002.

[4] L. Chen and E. MacDonald. Considering landowner participation in wind farm layout optimization. *Journal of Mechanical Design*, 134(8):084506–084506, 07 2012.

[5] Global Wind Energy Council. *Global Wind Energy Outlook 2014*. 2014.

[6] J. S. González, Á. G. Rodríguez, J. C. Mora, M. B. Payán, and J. R. Santos. Overall design optimization of wind farms. *Renewable Energy*, 36(7):1973 – 1982, 2011.

[7] J. F. Herbert-Acero, O. Probst, P.-E. Réthoré, G. C. Larsen, and K. K. Castillo-Villar. A review of methodological approaches for the design and optimization of wind farms. *Energies*, 7(11):6930, 2014.

[8] Y. Jin. Surrogate-assisted evolutionary computation: recent advances and future challenges. *Swarm and Evolutionary Computation*, 1:61–70, 2011.

[9] A. Kusiak and Z. Song. Design of wind farm layout for maximum wind energy capture. *Renewable Energy*, 35:685–694, 2010.

[10] W. Kwong, P. Yun Zhang, D. Romero, J. Moran, M. Morgenroth, and C. Amon. Multi-objective wind farm layout optimization considering energy generation and noise propagation with NSGA-II. *Journal of Mechanical Design*, 136(9):091010–091010, 07 2014.

[11] W. Kwong, P. Y. Zhang, and D. Romero. Wind farm layout optimization considering energy generation and noise propagation. In *Proceedings of the ASME 2012 International Design Engineering Technical Conferences & Computers and Information in Engineering Conference IDETC/CIE 2012*, 2012.

[1] We note that we also tried a (5+10)-ES with little difference in performance

[12] D. Lückehe, M. Wagner, and O. Kramer. On evolutionary approaches to wind turbine placement with geo-constraints. In *Proceedings of the 2015 on Genetic and Evolutionary Computation Conference*, GECCO '15, 2015.

[13] S. Luke. *Essentials of Metaheuristics*. Online version 2.1 edition, 2014.

[14] M. Mayo and M. Daoud. An adaptive model-based mutation operator for the wind farm layout optimisation problem. In *Proc. IEEE Conference on Systems, Man and Cybernetics*, 2015.

[15] G. Mosetti, C. Poloni, and B. Diviacco. Optimization of wind turbine positioning in large wind farms by means of a genetic algorithm. *Journal of Wind Engineering and Industrial Aerodynamics*, 51(1):105–116, 1994.

[16] A. Neubert, A. Shah, and W. Schlez. Maximum yield from symmetrical wind farm layouts. In *10th German Wind Energy Conference (DEWEK)*, 2010.

[17] R. Preen and L. Bull. Torward the coevolution of novel vertical-axis wind turbines. *IEEE Transactions on Evolutionary Computation*, 19(2):284 – 294, 2015.

[18] R. Razavi-Far and M. Saif. Imputation of missing data for diagnosing sensor faults in a wind turbine. In *Proc. IEEE Conference on Systems, Man and Cybernetics*, 2015.

[19] M. Samorani. The wind farm layout optimization problem. In P. Pardolas, editor, *Handbook of Wind Power Systems*, pages 21–38. Springer-Verlag, 2013.

[20] K. Veeramachaneni, A. Cuesta-Infante, and U.-M. O'Reilly. Copula graphical models for wind resource estimation. In *Proc. International Joint Conference on Artificial Intelligence*, pages 2646–2654, 2015.

[21] M. Wagner, J. Day, and F. Neumann. A fast and effective local search algorithm for optimizing the placement of wind turbines. *Renewable Energy*, 51:64–70, 2013.

[22] L. Wang, A. C. Tan, Y. Gu, and J. Yuan. A new constraint handling method for wind farm layout optimization with lands owned by different owners. *Renewable Energy*, 83:151–161, 2015.

[23] D. Wilson. http://www.irit.fr/wind-competition/. URL, 2015.

[24] D. Wilson, S. Cussat-Blanc, K. Veeramachaneni, U. O'Reilly, and H. Luga. A continuous development model for wind farm layout optimization. In *Proceedings of the 2014 Conference on Genetic and Evolutionary Computation*, GECCO '14, pages 745–752, New York, NY, USA, 2014. ACM.