# Artificial Ecosystem Algorithm Applied to Multi-Line Steel Scheduling

Manal T. Adham
University College London
Computer Science
m.adham@ucl.ac.uk

Peter J. Bentley
University College London
Computer Science
peter.bentley@ucl.ac.uk

***Abstract*—Steel production scheduling is recognised as a major global industry with some of the most difficult industrial logistical problems. Galvanised steel can be used as a raw material for the automotive or construction industries. This paper focuses on scheduling four lines involved in the production of galvanised steel. Specifically, we consider lines in ArcelorMittal's manufacturing plant. ArcelorMittal is one of the largest steel producers in the world. The lines considered include the following: (a) Pickling Line, (b) Tandem Mill Line, (c) Hot Dip Galvanizing Line 1, and (d) Hot Dip Galvanizing Line 2. We apply four variants of the Artificial Ecosystem Algorithm to the multi-line steel scheduling problem. In addition, we compare their performance against several alternative solutions including Simulated Annealing, Tabu Search, Hill Climbing, Branch and Bound, Monte Carlo Tree Search, Genetic Algorithm, Cuckoo Search and Particle Swarm Optimisation.**

***Index Terms*—scheduling, manufacturing, optimisation, nature-inspired, real-world**

## I. Introduction

The primary goal in steel scheduling is to identify the sequence whereby steel coils can be processed through lines with minimum cost and within a limited time frame. The sequences must respect a large number of predefined cost functions and constraints, which consider the evolution of both coil characteristics and line settings simultaneously.

Finding coil sequences with minimum cost is critical to ensuring a smooth transition between coils, avoiding lost coil strips due to bad quality and breakages in the line, and reducing incidents in the manufacturing plant. Moreover, identifying coil sequences within a limited time frame is necessary, as lines can only process one coil at a time and continuously process coils in the yard.

We consider four lines involved in the production of galvanised steel. Each line abides by constraints and cost functions. In the Pickling Line, coils are washed in an acid bath to remove surface impurities. In the Tandem Mill Line, coils pass through rolls which reduce thickness and increase strength. In the Hot Dip Galvanizing Line 1 and Hot Dip Galvanizing Line 2, steel is coated with zinc as a form of protection from moisture. The two galvanizing lines are set to different configurations. The selection of which galvanizing line the coil needs to be processed by corresponds to production order specification.

Each steel coil corresponds to a production order which specifies processing settings such as speed, thickness and temperature. Steel is stored in the form of coils in yards. When these coils are ready to be processed, they are uncoiled and welded together (the head of one coil is the tail of another) before entering a line; thus, the input to each line is a continuous strip.

The optimisation goal is twofold. (a) Find a coil sequence with minimum cost. The machine settings of each line gradually change; therefore coils in a sequence must have similar properties to facilitate smooth transitions between coils. (b) The sequence must be generated in less than two minutes. Coils are continuously processed through lines, making it important to identify sequences within a reasonable time.

## II. Related Work

Research on steel scheduling can be grouped into the following streams: Operations research techniques, which use mathematical programming approaches. These guarantee optimality, but become slow as the problem grows in size and complexity, and the interdependence of different scheduling functions makes the process of developing mathematical approaches difficult [1].

Artificial intelligence techniques, including Ant Colony Optimisation [2] and Particle Swarm Optimisation [3]. They cannot guarantee solution quality, but they are able to find solutions effectively [4].

Puchinger and Raidl ([5]) classify research efforts which combine these streams as either collaborative combinations, where exact and heuristic approaches are intertwined, sequential or parallel, or integrative combinations, where one technique is embedded in the other. This paper considers the first option, in which a combination of approaches collaborate in parallel within the Artificial Ecosystem Algorithm to tackle the multi-line steel scheduling problem. In addition, a number of alternative solutions have been described and implemented in Section IV.

## III. Proposed Approach

Four variants of the Artificial Ecosystem Algorithm were applied and empirically evaluated against alternative solutions (described in Section IV). A more detailed description of the aforementioned approaches can be found in [6]. All the algorithms considered have been evaluated both as separate

978-1-7281-2153-6/19/$31.00 ©2019 IEEE

entities and with the decomposition techniques (described in Section III-E).

The utility function used to evaluate the quality of a complete solution sequence is defined in Eqn 2 and the fitness function is defined in Eqn 1.

### A. Artificial Ecosystem Algorithm 1

We present a holistic solution to the multi-line steel scheduling problem by applying the Artificial Ecosystem Algorithm (AEA1) [7, 8]. The ecosystem-inspired phenomena harnessed include: environment, population, homeostasis, evolution, nice and diversity. In AEA1, the problem (environment) is divided into subproblems (niches), and each subproblem is tackled using a subpopulation. An individual in a population corresponds to the transition between two coils. The population is formed by randomly creating complete solutions, which are subsequently split into solution building blocks (individuals). Tournament selection [9] is used to assemble individuals to form a solution. This involves: (1) Selecting t individuals from a population, (2) copying the best-fit individual to an intermediate population, and (3) repeating n times.

Individuals with low fitness values are removed using the REMOVEINDIVIDUALS procedure, and new individuals are created to replace them using the REPLACEINDIVIDUALS procedure. In the REPLACEINDIVIDUALS procedure, a portion of individuals are recreated using the CROSSOVER procedure and others are recreated randomly. CROSSOVER focuses on producing building blocks that are close to reaching optimality, while replacing individuals randomly allows the algorithm to jump out of local optima and move towards the global optimum by exploring the solution space. Edges not contained within the parents and perhaps not within the population can be introduced into the created individuals. As duplicate individuals are permitted, a BALANCEFITNESS procedure was devised to allow duplicates to maintain uniform fitness.

The subsolutions formed are connected based on the width constraint, which specifies that the difference in width between two consecutive coils must be less than a predefined threshold. The ability to satisfy this constraint has a strong influence on the quality of the solution. Thus a complete solution is formed using a combination of subsolutions obtained from different subpopulations. AEA1 is presented in Algorithm 1.

### B. Artificial Ecosystem Algorithm 2

The parallelism of AEA1 is further developed by enabling it to simultaneously build multiple subsolutions for each subproblem as opposed to developing just one solution at a time. The subsolution with the greatest utility is selected from each subproblem and connected to form a complete solution. In this way, multiple solutions are evolved simultaneously.

### C. Artificial Ecosystem Algorithm 3

In the original AEA1 formulation, a population of individuals was initialised by randomly generating a complete solution sequence, which was split into equally sized building blocks. This implies that it is possible to have duplicate individuals

**Algorithm 1** Artificial Ecosystem Algorithm 1

```
Initialise: Environment, Population, Iteration = 1
repeat
    Select individual i_i at random
    repeat
        Use tournament selection to select a subsequent individuals i_j
    until Candidate solution (C_i) formed
    if U(C_i) < U(C_best) then
        C_best = C_i
    Update F1(j_i) for all individuals in (C_i)
    REMOVEINDIVIDUALS
    REPLACEINDIVIDUALS
    BALANCEFITNESS
    Iteration++
until Stopping Criteria Satisfied
```

in one population. AEA3 reformulates the representation of a population and an individual to frame it as a matrix:

$$P = \begin{vmatrix} & A & B & C \\ A & \infty & 5 & 1 \\ B & 4 & \infty & 8 \\ C & 2 & 9 & \infty \end{vmatrix}$$

Each letter corresponds to a coil, and the matrix shows the cost of each potential coil combination. Each number represents an individual building block, and the matrix represents a population. A set of the widest coils is identified, and used to randomly select a start point. An example starting point is the transition from Coil $A$ to Coil $B$. The algorithm then starts to build a solution $X_i$ using tournament-based selection, considering all the transition costs from Coil $B$, which have not been visited previously. Once $X_i$ represents a complete solution, the best solution obtained so far is updated if $X_i$ has a lower cost. These steps are repeated until the stopping criterion is met: in this case, a predefined maximum number of iterations. Each iteration builds one solution; future variants of AEA3 could build multiple solutions simultaneously in one iteration in a manner similar to that used in AEA2.

### D. Artificial Ecosystem Algorithm 4

Artificial Ecosystem Algorithm 4 mimics the diversity of ecosystems by allowing subproblems to employ different strategies. There is no one algorithm that can be applied to effectively tackle all problems [10]. In AEA4, the problem is first decomposed to form subproblems, and a collection of different strategies are applied simultaneously for each subproblem. The complementary strengths of a range of strategies are exploited to solve a task. These strategies include mathematical optimisation, nature-inspired and local search.

The best result found for each subproblem is then used to construct the solution (the lowest cost with respect to Eqn 2). Therefore, a complete solution is generated using a combination of subsolutions created using different strategies. In this way, AEA4 acts as an ensemble which holds

autonomous species that operate in a decentralised fashion, thereby imitating the relative autonomy of species in an ecosystem.

Currently, each subproblem executes all the approaches considered in Section IV simultaneously. To date, ensembles which apply optimisation algorithms have received little attention in the literature [11].

### E. Decomposition Strategies

Using decomposition allows us to capitalise on scaling enabled by dividing the problem space as opposed to the solution space [12]. A partitioning strategy is implemented, which divides the problem (set of coils) into subproblems using coil characteristics. These characteristics vary according to each line considered: they include width, thickness, strength and zinc coating. Three decomposition strategies are used. Domain-based strategy ([13]): this is essentially a sorting mechanism which separates coils into a predefined number of groups. $k$-modes [14]: $k$ initial points are selected for each cluster; then, coils are allocated to clusters by identifying the nearest mode and the mode of each centroid is updated. Self-Organising Map [15] allows the mapping from high dimensional space to low dimensional space, whilst preserving the topological structure. Neurons represent nodes of a 2-dimensional lattice, then training and mapping phases are applied for a predefined number of iterations. Each neuron in the lattice is considered as a cluster; if the number of items in the cluster is below a predefined threshold then it is merged with its nearest cluster.

### F. Fitness Function

Each individual is associated with a fitness value, which reflects its ability to collaborate with other individuals to form an effective solution. This value is calculated by using the weighted function defined in Eqn 1.

$$F1(j_i) = w_1 F_{ni} + w_2 j_{ti} + w_3 j_{ci} \tag{1}$$

where $F1(j_i)$ = the fitness value for job $j_i$, $F_{ni}$ = number of coils visited in the sequence, $j_{ti}$ = transition cost of $j_i$, $j_{ci}$ = cost of the solution sequence, and $w_1$, $w_2$ and $w_3$ are weights.

### G. Evaluation Measures

The goal is twofold: (a) minimise sequence cost and thereby maximise productivity, (b) build a solution in less than two minutes. The cost of the solution is defined in Eqn 2. The cost functions and weights were defined by ArcelorMittal, thereby enabling us to perform a direct comparison.

$$c(x) = CF + w_1 S + w_2 H \tag{2}$$

where $CF$ = sum of cost functions for a given line, $S$ = sum of soft constraints violated for a given line, $H$ = sum of hard constraints violated for a given line, and $w_1$ and $w_2$ are the weights associated with the constraints violated.

Typically, hard constraints are fixed conditions that must be satisfied. However in practice, it can be difficult to build sequences that do not violate hard constraints. In an effort to deal with this problem, ArcelorMittal accounts for hard constraints by adding a substantially higher cost to sequences which violate them.

## IV. Alternative Solutions

This section describes alternative solutions implemented to tackle the multi-line steel scheduling problem, thereby allowing us to gauge how well the Artificial Ecosystem Algorithm variants are able to perform against solutions present in the literature. The objective function used to determine the solution quality is specified in Eqn 2, in which a lower weighted cost entails a higher fitness value. The stopping criterion used is a predefined maximum number of iterations. A more detailed account of all the following algorithms is provided in [6].

Branch and Bound [16]: In our implementation an initial greedy solution is generated by selecting coils with the lowest costs present in the matrix. Nodes represent coils and edges represent the transition between coils. The algorithm selects a node $k$ and children of $k$ are generated with their corresponding lower bounds. To help speed up the algorithm multiple branches are explored simultaneously using threads and a timer is added to stop the algorithm from running for long periods of time.

Monte Carlo Tree Search [17]: In our implementation each iteration begins at the root and descends the tree by using a tree policy until a leaf node is reached. The tree policy selects the root node with the highest reward. A simulation is executed from the selected node, and the score is back-propagated and used by the tree policy in subsequent iterations.

Genetic Algorithm [18]: Candidate solutions represent complete solutions initialised randomly. Fitness-proportionate selection is used, making the probability of selecting a candidate solution inversely proportional to the objective function. A two-point crossover operator is used to generate offspring for the next generation, and a random swap between two coils in a candidate solution is used for mutation.

Cuckoo Search [19]: We randomly select a cuckoo and replace its solution via Lévy flights, as per the original implementation. This solution is then evaluated to identify its fitness, and a nest is chosen randomly. If the new solution improves the quality of eggs in the nest, then a randomly selected solution in the nest is replaced with the newly generated solution. Some of the worst nests are then abandoned, and new solutions are built.

Particle Swarm Optimisation [20]: We randomly initialise particles, set their best known position as their current position, and update the swarm's best known position accordingly. The velocity is used to update the position of each particle. If the particles fitness is higher, then its best position is updated. The global fitness is also updated if it has been improved.

Tabu Search [21]: The algorithm monitors the search and reacts to cycling and repetition of solutions by adapting the tabu list size and using long-term memory to promote diversification across the search space.

TABLE I
THE NUMBER OF COILS IN EACH TEST CASE

| | Test Cases | | |
|---|---|---|---|
| | **1** | **2** | **3** |
| **Pickling Line** | 90 | 90 | 83 |
| **Tandem Mill Line** | 68 | 87 | 57 |
| **Hot Dip Galvanizing 1 Line** | 68 | 64 | 75 |
| **Hot Dip Galvanizing 2 Line** | 80 | 81 | 90 |

Simulated Annealing [22, 23]: The algorithm begins by initialising a random solution $x$, then at each iteration 3-opt is applied to modify the solution and generate a new solution $y$. Next, an acceptance probability is used to decide whether or not to move to the modified solution $y$. As the search progresses and the solution space is explored, there is a slow decrease in the probability of accepting worse solutions.

Hill Climbing Algorithm [24]: This algorithm starts with a single, randomly generated solution; it then randomly modifies the coil sequence whilst maintaining the best solution so far. This is performed until the stopping criterion is met.

All of the aforementioned approaches are applied in the following three ways: on their own without decomposition, with each of the decomposition strategies described in Section III-E, and as strategies to solve subproblems in AEA4.

## V. BENCHMARK TEST CASES

Three test cases were provided by ArcelorMittal for each of the four lines considered.

1) Test case 1: a typical batch taken from production history.
2) Test case 2: a batch designed to mimic situations expected when working with lower stock levels, resulting in a limited range of coils available. This test case is harder to schedule as there are fewer options for finding compatible coils.
3) Test case 3: another batch similar to test case 1, taken from production history.

## VI. EXPERIMENTS

The performance of the four Artificial Ecosystem Algorithm variants are compared against the alternative approaches using benchmark test cases described in Section V.

1) Experiment 1 - evaluates performance against the three test cases.
2) Experiment 2 - evaluates performance on the four lines, specifically the Pickling Line, Tandem Mill Line, Hot Dip Galvanising Line 1 and Hot Dip Galvanising line 2. In addition, the overall performance across lines and test cases is also considered.

### A. Performance Metrics

Two metrics are used to evaluate the performance of the proposed and alternative approaches. 1) Solution quality: the goal is to minimise the cost of the coil sequence produced. This is the cost in Eqn 2, which considers cost functions, soft constraints and hard constraints. 2) Computational effort: the solution must be generated in 2 minutes or less.

The error bars in the following experiments represent the standard deviation with two significant figures. Results are also compared against ArcelorMittal's current solution, Ant Colony Optimisation (ACO) [2]. The only metric available for ACO is solution quality. Data on time is not available, although it was communicated that ACO was given 2 minutes to find a solution.

TABLE II
SETTINGS FOR THE ARTIFICIAL ECOSYSTEM ALGORITHM.

| **Parameter** | **Setting** |
|---|---|
| MaxGen | 500 |
| Turnover | 10% |
| Xover | 20% |
| PopulationSize | 50 |
| BalanceFitnessLimit | 20 |
| Solutions built (AEA2) | 10 |

TABLE III
SETTINGS FOR MATHEMATICAL PROGRAMMING APPROACHES

| **Branch and Bound** | | **Monte Carlo Tree Search** | |
|---|---|---|---|
| Threads | 6 | Number of actions | 5 |
| Timer | 30 seconds | | |

### B. General Experimental Setup

To ensure reliability 50 independent runs were conducted for each experiment. All the following approaches were developed using a combination of Java and Python. More implementation details can be found in [6]. Experiments were performed on the University College London computer cluster, which consists of 628 nodes and 4370 cores, but only the specified resources are allocated to a task. The specification for each task is: 2 symmetric multiprocessing, 7G virtual memory and 2 threads. Each task represents one test case and there are three test cases for each line.

The algorithm settings were identified through preliminary tests and are specified in Tables II to V. Parameters were selected using a trial and error approach. As a starting point we used default parameter settings, then a range of values were investigated to determine a final setting. This investigation was performed by selecting three problems on which to test the algorithms. For each problem, the algorithms were executed 10 times using the different configurations selected. The best, average and worst solutions generated were then compared against the best known solution and the best configuration was used to obtain the final setting. To enable a fair comparison, the same tuning process was used to derive each parameter.

TABLE IV
SETTINGS FOR NATURE-INSPIRED APPROACHES

| **Genetic Algorithm** | | **Particle Swarm Optimisation** | | **Cuckoo Search** | |
|---|---|---|---|---|---|
| MaxGen | 500 | Maximum epochs | 300 | MaxGen | 500 |
| Mutation Rate | 0.025 | Particle count | 100 | Nests | 20 |
| Tournament size | 5 | Maximum velocity | 10 | Abandon probability | 0.01 |

TABLE V
SETTINGS FOR LOCAL SEARCH APPROACHES

| Simulated Annealing | | Tabu Search | | Hill Climber | |
|---|---|---|---|---|---|
| Temperature | 10000 | MaxGen | 500 | MaxGen | 1000 |
| Cooling rate | 0.003 | Tabu length | 30 | | |

### C. Experiment 1 - Test Case Comparison

This experiment compares the performance of the Artificial Ecosystem Algorithm against the alternative approaches on the benchmark test cases.

*1) Performance Comparison:* Figures 1 and 2 demonstrates the performance of the algorithms considered for all three test cases using Self-Organising Map as it achieved the most promising results from preliminary experiments. More detailed results can be found in [6]. Test case 1 demonstrates the most favourable results, followed by test case 3 and then 2. This was anticipated as test case 2 models a more constrained situation.

Figure 1 shows that AEA4 is competitive with Ant Colony Optimisation. It is able to improve on Ant Colony Optimisation's results for test case 1 by 34.35%, though Ant Colony Optimisation still outperforms AEA4 in test case 2 by 14.12% and outperforms test case 3 by 27.11%. The error lines for AEA4 represent consistency, thereby demonstrating the reliability of the results attained. This is a promising result, as Ant Colony Optimisation has been optimised by ArcelorMittal to operate in the manufacturing plant.

We can also observe that Monte Carlo Tree Search and Hill Climbing Search struggle to identify solutions effectively. This is understandable, as Monte Carlo Tree Search was originally developed to determine the best moves in a game [17]. The results for Hill Climbing Search were anticipated, as it is a local search method, which aims to find better solutions through incremental changes. An effective way to improve Hill Climbing Search is by using heuristics to identify promising start points [25]. Nevertheless, neither of these approaches were designed to find solutions to complex, real-world problems. The Simulated Annealing approach taken here is relatively simple. A more advanced and possibly adaptive perturbation can be used in place of 3-opt to improve its performance [23]. Nahar *et al.* ([26]) have shown that perturbation functions have a substantial impact on Simulated Annealing's performance.

Notably, Tabu Search performs relatively well compared to the other local search approaches, which can be attributed to its adaptive nature and its ability to diversify in order to escape from local optima. Furthermore, the Genetic Algorithm also performs well compared to the other nature-inspired approaches considered. The results generated for this algorithm are similar to that generated by Branch and Bound, whilst the solution is generated in almost half the time.

Figure 2 demonstrates the time consumed by all the approaches considered. The figure shows that Monte Carlo Tree Search consumes the most time. In addition to not producing solutions effectively, Monte Carlo Tree Search exceeds the two-minute time constraint. The remaining solutions were found in under two minutes and therefore satisfy the runtime constraint. Again, AEA4 takes a relatively long time, as performance depends on the speed of the strategies implemented. The trade-off between solution quality and time is clear in the case of AEA4. This was anticipated, as AEA4 runs multiple strategies simultaneously. Fine-tuning the strategies so that only the most effective solutions are considered for a given problem could reduce this time without compromising savings in cost.

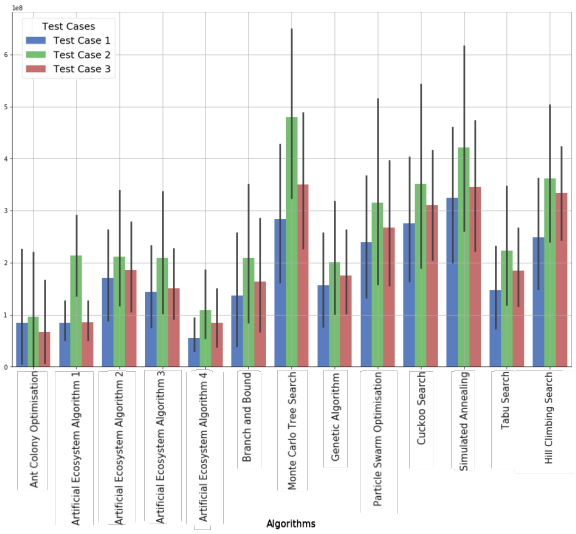


Fig. 1. Average cost for solutions generated for the different test cases considered.

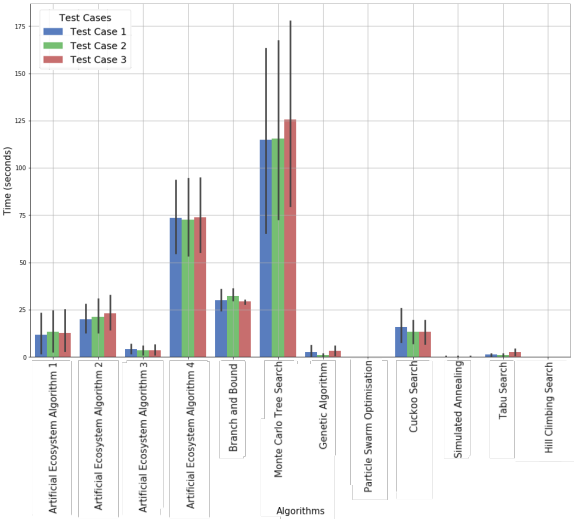


Fig. 2. Average time consumed to identify a solution for each test case.

### D. Experiment 2 - Compare Performance on Lines

This experiment compares the performance of the different approaches on different lines; pickling (pkl), tandem mill (tdm), hot dip galvanizing 1 (hdg1) and hot dip galvanizing 2 (hdg2). The results presented are an average of all three test cases.

*1) Performance Comparison:* Figures 3 and 4 demonstrate the average weighted cost and time consumed to find a solution for the four lines considered. Self-Organising Map was used for all of these algorithms, as it achieved the most promising results in Experiment 1. Complete details of this experiment can be found in [6]. In addition, the solutions implemented are compared against Ant Colony Optimisation. In Figure 3, AEA4 is again seen to be competitive with Ant Colony Optimisation. AEA4 reduces the cost associated with the tdm line by 20.89%, although it can be seen that Ant Colony Optimisation still generates lower costs for pkl by 71%, for hdg1 by 77.62% and for hdg2 by 149%.

AEA4 comes at the expense of a higher computational cost than other approaches presented in Figure 4 (with the exception of Monte Carlo Tree Search). Nonetheless, the time consumed is still within two minutes. The figure also demonstrates that tdm requires the least time to find a solution, followed by the pkl, hdg1 and then hdg2. However, solutions generated for tdm carry a relatively high cost compared to the other lines, due to the extend of cost functions and constraints violated. Allowing the algorithms to run on the tdm line for a longer period of time could potentially allow a better solution to be generated.

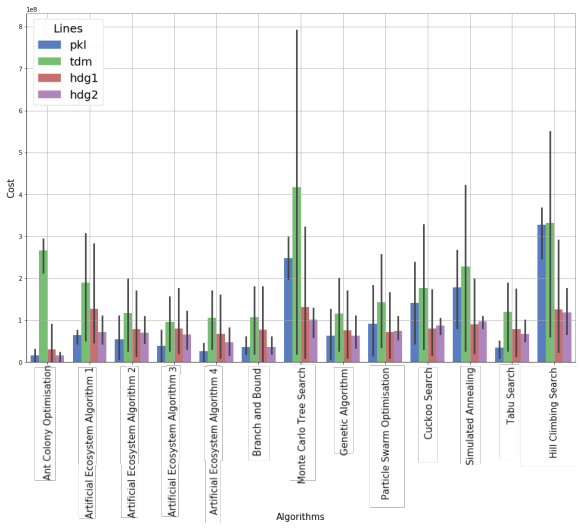


Fig. 3. Average cost for solutions generated for the different test cases considered.

### E. Overall Analysis

Figures 5 and 6 demonstrates the overall performance of all the approaches considered across all the lines using

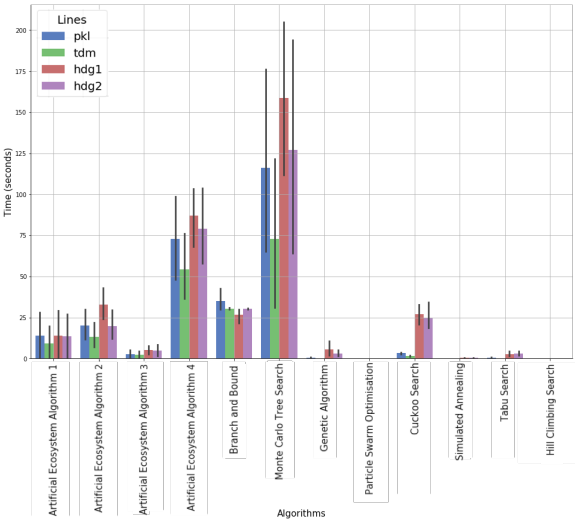


Fig. 4. Average time consumed to identify a solution for each test case.

Self-Organising Map. The results demonstrate the continuous improvement of the Artificial Ecosystem Algorithms as we evolve the algorithm. AEA2 was able to improve on AEA1 by 41.04%, AEA3 was able to improve on AEA2 by 14.28%, and AEA4 was able to improve on AEA3 by 14.44%. Allowing multiple solutions to evolve simultaneously in AEA2 permits it to identify a solution with higher utility more quickly than AEA1. In addition, all the approaches considered, with the exception of Monte Carlo Tree Search, satisfy the two-minute time limit.

Results also verify AEA4's ability to match an optimised variant of Ant Colony Optimisation. AEA4 was able to reduce Ant Colony Optimisation's overall cost by 25.61%. This is mainly attributed to its performance in the tandem mill line, where it was able to save substantially on cost. AEA4 also outperformed Branch and Bound by 4%, Tabu Search by 5%, and Genetic Algorithm by 22%. Although these are small margins it is still a substantial gain especially in a real-world application. This was enabled by allowing AEA4 to implement different strategies to solve the subproblems. Within AEA4, each line considered used different strategies for the subprob lems encountered.

### F. Conclusion

This paper further developed the Artificial Ecosystem Algorithm and applied four variants of it to the multi-line steel scheduling problem. We also developed and applied eight alternative solutions to the multi-line steel scheduling problem. Considering alternative solutions allowed us to gauge how well the Artificial Ecosystem Algorithm performs in relation to solutions present in the literature.

All the approaches were evaluated using three benchmark test cases based on two quality metrics: solution quality and

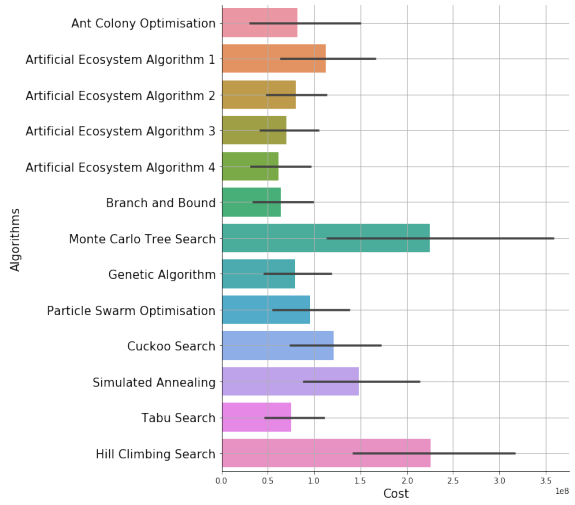


Fig. 5. Average cost for solutions generated for the different test cases considered.

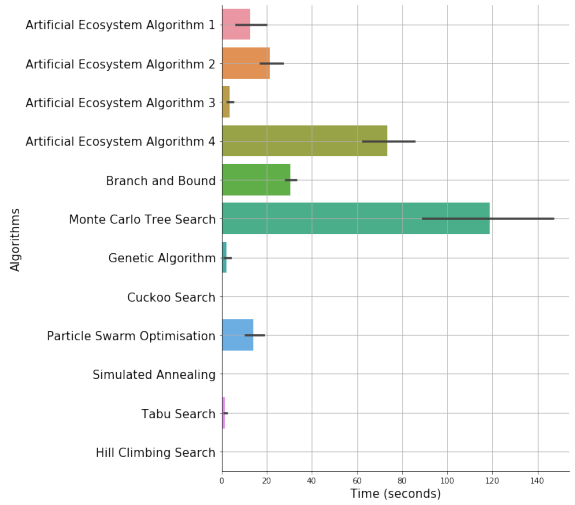


Fig. 6. Average time consumed to identify a solution for each test case.

computational effort. In addition, the solutions implemented were compared against the current solution implemented by ArcelorMittal, Ant Colony Optimisation. As the cost functions and weights used were those defined by ArcelorMittal, we were able to perform a direct comparison. Moreover, all the approaches considered were tested as separate entities and with three different decomposition strategies including Self-Organising Map, $k$-modes and domain-specific. Results indicated that Self-Organising Map was effective in obtaining high-quality solutions, demonstrating the most promising results.

All the results demonstrated a continuous improvement of the Artificial Ecosystem Algorithms as we improved the algorithm from AEA1 to AEA4. Furthermore, results showed that AEA4 using Self-Organising Map outperformed Ant Colony Optimisation on test case 1 results by 34.35% (Experiment 2) and outperformed Ant Colony Optimisation for the Tandem Mill Line by 20.89% (Experiment 2). Moreover, on average, and across all the lines, AEA4 outperformed Ant Colony Optimisation by 25.61%. This is a substantial finding, as Ant Colony Optimisation is the current solution implemented in a real-world manufacturing plant. The success of AEA4 can be attributed to the diversity of its approach, suggesting that this is another useful ecosystem-inspired feature to incorporate within our algorithm. In future work, the goal would be to consider the most suitable set of strategies for a particular problem instance.

## Acknowledgment

The authors would like to thank ArcelorMittal for their help and cooperation in this project.

## References

[1] Geoff Buxey. “Production scheduling: Practice and theory”. In: *European Journal of Operational Research* 39.1 (1989), pp. 17–31.

[2] Silvino Fernandez et al. “Scheduling a galvanizing line by ant colony optimization”. In: *International Conference on Swarm Intelligence*. Springer. 2014, pp. 146–157.

[3] Tsung-Lieh Lin et al. “An efficient job-shop scheduling algorithm based on particle swarm optimization”. In: *Expert Systems with Applications* 37.3 (2010), pp. 2629–2636.

[4] Karla L Hoffman. “Combinatorial optimization: Current successes and directions for the future”. In: *Journal of computational and applied mathematics* 124.1 (2000), pp. 341–360.

[5] Jakob Puchinger and Günther R Raidl. “Combining metaheuristics and exact algorithms in combinatorial optimization: A survey and classification”. In: *International Work-Conference on the Interplay Between Natural and Artificial Computation*. Springer. 2005, pp. 41–53.

[6] Manal Tarek Adham. “A Decomposition-Based Ecosystem-Inspired Approach For Solving Real-World Logistics Problems”. PhD thesis. UCL (University College London), 2019.

[7] Manal T. Adham and Peter J. Bentley. “An Ecosystem Algorithm for the Dynamic Redistribution of Bicycles in London”. In: *the proceedings of the 10th International Conference on Information Processing in Cells and Tissues*. IPCAT 15. San Diego, CA, USA: Springer International Publishing, 2015, pp. 39–51. ISBN: 978-3-319-23108-2. DOI: 10.1007/978-3-319-23108-2_4. URL: http://dx.doi.org/10.1007/978-3-319-23108-2_4.

[8] Manal T. Adham and Peter J. Bentley. “An Artificial Ecosystem Algorithm Applied to Static and Dynamic Travelling Salesman Problems”. In: *2014 IEEE Symposium Series on Computational Intelligence (SSCI)*. IEEE. 2014, pp. 149–156. ISBN: 978-1-4799-4479-8. DOI: 10.1109/ICES.2014.7008734.

[9] Tobias Blickle and Lothar Thiele. “A Mathematical Analysis of Tournament Selection.” In: *ICGA*. Citeseer. 1995, pp. 9–16.

[10] David H Wolpert and William G Macready. “No free lunch theorems for optimization”. In: *IEEE transactions on evolutionary computation* 1.1 (1997), pp. 67–82.

[11] Emma Hart and Kevin Sim. “On constructing ensembles for combinatorial optimisation”. In: *Evolutionary computation* 26.1 (2018), pp. 67–87.

[12] Christine L Valenzuela and Antonia J Jones. “Evolutionary divide and conquer (I): A novel genetic approach to the TSP”. In: *Evolutionary Computation* 1.4 (1993), pp. 313–333.

[13] Iiro Harjunkoski and Ignacio E Grossmann. “A decomposition approach for the scheduling of a steel plant production”. In: *Computers & Chemical Engineering* 25.11-12 (2001), pp. 1647–1660.

[14] Zhexue Huang. “Clustering large data sets with mixed numeric and categorical values”. In: *Proceedings of the 1st pacific-asia conference on knowledge discovery and data mining,(PAKDD)*. Singapore. 1997, pp. 21–34.

[15] Teuvo Kohonen. “Self-organized formation of topologically correct feature maps”. In: *Biological cybernetics* 43.1 (1982), pp. 59–69.

[16] Ailsa H Land and Alison G Doig. “An automatic method of solving discrete programming problems”. In: *Econometrica: Journal of the Econometric Society* (1960), pp. 497–520.

[17] Diego Perez, Philipp Rohlfshagen, and Simon M Lucas. “Monte-Carlo tree search for the physical travelling salesman problem”. In: *European Conference on the Applications of Evolutionary Computation*. Springer. 2012, pp. 255–264.

[18] David E Goldberg and Jon Richardson. “Genetic algorithms with sharing for multimodal function optimization”. In: *Genetic algorithms and their applications: Proceedings of the Second International Conference on Genetic Algorithms*. Hillsdale, NJ: Lawrence Erlbaum. 1987, pp. 41–49.

[19] Xin-She Yang and Suash Deb. “Engineering optimisation by cuckoo search”. In: *International Journal of Mathematical Modelling and Numerical Optimisation* 1.4 (2010), pp. 330–343.

[20] Russ C Eberhart and James Kennedy. “A new optimizer using particle swarm theory”. In: *Proceedings of the sixth international symposium on micro machine and human science*. Vol. 1. New York, NY. 1995, pp. 39–43.

[21] Roberto Battiti and Giampietro Tecchiolli. “The reactive tabu search”. In: *ORSA journal on computing* 6.2 (1994), pp. 126–140.

[22] Scott Kirkpatrick, C Daniel Gelatt, Mario P Vecchi, et al. “Optimization by simulated annealing”. In: *science* 220.4598 (1983), pp. 671–680.

[23] Attahiru Sule Alfa, Sundresh S Heragu, and Mingyuan Chen. “A 3-opt based simulated annealing algorithm for vehicle routing problems”. In: *Computers & Industrial Engineering* 21.1-4 (1991), pp. 635–639.

[24] Fabio Romeo. “Probabilistic Hill Climbing Algorithm: Properties and Applications”. In: *Chapel Hill Conference on VLSI, 1985*. 1985.

[25] Craig A Tovey. “Hill climbing with multiple local optima”. In: *SIAM Journal on Algebraic Discrete Methods* 6.3 (1985), pp. 384–393.

[26] Surendra Nahar, Sartaj Sahni, and Eugene Shragowitz. “Simulated annealing and combinatorial optimization”. In: *Proceedings of the 23rd ACM/IEEE design automation conference*. IEEE Press. 1986, pp. 293–299.