Multi-Objective Optimization of a Real-World Manufacturing Process Using Cuckoo Search

Anna Syberfeldt

Abstract This chapter describes the application of Cuckoo Search in simulation-based optimization of a real-world manufacturing process. The optimization problem is a combinatorial problem of setting 56 unique decision variables in a way that maximizes utilization of machines and at the same time minimizes tied-up capital. As in most real-world problems, the two optimization objectives are conflicting and improving performance on one of them deteriorates performance of the other. To handle the conflicting objectives, the original Cuckoo Search algorithm is extended based on the concepts of multi-objective Pareto-optimization.

Keywords Cuckoo search · Simulation-based optimization · Multi-objective problem

1 Introduction

Discrete-event simulation combined with optimization, so called simulation-based optimization, is a powerful method to improve real-world systems [1]. While traditional, analytical optimization methods have been unable to cope with the challenges imposed by many simulation-based optimization problems in an efficient way, such as multimodality, non-separability and high dimensionality, so called metaheuristic algorithms have been shown to be applicable to this type of problem [2][3]. Metaheuristic algorithms are powerful stochastic search algorithms with mechanisms inspired from natural science. A metaheuristic algorithm optimizes a problem by iteratively improving one or several candidate solution(s) with regard to a given objective function. The algorithms are not guaranteed to find the optimal solution for the given problem, but are instead computationally fast and make no explicit assumptions about the underlying structure of the function to be optimized. These

A. Syberfeldt (✉)
University of Skövde, PO. 408, SE-54148 Skövde, Sweden
e-mail: anna.syberfeldt@his.se

X.-S. Yang (ed.), *Cuckoo Search and Firefly Algorithm*,
Studies in Computational Intelligence 516, DOI: 10.1007/978-3-319-02141-6_9,

properties make them well suited for simulation-based optimization as the simulation is often time consuming and can be seen as a black-box [4].

There exist plenty of metaheuristic algorithms, among the most well-known including simulated annealing, tabu search, and genetic algorithms. One of the most recently proposed is called Cuckoo Search, which is inspired by the parasitic breeding behavior of cuckoos [5, 6]. Several studies indicate that Cuckoo Search is a powerful algorithm and successful results have been achieved in various applications such as welded beam design, nurse scheduling and wireless sensor networks [7–9]. A review of the literature, however, reveals no applications in manufacturing optimization and the aim of this study is therefore to investigate the algorithm's performance in this domain.

A real-world problem of engine manufacturing at a Volvo factory in Sweden is focus of the study. In short, the problem is about finding the best prioritization of the different engine components being simultaneously processed in a manufacturing line. Basically, this is a combinatorial problem involving the setting of 56 unique decision variables. The prioritization is used to determine which specific component has precedence when two or more components are available at a machine or station. Based on the priorities, it is possible to create a schedule showing which component should be processed in which machine at each point in time. However, finding an efficient processing schedule is not trivial, due to the considerable complexity of the cell in combination with a fluctuating inflow and resource constraints. This fact has raised a need to perform automatic simulation-optimizations and has motivated an evaluation of different algorithms for this purpose. To perform simulation-optimizations, a discrete-event simulation of the model is constructed by simulation experts at Volvo using the SIMUL8 software. The next section of this chapter presents the optimization in further detail.

2 Real-World Manufacturing Optimization

In Volvo's manufacturing line under study, engine components of eleven different types are being processed. The line includes ten different processing stations that can perform single or multiple operations (including for example burring, welding, and assembly). The operations performed at a specific station and the tools used vary depending on the type of component. The high degree of variety in processing of different components, in combination with a fluctuating inflow and several resource constraints, make it virtually impossible to plan the production manually in an efficient way. Automatic optimizations are instead needed, and for this purpose a discrete-event simulation model of the line has been developed by simulation experts at the company using the SIMUL8 software package.[1] The simulation model has a front-end interface developed in Excel, which facilitates the user in entering input parameters to the model without the need to learn the simulation language. Valid-

[1] www.simul8.com

ity tests indicate that the simulation model represents reality well, and the model is generally accepted among operators working in the line.

The optimization problem to be solved using the simulation model consists of constructing a processing schedule for the manufacturing line by setting 56 unique decision variables. These variables basically represent the order of operations for different components and also which component has precedence in case of queues in front of machines. According to the company, a processing schedule should preferably be configured so that a high utilization of all machines is achieved, as these involve expensive investments. The company has also stated that it is important to reduce tied-up capital by minimizing each component's lead time (that is, the time between a component entering and exiting the line). Altogether there are two objectives that must be considered simultaneously in the optimization process: (a) utilization, and (b) tied-up capital.

For high utilization, a large number of components within the line is needed in order to avoid machine starvation. However, a large number of components imply occasional queues in front of some machines, which results in longer lead-times for components and thereby tied-up capital. In relation to each other, the two objectives of maximal utilization and minimal tied-up capital are therefore conflicting. No single optimal solution with respect to both objectives exists, as improving performance on one objective deteriorates performance of the other objective.

One way to handle conflicting objectives is to derive a set of alternative trade-offs, so called Pareto-optimal solutions [10]. Figure 1 illustrates the Pareto concept for a minimisation problem with two objectives f_1 and f_2. In this example, solution *A*-*D* are non-dominated, i.e. Pareto optimal, since for each of these solutions there exist no other solution that is superior in one objective without being worse in another objective. Solution *E* is dominated by *B* and *C* (but not by *A* or *D*, since *E* is better than these two in f_1 and f_2, respectively).

Although it is important to find as many (trade-off) optimal solutions as possible in multi-objective optimisation, the user needs only one solution regardless of the number of objectives [10]. Which of the optimal solutions to choose is up to the user to decide based on previous experiences and qualitative information (for example, ergonomic conditions or set-up of factory workers for the day).

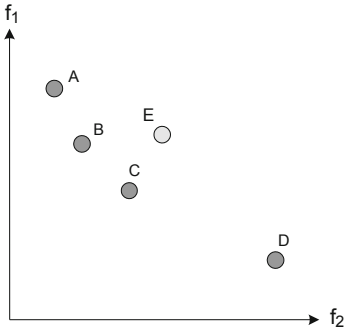

Fig. 1 Illustration of dominance

3 Cuckoo Search

This section presents the optimization of the manufacturing line using the Cuckoo Search algorithm.

3.1 Algorithm Description

In nature, cuckoos lay eggs in other birds' nests and rely on those other birds to rear the cuckoo's offspring. According to the so called "selfish gene theory" proposed by Dawkins in 1989 [11], this parasitic behavior increases the chance of survival of the cuckoo's genes since the cuckoo needs not expend any energy rearing its offspring. Instead, the cuckoos can spend more time on breeding and thereby increasing the population. However, the birds whose nests are invaded have developed counter strategies and increasingly sophisticated ways of detecting the invading eggs.

These behaviors found in nature are utilized in the Cuckoo Search algorithm in order to traverse the search space and find optimal solutions. A set of nests with one "egg" (candidate solution) inside are placed at random locations in the search space. A number of "cuckoos" traverse the search space and records the highest objective values for different encountered candidate solutions. The cuckoos utilize a search pattern called Lévy flight which is encountered in real birds, insects, grazing animals and fish according to Viswanathan et al. [12]. The Lévy flight is characterized by a variable step size punctuated by 90-degree turns, as can be seen in Fig. 2. The large steps occasionally taken make the algorithm suitable for global search. Lévy flights, according to Yang [6] and Gutowski [13], are more efficient for searching than regular random walks or Brownian motions.

Lévy flights are used in the Cuckoo Search algorithm to globally explore the search space, while local random walks are used for exploitation. A switching parameter p_a is used to balance between exploration and exploitation. Eq. 1 describes how the

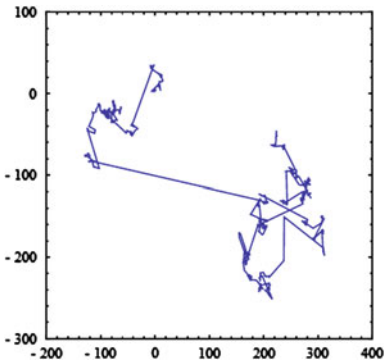

Fig. 2 Example of Lévy flight starting at [0,0]

local walks are performed.

$$x_i^{(t+1)} = x_i^{(t)} + \alpha s \oplus H(p_a - \varepsilon) \oplus \left(x_j^{(t)} - x_k^{(t)}\right) \tag{1}$$

In the equation, x_j and x_k are two different solutions randomly selected, is H a Heaviside function, is a random uniform number, and is the step size. The global random search using the concept of Lévy flights are described in Eq. 2.

$$x_i^{(t+1)} = x_i^{(t)} + \alpha L\acute{e}vy(s, \lambda) \tag{2}$$

In the equation, $\alpha > 0$ represents the step size scaling factor, and this must be fine-tuned for the specific problem at hand. Yang [6] advocates the use of α as $\alpha = O(L/100)$, where L represents the difference between the maximum and minimum valid value of the given problem. In order to implement the Lévy flight, a fast algorithm needed to be used to approximate the Lévy distribution. Leccardi [14] compared three different approaches to generating Lévy distributed values and found that the algorithm published in [14] proved to be the most efficient. Mantegna's algorithm is divided into three steps in order to generate the step length, in Eq. 3 needed for the Lévy flight.

$$s = \frac{u}{|\nu|^{\frac{1}{\beta}}} \tag{3}$$

The parameters u and ν are given by the normal distributions in Eq. 4.

$$u = N(0, \sigma_u^2), \qquad \nu = N(0, \sigma_\nu^2) \tag{4}$$

The variance, σ, is calculated as Eq. 5 with $1 \leq \beta \leq 2$ and where $\Gamma(z)$ is the gamma function.

$$\sigma_u = \left\{ \frac{\Gamma(1+\beta)\sin(\pi\beta/2)}{\Gamma[(1+\beta)/2]\beta 2^{(\beta-1)/2}} \right\}^{\frac{1}{\beta}}, \quad \sigma_\nu = 1 \tag{5}$$

Cuckoo Search is a population based, elitist, single-objective optimization algorithm. The pseudo code for the algorithm is presented in Fig. 3 [5]. Note that only two parameters need to be supplied to the algorithm; the discovery rate $p_a \in [0, 1]$ and the size of the population, n. When n is fixed, p_a controls the elitism and the balance of randomization and local search [6]. The fact that the algorithm comes with just two parameters does not only increase the ease of implementation, but also potentially makes it a more general optimization solution to be applied to a wide range of problems.

The real-world problem to be optimized by Cuckoo Search in this study involves the optimization of two objectives. The next section of this chapter describes how the algorithm is adjusted to consider more than one objective.

```
Generate initial population of n host nests
while (t < MaxGeneration or stopCriterionFulfilled)
    Generate a solution by Lévy flights and then evaluate its quality/fitness F_i
    Choose a nest among n (say, j) randomly
    if (F_i > F_j),
        Replace j by the new solution
    end
    Abandon the p_a nests with worst quality and build new ones
    Keep best solutions/nests
    Rank the solutions/nest and find the current best
    Pass the current best to the next generation.
end while
```

Fig. 3 Pseudo code for Cuckoo Search with Lévy flight

3.2 Multi-Objective Extension

In order to convert Cuckoo Search from a single-objective optimization algorithm into multi-objective optimization algorithm, concepts from an existing Pareto-based algorithm is used, namely elitist non-dominated sorting genetic algorithm (NSGA-II). NSGA-II is a widely used multi-objective algorithm that has been recognized for its great performance and is often used for benchmarking [16]. The pseudo-code for NSGA-II can be seen in Fig. 4 and for details about the algorithm the reader is referred to [17].

In the NSGA-II algorithm, the selection of solutions is done from a set R, which is the union of a parent population and an offspring population (both of size N) [18]. Non-dominated sorting is applied to R and the next generation of the population is formed by selecting solutions from one of the fronts at a time. The selection starts with solutions in the best Pareto front, then continues with solutions in the second best front, and so on, until N solutions have been selected. If there are more solutions in the last front than there are remaining to be selected, niching is applied to determine which solutions should be chosen. In other words, the highest ranked solutions located in the least crowded areas are the ones chosen. All the remaining solutions are discarded. The selection procedure is illustrated in Fig. 5 (adopted from [10]).

The concept of non-dominated sorting utilized in the NSGA-II algorithm has been used to create a multi-objective extension of Cuckoo Search (a similar approach is presented in [19]). Besides the non-dominated sorting procedure, other differences between NSGA-II and the new version of the Cuckoo Search algorithm include the use Lévy flights instead of mutation and the abandonment and movement of nests instead of crossover. The pseudo code for the multi-objective version of Cuckoo Search is presented in Fig. 6.

The next section of this chapter presents results from applying the multi-objective version of Cuckoo Search on the real-world manufacturing problem.

```
Initialize Population
Generate N random solutions and insert into Population
for (i = 1 to MaxGenerations) do
    Generate ChildPopulation of size N
    Select Parents from Population
    Create Children from Parents
    Mutate Children
    Combine Population and ChildPopulations into CurrentPopulation with size
     2N
    for each individual in CurrentPopulation do
        Assign rank based on Pareto – Fast non-dominated sort
    end for
    Generate sets of non-dominated vectors along PF known
    Loop (inside) by adding solutions to next generation of Population starting
     from the best front
        until N solutions found and determine crowding distance between
         points on each front
end for
Present results
```

Fig. 4 Pseudo code for NSGA-II

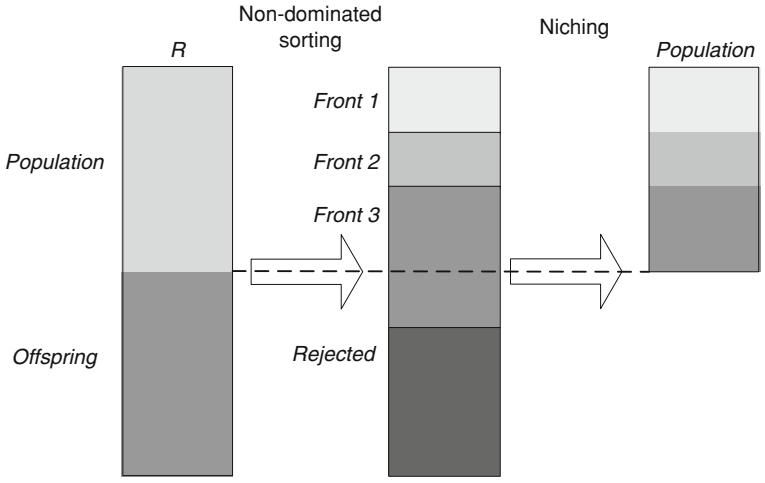

Fig. 5 Non-dominated sorting

```
Initialize Population
Generate N random solutions and insert into Population
for (i = 1 to MaxGenerations) do
    Generate ChildPopulation using Cuckoo Search (pseudo code in Fig. 3)
    Non-dominated-sort
    for each individual in CurrentPopulation do
        Generate sets of non-dominated vectors along PF_known
        Loop (inside) by adding solutions to next generation starting from the
        best front
            until N solutions found and determine crowding distance between
              points on each front
    end for
end for
Present results
```

Fig. 6 Pseudo code for the proposed multi-objective Cuckoo Search

4 Evaluation

4.1 Configuration

As previously mentioned in Sect. 2 of this chapter, the real-world manufacturing problem is basically a combinatorial problem with 56 unique decision variables representing a processing schedule. The proposed multi-objective extension of Cuckoo Search is applied on the problem with the maximum number of simulation evaluations set to 1000 (according to the time budget stated by the company). The population size is set to 20, and the percentage of abandoned nests to 40% (the values have been found by trial-and-error tests). The algorithm uses a swap operator utilizing the Lévy flights to determine the indices to swap. The step size for the Lévy flight is set to $\frac{upperbound}{100}$, where upperbound is the maximum permitted value of each parameter.

For comparison, the NSGA-II algorithm is also applied on the problem. NSGA-II is run with the same number of simulation evaluations as Cuckoo Search and implemented with the same population size. NSGA-II uses swap range mutation with a mutation probability of 10%. Furthermore, the algorithm is implemented with a partially mapped crossover operator and a crossover probability of 90%.

As baseline for the comparison between Cuckoo Search and NSGA-II, a basic scheduling function defined by the company is being used. In this function, a Critical Ratio (CR) value of each component is calculated to determine how well it is on schedule. The CR value is derived by dividing the time to due date (i.e. scheduled completion) by the time expected to finish the component, according to Eq. 6.

$$CR = \begin{cases} due \geq now : \frac{1+due-now}{1+TRPT} \\ due < now : \frac{1}{(1+now-due)*(1+TRPT)} \end{cases} \quad (6)$$

In this equation, *due* is the due date of the component (i.e. deadline), *now* is the current time, and *TRPT* is the theoretical total remaining processing time (the active operation time including set-up times and movements between machines/stations). A component with a CR value of 1.0 is "according to schedule", while it is behind if the value is less than 1.0 and ahead if the value is larger than 1.0. In case of a race condition, the component with the lowest CR value has precedence.

4.2 Integrating the Optimization Algorithm and the Simulation

To run the algorithms along with the SIMUL8 simulation model, a component called "simulation controller" is implemented. The simulation controller, whose purpose is to start the simulation and collect results, is implemented using the application programming interface (API) provided by SIMUL8. This API enables the simulation model to be controlled programmatically. When the simulation controller receives a request (i.e., a set of priority values to be evaluated) from the optimization algorithm, the simulation controller invokes the simulation model with a "Run" command and provides the priority values. When the simulation has been completed (after about two seconds), the simulation controller collects the results and sends them back to the optimization algorithm.

4.3 User Interface

The user interacts with the simulation-optimization through a web page displayed in a web browser. Using a web page as user interface enables access to the platform from any hardware (e.g., PC, mobile phone, tablet, etc.) at any site, as long as a web browser and an internet connection are available. This is especially advantageous in industrial environments, since employees often have limited (or no) possibilities of installing, running, and maintaining software at their work stations.

The content of the web page was developed during an iterative process undertaken in close cooperation between the company and the University's software developers. A screenshot of one part of the resulting web page is presented in Fig. 7 (for integrity reasons, company specific information was deleted before the screenshot was taken). The panel to the left in the screenshot constitutes the menu from which the user accesses various functions in the platform. In the screenshot, the menu alternative "RESULTS" have been chosen. This alternative shows the output from a simulation-optimization run, namely, a production schedule derived from a prioritization of all components to be processed during a specific time period. The aim of the production schedule is to support the operators of the manufacturing cell by specifying which component should be processed in which machine at each point in time.

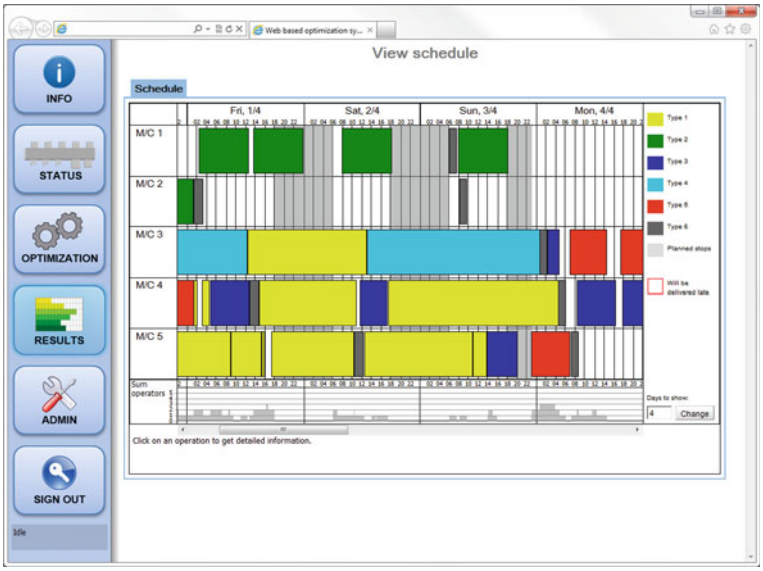

Fig. 7 Screenshot of user interface

4.4 Results

The optimization results achieved by Cuckoo Search and NSGA-II are presented in Table 1 and stated in relation to the CR function. More specifically, the percentage numbers given in the table represent the relative improvement achieved by each algorithm when compared to the result of the CR function. No real numbers can be presented due to company restrictions.

As shown in Table 1, the results clearly indicate that NSGA-II outperforms Cuckoo Search considering both optimization objectives. An analysis of this result is given in the next section of this chapter.

5 Analysis

This section presents an analysis of the results, both from a technical point of view and from a user perspective.

Table 1 Results with CR function as baseline

	Utilization (improvement)	Tied-up capital (improvement)
Cuckoo Search (%)	10	15
NSGA-II (%)	19	23

Average of 20 replications

5.1 Technical Analysis

As mentioned in the introduction, Cuckoo Search has proven to be able to efficiently solve different optimization problems in various domains. Nevertheless the algorithm shows a weak performance on the real-world manufacturing problem of focus for this paper. When going through previous studies on Cuckoo Search and analyzing the nature of the optimization problems solved in these papers, it is clear that continuous problems have been the main target for Cuckoo Search so far. In this study, the problem to be solved is combinatorial and a hypothesis is this fact is the reason behind the results. To investigate the hypothesis, five continuous problems from the well-known ZDT test suit are implemented. A detailed description of the ZDT problems is provided in [20].

Results from applying the multi-objective version of Cuckoo Search on the continuous ZDT problems are presented in Table 3. For comparison, the NSGA-II algorithm has also been applied on the same problems. Two performance measures are being used: convergence and spread (both are to be minimized). These two are commonly used for the ZDT problems and a detailed explanation of their implementation can be found in [10].

As can be seen in Table 2 Cuckoo Search achieves the best results on the ZDT1, ZDT2 and ZDT4 problems. On the ZDT3 problem, it achieves a better convergence than the NSGA-II algorithm, but a worse spread. On the ZDT6 problem, NSGA-II performs the best on both objectives.

The results from the test suit show that Cuckoo Search outperforms NSGA-II on the majority of the ZDT problems, which is probably due to use of the efficient Lévy flight method. These results are in line with results from previous studies found in the literature on applying Cuckoo Search on continuous optimization problems.

To investigate further if Cuckoo Search is better suited for continuous optimization problems, the algorithm is also evaluated on a combinatorial test problem. The problem implemented is the symmetrical traveling salesman problem named berlin52

Table 2 Results from ZDT problems

		Convergence (minimize)	Spread (minimize)
ZDT1	NSGA-II	0.037582	0.613663
	Cuckoo Search	0.002561	0.613206
ZDT2	NSGA-II	0.030753	0.684501
	Cuckoo Search	0.003855	0.649924
ZDT3	NSGA-II	0.001460	0.615773
	Cuckoo Search	0.001149	0.616611
ZDT4	NSGA-II	0.075159	0.700399
	Cuckoo Search	0.000437	0.623540
ZDT6	NSGA-II	0.002848	0.616384
	Cuckoo Search	0.085899	0.675200

Average of 1000 replications

Table 3 Results from combinatorial TSP problem

	Distance (minimize)
Cuckoo Search	21716
NSGA-II	10961

Average of 1000 replications

[21], which was chosen due to the similar number of parameters to the real-world problem. The objective of the problem is to find the shortest route between a number of points, while only going through each point once, and then returning to the starting point. The distance between the points is measured through Euclidean distance. Results from applying Cuckoo Search and NSGA-II on the problem are presented in Table 3. As shown in the table, NSGA-II clearly outperforms the Cuckoo Search algorithm on the combinatorial TSP problem. It should be noted that there exist a specialized Cuckoo Search tailor-made for the travelling salesman problem that has been proven to perform very well [22], but as this version is single-objective it has not been considered in the study.

Although a more extensive evaluation including a larger number of optimization problems are needed for a general statement, the results obtained in this study indicates that Cuckoo Search in the form implemented in this study is suited for continuous problems rather than combinatorial ones. A probable reason for the weak performance on combinatorial problems is that the Lévy flight pattern is not suited to be used as a basis for swap mutation. As previously described, the algorithm uses a swap operator utilizing the Lévy flights to determine the indices to swap.

5.2 User Perspective

The simulation-optimization has also been analyzed by a test group at Volvo Aero. This test group included eight employees who represented all three user groups supported by the platform (operator, shift leader, and manager), as well as logistics engineers at the company. University representatives first demonstrated the simulation-optimization on site at the company, after which it was made available to the test group. They tried the system out for a couple of weeks, during which their feedback was collected and compiled.

After the test period, the evaluation feedback was discussed and analyzed. The feedback highlighted the great potential the simulation-optimization demonstrated with regard to improving processing schedules, especially during periods of heavy workloads. Besides improving the processing schedules, using the platform was also considered a way of significantly reducing the human effort currently associated with creating the schedules. A further advantage raised by the test group was the possibility of easily sharing optimized production schedules in real-time among stakeholders. Since the platform is web-based, it allows all users to easily view

results simultaneously, which stands in contrast to traditional desktop applications where results must be printed or e-mailed.

A negative aspect of the platform raised, more or less, by everyone in the test group was the need to manually specify the current status and configuration of the manufacturing cell before each simulation-optimization. The required data includes shift period, status of components currently under process, list of components entering, availability of fixtures, availability of operators, and scheduled maintenance. The simulation model needs all this data to be able to mimic the operations of the cell as close to reality as possible. Collecting and specifying the data can be time-consuming and should preferably be eliminated so that the optimization can be fully automatic and run frequently, or even constantly. To realize this, the platform can be integrated with the computerized control system of the manufacturing cell. It is essential to strive for such integration, which would probably not be too difficult considering the manufacturing cell has modern, built-in control systems that process the data needed.

6 Conclusions

This paper describes a case study of applying Cuckoo Search, a recently proposed metaheuristic algorithm, in simulation-based optimization of a real-world manufacturing line. The manufacturing line is highly automated and produces engine components of eleven different types. The Cuckoo Search algorithm has previously shown promising results in various problem domains, which motives to evaluate it also on the optimization problem under study in this paper. The optimization problem is a combinatorial problem of setting 56 unique decision variables in a way that maximizes utilization of machines and at the same time minimizes tied-up capital. As in most real-world problems, these two objectives are conflicting and improving performance on one of them deteriorates performance of the other. To handle the conflicting objectives, the original Cuckoo Search algorithm is extended based on the concepts of multi-objective Pareto-optimization. We argue that a multi-objective version of Cuckoo Search is needed not only for this specific study, but for Cuckoo Search to be truly useful in general as most real-world problems involve the optimization of more than one objective.

Optimization of the manufacturing line is performed based on a discrete-event simulation model constructed using the SIMUL8 software. Results from the simulation-based optimization show that the extended Cuckoo Search algorithm is inefficient in comparison with the multi-objective benchmark algorithm NSGA-II. A possible reason might be that the Cuckoo Search algorithm is not suited for combinatorial optimization problems due to that the Lévy flight pattern is not suited to be used as a basis for swap mutation. To investigate this further, the algorithm is applied on five continuous test problems and also one combinatorial test problem. Results from these test problems shows that the Cuckoo Search algorithm outperforms NSGA-II on a majority of the continuous test problems. However, the Cuckoo

Search algorithm performs considerable worse than the NSGA-II algorithm on the combinatorial test problem. An explanation for the weak results on this problem, as well as the real-world problem, might be that the Lévy flight pattern is not suited to be used as a basis for swap mutation. Further evaluations are needed, however, to investigate this circumstance further. It is also recommended to study other possible ways to adapt Cuckoo Search to combinatorial optimization problems possibly suggest an improved version that is able to handle this type of problem effectively.

Another interesting improvement of the Cuckoo Search algorithm would be to consider simulation randomness. To capture the stochastic behavior of most real-world complex systems, simulations contain randomness. Instead of modeling only a deterministic path of how the system evolves in the process of time, a stochastic simulation deals with several possible paths based on random variables in the model. To tackle the problem of randomness of output samples is crucial because the normal path of the algorithm would be severely disturbed if estimates of the objective function come from only one simulation replication. The common technique to handle randomness is to send the algorithm with the average values of output samples obtained from a large number of replications. Although this technique is easy to implement, the large number of replications needed to obtain statistically confident estimates from simulation of a complex system that requires long computation time can easily render the approach to be totally impractical. There are methods that guarantee to choose the "best" among a set of solutions with a certain statistically significance level, which require fewer replications in comparison to the others (e.g. Kim-Nelson ranking and selection method found in [23]). However, combining statistically-meaningful procedures that require relatively light computational burden with metheuristics is still an important topic for further research [24].

Besides technical issues, it is also important to carefully considering user aspects for a successful realization of the optimization. During the Volvo case study, it was clear that the graphical design and layout of the user interface was an important factor in gaining user acceptance of the simulation-optimization. In general, software development more easily focuses on algorithms and technical details, but putting at least as great an effort into the graphical design is recommended. It is especially important to keep usability in mind when developing systems for industrial adoption, since the success of a system is dependent on its acceptance by its users [25].

References

1. April, J., Better, M. Glover, F., Kelly, J.: New advances for marrying simulation and optimization. In: Proceedings of the 2004 Winter Simulation Conference, Washington, DC, pp. 80–86 (2004)
2. Boesel, J., Bowden, R., Glover, F., Kelly, J., Westwig, E.: Future of simulation optimization. In: Proceedings of the 2001 Winter Simulation Conference, Arlington, VA, pp. 1466–1470 (2001)
3. Laguna, M., Marti, R.: Optimization Software Class Libraries. Kluwer Academic Publishers, Boston (2003)

4. Bäck, T., Fogel, D., Michalewicz, Z.: Handbook of Evolutionary Computation. Oxford University Press, Oxford (1997)
5. Yang, X.-S., Deb, S.: Cuckoo Search via Levy Flights. In: Proceedings of World Congress on Nature and Biologically Inspired Computing, India, pp. 210–214 (2009)
6. Yang, X.-S.: Nature-Inspired Metaheuristic Algorithms, 2nd edn. Luniver Press, Beckington (2010)
7. Yang, X.-S., Deb, S.: Engineering optimisation by Cuckoo Search. Int. J. Math. Model. Numer. Optim. **1**(4), 330–343 (2010)
8. Tein, L.H., Ramli, R.: Recent advancements of nurse scheduling models and a potential path. In: Proceedings of 6th IMT-GT Conference on Mathematics, Statistics and its Applications, pp. 395–409 (2010)
9. Dhivya, M., Sundarambal, M., Anand, L.N.: Energy efficient computation of data fusion in wireless sensor networks using Cuckoo based particle approach (CBPA). Int. J. Commun. Netw. Syst. Sci. **4**(4), 249–255 (2011)
10. Deb, K.: Multi-Objective Optimization Using Evolutionary Algorithms, 2nd edn. Wiley, Chichester (2004)
11. Dawkins, R.: The Selfish Gene, 30th edn. Oxford University Press Inc, New York (1989)
12. Viswanathan, G.M., Buldyrev, S.V., Havlin, S., da Luz, M.G.E., Raposo, E.P., Eugene Stanley, H.: Optimizing the success of random searches. Lett. Nat. **401**, 911–914 (1999)
13. Gutowski, M.: Lévy flights as an underlying mechanism for global optimization algorithms. Optimization **8** (2001)
14. Leccardi, M.: Comparison of three algorithms for Lèvy noise generation. In: Proceedings of Fifth EUROMECH Nonlinear Dynamics Conference, Mini Symposium on Fractional Derivatives and their Applications(2005)
15. Mantegna, R.N.: Fast, accurate algorithm for numerical simulation of Levy stable stochastic processes. Phys. Rev. E **49**, 4677–4683 (1994)
16. Coello Coello, C.A., Lamon, G.B., Van Veldhuizen, D.A.: Evolutionary Algorithms for Solving Multi-objective Problems, 2nd edn. Springer Science Business Media LLC, New York (2007)
17. Deb, K., Pratap, A., Agarwal, S., Meyarivan T.: A fast and Elitist multi-objective genetic algorithm: NSGA-II. IEEE Trans. Evol. Comput. **VI:2**, 182–197 (2002)
18. Deb, K., Pratap, A., Agarwal, S., Meyarivan, T.: A fast and elitist multi-objective genetic algorithm NSGA-II. KanGAL Report 2000001, Indian Institute of Technology Kanpur, India (2000)
19. Yang, X.-S., Deb, S.: Multiobjective cuckoo search for design optimization. Comput. Oper. Res. **40**(6), 1616–1624 (2013)
20. Zitzler, E., Deb, K., Thiele, L.: Comparison of multiobjective evolutionary algorithms: empirical results. Evol. Comput. **8**(2), 173–195 (2000)
21. Reinelt, G. TSPLIB, 8 August, [Online]. http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/(2008)
22. Ouaarab, A., Ahiod, B., Yang, X-S.: Discrete cuckoo search algorithm for the travelling salesman problem. Neural Comput. Appl. (2013)
23. Gosavi, A.: Simulation-based Optimization: Parametric Optimization Techniques and Reinforcement Learning. Kluwer Academic, Boston (2003)
24. Jin, Y., Branke, J.: Evolutionary optimization in uncertain environments—a survey. IEEE Trans. Evol. Comput. **9**(3), 303–317 (2005)
25. Rogers, Y., Sharp, H., Preece, J.: Interaction Design: Beyond Human-Computer Interaction, 3rd edn. Wiley, New York (2007)