Neural Computing and Applications (2021) 33:11125–11155
https://doi.org/10.1007/s00521-020-05657-1

ORIGINAL ARTICLE

# Optimization of decoupling point position using metaheuristic evolutionary algorithms for smart mass customization manufacturing

C. D. James[1] · Sandeep Mondal[2]

Received: 25 April 2020 / Accepted: 22 December 2020 / Published online: 6 January 2021

**Abstract**
In this paper, we present two metaheuristic evolutionary algorithms-based approaches to position the customer order decoupling point (CODP) in smart mass customization (SMC). SMC tries to autonomously mass customize and produce products per customer needs in Industry 4.0. SMC shown here is from the perspective of arriving at a CODP during manufacturing process flow designs meant for fast moving and complex product variants. Learning generally needs several repetitive cycles to break the complexity barrier. We make use of fruit fly and particle swarm optimization (PSO) evolutionary algorithms with the help of MATLAB programming to constantly search better fitting consecutive process modules in manufacturing chain. CODP is optimized by increasing modularity and reducing complexity through evolutionary concept. Learning-based PSO iterations are performed. The methods shown here are recommended for process flow design in a learning-oriented supply chain organization which can involve in-house and outsourced manufacturing steps. Finally, a complexity reduction model is presented which can aid in deploying this concept in design of supply chain and manufacturing flows.

**Keywords** Customer order decoupling point (CODP) · Smart mass customization (SMC) · Evolutionary algorithm (EA) · Optimization · Process flow design · Learning

## 1 Introduction

Today's fast-moving world needs manufacturing and supply chains to be smart with capabilities for quick learning and reconfiguration of process flows. Slow in-house evolution of technical knowhow could mean lost opportunities in a rapidly changing market. Rapid improvements in customer order decoupling point (CODP) positioning would mean quicker standardization and maturity of designed process flows. CODP is the point in the manufacturing process flow which differentiates the standardized production flow from the customized production flows. This would constantly simplify the process for fulfillment of customer needs in a much faster and effective manner without compromising the customization requirements. This paper proposes smart deployment of best fit opportunities in manufacturing process flow design by selecting solutions from a wider set of choices (both external and internal). Best fit opportunities mean easier access to create new custom products without having to start from the concept stage, by exploiting existing modular sub-assemblies which are readily available. And is particularly suited for designing process flow for complex and custom products in smart manufacturing. We also demonstrate some learning models and complexity reduction methods in the end of the paper to facilitate this.

Smart manufacturing or Industry 4.0 is an emerging trend making use of Internet of Things (IoT), cloud-based systems, big data analytics, artificial intelligence-based

**Supplementary Information** The online version contains supplementary material available at https://doi.org/10.1007/s00521-020-05657-1.

✉ Sandeep Mondal
sandeepmondal@iitism.ac.in

C. D. James
JamesDevassia.Cherupillet@infineon.com

[1] Cypress Semiconductor Technology India Pvt Ltd. (An Infineon Technologies Company), 7th Floor, 65/2, Bagmane Tech Park, Block C, Bagmane Laurel, C V Raman Nagar, Bengaluru, Karnataka 560093, India

[2] Department of Management Studies, Indian Institute of Technology (ISM), Dhanbad, Indian School of Mines P.O-ISM, Dhanbad, Jharkhand 826001, India

analytics, machine learning, additive manufacturing and digitalization [1–3]. Hajrizi [4] proposed the usage of modeling, simulation and optimization in Industry 4.0 for multi-objective problem solving and capacity building. Zhang et al. [5] highlighted SMC as a futuristic baseline for successful deployment of smart manufacturing.

Lehmhus et al. [6] discussed about increasing smartness in manufacturing by making use of the cloud-based design and sensor integrated intelligence, with IoT and additive manufacturing. Suginouchi et al. [7] illustrated a co-creative manufacturing system for SMC using smart factory operation method. They proposed an Industrial Internet Consortium for autonomous negotiation mechanism.

Smart mass customization (SMC) is defined as autonomous mass customization and manufacture of products per customer needs in Industry 4.0. Positioning of CODP is important in SMC because it enables the manufacturing system to autonomously design and control manufacturing process flow. Our paper mentions the optimization of CODP position and process flow design through evolutionary algorithms (EA) by smartly joining the best fit steps from in-house and/or external sources, using modularity concept. This is aimed at breaking the complexity barrier in a faster and efficient manner. We explain the EA problem and its solution embedded in a smart manufacturing environment by providing a framework.

In few next paragraphs, we summarize various definitions of customer order decoupling point (CODP), followed by the same for EAs, provided by earlier researchers. Daaboul and Da Cunha [8] defined customer order decoupling point (CODP) as a step in product manufacturing flow which differentiates basic and additional steps that derive new variants in mass customization (MC). They also mention it as the point which splits the overall production line into the build to stock left side portion and the build to order right side portion.

Fogliatto et al. [9] explained CODP as a point which helps manufacturability in MC through postponement of customization. They mentioned it as a point in the upstream of the value chain which gets controlled through modularity, while the downstream being the portion that is customer input based. This helps in efficient order fulfilment by increasing predictability in the supply chain while retaining some control on the customer-driven or custom needs.

Genetic algorithms (GA) are evolution-based EA techniques which perform empirical search optimization as defined by McCall [10]. McCall [10] mentioned that GA are easy to implement in a variety of techniques due to their modular nature. This property of the GA is exploited in our paper to optimize the CODP position by reducing complexity while increasing modularity of the manufacturing process flow. CODP position is determined in the manufacturing process flow through optimization which is achieved with the iterations in the population involved and the fitness function which is made use of. However, we make use of metaheuristic EA techniques, namely fruit fly (FF) and particle swarm optimization (PSO) algorithms, to reduce the time lag of process design by selection of closest solution during the process development. It is different from conventional cross-over chromosomal evolution used in GA method and uses a direct best fit search instead. The EA algorithms used here target and select entities in the supply chain to evolve the best feasible manufacturing process flow. Moving CODP towards right on a process line improves process standardization and execution in the left side of the line. Moving CODP towards the left helps supports flexibility. Hence modularity-based opportunities are deployed to mature the processes and improve standardization while leaving some optimum space based on the flexibility required to break the complexity barrier.

The modular steps in manufacturing process flow could be internal or external. When the internal manufacturing processes are mature, the process may be designed exclusively with in-house processing steps. However, when a firm that leads in a certain manufacturing domain is weaker in certain additional processes required to fulfil newer and complex customer needs, external supplier(s) could be involved. This is a general phenomenon observed during technology advancements and at the arrival of next level of superior product variants. Outsourcing may be allowed intentionally too at times, to reduce the management overhead or cost, even while internal processes exist. Supplier selection is the process of matching a supplier based on performance attributes such as cost, quality, delivery, response, and other services [11].

The next section includes a detailed literature review on CODP and tries to connect EA and SMC. This is followed by problem description and methodology section where we explain the problem statement, modeling assumptions and MATLAB coding for FF- and PSO-based CODP optimization. In the Results and discussion section, we explain the results of optimization achieved through FF and PSO algorithms performed through separate treatments. We studied some learning models and tried to connect the impact of the same to the optimization. We then explain a few conceptual models for modularity search and complexity reduction, which are smart enabled. The models are based on IoT and cloud-based systems for helping in combining internal and external steps for creation of manufacturing process flows. Lastly, we provide conclusion and directions for future research.

## 2 Literature review

In this section, we summarize the previous research combining CODP, Smart Industry 4.0 and EA-based optimization, respectively. We try to connect these three concepts to benefit manufacturing process flow design for SMC. We follow a chronological order while explaining the past research for each of the three areas starting with CODP, followed by EA, and finally Smart manufacturing related to this paper.

Active research on push pull production systems began in the 1980s; however, the term CODP was first seen in the research published by Giesberts and Van der Tang [12]. They provided a formal definition on CODP and explained it as the position between the forecast-driven and customer order-driven portion of manufacturing process chain. Rudberg and Wikner [13] dealt deeper into CODP research by exploring various combinations of engineering and production strategies to provide typologies of CODP for different mass customization (MC) environments. Ethiraj and Levinthal [14] introduced modularity concepts for innovation in managing complex systems, through simulation. Wikner and Rudberg [15] explained engineering and production perspective of CODP in a two-dimensional approach.

Wikner and Wong [16] explored different entities in postponement strategy that map with cases encountered in real-world manufacturing problems. Xu [17] analyzed positioning of CODP from the perspectives of market, product, and production. Hua et al. [18] proposed a MATLAB-based CODP positioning method with the lead time constraint and capacity constraint, with cost minimization objective. Luo et al. [19] showed optimal CODP positioning by considering product functionality, lead time and cost through information entropy and ideal point determination.

Liu et al. [20] discussed about positioning of CODP in leagile supply chain using polychromatic set theory. Ge et al. [21] explained CODP positioning in optimizing the overall cost of supply chain. Ge et al. [21] created separate MATLAB-based models to minimize cost in supply chain while optimizing the CODP using case and simulation data, with constrained lead time, value, productivity and logistics.

Brun and Zorzini [22] evaluated customization strategies through modularity by analyzing complexity of process and product. They coined a term called information decoupling point. Daaboul et al. [23] suggested value network modeling for positioning CODP-based on its overall generated value on a given MC manufacturing system. Olhager [24] explained CODP’s role in supply chain management for upstream and downstream portions to suit supply chain requirements. DaCunha et al. [25] provided methodology for matching modules developed by suppliers using evaluation criteria to select appropriate fit into the product design cycle. McIntosh et al. [26] proposed that late customization and product differentiation are advantageous in achieving MC with emphasized individuality. They studied the applicability and standardization of this concept from food industry perspective.

Buffington [27] introduced a concept of generative mass customization in mass markets which supports the MC paradigm because of many options available in the creative choice space. Qin [28] proposed for moving some portions of customization from in-house assembly line to distribution center managed by third party. Xu and Liang [29] analyzed positioning of CODP from the perspectives of market, product, resource and production using extension superiority evaluation.

Bask et al. [30] provided an outline for modularity and customization by varying modularity for customer service functions. Elmaraghy et al. [31] expressed complexity of MC in terms of the product and manufacturing process needs and recommended for flexibility through innovative collaboration. Jeong [32] built a model to find optimum position for CODP while minimizing cost of deviating from inventory and throughput targets. Buffington [27] explained generative customization by combination of basic designs provided by suppliers which can be combined into customized product through the modularity concept.

Lin et al. [33] discussed about hybrid push–pull production system in MC using planning model of the push–pull production with a single-CODP mass customization system and extended it to model is extended to the multi-CODP mass customization production system. Medini et al. [34] explained CODP positioning for different customization levels by explaining key enablers to sustain production depending on planned objectives. Kim and Kim [35] explained positioning of CODP in a semiconductor supply chain under demand and lead time uncertainty. Mehrsai et al. [36] explained the use of modularity in structure, cloud computing, and make-to-upgrade customization concept to integrate supply sources. Agrawal et al. [37] proposed a hybrid model to make use of modularity that helps in mass customization while also reducing the negative effects of lost sales due to stoppage of using custom designs.

Sjøbakk et al. [38] explained different production situations for CODP in decision making for automation robot purchase. Daaboul and Da Cunha [8] proposed to split product attributes into standardized and customizable portions. They studied product differentiation, value and CODP.

Wikner [39] explained decoupling zone to improve continuity in changing decoupling points triggered due to mixed ingredients and assorted property requirements of end products. Wikner [40] explained eight supply chain strategies in which CODP was explained as a postponement strategy in conjunction with customization and outsourcing but didn't explain any implementation mechanism for the same.

Ngniatedema et al. [41] explained a delayed product differentiation model for raw material supply is matched with uncertain demand where supplier lead time is a constraint. Ridwan et al. [42] explained a CODP positioning-based simulation for performance improvement in mass customization for make to order furniture making company. Keddis et al. [43] explained that decoupling can be done on the different types of workflows.

Shahin et al. [44] explained a data envelopment analysis-based method to find CODP position for a lean cum agile supply chain. They mentioned that decoupling point position and lean-to agile distance can determine the selection between lean versus agile strategy. Yao and Xu [45] studied making of dynamic decisions for mass customization and performed sensitivity analysis with CODP as one of the important factors.

Cannas et al. [46] explored decoupling configurations for speculative machine manufacturing in engineer to order environment using four categories, namely special, custom, standard-custom and modular.

Tookanlou and Wong [47] explained vertical product differentiation enabled customization with lead time versus customization as conflicting objectives.

Figure 1 illustrates the evolution of CODP over the years through past 4 decades from its inception to the current state of mass customization. There is no present research about adding smartness into CODP, though we are into the era of Industry 4.0. We intend to fill that gap through this research paper.

We propose to exploit CODP at the horizontal as well as vertical product differentiation levels to enable growth of firms which are agile in nature by using the EA technique with leverage of internal as well as external expertise (in-house cum outsource supported manufacturing chain) using smart manufacturing, in process flow design as explained above. CODP studies using smart driven and EA-driven approaches are limited as shown below with very few papers that were found.

Zheng et al. [48] proposed a fruit fly algorithm-based optimization for semiconductor final test scheduling. They identified a few parameters and tested them using Taguchi-based design of experiments. Zheng and Wang [49] studied a two-stage adaptive fruit fly optimization algorithm for unrelated parallel machine scheduling problem with additional resource constraints. They made use of the initial results as an initial swarm center for subsequent evolution and made use of ANOVA for validation. Yusof and Deris [50] created machine constraint-based GAs for machine requirement of semiconductor assembly industry while minimizing cost and risk. Pan [51] provided several simplified models of fruit fly optimization problem which are easy to adopt into individual research areas. Ma and Zhang [52] provided genetic algorithm (GA)-based solution for computer aided process planning (CAPP).

Saldivar et al. [53] used clustering in genetic algorithm by identifying patterns in various areas of the supply chain to make smart customization affordable for industry 4.0. Suginouchi et al. [7] mentioned CPLEX simulation-based method for solving scheduling issues using combinatorial

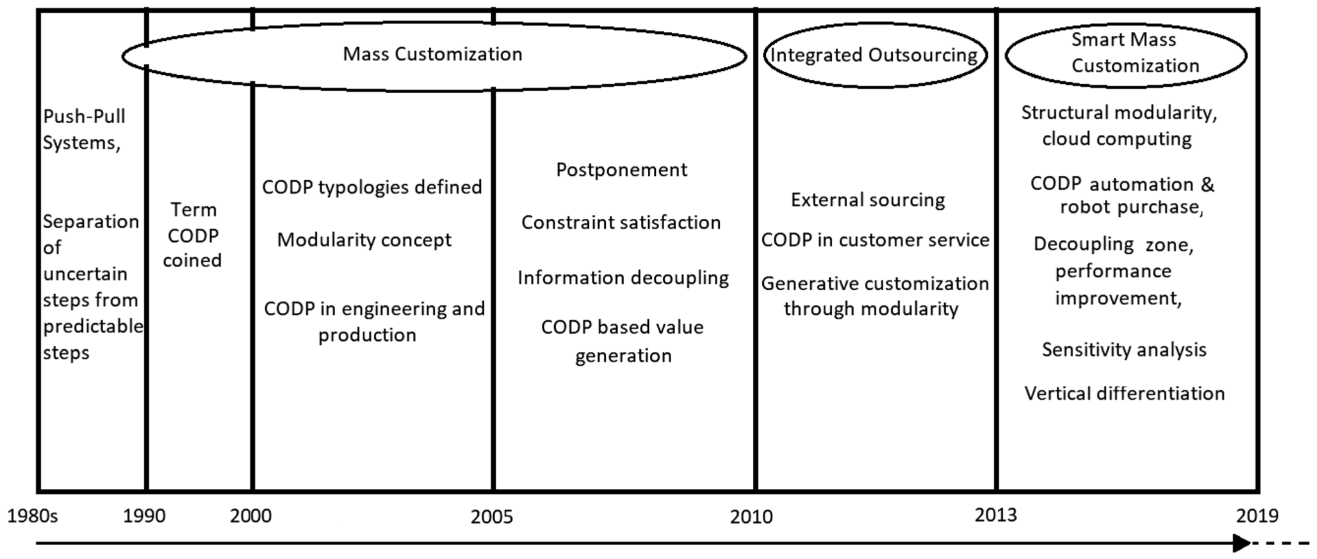

**Fig. 1** Evolution of CODP research into Smart Mass Customization

auction concept, for smart factories which are linked through the industrial internet consortia.

Zhang et al. [5] proposed big data-driven smart customization model which collects data from various sources to take decision on product customization. Zhang et al. [54] showed a cloud-based manufacturing paradigm with ubiquitous robots for product customization through 3D printing but is not about process step selection though it is a related topic.

Wan et al. [55] demonstrated a cognitive learning-driven model through IoT and big data which helps to design industrial equipment intelligently. This model was proposed to aid designers in decision making and analyzing processes while performing equipment designs.

Table 1 chronologically summarizes the converging literature which tries to link the CODP, EA and Industry 4.0 for smart mass customization. CODP and EA started off as separate research areas at separate points in time but eventually get connected in the current decade through Industry 4.0, which is a relatively newer research area.

## 3 Problem description and methodology

For developing and manufacturing new products, Industry 4.0 requires autonomous manufacturing process flow design. Planning this process flow design is essential for smart mass customization with true fulfillment of customer needs autonomously. Industry 4.0 should be agile enough to identify and create new opportunities. There is a constant evolution of new opportunities as processes and products mature. MC generally is defined to have limited boundary of options but as part of smart mass customization, the independent achievements in diverse areas can be brought together by making use of modularity, to expand the scope and develop new applications. Brun and Zorzini [22] mentioned about modularity which can be helped to interface independent systems.

As part of this constant evolution, yesterday's advance features (which were reserved to only custom or elite variants) are part of today's standard features in any product. Example is infotainment which was once part of the full end versions of cars but nowadays is getting into low end versions too. Similarly, today's advancements will become part of standard equipment in future, while future technology adds new advancements into elite or custom products of tomorrow. This leads to CODP's evolution and its shift from left side (upstream side) to rightwards (to downstream side). With new features gradually becoming part of standard equipment, the CODP or point of differentiation shifts rightwards as the earlier product differentiating (or product customization) steps now become part of the standard process flow and shift to the left side of the CODP.

Hence it is important to correctly position the CODP in autonomous smart mass customization. This will help in better material planning, layout modification and cost control. There are numerous methods used to determine and optimize CODP as explained in our literature review. We have made use of two types of genetic algorithms, namely fruit fly optimization and particle swarm optimization. The primary reseason for selecting these methods is their direct applicability in solution search across a large solution space with flexibility to choose from available modular options that can help the best fit approach. This helps in lowering time to market and easier customization that opens up diverse opportunities for mass customization smartly. We made use of the FF method for easier explanation of the concept and then made use of the same in PSO method which can help in more complex problem solving. These methods are selected preferably over other methods due to their higher efficiency, better granularity of search and relative simplicity.

In this section, we begin with defining of the problem which describes the background of problem creation and shows different problem designs. We select a specific problem type from it which is explained in the problem statement. Further, we explain and demonstrate the FF and PSO methods including detailed assumptions, explanations and simulated values to solve it. MATLAB Online R2020b was used to run the codes on a computer system with Intel® Core™ i5-6300 U CPU having dual-core processor with clock speed of 2.4 GHz and 16 GB of RAM. We describe the solution with the deployed MATLAB codes and results. We first explain the FF algorithm-based optimization to locate CODP position, followed by the same using PSO algorithm and then compare the results.

### 3.1 Defining the problem

Product lines with common process flow may have one CODP but diversified businesses with multiple vertical and horizontal product differentiations may require maintaining multiple CODPs. The latter is shown in Fig. 2. In this paper, we consider the simplistic portion of the same which is unidimensional and single product line based. This is highlighted with the dotted portion in Fig. 2. Process flow design for this portion in SMC would need sourcing of best fit modules that can be assembled together for getting final product. For this product line, one or more major specifications of the module can be decided in terms of size, geometry, material, color, reliability, fit, functionality, etc., that can be used as matching criteria in real-world for doing the search of the best fit. We make use of two algorithms here to solve the problem, namely FF and PSO. The reason

**Table 1** Chronology of converging literature

| Research area | Year | Authors | Focus area |
|---|---|---|---|
| Industry 4.0 | 2016 | Hajrizi | Digitalization, big data, complex models, simulation, online control, multicriteria optimization, knowledge and capacity |
| | 2016 | Lehmhus | Cloud-based design and sensor integrated intelligence, with IoT and additive manufacturing |
| | 2017 | Suginouchi, Kokuryo and Kaihara | Co-creative manufacturing system, SMC, smart factory linking, industrial internet consortium, autonomous negotiation, CPLEX simulation, scheduling, combinatorial auction concept |
| | 2018 | Mittal et al. | IoT, cloud, AI, big data, ML, digitization, AM, analytics |
| | 2019 | Frank, Dalenogare and Ayala | Smart manufacturing, smart products, smart supply chain and smart working, IoT, cloud services, big data, analytics, digitalization |
| | 2019 | Alcácer and Machado | IoT, cloud services, big data, analytics, digitalization |
| | 2019a | Zhang et al. | Cloud-based manufacturing, ubiquitous robots, 3D printing |
| | 2019b | Zhang et al. | Smart mass customization (SMC), big data |
| CODP | 1992 | Giesberts and Van der Tang | Position between the forecast-driven and customer order-driven portion of manufacturing process chain |
| | 2004 | Rudberg and Wikner | Combinations of engineering and production strategies, typologies of CODP for different mass customization (MC) environments |
| | 2004 | Ethiraj and Levinthal | Modularity concepts for innovation in managing complex systems, simulation |
| | 2005 | Wikner and Rudberg | Engineering and production perspectives of CODP, two-dimensional approach |
| | 2007 | Wikner and Wong | Mapped entities of postponement strategy to real-world manufacturing problems |
| | 2007 | Xu | Positioning of CODP from the perspectives of market, product, and production |
| | 2007 | Hua, Li and Lun | MATLAB-based CODP positioning with lead time constraint, capacity constraint and cost minimization objective |
| | 2008 | Luo, Han and Zhou | Optimal CODP positioning by considering product functionality, lead time and cost through information entropy and ideal point |
| | 2009 | Wang, Chang and Wang | Matching suppliers based on performance attributes like cost, quality, delivery, response, and other services |
| | 2009 | Liu et al. | Positioning of CODP in leagile supply chain using polychromatic set theory |
| | 2009 | Ge et al. | CODP positioning to optimizing the overall cost of supply chain using MATLAB simulation, constraint in lead time, value, productivity and logistics |
| | 2009 | Brun and Zorzini | Evaluation of customization strategies through modularity by analyzing complexity of process and product. Coined a term called information decoupling point |
| | 2010 | Daaboul, Laroche and Bernard | Value network modeling for positioning CODP-based on its overall generated value on a given MC manufacturing system |
| | 2010 | Olhager | CODP role in supply chain management for upstream and downstream portions to suit supply chain requirements |
| | 2010 | DaCunha, Agard and Kusiak | Matching modules developed by suppliers using evaluation criteria to select appropriate fit into the product design cycle |
| | 2010 | McIntosh et al. | Late customization, product differentiation, individuality |
| | 2011 | Buffington | Generative mass customization in mass markets, basic designs from suppliers combined into customized product through the modularity concept |
| | 2011 | Qin | Moving some portions of customization from in-house assembly line to distribution center managed by third party |
| | 2011 | Bask et al. | Outlined modularity and customization by varying modularity for customer service functions. |
| | 2011 | Xu and Liang | Positioning of CODP from the perspectives of market, product, resource and production using extension superiority evaluation |
| | 2011 | Jeong | Model to find optimum position for CODP while minimizing cost of deviating from inventory and throughput targets |
| | 2012 | Fogliatto, Da Silveira and Borenstein | Postponement of customization, Modularity in upstream, Customer input in downstream |
| | 2012 | Lin, Shi, and Wang | Hybrid push-pull production system for single-CODP mass customization system and multi-CODP mass customization production system |
| | 2012 | Medini, Da Cunha and Bernard | CODP positioning for different customization levels by with key enablers to sustain production for planned objectives |

**Table 1** (continued)

| Research area | Year | Authors | Focus area |
|---|---|---|---|
| | 2012 | Kim and Kim | Positioning of CODP in a semiconductor supply chain under demand and lead time uncertainty |
| | 2013 | Mehrsai, Karimi and Thoben | Modularity in structure, cloud computing, and make-to-upgrade customization concept to integrate supply sources |
| | 2013 | Agrawal et al. | Hybrid model to make use of modularity in mass customization with reducing effects of lost sales |
| | 2014 | Daaboul and Da Cunha | Product manufacturing flow, differentiate basic and additional steps to derive new MC variants |
| | 2014 | Sjøbakk, Thomassen and Alfnes | CODP decision making for automation robot purchase |
| | 2014 | Daaboul and Da Cunha | Split product attributes into standardized and customizable portions, product differentiation, value and CODP |
| | 2014a | Wikner | Decoupling zone to improve continuity in changing decoupling points for mixed ingredients and assorted requirements |
| | 2014b | Wikner | Eight supply chain strategies with CODP postponement in conjunction with customization and outsourcing |
| | 2014 | Ngniatedema, Fono, Mbondo | Delayed product differentiation model, uncertain demand, constraint supplier lead time |
| | 2015 | Ridwan, Purnomo and Sufa | CODP positioning simulation for performance improvement in mass customization in make to order production |
| | 2015 | Keddis et al. | Decoupling with different types of workflows |
| | 2016 | Shahin et al. | Data envelopment analysis to find CODP position in lean cum agile supply chain |
| | 2018 | Yao and Xu | Dynamic decisions for mass customization and performed sensitivity analysis with CODP factor |
| | 2019 | Cannas et al. | Decoupling configurations for speculative machine manufacturing in engineer to order environment |
| | 2019 | Tookanlou and Wong | Vertical product differentiation enabled customization with lead time versus customization as conflicting objectives |
| EA | 2005 | McCall | Evolution-based empirical search optimization techniques |
| | 2009 | Yusof and Deris | Machine constraint-based GAs for machine requirement of semiconductor assembly industry while minimizing cost and risk |
| | 2012 | Ma and Zhang | Genetic algorithm (GA)-based solution for computer aided process planning (CAPP) |
| | 2014 | Zheng, Wang and Wang | Fruit fly algorithm-based optimization for semiconductor final test scheduling, Taguchi-based design of experiments |
| | 2014 | Pan | Simplified models of adoptable fruit fly optimization problem |
| | 2016 | Zheng and Wang | Two-stage adaptive fruit fly optimization algorithm for unrelated parallel machine scheduling problem with resource constraints, ANOVA |
| | 2016 | Saldivar et al. | Clustering in GA, Identifications of patterns in areas of the supply chain, affordable Smart Customization for industry 4.0 |

for choosing FF and PSO here is their relatively simpler possibilities in application into generic problem solving with fast computing and near optimal results that can be generated on commonly available computing platform as explained above. FF technique is used to address simpler modularity search problems, while the PSO technique is used for addressing slightly more complex applications which involve multiple search criteria for matching modularity needs. Although we begin with unidimensional portion in a two-dimensional search space for FF algorithm, in the later stage (in Sect. 4) we use PSO algorithm in a ten-dimensional search space with the aim to target multiple characteristics and to standardize or unify multiple CODPs to a possible extent. The models developed and shown here are generic and for demonstration purposes. These can be deployed suitably in the real-world scenarios based on actual product and process flow design needs.

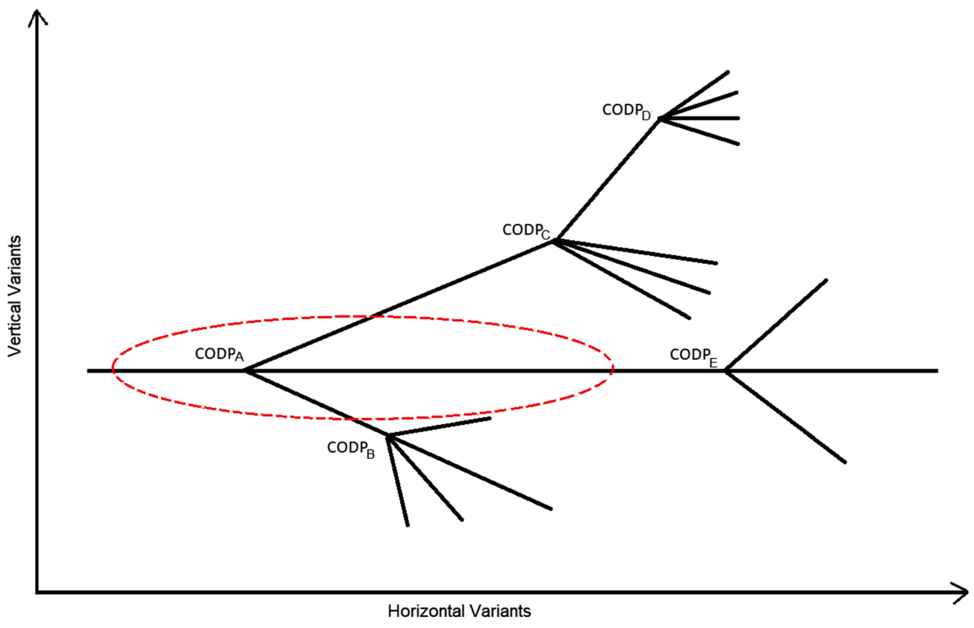

**Fig. 2** Multiple CODPs due to vertical and horizontal variant portfolios

## 3.2 Problem statement

The problem statement is to locate the most suitable (best fit) CODP position in the unidimensional manufacturing process flow design. The objective is to optimize the results with highest possible standardization in processes without compromising the customization capability of the manufacturing chain. Higher standardization is meant to reduce complexity and improve efficiency of manufacturing chain. This is suggested to be achieved without reducing the options for fulfillment of customer choices.

The objective is to break the complexity barrier in manufacturing by an increase in modularity.

Complexity reduction can help in simplified process flow, which in turn improves machine efficiency [56]. James and Mondal [56] ranked complexity as the 2nd highest influencing parameter that reduces machine efficiency in mass customization.

The modularity is maximized here by fruit fly's smell-based search algorithm. Higher smell concentration due to existence of food drives the foraging behavior of drosophila (fruit fly), per Pan [51]. In this paper, smell concentration is assumed analogous to availability of next modular manufacturing step which could be in-house or externally sourced while designing a manufacturing process chain. The CODP position is determined with the best fit point which balances between the modularity and customization. This concept is applicable to the overall supply chain design too in addition to manufacturing process flow design.

## 3.3 Fruit fly model assumptions

We assumed manufacturing process chain with an upstream (left-hand side) versus downstream (right-hand side) with CODP position (represented by CODP(i)), which needs to be located somewhere in between. This position of CODP is intended to be optimized by maximizing modularity and minimizing complexity, while retaining the customization capability.

A longer upstream section compared to downstream section would mean most of the chain has a common process flow. This requires greater modularity and high standardization. It would mean high manufacturing efficiency as majority of the processes would be generic. However, this would be with a lesser scope of customization in the downstream as most of the product's processing is completed in the generic upstream section itself. Conventional mass customization (MC) is characterized by a bounded space (limited space) and is analogous to this scenario.

In the opposite scenario, a shorter upstream portion would limit the standardization, while the longer downstream would have more options and space for customization. In other words, more options of flexibility would exist within the manufacturing chain. But it may come with a higher risk of investment and need for added custom sub-processes. This can make it less efficient from manufacturing perspective as compared to the scenario explained in the previous paragraph. This issue can be helped by SMC because outsourcing or finding alternate options internally through smart enabled planning can mitigate such risks.

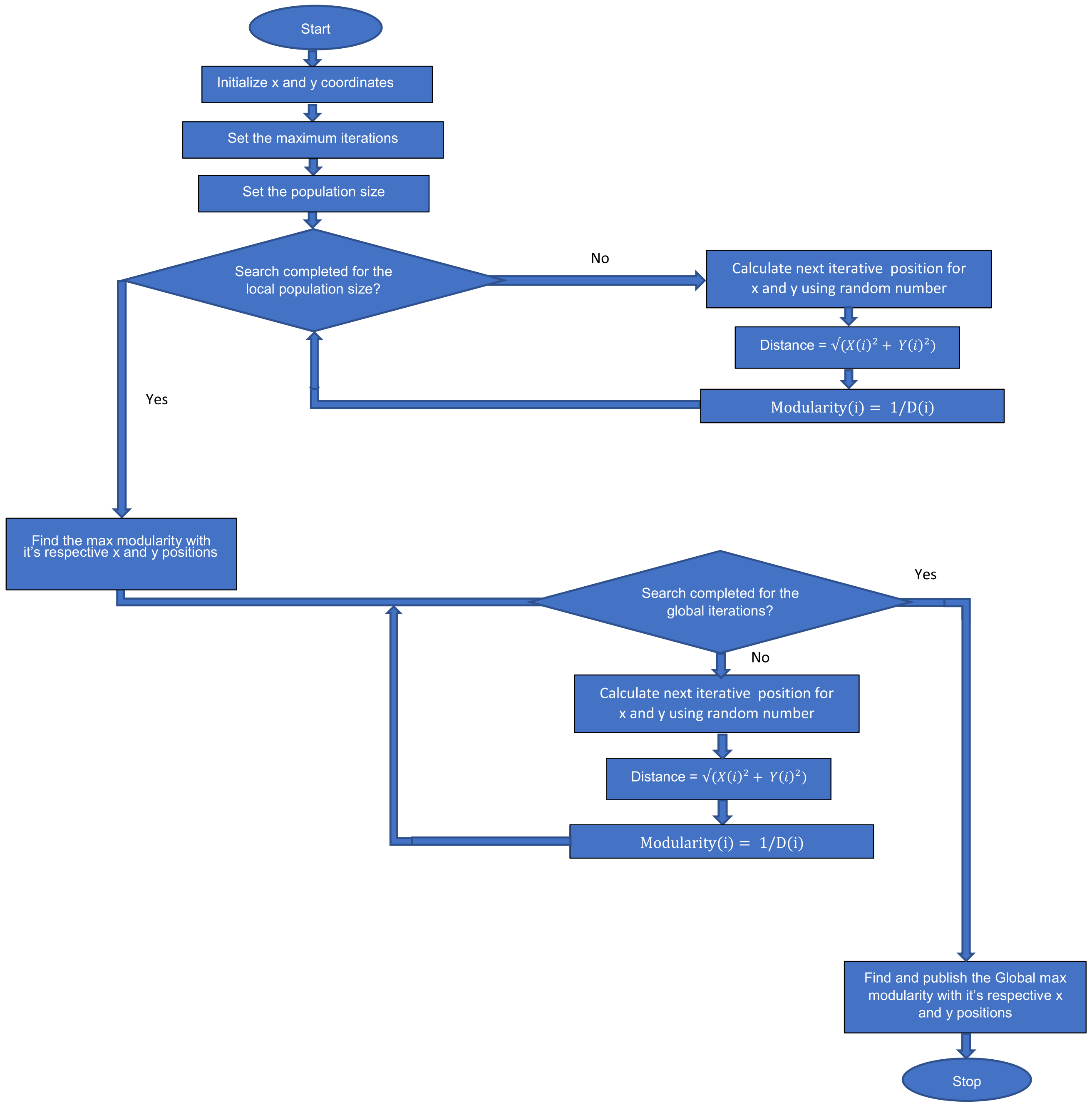
Start
Initialize x and y coordinates
Set the maximum iterations
Set the population size
Search completed for the local population size?
No
Calculate next iterative position for x and y using random number
Distance = $\sqrt{(X(i)^2 + Y(i)^2)}$
Modularity(i) = 1/D(i)
Yes
Find the max modularity with it's respective x and y positions
Search completed for the global iterations?
Yes
No
Calculate next iterative position for x and y using random number
Distance = $\sqrt{(X(i)^2 + Y(i)^2)}$
Modularity(i) = 1/D(i)
Find and publish the Global max modularity with it's respective x and y positions
Stop

**Fig. 3** Fruit fly algorithm-based modularity maximization

The modularity parameter 'Modularity($i$)' considered here is analogous to smell parameter which influences fruit fly's foraging behavior. Like the fruit fly's movement towards source of greater smell, the algorithm searches better source of finding a best fit module for next manufacturing step in the chain. The goal here is to lessen the effort in designing and executing the manufacturing process steps by smart enabled algorithm through IoT. Wireless fidelity (WIFI) and IoT enabled inputs to the Internet can help in selecting the best fit modular steps until the optimum CODP position is achieved. In-house sources (internal) or well-established vendors (external) could be chosen as the best fit selection depending on the manufacturing step.

The distance parameter '$D(i)$' represents the distance from initialized arbitrary position to the next step which is slightly better in terms of modularity. The iteration is repeated until the best fit modular step is located (internally or externally). We however, in this paper, don't identify internal or external in the coding. We recommend using it

**Table 2** Parameters for FF runs

| EA category | Problem type | Parameters used | Iterations | Search space | Figures | Matlab code |
|---|---|---|---|---|---|---|
| Fruit fly optimization | Modularity maximization | Complexity = nd<br>Distance | < 10,000 | Two dimensions | 3 | Appendix 1 |
| | CODP optimization and modularity maximization | Complexity = 5 | < 1000 | Two dimensions | 23 | Appendix 2 |
| | CODP optimization | Complexity = 1<br>Complexity = 10 | < 1000 | | | Appendix 2 refer note and put complexity = 1, 10 |

**nd* not defined

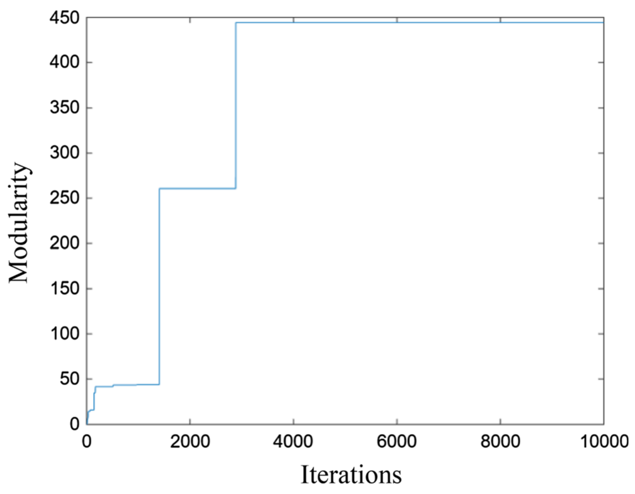

**Fig. 4** Modularity maximization

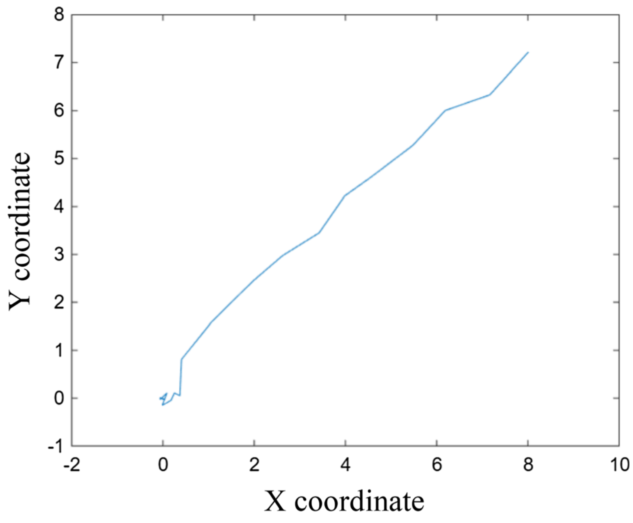

**Fig. 5** Fruit fly coordinates

in application so that the optimization achieved is at a global optimum. It could be a modular production part which is directly outsourced from a supplier, or sourced internally through internal search within the organization's own factory(s). Separate qualification processes may be needed for matching the specifications (like materials, dimensions, properties, etc.) to select such product part or process modules in the real-world. We shall explain the same in later sections. Smell concentration increases with inversely with distance traveled nearer to the food source. Similarly, in this algorithm, modularity increases with closeness to the new modular part or process that is sourced.

$$\text{Distance } = \surd\left(X(i)^2 + Y(i)^2\right) \tag{1}$$

Extended downstream means more new processes on top of existing standard processes so that more customization features could be added. Higher customization can lead to high complexity. We manage it with the complexity parameter. Complexity parameter termed simply as 'complexity' here is initially set at an absolute value or number.

In our simulation, we assumed complexity = 5. The assigned number for complexity is based on the difficulty level which is driven by higher number of customized product features or the technological barrier which increases the effort of manufacturing execution.

CODP is initialized at inverse of initial distance parameter-based position.

$$\text{CODP}(i) = 1/D(i) \tag{2}$$

With each increment of the distance flown by the fruit fly principle, the CODP shifts to the right side of the process chain. The CODP keeps shifting till the manufacturing process and CODP position are optimized. The loop in the MATLAB program helps in finding optima within the local population. Local population size is defined as 'Num$_{\text{pop}}$' was set as 15. The maximum iterations allowed is represented with 'It$_{\max}$', which were set as 1000. At 100, the final optimum wasn't achieved, hence we set it at 1000. The 'It$_{\max}$' allows the algorithm to search in other populations also, to arrive at global optimum.

Fitness function which is used here is,

$$\text{Modularity}(i) = \text{complexity} - \text{CODP}(i) \tag{3}$$

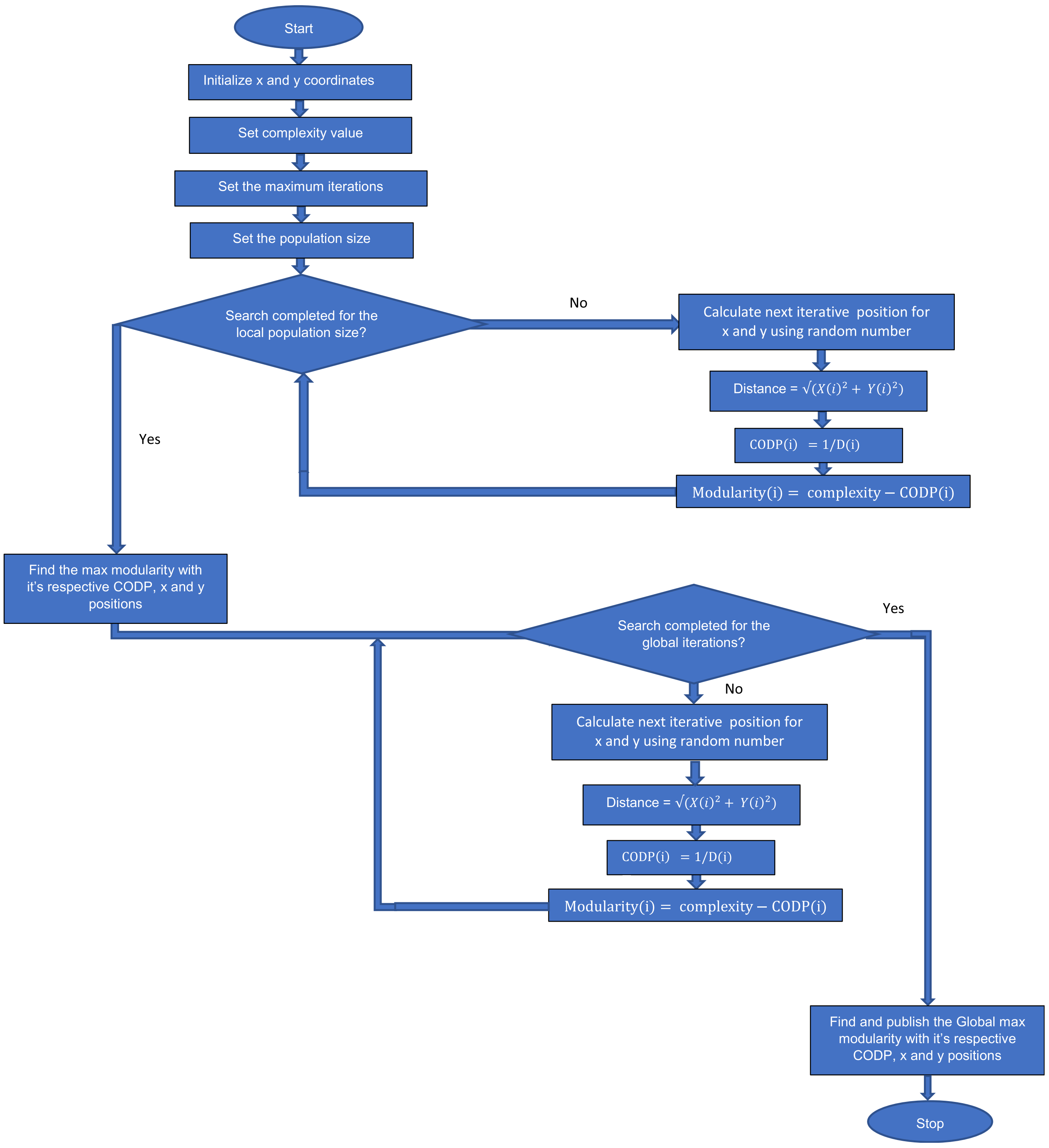
Start
Initialize x and y coordinates
Set complexity value
Set the maximum iterations
Set the population size
Search completed for the local population size?
No
Calculate next iterative position for x and y using random number
Distance = $\sqrt{(X(i)^2 + Y(i)^2)}$
CODP(i) = 1/D(i)
Modularity(i) = complexity − CODP(i)
Yes
Find the max modularity with it's respective CODP, x and y positions
Search completed for the global iterations?
Yes
No
Calculate next iterative position for x and y using random number
Distance = $\sqrt{(X(i)^2 + Y(i)^2)}$
CODP(i) = 1/D(i)
Modularity(i) = complexity − CODP(i)
Find and publish the Global max modularity with it's respective CODP, x and y positions
Stop

**Fig. 6** Fruit fly algorithm-based CODP optimization

The assumption used is the complexity reduces and modularity achieves maximum possible value. This happens while CODP shifts rightwards on the process chain, depending on the modular processes sourced. Increase in modularity causes the CODP to shift towards right to improve standardization.

The model and parameter assumptions for different scenarios of FF runs performed are summarized in Table 2. The flowcharts and descriptions of the same are explained in next section.

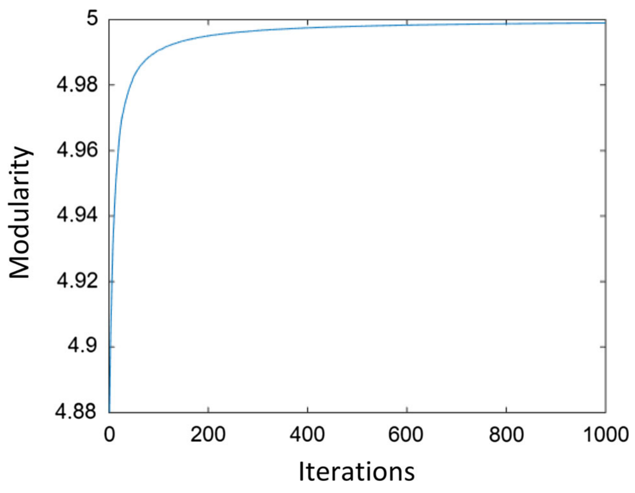

**Fig. 7** Modularity maximization for treatment 1

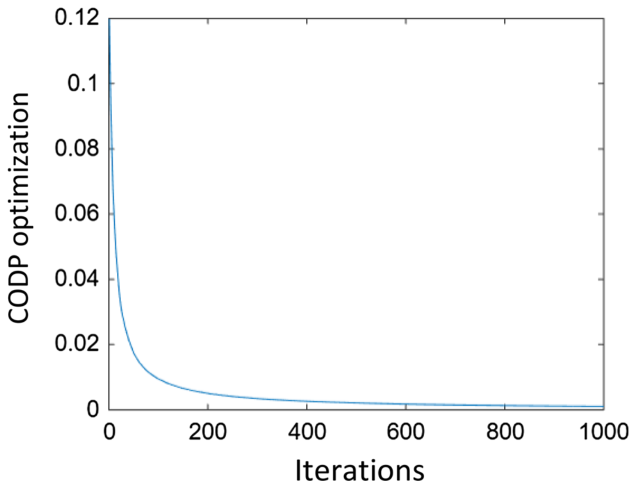

**Fig. 8** CODP optimum for treatment 1

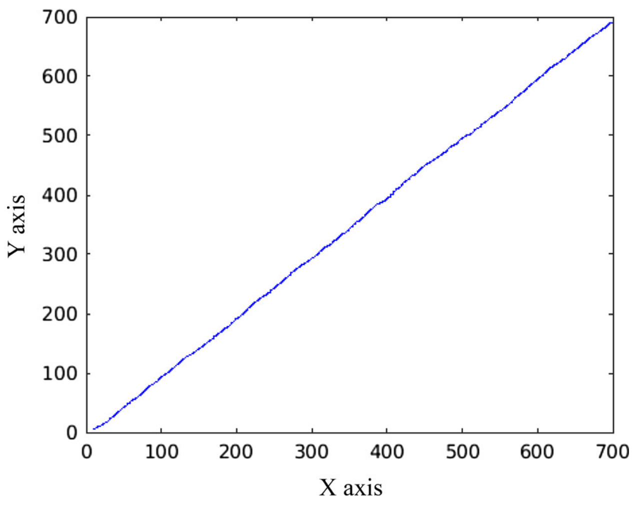

**Fig. 9** Fruit fly route for treatment 1

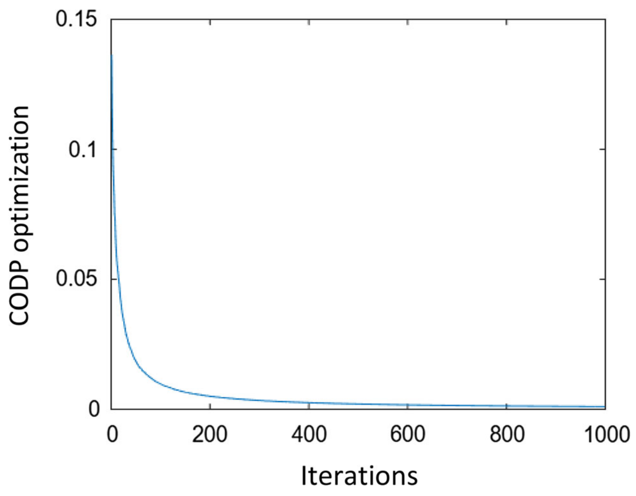

**Fig. 10** CODP optimum for treatment 3

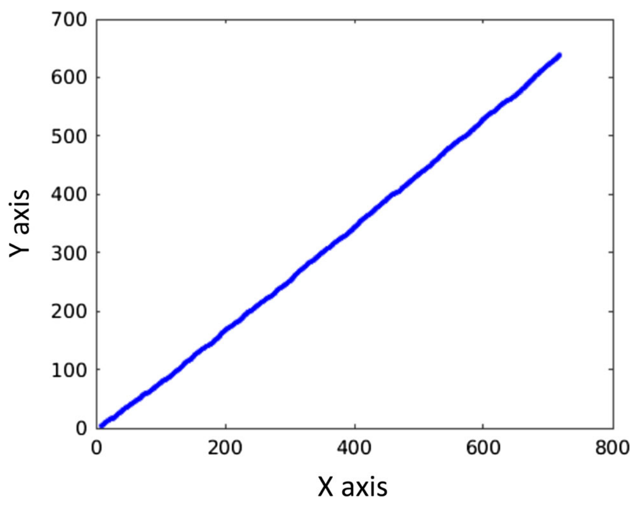

**Fig. 11** Fruit fly route for treatment 3

## 3.4 Fruit fly program code and experimental run

In this section, we show the flowcharts used for modularity maximization and CODP optimization based on fruit fly algorithm. The problem definition, parameters and model assumptions have been explained in above section. The MATLAB codes for this were prepared by modifying the framework from Pan [51] and are shown in Appendices 1 and 2 in Supplementary Information for modularity maximization and CODP optimization, respectively. Initially, we performed modularity maximization as explained in Fig. 3. There was no reference complexity value set here, and the evolution of modularity is based on number of iterations followed. Figures 4 and 5 illustrate the results of the same.

Appendix 2 in Supplementary Information shows the program code for CODP optimization. The MATLAB code shown in Appendix 2 in Supplementary Information is for

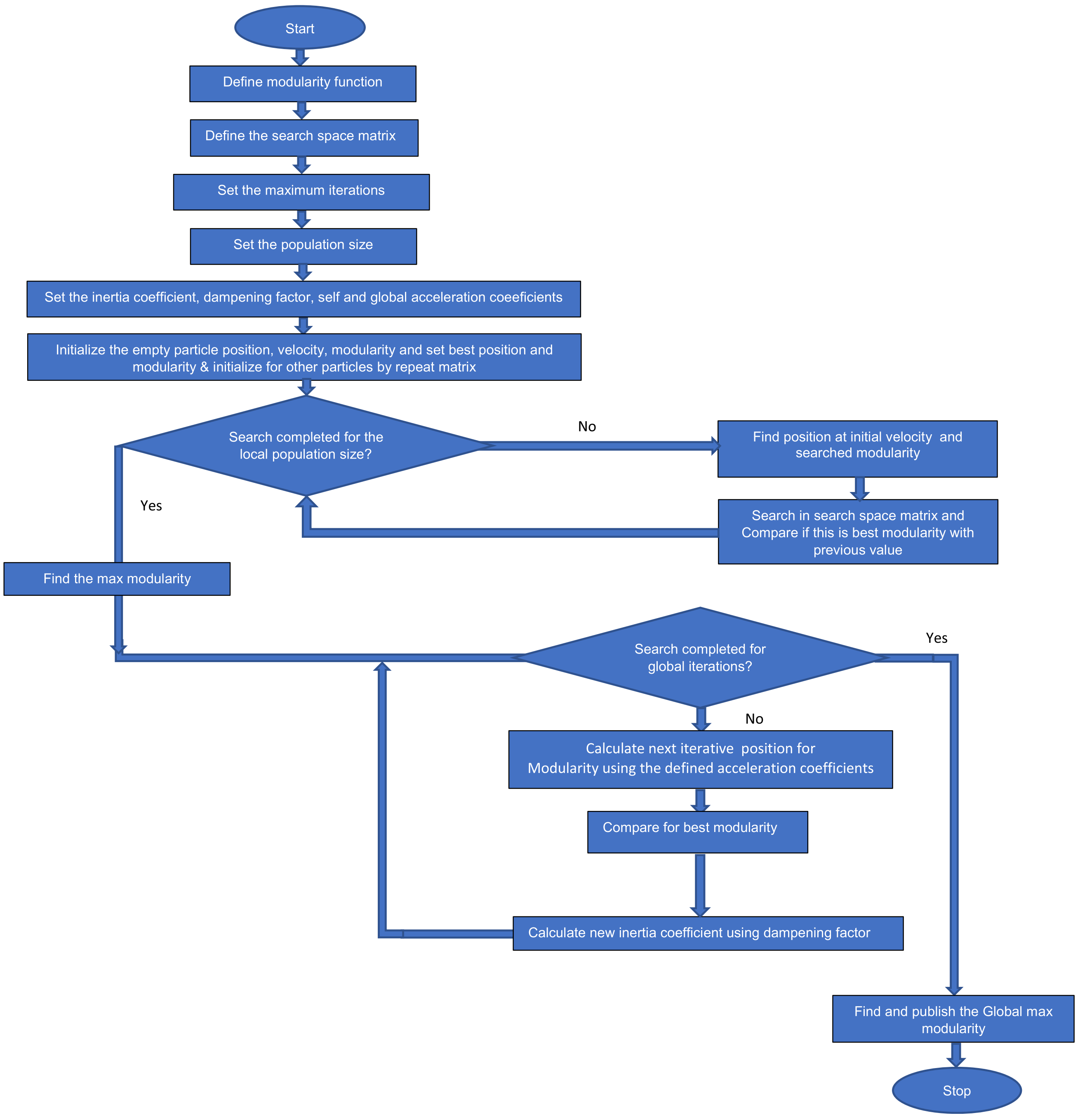
Start
Define modularity function
Define the search space matrix
Set the maximum iterations
Set the population size
Set the inertia coefficient, dampening factor, self and global acceleration coeeficients
Initialize the empty particle position, velocity, modularity and set best position and modularity & initialize for other particles by repeat matrix
Search completed for the local population size?
No
Find position at initial velocity and searched modularity
Search in search space matrix and Compare if this is best modularity with previous value
Yes
Find the max modularity
Search completed for global iterations?
Yes
No
Calculate next iterative position for Modularity using the defined acceleration coefficients
Compare for best modularity
Calculate new inertia coefficient using dampening factor
Find and publish the Global max modularity
Stop

**Fig. 12** PSO algorithm with modularity maximization

treatment 1 which was done for complexity equal to 5. Graphs were generated based on the same. Figure 6 shows the flowchart of this algorithm.

Figures 7, 8 and 9 show results for treatment 1 as shown below. The approximate optimum is achieved between 800th and 900th iteration but value improved further, as we ran the code for 1000 iterations.

Since complexity set at 5, modularity tries to reach 5 also towards end of process maturity which means CODP shifted towards right-hand side and left-hand side distance (upstream distance) is maximized, while right-hand side distance (downstream distance) is minimum. This denotes high standardization and efficient customization, as the process matures towards end of the optimization cycle. In the real manufacturing world, once the complexity barrier is broken, the processes mature overtime. This helps for targeting next level of complexity and to produce the next generation of products by production companies. This

**Table 3** Parameters for PSO runs

| EA category | Problem type | Parameters used | Iterations | Search space | Figures | Matlab code |
|---|---|---|---|---|---|---|
| Particle swarm optimization | Modularity maximization | Complexity = nd $w_{\text{damp}} = 0.98$; $a = 0.1$; $b = 0.2$ | 40 | Ten dimensions | 12 | Appendix 3 |
| | | Complexity = nd $w_{\text{damp}} = 0.97$; $a = 0.2$; $b = 0.5$ | 40 | | | |
| | CODP optimization and modularity maximization | Complexity = 100,000 $w_{\text{damp}} = 0.98$; $a = 0.3$; $b = 0.6$; | $< 70$ | Ten dimensions | 15 | Appendix 4 |
| | CODP and modularity optimum with learning function $\log(x^2)$ | Complexity = 100,000 $w_{\text{damp}} = 0.98$; $a = 0.6$; $b = 0.8$; | $< 100$ | Ten dimensions | 19, 23 | Appendices 5, 6 |
| | CODP and modularity optimum with learning function $= 1 + (\log 2(x))$ | Complexity = 100,000 $w_{\text{damp}} = 0.98$; $a = 0.6$; $b = 0.8$; | $< 100$ | Ten dimensions | 19, 23 | Appendices 5, 6 |
| | CODP and modularity optimum with Wright's learning function $= 10{,}000 + \log2(x)$ | Complexity = 100,000 $w_{\text{damp}} = 0.98$; $a = 0.6$; $b = 0.8$ | 100 | Ten dimensions | 23 | Appendix 7 |
| | CODP and modularity optimum with Wright's learning function $= 2900 + 50 * \log2(x)$ | Complexity = 100,000 $w_{\text{damp}} = 0.98$; $a = 0.4$; $b = 0.6$ | $< 50$ | Ten dimensions | 23 | Appendix 8 |

**nd* not defined

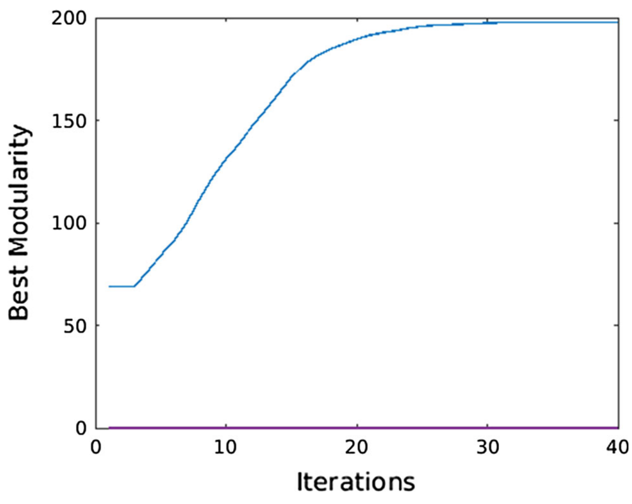

**Fig. 13** Modularity optimum for PSO model—treatment 1

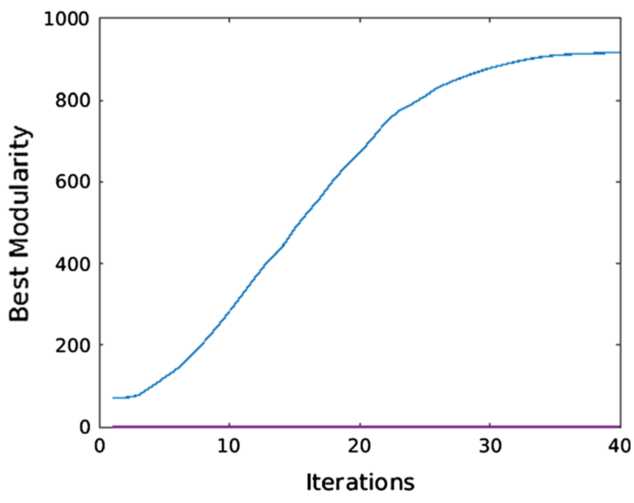

**Fig. 14** Modularity optimum for PSO model—treatment 2

leads to evolution of better featured products over a given span of time.

We performed treatment 2 with this algorithm by assuming complexity equal to 1. The results of that looked almost the same (similar to Figs. 8, 9). Next, we performed treatment 3 with this algorithm by assuming complexity equal to 10. The results looked similar like treatments 1 and 2. Here complexity is 10, so modularity tries to catch up with it based on smell concentration which is closeness to CODP or similarity or fitness. In the end, it maximized the modularity to a value closer to 10 (= 9.999) while optimizing CODP to 0.001. This is shown in Figs. 10 and 11.

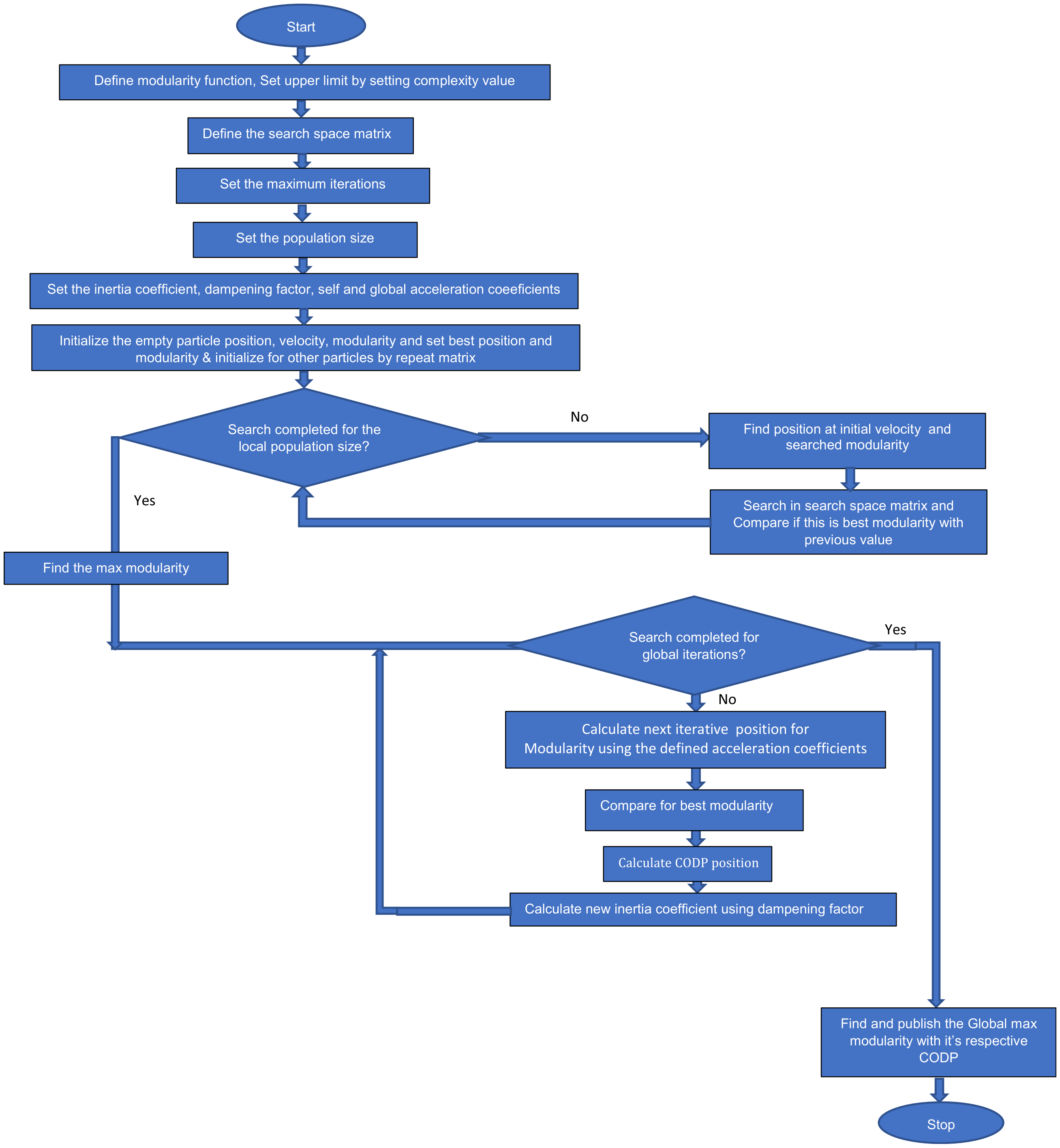
Start
Define modularity function, Set upper limit by setting complexity value
Define the search space matrix
Set the maximum iterations
Set the population size
Set the inertia coefficient, dampening factor, self and global acceleration coeeficients
Initialize the empty particle position, velocity, modularity and set best position and modularity & initialize for other particles by repeat matrix
Search completed for the local population size?
No
Find position at initial velocity and searched modularity
Search in search space matrix and Compare if this is best modularity with previous value
Yes
Find the max modularity
Search completed for global iterations?
Yes
No
Calculate next iterative position for Modularity using the defined acceleration coefficients
Compare for best modularity
Calculate CODP position
Calculate new inertia coefficient using dampening factor
Find and publish the Global max modularity with it's respective CODP
Stop

**Fig. 15** PSO algorithm with modularity maximization and CODP positioning

## 3.5 PSO model assumptions

Particle swarm optimization-based EA run is explained in this section with the objective to optimize CODP by maximizing modularity. This is done with the assumption that higher modularity can break complexity barrier of manufacturing for evolving SMC systems. We initially present treatments 1 and 2 just to maximize modularity without optimization of CODP, to focus on modularity. Later, in treatment 3 we illustrate and explain CODP optimization through a modified MATLAB code. In treatments 1 and 2, the modularity isn't limited with any upper complexity bound unlike the FF model. The benefit of this model is that modularity keeps evolving with higher

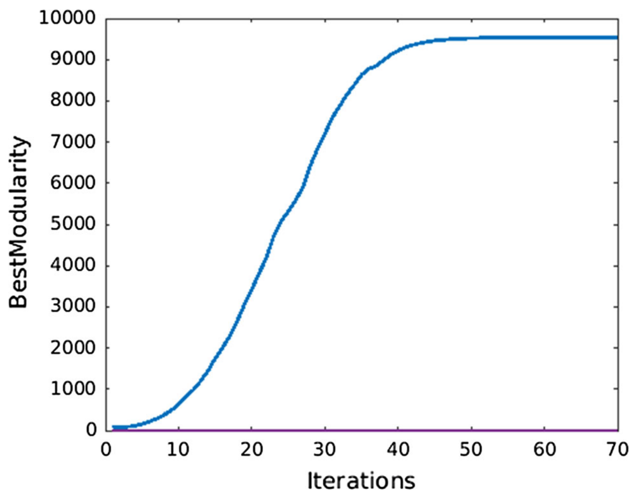

**Fig. 16** Modularity optimum for PSO model—treatment 3

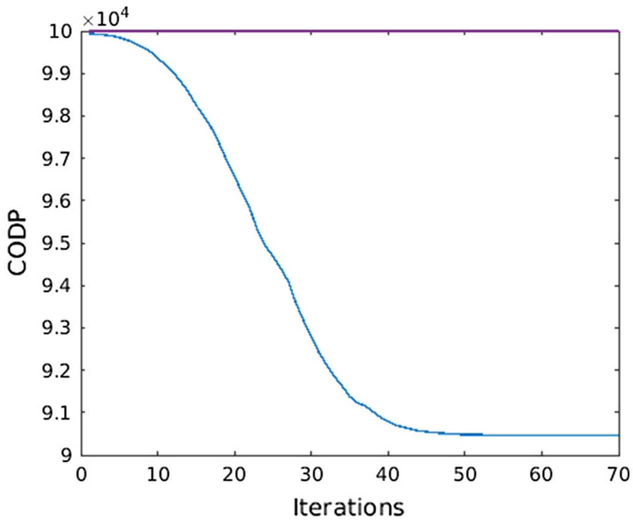

**Fig. 17** CODP optimum for PSO model—treatment 3

values of maximized modularity, to evolve into higher levels of evolution. This is explained later with its real-world applications. FF algorithm was used by us to explain the complexity simplified in one dimension, but in PSO the coding has been performed for multi-dimensional match of product and process modules. In treatment 3, we placed an upper bound for complexity and optimized CODP while maximizing modularity. In this code, we find that higher levels and values of complexity can be simulated to optimize the CODP which could be closer to real-world manufacturing problems like semiconductor and automobiles.

In the PSO model, we define a ten-dimensional search space. This space is created to represent the set of all solutions offered globally which includes internal and outsourcing process options, while designing the manufacturing chain. This search space created for each dimension can be used to represent various parameters like fit in various axes, functions, features, sizes, aesthetics, material compositions, etc., while selecting a module that fits into a final product.

The modularity function operates here with a modularity value by summing the different attributes in each dimension. Coding was done by us for the PSO algorithm with reference from Heris [57]. The particle is initialized with empty position, zero velocity, empty modularity, a personal best, and a social best. Particle's position is filled by using uniform random function. The velocity vector and the comparative drive to reach best modular position between oneself and the swarm members help the PSO algorithm to achieve the optimum position which represents the global best modularity.

A damping factor '$w_{\mathrm{damp}}$' of 0.98 is used by us. An inertia coefficient '$w$' = 1 is used. Self-acceleration coefficient '$a$' is set at 0.1. Global acceleration coefficient '$b$' is set at 0.2. We set 25 iterations in internal program loop (swarm size represented by the variable '$\mathrm{size}_{\mathrm{Pop}}$' here) and 45 overall iterations (represented the variable by '$\mathrm{It}_{\max}$' here). Well within 40 overall iterations, the PSO program maximized the modularity function. The code used is mentioned below.

Various scenarios of PSO runs are explained in the next section. The model and parameter assumptions used for performing those runs are summarized in Table 3.

## 3.6 PSO algorithm code and experimental run

For PSO, we make use of a different set of assumptions and treatments compared to FF run, Code used for treatment 1, 2 and 3 here are explained below. It is to be noted that the treatments 1, 2 and 3 mentioned for PSO runs here are entirely different and not to be confused with the experiments in FF runs.

A function tab is used to define modularity function called modularity($x$), which is a summation function of all attributes pertaining to modularity, which needs to be searched against the multidimensional search space. Modularity function is defined here using $z$ = modularity($x$), where $z$ = sum($x$) and is saved as a function in MATLAB. Main code is written in the second tab as shown below. The attributes of the modules in the search space are arranged in a 1 × 10 matrix with values ranging from 1 to 10 represented by '$\mathrm{Va}_{\mathrm{Min}}$' and '$\mathrm{Va}_{\mathrm{Max}}$,' respectively. The remaining assumptions and variables are already explained in the previous section.

Figure 12 shows the algorithm for modularity maximization. Appendix 3 in Supplementary Information shows the MATLAB code for this.

In this run, the modularity got maximized at a score of 197.74. The graph which shows maximization of best

**Table 4** Summary of run time and results for FF runs

| EA category | Problem type | Figure numbers | Average run time | Output | Results | Matlab code |
|---|---|---|---|---|---|---|
| Fruit fly optimization | Modularity maximization | 3, 4, 5 | 539.86 s | Max modularity = 445 | Modularity maximized unconstrained, increases depending on number of iterations | Appendix 1 |
| | CODP optimization and Modularity maximization | 6, 7, 8, 9 | 5.598 s | CODP = 0.01@complexity of 5 | Modularity nears complexity, CODP optimized with reference to the set complexity value | Appendix 2 |
| | CODP optimization | 10, 11 | 0.177 s | CODP = 0.001@complexity of 1, CODP = 0.01@complexity of 10 | CODP optimized with reference to the set complexity value | Appendix 2 refer note and put complexity = 1, 10 |

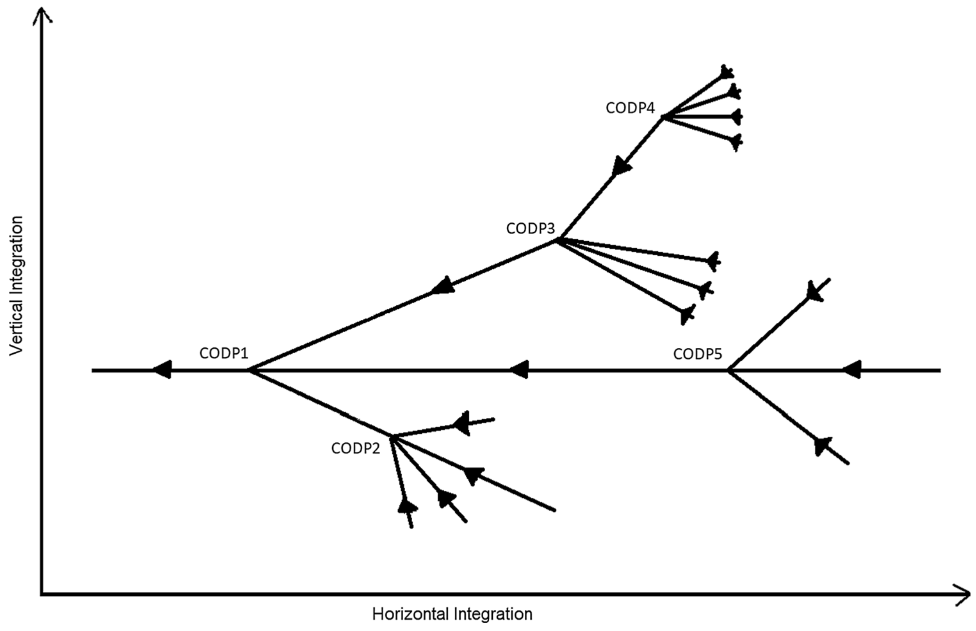

**Fig. 18** Evolution of CODP by vertical and horizontal integration

modularity variable is plotted against the iterations in Fig. 13.

Within about 100 iterations ($It_{max}$ value = 100), the optimum solution was arrived for the modularity function. The values achieved for the modularity value differed in different treatments and runs. Hence, a sensitivity analysis was performed to check the influence of parameters on the results of the maximized (best modularity) parameter. So, further treatments were performed with varied values of the '$a$' and '$b$' values. An observation made was that the social (global) acceleration coefficient has higher influence in magnifying the modularity maxima than self-acceleration coefficient. It was found that upper bound of the result can be limited mainly based on the $Va_{Min}$ to $Va_{Max}$ range selected, acceleration coefficient values selected and with the self-acceleration coefficient to a certain extent.

As explained in the previous paragraph, with changes in parameter inputs, we were able to maximize value of modularity function. Figure 14 shows modularity function maximized at close near to value between 900 and 1000. Figure 16 shows the modularity maximized at near to a value of 10,000 in next paragraphs.

In treatment 3, we slightly modified the code and added an upper bound for complexity to optimize the CODP. We find that this model is applicable for highly complex products and processes because this model has the flexibility to accommodate multidimensional attributes in the module search for SMC. Complexity value used here is 100,000.

The modified main code (second tab) for CODP optimization (treatment 3) is shown below. The first tab is the same as used in treatments 1 and 2.

Figure 15 shows the algorithm for modularity maximization cum CODP optimization. Appendix 4 in Supplementary Information shows the MATLAB code for this.

The results are plotted in Figs. 16 and 17.

## 4 Results and discussion

As explained in Sect. 3, we successfully modeled the problem. We then coded the two EA methods using MATLAB to simulate and generate results.

### 4.1 Results of FF optimization run

The FF algorithm was focused on fruit fly foraging behavior to maximize modularity by flying longer distances to source the best suited module for a manufacturing chain design. The module was assumed to be a process or a part of the MC product. Figure 4 shows the results of modularity maximization for the coordinates achieved by the FF run shown in Fig. 5. In this run, we did 10,000 iterations for higher modularity achievement and the run time was 539.86 s with maximum modularity of 445. Reduced iteration of 1000 produced smaller modularity value of 43.82 in run time of 5.19 s.

The complexity value was set to aid as a reference for the modularity improvements to be made (refer to Fig. 7 for modularity maximization) with each iteration, to arrive at the CODP. The result of CODP optimization through modularity maximization is plotted in Fig. 8 and the fruit fly route plotted in Fig. 9. We did sensitivity analysis with various changes in iterations and complexity values, but the results were found similar as shown in Figs. 10 and 11. The modularity gets maximized to a value closer to the complexity value which was assumed at the initialization of the algorithm. We made use of different values like 5 (treatment 1), 1 (treatment 2) and 10 (treatment 3) to set a reference value for the product cum process complexity. These values could be appropriately selected to suit the real-world requirements. The process for deploying this is summarized in Sects. 3.3 and 3.4.

In treatment 1, CODP got optimized at a modularity value of 0.01. This needs to be visualized as a manufacturing process flow line with a scale defined based on complexity. With higher modularity achieved after many iterations, the CODP shifts towards downstream creating a standardized and mature process. For all the three treatments, the CODP reaches towards $> 99\%$ of complexity value. As the position of the FF gets closer to (700, 700) (as shown in the $X$–$Y$ axis plot), with about 800 iterations, the optima got achieved for all the three treatments.

In Treatment 1, (Complexity $= 5$), $\text{CODP}_{\text{best}}$ was at Max(Modularity) which equaled 4.990.
$X_{\text{axis}} = 699.4717$.
$Y_{\text{axis}} = 690.9756$.
In Treatment 2, (Complexity $= 1$), $\text{CODP}_{\text{best}}$ was at Max(Modularity) of 0.9990.
In Treatment 3, (Complexity $= 10$), $\text{CODP}_{\text{best}}$ was at Max(Modularity) of 9.990.

Table 4 shows the average run times and results for the FF run scenarios explained above and helps to make their comparisons. It also shows the respective output values obtained for experimental runs of the parameter inputs shown in Table 2.

### 4.2 Results of PSO optimization run

The same concept was then deployed into PSO-based optimization first by maximizing modularity in the code for treatments 1 and 2.

In Treatment 1, we had following parameter values assumed.
$w_{\text{damp}} = 0.98$; (dampening factor).
$a = 0.1$; (self-acceleration coefficient).
$b = 0.2$; (global acceleration coefficient).

Run results of treatment 1 start from best modularity $= 68.7232$ in iteration 1 to $= 197.7457$ in iteration 40.

Modularity was maximized to 197.7457 after 40 iterations (refer to Fig. 13).

In Treatment 2, we had following parameter values assumed.
$w_{\text{damp}} = 0.97$; (dampening factor).
$a = 0.2$; (self-acceleration coefficient).
$b = 0.5$; (global acceleration coefficient).

Modularity was maximized to 912.0715 after 40 iterations (refer to Fig. 14).

Since we didn't limit the upper bound of the modularity in treatments 1 and 2, the modularity value showed evolution into higher values depending on the values of self-acceleration, the social acceleration coefficient in the swarm, and the variable sizes used in the search space as explained earlier. This code can be used where modularity can be higher, for highly complex products. One of the practical examples of this application is Moore's law which stated that the number of transistors in a chip doubles in a short period of a few years [58]. This illustrates the evolution of modularity as more complex product requirements come up in time. Figures 13 and 14 illustrate the maximization of modularity. Real-world scenarios exist

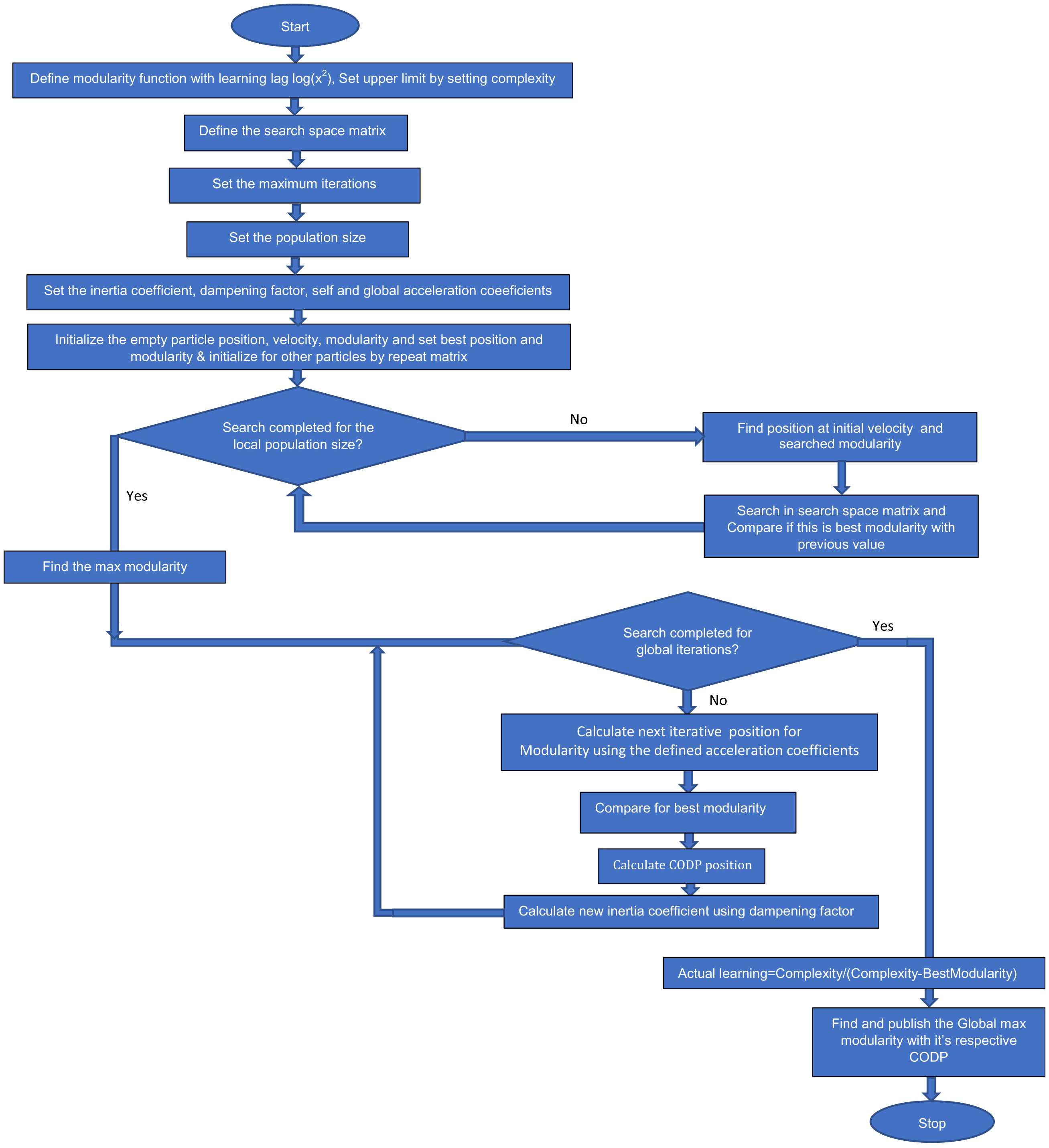
Start
Define modularity function with learning lag log(x2), Set upper limit by setting complexity
Define the search space matrix
Set the maximum iterations
Set the population size
Set the inertia coefficient, dampening factor, self and global acceleration coeeficients
Initialize the empty particle position, velocity, modularity and set best position and modularity & initialize for other particles by repeat matrix
Search completed for the local population size?
No
Find position at initial velocity and searched modularity
Search in search space matrix and Compare if this is best modularity with previous value
Yes
Find the max modularity
Search completed for global iterations?
Yes
No
Calculate next iterative position for Modularity using the defined acceleration coefficients
Compare for best modularity
Calculate CODP position
Calculate new inertia coefficient using dampening factor
Actual learning=Complexity/(Complexity-BestModularity)
Find and publish the Global max modularity with it's respective CODP
Stop

**Fig. 19** PSO algorithm with fast learning, modularity maximization and CODP positioning

where semiconductor companies eventually integrated more modules and functions into single chip, thus reducing the overall components in a mother printer circuit board of any product. Thus, treatments 1 and 2 attempt to simulate the real-world scenario, which involves evolution of CODP through absorbing of vertical and horizontal CODP chains into a single unified CODP, as shown in Fig. 18. The different nodes are different CODPs, i.e. CODP2,3,4,5… which get moved inwards and get consolidated into CODP1 which is the unified CODP.

Hence, Fig. 18 represents evolution of CODP by vertical and horizontal integration due to process maturity over a span of time and learning cycles. Schuh et al. [59]

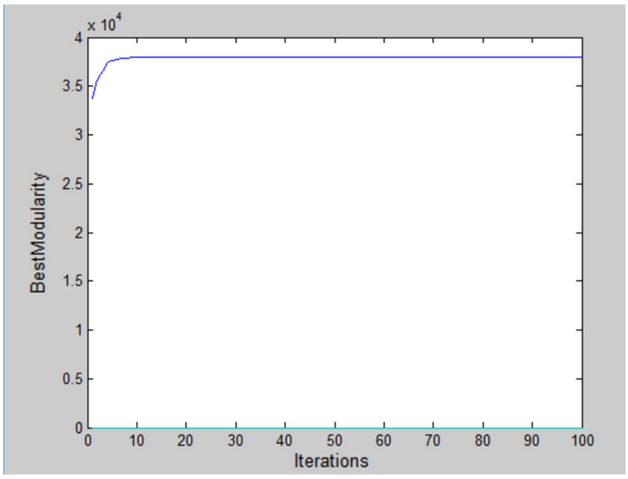

**Fig. 20** Modularity maximization (fast learning)

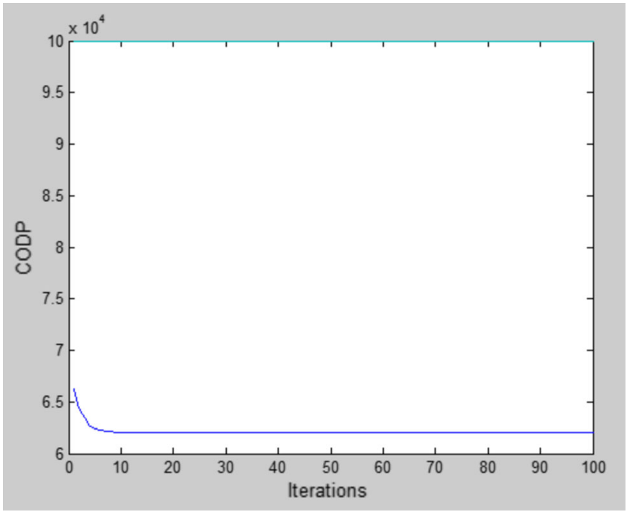

**Fig. 21** CODP optimum (fast learning)

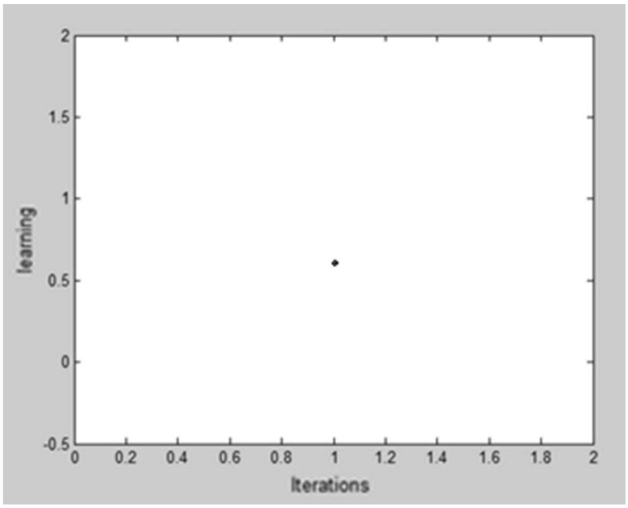

**Fig. 22** Learning achieved (fast learning)

explained vertical and horizontal integration in new software product development.

In Treatment 3, we had following parameter values assumed.

Complexity = 100,000.
$w_{\text{damp}} = 0.98$; (dampening factor).
$a = 0.3$; (self-acceleration coefficient).
$b = 0.6$; (global acceleration coefficient).

Modularity was maximized to 18,910.699 and CODP optimized to 81,089.3 after 70 iterations (refer to Figs. 16, 17, respectively).

In treatment 3, we set an upper bound using complexity value, controlled values of variables $\text{Va}_{\text{Min}}$, $\text{Va}_{\text{Max}}$, acceleration coefficients a and b. We optimized CODP after maximizing modularity and reducing the complexity (like in FF run), as shown in Fig. 16. Sensitivity analysis showed that the swarm's global acceleration coefficient has high influence on maximizing modularity. In real-world sourcing problems, higher collaboration and search in a larger space could yield the best fit results is the interpretation for this phenomenon. Self-acceleration coefficient and range of $\text{Va}_{\text{Min}}$ and $\text{Va}_{\text{Max}}$ variables selected can also impact significantly in improving modularity. Hence depending on the required complexity of product or process design, suitable search spaces and search criteria can be set to achieve optimized results.

## 4.3 Incorporation of learning functions into optimization

In this subsection, we factor a learning lag into the modularity function during PSO run. This is because any arbitrary module selection through universal source will involve learning and qualifications. Complexity of a step level advancement in technology is high or beyond reach due to knowledge barrier, time and cost. So, complexity barrier can also be linked to the cost of learning. However, since we perform the PSO optimization using quantified features which are looked up from search matrices of supplier's offering, we assume learning as a delay in picking up and matching some features in achieving modularity instead of the cost in this iteration.

Learning can act as an important factor in complexity reduction by lowering the engineering hours needed, to speed up production. This can reduce the value of numerator in the complexity expression of Willner et al. [60], which typically focuses on higher denominator of pieces produced or demand, for complexity reduction. However, a lag or delay in learning can slow this process. We make use of a planned learning function and an actual learning measurement. The planned learning lag is used as follows to modify the modularity function. The new modularity function is defined as

**Table 5** Summary of run time and results for PSO runs

| EA category | Problem type | Figure numbers | Average run time | Output | Results | Matlab code |
|---|---|---|---|---|---|---|
| Particle swarm optimization | Modularity maximization | 12, 13 | 0.413 s | Max modularity = 68.72 | Modularity maximizes depending on 'a' value, 'b' value and variable sizes defined in search space | Appendix 3 |
| | | 14 | 0.395 s | Max modularity = 912.07 | | |
| | CODP optimization and modularity maximization | 15, 16, 17 | 1.600 s | Max modularity = 18,910.6<br>CODP = 81,089.3 | Modularity nears complexity, CODP optimized with reference to the set complexity value | Appendix 4 |
| | CODP and modularity optimum with learning function $\log(x^2)$ | 19, 20, 21, 22, 23 | 0.947 s | Max modularity = 37,952.1<br>CODP = 62,047.1<br>Actual learning = 0.611 | Modularity maximizes and CODP optimizes with fast learning | Appendices 5, 6 |
| | CODP and modularity optimum with learning function $= 1 + (\log2(x))$ | Similar output as Figs. 19, 20, 21, 22 and 23 | 0.989 s | Max modularity = 39,266.8<br>CODP = 60,733.2<br>Actual learning = 0.646 | Modularity maximizes and CODP optimizes with fast learning | Appendices 5, 6 |
| | CODP and modularity optimum with Wright's learning function $= 10{,}000 + \log2(x)$ | 23, 24, 25, 26 | 1.091 s | Max modularity = 0<br>CODP = 0<br>Actual learning = 0 | Nil optimization of CODP and modularity, Zero learning | Appendix 7 |
| | CODP and modularity optimum with Wright's learning function $= 2900 + 50 * \log2(x)$ | 23, 26, 27, 28 | 1.735 s | Max modularity = 980<br>CODP = 99,020<br>Actual learning $\sim$ 0.1 | Optimization of CODP and maximization modularity with steady learning progress | Appendix 8 |

$$z = \text{sum}\left(x - \log(x)^\wedge 2\right) \quad (4)$$

We also add an actual measurement of learning achieved post-optimization,

$$\text{Learning} = \text{Complexity}/(\text{Complexity} - \text{BestModularity}) \quad (5)$$

We name this iteration as treatment 4. We assume a learning associated delay in achieving modularity, a gap which means an absolute fit of module isn't necessarily possible and syncing up of two different organizations or types of processes needs a qualification and aligning fitness.

Following lines are added and the code is modified. Actuals-based learning is used to modify the inertia coefficient.

$$\text{Learning} = \text{BestModularity}(it)/(\text{Complexity} - \text{BestModularity}(it)); \quad (6)$$

$$w = w * w_{\text{damp}} * \text{learning}; \quad (7)$$

Figure 19 shows the algorithm for modularity maximization with learning lag. The learning function $\log(x)^\wedge 2$ used here gives fast learning compared to other learning functions which follow in this paper. Appendix 5 in Supplementary Information shows the MATLAB code for this algorithm.

Results showed that addition of the learning function doesn't necessarily slow down the optimization. The higher value of modularity achieved is dependent on the $\text{Va}_{\text{Max}}$ limit and accordingly helps the CODP to shift closer to upstream of the process flow. Initial learning lag too doesn't have any significant effect as eventually the search evolves and improves modularity.

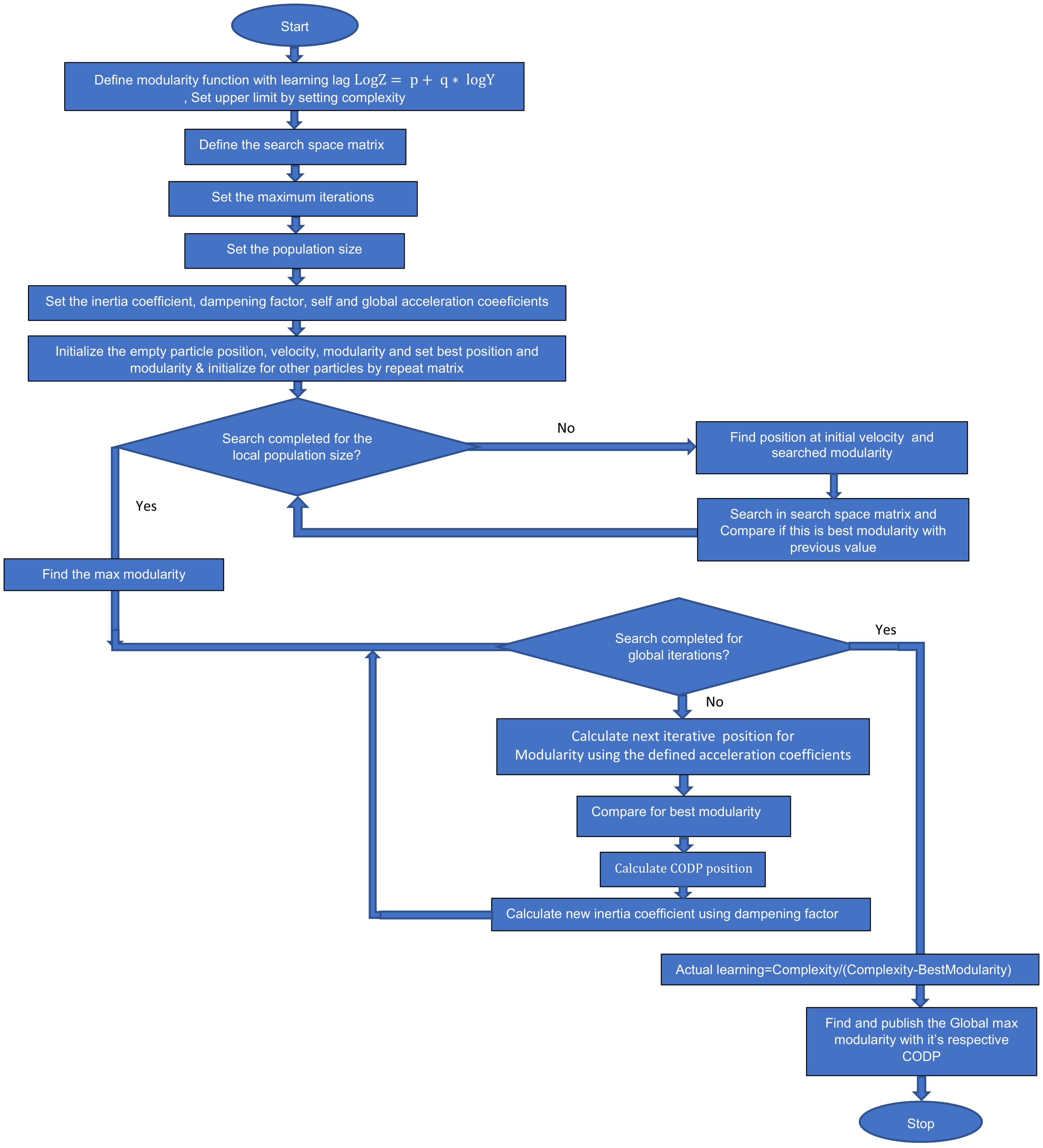
Start
Define modularity function with learning lag LogZ = p + q ∗ logY , Set upper limit by setting complexity
Define the search space matrix
Set the maximum iterations
Set the population size
Set the inertia coefficient, dampening factor, self and global acceleration coeeficients
Initialize the empty particle position, velocity, modularity and set best position and modularity & initialize for other particles by repeat matrix
Search completed for the local population size?
No
Find position at initial velocity and searched modularity
Search in search space matrix and Compare if this is best modularity with previous value
Yes
Find the max modularity
Search completed for global iterations?
Yes
No
Calculate next iterative position for Modularity using the defined acceleration coefficients
Compare for best modularity
Calculate CODP position
Calculate new inertia coefficient using dampening factor
Actual learning=Complexity/(Complexity-BestModularity)
Find and publish the Global max modularity with it's respective CODP
Stop

**Fig. 23** PSO algorithm with Log $Z = p + q * \log Y$ learning, modularity maximization and CODP positioning

From iteration 43, Best Modularity = 37,952.1121 and continues at same optimized value till iteration 100.

Learning (actual learning achieved) = 0.6117.

Sensitivity analysis showed that planned learning $z = \mathrm{sum}(x - \log(x^2))$ doesn't have significant impact on lowering speed or efficiency of learning. Also, actual learning varied from 0.056 to 1.0032 didn't have any significant impact. This is because $\mathrm{Va}_{\mathrm{Max}}$, $a$, $b$, $w$, $w_{\mathrm{damp}}$ have higher impact due to the global availability of faster solutions by the search compared to the slower learning process.

Lu et al. [61] studied various models for incorporating learning effect like Anderson and Parker's model, and Gu

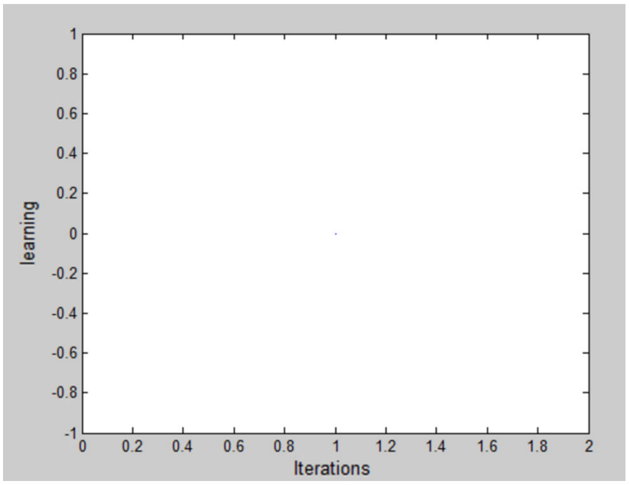

**Fig. 24** Learning results (zero learning)

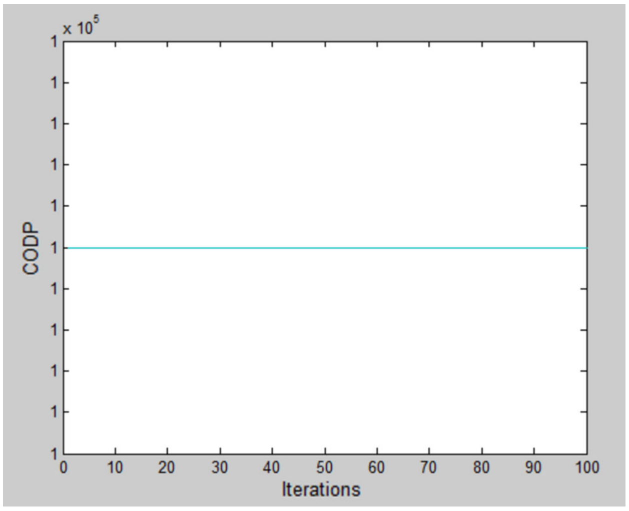

**Fig. 25** Zero optimization of CODP (zero learning)

and Takahashi's model with respect to buying decisions of components, learning curves comprehending complexity, etc. They concluded that Wright's function (published in 1936) for learning continues to be excellent for forecasting the processes of future.

The function is

$$\operatorname{Log} Z = p + q * \log Y \tag{8}$$

We adopted this relationship in our Eq. (8) by adopting suitable values of $p$, $q$ and $Y$. Figure 23 demonstrates the algorithm which deploys this learning function. Firstly, we made use of $1 + (\log 2(x))$, then $10{,}000 + \log\, 2(x)$ and lastly $2900 + 50 * \log 2(x)$, which are explained with their MATLAB programs (in Appendices 6, 7 and 8 in Supplementary Information, respectively), and respective results. These learning functions use logarithmic relation with base 2. The results of using $1 + (\log 2(x))$ as the

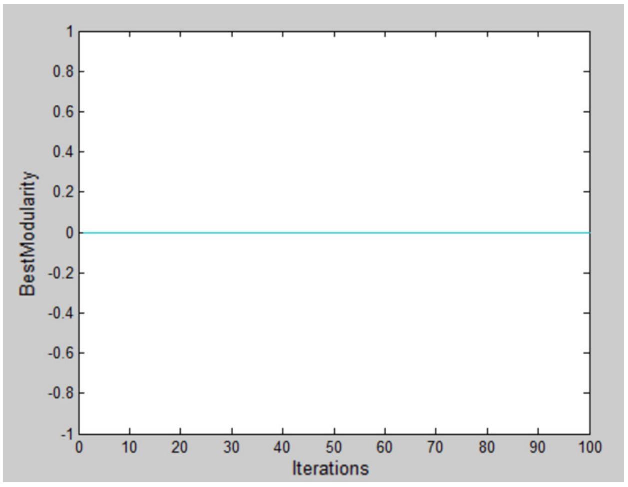

**Fig. 26** Negligible optimization of modularity (zero learning)

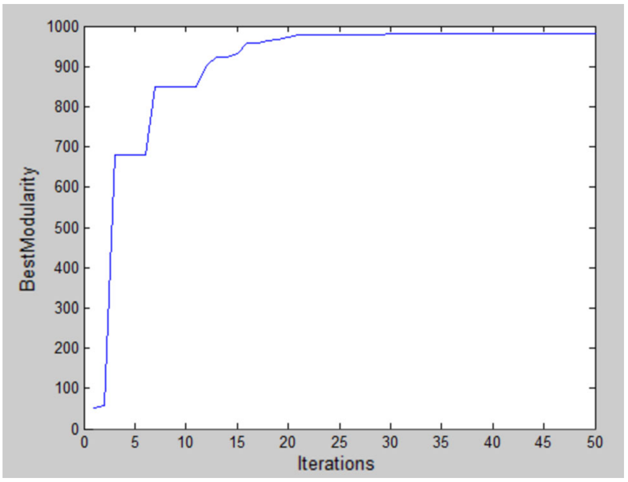

**Fig. 27** Modularity maximized with learning lag (medium learning)

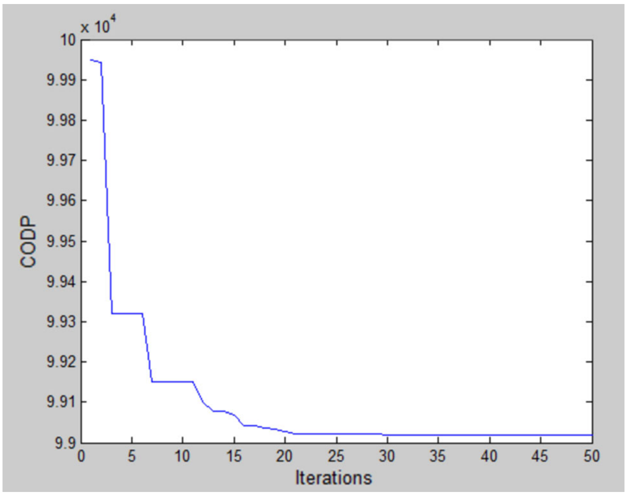

**Fig. 28** CODP optima with learning lag (medium learning)

learning function were like the results of the fast learning iteration performed with that using the learning function $\log(x^\wedge 2)$.

The learning function $10{,}000 + \log2(x)$ simulates the worst case of learning (zero learning) (refer to Figs. 24, 25). The learning function $2900 + 50 * \log 2(x)$ shows the catching up type of learning (medium learning) (refer to Figs. 26, 27, 28) with desirable results. This is applicable for slower learning projects unlike the previous fast learning functions $\log(x^\wedge 2)$ and $1 + (\log 2(x))$, which are explained earlier in Figs. 20, 21 and 22.

Further, we identified that a limit of $Z = 10{,}000 + \log2(x)$ gave a high learning lag with no learning progress. Figures 24, 25 and 26 show that learning is zero with no increase of modularity in the results, and zero optimization of CODP.

Finally, we found a variant of Wright's learning function for increase delay of optimization after trial and error. Further to that, the simulation changed from $< 10$ iterations, to 30 iterations for achieving CODP optimization.

Table 5 shows the average run times and results for PSO run scenarios explained above and helps to make their comparisons. It also shows the respective output values obtained for experimental runs of the parameter inputs shown in Table 3.

Yusof and Deris [50] made use of conventional GA technique for cost and risk minimization for machine constraint "environment. According to them, the conventional GA technique depends on mutation which causes creation of some infeasible chromosomes that makes computation slow unless we eliminate those. This elimination reduces the opportunities of solution sets and may not be applicable for the best fit approach used by us as part of modularity matching. Borna and Khezri [62] created a heuristic algorithm by combining crossover operator of GA into PSO for traveling salesman problem. Sharma and Singhal [63] tested a hybrid GA-PSO algorithm on five test functions by feeding the results of the GA algorithm into PSO algorithm input. They mentioned that this hybrid method helps to overcome the limitations of both individual methods for different population sizes. The population size used by them is 30 which is significantly small compared to the population size used in the algorithms here as the objective here is to do best fit search from larger population.

Gen et al. [64] focused on hybrid evolutionary algorithms and multi-objective hybrid genetic algorithms for solving multi-objective optimization problem in the area of scheduling. They target a computation time of $< 3$ min. These methods are not studied for their application into CODP optimization and need to be explored further. As mentioned, the application of algorithms as explained in this paper uses complementary objectives of modularity maximization and CODP optimization and we get output within a few seconds. Our FF runs had $< 6$ s run time, while PSO runs had $< 2$ s run time. The only exception to this is the case of 10,000 iterations used in FF technique for maximizing modularity which took 539.86 s but reduced to 5.19 s when iterations were limited to 1000.

Module outsourcing can include outsourcing additive manufacturing facility or 3D printing also who already have technology to use the new material in the printing.

The basic concept and results are similar in the FF and PSO algorithms used here. The FF code is easier to understand, while the PSO is more adapted to real-world sourcing problems and production process chain design. The salient feature of PSO is that we were able to model multi-dimensional search space and calculate aggregate modularity score using summation, to make comparison between required specifications and available modules searched. The codes shown are flexible to adapt into different real-world situations.

These codes can be incorporated into SMC through qual module illustrated in Fig. 31 after breaking the complexity into modular form as shown in Fig. 30. This is explained in the next section.

## 5 Models for modularity search and complexity reduction

A new complex requirement from customer may trigger several other complexities throughout the supply chain which can impact the manufacturing machine setups, staffing, programs, locations and the process flows. In this section, we explain a few steps for connecting the earlier

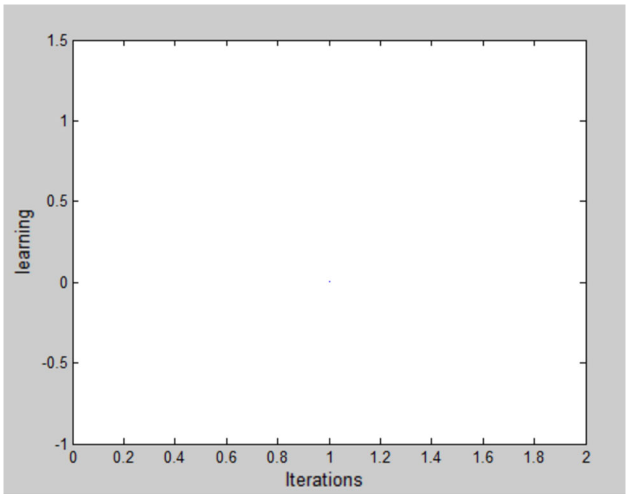

**Fig. 29** Learning value $\sim$ 0.1 (medium learning)

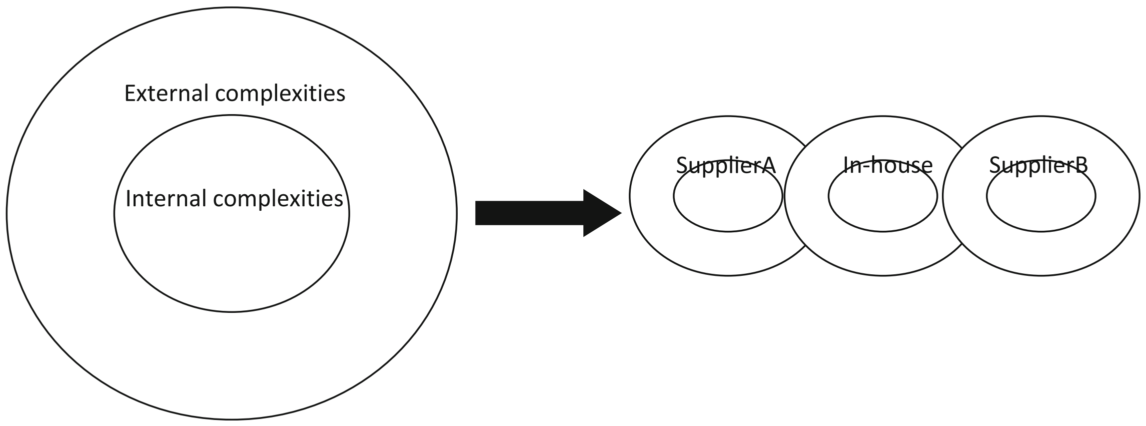

**Fig. 30** Complexity reduction by separating internal and external modules

explained EA techniques to the real-world problems encountered in process design.

## 5.1 Splitting internal and external complexities

To reduce the time to market and to gain the competitive advantage of a novel customer need, it is recommended to divide product or process complexity by splitting into internal and external complexities. This is represented by the left side of Fig. 30. The next step to be followed is to break the internal and/or external complexity clearly into different independent manufacturable or sourceable modules. These modules should then be addressed through internal departments or cross functional teams which can recommend to outsource some of the low risk but high time consuming activities. Such outsourcing can be through qualification of external supply chains by signing suitable agreements or licensing to avoid breach of novelty. It is recommended to outsource the external complexities to outside organizations from the market, which have expertized in dealing with those. As explained above, the social or global acceleration coefficient value is high in such cases and can aid in achieving higher value of modularity with ease.

If a direct fit isn't available in the global search space, then the closest fit source can be identified and qualified for external or internal complexity reduction. Our paper mentions the optimization of CODP position and process flow design using genetic algorithm by smartly joining the best fit steps from in-house and/or external sources.

Practical examples of sourced modularity are mentioned here. Versions of ARM® cortex belonging to ARM Holdings (a semiconductor company) are incorporated and programmed by other semiconductor companies in their design and manufacturing, for achieving complexity reduction [65]. ARM licenses its intellectual property which is utilized by various semiconductor companies like Toshiba who utilize those design blocks and incorporate them into their individual custom products. Hino engines of Toyota group are used by Ashok Leyland Trucks in India [66]. Fiat engines are fit in most of the Maruti Suzuki diesel cars and Tata Motors, in India [67]. The examples shown here are manually done; however, in a smart environment, the sourcing should be through artificial intelligence (AI)-based match using EA method. The concept explained in this paper is usable for production of medical equipment as well. Quick process flow design for assembling support equipment to treat COVID-19 can be achievable.

## 5.2 Usage of smart models

Once the modules are identified, the next step is to source them or to fabricate them to fulfill the customer need. Figure 31 illustrates a smart environment to achieve the same. An IoT-based interface is shown as the input of customer expectations which can be studied for viability through interactive and concurrent communication. The feedback loop and choice navigation help in perfecting the synchronization between need and feasibility to deliver. The cloud-based information storage of available modules through licensing or collaboration can help in further development through involvement of the design team. It may be helpful right from the problem statement or decision of features to be offered while coming up with new design. Even between different companies or within same company which have widespread teams of expertise but have scattered knowhow, this method could help access a pool of information.

A EA-based qual module is explained here to address the fitness of the identified modules. Open specifications by licensing fee, protocols and rules for smart enabled fitness matching of process modules need to be defined to enable this. Qual module should be empowered to take decision in weeding out unfit modules from the real-world supply

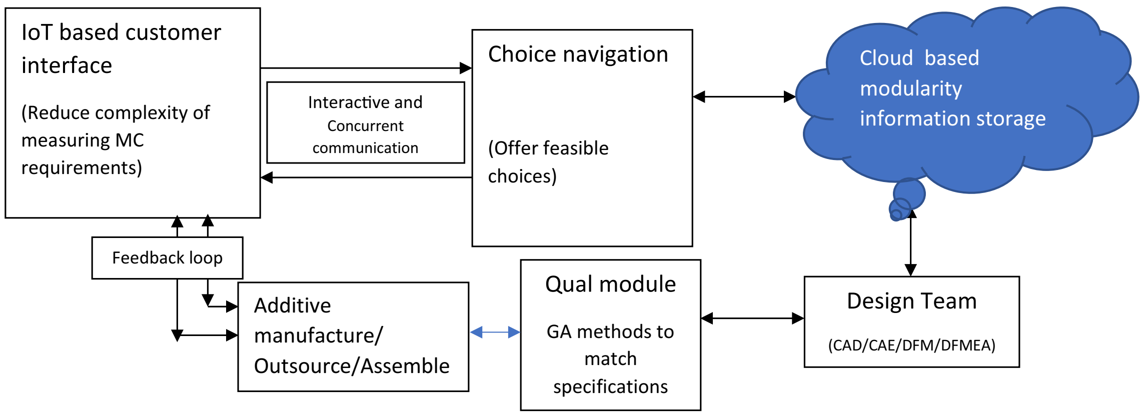
IoT based customer interface
(Reduce complexity of measuring MC requirements)
Interactive and Concurrent communication
Choice navigation
(Offer feasible choices)
Cloud based modularity information storage
Feedback loop
Additive manufacture/ Outsource/Assemble
Qual module
GA methods to match specifications
Design Team
(CAD/CAE/DFM/DFMEA)

**Fig. 31** Cloud- and IoT-based smart mass customization

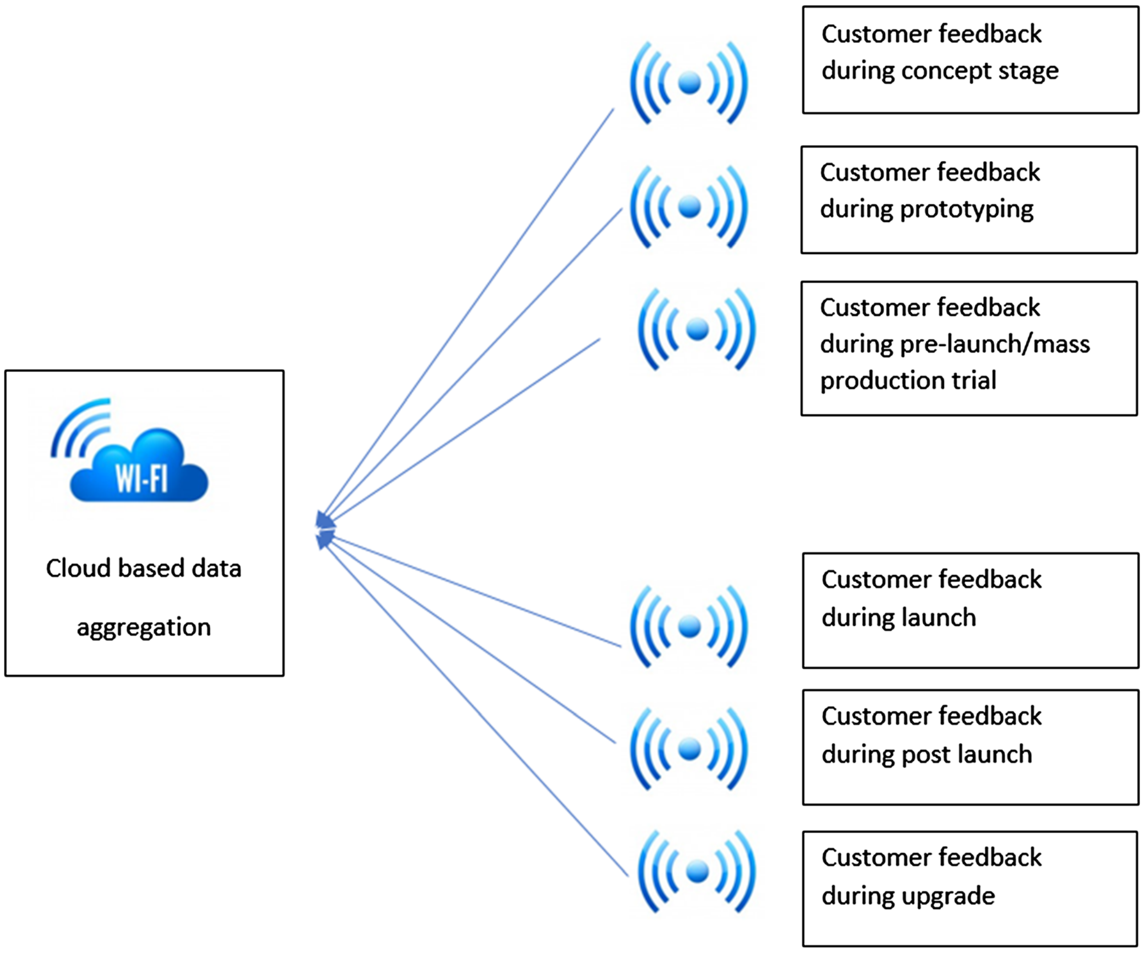
Customer feedback during concept stage
Customer feedback during prototyping
Customer feedback during pre-launch/mass production trial
WI-FI
Cloud based data aggregation
Customer feedback during launch
Customer feedback during post launch
Customer feedback during upgrade

**Fig. 32** Stages in the IoT cum Wifi customer interface

chain. Real-world problems have to be mapped and validated while implementing the EA-based process flow design in the virtual world. IP protection barrier and protection of technical knowhow from R&D investments, i.e. IP, needs to be protected.

Our FF- and PSO-based techniques explained above could be deployed here to perform the search. Additive manufacture, outsourcing and final assembly step can help to complete the manufacturing process.

Figure 32 explains the detailed steps involved in the IoT-based customer interface module explained in Fig. 31. The cloud-based date aggregation shown in Fig. 32 can be fed into the cloud-based modularity information storage shown in Fig. 31.

## 6 Conclusion

In this paper, we studied and proposed the opportunities for optimization of customer order decoupling point in smart custom manufacturing. We identified the gap in existing research and tried to find a direction towards closing the missing link between the position of CODP and SMC, through evolutionary algorithms. We tried to achieve this objective using fruit fly and particle swarm-based evolutionary techniques for modular sourcing of parts and sub processes, in manufacturing process design. Through MATLAB coding, we created a solution for reducing complexity with increasing modularity which evolves into finding an optimum CODP position. Figure 1 shows evolution of CODP research and the directions it took in the past four decades starting from its inception to the emerging era of smart manufacturing. Figures 4, 5, 7, 8, 9, 10 and 11 show the different combinations of influencing parameters and results of maximizing modularity, while also optimizing CODP position using FF method. The FF-based method is a simpler technique for sourcing modules in one dimensional search space internally and externally. The CODP optimization is achieved within about 1000 iterations based on the set parameters.

The PSO method is closer to real-world problems with multi-dimensional, multi-attribute based search. The results looked identical to FF algorithm; however, the results got achieved within 40–70 iterations (Figs. 13, 14, 16, 17). The value of modularity can be maximized in PSO algorithm, by using higher values of self-acceleration coefficient, social acceleration coefficient, damping factor, and defined variable ranges ($\mathrm{Va}_{\mathrm{Min}}$ and $\mathrm{Va}_{\mathrm{Max}}$). It was demonstrated in the sensitivity analysis that the global (social) acceleration coefficient has a higher impact in maximizing modularity compared to the other influencing factors. This simulates the real-world scenario to achieve increased smartness globally in manufacturing

The managerial interpretation of the achieved results is as follows. The model shows that maturity of external environment plays a vital role in sourcing modular portions of manufacturing process flow in SMC. It shows real-world scenario of external environment being able to influence easier decision making in sourcing due to availability of mature module suppliers in the solution space. The self-acceleration coefficient has the next level in terms of impact in maximizing modularity. Self-driven approach is the basic direction setter which leads to initiation of a smart module-based customization search but gets accelerated by availability of better options globally.

In PSO, we explained a MATLAB code which simulates and unbounded maximization for modularity which is analogous to Moore's law and further shows possibilities of unified CODP for diverse product portfolio, while helping it to evolve (Fig. 18). We also made use of the upper limit in modularity maximization and CODP achievement within it by setting specific complexity value to control the results (Fig. 17).

We then modified the simulation using learning lag functions in Sect. 4.3. A method to measure actual learning was also shown in the same. This section represents learning and maturity of any given technology overtime. The learning lag using $\log(x)$^2 did not slow down optimization. It showed fast learning. Figures 20, 21 and 22 show that optimization completes within about 10 iterations. Initial sensitivity analysis of learning showed that planned learning $z = \mathrm{sum}(x) - \log(x\text{^}2)$ doesn't have significant impact on lowering speed of learning. Also, actual learning varied from 0.056 to 1.0032 didn't have any significant impact. This is also because $\mathrm{Va}_{\mathrm{Max}}$, $a$, $b$, $w$, $w_{\mathrm{damp}}$ have higher impact due to the global availability of faster solutions by the search compared to the slower learning process.

We then explored impact of Wright's learning function in the last two simulations which show the sensitivity in learning when applied to PSO model. Figures 24, 25 and 26 illustrate the case where learning postpones modularity maximization indefinitely with zero learning and no optimization of CODP. Finally, in Figs. 27, 28 and 29, we modified the values in Wright function through random generation of $p$ and $q$ values which provided the optima though with a learning lag that slowed down optimization.

The benefits of above models can be achieved through full modularity sourced externally and internally as explained in Figs. 30, 31 and 32. In Sect. 5, we suggested few steps to reduce complexity of processes and process through splitting of internal and external complexities into modules. We illustrated splitting of the modules into internal and external sources in Fig. 30, which could be matched through a qual module in SMC environment using IoT and cloud-based system, as shown in Fig. 31. In Fig. 32, we show the customer involvement through WiFi and IoT which can be fed into the cloud-based modularity information storage of Fig. 31.

Comparability aspects need to be defined in the qual module. Comparison needs to be performed to assess similarity and fitness of available product and process modules while finding matches in the module search. Search from internet, collaborative consortia, collaborative networks or verification with nearby suppliers with licensing rules can enable the process. Modularity needs to be achieved through exchange of information about required and available module specifications in SMC. New bill of materials can be scrutinized for outsourcing or in-house breakthroughs using this concept.

With emerging technology, higher computing capabilities can be achieved which get imbibed into evolving modules as explained in Moore's law. This is demonstrated by the high values of modularity which is shown as achievable in our optimization results. Few examples are LED Television manufacturing, where only the labeling and packing are left in downstream processes as the upstream processes are mostly matured and modularized. Semiconductor manufacturing where part of the technology or modules may be outsourced and combined with one's own product to create faster processing solutions and new applications. Hence, the CODP shifts from left to right as the process matures. However, new advancements trigger addition of newer downstream steps in the process flow to manufacture next generation of products. With this, CODP moves back towards upstream. To conclude, a flexible CODP is needed for higher agility and flexibility, to enable evolution in smart custom manufacturing.

In this and next paragraphs, we discuss limitations and directions for future research. Our paper provides a framework for smart built supply chain of the future, through recommended EA-based methods in manufacturing process chain design. We achieved two objectives with the FF- and PSO-based methods. Firstly, we provided modularity maximization. Secondly, we optimized CODP by minimizing complexity through the maximized modularity. Less complexity doesn't essentially mean lowered MC, but lessened complexity may help in improving MC and increasing customized variants. Smart enabled concepts are illustrated stepwise in Sect. 5 which can help in practical implementation of the modularity search thereby to optimize CODP. We tried to explain and connect the gap between EA and IoT model to connect manufacturer and supplier for efficient knowledge exchange, but the technical and licensing modalities need to be addressed.

Shift of CODP towards right with increased standardization can force custom requirements into a generic flow. Hence, the chances of error (in customization) are high and will be discovered only during final assembly or after final assembly causing efficiency to drop. This needs to be further investigated. The sensitivity of learning functions studied and illustrated from Figs. 20, 21, 22, 24, 25, 26, 27, 28 and 29 needs to be to be related to real-world situations to get good results in practice. Other learning functions and scenarios using Nesterov's accelerated gradient needs to be studied. We recommend extending the EA techniques from MATLAB to Python language for open sourcing to facilitate machine learning to Automate and expand into creating a real global SMC environment. This is because as explained earlier, global acceleration coefficient can help drive SMC faster.

Inventory implications of this result needs to be studied in future research. A poorly selected modularity may leave CODP at the leftmost point of the supply chain causing huge variety of incoming inventory at raw material and wip stage. It's because there can be highest level of uncertainty and lack of standardization due to the lack of knowledge and poor clarity on fitness to use of the raw material and wip inventory. On the contrary, if higher standardization is not accepted in terms of customization options by the consumer, it could lead to high unsold finished goods inventory. MC involves limited solution space (by definition); hence, in a universal search space, the manufacturing flow line can have CODP move towards an optimum downstream position to achieve ideal goal. This could lead production machines till downstream steps to get more standardized. At the same time, efficient SMC would mean efficient processes which could convert one product or base part to other. The inventory-based supply should be more on certain variant which has stable demand with options of customizing it to the variants. So, inventory risk on variants should ideally get reduced. This needs to be coded as a SMC objective into this problem. Thus, an efficient MC would mean a robust product towards right of CODP which could then be customized to final step by just a single process change. The methodology needs to be further expanded for autonomous decision making while identifying and selecting suitable modules as proposed by Ma et al. [68]. This can support further evolution into Industry 5.0 applications.

Our investigation is based on absolute values of complexity and modularity. In real-world situations, it may be a challenging task to quantify these parameters into absolute numbers. There is some interesting prior research available on degree of customization. The methods shown in our paper need to be linked to the same, in future studies. We assume in our fitness function that complexity reduces with increase in modularity. Thus, we subtract absolute value of searched and selected modularity from complexity value to shift the CODP to its optimum position. The fitness function can be suitably modified based on empirical research for future studies. Also, the shift of CODP needs to be researched from the perspective of horizontal and vertical integration during merger and acquisitions.

**Code availability** MATLAB codes are provided in appendices in Supplementary Information.

## Compliance with ethical standards

**Conflict of interest** All authors declare that they have no conflict of interest.

## References

1. Alcácer V, Machado VC (2019) Scanning the Industry 4.0: a literature review on technologies for manufacturing systems. Eng Sci Technol Int J 22(3):899–919. https://doi.org/10.1016/j.jestch.2019.01.006
2. Frank AG, Dalenogare LS, Ayala NF (2019) Industry 4.0 technologies: implementation patterns in manufacturing companies. Int J Prod Econ 210:15–26. https://doi.org/10.1016/j.ijpe.2019.01.004
3. Mittal S, Khan MA, Romero D, Wuesta T (2018) A critical review of smart manufacturing and Industry 4.0 maturity models: implications for small and medium-sized enterprises (SMEs). J Manuf Syst 49:194–214. https://doi.org/10.1016/j.jmsy.2018.10.005
4. Hajrizi E (2016) Smart solution for smart factory. IFAC-Papers OnLine 49(29):001–005. https://doi.org/10.1016/j.ifacol.2016.11.052
5. Zhang C, Chen D, Tao F, Liu A (2019) Data driven smart customization. Proc CIRP 81:564–569. https://doi.org/10.1016/j.procir.2019.03.156
6. Lehmhus D, Kopp CA, Petzoldt F, Godlinski D, Haberkorn A, Zöllmer V, Busse M (2016) Customized Smartness: a survey on links between additive manufacturing and sensor integration. Proc Technol 26:284–301. https://doi.org/10.1016/j.protcy.2016.08.038
7. Suginouchi S, Kokuryo D, Kaihara T (2017) Value co-creative manufacturing system for mass customization: concept of smart factory and operation method using autonomous negotiation mechanism. Proc CIRP 63:727–732. https://doi.org/10.1016/j.procir.2017.03.313
8. Daaboul J, Da Cunha CM (2014) Differentiation and customer decoupling points: key value enablers for mass customization. In: Grabot B et al (eds) APMS 2014, Part III, IFIP AICT 440. IFIP international federation for information processing, pp 43–50, Springer. https://doi.org/10.1007/978-3-662-44733-8_6
9. Fogliatto FS, DaSilveira GJC, Borenstein D (2012) The mass customization decade: an updated review of the literature. Int J Prod Econ 138:14–25. https://doi.org/10.1016/j.ijpe.2012.03.002
10. McCall J (2005) Genetic algorithms for modelling and optimization. J Comput Appl Math 184(1):205–222. https://doi.org/10.1016/j.cam.2004.07.034
11. Wang SY, Chang SL, Wang RC (2009) Assessment of supplier performance based on product-development strategy by applying multi-granularity linguistic term sets. Omega 37:215–226. https://doi.org/10.1016/j.omega.2006.10.003
12. Giesberts PMJ, Van Den Tang L (1992) Dynamics of the customer order decoupling point: impact on information systems for production control. Prod Plan Control 3(3):300–313. https://doi.org/10.1080/09537289208919402
13. Rudberg M, Wikner J (2004) Mass customization in terms of the customer order decoupling point. Prod Plan Control 15(4):445–458. https://doi.org/10.1080/0953728042000238764
14. Ethiraj SK, Levinthal D (2004) Modularity and innovation in complex systems. Manag Sci 50(2):159–173. https://doi.org/10.1287/mnsc.1030.0145
15. Wikner J, Rudberg M (2005) Integrating production and engineering perspectives on the customer order decoupling point. Int J Oper Prod Manag 25(7):623–641. https://doi.org/10.1108/01443570510605072
16. Wikner J, Wong H (2007) Postponement based on the positioning of the differentiation and decoupling points. In: Olhager J, Persson F (eds) IFIP International federation for information processing 246, Advances in production management systems. Springer, Boston, pp 143–150. https://doi.org/10.1007/978-0-387-74157-4_17
17. Xu XG (2007) Position of customer order decoupling point in mass customization. In: Proceedings of the sixth international conference on machine learning and cybernetics, Hong Kong, IEEE. https://doi.org/10.1109/ICMLC.2007.4370159
18. Hua JJ, Li Q, Lun GQ (2007) Study on CODP position of process industry implemented mass customization. Syst Eng Theory Pract 27(12):151–157. https://doi.org/10.1016/S1874-8651(08)60079-4
19. Luo JQ, Han YQ, Zhou X (2008) Positioning of CODP based on entropy technology and ideal point principle. In: 4th international conference on wireless communications, networking and mobile computing, Dalian, IEEE. https://doi.org/10.1109/WiCom.2008.1482
20. Liu D, Wang W, Fu W, Liu D (2009) CODP position of leagile supply chain based on polychromatic sets theory. Proc IEEE Int Conf Autom Logist. https://doi.org/10.1109/ICAL.2009.5262884
21. Ge J, Wei F, Huang Y, Gao G (2009) Research on customer order decoupling point positioning model for supply chain cost optimization. In: Proceedings of the IEEE international conference on automation and Logistics, Shenyang, IEEE. https://doi.org/10.1109/ICAL.2009.5262581
22. Brun A, Zorzini M (2009) Evaluation of product customization strategies through modularization and postponement. Int J Prod Econ 120:205–220. https://doi.org/10.1016/j.ijpe.2008.07.020
23. Daaboul J, Laroche F, Bernard A (2010) Determining the CODP position by value network modeling and simulation. In: International technology management conference (ICE), IEEE, Lugano. https://doi.org/10.1109/ICE.2010.7476995
24. Olhager J (2010) The role customer order decoupling point in production and supply chain management. Comput Ind 61:863–868. https://doi.org/10.1016/j.compind.2010.07.011
25. Da Cunha C, Agard B, Kusiak A (2010) Selection of modules for mass customization. Int J Prod Res 48(5):1439–1454. https://doi.org/10.1080/00207540802473989
26. McIntosh RI, Matthews J, Mullineux G, Medland AJ (2010) Late customisation: issues of mass customisation in the food industry. Int J Prod Res 48(6):1557–1574. https://doi.org/10.1080/00207540802577938
27. Buffington J (2011) Comparison of mass customization and generative customization in mass markets. Ind Manag Data Syst 111(1):41–62. https://doi.org/10.1108/02635571111099721
28. Qin Y (2011) On delaying CODP to distribution center in mass customization. In: Shen G, Huang X (eds) Communications in computer and information science 152. Advanced research on computer science and information engineering, international conference. CSIE, Springer, Heidelberg, pp 271–276. https://doi.org/10.1007/978-3-642-21402-8_44
29. Xu X, Liang Z (2011) CODP Positioning based on extension superiority evaluation model. In: International conference on electronic and mechanical engineering and information technology, Harbin. IEEE. https://doi.org/10.1109/EMEIT.2011.6023940
30. Bask A, Lipponen M, Rajahonka M, Tinnila M (2011) Framework for modularity and customization: service perspective. J Bus Ind Market 26(5):306–319. https://doi.org/10.1108/08858621111144370
31. ElMaraghy W, ElMaraghy H, Tomiyama T, Monostori L (2012) Complexity in engineering, design and manufacturing. CIRP Ann Manuf Technol 61:793–814. https://doi.org/10.1016/j.cirp.2012.05.001
32. Jeong IJ (2011) A dynamic model for the optimization of decoupling point and production planning in a supply chain. Int J Prod Econ 131:561–567. https://doi.org/10.1016/j.ijpe.2011.02.001

33. Lin J, Shi X, Wang Y (2012) Research on the hybrid push/pull production system for mass customization production. In: Shaw MJ, Zhang D, Yue WT (eds) E-life: web-enabled convergence of commerce, work, and social life. Springer, pp 413–420. https://doi.org/10.1007/978-3-642-29873-8_38
34. Medini K, Da Cunha C, Bernard A (2012) Sustainable mass customized enterprise: key concepts, enablers and assessment techniques. IFAC Proc 45(6):522–527. https://doi.org/10.3182/20120523-3-RO-2023.00242
35. Kim JI, Kim SH (2012) Positioning a decoupling point in a semiconductor supply chain under demand and lead time uncertainty. Int J Adv Logist 1(2):31–45. https://doi.org/10.1080/2287108X.2012.11006075
36. Mehrsai A, Karimi HR, Thoben KD (2013) Integration of supply networks for customization with modularity in cloud and make-to-upgrade strategy. Syst Sci Control Eng An Open Access J 1(1):28–42. https://doi.org/10.1080/21642583.2013.817959
37. Agrawal T, Sao A, Fernandes KJ, Tiwari MK, Kim DY (2013) A hybrid model of component sharing and platform modularity for optimal product family design. Int J Prod Res 51(2):614–625. https://doi.org/10.1080/00207543.2012.663106
38. Sjøbakk B, Thomassen MK, Alfnes E (2014) Implications of automation in engineer-to-order production: a case study. Adv Manuf 2:141–149. https://doi.org/10.1007/s40436-014-0071-4
39. Wikner J (2014) On decoupling points and decoupling zones. Prod Manuf Res An Open Access J 2(1):167–215. https://doi.org/10.1080/21693277.2014.898219
40. Wikner J (2014b) Supply chain management strategies in terms of decoupling points and decoupling zones. In: Grabot B et al (eds) APMS, Springer, Berlin, pp 371–378. https://doi.org/10.1007/978-3-662-44739-0_45
41. Ngniatedema T, Fono LA, Mbondo GD (2014) A delayed product customization cost model with supplier delivery performance. Eur J Oper Res 243(1):109–119. https://doi.org/10.1016/j.ejor.2014.11.017
42. Ridwan M, Purnomo A, Sufa MF (2015) Simulation-based performance improvement towards mass customization in make to order repetitive company. Proc Manuf 2:408–412. https://doi.org/10.1016/j.promfg.2015.07.072
43. Keddis N, Kainz G, Zoitl A, Knoll A (2015) Modeling production workflows in a mass customization era. In: IEEE international conference on industrial technology (ICIT), Seville, IEEE. https://doi.org/10.1109/ICIT.2015.7125374
44. Shahin A, Gunasekaran A, Khalili A, Shirouyehzad H (2016) A new approach for estimating leagile decoupling point using data envelopment analysis. Assemb Autom 36(3):233–245. https://doi.org/10.1108/AA-07-2015-063
45. Yao Y, Xu Y (2018) Dynamic decision making in mass customization. Comput Ind Eng 120:129–137. https://doi.org/10.1016/j.cie.2018.04.025
46. Cannas VG, Gosling J, Pero M, Rossi T (2019) Engineering and production decoupling configurations: an empirical study in the machinery industry. Int J Prod Econ 216:173–189. https://doi.org/10.1016/j.ijpe.2019.04.025
47. Tookanlou PB, Wong H (2019) Determining the optimal customization levels, lead times, and inventory positioning in vertical product differentiation. Int J Prod Econ. https://doi.org/10.1016/j.ijpe.2019.08.014
48. Zheng XL, Wang L, Wang SY (2014) A novel fruit fly optimization algorithm for semiconductor final testing scheduling problem. Knowl-Based Syst 57:95–103. https://doi.org/10.1016/j.knosys.2013.12.011
49. Zheng XL, Wang L (2016) A two-stage adaptive fruit fly optimization algorithm for unrelated parallel machine scheduling problem with additional resource constraints. Expert Syst Appl 65:28–39. https://doi.org/10.1016/j.eswa.2016.08.039
50. Yusof UK, Deris S (2009) Constraint-based genetic algorithms for machine requirement of semiconductor assembly industry: a proposed framework. In: Third Asia international conference on modelling and simulation. IEEE. https://doi.org/10.1109/ams.2009.119
51. Pan WT (2014) A new evolutionary computation: fruit fly optimization algorithm, 2nd edn. The MathWorks Textbook, Taiwan
52. Ma G, Zhang F (2012) Genetic algorithms for manufacturing process planning. In: Variants of evolutionary algorithms for real-world applications. Springer, Heidelberg, pp 205–244. https://doi.org/10.1007/978-3-642-23424-8_7
53. Saldivar AAF, Goh C, Li Y, Chen Y, Yu H (2016) Identifying smart design attributes for Industry 4.0 customization using a clustering genetic algorithm. In: Proceedings of the 22nd international conference on automation and computing, University of Essex, Colchester city, UK, IEEE. https://doi.org/10.1109/IConAC.2016.7604954
54. Zhang Z, Wang X, Zhu X, Cao Q, Tao F (2019) Cloud manufacturing paradigm with ubiquitous robotic system for product customization. Robot CIM Int Manuf 60:12–22. https://doi.org/10.1016/j.rcim.2019.05.015
55. Wan J, Li J, Hua Q, Celesti A, Wang Z (2020) Intelligent equipment design assisted by Cognitive Internet of Things and industrial big data. Neural Comput Appl 32:4463–4472. https://doi.org/10.1007/s00521-018-3725-5
56. James CD, Mondal S (2019) A review of machine efficiency in mass customization. Benchmark Int J 26(2):638–691. https://doi.org/10.1108/BIJ-05-2018-0120
57. Heris SMK (2016). Particle swarm optimization in MATLAB—Yarpiz Video Tutorial—Part 2/3, [Yarpiz]. https://www.youtube.com/watch?v=xPkRL_Gt6PI. Accessed 15 October 2019
58. Schaller RR (1997) Moore's law: past, present and future. IEEE Spectr 34(6):52–59. https://doi.org/10.1109/6.591665
59. Schuh G, Dölle C, Kantelberg J, Menges A (2018) Identification of agile mechanisms of action as basis for agile product development. Proc CIRP 70:19–24. https://doi.org/10.1016/j.procir.2018.02.007
60. Willner O, Powell D, Gerschberger M, Schönsleben P (2016) Exploring the archetypes of engineer-to-order: an empirical analysis. Int J Oper Prod Manag 36(3):242–264. https://doi.org/10.1108/IJOPM-07-2014-0339
61. Lu RF, Petersen TD, Storch RL (2009) Asynchronous stochastic learning curve effects in engineering-to-order customisation processes. Int J Prod Res 47(5):1309–1329. https://doi.org/10.1080/00207540701484921
62. Borna K, Khezri R (2015) A combination of genetic algorithm and particle swarm optimization method for solving traveling salesman problem. Cogent Math 2(1):1–13. https://doi.org/10.1080/23311835.2015.1048581
63. Sharma J, Singhal RS (2015) Comparative research on genetic algorithm, particle swarm optimization and hybrid GA-PSO. In: 2nd international conference on computing for sustainable global development, New Delhi. IEEE, pp 110–114. https://ieeexplore.ieee.org/document/7100231
64. Gen M, Zhang W, Lin L, Yun YS (2017) Recent advances in hybrid evolutionary algorithms for multiobjective manufacturing scheduling. Comput Ind Eng 112:616–633. https://doi.org/10.1016/j.cie.2016.12.045
65. Medical and Design Outsourcing (2015) https://www.medicaldesignandoutsourcing.com/toshiba-expands-line-up-of-arm-cortex-m-based-microcontrollers/. Accessed 27 November 2019
66. Ashok Leyland, Hino Motors renew partnership for Euro VI engines (2018) The Economic Times.https://auto.economictimes.indiatimes.com/news/commercial-vehicle/mhcv/ashok-leyland-hino-motors-renew-partnership-for-euro-vi-engines/61818121. Accessed 20 November 2019

67. Thakkar K (2017) Fiat India inks fresh deals to supply 2.2 lakh diesel engines to Maruti, Tata Motors. The Economic Times. https://economictimes.indiatimes.com/industry/fiat-india-inks-fresh-deals-to-supply-2-5-litre-diesel-engines-to-maruti-tata-motors/articleshow/58491966.cms?from=mdr. Accessed 24 November 2019

68. Ma A, Nassehi A, Snider C (2018) Anarchic manufacturing. Int J Prod Res 57(8):2514–2530. https://doi.org/10.1080/00207543.2018.1521534