Int J Adv Manuf Technol (2012) 60:1125–1137
DOI 10.1007/s00170-011-3670-2

ORIGINAL ARTICLE

Constraint-based simulated annealing (CBSA) approach to solve the disassembly scheduling problem

PKS Prakash · D. Ceglarek · M. K. Tiwari

Received: 13 April 2011 / Accepted: 26 September 2011 / Published online: 5 November 2011

Abstract Globalization, coupled with environmental requirements, has spearheaded new levels of requirements for product end-of-life, the last phase of product lifecycle management especially for product remanufacturing and recycling which involves product disassembly to retrieve the desired parts and subassemblies. Selection of optimal disassembly schedule is a major challenge for remanufacturing and recycling industries as it directly affects the inventory of the manufacturing unit and influences the final product cost. This paper proposes a constraint-based simulated annealing (CBSA) algorithm methodology to determine the ordering and disassembly schedule to minimize inventory level for products with general assembly product structure, i.e., taking into consideration part commonalities. The proposed CBSA algorithm uses the constraint-based genetic operators integrated with the simulated annealing (SA) approach that makes the algorithm more search exploratory (guarantee the optimal or near-optimal solution) and converge efficiently to the optimal solutions (less time-consuming). The proposed algorithm has higher likelihood of avoiding local optima as compared with standard SA and genetic algorithms. This is achieved by exploring a population of points, rather than a single point in the solution space. The proposed methodology is validated using a numerical case study for disassembly scheduling problem with part commonality.

Keywords Disassembly · Scheduling · Simulated annealing · Inventory management

PKS Prakash
The Irish Centre for Manufacturing Research,
National University of Ireland,
Maynooth,
Co. Kildare, Republic of Ireland

PKS Prakash · D. Ceglarek
The International Digital Laboratory, WMG,
University of Warwick,
Coventry CV4 7AL, UK

PKS Prakash (✉) · D. Ceglarek (✉)
Industrial and Systems Engineering, University of Wisconsin,
Madison, WI 53706, USA
e-mail: prakash@eeng.nuim.ie
e-mail: d.j.ceglarek@warwick.ac.uk

M. K. Tiwari
Department of Industrial Engineering & Management,
Indian Institute of Technology,
Kharagpur 721302, India
e-mail: mkt09@hotmail.com

1 Introduction

Current requirements for eco-efficient and sustainable products and production have caused product disassembly to receive considerable attention due to potential value recovery from the used products [1–3]. It helps inventory to meet the increasing global consumer demand for greener, more customized product through the necessary transition to a demand-driven industry with lower waste generation and energy consumption. The disassembly process is recognized as one of the basic activities in product and material recoveries such as re-use, remanufacturing, and recycling. Gungor and Gupta [4] state seven disassembly objectives which are as follows: (a) recovery of valuable parts or subassembly (in short supply), which are common to other products still being produced; (b) retrieval of parts or subassemblies of discontinued products to satisfy a sudden demand for these parts; (c) removal of hazardous parts; (d) increasing the purity of the remainder of the product for the purpose of chemical reclamation; (e) extraction of parts from the remainder of the product,

which can be sent as inventory for future use; (f) decreasing the amount of residue to be sent to landfills; and lastly, (g) achieving environment friendly manufacturing standards (i.e., meeting the required ratio of using recycled parts to using new parts).

The process of *material recovery* incorporates recycling and utilization of recovered material content of the used product. *Product recovery* targets to remanufacture the parts/components by performing the required disassembly, sorting, refurbishing, and other operations [5]. In both types of recovery processes, disassembly is an important step. Currently, operational problems in disassembly such as disassembly planning and scheduling are performed on ad hoc basis in most remanufacturing and recycling companies, which results in significant cost pressures [5, 6]. To date, very little work has been done in the area of disassembly scheduling [7, 8]. Below we briefly review the current state-of-the-art.

Various researchers have applied stochastic and deterministic algorithms to determine the optimal disassembly sequence and scheduling of complex products. Lambert et al. [9, 10] used the disassembly graph concept to determine the optimal disassembly sequence of discarded products. Xirouchakis and Kiritsis [11], Moore et al. [12–15], Zussman et al. [16], Zussman and Zhou [17], Rai et al. [18], Singh et al. [19], and Tiwari et al. [20] used Petri net-based disassembly model to address the disassembly planning problem. Prakash and Tiwari [21] applied the evolutionary-based psychoclonal algorithm to address the disassembly line balancing considering task failure. The scheduling aspects of disassembly for discrete parts without part commonality were first addressed by Gupta and Taleb [22]. Taleb et al. [23] further expanded the disassembly scheduling problem proposed by Gupta and Taleb [22] by including part commonality and applied reverse material requirements planning (MRP)-based methodology. Lee et al. [24] extended the disassembly scheduling problem without part commonality by taking into account the capacity constraints and proposed integer programming-based model to solve disassembly scheduling problem. Kim et al. [25, 26] also propose direct approaches such as Lagrangian heuristic algorithm and branch and bound algorithms to address the disassembly scheduling problem. Kim et al. [27] extended the proposed model from single product to multiple products and proposed a linear programming relaxation-based approach with the objective to minimize the setup, disassembly operation, and inventory holding costs.

In this paper, the problem of disassembly scheduling of product structure as discussed by Taleb et al. [23] is examined with capacity constraints as discussed by Kim et al. [27]. A heuristic approach "constrained-based simulated annealing" (CBSA) is developed and applied to solve the disassembly scheduling problem. The algorithm aims to determine the ordering and disassembly schedules of a product to generate a near-optimal schedule by primarily considering end-of-life requirements as criteria. The disassembly scheduling helps the manufacturing units to determine which product, when, and in what amounts have to be disassembled in order to meet the market demand of components/subassemblies and also keep inventory at a minimum. McGovern et al. [28] applied combinatorial optimization techniques to the disassembly line balancing problem (DLBP). McGovern and Gupta [29] further extended the work by comparing performance of various combinatorial optimization techniques for use with DLBP. A recent book by McGovern and Gupta [30] provides the general understanding of the disassembly problems.

In general, the disassembly problem can be regarded as a reverse MRP problem [22]. However, as the assembly process converges to a single demand source: final product (convergence property), the disassembly process diverges to its multiple demand source of product/component (divergence property). Thus, complexity grows drastically with the number of product to be disassembled, and the existing lot-sizing algorithm (MRP-based approaches) cannot be used directly to solve the disassembly scheduling problems. The disassembly problem is considerably more complicated when compared with the MRP problem, mainly due to multiple demand sources in the case of disassembly as compared to a single demand source in the assembly processes.

Due to the computational complexity involved in the disassembly problem, researchers are motivated to develop an algorithm that is free from the disadvantages associated with previously developed simple genetic algorithm (GA) and SA. The drawbacks associated with GA are premature convergence, extreme alliance of crossover, and too slow mutation rate. SA has a very high tendency of getting trapped into the local optima [31]. This paper proposes the CBSA approach to address the aforementioned two issues in solving the complex problem of disassembly scheduling. The main feature of the CBSA algorithm is its ability to converge to a near-optimal solution quickly, despite the challenges related to the disassembly scheduling problems such as high dimensionality, discontinuity, and multi-modality. The convergence of CBSA without degrading the solution quality is attained by integrating following features: (a) population-based selection, CBSA looks for a population of points in the whole search space rather than the single point and achieve toward the optimal solution; (b) in spite of the deterministic descent rule, a probabilistic transition rule is employed; the proposed algorithm replaces the Boltzmann function by the Cauchy function in the annealing process, which helps to avoid the local minimum more efficiently.

The rest of the paper is arranged as follows: Section 2 provides the mathematical model of the problem. The background and the proposed CBSA algorithm are discussed in Section 3. The CBSA-based heuristic solution methodology is presented in Section 4. Section 5 contains the result and discussion of the proposed solution, and Section 6 provides concluding remarks.

2 Mathematical model

Determining delivery schedule in order to have quick and timely deliveries of components/subassemblies requires a high level of synchronization between all business processes from decision on sourcing to deliveries. The objective of the paper is to decrease product inventory and economically optimize product structure disassembly considering parts and materials commonality. In disassembly structure, the part commonality refers to the product itself to be ordered, and leaf item is an item not to be disassembled further. The part commonality in disassembly scheduling problem refers to common items that can be obtained from more than one parent. For example, in Fig. 1, *part 5* is a common item and can be obtained from either *parent 2* or *parent 3*. The commonality in disassembly helps to lower inventory cost and disassembly cost due to quantity discount. It also helps in reducing the use of parts/materials across several end products. The part commonality makes the problem of scheduling more complex since it adds more degrees of freedom to the problem while creating dependencies between the components in the product structure. This section discusses the mathematical formulation used for disassembly scheduling problem.

2.1 Assumptions

As the problem of disassembly is quite complex, for the purposes of this paper, several assumptions are taken into account during the formulation of disassembly scheduling problem:

1. There is no shortage of the root item, i.e., products used for disassembly. The root item can be supplied whenever they are ordered.
2. The parts coming from external sources and not just from disassembly are also considered in this paper. The scheduled receipts from the external sources are assumed to be known.
3. Demands of leaf items are given and deterministic.
4. There is no deadlock occurrence; hence, the parts are produced on time.
5. Disassembly task failure rate is considered to be zero.
6. The disassembly lead time, i.e., the time required to disassemble the product, is known and deterministic.
7. Inventory holding costs are computed at the end-of-period.

2.2 Notations

cp_q purchase cost of root item in period q
cd_{rq} cost of disassembly of item r in period q

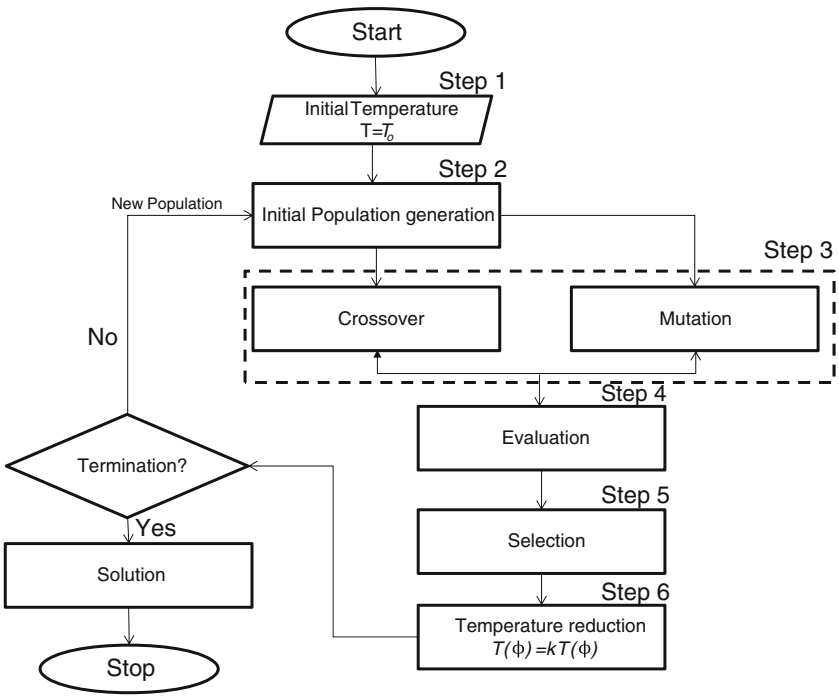

Fig. 1 Flowchart illustrating the process of constraint-based simulated annealing

cs_r setup cost of item r
cI_{rq} inventory holding cost of one unit of item r in period q
D_{rq} amount of item r disassembled in period q
E_{rq} item r purchased from external resources in period q
P_q purchase quantity of root item in period q
R_i total number of leaf items
H_{rq} level of inventory for item r in period q
J_{rq} demand of item r in period q
n_{xr} number of units of item r obtained after disassembly of item x
$\zeta(r)$ it represents the part commonality for item r

The problem is formulated by numbering all the items in a topological order from bottom to top The basic objective functions for disassembly scheduling problem as described by Lee et al. [24] is given in Eq. 1.

$$\text{Minimize}\left(\sum_{q=1}^{Q} cp_q P_q + \sum_{r=1}^{R_i}\sum_{q=1}^{Q} cs_r S_{rq} + \sum_{r=1}^{R}\sum_{q=1}^{Q} cI_{rq} H_{rq} + \sum_{r=1}^{R_i}\sum_{q=1}^{Q} cd_{rq} D_{rq}\right) \tag{1}$$

where

$$S_{rq} = \begin{cases} 1, & \text{if item } r \text{ require setup in interval } q \\ 0 & \text{else} \end{cases}. \tag{1a}$$

$$H_{rq} = H_{r,q-1} + E_{rq} + \sum_{x\in\zeta(r)} n_{xr} D_{x,q-R_i} - D_{rq} \text{ for} \tag{1b}$$

$$r = 2, 3, \ldots, R_i \text{ and } q = 1, 2, \ldots Q$$

$$H_{rq} = H_{r,q-1} + E_{rq} + \sum_{x\in\zeta(r)} n_{xr} D_{x,q-R_i} - J_{rq} \text{ for} \tag{1c}$$

$$r = R_i + 1, R_i + 2, \ldots R, \text{ and } q = 1, 2, \ldots Q$$

$$P_q \geq 0 \text{ for } q = 1, 2 \ldots \text{Q} \tag{1d}$$

$$H_{rq} \geq 0 \text{ for } r = 1, 2 \ldots R, \text{ and } q = 1, 2, \ldots Q \tag{1e}$$

$$D_{rq} \geq 0 \text{ for } r = 1, 2 \ldots R_i, \text{ and } q = 1, 2, \ldots Q \tag{1f}$$

Equation 1 represents the objective function which minimizes the production cost of the disassembly operation(s) for a given product structure by taking into consideration the purchasing cost, setup cost, inventory holding cost, and cost of disassembly. Equation 1b represents a decision variable which helps in deciding whether there is a setup change for item r in period q. Equation 1c represents the inventory effect at the end of the period q for item r when the parent items are disassembled. Inventory of the leaf item r at period q can be calculated by using Eq. 1d. Equations 1e–1f represent the constraint boundaries for selecting P_q, H_{rq}, and D_{rq}.

Most researchers have addressed the disassembly scheduling problem with the main aim to minimize the number of products to be disassembled [23]. Lee et al. [32] and Kim et al. [27] extended the disassembly scheduling problem by incorporating the capacity constraints for single and multi-product. This paper extends the previous work by taking into consideration the capacity constraints and parts coming from the external sources during disassembly scheduling. The objective function used for solving the problem of disassembly scheduling in this research is minimization of total inventory holding with the total number of root item disassembled as given by Eq. 2.

$$\text{Minimize}\left(\sum_{q=1}^{Q} cp_q P_q + \sum_{r=1}^{R}\sum_{q=1}^{Q} cI_{rq} H_{rq}\right) \tag{2}$$

The CBSA approach is used to determine the disassembly schedule taking into account common parts and common materials.

3 Constraint-based simulated annealing (CBSA)

Intricacies involved with the satisfaction of constraints in disassembly scheduling problems make the exact methods such as *mathematical programming* and *branch and bound* time-consuming, whereas heuristic approaches do not guarantee optimal solution. Thus, there is a trade-off between computational time for solution evaluation and quality of solution.

Several algorithms such as GA, SA, tabu search, and ant colony among others have been used to solve challenging optimization problems. The diversity of the aforementioned algorithms motivated authors to develop the CBSA algorithm, which endeavors to strike a balance between the problems of exact and heuristic methods. The CBSA algorithms uses *constraint-based genetic operators* integrated with the *SA approach* to make the algorithm more search exploratory and converge efficiently.

The CBSA algorithm satisfies all problem constraints defined by Eq. 1d–1f using the constraint-based genetic

operator. The objective function as shown in Eq. 2 is used to guide the solution toward the optimal/near-optimal solution. In this research, simulated annealing is used to enhance the probability of the algorithm to escape the local optima. In the upcoming sections (Sections 3.1 and 3.2), a brief review of SA and GA is presented.

3.1 Simple genetic algorithm

GA are stochastic search techniques that rely on the mechanics of natural selection and natural genetics [33]. In GA, the search starts with an initial set of random solutions known as *population*. Each solution of the population is referred as *chromosome*. Each chromosome of population is evaluated using some measure of fitness function. Based on the fitness function, a set of chromosomes is selected for breeding. In order to simulate a new generation, several biologically inspired operators such as selection, reproduction, crossover, and mutation are applied on the selected set of chromosomes. Based on the fitness value, parents and offsprings are selected, while rejecting some of the chromosomes to keep the population size constant. The execution of the algorithm continues until the optimal or near-optimal solution is found.

The objective of mutation operator in GA is to search the solution in the neighborhood of the solution string; if the mutation rate is too high, the objective of neighborhood search will not be fully achieved whereas the crossover rate are assigned a higher values to compliment the mutation in terms of searching solutions by jumping out of the local optima sub-space. Thus, care must be taken while designing the GA operators, their probability, and stopping criteria; otherwise, there is always a chance for this approach to get trapped into some local optima [34–41]. Moreover, due to the large search space, the computational time needed to converge to the final optimal or near-optimal solution can be large making the approach inefficient for handling multi-objective multi-constrained combinatorial problems.

3.2 Simulated annealing

The SA approach was proposed as a mechanism to overcome the difficulties arising from GA [42, 43]. SA is a random local search technique analogous to the physical annealing of solids. It starts with an arbitrary estimate of the optimal solution. New estimates are obtained by introducing random changes into the previous solution, and an energy function is used to determine the next generation solution. In SA, uphill moves (for a minimization problem) are also accepted, though with certain probability, for coming out of the local minima [44]. The value of the probability depends on two factors: (1) the difference between energy functions of the current state and the previous solution and (2) a parameter known as temperature. The search is initiated with a high temperature, and as the search progresses, the temperature declines according to an assumed cooling schedule. After several generations, the algorithm converges to the optimal solution of the problem. Although the technique proposes a useful approach to constrained optimization problems, it uses computationally expensive Monte Carlo simulation for constraint satisfaction [45] which appears to prohibit its application for large-sized problems.

The novel search technique, CBSA, proposed in this paper amalgamates the salient features of GA and SA, with some modifications, in order to have greater likelihood to escape from the local optima by uphill and downhill moves. Besides, the number of initial population could also be reduced. The next sub-section presents the CBSA algorithm in detail.

3.3 Constrained-based simulated algorithm

The proposed constraint-based simulated algorithm is a general purpose search technique that combines the elements of directed and stochastic search with a balance between exploration and exploitation of the search space. At the beginning, there is a randomly generated diversified population. Constraint-based crossover and mutation operators are then used to explore the widespread solution space. Later, SA is introduced as the selection process for the acceptance of best children as parents. The proposed CBSA algorithm employs a probabilistic transition rule rather than a deterministic descent rule. In CBSA, the Boltzmann function is replaced by the Cauchy function in the annealing process. The Cauchy function provides the algorithm with a greater opportunity to move away from the local minima [31]. Fitness function is used throughout the evolution operation to decide which solution will be fitter to survive in the new population. The process is repeated for a finite number of generations or till the temperature freezes to allow convergence to take place at the optimal or near-optimal solution to the problem. The basic structure of the algorithm is as follows:

1. Start with initial temperature T_o.
2. Randomly generate P organisms as parents.
3. Children were produced from each parent.
4. The best one in every family is selected by competition among children.
5. The parent for next generation is obtained of each family. Best child is adopted as parent for next generation, if $\Delta F = F_2 - F_1$ (difference of energy function between different states, where energy refers to the objective function pertaining to the various types of problem) If $\Delta F > 0$ or $\frac{T(\phi)}{T(\phi)^2+\Delta F^2} > \psi$

where

F_1 energy function of children
F_2 energy function of its parent
$T(\varphi)$ is temperature at φth generation
Ψ is a random number between 0 and 1

6. Reduce the temperature as follows: $T(\phi) = k\,T(\phi)$ (where k is a temperature reduction factor).
7. Repeat steps 3 to 6 for a finite number of generations or till the temperature freezes.

The structure of the algorithm for the above described steps are shown in Fig. 1.

4 Solution methodology

The proposed CBSA was tested on a well-known problem of disassembly scheduling of product structure taken from Taleb et al. [23]. This section discusses the different design issues of the CBSA in a disassembly scheduling problem.

4.1 Problem description

The disassembly structure of a product incorporating part and material commonality is shown in Fig. 2 [23]. Figure 2 represents the product structure, where the number in the parenthesis represents the yield of that item, i.e., units of item obtained when its parent is disassembled. *Parts 5*, *11*, and *12* in Fig. 2 represent the part commonality of *product 1*. Table 1 presents the data related to the disassembly lead time (DLT), i.e., the time needed to disassemble a certain item and ordering lead time (OLT), which is the time it takes for an order to arrive. Table 2 shows the demand of

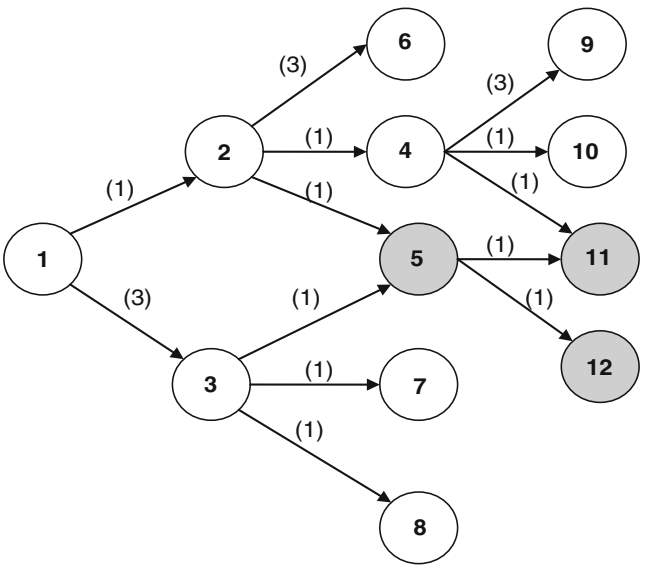

Fig. 2 Disassembly of single product type with parts commonality

Table 1 DLT and OLT of different parts given Fig. 2

	1	2	3	4	5	6	7	8	9	10	11	12
OLT (period)	1	0	0	0	0	0	0	0	0	0	0	0
DLT (period)	1	1	2	1	2	0	0	0	0	0	0	0

leaf in different time periods. Additionally, Table 3 contains the data on procurement of parts from external sources. The parts coming from external sources are considered deterministic in nature. Table 4 provides the initial inventory present in the disassembly firm.

4.2 Encoding schema

Keeping in mind the influence of an encoding scheme on all the subsequent steps of the algorithm, it is crucial to have a good encoding scheme that can clearly describe the problem-specific characteristic. The disassembly scheduling problem considered in this paper can be divided into two sub-problems: minimizing the number of root item to be disassembled during disassembly and minimizing inventory. As the disassembly scheduling problem is complex in nature due to large search space, the concept of masking is initiated in order to decrease the search space and to speed up the convergence rate of CBSA. In masking, infeasible positions are determined in the gene using Eq. 3. Genes masked positions are considered as infeasible genes.

Q total number of periods
l_r longest disassembly time for parent item r
S_r shortest disassembly time for parent item r
If_r period at which first demand is made for parent item r
M_{rq} genes masking status for part r at interval q

$$M_{rq} = (If_r - q) > l_r \text{ OR} (Q - q) < S_r \text{ for} \quad (3)$$

$$q = 1, 2 \ldots Q \text{ and } r = 1, 2 \ldots R_i$$

where M_{rq} is Boolean operator where its value is 0 when both conditions are not satisfied and 1 if either of the

Table 2 Demand for leaf items

Leaf item	Time period									
	1	2	3	4	5	6	7	8	9	10
6	0	0	0	0	0	75	50	150	100	25
7	0	0	0	0	0	0	15	20	500	50
8	0	0	0	0	0	0	15	0	100	0
9	0	0	0	0	0	25	50	300	0	600
10	0	0	0	0	0	130	20	450	0	10
11	0	0	0	0	0	0	20	20	50	400
12	0	0	0	0	0	0	40	20	600	400

Table 3 Parts procured from external sources

Part	Time period									
	1	2	3	4	5	6	7	8	9	10
1	0	0	0	2	0	1	0	0	2	0
2	0	0	2	1	0	0	0	0	0	0
3	0	0	1	0	0	0	0	0	2	0
4	0	25	0	0	11	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	2	0	0	0	0
7	0	0	0	4	0	0	0	8	0	0
8	0	0	0	13	0	0	0	0	25	0
9	0	0	0	0	0	25	0	0	0	0
10	0	1	0	1	0	0	3	0	0	0
11	0	0	0	0	10	0	0	0	0	0
12	4	0	0	0	0	8	0	0	0	0

condition is satisfied The truth table used to decide the value of M_{rq} is given in Table 5.

If the obtained M_{rq} is equal to 1, then the gene is masked and cannot be scheduled for disassembly. For the present problem, the total number of period Q is taken as 10. l_r and S_r are calculated using the precedence relation product structure, for example, l_r and S_r for part 2 is 3 and 1 U, respectively; the path used in calculating the l_r and S_r for part 2 is shown in Fig. 3.

An example of chromosome generated is given in Table 6. For clarity, the masked or infeasible positions are represented by "*" in the given chromosome. Table 7 illustrates the inventory status for the chromosome.

4.3 Initialization and evaluation

The proposed CBSA operates on a population of randomly generated individual strings called chromosomes. Each element of a chromosome is called a gene. The number of chromosomes in a population is called population size denoted by pop_size. In literature, researchers have taken either heuristic procedures or random techniques to generate feasible strings that form the initial population [31]. Here, the initial population is generated randomly. It is believed that a more diverse population initiates a more effective search. It is also a known fact that the computational efforts are reduced by improving the diversity in the solution as it helps in faster

Table 4 Initial inventory

Part	1	2	3	4	5	6	7	8	9	10	11	12
Initial inventory	2	1	5	0	5	8	10	0	12	4	10	12

Table 5 Truth table of OR gate with $(If_r - q) > l_r$ and $(Q - q) < S_r$ as input, M_{rq} as output

Input		Output
$(\mathrm{If}_r - q) > l_r$	$(Q - q) < S_r$	M_{rq}
0	0	0
0	1	1
1	0	1
1	1	1

convergence. Therefore, the population is generated randomly to improve the diversity within the generated initial population. Initial temperature (T_o) is initialized at the start of the algorithm. After initialization, the evaluation of each chromosome is carried out using the fitness function given in Eq. 2. The fitness function plays an important role in deciding the population for each generation. The Cp_q and CI_{rq} cost associated with the disassembly problem is 15 and 1 per unit, respectively. The fitness function calculated for chromosome as shown in Tables 6 and 7 is evaluated as 36,795.

4.4 Selection

The best offspring produced in each family is included in the next generation's population according to some selection criteria. The selection criteria used in CBSA algorithm is inspired by the transition probability function used in the SA approach. The transition probability function accepts both uphill and downhill moves; thus, the function is capable of avoiding entrapment at the local maxima. These criteria are given in Sections 4.4.1 and 4.4.2.

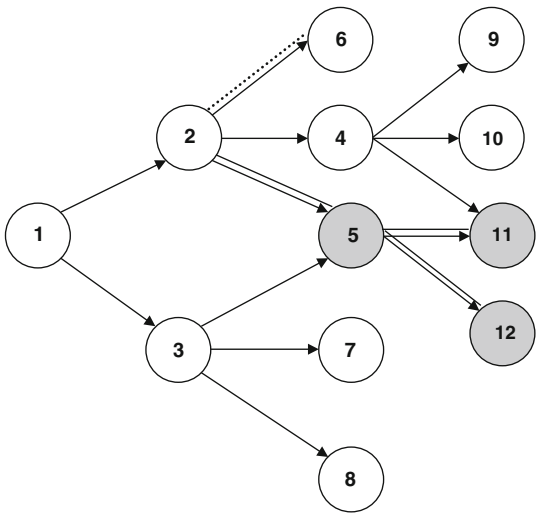

Fig. 3 Shortest and longest paths for *parent 2* to reach leaf item

Table 6 Initial solution

Items	Period									
	1	2	3	4	5	6	7	8	9	10
1	*	60	35	50	500	0	0	0	*	*
2	*	*	60	30	59	450	50	0	0	*
3	*	0	40	10	50	400	60	50	*	*
4	*	*	*	*	124	20	450	0	10	*
5	*	*	*	20	20	156	450	500	*	*

4.4.1 Fitness function criterion

If an offspring chromosome is fitter than the parent chromosome, it will move to the next generation. This can be calculated as:

$$\Delta F = F_2 - F_1 \tag{4}$$

where

F_2 fitness function of the best offspring in each family
F_1 fitness function of the parent of that family

If ΔF comes greater than 0, then this offspring of the family is accepted as a parent for the new generation. In case of duplicate offspring generated during the algorithm, the ΔF=0; thus, they are not selected as a parent for the next generation as this prevents the pre-convergence of the algorithm.

4.4.2 Probabilistic criterion

Even if the best offspring of any family is inferior to its parent, i.e., it may have a lower fitness value, there is still some probability for its acceptance in the new generation, so as to escape the chances of convergence at any local optimum. The probabilistic function used here for this purpose is the Cauchy distribution defined as:

$$C(T(\phi), \Delta F) = \frac{T(\varphi)}{T(\varphi)^2 + \Delta F^2} > \psi \tag{5}$$

where $T(\phi)$ is temperature during the ϕth generation decided by cooling schedule where ϕ represents the iteration of the algorithm. If an inferior offspring moves to the next generation, $C(T(\phi), \Delta F)$ should be greater than ψ, where ψ is any randomly generated number between 0 and 1.

In this research, the conventionally used Boltzmann distribution function is replaced by the spherically symmetric Cauchy distribution function due to its ability to escape the local optima much more effectively. The justification to use spherically symmetric Cauchy function as a transition function instead of Boltzmann distribution, during the SA stage for the selection of best children as a parent, is provided in Tiwari et al. [31].

4.5 Cooling schedule

Cooling schedule determines the value of the transition probability function used in the selection criterion and thus carries significance. In the proposed algorithm, the cooling schedule employed is defined by Eq. 6.

$$T(\varphi) = k\ T\ (\varphi) \tag{6}$$

where k is the reduction factor. The value of k in this research has been set to 0.9 after carrying out several simulation runs.

4.6 Constraint-based crossover

Crossover is a process by which two parent strings recombine to produce two new offspring strings. The idea behind

Table 7 Inventory status for initial solution

Items	Periods									
	1	2	3	4	5	6	7	8	9	10
1	2	0	0	0	0	1	1	1	3	3
2	1	1	3	9	0	50	0	0	0	0
3	5	5	146	241	341	1,441	1,381	1,331	1,333	1,333
4	0	25	25	85	2	41	41	91	81	81
5	5	5	5	45	95	8	58	8	68	118
6	8	8	8	188	278	382	1,682	1,690	1,590	1,565
7	10	10	10	14	54	64	99	479	39	39
8	0	0	0	13	53	63	98	498	483	533
9	12	12	12	12	12	384	394	1,444	1,444	874
10	4	5	5	6	6	0	3	3	3	3
11	10	11	11	11	21	165	185	771	1,171	1,281
12	16	16	16	16	16	44	24	160	10	110

crossover is that the new solution may be better than both parents if it takes the best characteristics from each. In the literature, there exist several crossover operators that have been used for sequencing and scheduling problems such as partially mapped crossover [34], enhanced edge recombination, cycle crossover [46], and order crossover [47]. In this research, constraint-based crossover operator proposed by Cormier et al. [48] has been used to solve the disassembly scheduling problem. In constraint-based crossover, depending on the problem constraints, the attribute values are assigned from the parents into the child. Two parent solutions are randomly selected from the survivor population, and then attribute values from the parents are assigned to the offspring. The pseudocode of the constraint-based crossover is given as follows:

Nomenclature:

$A_{c,r}(t, q)$	value of item r in for chromosome c at generation t in period q
$D_{c,r}(t, q)$	domain for characteristic r of chromosome c at generation t in period q
Wq	set of attributes in period q where Wq= {1, 2, …R}
Pc	population generated from crossover
C_p	set of leaf item having more than one part parents in the product architecture
Update (Pc)	function that updates the characteristics r of the chromosome c at generation t and period q

Pseudocode of the constraint-based crossover algorithm:

For x = 1 to n; $x \le n$ // for every new child in the pool
{
P=[P_1, P_2 …, Pn] // P is set of available Parent
}
Select two parents $\{P_1\, P_2\}$ where $P_1 \neq P_2$ and $\{P_1\, P_2\} \in \mathbf{P}$
Wq = {1, 2, …, R} // a set of attributes
Pc=*random*(P_1, P_2)
For q=1 to Q; $q \le Q$
{
If $A_{P1,r}(t, q) \ge D_{c,r}(t, q)$ && $A_{P1,r}(t, q) < D_{c,r}(t, q)$
{
$A_{c,r}(t, q) = A_{P1,r}(t, q)$
Update (P_c)
Elseif $A_{P1,r}(t, q) < D_{c,r}(t, q)$ && $A_{P1,r}(t, q) \ge D_{c,r}(t, q)$
$A_{c,r}(t, q) = A_{P2,r}(t, q)$
Update (P_c)
Elseif $A_{P1,r}(t, q) \ge D_{c,r}(t, q)$ && $A_{P1,r}(t, q) \ge D_{c,r}(t, q)$
$A_{c,r}(t, q) = min\{A_{P1,r}(t, q), A_{P2,r}(t, q)\}$
Update (P_c)
Else
If $r \notin C_p$
{
Update (P_c) // update the population based on disassembly product structure
s. t., $A_{c,r}(t, q) \ge D_{c,r}(t, q)$
Elseif $r \in C_p$
Update (P_c) // update parent items producing r in product structure, randomly
s. t., $A_{c,r}(t, q) \ge D_{c,r}(t, q)$
}
}
}

In the present disassembly scheduling product structure problem, the crossover takes place between two scheduling matrix of order $R_i \times Q$ by interchanging the rows where R_i represent the parent items and Q is the total number of periods considered for scheduling. The rows interchange during crossover may lead to a violation of constraint defined by Eq. 1d–1f. Thus, constraint-based crossover is needed to satisfy all the required constraints.

4.7 Constraint-based mutation

Constraint-based mutation is a unary genetic operator that operates on a single chromosome at a time and generates offspring by altering one or more values in a solution, based on the user-definable mutation probability p_m. The mutation probability, defined as the percentage of the total number of genes undergoing alteration, controls the rate at which new genes are introduced into the population for trial. Mutation modifies the attribute values to introduce variety into the population of design alternatives. The main aim of the constraint-based mutation operator is to prevent mutation from rendering a previously feasible alternative which may be infeasible in the constrained environment. The pseudocode of the constraint-based mutation is given as follows:

Pseudocode of the constraint-based mutation algorithm:

```
   P=[P1, P2 …, Pn] // list of populations
For K = 1 to M // for each mutation to be performed
{
  Select population P, i.e., P ∈ P
  PM =mutation (P); // perform mutation
  Wq = {1, 2, …, R} // a set of attributes
For q=1 to Q; q ≤ Q
 For r=1 to R; r ≤ R
 {
  If AM,r(t, q) < Dc,r(t, q)
  {
    If r ∉ Cp
    {
    Update (PM) // update the population based on disassembly product structure
    s. t., AM,r(t, q) ≥ Dc,r(t, q)
    Elseif r ∈ Cp
    Update (PM) // update parent items producing r in product structure, randomly
    s. t., AM,r(t, q) ≥ Dc,r(t, q)
    }
   }
  }
}
```

Constraint-based mutation is carried out by interchanging the gene in the same row by taking into account the demand of leaf items. In the present problem, mutation helps to reschedule the items that lead to the minimization of the inventory level.

4.8 Stopping criteria

Stopping criteria for proposed constraint-based simulated annealing algorithm are as follows:

1. As the temperature falls to a certain prescribed value, the iterations are stopped.
2. If GEN=MAX_GEN, then the search procedure terminates and the maximum value of the objective functions is obtained.
3. For collecting the number of rejection of perturbed solution, a reject counter is set as REJECT_MAX. In the case of the rejection of a solution, the reject counter increases by 1. As soon as the counter reaches a predetermined fixed value, the search procedure termi-

Table 8 Optimal chromosome

Items	Periods									
	1	2	3	4	5	6	7	8	9	10
1	*	0	84	17	460	0	0	0	*	*
2	*	*	0	88	17	460	0	0	0	*
3	*	0	0	0	66	400	60	50	*	*
4	*	*	*	*	124	17	450	0	10	*
5	*	*	*	0	16	20	600	400	*	*

nates which signifies that a near-optimal solution has been achieved. REJECT_MAX in this paper is set to 3 which signifies the algorithm will be terminated if the solution is not improved in the last three steps. This criterion stops the algorithm if the average change in the fitness function value is stall over number of generations.

5 Results and discussion

The results obtained for the disassembly scheduling problem illustrated in Section 4.1 are shown in Tables 8, 9, and 10. Table 8 shows in which period and how much of the parent item should be send for disassembly so that demand for the leaf item can be met. Table 9 shows the effect of disassembly on the inventory of the manufacturing unit. Based on the OLT for the root item 1, the purchasing schedule can be prepared to mean the demand of all the leaf items. Table 10 shows the purchasing schedule of the root item.

The solution obtained from the proposed CBSA methodology is compared with Taleb et al. [23]. The total number of root part disassembled is reduced from 560 to 557, and the total inventory holding is reduced from 23,760 to 23,016 as obtained by Taleb et al. [23].

Table 10 Purchasing schedule for root items

Root item	Periods									
	1	2	3	4	5	6	7	8	9	10
1	0	82	17	458	0	0	0	0	0	0

Further, simulation is conducted for five test problems on the proposed problem by varying initial inventory, items received from external sources, and demand of leaf item. The initial inventory, item received from the external sources, and demand of leaf item are generated if based on the uniform distribution U(a, b) where {a, b} represents the lower and upper bound of the distribution, respectively. With initial inventory varying from $U(0,15)$, parts received from external sources vary from $U(0,30)$, and demand for the leaf item varies from $U(0, 700)$ for periods 3 to 10. The performance of the CBSA algorithm has been compared with the integer programming techniques in terms of quality of solution and computation time used to obtained the solution. The result obtained is shown in Table 11. The CBSA-based algorithm was coded in C++ language, and the program was run on IBM PC with Pentium CPU at 2.0 GHz.

6 Conclusion

This paper presents a novel algorithm, namely CBSA to tackle constrained optimization problems. The proposed CBSA algorithm uses the *constraint-based genetic oper-*

Table 9 Inventory status for optimal chromosome

Items	Periods									
	1	2	3	4	5	6	7	8	9	10
1	2	2	0	0	0	1	1	1	3	3
2	1	1	3	0	0	0	0	0	0	0
3	5	5	6	258	243	1,223	1,163	1,113	1,115	1,115
4	0	25	25	25	0	0	10	10	0	0
5	5	5	5	5	77	74	0	0	60	110
6	8	8	8	8	272	250	1,580	1,438	1,338	1,313
7	10	10	10	14	14	14	65	445	5	5
8	0	0	0	13	13	13	64	464	449	499
9	12	12	12	12	12	384	385	1,435	1,435	865
10	4	5	5	6	6	0	0	0	0	0
11	10	11	11	11	21	145	158	608	1,158	1,168
12	16	16	16	16	16	24	0	0	0	0

Table 11 Results schedule for root items

Problem	Heuristic gap (%)	CPU time (s)	
		CBSA	Integer programming
1	0.56	0.028	12.64
2	1.32	0.041	8.56
3	0.36	0.052	4.3
4	2.25	0.15	4.35
5	1.35	0.025	7.66
Average	1.168	0.0592	7.502

ators integrated with the SA approach to make the algorithm more search exploratory. The algorithm's robustness has been enhanced by incorporating the Cauchy function, which provides a greater likelihood to escape from the local optima.

The verification of the proposed methodology is presented through a case study of a disassembly scheduling problem with part commonality. The proposed methodology is robust in identifying the optimal/sub-optimal solution for disassembly scheduling problem involving constraints.

This research can be expanded to various problems that encompass the sequencing or allocation of resources, such as material handling, pallets, and fixture in an flexible manufacturing system environment. Also, this research can be exploited to solve the assembly sequencing problem in the multi-stage optimization problem and the multi-objective loading and scheduling problems by introducing more flexible attributes.

References

1. Takata S, Kimura F, van Houten FJAM, Wdestkamper E, Shpitaini M, Ceglarek D, Lee J (2004) Maintenance: changing role in life cycle management. CIRP Annals-Manufacturing Technology 53(2):643–655
2. Maropoulos PG, Ceglarek D (2010) Design verification and validation in product lifecycle. CIRP Annals-Manufacturing Technology 59(2):740–759
3. Tolio T, Ceglarek D, ElMaraghy HA, Fischer A, Hu SJ, Laperriere L, Newman, ST, Vancza J (2010) Species-Co-evolution of products, processes and production systems. CIRP Annals-Manufacturing Technology 59(2):672–693
4. Gungor A, Gupta SM (2001) A solution approach to the disassembly line balancing problem in the presence of task failure. Int J Prod Res 39(7):1427–1467
5. Gungor A, Gupta SM (1999) Issues in environmentally conscious manufacturing and product recovery: a survey. Comput Ind Eng 36:811–853
6. Alting L, Legarth JB (1995) Life cycle engineering and design. Ann CIRP 44(2):569–580
7. Lee DH, Kang JG, Xirouchakis P (2001) Disassembly planning and scheduling: review and further research. Proc Inst Mech Eng: J Eng Manuf—Part B 215:695–710
8. Kim H-J, Lee D-H, Xirouchakis P (2007) Disassembly scheduling: literature review and future research directions. Int J Prod Res 45(18):4465–4484
9. Lambert AJD, De Ron AJ, Splinter MAM (1996) Optimal disassembly. In: Proceeding of 3rd international seminar on life cycle engineering, Zurich, pp 203–210
10. Lambert AJD (1997) Optimal disassembly of complex products. Int J Prod Res 35(9):2509–2523
11. Xirouchakis P, Kiritsis D (1997) Petri net modeling of disassembly process planning. American Society of Mechanical Engineers: Design Engineering Division, New York, pp 255–262
12. Moore KE, Gungor A, Gupta SM (1998) A Petri net approach to disassembly process planning. Comp Ind Eng 35(1–2):165–168
13. Moore KE, Gungor A, Gupta SM (1998b) Disassembly Petri net generation in the presence of XOR precedence relationships. In: Proceeding of the 1998 IEEE international conference on systems, man and cybernatics, La Jolla, California, 11–14 October, pp 13–18
14. Moore KE, Gungor A, Gupta SM (1998c) Disassembly process planning using Petri nets. In: Proceeding of the 1998 IEEE symposium on electronics and environment, Oak Brook Illinois, 4–6 May, pp 88–93
15. Moore KE, Gungor A, Gupta SM (2001) Petri net approach to disassembly process planning for products with complex AND/OR precedence relationships. Eur J Oper Res 135(2):428–449
16. Zussman E, Zhou MC, Caudill R (1998) Disassembly Petri net approach to modeling and planning disassembly processes of electronics products. In: Proceeding of the 1998 IEEE international symposium on electronics and environment, Oak Brooks, Illinois, 4–6 May, pp 331–336
17. Zussman E, Zhou MC (1999) Methodology for modeling and adaptive planning of disassembly process. IEEE Trans Robot Autom 15(1):190–194
18. Rai R, Rai V, Tiwari MK, Allada V (2002) Disassembly sequence generation: a Petri net based heuristic approach. Int J Prod Res 40 (13):3183–3198
19. Singh AK, Tiwari MK, Mukhopadhyay SK (2003) Modeling and planning of the disassembly processes using an enhanced expert Petri net. Int J Prod Res 41(16):3761–3792
20. Tiwari MK, Sinha N, Kumar S, Rai R, Mukhopadhyay SK (2001) A Petri net based approach to determine the disassembly strategy of a product. Int J Prod Res 40(5):1113–1129
21. Prakash PKS, Tiwari MK (2005) Solving a disassembly line balancing problem with task failure using a psychoclonal algorithm. In: Proceedings of ASME 2005 IDETC/CIE, Long Beach, California, USA, September 24–28, pp 1–7
22. Gupta SM, Taleb KN (1994) Scheduling disassembly. Int J Prod Res 32(8):1857–1866
23. Taleb KN, Gupta SM, Brennan L (1997) Disassembly of complex product structures with parts and materials commonality. Prod Plan Cont 8:255–269
24. Lee DH, Kim HJ, Choi G, Xirouchakis P (2004) Disassembly scheduling: integer programming models. Inst Mech Eng Part-B: J Eng Manuf 38:1357–1372
25. Kim HJ, Lee DH, Xirouchakis P (2006) A Lagrangian heuristic algorithm for disassembly scheduling with capacity constraints. J Oper Res Soc 57(10):1231–1240
26. Kim HJ, Lee DH, Xirouchakis P (2006) A branch and bound algorithm for disassembly scheduling with assembly product structure. J Oper Res Soc 60(3):419–430
27. Kim HJ, Lee DH, Xirouchakis P, Zust R (2003) Disassembly scheduling with multiple product types. CIRP Ann Manuf Technol 52(1):403–406
28. McGovern SM, Gupta SM, Kamarthi SV (2003) Solving disassembly sequence planning problems using combinatorial

optimization. In: Proceedings of Northeast Decision Sciences Institute 32nd annual meeting, pp 178–18
29. McGovern SM, Gupta SM (2004) Combinatorial optimization methods for disassembly line balancing. In: Proceedings of the 2004 SPIE international conference on environmentally conscious manufacturing IV, Philadelphia, Pennsylvania, October 25–28, pp 53–66
30. McGovern SM, Gupta SM (2001) The disassembly line: balancing and modeling. McGraw-Hill, New York
31. Tiwari MK, Kumar S, Kumar S, Prakash PKS, Shankar R (2006) Solving part-type selection and operation allocation problems in an FMS: an approach using constraint-based fast simulated annealing algorithm. IEEE Trans Sys Man and Cybernatics-Part A 36(6):1170–1184
32. Lee DH, Xirouchakis P, Zust R (2002) Disassembly scheduling with capacity constraints. CIRP Ann Manuf Technol 51(1):387–390
33. Goldberg DE (1989) Genetic algorithm in search, optimization, and machine learning. Addison Wesley, Reading
34. Goldberg DE, Lingle R (1985) Alleles, loci and the traveling salesman problem. In: Proceeding of the first international conference on genetic algorithm and their applications, 30(4). Lawrence Erlbaum Associates, Hillsdale, pp 931–940
35. Tiwari MK, Vidyarthi NK (2000) Solving machine loading problems in a flexible manufacturing system using a genetic algorithm based heuristic approach. Int J Prod Res 38(14):3357–3384
36. Aytug H, Khouja M, Vergara FE (2003) Use of genetic algorithms to solve production and operations management problems: a review. Int J Prod Res 41(17):3955–4009
37. Choubey AM, Prakash PKS, Tiwari MK (2005) Solving a fixture configuration design problem using genetic algorithm with learning automata approach. Int J Prod Res 43(22):4721–4743
38. Phoomboplab T, Ceglarek D (2008) Process yield improvement through optimum design of fixture layouts in 3D multistation assembly systems. Journal of Manufacturing Science and Engineering-Transactions of the ASME 130(6)
39. Huang W, Phoomboplab T, Ceglarek D (2009) Process capability surrogate model-based tolerance synthesis for multi-station manufacturing systems. IIE Transactions 41(4):309–322
40. Chen Yong, Ding Yu, Jin JH, Ceglarek D (2006) Integration of process-oriented tolerancing and maintenance planning in design of multistation manufacturing processes. IIE Transactions on Automation Science and Engineering 3(4):440–453
41. Camelio JA, Hu SJ, Ceglarek D (2004) Impact of fixture design on sheet metal assembly variation. Journal of Manufacturing Systems 23(3):182–193
42. Kirkpatrick F, Gelatte CD, Vecchi MP (1983) Optimization by simulated annealing. Science 220:671–780
43. Cerny V (1985) Thermodynamical approach to the traveling salesman problem: an efficient simulation algorithm. J Optim Theory Appl 45:41–51
44. Mukhopadhyay SK, Singh MK, Srivastava R (1998) FMS machine loading: a simulated annealing approach. Int J Prod Res 36:1526–1547
45. Creutz M (1983) Microcanonical Monte Carlo simulation. Phys Rev Lett 50(19):1411–1414
46. Oliver I, Smith D, Holland J (1987) A study of permutation crossover operators on the travelling salesman problem. In: Proc. int. conf. on genetic algorithm and their applications
47. Davis L (1985) Applying adaptive algorithm to epistatic domains. In: Proceeding of the international joint conference on artificial intelligence, pp 162–264
48. Cormier D, Grady PO, Sanii E (1998) A constraint based genetic algorithm for concurrent engineering. Int J Prod Res 36(6):1679–1697