J Intell Manuf (2009) 20:389–400
DOI 10.1007/s10845-008-0113-5

Bi-objective optimization algorithms for joint production and maintenance scheduling: application to the parallel machine problem

A. Berrichi · L. Amodeo · F. Yalaoui · E. Châtelet · M. Mezghiche

Received: 14 July 2007 / Accepted: 8 May 2008 / Published online: 7 June 2008

Abstract This paper deals with the joint production and maintenance scheduling problem according to a new bi-objective approach. This method allows the decision maker to find compromise solutions between the production objectives and maintenance ones. Reliability models are used to take into account the maintenance aspect of the problem. The aim is to simultaneously optimize two criteria: the minimization of the makespan for the production part and the minimization of the system unavailability for the maintenance side. Two decisions are taken at the same time: finding the best assignment of n jobs to m machines in order to minimize the makespan and deciding when to carry out the preventive maintenance actions in order to minimize the system unavailability. The maintenance actions numbers as well as the maintenance intervals are not fixed in advance. Two evolutionary genetic algorithms are compared to find an approximation of the Pareto-optimal front in the parallel machine case. On a large number of test instances (more than 5000), the obtained results are promising.

Keywords Production scheduling · Preventive maintenance (PM) · Reliability · Multi-objective optimization · Genetic algorithms

A. Berrichi (✉) · M. Mezghiche
Université M'hamed Bouguerra de Boumerdès, LIFAB, Avenue de l'indépendance, 35000 Boumerdès, Algèrie
e-mail: aberrichi@umbb.dz

L. Amodeo · F. Yalaoui · E. Châtelet
Université de Technologie de Troyes, ICD (FRE CNRS 2848), Rue Marie Curie, BP 2060, 10010 Troyes, France

Introduction

In the manufacturing industry, the machines are considered as main resources among others to carry out the production plan. The production and maintenance are the two services which directly act on the machines. Production scheduling is concerned with allocating limited resources to a set of jobs and certain objective functions have to be optimized, for example to meet at best deadlines fixed with customers by minimizing the sum of tardiness or makespan. Numerous studies have been devoted to solve this problem, according to the configuration of the workshop (single machine, parallel machines, flow shop, job shop, open shop and hybrid systems), the objectives to optimize and the constraints to take into account (preemption, setups, etc). Most production scheduling problems are NP-hard (Garey and Johnson 1979). In research literature related to production scheduling, it is assumed that the machines are always available. However, in real manufacturing systems, machines may be subject to some unavailability periods such as maintenance activities. For the maintenance consideration, the most important tasks of the service is to establish an appropriate preventive maintenance planning optimizing a certain objective function, like maintenance costs or keeping the machines in a good working order at every moment. Several studies in this area have been also conducted to solve this problem in the past decades. However, most of these studies do not take into account the production requirements.

Despite the interdependent relationship between the production scheduling and the maintenance planning, the two activities are generally planned and executed separately in real manufacturing systems. For many years the relationship between production and maintenance has been considered as a conflict in management decision. This situation stills the same because of the lack of communication regarding the scheduling requirements of each function (Weinstein and Chung 1999). The conflicts may result in an unsatisfied demand in production due to the interruptions coming from the preventive maintenance (PM) interventions or machine breakdowns if the production service does not respect the expected periods of PM. To avoid these conflicts, we propose in this paper an integrated multi-objective model taking into account the machines reliability for the PM aspect. This model allows the decision maker to have compromise solutions meeting at best two criteria, one related to production and another to PM. The paper is organized as follows: section "Literature review" gives a survey of scheduling problems taking into account preventive maintenance. Section "Modeling of the integrated problem" describes the integrated model of the joint production and maintenance scheduling proposed and the method to evaluate system unavailability. The solution methods proposed are presented in section "Solving method". Section "Test and results" experiments with the two meta-heuristics presented in section "Solving method". Finally, section "conclusions" concludes the paper and gives future research.

Literature review

The machine scheduling literature taking into account the maintenance can be classified into two categories: the deterministic (or sequential) approach and the stochastic (or integrated) approach. In scheduling literature, a job processing is called non preemptive if a job must be reprocessed fully after the maintenance if its processing is interrupted by the maintenance activity on a machine.

In deterministic (or sequential) approach, the time intervals of PM actions as well as their number are known and fixed in advance. The majority of the research literature with maintenance adopts this approach commonly called "scheduling with machine availability constraints". All the configurations of known workshops were approached by the researchers: single machine, parallel machines, flow shop, job shop, open shop and hybrid systems. Considering the significant number of works carried out in this category, we only reviewed the parallel machine case. For the makespan criterion, (Lee 1991) has proved that the problem of minimizing the makespan with availability constraints in the non preemptive case is NP-hard. He studied the problem when some machines may not be available at time zero. (Schmidt 1984) has examined the problem on m parallel machines where each machine can be only used during several availability periods. (Liao et al. 2005) considered a two parallel machine problem where one machine is not available during a time period. They proposed several algorithms for the preemptive and the non preemptive cases. In (Liao et al. 2007), instead of fixing the maintenance periods, the maintenance actions are carried out after processing a fixed number of production jobs. Two different scheduling horizons have been investigated for this problem.

Another criterion commonly studied by researchers is the total completion time. In the case of machines which are not available at time zero, (Kaspi et al. 1988) and (Liman 1991) showed that scheduling jobs according to SPT (Shortest Processing Time) rule is optimal for minimizing total completion time. (Mosheiv 1994) has treated the problem supposing each machine is available within a specific interval. (Lee and Chen 2000) studied the problem of processing a set of n jobs on m parallel machines where each machine must be maintained once during the planning horizon. The same problem is considered by (Mellouli et al. 2006). They have proposed three exact methods to minimize the total completion time. (Schmidt 2000) and (Lee 1996) investigated and analyzed the scheduling problem with limited machine availability in greater detail for different constraints and machine environments.

We note that most studies in deterministic approach consider the maintenance periods as constraints to schedule production. In other words, the production scheduling is optimized according to an established maintenance planning. That is, in this approach the maintenance is always privileged compared to production. In certain cases, it is the production which has priority as in (Liao et al. 2007).

In stochastic (or integrated) approach, the beginning times of PM tasks are considered as decision variables as well as production jobs. Production jobs and maintenance tasks are jointly (but not sequentially) scheduled. There are few papers in the literature related to this approach. (Kaabi et al. 2002) and (Kaabi et al. 2003) have respectively studied the single machine and permutation flow shop case where the maintenance periods must be carried out within a predefined interval. (Xu et al. 2008) have used the same idea as (Kaabi et al. 2002) for the parallel machine case to minimize the makesapn. These studies have the advantage of decreasing the conflicts between the two services, but they still favor the maintenance to the detriment of the production. (Cassady and Kutanoglu 2003) have formulated an integrated mathematical model for single machine problem to minimize the total weighted tardiness of production. However, a total enumeration approach is used. Indeed, the authors observed that the computational time becomes unbearable when the number of jobs exceeds eight which is not practical. (Ruiz et al. 2007) propose an integrated method for permutation flow shop problem to minimize the makespan. The authors have used reliability models to determine the maintenance periods keeping a minimum level of reliability during the scheduling horizon. However, the maintenance periods are established without taking into account the production requirements. They are fixed in advance for each machine separately, once and for all. Then, machines reliability is considered as constraint to optimize only one criteria related to production. To schedule maintenance actions in a sequence of production, the authors used a strategy known as preserving strategy since the maintenance tasks are always advanced when there is overlapping with production jobs. In the long term, this criterion will certainly increase the maintenance costs. At present, we can note that all these studies are only interested in one criterion always related to production aspect. In (Kaabi et al. 2003), the objective function is a weighted sum of two criteria: total tardiness for production aspect and lateness and tardiness sum for maintenance aspect. The authors have not used reliability models in their study but it is the only one having tried to jointly consider a criterion related to a production aspect and another related to a maintenance aspect. They have adopted the scalar function as fitness function to select solutions. This method is easy to implement by transforming the problem into a single optimization problem. However, in this paper, the weights of the two objective functions are fixed during the execution of the algorithm, driving with only one solution. In this study, no precaution was taken to avoid certain known disadvantages of this method. Moreover, if the Pareto front is not convex then the concave areas of the front remain inaccessible even if the weight values are changed.

The maintenance and production services must collaborate to achieve a common goal, that of maximizing system productivity. To do that, both objectives of maintenance and production must be regarded with the same level of importance. However, criteria related to production are generally antagonist with those of maintenance: the decrease of one raises the other one and conversely. Hence a solution of the joint production and maintenance problem must be a trade-off between the objectives of the two services. Multi-objective evolutionary methods are the most appropriate to find these trade-offs. In addition, the ultimate goal of the manager is generally to have a system which is the most available (or the least unavailable) possible enabling him to execute the production jobs as soon as possible.

In this paper, we propose an integrated bi-objective model for parallel machine problem using reliability models to take into consideration the maintenance aspect. Pareto evolutionary algorithms will be applied to generate several compromise solutions minimizing simultaneously two important criteria in the manufacturing systems: the makespan and the unavailability of the system. The intervals of the maintenance periods as their numbers for every machine are optimized during the global optimization process as well as the sequence of production. Moreover, reliability has used as performance criteria in the model. In other words, it is the unavailability of the entire system that is optimized instead of separately fixing a threshold of reliability for every machine as in (Ruiz et al. 2007). In addition, we proposed a criterion to insert PM actions that we called rational strategy (in opposition to the strategy proposed in (Ruiz et al. 2007) called conservative strategy) which makes it possible to delay or to advance a maintenance task. This criterion allows finding a certain balance between the unavailability of the system and the maintenance costs. To our knowledge, there are no works related to the integrated approach dedicated to parallel machine case, dealing with the problem according to a Pareto evolutionary approach and taking into account the machines unavailability as a performance criterion of the production system.

Modeling of the integrated problem

This section describes separately the definition of the production scheduling problem and PM planning problem, then the bi-objective integrated model proposed.

From production point of view, we will consider the identical parallel machine scheduling problem and the makespan as performance measure. We suppose in our investigations that jobs are available at the beginning of the production period and preemption is not allowed. In parallel-machine problem, there are generally two decisions to be made. One is to assign jobs to the machines and the other is to determine the sequence of the jobs on each machine. This problem is known to be NP-hard (Garey and Johnson 1979). Numerous studies have been conducted to solve this problem either by exact methods for problems of moderate sizes or by developing heuristics.

From maintenance point of view, we will focus our study on the systematic preventive maintenance. The PM actions help to keep production tools in good operating conditions (they increase the availability of a system) and allow decreasing the costs by avoiding unexpected failures. The problem for the maintenance aspect is to determine the PM intervention' dates for each machine minimizing the system unavailability. *The availability* is defined as "the probability that a system or a component is performing its required function at a given point in time or over a stated period of time when operated and maintained in a prescribed manner" (Ebeling 1997).

The availability of a machine M is defined at a given point in time t as:

$$A(t) = P(M \; is \; operating \; at \; time \; t) \tag{1}$$

The opposite of the availability is the unavailability, defined as:

$$\overline{A}(t) = 1 - A(t) \tag{2}$$

The availability expression of a machine M_i depends on its failure rate λ_i and its repair rate μ_i. Here, we consider only machines whose failure rates λ_i and repair rates μ_i are constant. In other words, we assume the time to failure (*resp.*

time to repair) of a machine M_i is represented by an exponential probability distribution having failure rate parameter λ_i (*resp*. repair rate parameter μ_i). We also suppose that PM actions are used to restore the machine to "as good as new" condition. By taking into account these hypotheses, from the initial instant $t = 0$, the availability of a machine M_i at time t is given by the following expression (Ebeling 1997; Villemeur 1991):

$$A_i(t) = \frac{\mu_i}{\lambda_i + \mu_i} + \frac{\lambda_i}{\lambda_i + \mu_i}\exp[-(\lambda_i + \mu_i)t] \tag{3}$$

The availability $A_i\ (t)$ of a machine M_i being a time decreasing function, the unavailability $1 - A_i(t)$ is therefore a time increasing function. Then, if no PM action is performed on M_i, its unavailability will increase. If T is the completion time of a PM action on the machine M_i, the expression of the availability $A_i(t)$ at time t is given by the following expression (Ebeling 1997; Villemeur 1991):

$$A_i(t) = \frac{\mu_i}{\lambda_i + \mu_i} + \frac{\lambda_i}{\lambda_i + \mu_i}E(t), \tag{4}$$

where $E(t) = \exp[-(\lambda_i + \mu_i)(t - T)]$.

The system availability depends on the system structure (parallel, serial or hybrid) as well as its component characteristics. For m independent parallel components, each having an availability function $A_i(t)$, the system availability $A_s(t)$ at time t is given by (Ebeling 1997; Villemeur 1991):

$$A_s(t) = 1 - \prod_{i=1}^{i=m}[1 - A_i(t)] \tag{5}$$

Consequently, the system unavailability is:

$$\overline{A}_s(t) = 1 - A_S(t) = \prod_{i=1}^{i=m}[1 - A_i(t)] \tag{6}$$

The integrated model take into consideration two objectives to be optimized simultaneously: the minimization of makespan for the production aspect and the minimization of system unavailability for the maintenance aspect under the constraints defined previously. Thus, two decisions must be taken simultaneously. The first one is to find the best assignment of n jobs to m parallel machines in order to minimize the makespan. The other one is to decide when to perform the PM actions in order to minimize the system unavailability, the number of PM actions on each machine being not fixed in advance. Both objectives contribute to the system productivity, but they are antagonist. Indeed, if PM actions are carried out, the system unavailability will decrease but the makespan will increase. Conversely, if we don't carry out PM actions, the system unavailability will increase but the makesapn will decrease.

Let C_j be the completion time of a job j and $C_{\max}$ be the completion time of the last job carried out (the makespan): $C_{\max} = max_{j=1,n}\ \{C_j\}$.

Let $T = \{0, t_1, t_2, \ldots, t_s, C_{\max}\}$ where $t_1, t_2, \ldots, t_s$ are the starting times of the PM actions on all machines. Since the unavailability is an increasing function in each interval $[t_i, t_{i+1}]$, $i = 0, \ldots, s$, with $t_0 = 0$ and $t_{s+1} = C_{max}$, and as assumed a machine becomes "as good as new" at the end of each PM action, the system unavailability is only computed at the times $t_1, t_2, \ldots, t_{s+1}$. The processing time of a PM action on machine M_i is assumed to be the mean time of preventive action, whose value is equivalent to $1/\mu_i$ (Adzapka et al. 2004).

The two objectives functions to be minimized, under the constraints defined previously, are:

$F_1 = \{C_{max}\}$, which is the makespan.

$F_2 = \{\max_{t\in T}\{\overline{A}_s(t) = \prod_{i=1}^{i=m}[1 - A_i(t)]\}\}$, which is the system unavailability.

Solving method

As solution methods, we implemented and compared two genetic algorithms. The first one is based on the scalar function for the selection of parents. The second one is the well-known NSGA-II (Non dominated Sorting Genetic Algorithm) which is based on the non dominance concept. The first choice is motivated by the fact that the only study having taken into account two criteria used the weighted sum of objectives as fitness function. The choice of NSGA-II is justified by its effectiveness in the multi-objective optimization field. However, the use of the weighted sum model with fixed weight values such as it is used in (Kaabi et al. 2003) will not be fair compared to NSGA-II. Therefore, we consider the weighted sum genetic algorithm with the same operators and advantage that NSGA-II: same selection, crossover and mutation operators. Moreover, an elitist strategy will be used for this algorithm by adding a secondary population, where elite (nondominated) solutions will be stored. The objective weights will be randomly generated to select a pair of solutions. With these characteristics, this algorithm is similar to the non hybrid version of MOGLS algorithm described in (Ishibuchi et al. 2003), except the selection scheme. Thereafter, it will be named WSGA (Weighted Sum Genetic Algorithm) to differentiate it from MOGLS.

In multiobjective optimization, if two objective functions (f_1 and f_2) are to be minimized then for any two decision vectors x and y, we say that x dominates y ($x \prec y$) if ($f_1(x) \leq f_1(y)$ and $f_2(x) < f_2(y)$) or ($f_1(x) < f_1(y)$ and $f_2(x) \leq f_2(y)$). The non dominated solutions set obtained by an evolutionary algorithm is called Pareto front or tradeoff surface.

In WSGA, the following weighted sum of the two objectives is used as fitness function.

$$f(x) = w_1 f_1(x) + w_2 f_2(x) \tag{7}$$

where w_1 and w_2 are nonnegative weights for the two objectives, which satisfy the following relations:

$$w_i \geq 0 \text{ for } i = 1, 2 \text{ and } w_1 + w_2 = 1. \tag{8}$$

The fitness function (7) subject to relations (8) is used to select a pair of new solutions from a pair of parent solutions by crossover and mutation. The weight values are randomly specified whenever a pair of parent solutions is selected. That is, when N pairs of parent solutions are selected for generating a new population of N solutions, N different weight couples are randomly generated. This means that N search directions are explored in a single generation. However, if constant weight values are used, the search direction is fixed. As in (Ishibuchi et al. 2003), in WSGA we used an elitist strategy by storing all non dominated solutions obtained during its execution in a secondary population. A few nondominated solutions are randomly selected from the secondary population and their copies are added to the current population. Let N_{pop} be the population size and N_{elite} be the number of nondominated (i.e., elite) solutions added to the current population. Using these notations, the WSGA algorithm can be written as follows.

Algorithm 1 WSGA

```
Step 0: Randomly generate an initial population of size $N_{\text{pop}}$.
Repeat
  Step 1: Evaluate the objectives for each in the current population. Then update the secondary population with the nondominated solutions obtained from the current population.
  Step 2: Select $(N_{\text{pop}} - N_{\text{elite}})$ pairs of parents by repeating the following procedures:
      a) Randomly specify the weights $w_1$ and $w_2$.
      b) Select a pair of parents based on the scalar fitness function in (7). The binary tournament selection is used.
  Step 3: Perform evolutionary operators (crossover and mutation) to each of the selected $(N_{\text{pop}} - N_{\text{elite}})$ pairs of parents to generate new two solutions fom each pair.
  Step 4: Randomly select $N_{\text{elite}}$ solutions from the secondary population. Then add their copies to the $(N_{\text{pop}} - N_{\text{elite}})$ solutions generated in Step 3 to construct a population of solutions.
Until the stopping condition is met
```

NSGA-II is an elitist multi-objective evolutionary algorithm which calculates an approximation of the non dominated solutions set, based on the non dominance concept. At each generation a ranking procedure is used to identify the different fronts of non dominated solutions. The diversification is ensured by a crowding operator. Several studies in the research literature classified NSGA-II among the most competitive algorithm in multi-objective optimization (Basseur 2005; Coellom and Cortes 2002; Deb et al. 2000; Gaspar-Cunta and Covas 2003 and Ishibuchi et al. 2003). Without being exhaustive, this algorithm has been used in the following fields: in chemistry to optimize polymer extrusion (Gaspar-Cunta and Covas 2003), in vehicle routing optimization (Velasco et al. 2006), in scheduling (Landa-Silva et al. 2003; Vilcot et al. 2006) and in supply chain optimization (Amodeo et al. 2007). Also, NSGA-II is often chosen to compare the performance of new or existent algorithms. In the following we describe the general operating of NSGA-II and give its pseudo code according to the original version (Deb et al. 2000).

First, a random initial population P_0 of size N is generated. The population is sorted on several fronts based on the non-domination. Each solution is measured by its non-domination level (1 is the best level). Then, binary tournament selection, crossover and mutation operators are used to create a child population Q_0 of size N. For a generation $t \geq 1$, the procedure is different. The first phase consists of creating the population $R_t = P_t U Q_t$ of size $2N$ and applying the non dominated sorting procedure (ranking) to return the list of the non-dominated fronts. At the second phase, a new parent population P_{t+1} containing the N best solutions of R_t is built by including the best fronts as long as the number of solutions in P_{t+1} is less than N. To complete P_{t+1} with the remaining $N - |P_{t+1}|$ solutions, the crowding procedure is applied to the first front not included. The population P_{t+1} is then used to create a new child population Q_{t+1} of size N by applying selection, crossover and mutation operators. The pseudo code of NSGA-II is given by Algorithm2.

Algorithm 2 NSGA-II

```
1:  Create initial populations $P_0$ and $Q_0$ of size $N$
2:  While stopping conditions are not verified do
3:     Create population $R_t = P_t \text{ U } Q_t$
4:     Construct the different fronts $F_i$ of $R_t$ by the
       nondominated sorting procedure (ranking)
5:     Put $P_{t+1} = \Phi$ and i = 0,
6:     While $|P_{t+1}| + |F_i| < N$ do
7:        $P_{t+1} = P_t \text{ U } F_i$
8:          i = i + 1
9:     End While
10:    Include in $P_{t+1}$ the $(N - |P_{t+1}|)$ of $F_i$ according
       to the crowding procedure
```

```
11:   Create $Q_{t+1}$ from $P_{t+1}$ by Selection, crossover
      and mutation
12: End While
```

The following procedure (Algorithm 3) computes the crowding distance for each solution i in population S.

Algorithm 3 Distance_crowding (S)

```
1: L = |S|                                    // number of solutions in S
2: For each i, $S[i]_{distance} = 0$          // initialize distance
3: For each objective m
4:   S = sort (S, m)                          // sort using each objective function
5:   $S[1]_{distance} = S[L]_{distanc} = \infty$   // boundary solutions are always selected
6:   For i=2 to (L−1)                         // for all other points
7:     $S[i]_{distanc} = S[i]_{distance} + (S[i+1].m - S[i-1].m)$
       S[i].m is the m^th objective function value of the i^th
       solution in population S.
```

The non-dominated sorting procedure (Algorithm 4) returns a list of the non-dominated fronts F_i when applied to a population P. Two entities are calculated for each solution: n_i, the number of solutions which dominate the solution i and S_i a set of solutions which the solution i dominates.

Algorithm 4 Non_dominating_sorting (P)

```
1: For each $p \in P$
2:   For each $q \in P$
3:   If $(p \prec q)$ then $S_q = S_q$ U {q}       // if p dominates q then include q in S
4:   Else If $(q \prec p)$ then $n_p = n_p + 1$    // if q dominates p then increment $n_p$
5:   If $n_p = 0$ then $F_1 = F_1$ U {p}           // if no solution dominates p then p is member of the 1st front
6: i = 1
7: While $F_i \neq \Phi$ do
8:   $H = \Phi$
9:   For each $p \in F_i$
10:    For each $q \in S_p$
11:      $n_q = n_q - 1$                          // decrement $n_q$ by one
12:      If $n_q = 0$ then H = H U {q}
13:  i = i + 1
14:  $F_i = H$    // current front is formed with all members of H
15: End While
```

A solution (or a chromosome) is composed of two segments: production and maintenance parts. The production part is a sequence of job numbers. The maintenance part is a series of numbers representing the theoretical periods of PM actions. The maintenance periods are randomly generated for each machine i within the interval $[P_i, C_i]$. P_i is the processing time of the shortest job assigned to machine i and C_i is the completion time of the latest job assigned to machine i. For instance, the series (10, 18) concerns two machines and means a PM action on M_1 should be done every 10 units of time and on M_2 every 18 units of time.

The chromosome evaluation is done according to Algorithm 5. The list of production jobs is assigned to machines according to the "list scheduling" heuristic: assign the next job in the list to the first idle machine.

The maintenance activities are inserted according to the rational strategy: Let B_j and C_j the beginning (*resp.* the completion) time of a production job j. Let T_e be the expected date to perform the maintenance task. If $C_j - T_e \geq T_e - B_j$ then the maintenance task is advanced to the date B_j, else it is delayed to the date C_j.

Algorithm 5 Evaluate_solution

```
Step 1: Assign production jobs to machines according to
        the list scheduling heuristic.
Step 2: Calculate the makespan $C_{max}$.
Step 3: Insert the PM tasks in production scheduling using
        the rational strategy and update production jobs'
        dates.
Step 4: Reevaluate the makespan after inserting the PM
        actions.
Step 5: Compute the system unavailability. It is equal to the
        greatest value of the system unavailability encoun-
        tered during the scheduling horizon.
```

Example

Let a production system be composed of two parallel machines on which eight jobs are to be scheduled. The characteristics of production jobs are:

Job n°	1	2	3	4	5	6	7	8
Duration	4	6	8	10	12	14	16	18

We assume the two machines have the same characteristics: $\lambda_1 = \lambda_2 = 0.1$, $\mu_1 = \mu_2 = 0.5$. Thus, the duration of PM actions on both machines is equal to 2 units of time. Suppose

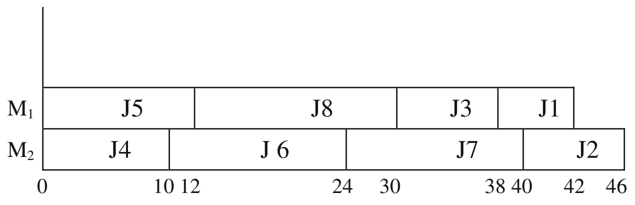

Fig. 1 Gantt chart of production scheduling without PM tasks

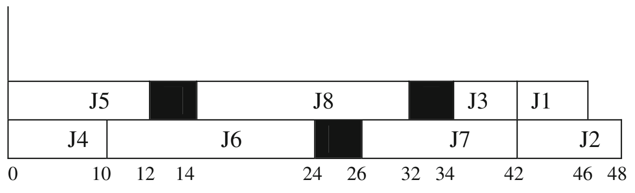

Fig. 2 Gantt chart of the joint production-maintenance scheduling

we have to evaluate the chromosome:

5	4	6	8	7	3	1	2	**16**	**20**

The first eight alleles determine the scheduling of the production jobs. The last two alleles (16, 20) represent the maintenance part which indicates that an intervention on machine M_1 must a priori be carried out every 16 units of time and on the second machine M_2 every 20 units of time. The application of Algorithm 4 to this chromosome is presented by the steps below:

Step 1. The list scheduling heuristic gives the scheduling shown by Fig. 1.
Step 2. The makespan value obtained is $C_{\max} = 46$.
Step 3. Using the rational strategy and updating the production jobs' dates, we obtain the scheduling shown by Fig. 2. The maintenance actions begin at times 12 and 32 for machine M_1 and at time 24 for machine M_2.
Step 4. The makespan recalculated is $C_{\max}$ =48.
Step 5. The system unavailability is computed at instants: 12, 24, 32 and $C_{\max}$. The system unavailability corresponds to the greatest value among those obtained at the four instants.

Table 1 (*resp.* Table 2) gives the unavailability (*resp.* availability) of each machine at the four instants. The system unavailability (*resp.* availability) is computed by formula (6) (*resp.* (5)). Thus, the system unavailability (*resp.* availability) is equal to 0.0809 (*resp.* to 0.9190).

The vector of objective functions for this solution is $(F1, F2) = (48, 0.0809)$.

The initial population is randomly generated. Several crossover and mutation operations were examined in genetic algorithms for scheduling problems, where good results were obtained from the combination of the two-point crossover and the insertion mutation (Ishibuchi et al. 2003). The stopping condition is a fixed number of generations.

For both production and maintenance part, a two-point crossover method is implemented. The first (*resp.* second) offspring is generated by keeping the extremities of the first (*resp.* second) parent and completing it with the missed jobs according to their appearance order in the second (*resp.* first) parent, as illustrated in Fig. 3.

Concerning the mutation operators, for the maintenance part a gene is randomly selected and its value is replaced with another value randomly generated. For the production part, we used a shift (or insertion) mutation as illustrated in Fig. 4, where a randomly selected job is removed and inserted into a randomly selected position. The neighborhood structure

Table 1 Unavailability of machines and system

t	$\overline{A}_1(t)$	$\overline{A}_2(t)$	$\overline{A}_S$(t)
12	0.2814	0.2814	0.0792
24	0.2770	0.2856	0.0791
32	0.2851	0.2507	0.0715
48	0.2835	0.2855	0.0809

Table 2 Availability of machines and system

t	$A_1(t)$	$A_2(t)$	A_S(t)
12	0.7185	0.7185	0.9207
24	0.7229	0.7143	0.9208
32	0.7148	0.7492	0.9284
48	0.7164	0.7144	0.9190

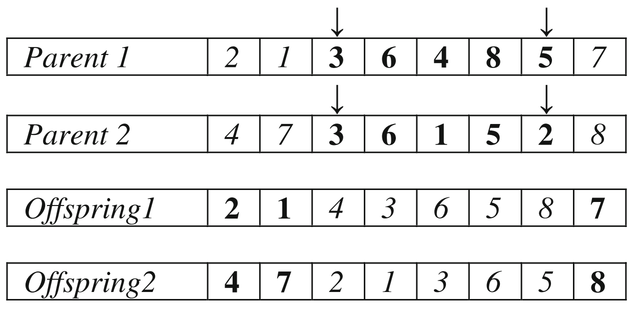
Parent 1 | 2 | 1 | 3 | 6 | 4 | 8 | 5 | 7
Parent 2 | 4 | 7 | 3 | 6 | 1 | 5 | 2 | 8
Offspring1 | 2 | 1 | 4 | 3 | 6 | 5 | 8 | 7
Offspring2 | 4 | 7 | 2 | 1 | 3 | 6 | 5 | 8

Fig. 3 Two-point crossover

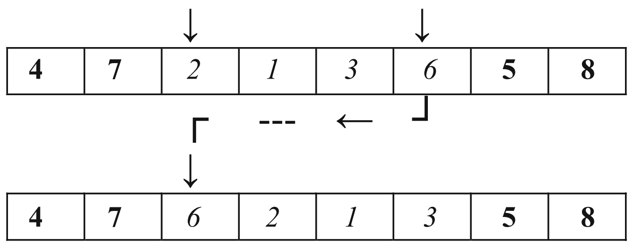

Fig. 4 Shift mutation

defined by the shift mutation was often used for taboo search and simulated annealing algorithms.

Tests and results

Performance measures

In multi-objective problems, a set of solutions cannot be totally ordered, because a solution may be the best for some objectives but not for other objectives. The comparison between two solutions remains a tricky problem seeing that several aspects are taken into consideration: the distance of the resulting non-dominated front to the Pareto-optimal front should be minimized, a good (in most cases uniform) distribution of the solutions found is desirable and the spread of the obtained non-dominated front should be maximized, i.e., for each objective a wide range of values should be covered by the non-dominated solutions (Zitzler 1999). Unfortunately, the different aspects cannot be taken into account by one numerical value. Hence, several measures (or metrics) are generally used to test a particular aspect of a tradeoff surface. There are several metrics in the research literature, in particular in (Barichard 2005; Basseur 2005; Zitzler 1999; Zydallis 2003). These metrics allow quantifying one or more aspects of the solution quality: convergence, distribution and spread. Certain metrics are used when the Pareto front is known and others if it is unknown.

The first metric we have used is the hyper-volume (or hyper-area) metric H, introduced by (Zitzler 1999). This metric calculates an approximation of the volume included under the curve formed by the Pareto front. When two Pareto fronts contain the same number of points, the smaller the value of H the better is the front. It is used and recommended by many researchers to evaluate Pareto sets (Barichard 2005; Basseur 2005; Fleischer 2003; Knowles and Hughes 2005; Knowles and Corne 2002; Zitzler 1999), especially since (Fleischer 2003) has shown that optimization of this metric is a necessary and sufficient condition for Pareto front optimization. This measure reflects different aspects of solution quality in particular convergence and spread. In the two dimensional (bi-objective) case, the volume H is the union of the rectangles defined by points (0, 0) and $(f_1(x_i),\ f_2(x_i))$, for a decision vector $x_i (i = 1, n)$.

Another important metric often used is the C metric introduced also by (Zitzler 1999). It's a relative measure which allows clearly differentiating two fronts A and B when other metrics (the hyper-volume for example) give very similar values. The value of $C(A, B)$ represents the percentage of solutions in B dominated by at least one solution of A. It is computed by formula (9).

$$C(A,\ B) = \frac{|\{b \in B / \exists a \in A : a \prec b\}|}{|B|} \tag{9}$$

The value $C(A,\ B) = 1$ means that all solutions in B are dominated by A. The opposite, $C(A,\ B) = 0$, represents the situation when none of the points in B are dominated by A. Thus, the closer the value of $C(A, B)$ to 1, the better is the front A compared to B. Since this measure is not symmetrical, that is $C(A, B) \neq 1 - C(B, A)$, it is necessary to calculate $C(B, A)$. Therefore, A is better than B if $C(A, B) > C(B, A)$.

Computational results

To compare between the two algorithms used to solve our problem, we generated ten m-machine, n-job parallel machine problems. Using the number of machines (m) and the number of jobs (n), we noted each test problem by the couple (m, n). The test problems proposed are (2, 10), (2, 20), (3, 20), (3, 40), (3, 60), (5, 20), (5, 40), (5, 60), (8, 40) and (8, 60). The processing time of each job was specified as a random integer in the interval [1, 50]. In multi objective optimization, it is not easy to compare efficiently two algorithms, especially if the two algorithms include several parameters. Hence, as in Ishibuchi et al. (2003), we examined 27 combinations of the following parameter values:
Population size (N_{pop}): 30, 60, 120.
Crossover probability (p_c): 0.6, 0.8, 1.0.
Mutation probability (p_m): 0.4, 0.6, and 0.8 (per string).

We used a number of generations of 100 for the two algorithms. The number of elite solutions at each generation is 10 ($N_{\text{elite}} = 10$), which is the same value selected in (Ishibuchi et al. 2003) for their MOGLS algorithm. The crossover probability (p_c) and mutation probability (p_m) are related to the production part. For the maintenance part, these two parameters are set to 0.8 and 0.01 (per bit) respectively. Each of the two algorithms was applied to each test problem 20 times for each of the 27 combinations of the parameter values. Therefore, 540 solution sets (nondominated fronts) were obtained by each algorithm for each test problem. For each run, a new instance of processing times is generated. Three performance measures were used to compare the two algorithms: the hyper volume metric H, the C metric and the number of solutions obtained. We have used the Mann-Whitney test to test if there exist a significant differences between the results obtained from the two algorithms for a certain performance measure. The Mann-Whitney U test is a non-parametric test to determine whether two samples of observations come from the same distribution. The null hypothesis is that there is no significant differences between the two samples (they come from a single population) and thus that their probability distribution are equal. The conditions to its application are the independence of the two samples and the type of observations must be ordinal or continuous.

Fair comparison between two solution sets with the H metric requires the equality of the number of solutions

Table 3 Best, average and worst values of the number of obtained solutions over 540 solutions sets obtained by each algorithm for each test problem

Test problem	WSGA			NSGA-II		
(m, n)	Best	Av.	Worst	Best	Av.	Worst
(2, 10)	7	3.31	1	10	5.27	2
(2, 20)	7	2.59	1	12	5.91	1
(3, 20)	7	3.52	1	11	6.17	2
(3, 40)	5	2.41	1	14	5.52	1
(3, 60)	4	2.03	1	14	4	1
(5, 20)	6	3.22	1	9	3.91	1
(5, 40)	7	3.03	1	11	5.65	1
(5, 60)	5	2.34	1	11	4.48	1
(8, 40)	3	1.46	1	4	1.70	1
(8, 60)	2	1.23	1	3	1.30	1

Table 4 Best, average and worst values of the C metric over 540 solutions sets obtained by each algorithm for each test problem

Test problem	WSGA			NSGA-II		
(m, n)	Best	Av.	Worst	Best	Av.	Worst
(2, 10)	1	0.19	0	1	0.42	0
(2, 20)	1	0.11	0	1	0.49	0
(3, 20)	1	0.17	0	1	0.42	0
(3, 40)	0.67	0.12	0	1	0.50	0
(3, 60)	1	0.18	0	1	0.46	0
(5, 20)	1	0.24	0	1	0.48	0
(5, 40)	0.8	0.13	0	1	0.49	0
(5, 60)	1	0.14	0	1	0.52	0
(8, 40)	1	0.27	0	1	0.39	0
(8, 60)	1	0.26	0	1	0.45	0

Table 5 Best, average and worst values of the H metric over 540 solutions sets obtained by each algorithm for each test problem

Test problem	WSGA			NSGA-II		
(m, n)	Best	Av.	Worst	Best	Av.	Worst
(2, 10)	4.978949	11.436936	16.408154	5.120944	10.980644	16.408154
(2, 20)	15.591825	23.345837	30.448923	15.673394	24.239733	30.204052
(3, 20)	1.058309	1.619620	2.422747	1.049317	1.609786	2.230322
(3, 40)	2.720125	3.335351	4.023336	2.667646	3.306552	4.049576
(3, 60)	3.778438	4.771540	5.623925	3.778438	4.736960	5.553959
(5, 20)	0.027446	0.045566	0.063287	0.028354	0.045526	0.062471
(5, 40)	0.072459	0.092899	0.117456	0.070385	0.092400	0.118705
(5, 60)	0.107876	0.141149	0.162024	0.105777	0.140193	0.160358
(8, 40)	0.000207	0.000291	0.000369	0.000193	0.000290	0.000367
(8, 60)	0.000338	0.000430	0.000504	0.000331	0.000429	0.000502

obtained by the two sets. Therefore, initially the two algorithms were only compared according to two criteria, the number of solutions obtained and the C metric. Then, other comparisons were carried out with an equal number of solutions obtained (WSGA is executed first, the comparison is accepted only if the number of solutions obtained by NSGA-II is equal to that of WSGA) according to both metrics H and C. Table 3 gives the best, average, and worst values of the number of solutions obtained from the 540 solution sets. As we can see from this table, the two methods may obtain low values for this criterion. In general, the number of solutions obtained decreases as the number of machines and tasks increase. This is related to the unavailability function specification and the fact that the machines are in parallel. For this criterion, NSGA-II significantly outperforms WSGA with the 99% confidence level by the Mann-Whitney U test.

Table 4 gives the best, the average, and the worst values of the C metric obtained over the 540 solution sets. For each test

Table 6 Best, average and worst values of the C metric over 540 solutions sets obtained by each algorithm for each test problem

Test problem	WSGA			NSGA-II		
(m, n)	Best	Av.	Worst	Best	Av.	Worst
(2, 10)	1	0.20	0	1	0.48	0
(2, 20)	1	0.20	0	1	0.46	0
(3, 20)	1	0.18	0	1	0.50	0
(3, 40)	1	0.28	0	1	0.46	0
(3, 60)	1	0.34	0	1	0.43	0
0(5, 20)	1	0.20	0	1	0.52	0
(5, 40)	1	0.18	0	1	0.52	0
(5, 60)	1	0.30	0	1	0.47	0
(8, 40)	1	0.28	0	1	0.40	0
(8, 60)	1	0.30	0	1	0.47	0

Table 7 Comparison of the two algorithms using the H metric over 540 solutions sets obtained by each algorithm for each test problem

(m, n)	WSGA	NSGA-II
(2, 10)	10.975436 (0.107571)	10.889418 (0.103722)
(2, 20)	23.289782 (0.139554)	23.881611 (0.097461)
(3, 20)	1.597960 (0.010041)	1.573420 (0.008866)
(3, 40)	3.247095 (0.013904)	3.348115 (0.012401)
(3, 60)	4.751911 (0.012626)	4.564301 (0.014518)
(5, 20)	0.044163 (0.000216)	0.044030 (0.000213)
(5, 40)	0.089986 (0.000556)	0.090169 (0.000515)
(5, 60)	0.138199 (0.000386)	0.136138 (0.000382)
(8, 40)	0.000296 (0.000001)	0.000294 (0.000001)
(8, 60)	0.000429 (0.000001)	0.000427 (0.000001)

Table 8 Comparison of the two algorithms using the C metric over 540 solutions sets obtained by each algorithm for each test problem

(m, n)	WSGA	NSGA-II
(2, 10)	0.20 (0.011)	0.52 (0.014)
(2, 20)	0.17 (0.013)	0.31 (0.017)
(3, 20)	0.19 (0.010)	0.53 (0.012)
(3, 40)	0.12 (0.010)	0.33 (0.012)
(3, 60)	0.32 (0.014)	0.70 (0.018)
(5, 20)	0.30 (0.016)	0.32 (0.013)
(5, 40)	0.19 (0.009)	0.46 (0.013)
(5, 60)	0.36 (0.015)	0.37 (0.013)
(8, 40)	0.15 (0.017)	0.35 (0.023)
(8, 60)	0.20 (0.018)	0.60 (0.023)

problem, we can see that one of the two methods completely dominated the other at least once over the 27 combinations for each test problem, except for two test problems where WSGA had not managed to entirely dominate NSGA-II. It is obvious that NSGA-II significantly outperforms WSGA on the average.

Table 5 (*resp*. 6) presents the best, average and worst values of the H (*resp*. C) metric obtained over the 540 solution sets when the equality of the number of solutions is required to carry out comparisons. On the average, there is no significant difference in terms of the H metric between the two methods with the 99% confidence level by the Mann-Whitney U Test. On the other hand, NSGA-II clearly outperforms WSGA in terms of the C metric as we can see in Table 6.

Moreover we tested the performance of each algorithm using the best combination of the three parameters N_{pop}, p_{c} and p_{m} for each test problem. We chose the combination of the parameters having given the best solution set for each test problem for each algorithm in Table 5. The obtained results by each algorithm for each test problem using the H metric and the C metric can be seen in Tables 7 and 8, respectively.

We can see from Table 7 that WSGA outperformed NSGA-II for three test problems in terms of the H metric. Moreover, WSGA obtains similar results to NSGA-II for two test problems in terms of the C metric. However, for the other test problems NSGA-II widely outperformed WSGA, particularly in terms of the C metric as we can see in Table 8.

Conclusions

In this paper, we have proposed a bi-objective integrated model to solve the joint production and maintenance scheduling problem in parallel machine case. The reliability of the production system is considered as a criterion of performance. Two important criteria in the manufacturing systems are fairly optimized: the makespan and the unavailability of the system. We have proposed a new criterion to schedule maintenance preventive actions within the production sequence which allows finding a certain balance between the unavailability of the system and the maintenance costs. We have tested two multi objective evolutionary algorithms adapted to our problem. The Pareto solutions obtained are tradeoffs between the two objectives, enabling the manager to make two types of decisions at the same time: Deciding to which machine each production job is assigned and deciding when to carry out maintenance actions on each machine such that the objectives of both services are simultaneously optimized. Computational evaluations with ten test problems and a total of 5400 instances have showed that the genetic algorithm based on dominance concept provide very efficient solutions than the one based on the scalar function according to two performance measures.

We plan to study this problem by taking into account other criteria like the total tardiness for the production or the costs of PM through the number of interventions. We also think to include in our problem constraints related to production or maintenance. Other production environments more complex like hybrid systems are envisaged. From point of view of solution methods, we are interested to study the effect of hybridization of the genetic algorithms used in our study.

Acknowledgements The authors would like to thank the anonymous referees for their insightful comments on an earlier version of this paper. Their suggestions led to a much better presentation of the paper.

References

Adzapka, K. P., Adjallah, K. H., & Yalaoui, F. (2004). On-line maintenance job scheduling and assignment to resources in distributed systems by heuristic-based optimization. *Journal of intelligent manufacturing, 15*, 131–140.

Amodeo, L., Chen, H., & El-Hadji, A. (2007). *Multi-objective supply chain optimization: An industrial case study* (pp. 732–741). Springer-Verlag, EvoWorkshops, LNCS 4448.

Barichard, V. (2005). *Hybrid approaches for multiobjective problems*. PhD thesis, Doctoral School of Angers, France (in French).

Basseur, M. (2005). *Design of cooperative algorithms for multiobjective optimization: Application to Flowshop scheduling problems*. PhD thesis, University of Sciences and Technologies of Lille, France (in French).

Cassady, C. R., & Kutanoglu, E. (2003). Minimizing job tardiness using integrated preventive maintenance planning and production scheduling. *IIE Transactions, 35*, 503–513.

Coellom, C. A., & Cortes, N. C. (2002). *Solving multi-objective optimization problems using an artificial immune system*. Evolutionary Computation Group, Instituto Politécnico Nacional No. 2508 Col. San Pedro Zacatenco México.

Deb, K., Agrawal, S., Pratab, A., & Meyarivan, T. (2000). *A fast elitist non dominated sorting genetic algorithm for multi-objective optimization: NSGA-II*. KanGAL report 200001, Indian Institute of Technology, Kanpur, India.

Ebeling, C. E. (1997). *An introduction to reliability and maintainability engineering*. USA: McGraw-Hill.

Fleischer, M. (2003). The measure of Pareto optima, applications to the multi-objective metaheuristics. *Lecture Notes in Computer Science, Springer (EMO'03), 2632*, 519–533.

Garey, M. R., & Johnson, D. S (1979). *Computers and Intractability: A guide to the theory of NP-Completeness*. California, San Francisco: W.H., Freeman and Company.

Gaspar-Cunta, A., & Covas, J. A. (2003). A real-word test problem for EMO algorithms. *Lecture Notes in Computer Science, Springer, 2632*, 752–766.

Ishibuchi, H., Yoshida, T., & Murata, T. (2003). Balance between genetic search and local search in memetic algorithms for multi-objective permutation flowshop scheduling. *IEEE Transactions on Evolutionary Computation, 7*(2), 204–223.

Kaabi, J., Varnier, C., & Zerhouni, N. (2002). Heuristics for scheduling maintenance and production on a single machine. *IEEE Conference on Systems, Man and Cybernetics*. October 6–9 Hammamet, Tunisia.

Kaabi, J., Varnier, C., & Zerhouni, N. (2003). *Genetic algorithm for scheduling production and maintenance in a Flow Shop*. Laboratory of automatic of Besançon, France (in French).

Kaspi, M., & Montreuil, B. (1988). *On the scheduling of identical parallel processes with arbitrary initial processor available time*. Technical report, School of Industrial Engineering, Purdue University.

Knowles, J., & Corne, D. (2002). On metrics for comparing nondominated sets. *Proceedings of the 2002 Congress on Evolutionary Computation (CEC 2002)*, IEEE, pp. 711–716.

Knowles, J., & Hughes, E. J. (2005). *Multi-objective optimization on a budget of 250 evaluations*. School of Chemistry, University of Manchester, UK.

Landa-Silva, J. D., Burke, E. K., & Petrovic, S. (2003). *An Introduction to multi-objective metaheuristics for scheduling and timetabling*. Automated Scheduling, Optimisation and Planning Research Group, School of Computer Science and IT, University of Nottingham, UK.

Lee, C.-Y. (1991). Parallel machine scheduling with non simultaneous machine available time. *Discrete and Applied Mathematics, 30*, 53–61.

Lee, C.-Y. (1996). Machine scheduling with an availability constraint. *Journal of Global Optimization, 9*, 395–416.

Lee, C.-Y., & Chen. Z. L. (2000). Scheduling jobs and maintenance activities on parallel machines. *Naval Research Logistics, 47–2*, 145–165.

Liao, C. J., Chen. C. M., & Lin. C. H. (2007). Minimizing makespan for two parallel machines with job limit on each availability interval. *Journal of the Operational Research Society, 58*(7), 938–947.

Liao, C.-J., Shyur, D.-L., & Lin, C.-H. (2005). Makespan minimization for two parallel machines with an availability constraint. *European Journal of Operational Research, 160*, 445–456.

Liman, S. D. (1991). *Scheduling with capacities and due-dates*. PhD thesis, Industrial and Systems Engineering Department, University of Florida, USA.

Mellouli, R., Sadfi, C., Kacem, I., & Chu, C. (2006). Scheduling on parallel machines with availability constraints. *Sixth International Francophone Conference of Modeling and Simulation, Mosim'06*, Marocco (in French).

Mosheiv, G. (1994). Minimizing the sum of job completion times on capacitated parallel machines. *Mathematical and Computer Modeling, 20*, 91–99.

Ruiz, R., García-Diaz, J. C., & Maroto, C. (2007). Considering scheduling and preventive maintenance in the flowshop sequencing problem. *Computers & Operations Research, 34*(11), 3314–3330.

Schmidt, G. (1984). Scheduling on semi-identical processors. *Zeitschrift fur Operation Research, 28*, 153–162.

Schmidt, G. (2000). Scheduling with limited machine availability. *European Journal of Operational Research, 121*, 1–15.

Velasco, N., Dejax, P., Guéret, C., & Prins, C. (2006). Genetic algorithm for the bi objective collection and delivery problem. *Sixth International Francophone Conference of Modeling and Simulation, Mosim'06*, Marocco.

Vilcot, G., Billaut, J.-C., Esswein, C. (2006). A genetic algorithm for a bi-criteria flexible job shop scheduling problem. *IEEE International Conference on Service Systems and Service Management (ICSSSM'06)*.

Villemeur, A. (1991). *Reliability, availability, maintainability and safety assessment*. USA: Wiley.

Weinstein, L., & Chung, C.-H. (1999). Integrated maintenance and production decisions in hierarchical production planning environment. *Computers and Operations Research, 26*, 1059–1074.

Xu, D., Sun, K., & Li, H. (2008). Parallel machine scheduling with almost periodic non-preemptive maintenance and jobs to minimize makespan. *Computers and Operations Research, 35*, 1344–1349.

Zitzler, E. (1999). *Evolutionary algorithms for multi- objective optimization: Methods and applications*. PhD thesis, Swiss Federal Institute of Technology, Zurich.

Zydallis, J. B. (2003). *Explicit building-block multi-objective genetic algorithms: Theory, analysis, and development*. PhD dissertation, Air Force Institute of Technology, Ohio.