Int J Adv Manuf Technol (2016) 83:475–488
DOI 10.1007/s00170-015-7555-7

ORIGINAL ARTICLE

Multi-objective optimization of collation delay and makespan in mail-order pharmacy automated distribution system

Kevin Mei[2] · Debiao Li[1] · Sang Won Yoon[2] · Jong-hynn Ryu[3]

Received: 26 March 2015 / Accepted: 5 July 2015 / Published online: 25 July 2015

Abstract In this research, the collation delay (CD) and makespan minimization problem in a mail-order pharmacy automated distribution (MOPAD) system are studied. The MOPAD systems, which are integrated with pharmaceutical auto-dispenser machines, auto-packer machines, and conveyor, are utilized to fulfill the increasing prescription demand in recent years. The motivation of this research is derived from the practical deadlock problem in a MOPAD system of the central fill pharmacies (CFP). Most of the customer orders consist of multiple medications, which need to be collated together before being packaged and shipped. The CD is defined as the fulfillment completion time difference between the first and last medications within the same order, which is a critical factor of the MOPAD systems throughput. When CD is minimized, the makespan often increases. Therefore, alternative scheduling solutions are often needed to balance the CD and makespan in the MOPAD system. This paper presents the trade-off solutions between minimizing CD and the makespan. Three multi-objective genetic algorithms with a three-tuples chromosome design, including Vector Evaluated Genetic Algorithm (VEGA), Multi-Objective Genetic Algorithm (MOGA), and non-dominated sorted genetic algorithm-II (NSGA-II), are implemented and compared under various system settings. Compared to the current implemented longest processing time (LPT) heuristic, three multi-objective genetic algorithms save the CD by 95.67 %, but only increase the makespan by 5.62 % on average. The results also show that the NSGA-II provided the best frontier.

Keywords Collation delay minimization · Mail-order pharmacy automated distribution system · Multi-objective genetic algorithm · Three-tuple chromosome

✉ Debiao Li
debiaoli5@gmail.com

Sang Won Yoon
yoons@binghamton.edu

1 Fuzhou University, Fuzhou, Fujian, China

2 State University of New York at Binghamton, Binghamton, New York, USA

3 Hongik University, Sejong, South Korea

1 Introduction

In recent years, the demand for prescription medications has continuously increased, which is in part due to a growing elderly population [13]. The number of prescriptions that are dispensed in the USA climbed from 2.1 billion to 3.9 billion, which is an 86 % increase from 1994 to 2009 [13]. In order to keep up with this trend, the mail-order pharmacy distribution (MOPAD) system has been utilized in the central fill pharmacy (CFP), which allows pharmacies to complete larger workloads in a shorter time, decreases the number of work staff and medication filling errors, improves workflow, and reduces inventory cost [25]. The CFP usually receive orders from hundreds of different retail pharmacy locations, fill and collate the orders within $1-2$ days, and deliver the orders to the local pharmacies or patients. By working with a MOPAD system, a large portion of a pharmacist's workload in filling and verification can be moved to an off-site location and more time can be spent on patient consultations and addressing their needs. The MOPAD system also leads to savings on prescription dispensing and inventory holding costs, improving drug safety and quality [2]. As a result, the MOPAD system

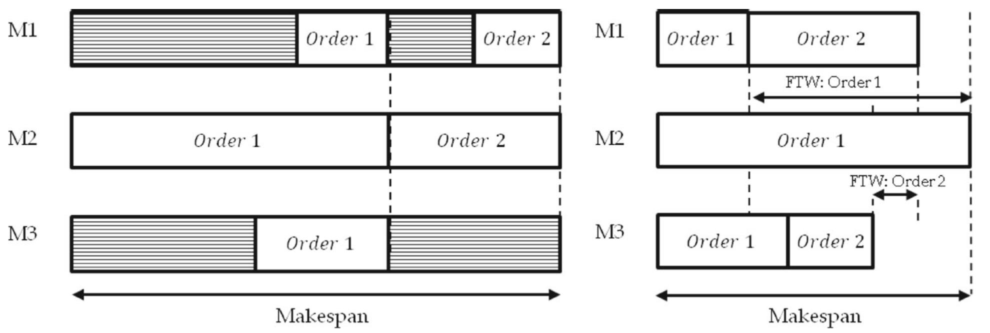


Fig. 1 Relationship between CD and makespan (a) Minimizing CD increases makespan. (b) Minimizing makespan increases CD.

has become more popular and many retail pharmacies are utilizing MOPAD systems to obtain business and societal benefits.

MOPAD systems utilize pharmaceutical auto-dispensing robots and conveyors to fill large amounts of orders with little human intervention. Auto-dispensing robots are comprised of a mechanical arm and hundreds of auto-dispensers. The arm picks up a vial and brings it to one of the dispensers so the correct medication is dropped into the vial. For more commonly ordered medications, multiple dispensers may be used to hold a large quantity of the same medication to reduce replenishment time. This also reduces filling downtime because dispensing can continue from another dispenser while an empty dispenser is being replenished. By utilizing automated robots, more orders can be filled in a shorter time, compared to pharmacists performing the same work. After the medications are filled, a pharmacist must verify before the vial is sealed and put into the robot. Images are taken during the dispensing process and compared to stock photos, which allow the pharmacists to verify the medications remotely and reduce the verification time.

One of the most important factors in the operation of a MOPAD system is the collation delay (CD). The CD is defined as the fulfillment completion time difference between the first and last medications within the same order [16]. Due to the aging population and long-term effects of disease, 50–80 % of prescription orders received in a MOPAD system are multiple medications orders, which may generate CDs. These medicines from the same prescription order may be dispensed from different robots. After being filled, the medicines must be transferred together and collated. The CD is identified as a critical factor to the MOPAD system's throughput according to a factorial multivariate analysis of variance (MANOVA) based on simulation results [17]. The large CDs indicate a large amount of medications are looping in the conveyor and waiting to be collated. When the number of medications exceed the conveyor capacity, the MOPAD system will be a deadlock. No orders can be processed through the system without manually clearing out the collation stations and conveyors after deadlock. This situation can be avoided by having additional resources (e.g., collation stations and conveyors). However, this is not always feasible due to the constraints of cost and space. To solve this problem, the CD problem is proposed as a flexible order scheduling problem with the objective of minimizing the CD. However, just minimizing the CD could have adverse effects on the makespan. As shown in Fig. 1a, the CD is minimized to zero for order 1 and 2 by adding idle times, such that all the jobs within an order are fulfilled at the same time. However, the makespan increases because of the non-value-added machine idle time. If minimizing the makespan is considered as a single objective, the CD becomes greater than zero after applying the longest processing time heuristic, as shown in Fig. 1b. The makespan is conflicting with CD in some cases, particularly when the majority prescription orders are consist of multiple medications. Therefore, alternative scheduling solutions are often needed to balance the CD and makespan in the MOPAD system. This problem is treated as a multi-objective optimization problem in this research, which minimizes both CD and makespan.

This paper discusses the relationship between collation delay and makespan, formulates a bi-objective integer programming, proposes three-tuple chromosome design in the Vector Evaluated Genetic Algorithm (VEGA), Multi-Objective Genetic Algorithm (MOGA), and non-dominated sorted genetic algorithm-II (NSGA-II). A literature review is presented in Section 2. A mathematical model developed for the CD-makespan problem is stated in Section 3, and three multi-objective genetic algorithms are discussed in Section 4. The experimental results are shown in Section 5, followed by the conclusion and future work in Section 6.

2 Literature review

The automations in pharmacies increase the efficiency, consistency, and safety in pharmaceutical practice [10], ensure the five-right of medication (right patient, route, dose, time, and medication), and improve the healthcare quality and

Table 1 List of notations used for the CD-$C_{\max}$ mathematical model

Indexes	Description
i	Order index, $i \in \{1, 2, \ldots, m\}$
j	Job index, $j \in \{1, 2, \ldots, n_i\}$
k	Machine index, $k \in \{1, 2, \ldots, o\}$
l	Position index, $l \in \{1, 2, \ldots, L\}$
Data	
m	Number of orders
n_i	Number of jobs in order i
o	Number of machines
L	Total number of jobs $L = \sum_{i=1}^{m} n_i$
p_{ij}	Given processing time of job j from order i
Variables	
x_{ijkl}	Binary variable, 1 if job j from order i is processed in position l of machine k; 0 otherwise
s_{ijkl}	Integer variable, starting time of job j from order i on machine k in position l
$C_{\max}$	Integer variable, production makespan
$C_i^{\min}$	Integer variable, the earliest completion time of order i
$C_i^{\max}$	Integer variable, the latest completion time of order i

safety [2]. The design of pharmaceutical auto-dispensing systems has been proposed and implemented to increase the feasibility and accuracy of dispensing process [21]. A fuzzy self-adaptive proportional-integral-derivative (PID) control method has been presented to study the dynamic characteristics of the lifter in pharmaceutical auto-dispensing systems, and a GA has been implemented to search its shortest routes [27]. The medication location planning problem in pharmaceutical auto-dispensing systems has also been studied as an location assignment problem [23]. A multi-objective particle swarm algorithm has also been applied to optimize the medicine location and planning. The simulation results indicate that the proposed algorithm reduces the cost and improves the efficiency.

MOPAD systems are utilized to handle the increasing prescription demand in CFP. A typically MOPAD system consists of auto-dispensing machines, auto-cappers, auto-packaging machines, and auto-transportation systems (e.g., conveyors, vacuum transport tube system, etc.). Based on a factorial multivariate analysis of variance (MANOVA) of simulation results, the CD is identified as a critical factor to the MOPAD system's throughput [17]. The CD is defined as the fulfillment completion time difference between the first and last medications within the same order [16]. An adaptive parallel tabu search algorithm is proposed to minimize the CD problem in a MOPAD system [18]. The CD is minimized as a single objective optimization approach, while the makespan is considered as a constraint. A makespan tolerance is enforced by 10 % of the optimal makespan, which means that valid schedules are allowed to have at most 110 % of the optimal makespan. The result from the adaptive parallel tabu search is compared to classical tabu search and longest processing time (LPT) heuristic. The final results showed that the adaptive parallel tabu search algorithm is superior, especially at larger job sizes [18]. However, the makespan often increases as CD is minimized in MOPAD systems. The single solution approach has a limitation in investigating the Pareto frontier set when the trade-off (CD and makespan) surface is non-convex. Therefore, alternative scheduling solutions are considered to minimize the CD and makespan in this research.

There are two general approaches when solving multiple-objective problems. The first is to frame the objective into a single objective or to set all objectives except one as constraints [15]. Optimization of problems using these methods would return only a single solution. The second approach is to find the Pareto optimal solution set, the set of solutions that are non-dominated when compared to each other [15]. This approach is often preferred because it provides a set of solutions that considers trade-offs in resource usage and are more useful when real-life problems are considered. In multi-objective optimization problems, Pareto optimal solutions are points that cannot be improved without compromising at least one of other objectives. The Pareto front is a set of the solutions that are called non-dominated solutions. It means that the solutions cannot be better in terms of all the considered objectives simultaneously [5]. The concept of Pareto optimality is widely applied in the evolutionary metaheuristics for the multi-objective optimization problems [19, 20]. The Pareto-based fitness assignment is first proposed by Goldberg [9]. The Pareto optimality ranking method has been used in a variety of scheduling problems [7, 22]. Although they do not guarantee optimality, they are usually applied to efficiently seek good quality solutions for large and complex problems. There are many

Table 2 Example jobs and their processing times

Order	Job index	Processing time
1	1	3
2	1	2
3	1	7
4	1	4
5	1	9
6	1	8
6	2	14
7	1	2
7	2	9
7	3	5

Table 3 Example machine assignment based on chromosome representation

Machine	Jobs (In order of processing)
1	(7,2) (4,1) (1,1) (2,1) (6,2)
2	(3,1) (5,1) (7,1)
3	(7,3) (6,1)

parameters have to be set by the user, and the solution quality is usually sensitive to the parameter settings [14]. A genetic algorithm is applied to maximize the orders' satisfaction level and minimize their total processing times by considering uncertain orders, processing time, and arrival times [11]. Their results indicate that the algorithm can efficiently find multiple trade-off solutions that simultaneously minimize WIP and makespan. MOGA is proposed and applied to a flowshop scheduling problem [4]. MOGA performs better than the VEGA when the objectives minimized makespan, total completion times, and total tardiness [26]. The NSGA-II proposed by Deb et al. is a well-known algorithm for solving multi-objective optimization problems using a non-dominated approach [6]. It has been successfully applied to many multi-objective problems, e.g., facility location problem [3], multi-site order scheduling problem (MSOS) [12], generation expansion planning problem [24], and constrained grinding optimization for time, cost, and surface roughness [8]. To minimize the CD and makespan in MOPAD, a mathematical model is established and three multi-objective genetic algorithms with a three-tuples chromosome design are proposed in this paper.

3 Mathematical model

In this study, both minimizing makespan and CD are considered as the objective functions. It is assumed that all the automated dispensing machines are identical and are capable of processing one job assigned at a time with a deterministic processing time. There is no set-up time, no breakdowns, and no replenishment needed, such that each machine has L ($L = \sum_{i=1}^{m} n_i$) positions to locate jobs. It is also assumed that the number of jobs in a multi-medication order will not exceed the number of machines. In reality, there is a distance between machines that would be included in the CD calculation. In this research, it is assumed that

Table 4 Example of total processing time by machine

Machine	Total processing
1	32
2	18
3	13

Table 5 Example of start and finish times for jobs using Table 3 sequence

Order	Start	First finish	Last finish	CD
1	13	16	16	0
2	16	18	18	0
3	0	7	7	0
4	9	13	13	0
5	7	17	17	0
6	5	13	32	19
7	0	5	18	13

after an job is finished auto-filling, it is waiting for the other jobs that are part of its order. A mathematical model of the CD-$C_{\max}$ minimization problem is developed based on these assumptions. The notations used in this model is presented in Table 1.

$$f_1 : \min \sum_{i=1}^{\mathrm{m}} \left(C_i^{\max} - C_i^{\min} \right) \tag{1}$$

$$f_2 : \min C_{\max} \tag{2}$$

subject to:

$$s_{ijkl} + p_{ij} \cdot x_{ijkl} \leq C_i^{\max} \quad \forall i, j, k, l \tag{3}$$

$$s_{ijkl} + p_{ij} \cdot x_{ijkl} \geq C_i^{\min} \quad \forall i, j, k, l \tag{4}$$

$$s_{ijkl} + p_{ij} \cdot x_{ijkl} \leq s_{i'j'kl'} \quad \forall i, j \mid i \neq i' \wedge j \neq j', k, l \leq l' \tag{5}$$

$$\sum_{k=1}^{o} x_{ijkl} \leq 1 \quad \forall i, j, l \tag{6}$$

$$\sum_{l=1}^{L} x_{ijkl} \leq 1 \quad \forall i, j, k \tag{7}$$

$$\sum_{i=1}^{m} \sum_{j=1}^{n} \sum_{l=1}^{L} p_{ij} \cdot x_{ijkl} \leq C_{\max} \quad \forall k \tag{8}$$

$$x_{ijkl} \in \{0, 1\} \quad \forall i, j, k, l \tag{9}$$

$$s_{ijkl}, p_{ij}, C_{\max} \in \mathbb{Z}^+ \quad \forall i, j, k, l \tag{10}$$

The collation delay minimization objective, minimizing the time difference between the earliest and latest completed jobs in a multi-medication order for all orders, is shown in Eq. 1. The makespan minimization objective, the minimization of the time to process all jobs, is shown in Eq. 2. Equations 3 and 4 define the earliest and latest completion times within an order, $s_{ijkl} + p_{ij} \cdot x_{ijkl}$. If $x_{ijkl} = 1$ then job j of order i is processed on machine k in position l of the machine and $p_{ij} \cdot x_{ijkl} = p_{ij}$; otherwise, $p_{ij} \cdot x_{ijkl} = 0$. Equation 5 establishes the sequence of jobs being processed on the same machine. If a job j in order i is processed on machine k in position l at time s_{ijkl}, job j' from order i' being processed on the same machine k in a position $l' > l$, must necessarily start at a time after $s_{ijkl} + p_{ij}$, the starting time of job j from order i plus its processing time p_{ij}. i cannot equal i' and j cannot equal j' simultaneously otherwise the exact same job will be referenced twice. A job

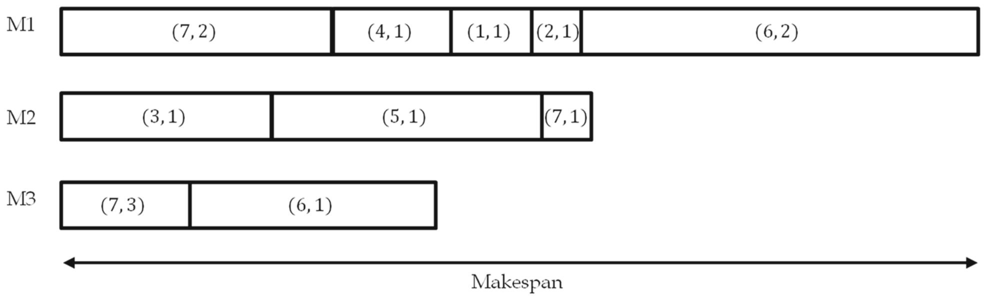


Fig. 2 Example of scheduling jobs based on Table 3 sequence

is only processed on one machine, as stated in Eq. 6. Equation 7 states that a job can only be in one position; a job cannot be split and preempted by another job. The makespan is defined in Eq. 8. x_{ijkl} is a binary variable, and the starting times, processing times, and makespan must be all positive integers, as stated in Eqs. 9 and 10.

4 Methodology

In this research, three multi-objective genetic algorithms with the three-tuples chromosome design, including VEGA, MOGA, and NSGA-II, are applied in solving this CD problem under various conditions. The three-tuples chromosome introduced in Section 4.1 is implemented in multi-objective genetic algorithms. Different fitness function designs of VEGA, MOGA, and NSGA-II are presented in Section 4.2. The crossover and mutation are discussed in Sections 4.3 and 4.4.

4.1 Chromosome design

In order for the data to be input in the GA, it must be preprocessed into a chromosome. The chromosome will contain all the information for the order of jobs to be processed on the assigned machines for each job. Each job will be in the form of a three-tuple (i, j, k). It indicates that job j of order i will be processed on machine k. The order in which the three-tuples appear will determine the order the machine will process jobs. For example, there are seven orders with a total of ten jobs as shown in Table 2. Therefore, if (1,2,3) appears to the left of (2,1,3) then job 2 of order 1 will be processed before job 1 of order 2 on machine 3.

Table 3 shows the resulting machine assignment if there are three machines and the orders from Table 2 are assigned to machines randomly: (7,2,1) — (3,1,2) — (4,1,1) — (1,1,1) — (2,1,1) — (6,2,1) — (7,3,3) — (6,1,3) — (5,1,2) — (7,1,2)

4.2 Fitness

The fitness of each sequence will be calculated and determined based on CD and makespan. The makespan of a schedule is the maximum of completion time of all machines. The workload is balanced by minimizing makespan. The lower bound of the makespan in a parallel machine problem can be determined by the average machine processing time ($\frac{\sum P_{ij}}{o}$). One of the methods to reduce makespan is the longest processing time (LPT) rule. For o parallel machines and L jobs, the jobs are sorted in descending order of processing times. For each job in order, it is assigned to the first available machine in time (break ties arbitrarily). If $L \leq o$, then the optimal makespan following the LPT rule is just the longest processing time among the jobs. Otherwise, the LPT will approximate the following ratio: $C_{\max(\mathrm{LPT})}/C_{\max(\mathrm{OPT})} \leq (4/3 - -1/3o)$ [1].

The CD is calculated by the difference between the earliest and the latest completion time in one order. The CD

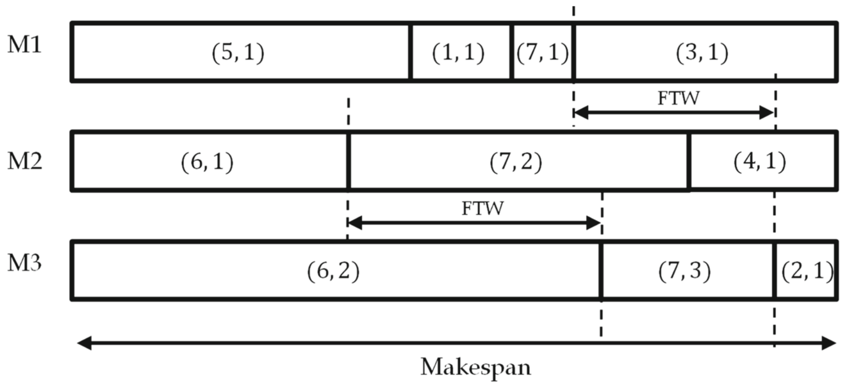


Fig. 3 Example of scheduling jobs to minimize the makespan

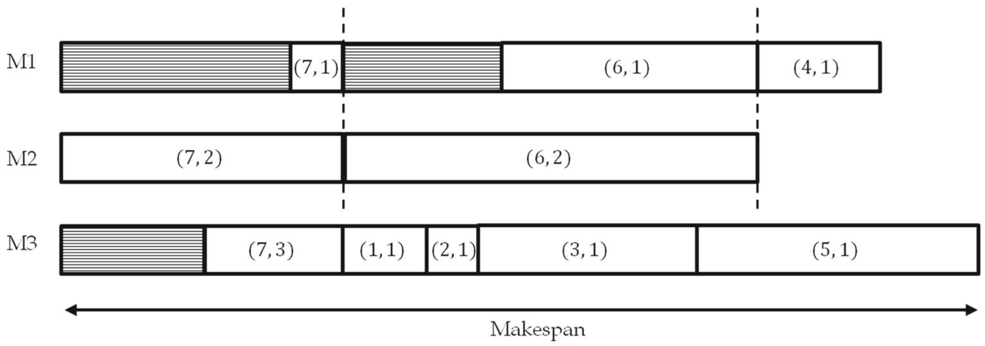


Fig. 4 Example of scheduling jobs to minimize the CD

for the entire schedule is the sum of all CDs for all orders, as shown in Eq. 1. For example, the makespan and the CD will be calculated based on Table 3. All jobs will be processed at the earliest possible time so that there is no idle time between jobs on a machine. Table 4 shows the total processing time for each machine given the job assignments from Table 3. The makespan for the schedule is the maximum processing time of the three machines, 32 s. As shown in Table 5, the CD is the difference between the earliest and the latest completion time of a job. The CD of the schedule is 32 s and the schedule is depicted in Fig. 2.

Figure 3 indicates a schedule that minimizes makespan without considering the CD. There machines have the equal balance workload, but the CD is generated for order 6 and 7. Figure 4 shows a schedule that minimizes the CD without considering the makespan. Machine 1 and 3 have idle time to achieve zero CD, but the makespan is increased. In this case, there is a trade-off between minimizing makespan and minimizing the CD. The makespan measures the dispensing throughput, and the CD measures the auto-collation efficiency. Therefore, multi-objective approaches are applied to minimize the makespan and CD.

4.2.1 VEGA

The VEGA is the first implemented to solve this multi-objective problem. It approximates the Pareto front by dividing the population into equally sized sub-populations. Each subpopulation optimizes a different objective and assigns a fitness value to each member in the subpopulation. In VEGA, each subpopulation is only evaluated against its objective z. The roulette-wheel selection method is adopted to select parents. The fitness is associated with the probability of selection. Solutions having a better z value are more likely to be chosen and placed into the subpopulation. Therefore, they are more likely to be placed into the subpopulation. The population in the next generation are generated based on the chromosome crossover and mutation. The main drawback to using this approach is that Pareto optimal solutions may be discarded because they are evaluated with respect to only one objective in each subpopulation. Solution population tends to be crowded around the extreme of each objective [15, 26].

4.2.2 MOGA

In the MOGA proposed by Fonseca and Flemming, the ranking is done by sorting each solution into a front [7], as shown in Algorithm 1. The equation used to determine ranking is $r(x,t) = 1 + nq(x,t)$, where $nq(x,t)$ is the number of solutions that dominate solution x at generation t. Each solution in the population is considered, and then compared to every other solution. A solution is dominated if the other solution has either a better CD or a better $C_{\max}$.

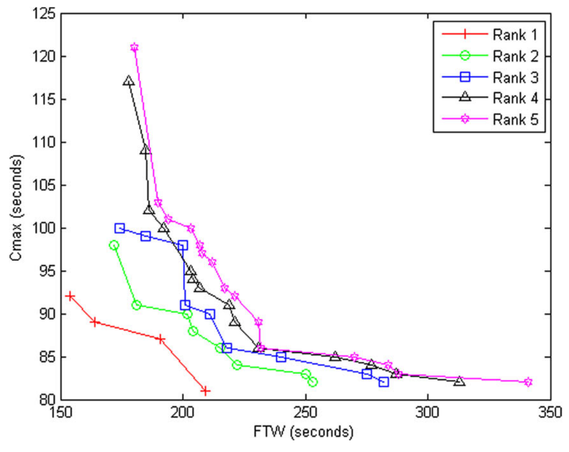
Algorithm 1 Sort algorithm in MOGA

```
1: Sort the population ($P$) in decreasing fitness based on one objective
2: If $|P| = 1$ return $P$ as $Front(P)$. Else, $T = Front(P_{(1)} - P_{(|P/2|)})$ and $B = Front(P_{(|P|/2+1)} - P_{(P)})$. If the $i$th non-dominated solution $B$ is not dominated by any non-dominated solution of $T$, create a merged set $M = T \cup i$. Return $M$ as output of $Front(P)$
```

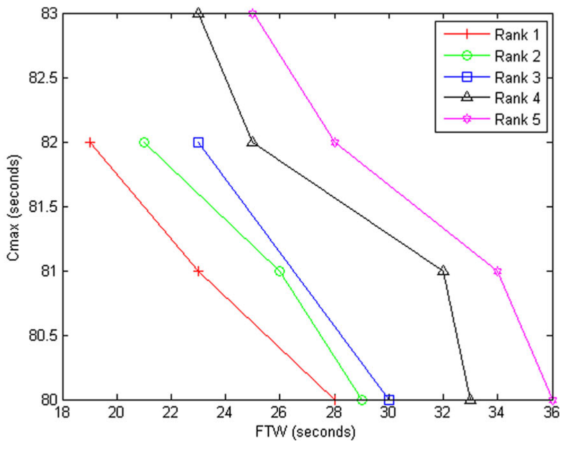
In MOGA, a fitness penalty is applied for solutions that are in the same ranking. The Euclidean distance between

Table 6 Parameters used in multi-objective genetic algorithms

Population size ($L = 12, 24$)	$10 \cdot L$
Population size ($L = 120$)	2000
Crossover method	Single point
Crossover rate	100 %
Mutation method	2-opt swap
Mutation rate	20 %
Parent selection method	Roulette wheel



(a) Example ranking of an intial population up to rank 5.



(b) Example ranking of solution population.

Fig. 5 Example ranking before and after the VEGA process in the 24–12 problem

every solution pair is calculated, and a niche count is determined. The niche is a defined neighborhood of solutions, and the solutions in the neighborhood contribute to each others niche count. A higher niche count indicates a more densely populated area. Their fitness is scaled down, which reduces the probability that they are selected as a parent for crossover. Niching limits continuously explore of areas in the objective function space. It is difficult to determine what the niche parameter should be. Fonseca and Fleming [7] describe a method to approximate the niche size in the objective value domain. Given a population of size N, σ_{share} can be estimated by solving:

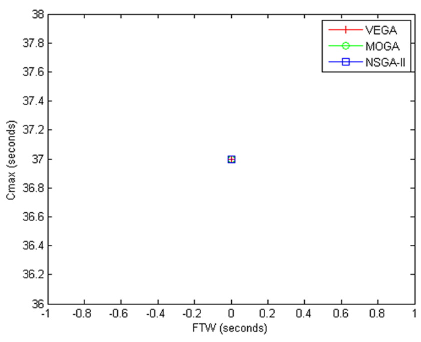
$$N\sigma_{\text{share}}^{q-1} - \frac{\prod_{z=1}^{q}(M_z - m_z + \sigma_{\text{share}}) - \prod_{z=1}^{q}(M_z - m_z)}{\sigma_{\text{share}}} = 0 \quad (11)$$

where q is the number of objectives, M_z and m_z are the maximum and minimum values of objective z for the current generation, respectively. This formula is applied at every generation.

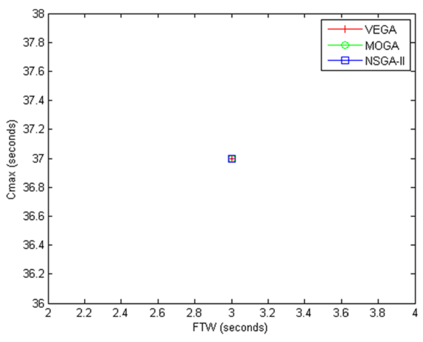
In this study, MOGA uses stochastic universal selection to determine the mating pool. This selection method is similar to roulette-wheel selection. Instead of re-rolling a random number for each selection, all selections are made at one time based on a single random number. For N solutions to be selected, the selections will be made $1/N$ distance apart. The random number to be generated will be in the range of $[0, 1/N]$. After the mating population is selected, a binary tournament is used to determine parents for crossover. The stochastic universal selection keeps consistency in the mating pool.

4.2.3 NSGA-II

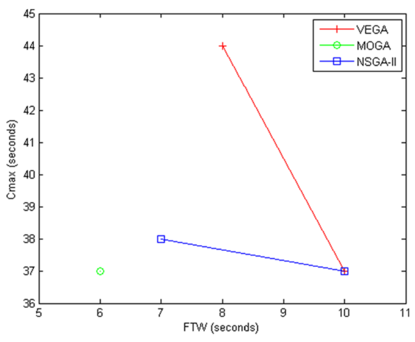
An fast non-dominated sort algorithm is used in the NSGA-II, as shown in Algorithm 2. A temporary set P' is initialized and a first solution is entered into the set. Every solution x is compared with all members in the set P' one at a time. If x dominates any member of P', the member is removed from P'. If x is dominated by any member in P', the solution is ignored. If x is non-dominated to all the remaining members



(a) Frontier of the 12-2 problem.



(b) Frontier of the 12-6 problem.



(c) Frontier of the 12-10 problem.

Fig. 6 Frontier plots for the 12 problem

Table 7 Summary of experimental results (CD and C_{max}) with $L = \{12, 24\}$ and $o = 3$

		LPT		VEGA		MOGA		NSGA-II	
L	p	CD	$C_{\max}$	μ_{CD}	$\mu_{C_{\max}}$	μ_{CD}	$\mu_{C_{\max}}$	μ_{CD}	$\mu_{C_{\max}}$
12	2	22	38	0	37.2	0	37	0	37
12	6	45	38	3.2	39	3.4	37	3	37
12	10	62	38	10.8	39.4	9.6	37.8	9.2	37.6
24	4	43	80	0	80.6	0	80	0	80
24	12	160	80	8.6	85	11.6	80.2	7.6	80.2
24	20	216	80	25.8	82.8	24.2	82.4	21	81.4

All units are in seconds

in P', then it enters into P'. The first iteration gives the non-dominated solutions of the current population and gives them a rank of 1. All solutions in P' are then ignored and to rank the rest of the population, the procedure is repeated, with each iteration producing a next non-dominated front and then discounting them for the remaining solutions.

Algorithm 2 Sort algorithm in NSGA-II

```
1: Initialize population $P$. $r$=1.
2: Sort solutions in a decreasing order of one objective function and create sorted lists.
3: Initialize a set $P'$ and add the first solution in the sorted list to $P'$. Remove the solution from $P$.
4: For every solution $x$ (other than the first solution), compare solution $x$ to every solution in $P'$
    (a) If any solution in $P'$ dominates $x$, move on to the next solution
    (b) If $x$ is non-dominated to all solutions in $P'$, add solution $x$ to $P'$. Remove the solution from $P$.
    (c) If $P'$ becomes empty, immediately add the solution $x$ to $P'$. Remove the solution from $P$.
5: Return $P'$ as the non-dominated set and Pareto front, all solutions in $P'$. $r$=$r$+1
6: Repeat Steps 3 - 5 until $P$ is empty.
```

In NSGA-II, a crowding distance is implemented to spread solutions more evenly along a frontier. After ranking the population, the crowding distance of each solutions is set as ∞, and then it is calculated by Eq. 12.

$$cd_z(x_{[i,z]}) = \frac{y_z(x_{[i+1,z]}) - y_z(x_{[i-1,z]})}{M_z - m_z} \qquad (12)$$

where i is the ith solution in the rank, z is the objective index, M_z is the maximum value found for objective z, m_z is the minimum value found for objective z, and $y_z(x)$ is the fitness value for solution x in objective z. The crowding distance for a solution, x, is then $cd(x) = \sum_z cd_z(x)$. The advantage of the crowding distance measurement is that it does not require a user-defined parameter such as σ_{share} in Eq. 11.

4.3 Crossover

The single point crossover method is applied to reproduce children. The crossover operator randomly select a cut point in a parent, dividing the job sequence into two. The portion of jobs before the cut will remain the same on the child. The portion of jobs after the cut will be taken from parent 2 in the same sequence the remaining jobs appear. The step-by-step procedure is presented in Algorithm 3:

Algorithm 3 Three-tuples chromosome crossover

```
1: Randomly select two parents $Pa_1$ and $Pa_2$ from population set $P$.
2: Randomly determine a cut point for $Pa_1$ and for $Pa_2$
3: Begin crossover
    3.1 Child 1 takes the sequence of jobs and machines from $Pa_1$ before the cut.
    3.2 Identify the jobs in $Pa_2$ that are not yet in Child 1. This set of jobs in the sequence they appear in $Pa_2$ is appended to the end of Child 1
    3.3 Child 2 takes the sequence of jobs and machines from $Pa_2$ before the cut.
    3.4 Identify the jobs in $Pa_1$ that are not yet in Child 2. This set of jobs in the sequence they appear in $Pa_1$ is appended to the end of Child 2
4: Evaluate the fitness of the newly created solutions
```

4.4 Mutation

Mutation is a procedure to improve diversity of the population by simply replacing part(s) of the population with random individual(s). Mutation occurs with a low probability because it is not desirable to change the entire population frequently. A common form of mutation is swap mutation, where two genes within a chromosome are randomly selected and their positions are swapped, as shown in Algorithm 4. This is equivalent to having two jobs selected and then having their positions in the sequence switched.

Table 8 Summary of experimental results (σ and computational time) with $L = \{12, 24\}$ and $o = 3$

		VEGA			MOGA			NSGA-II		
L	p	σ_{CD}	$\sigma_{C_{\max}}$	CT	σ_{CD}	$\sigma_{C_{\max}}$	CT	σ_{CD}	$\sigma_{C_{\max}}$	CT
12	2	0	0.45	44.72	0	0	797.54	0	0	752.73
12	6	0.45	3.94	53.00	0.55	0	1297.58	0	0	907.25
12	10	2.17	3.36	47.93	2.61	0.84	1025.42	2.49	0.55	969.89
24	4	0	1.34	53.77	0	0	977.33	0	0	950.50
24	12	5.13	4.06	68.73	5.51	0.45	1690.22	2.07	0.45	1294.10
24	20	3.90	2.95	83.59	6.14	2.88	1735.26	1.87	1.14	1329.48

All units are in seconds

A pair wise exchange is applied as the mutation operator. Jobs are randomly selected and swapped from within a single chromosome, which allow the possibility of the order of jobs being processed to be changed while the machine assignment stays the same.

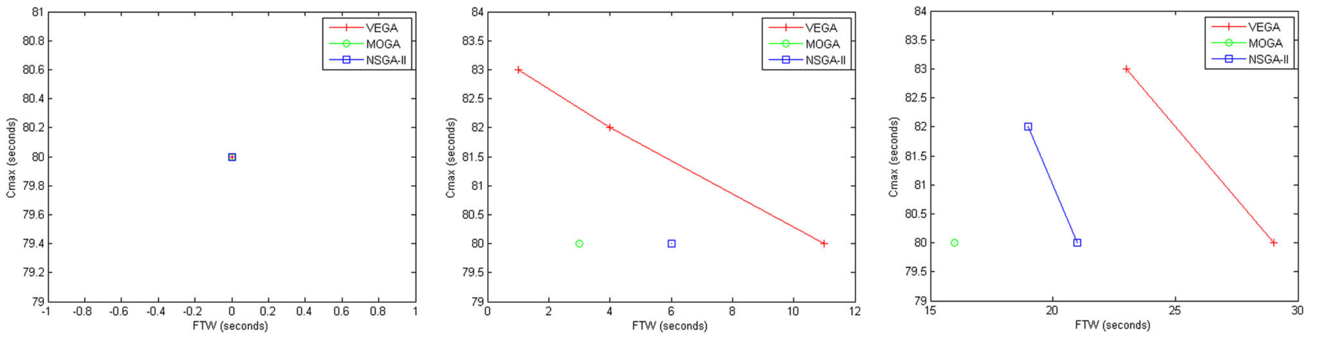
Algorithm 4 Mutation

```
1: For each child created as a result of crossover, determine if mutation will occur. If mutation does occur, randomly select two jobs
2: Swap the positions of the two jobs. The swapped jobs are assigned to the other's machine
3: Re-evaluate the fitness of the solution due to the swapping of jobs
```

5 Experimental results and analysis

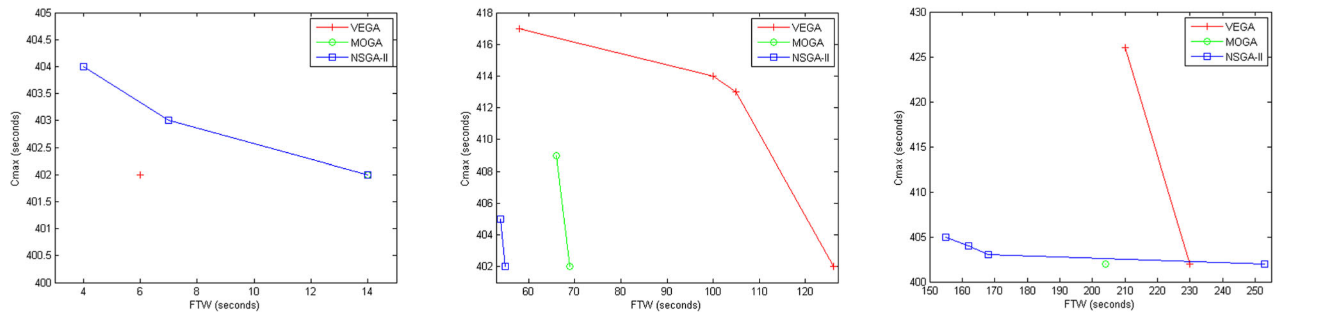
Three multi-objective genetic algorithms with a three-tuples chromosome design, including VEGA, MOGA, and NSGA-II, are implemented to minimize the CD and makespan. Experiments are performed with different problem sizes to study their impact on the genetic algorithms' performances. The genetic algorithms are evaluated by comparing their results to each other and also to the results obtained by the LPT heuristic, which will be used as a baseline solution. The LPT is a commonly used scheduling heuristic when trying to minimize makespan. A set of orders and jobs are randomly initialized and used for the whole experiments. There are three cases of different job numbers, $L = \{12, 24, 120\}$. In each case, the jobs are grouped by orders and a processing time for each is assigned. It is assumed that the number of jobs in an order will not exceed the number of machines; for example, if there are three machines, then there can only be a maximum of three jobs in an order. The different number of jobs are used to study the effect of job size on the solution quality of the genetic algorithms.

The heuristics' parameters are fine tuned based on taguchi method for each case, which is summarized in Table 6. Three genetic algorithms (VEGA, MOGA, and NSGA-II) are run and terminated when no progress to find



(a) Frontier of the 24-4 problem. (b) Frontier of the 24-12 problem. (c) Frontier of the 24-20 problem.

Fig. 7 Frontier plots for the 24 jobs problem



(a) Frontier of the 120-20 problem. (b) Frontier of the 120-60 problem. (c) Frontier of the 120-100 problem.

Fig. 8 Frontier plots for different number of multi-medication orders

better ranked solutions is made within $5 \cdot p \cdot m$ generations. All data sets are shaped for a three machine situation. The processing time for each job follows the discrete uniform distribution from 1 to 20 s. After fixing the number of orders having a single job, half of the remaining jobs belong to 2-job orders and the other half belong to 3-job orders. For example in the 120–100 job problem, 100 of the jobs are a part of a multi-medication order. Fifty of the jobs will be a part of 2-job orders, and the other 50 will be a part of 3-job orders. However, since 50 is not a multiple of 3, more orders are created to be 2-job orders until the remainder of jobs is a multiple of 3. In this case, 52 jobs are used to create 2-job orders, and the remaining 48 jobs are used to create 3-job orders. Initial randomized population size for each GA is set to 2000 with the exception of MOGA. Preliminary tests show that with an initial population size of 2000, MOGA is frequently unable to complete a run before reaching the time limit. It is desirable to keep a large mutation rate so that different assignments can be explored. The main performance metrics are the non-dominated CDs and $C_{\max}$ in the Pareto frontier. Additional experiments are conducted to analyze the impact of the number of multi-medication jobs, p, and the number of machines, o.The data sets are obtained from [18].

The LPT method of scheduling is used to provide a solution that does not take into account the CD, but a heuristic to minimize makespan. The lower bound is determined by $\left|\frac{\sum P_{ij}}{o}\right|$. If it matches the makespan from the experimental results, then the makespan obtained is optimal. If the CD is 0, then it is the optimal CD. The mean and standard deviation of the CD(μ_{CD}, σ_{CD}), makespan ($\mu_{C_{max}}, , \sigma_{C_{max}}$), and computational time (CT) are measured from 10 independent runs of each case. $C_{\max}$ obtained is considered optimal if the result is equal to the sum of the processing times for all jobs divided by the number of machines, $\sum_{i=1}^{m}\sum_{j=1}^{n} p_{ij}/o$, and then rounded up to the nearest whole number. All performance metrics are measured in seconds. All genetic algorithms are developed in Matlab R2013a, and the experiments are conducted on a PC with an Intel Core i7-3770 CPU @ 3.4 GHz and 16 GB RAM.

5.1 Analysis of small job-size problems

The GAs are applied to smaller job size problems with $L = 12, 24$ on three parallel machines ($o = 3$). The experimental results are shown in Tables 7 and 8 and Figs. 6 and 7.

The GAs save the CD by 82.58−−100 %, but increase the makespan by 0−−6.25 %. The VEGA is relatively simple to implement and run the fastest among all three GAs, which is terminated in less than 2 min on average. While VEGA can provide competitive results in the 12 − −2 and 12 − −6 problems, it is quickly surpassed by MOGA and NSGA-II as the number of multi-medication jobs or total number of jobs is increased. It is able to obtain less CD values than LPT, but it has the only advantage in computational time compared to other GAs. In terms of stability of solutions, VEGA has the highest standard deviation in both CD and $C_{\max}$ compared to the other GAs. MOGA among all

Table 9 Summary of experimental results (CD and $C_{\max}$) with $L = 120$, $o = 3$, and different amounts of multi-medication orders, p

	LPT		VEGA		MOGA		NSGA-II	
p	CD	$C_{\max}$	μ_{CD}	$\mu_{C_{\max}}$	μ_{CD}	$\mu_{C_{\max}}$	μ_{CD}	$\mu_{C_{\max}}$
20	1735	402	12	416	21.4	403.8	9.8	406.6
60	3721	402	101.2	444.2	79.8	403.8	58	404.6
100	6585	402	258.8	441.2	238.6	402.4	180	405.8

All units are in seconds

Table 10 Summary of experimental results (σ and computational time) with $L = 120$, $o = 3$, and different amounts of multi-medication orders, p

	VEGA			MOGA			NSGA-II		
p	σ_{CD}	$\sigma_{C_{\max}}$	CT	σ_{CD}	$\sigma_{C_{\max}}$	CT	σ_{CD}	$\sigma_{C_{\max}}$	CT
20	5.34	20.93	248.06	7.27	2.68	2878.12	3.63	3.21	2539.74
60	27.71	26.83	428.18	15.01	2.95	4424.72	4.74	2.30	4082.14
100	52.49	26.68	544.60	25.56	0.89	7205.32	21.55	3.03	5926.66

All units are in seconds

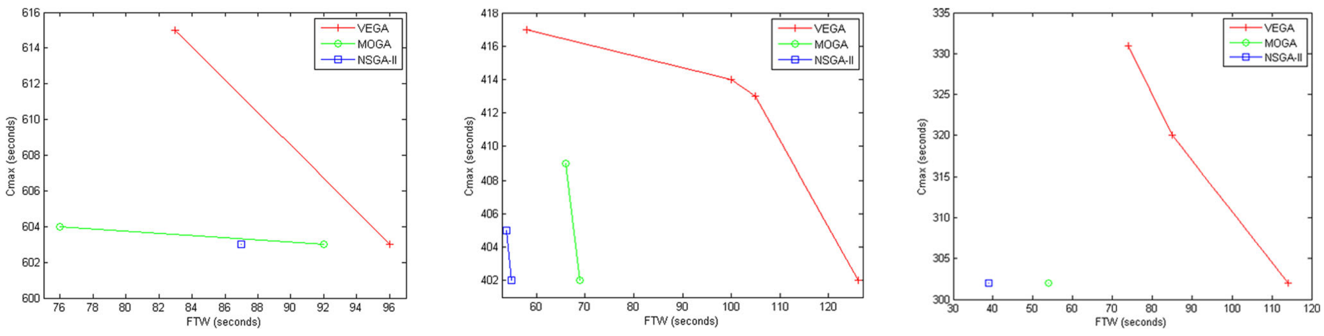
three GAs takes the longest time to be terminated for every case. Despite a small population size, MOGA is able to produce better results than VEGA, but takes a longer time to be terminated. The transition from 12 jobs to 24 jobs, however, shows that the MOGA struggles to produce better CD values. The NSGA-II is able to run the $12-2$ and $12-6$ jobs problem with optimal results. There is no variation in the solutions for any runs. In the $12-10$ job problem, the NSGA-II had more difficulty in producing a more stable CD. Overall, NSGA-II is able to obtain the least $C_{\max}$ with the least CD and variance.

For the 12 job problems, the optimal $C_{\max} = 37$. In the $12-2$, all GAs obtains the same result in CD and the optimal $C_{\max}$ with the exception of VEGA, which has a $C_{\max}$ that is 0.54 % larger. In the $12-6$ problem, MOGA and NSGA-II obtain the optimal $C_{\max}$, while VEGA has a 5.41 % increased $C_{\max}$. NSGA-II obtains an average CD = 3 with MOGA having a 13.33 % increase and VEGA having a 6.67 % increase in CD. In the $12-10$ problem, NSGA-II obtains an average CD of 9.2 with MOGA having a 4.35 % increase and VEGA having a 17.39 % increase in CD. NSGA-II obtains a 1.62 % increased $C_{\max}$ over the optimal, while VEGA's $C_{\max}$ is larger by 6.49 % and MOGA's $C_{\max}$ is larger by 2.16 %. For the 24 job problems, the optimal $C_{\max} = 80$. In the $24-4$ job problem, all GAs obtain the same result in CD and the optimal $C_{\max}$ with the exception of VEGA, where the $C_{\max}$ is 0.75 % larger. In the 24-12 problem, NSGA-II and MOGA obtain an average $C_{\max}$ that is 0.25 % larger than optimal, while VEGA has a $C_{\max}$ that is 6.25 % larger. NSGA-II obtains an average CD = 7.6 with MOGA having a 52.63 % larger average CD and VEGA having a 13.16 % larger average CD. In the 12-10 problem, NSGA-II obtains an average CD of 21 with MOGA having a 15.24 % larger average CD and VEGA having a 22.86 % larger average CD. NSGA-II, MOGA, and VEGA have an average $C_{\max}$ that is 1.75, 3.00, and 3.50 % larger than optimal, respectively.

When the percentage of multi-medication jobs is low and the total job number is low, as in the 12-2 and the 24-4 problems, the GAs are able to achieve a CD and $C_{\max}$ that is optimal in both respects. The result is $CD = 0$ and $C_{\max} = 37$ in the 12 job problem and $CD = 0$ and $C_{\max} = 80$ in the 24 job problem. There is no variation in the runs, all results come to the same CD and $C_{\max}$ value with the exception of VEGA having a slight variation in the $C_{\max}$.

Figure 5 shows an example of a population before and after VEGA processes. Figure 5a indicates that the initial randomly generated population are categorized into ranks 1 to 5 with the non-dominated frontier. The points in the same frontiers are connected with lines in the same color. After one iteration run of VEGA, the ranking graph is shown in Fig. 5b. In subsequent graphs, the non-dominated frontiers are generated by selecting the non-dominated frontier (rank 1) in each experiment.

The non-dominated frontiers generated from GAs show in Figs. 6 and 7. All three GAs are able to find the point that is non-dominated compared to all other points (CD = 3,



(a) 120-2 machine problem. (b) 120-3 machine problem. (c) 120-4 machine problem.

Fig. 9 Frontier plots for the 120-o machine job problem

Table 11 Summary of experimental results (CD and $C_{\max}$) with $L = 120$ and $p = 60$ on different numbers of machines, o

	LPT		VEGA		MOGA		NSGA-II	
o	CD	$C_{\max}$	μ_{CD}	$\mu_{C_{\max}}$	μ_{CD}	$\mu_{C_{\max}}$	μ_{CD}	$\mu_{C_{\max}}$
2	6,633	603	111.6	605.4	101.8	603.8	101.4	603.2
3	3,721	402	101.2	444.2	79.8	403.8	58	404.6
4	3,240	302	81.2	333.4	68.8	302.4	52.4	303.2

All units are in seconds

$C_{\max}$=37), as shown in Figs. 6a and 6b. In the 12–10 job problem, Fig. 6c indicates that MOGA is in one instance able to find a better CD value than VEGA or NSGA-II. All three GAs are able to reach the optimal $C_{\max}$ of 37. Figure 6a shows that all GAs are able to find the best non-dominated point with an optimal $C_{\max}$.

5.2 Analysis of different amounts of multi-medication orders

In the CFP, multiple medications orders influence the CD. To study the impact of multi-medication orders on the CD, the job size and number of machines are fixed. The varying factor is the number of jobs belonging to an order. For these scenarios, the job size is $n = 120$, with p being the amount of jobs that are part of a multi-medication order. The $C_{\max}$ for this problem is 402. Due to the larger size of the problem, the computational time limit is set as 2 h. A summary of the experimental results is shown in Tables 9 and 10 and Fig. 8.

The GAs save the CD by 96.07——99.44 %, but increase the makespan by 3.48——10.50 %. The $C_{\max}$ found by LPT is the optimal makespan of 402. In these experiments, NSGA-II is able to obtain the best CD out of all three GAs. NSGA-II has the best average CD and $C_{\max}$ for all three cases. In the 120–20 problem, MOGA has the worst CD,which is 118.37 % larger. In the 120–60 and 120–100 problems, VEGA has the worst average CD, which is 74.48 and 43.78 % larger than NSGA-II's average CD, respectively. MOGA is able to obtain the best average $C_{\max}$ in all cases and VEGA the worst. VEGA's $C_{\max}$ is larger than the optimal by 3.48, 10.50, and 9.75 % for the 120–20, 120–60, and 120–100 problems, respectively. MOGA's $C_{\max}$ is larger than the optimal by 0.45, 0.45, and 0.01 % for the 120–20, 120–60, and 120–100 problems, respectively. NSGA-II's $C_{\max}$ is larger than the optimal by 1.14, 0.65, and 0.95 % for the 120–20, 120–60, and 120–100 problems, respectively.

It can also be seen that having a larger amount of multi-medication orders is correlated with a larger CD. There is an increase in CD from 20 to 60 multi-medication jobs and a further increase from 60 to 100 multi-medication jobs. Computational time also increases as the number of multi-medication jobs increases. When standard deviation increases across the board as the number of multi-medication orders increases, this indicates that it becomes more difficult to obtain better solutions. From 20 to 100 jobs, VEGA's computational time climbs from 248.058 to 544.596 s, the fastest of the three GAs, while MOGA's computational time increases from 2878.12 to 7205.32 s. MOGA reaches a solution in the longest amount of time with a time difference of 72 min between 20 and 100 jobs, compared to VEGA which had a time difference of 5 min.

In Fig. 8, NSGA-II has the least CD solution in the 120–60 and 120–100 job problems. VEGA appears to have a good solution of performance in Fig. 8a on the low amount of multi-medication jobs. However, the best frontier of VEGA falls off when the number of multi-medication jobs increases. In this problem, all three GAs are able to obtain a point that had an optimal $C_{\max}$.

5.3 Analysis of different amounts of parallel machines

CFP operation involves many automatic pharmaceutical dispensing machines. There needs to be enough machines working in parallel in order to satisfy the prescription demands. The sensitivity of the CD and $C_{\max}$ is analyzed with respect to the amount of machines available. In this section, the number of jobs has been fixed at 120 with 60 of the jobs being part of a multi-medication order. The amount of machines has been varied from 2 to 4. The optimal $C_{\max}$ is 603, 402, and 302 for 2, 3, and 4 machines, respectively. The amount of multi-medication jobs in one order remains at 3, and the computational time limit is set as 2 h. A summary of the experimental results is shown in Tables 11 and 12 and Fig. 9.

The GAs save the CD by 97.28——98.47 %, but increase the makespan by 0.40——10.50 %. As the number of machines increases from 2 to 4, the CDs found by all three GAs decrease and there are more opportunities for the multi-order jobs to be scheduled such that there is no CD for that order. VEGA's CD increases from 111.6 to 81.2, MOGA's CD increases from 101.8 to 68.8, and NSGA-II's CD increases from 101.4 to 52.4. The MOGA and NSGA-II also obtain near optimal values of $C_{\max}$ with 603, 402, and 302 being the optimal $C_{\max}$ at 2, 3, and 4 machines, respectively. NSGA-II has the best average CD in all cases. For the 2 machine problem, VEGA has a larger average CD by 10.06 % and MOGA had a larger average CD by 0.39 %. For the 3 machine problem, the CD obtained by VEGA is 74.48 % larger in average, and the CD obtained by MOGA is 37.59 % larger in average. For the 4 machine problem, the CD obtained by VEGA is 54.96 % larger in average, and the CD obtained by MOGA is 31.30 % larger in average. VEGA has the worst performance out of all three GAs in terms of CD and $C_{\max}$, while MOGA has the best with the exception of the 2 machine case where NSGA-II had a better average $C_{\max}$. VEGA's C_{max} is larger than the optimal by 0.40, 10.50, and 10.40 % for the 2 machine, 3 machine, and 4 machine problems, respectively. MOGA's $C_{\max}$ is larger than the optimal by 0.13, 0.45, and 0.13 % for the 2 machine, 3 machine, and 4 machine problems, respectively. NSGA-II's $C_{\max}$ is larger than the optimal by 0.03, 0.65, and 0.40 % for the 2 machine, 3 machine, and 4 machine problems, respectively.

Figure 9 indicates the non-dominated frontiers for a varying number of parallel machine as scenario. In the 2 machine case, both MOGA and VEGA have a point with a better CD. However, all GAs can reach the same level of $C_{\max}$ in each case. The majority of VEGA's frontier is dominated by either MOGA's or NSGA-II's frontier in the 3 and 4 machine cases. In those problems, NSGA-II has the best frontier.

6 Conclusions and future work

In this paper, the CD problem is studied as the multi-objective optimization problem in MOPAD systems, which provide a set of solutions for the management decision. The makespan and CD are conflicting objectives. When the CD is minimized by adding the idle time, the makespan increases. Three multi-objective genetic algorithms (VEGA, MOGA, and NSGA-II) are applied to minimize both CD and makespan. Various job sizes, number of machines, and number of multi-medication jobs are considered. In the small job size and low multi-medication order count problems, all three GAs are able to find the optimal CD and makespan ($C_{\max}$). All outperform the common sequencing rule of longest processing time (LPT) heuristic in CD for all problem cases. In terms of makespan, LPT is able to achieve optimal or near optimal results. Based on the experimental results, three objective genetic algorithms save the CD by 95.67 % on average, but only increase the makespan by 5.62 %. In all instances, VEGA is able to reach a solution the fastest, but it has the worst performance of the three GAs. MOGA takes the longest amount of time to reach a solution but comes in second compared to VEGA and NSGA-II. NSGA-II provides the best frontier at the larger job sizes and larger number of multi-medication jobs. NSGA-II also has the least amount of variation in solutions. It becomes difficult for multi-objective GAs to find optimal solutions for the CD in large job size problems. The frontiers generated for each GA are often small, usually $1--2$ points, which indicates that the makespan can be minimized but the optimal CD is difficult to obtain in many cases. If a solution can achieve the optimal makespan and CD, the frontier will be only a single point, which is shown in the $12--2$ and $24--4$ problems. At the larger problem sizes, there is more variability in the solutions found and the frontier is more developed when the CD is much more difficult to optimize. Overall, NSGA-II is the most stable of the GAs and also has the best performance, especially at larger job sizes or when there are a larger number of multi-medication jobs. In small size problems, VEGA or MOGA has more chance to find a better CD in a single run, but their σ is much higher. The computational time for each GA is on the scale of minutes to hours. In the $12--10$ job problem, MOGA takes the longest computational time to process with an average time of 1735.26 s and for the 120-3o-60 job problem took on average 4424.72 s. VEGA takes the least amount of time with the 1210 job problem taking on average 83.594 s and 120-3o-60 job problem taking on average 544.596 s. NSGA-II falls in between MOGA and VEGA, but closer to MOGA's computational time. This research can be applied to MOPAD system of CFPs. It also can be applied to other distribution or manufacturing systems which have collation processes.

However, many researches can be conducted to advance our investigations and findings. One of the important assumptions is that all auto-dispensing machines are identical, such that each medication job would be processed on

Table 12 Summary of experimental results (σ and computational time) with $L = 120$ and $p = 60$ on different numbers of machines, o

	VEGA			MOGA			NSGA-II		
o	σ_{CD}	$\sigma_{C_{max}}$	CT	σ_{CD}	$\sigma_{C_{max}}$	CT	σ_{CD}	$\sigma_{C_{max}}$	CT
2	17.84	5.37	252.83	20.43	0.84	5537.70	9.97	0.45	3524.02
3	27.71	26.83	428.18	15.01	2.95	4424.72	4.74	2.30	4082.14
4	9.94	8.56	428.26	8.93	0.548	3117.06	14.15	1.30	5319.92

All units are in seconds

any machine and the processing time is the same. However, in the real MOPAD system, the auto-dispensing machines contain the specified medications. To increase the efficiency of the MOPAD system, an appropriate medication assignment and allocation can increase the dispensing efficiency and productivity. The demand of medications and the medications combinations can be analyzed and utilized to optimize the medication assignment and allocation to dispensing machines. When the medication of auto-dispensers are specified, the CD problem will become a parallel dedicated machines problem. Moreover, medication dispensing times vary depending on dispensing machine types and medication characteristics (e.g., shape, weight, capsule types, etc.). A different distribution may have been more appropriate and can improve the scheduling quality in MOPAD systems . Furthermore, it is assumed that the number of multi-medication jobs in an order would not exceed the number of parallel auto-dispensing machines. The order separation rules may be studied on how to split orders such that the CD and makespan can be minimized when the multi-medication jobs in an order exceed the number of machines.

Acknowledgments This study was supported by the Watson Institute of Systems Excellence (WISE) at Binghamton University and by Innovation Associates. The authors also thank several colleagues who have inspired and provided valuable comments to improve this study: Boyer, T. and Lashier, A. from Innovation Associates and Poch, L. from WISE.

References

1. Atallah MJ (1998) Algorithms and theory of computation handbook. CRC press, Boca Raton
2. Bepko RJ Jr., Moore JR, Coleman JR (2009) Implementation of a pharmacy automation system (robotics) to ensure medication safety at norwalk hospital. Qual Manag Health Care 18(2):103–114
3. Bhattacharya R, Bandyopadhyay S (2010) Solving conflicting bi-objective facility location problem by nsga ii evolutionary algorithm. Int J Adv Manuf Technol 51(1–4):397–414
4. Cochran JK, Horng S-M, Fowler JW (2003) A multi-population genetic algorithm to solve multi-objective scheduling problems for parallel machines. Comput Oper Res 30(7):1087–1102
5. Coello CAC (2000) An updated survey of ga-based multiobjective optimization techniques. Association for Computing Machinary Computing Surveys 32(2):109–143
6. Deb K, Pratap A, Agarwal S, Meyarivan T (2002) A fast and elitist multiobjective genetic algorithm: Nsga-ii. IEEE Trans Evol Comput 6(2):182–197
7. Fonseca CM, Fleming PJ (1993) Genetic algorithms for multiobjective optimization: Formulation discussion and generalization. In: Genetic Algorithms: Proceedings of the Fifth International Conference, vol 93, pp 416–423, San Mateo
8. Gholami M, Azizi M (2014) Constrained grinding optimization for time, cost, and surface roughness using nsga-ii. Int J Adv Manuf Technol 73(5-8):981–988
9. Goldberg DE (1989) Genetic algorithms in search, optimization, and machine learning, Addison-wesley Reading Menlo Park
10. Greb E, Rios M (2009) Robots: The next phase in pharmaceutical automation. Pharm Technol 33(9):38–44
11. Guo ZX, Wong WK, Leung SYS, Fan JT, Chan SF (2008) Genetic optimization of order scheduling with multiple uncertainties. Expert Syst Appl 35(4):1788–1801
12. Guo ZX, Wong WK, Li Z, Ren P (2013) Modeling and pareto optimization of multi-objective order scheduling problems in production planning. Comput Ind Eng 64(4):972–986
13. Jenkins A, Eckel SF (2012) Analyzing methods for improved management of workflow in an outpatient pharmacy setting. Am J Health Syst Pharm 69(11):966–971
14. Jones DF, Keyvan Mirrazavi S, Mehrdad Tamiz (2002) Multi-objective meta-heuristics: an overview of the current state-of-the-art. Eur J Oper Res 137(1):1–9
15. Konak A, Coit DW, Smith AE, Multi-objective optimization using genetic algorithms: a tutorial (2006) Reliab Eng Syst Saf 91(9):992–1007
16. Li D, Won Yoon S (2012) Minimizing fill-time window in central fill pharmacy. In: Industrial and systems engineering research conference (ISERC), Orlando
17. Li D, Won Yoon S (2012) Simulation based manova analysis of pharmaceutical automation system in central fill pharmacy. In: International conference on industrial engineering and engineering management (IEEM), Hong Kong, China
18. Li D, Won Yoon S (2014) A novel fll time window minimization problem and adaptive parallel tabu search algorithm in mail-order pharmacy automation system. Int J Prod Res. In press
19. Li Q, Gong J, Fung RYK, Tang J (2012) Multi-objective optimal cross-training configuration models for an assembly cell using nondominated sorting genetic algorithm-ii. Int J Comput Integr Manuf 25(11):981–995
20. Liu C-H (2014) Approximate trade-off between minimisation of total weighted tardiness and minimisation of carbon dioxide (co2) emissions in bi-criteria batch scheduling problem. Int J Comput Integr Manuf 27(8):759–771
21. Liu X, Chao Y, Zhao X, Wang W, Ma Y (2008) Design and application for automated medicine depositing and dispensing system of pharmacy. In: 2008 International conference on computer science and information technology, ICCSIT '08, pp 332–336
22. Mahfoud SW (1992) Crowding and preselection revisited. Parallel problem solving from nature 2:22–36
23. Ming F, Yi G, Chao Y (2010) Research on the parameter optimization model for the automated storage and retrieval system of pharmacy. In: 2010 International conference on computer application and system modeling (ICCASM), vol 1, pp V1–463–V1–467
24. Murugan P, Kannan S, Baskar S (2009) Nsga-ii algorithm for multi-objective generation expansion planning problem. Electr Power Syst Res 79(4):622–628
25. Oswald S, Caldwell R (2007) Dispensing error rate after implementation of an automated pharmacy carousel system. Am J Health Syst Pharm 64(13):1427–1431
26. Wang X-J, Zhang C-Y, Gao L, Li P-G (2008) A survey and future trend of study on multi-objective scheduling. In: Fourth International Conference on Natural Computation, vol 6, pp 382–391, Jinan. IEEE
27. Zhao X, Yun C, Liu X, Wang W (2008) Modeling and simulation of the automated pharmacy system. In: 2008 International conference on intelligent computation technology and automation (ICICTA), vol 1, pp 621–625