Proceedings of the 2018 Winter Simulation Conference
M. Rabe, A.A. Juan, N. Mustafee, A. Skoogh, S. Jain, and B. Johansson, eds.

IMPLEMENTING A NEW GENETIC ALGORITHM TO SOLVE THE CAPACITY ALLOCATION PROBLEM IN THE PHOTOLITHOGRAPHY AREA

Amir Ghasemi
Cathal Heavey

Enterprise Research Center
University Of Limerick
Castletroy
Limerick, V94 T9PX, IRELAND

Kamil Erkan Kabak

Department of Industrial Engineering
Izmir University Of Economics
Sakarya Caddesi, No:156
Balçova - İzmir, 35330, TURKEY

ABSTRACT

Photolithography plays a key role in semiconductor manufacturing systems. In this paper, we address the capacity allocation problem in the photolithography area (CAPPA) subject to machine dedication and tool capability constraints. After proposing the mathematical model of the considered problem, we present a new genetic algorithm named RGA which was derived from a psychological concept called Reference Group in society. Finally, to evaluate the efficiency of the algorithm, we solve a real case study problem from a semiconductor manufacturing company in Ireland and compare the results with one of the genetic algorithms proposed in the literature. Results show the effectiveness and efficiency of RGA to solve CAPPA in a reasonable time.

1 INTRODUCTION

Photolithography is one of the most crucial processes in semiconductor manufacturing. It is often regarded as a bottleneck process due to the layered nature of wafer fabrication, which is attributed to expensive clustered steppers and high tool utilization requirements. Essentially, the photolithography process includes three main steps. These steps are coat, expose and develop. Of these three steps, the expose step covers the masking of the integrated circuit pattern onto the wafer with ultraviolet light Kabak et al. (2008). In this paper, we consider the capacity allocation and scheduling problem for the photolithography area (CAPPA) subject to machine dedication and tool capability constraints. As the CAPPA problem is NP-Hard (Chung et al. 2008) and genetic algorithms are one of the most efficient metaheuristics to solve NP-Hard problems (Beheshtinia and Ghasemi 2017), first we propose a new mathematical formulation of this problem and then a new genetic algorithm. To evaluate our algorithm, different sets of real data from a real case data are solved and compared with the genetic algorithm (GA) proposed by Chen et al. (2016). The rest of paper is organized as follows: Section 2 provides a literature review on photolithography optimization papers. Section 3 proposes a mathematical formulation, Section 4 describes our solution methodology, and Section 5 presents numerical solutions. Finally, Section 6 gives the conclusion and future research area.

2 LITERATURE REVIEW

This study addresses the problem of capacity allocation for the photolithography in semiconductor manufacturing. The problem is commonly referred to as the CAPPA problem (Chung et al. 2008), which considers both the machine capability and the machine dedication constraints.

The study by Leachman and Carmon (1992) is one of the earliest studies on the capacity allocation problem considering the machine capability constraints. In this study, they define an `alternative machine set' to represent a capability for a particular operation. Also, a linear programming (LP) formulation of revisits of products to process areas requires a huge number of decision variables since the number of

variables increases as the number of alternative machine types to the power of the number of re-entrant visits (Leachman and Carmon 1992).

Another version of the CAPPA problem is defined by Toktay and Uzsoy (1998) as a maximum flow network model. In this formulation, tooling and set-up constraints apart from machine capabilities are included. The capacity allocation problem is also considered as a sub-problem by Akçah et al. (2001) for a shift scheduling problem together with the sub-problem of lot sequencing. Different than the other studies, they use two different sets for machine capabilities, one is for the processing capability of all operations, and, the other is for the processing capability of a partial set of operations. Hung and Cheng (2002) introduce a capacity partition generation procedure (CPGP) by relaxing the uniformity assumption applied by them.

Some earlier studies consider just the machine dedication constraints for the capacity allocation problem. To illustrate, Akcali and Uzsoy (2000) point out that both the average and variation of the photolithography cycle time can be reduced significantly with a flexible assignment policy. Pham et al. (2008) apply a machine dedication constraint using integer programming for the photolithography scheduling.

The CAPPA problem is tackled with both machine capability and dedication constraints since the studies by Chung et al. (2008) and Chung et al. (2006). The CAPPA problem is solved by leveling the capacity utilization rates of the machines using an integer programming model (Chung et al. 2006). The time complexity of the problem is reported in this study and the need for heuristics is emphasized for large-scale instances. To overcome the computational issue, Chen and Chen (2010) introduce six modified heuristics of Sule's Algorithm (MSAs) and a linear-programming based heuristic algorithm (LPBHA) for the CAPPA problem. Chen and Chen (2010) also compare five heuristics according to the required number of machines of each type with capability constraints. These heuristics are two heuristics for capacity and capability planning of lower bound (CCP-L) and of upper bound (CCP-U), two heuristics for modified versions of the former (MCCP-L) and latter (MCCP-U) heuristics, and modified capacity and capability planning of cost ratio (MCCP-CR). The algorithm using the best ratio of production efficiency and equipment cost to select the machine type with capability constraint results in the least required number of machines, the highest machine utilization, and the lowest equipment investment. Chen et al. (2010) analyze the capacity allocation under the infinite capacity planning system (ICPS) framework. They point out that only a few studies are performed at the back-end part of semiconductor manufacturing when compared to the studies at the front-end in the literature. Also, they highlight the consideration of dual resources of equipment and the jig together with capacity and capabilities simultaneously.

Another important factor in scheduling photolithography operations is the availability of reticles which is considered by Diaz et al. (2005). To show the impacts of the reticle requirements in the production environment they consider a discrete-event simulation model of the photolithography station and coupled with a network flow optimization model that optimizes the location of all reticles at 6-hour intervals. Recently, Chen et al. (2016) apply a genetic algorithm for solving the CAPPA problem considering reticle constraints apart from machine dedication and capability constraints. Kriett and Grunow (2017) evaluate linear capacity constraints for unrelated parallel machines by using a procedure that generates an irredundant set of low-dimensional constraints. That is, the constraints involve one decision variable for each product type.

3 AN IMPROVED MIXED INTEGER PROGRAMMING MODEL

In this section, the mathematical model of the problem is introduced based on the model proposed by (Chung et al. 2008). Before introducing the model formulation, the common notation used throughout this paper is provided below.

Indices:

i Index of order number, where $i = 1,2, \dots, I$

l Index of layer number, where $l = 1,2, \dots, L_i$

k Index of machine number, where $k = 1,2, \dots, K$
h Index of processing capability number, where $h = 1,2, \dots, H$
t Index of the planning period, where $t = 1,2, \dots, T$

Parameters:

C_{hk} If machine k has processing capability h, then $C_{hk} = 1$ otherwise $C_{hk} = 0$.
AC_{kt} Available capacity of the machine k in planning period *t*.
CL_{il} If layer l of order i is a critical layer then $CL_{il} = 1$ otherwise $CL_{il} = 0$.
CR_{ih} If the critical layer operations of order i requires process capability h then $CR_{ih} = 1$ otherwise $CR_{ih} = 0$.
CR_{ilh} If layer l of order i has a load on processing capability h then $CR_{ilh} = 1$ other wise $CR_{ilh} = 0$.
DML_{it} Loading of critical layer operations of order i in period t.
L_i A number of photolithography operations for order i.
LT_{ilt} If layer l of order i has a load in planning period t then, $LT_{ilt} = 1$ otherwise $LT_{ilt} = 0$.

p_{il} Processing time for layer l of order i.

Decision variables:

dm_{ik} If the first critical layer of order i is assigned to machine k then, $dm_{ik} = 1$ otherwise $dm_{ik} = 0$.
x_{ilk} If layer l of order i is assigned to machine k, then $x_{ilk} = 1$ otherwise $x_{ilk} = 0$.

ML_t The maximum loading level among machines in the planning period t.

Objective function:

$$Minimize \sum_t ML_t \tag{1}$$

Subject to:

$$\sum_t \sum_k \sum_h \sum_l x_{ilk} C_{hk} CR_{ilh} LT_{ilt} = \sum_t \sum_h \sum_l CR_{ilh} LT_{ilt} \ , for\ all\ i \tag{2}$$

$$\sum_k x_{ilk} = 1 \ , for\ all\ i, l \tag{3}$$

$$\sum_t \sum_h \sum_l (x_{ilk} Cl_{il} CR_{ilh} LT_{ilt}) = dm_{ik} * \sum_t \sum_h \sum_l (Cl_{il} CR_{ilh} LT_{ilt}) \ , for\ all\ i, k \tag{4}$$

$$\sum_i \sum_l \sum_h (x_{ilk} p_{il} C_{hk} CR_{ilh} LT_{ilt}) \leq ML_t \ , for\ all\ t \tag{5}$$

$$x_{ilk}, dm_{ik} \in \{0, 1\}\ for\ all\ i, l, k \tag{6}$$

$$ML_t \geq 0 \tag{7}$$

The objective function (1) is to minimize the sum of ML_t, the maximum loading level among machines in plannithe ng period *t*. Lower amounts of ML_t would guarantee better work flow between machines during weeks so by minimizing the sum of ML_t, the results tend to balance the load among machines.

Constraint (2) ensures that each layer of an order, including new released orders and WIP orders, must be assigned to a machine *k* if it has a capacity request in this planning horizon. In the machine assignment, the process window constraint must be considered. Constraint (3) is to make sure that each layer of an order can only be assigned to a single machine. Constraint (4) indicates the machine dedication control. Note that the orders in a planning horizon can either be orders planned to be released or WIP orders that were released to the shop floor in the previous planning horizon. Therefore, dm_{ik} is a decision variable if the order is a planned-to-release order or a WIP order for which its first critical layer has not been assigned to a particular machine in the previous planning horizon; otherwise, dm_{ik} is a known parameter. Constraint (5) ensures that capacity loading of each machine in a period must be smaller than or equal to the maximum loading among machines in that planning period, Ml_t. Constraint (6) mentions the type of decision variables and constraint (7) that ensures the positivity of Ml_t values.

4 PROBLEM-SOLVING

The genetic algorithm (GA) is one of the most frequently used algorithms in the capacity allocation and scheduling problems (Beheshtinia et al. 2018). In GA, the initial population is obtained from randomly generated chromosomes. Then, the initial population is improved through GA operators such as crossover, mutation, and selection. Research in GA presents various versions of algorithms. Qu et al. (2013) combine an improved genetic algorithm (GA) with a co-evolution mechanism to solve the global path-planning problem for multiple mobile robots. Li et al. (2015) use a combination of GA and simulated annealing (SA) to forecast vessel traffic flow by robust v-support vector regression model. In this paper, a new GA derived from sociological theory is presented. The sociological theory used in the algorithm is the theory of social role models.

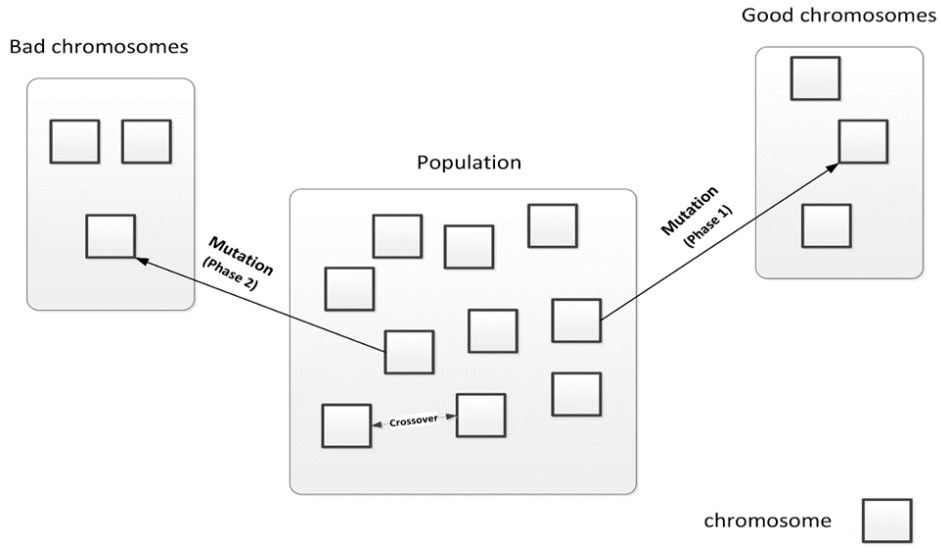

Figure 1: Mechanism of RGA.

The concept of reference groups in society is presented by Merton (1957). He believed that people like heroes or entertainment artists, whom he calls role models, influence other members of every society. This concept is employed in the proposed GA, namely Reference Group Genetic Algorithm (RGA).

In RGA, a list of good chromosomes having the best fitness functions and a list of bad chromosomes with the worst fitness functions are saved in two separate sets. RGA uses these lists in the algorithm operators like a mutation to achieve an optimal solution. Figure 1 (last page) illustrates the mechanism of RGA. This algorithm is introduced by Beheshtinia et al. (2018). They call their algorithm as RGGA and

apply it to a supply chain delivery planning problem. Here, this algorithm is adapted and applied to the CAPPA problem, where new chromosome types and update algorithm parameters are used.

4.1 Chromosomes Structure

A random-key genetic algorithm is an evolutionary metaheuristic for discrete and global optimization. Each solution is encoded as an array of n random keys where a random key is a real number, randomly generated, in the continuous interval. A decoder maps each array of random keys to a solution of the optimization problem being solved and computes its cost. The algorithm starts with a population of p arrays of random keys (Ruiz et al. (2015)). The RGA chromosomes have a bi-level random key structure. The first level defines the orders and related layers (i, j), and the second level addresses allocation and sequencing of order layers to machines.

Example 1- Assume that there are 3 orders with their specified number of layers planned to be produced in each week. In this period, Figure 2 determines the chromosome structure of an arbitrary production plan answer for a week. The string is composed of 6 numbers (each number is related to an order and its layer). The numbers of the string are randomly generated from the continuous distribution U[1, $Nm+1$]; in which Nm is the number of machines. The integer part of each number indicates the selected machine to process the related order. The sequence of assigned orders to a machine is obtained by sorting them by the decimal part of each number. It is worth mentioning here that machine dedication and process window constraints are considered when the random numbers are generated. Therefore, Figure 2 shows the assignment of order 2 with layer 3, order 1 with layer 1, order 3 with layer 1 and order 3 with layer 3 to the first machine, respectively. Order 2 with layer 2 and order 3 with layer 2 are assigned to the second machine. The sequence of assigned orders to a machine is obtained by sorting them by the decimal part of each number. For example, in figure 2 1st layer of order 1 assigned to the first machine the same as a 3rd layer of order 2 and because the decimal part of the 3rd layer of order 2 is smaller than the 1st layer of order 1 it should be produced first.

i, j	1,1	1,2	2,3	3,1	3,2	3,3
Assignment	1.25	2.17	1.13	1.47	2.29	1.57

Figure 2: A feasible chromosome.

4.2 Calculation of a Chromosome

In the remaining elements of the algorithm, it is required to calculate the objective function of newly obtained chromosomes. In each chromosome, different layers of orders assigned to the machines and their processing priority are determined. The objective function of each chromosome is calculated as follows:

- Step 1- Schedule the layers of orders based on their assignment to the machines and processing priority by considering machine dedication and process window constraints.
- Step 2- Calculate the loading time for each machine in the specified week.
- Step 3- Calculate the cumulative loading time for all weeks and machines.

4.3 Crossover and Mutation

Two sub procedures named imitation procedure and distinguish procedure are used in crossover and mutation operators.

Imitation procedure: There are two chromosomes in the imitation procedure; one of them as the influencer, and the other one as the influenced. Influenced chromosome inherits a number of the influencer chromosome features. In this procedure, a gene from the influenced chromosome is selected randomly. It is checked whether the corresponding gene on the influencer chromosome is equal to it or not. If the value

is not equal, then the value of the gene in the influenced chromosome turns into the influencer one. If the value of the gene in the influenced chromosome is similar to the influencer one, no change is needed. The overall structure of this procedure is shown in Figure 3.

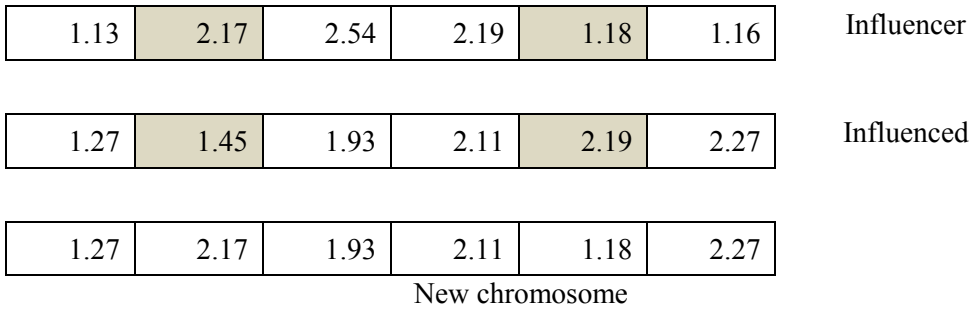

Figure 3: Imitation procedure.

Distinguish procedure: There are two chromosomes in the distinguish procedure, one of them as the influencer, and the other one as the influenced. In this procedure, the influenced chromosome wants to be different from the influencer in some features. In this procedure, a gene from the influenced chromosome is selected randomly. It is checked whether the integer part of the gene on the influencer chromosome is equal to it or not. If the value is equal, then the value of the gene in the influencer chromosome is replaced by a new random number. If the value of the gene in the influenced chromosome is not similar to the influencer one, no change is needed.

The crossover and mutation operators of the algorithm are as follows:

- **Mutation:** Some people in the community are known as good models, and some are known as bad models. People like to imitate good models and avoid the bad models. To perform a mutation operator in RGA, a chromosome from the population is considered as the influenced. Then, the mutation operator is employed in two steps. In the first step, a chromosome is selected from the good chromosomes set and an imitation procedure occurs between them. In the second step, a chromosome is selected from the bad chromosomes set and a distinguish procedure occurs between them. The mutation is done for all population members one by one.
- **Crossover:** People affect each other in society. In the crossover operator, two chromosomes are chosen randomly from the population. One of them is considered as the influenced and another as the influencer and an imitation procedure occurs between them.

It is worth mentioning here that as the first population generated completely feasible, during implementing mutation and crossover operators' solutions will remain in the feasible area. In other words, in RGA operators change the order assignments from one machine to another one and because all assignments generated for orders in the first generation are feasible just two feasible assignments will be changed.

4.4 The Algorithm Steps

The steps of the algorithm are as follows:

Step 1- Create an initial population: create *popsize* random chromosomes as following steps (*popsize* is one of the parameters of the algorithm that determines the size of the population.)

- **Step 1-1**- create chromosome: create a string each of which having *No* array (gene). The value of each array in the string should be a random number from the uniform distribution U[1, *Nm+1*].
- **Step 1-2**- Calculate the objective function value of each chromosome

Step 2- Create the set of good and bad chromosomes: assign the number of *Num_Good* chromosomes with the best objective function to the good chromosomes set and the number of *Num_Bad* chromosomes with the worst objective function to the bad chromosomes set.

Step 3- Perform crossover: the number of crossover operations in each iteration is constant and determined by a coefficient of *popsize* named *cross_rate* that is one of the parameters of the genetic algorithm. Perform crossover as follows:

- **Step 3-1-** Choose the chromosomes: selecting two chromosomes randomly and name them P1 and P2.
- **Step 3-2-** Perform the imitation procedure between P1 and P2: choose a random gene of P1 and copy its value to the related gene on chromosome P2. Likewise, select a random gene of P2 and copy its value to P1.
- **Step 3-3-** Calculate the objective function and replace the chromosomes: calculate the objective function new value of chromosomes P1 and P2, and replace the previous chromosomes with the new ones.

Step 4- Mutation: the number of mutations in each iteration is constant and determined by the coefficient of *popsize* named *mut_rate* that is one of the parameters of the algorithm. The mutation mechanism is as follows:

- **Step 4-1-** Select a chromosome for the mutation, a good and a bad chromosome: select a random chromosome from the population, a random chromosome from the good chromosomes set and a random chromosome from the bad chromosomes set and name them as C, GC, and BC, respectively.
- **Step 4-2-** Imitate from the good chromosome: select a random gene from GC and copy its value to its corresponding gene in C.
- **Step 4-3-** Avoid the bad chromosome: choose a random gene from C and consider its value as VC, and its corresponding value in BC as VBC. If the integer parts of both numbers are identical, change the integer part of VC. (Do this for the decimal part of the number, too).
- **Step 4-4-** Calculate the objective function and replace chromosome: calculate the objective function for the new chromosome C and replace the previous chromosome C with the new one.

Step 5- Check the termination criterion: If the best chromosomes objective function value does not improve in several consecutive generations, terminate the algorithm. The number of these consecutive repetitions is showed by *ter_num,* and it is a parameter of the algorithm. If the termination criteria is not achieved, go to step 6.

Step 6- Updating the set of good and bad chromosomes: In the current population, if there are chromosomes that their objective function is better (or worse) than good chromosomes (bad chromosomes) set, place it in the good (bad) chromosomes set and delete the worst (best) chromosome from the list.

We empirically found that the values of 0.8 for cross_rate, 0.2 for mut_rate, 50 for population size, 10 for *Num_Good* and *Num_Bad* and 20 for *ter_num* give good results in a reasonable time.

5 NUMERICAL EXPERIMENTS

To evaluate the performance of RGA, we compared it with the GA proposed by Chen et al. (2016) using a set of real case gathered from a semiconductor manufacturing fab in Ireland. To illustrate the input an example of a data set is shown in Tables 1 and 2. It includes 2 orders and different layers features besides the capability of different 9 machines in addition to Processing time (h), load occurrence time (week), required process capability, whether a critical layer operation is included or not (1: critical layer operation; 0: non-critical layer operation), respectively. In Table 2, 1 means that the machine has this certain process capability; 0 means that the machine does not have this certain process capability. The full real data sets can be found at data sets for CAPPA in Winter Simulation Conference 2018, last accessed August 18, 2018.

Table 1: Example of real data input sets.

Order	*Layer*	*1*	*2*	*3*	*4*	*5*	*6*
1	Process time	42	36	44	43	38	44
	week	1	1	1	1	1	1
	capability	9	10	3	1	5	9
	critical	0	1	0	0	0	0
2	Process time	40	41	40	45	42	38
	week	1	1	1	1	1	1
	capability	7	3	8	7	10	11
	critical	0	0	1	0	0	0

Table 2: Example of machine features in the input data sets.

	cap1	*cap2*	*cap3*	*cap4*	*cap5*
m1	1	1	0	1	1
m2	1	1	1	0	1
m3	0	0	0	0	1
m4	0	1	1	0	1
m5	1	1	1	1	1
m6	1	1	1	0	1
m7	1	0	1	0	1
m8	1	0	1	1	1
m9	1	1	0	0	1

5.1 Computational Results

The generated test data sets are solved by RGA and the GA proposed by Chen et al. (2016). All algorithms are coded in MATLAB and run on a PC with an Intel Core i7, 1.80 GHz CPU with 8 GB of Ram. Based on our real case, there are 85 orders with a different number of layers in 3 weeks for each order and the number of machines is 9 with the different production. Table 3 shows the results of the algorithms in solving

the problem. Results are based on different loading levels for each machine during 3 weeks with the objective function of maximum loading for each week based on different allocations of algorithms and finally, the cumulative amount of maximum loadings is considered.

Table 3: Results of algorithms applied to CAPPA real case.

		Machine loading		*Maximum loading*		*CPU time*	
		RGA	GA	RGA	GA	RGA	GA
	m1	3842	3703				
	m2	4184	4104				
	m3	3101	2700				
	m4	2764	3344				
week1	m5	5200	5824	5200	5824		
	m6	4705	4693				
	m7	3157	2946				
	m8	4075	4849				
	m9	4117	2982				
	m1	4480	4262				
	m2	4375	4120				
	m3	4320	2141				
	m4	2870	3415				
week2	m5	4440	5639	4480	5639	1373.2	1758.5
	m6	4407	4509				
	m7	3541	4355				
	m8	3293	4182				
	m9	4412	3515				
	m1	4010	3053				
	m2	4187	4359				
	m3	4153	4274				
	m4	3454	3204				
week3	m5	4189	4027	4189	5142		
	m6	4142	5095				
	m7	3748	2423				
	m8	3281	3734				
	m9	4147	5142				
Total				13869	16605		

5.2 Results Discussion

One of the most challenging problems in the photolithography process is to have a constant flow between machines which means planning not to have some machines idle and some others working during the whole planning period. Preventing such issues can increase the efficiency of the production line (Chen et al. 2016). In Table 3, there is a considerable difference between the loading levels of RGA and GA with lower loading

levels given by RGA. This has a direct influence on maximum loading during each week as we can see the maximum loading levels for the first week are 5200 and 5824 for RGA and GA, respectively. The other point worth mentioning here is that the CPU time to solve the problem using RGA is considerably less than GA which relates to special features of RGA.

5.2.1 Special Features of RGA in Solving CAPPA

To clarify why RGA can be more efficient in solving CAPPA here we describe some of its special features and compare it with GA:

- The imitation operator in RGA enables it to just search in the feasible area. In other words, the first population for both algorithms is from feasible solutions then the algorithm operators play a key role in remaining in the feasible solution. In the RGA the imitation operator changes allocations for a specific gene of each chromosome in two solutions, where the feasible allocation of layers from one solution is copied into another one, therefore, it guarantees the feasibility for the new solution. However, in the mutation procedure of GA, some infeasible allocations result in penalties which reduces the quality of solutions.
- RGA algorithm saves a set of best and worst obtained solutions in the good chromosomes set and the bad chromosomes set lists, respectively. However, GA loses this information.
- In RGA to perform a mutation operator, we need three chromosomes: an ordinary chromosome, a chromosome from good chromosomes set and another from bad chromosomes set. While in the GA the mutation operator needs a randomly selected chromosome.

6 CONCLUSION AND FUTURE RESEARCH

As the demand for semiconductor products increases, along with the changes in wafer technology, semiconductor manufacturers find it increasingly difficult to schedule their capacity efficiently. This study provides a solution to the issue of loading balance among photolithography machines. To solve this problem we proposed a new genetic algorithm named RGA which combines some psychological approaches into the classical GA. To examine our approach we gathered data from a semiconductor company in Ireland to solve the CAPPA problem. Results show the quality of RGA in finding efficient solutions for CAPPA in comparison with GA provided by Chen et al. (2016).

In this paper, the production planning problem is not considered and each layer of an order previously planned to be produced in specified weeks which can be a new research area to integrate production planning and capacity allocation together. In addition, all variables and parameters are deterministic, however, in real cases, the order volumes are highly stochastic. Considering order due dates and new objective functions like total tardiness would enable CAPPA to be more realistic.

ACKNOWLEDGMENTS

This project named Productive 4.0 has received funding from the Electronic Component Systems for European Leadership Joint Undertaking under grant agreement No 737459. This Joint Undertaking receives support from the European Union's Horizon 2020 research and innovation program and Germany, Austria, France, Czech Republic, Netherlands, Belgium, Spain, Greece, Sweden, Italy, Ireland, Poland, Hungary, Portugal, Denmark, Finland, Luxembourg, Norway, Turkey.

REFERENCES

Akcali, E. and R. Uzsoy. 2000. "A Sequential Solution Methodology for Capacity Allocation and Lot Scheduling Problems for Photolithography". In *Proceedings of the 26th IEEE/CPMT International Electronics Manufacturing Technology Symposium*, 374-381.

Akçah, E., K. Nemoto, and R. Uzsoy. 2001. "Cycle-time Improvements for Photolithography Process in Semiconductor Manufacturing". *IEEE Transactions on Semiconductor Manufacturing* 14(1):48-56.

Beheshtinia, M. A. and A. Ghasemi. 2018. "A Multi-objective and Integrated Model for Supply Chain Scheduling Optimization in a Multi-site Manufacturing System". *Engineering Optimization* 50(9): 1415-1433.

Beheshtinia, M. A., A. Ghasemi, and M. Farokhnia. 2018. "Supply Chain Scheduling and Routing in Multi-site Manufacturing System (Case Study: A Drug Manufacturing Company)". *Journal of Modelling in Management* 13(1):27-49.

Chen, J. C. and C. W. Chen. 2010. "Capacity Planning of Serial and Batch Machines with Capability Constraints for Wafer Fabrication Plants". *International Journal of Production Research* 48(11):3207-3223.

Chen, J. C., L. Su, Ch. J. Sun, and M. F. Hsu. 2010. "Infinite Capacity Planning for IC Packaging Plants". *International Journal of Production Research* 48(19): 5729-5748.

Chen, J.C., Y. Y. Chen, and Y. Liang. 2016. "Application of a Genetic Algorithm in Solving the Capacity Allocation Problem with Machine Dedication in the Photolithography Area". *Journal of Manufacturing Systems* 41(1):165–177.

Chung, S. H., C. Y. Huang, and A. H. I. Lee. 2006. "Capacity Allocation Model for Photolithography Workstation with the Constraints of Process Window and Machine Dedication". *Production Planning and Control* 17(7): 678-688.

Chung, S. H., C. Y. Huang, and A. H. I. Lee. 2008. "Heuristic Algorithms to Solve the Capacity Allocation Problem in Photolithography Area (CAPPA)". *OR Spectrum* 30(3): 431-452.

Diaz, S. L. M., J. W. Fowler, M. E. Pfund, G. T. Mackulak, and M. Hickie. 2005. "Evaluating the Impacts of Reticle Requirements in Semiconductor Wafer Fabrication". *IEEE Transactions on Semiconductor Manufacturing* 18(4):622–632.

Hung, Y. F. and G. J. Cheng. 2002. "Hybrid Capacity Modeling for Alternative Machine Types in Linear Programming Production Planning". *IIE Transactions* 34(2): 157-165.

Kabak, K. E., C. Heavey, and V. Corbett. 2008. "Analysis of Multiple Process Flows in an Asic Fab with a Detailed Photolithography Area Model". In *Proceedings of Winter Simulation Conference*, edited by: S. J. Mason et al., 2185-2193. Miami, FL: Informs.

Kriett, P. O. and M. Grunow. 2017. "Generation of Low-Dimensional Capacity Constraints for Parallel Machines". *IISE Transactions* 49(12): 1189-1205.

Leachman, R. C. and T. F. Carmon. 1992. "On Capacity Modeling for Production Planning with Alternative Machine Types". *IIE Transactions* 24(4): 62-72.

Li, M. W., D. F. Han, and W. Wang. 2015. "Vessel Traffic Flow Forecasting by RSVR with Chaotic Cloud Simulated Annealing Genetic Algorithm and KPCA". *Neurocomputing* 157(1): 243-255.

Merton, R. K. 1957. "The Role-Set: Problems in Sociological Theory". *The British Journal of Sociology 8*(2): 106-120.

Pham, H., A. Shr, P. P. Chen, and A. Liu. 2008. "Scheduling for Dedicated Machine Constraint Using Integer Programming". In *20th IEEE International Conference on Tools with Artificial Intelligence*, 499-506.

Qu, H., K. Xing, and T. Alexander. 2013. "An Improved Genetic Algorithm with Co-Evolutionary Strategy for Global Path Planning of Multiple Mobile Robots". *Neurocomputing* 120(1): 509-517.

Ruiz, E., M. Albareda-Sambola, E. Fernández, M. G. C. Resende. 2015. "A Biased Random-Key Genetic Algorithm for the Capacitated Minimum Spanning Tree Problem". *Computers & Operations Research* 57(1):95–108.

Toktay, L. B. and R. Uzsoy. 1998. "A Capacity Allocation Problem with Integer Side Constraints". *European Journal of Operational Research* 109(1): 170-182.

AUTHOR BIOGRAPHIES

AMIR GHASEMI is a Ph.D. researcher in the School of Engineering at the University of Limerick. He is an Industrial Engineering graduate of the Semnan University, Iran and holds an MSc and BSc there. He

published papers in the field of supply chain scheduling and planning using optimization algorithms. His research interests include optimization algorithms, simulation optimization methods applied to production planning problems and supply chain planning. His email address is amir.ghasemi@ul.ie.

KAMIL ERKAN KABAK is an Assistant Professor in the Department of Industrial Engineering, Izmir University of Economics. He received the Bachelor 's degree from the Middle East Technical University in Ankara, Turkey, the Master's degree from the Department of Industrial Engineering, Dokuz Eylul University, Izmir, Turkey, and the Ph.D. degree from the Department of Design and Manufacturing Technology, University of Limerick, Limerick, Ireland. His research interests include combinatorial optimization, data analytics, simulation modeling and decision support systems. His email address is erkan.kabak@ieu.edu.tr.

CATHAL HEAVEY is an Associate Professor in the School of Engineering at the University of Limerick. He is an Industrial Engineering graduate of the National University of Ireland (University College Galway) and holds an M. Eng.Sc. and Ph.D. from the same University. He has published in the areas of queuing and simulation modeling. His research interests include simulation modeling of discrete-event systems; modeling and analysis of supply chains and manufacturing systems; process modeling; and decision support systems. His email address is cathal.heavey@ul.ie.