2021

Goal Programming Approach For Bi-Objective Optimization For A Single Batch Processing Machine

Dheeban Kumar Srinivasan Sampathi
dheebankumar141@gmail.com

ABSTRACT

GOAL PROGRAMMING APPROACH FOR BI-OBJECTIVE OPTIMIZATION FOR A SINGLE BATCH PROCESSING MACHINE

Dheeban Kumar Srinivasan Sampathi, MS
Department of Industrial and Systems Engineering
Northern Illinois University, 2021
Dr. Purushothaman Damodaran, Director

This research considers a real-time problem where jobs need to be batched and scheduled to a single batch processing machine to minimize makespan and maximum tardiness. Jobs must be placed in batches such that the machine capacity is not violated. The jobs considered have unequal ready times, unequal processing times, and unequal sizes. This research aims to develop an effective solution approach for the proposed problem. The problem under study can be denoted as $1|p\text{-batch}, s_j, r_j| C_{\max}, T_{\max}$.

The problem under study is NP-Hard. A new Mixed Integer Linear Programming (MILP) formulation using Goal programming (MILP-G) and Column Generation (MILP-CG) are proposed as enhancements of formulations proposed in the literature and solved using the commercial solver. To avoid the symmetric solution in MILP two symmetry-breaking methods are proposed using goal programming (MILP-G^+ and MILP-GM^+) for a multi-objective function. An experimental study is conducted to evaluate the different goal programming formulation and Column Generation in terms of solution quality and run time.

A set of 225 instances is generated by varying the values of job size, ready time, processing time, and due date. All MILPs are solved using IBM ILOG CPLEX. This research

compares the results of the proposed methods with the results of the weighted residual method (MILP-W) given by Ghrayeb (2020).

Based on the results, MILP-G^+, and MILP-GM^+ outperform MILP-W for 100 and 150 job instances. MILP-G performed better than MILP-W for 100 and 150 job instances but not always. MILP-CG took a long time for higher job instances to solve the subproblem and relaxed problem, so the solution quality was surprisingly low.

The findings of this research directly benefit schedulers who are faced with the ardent task of scheduling hundreds of jobs each day on their batch processing machine. The MILPs proposed in the research give alternative solution approaches for practitioners and academics to explore further to solve the problem under study and extensions of it.

NORTHERN ILLINOIS UNIVERSITY

DEKALB, ILLINOIS

AUGUST 2021

GOAL PROGRAMMING APPROACH FOR BI-OBJECTIVE OPTIMIZATION FOR A SINGLE BATCH PROCESSING MACHINE

BY

DHEEBAN KUMAR SRINIVASAN SAMPATHI

A THESIS SUBMITTED TO THE GRADUATE SCHOOL

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE DEGREE

MASTER OF SCIENCE

DEPARTMENT OF INDUSTRIAL AND SYSTEMS ENGINEERING

Thesis Director:

Dr. Purushothaman Damodaran

ACKNOWLEDGEMENTS

I would like to express my gratitude to my advisor, Dr. Purushothaman Damodaran, for his guidance and mentorship throughout the course of my master's degree. Without his continuous guidance, support, and encouragement this thesis would not have been possible. I am forever grateful and privileged for the expertise and knowledge that he has shared with me. I would also like to thank Dr. Ziteng Wang and Dr. Christine Nguyen for their immense support and guidance. I would like to extend my thanks to all faculty of the Department of Industrial and Systems Engineering for providing me with a diverse set of skills that will be vital to me as an industrial engineer and researcher. Finally, I would like to thank my parents and friends for their continued support and encouragement.

DEDICATION

To my Parents, L. Srinivasan, and B. Thenmozhi, who taught me hard work never fails and to do everything with at most perfection. May I always make you proud.

TABLE OF CONTENTS

Page

LIST OF TABLES

LIST OF FIGURES

Page

CHAPTER 1. INTRODUCTION

Creating things has been an indispensable activity of human civilization. Manufacturing is the conversion of materials into items of higher value using one or more processing and/or assembly operations. The type of manufacturing done by a company depends on the kind of product it produces (Groover, 2010, p. 5). Manufacturing systems can be characterized by various factors: the number of machines, the level of automation, the type of material handling, and so on. Depending on the number of machines, the models are classified as single machines, parallel machines, and job shops. In a single machine, a job consists of one operation that machines can do (Pinedo, 2009, p. 20). The manufacturing setup can be classified as job production, batch production, and mass production based on the type of production (Singh, 2006, p. 3). A single batch processing machine (BPM) can be explained as one machine that can process multiple jobs at a time.

Batch production is similar to job production except in the quantity of product produced. Rather than making one single product, a group of products is produced at one time. Identical products or minor variations are produced in batches based on customer demand or expected demand for products (Kiran, 2019, p. 185). In a batch process, the sequence of operations is accomplished in a certain period. BPM can handle many jobs at the same time. The action on all jobs start and finish simultaneously. In batch processing, the machinery is utilized effectively. It is also cheaper to produce a whole batch than a single item at a time. By processing several jobs at a time, the setup of jobs can also be minimized. Moreover, it reduces the initial setup cost

because a single production line can produce many products (Research and reviews, 2020). Batch processing finds its application in stress testing of multiple printed circuit boards (PCBs) simultaneously (Abedi et al., 2014).

Planning and scheduling are modes of decision-making that are used regularly in many manufacturing and service industries. Scheduling processes play a crucial role in production planning. Orders released in a manufacturing environment must be translated into jobs with set deadlines associated with them. These jobs are often processed in a given sequence. In such an environment, developing a detailed schedule of tasks to be performed helps maintain operational efficiency (Pinedo, 2009). Scheduling of jobs has been the topic of improvement for decades for industries. There are many rules for the scheduling of jobs. Scheduling has found application in various fields such as healthcare, airline, manufacturing, etc.

It is crucial to schedule jobs on a batch processing system, as incorrect or unsuccessful batching can lead to a significant bottleneck. Industries must, therefore, be well equipped to quickly make decisions to ensure that jobs are completed on time. There is considerable research on scheduling discrete processing machines, where the machines can process only one job at a time. Seeing the advantages of BPM, the research in this field is increasing.

1.1 Problem Description

The problem discussed in this research is based on a real-life application where PCBs are tested in the environmental stress screening (ESS) chamber. In ESS, the PCBs are subjected to thermal stress for a specified period, which allows the detection of failures. This research intends to improve the efficiency of this ESS chamber while maintaining the customers' satisfaction. The

PCBs are considered as jobs, and ESS is considered as BPM. Given the list of jobs, the PCBs are batched together, and the batches are scheduled for testing on the BPM.

Usually, a PCB differs in size based on its application. Furthermore, each PCB has a specific testing time and ready time (i.e., available time for testing). The ESS chamber allows the testing of multiple boards simultaneously, provided its capacity is not exceeded. The PCBs can be tested longer than the specified time. Consequently, the batch processing time is equal to the longest processing time of all the PCBs in a batch. Similarly, the ready time of the batch is equal to the latest ready time of the PCBs in the batch. Pre-emption of PCBs is not allowed, which means once a batch begins processing, it cannot be stopped to add or remove the PCBs.

The aim is to form batches of jobs and schedule those batches on a BPM to minimize the machine's makespan and minimize the job's maximum tardiness. Minimizing makespan tends to maximize machine utilization. Minimizing the job's tardiness ensures the timely delivery of products to the customer and ensures customer satisfaction. The problem under study can be characterized as $1|p\text{-batch}, s_j, r_j| C_{\max}, T_{\max}$ using the three-field notation $\alpha|\beta|\gamma$ as proposed by Graham et al. (1979). This problem is an NP-hard problem. According to Pinedo (2009), a problem that does not have a polynomial-time algorithm is called NP-hard.

1.2 Objectives and Scope

This research aims to develop appropriate solution approaches to minimize the makespan and minimize the maximum tardiness on a single batch processing machine. Ghrayeb (2020) proposed a mixed-integer linear formulation to the problem under study and solved the formulation using IBM ILOG CPLEX solver. The solver required prohibitively long run times to

solve problems as the number of jobs increased. Consequently, Ghrayeb (2020) developed a simulated annealing approach and a greedy randomized adaptive search procedure. Through an experimental study, it was shown that the heuristic approaches were efficient to find a good solution when compared to the commercial solver – especially on larger problem instances. While heuristics are quick to find a solution, they do not guarantee an optimal solution. This research explores a goal programming formulation of the problem under study to improve computational time and solution quality. An experimental study is conducted, using the same data set as Ghrayeb (2020) did, to compare the different mathematical formulations for the problem under study.

The project's scope includes the batching and scheduling of jobs on the batch processing machine but does not include the processing of the job before or after the batch processing machine. The assumptions made in this research are listed below.

1. Each job has a deterministic processing time, ready time, due date, and size.
2. The capacity of the machine is known.
3. Machine breakdown is not considered.
4. Once the batches are formed, no jobs can be added or deleted.

1.3 Research Benefits

The benefits of this research are as follows:

1. The solution approaches will serve as a decision-making tool for schedulers.
2. The solution approaches will help improve machine utilization and customer satisfaction.

3. This research proposes new formulations that will contribute the body of knowledge on $1|p$-batch, s_j, $r_j|$ $C_{\max}$, $T_{\max}$ problems.

CHAPTER 2. LITERATURE REVIEW

Scheduling of jobs has been the topic of interest since the 19th century. The research on machine scheduling has outgrown various machine environments and research methodologies. The problem of scheduling with the bi-objective function has been extensively studied. Scheduling of jobs on discrete processing machines and batch processing machines has been reviewed. Multi-objective optimization cannot guarantee an optimal solution, unlike single-objective optimization. Therefore, the main aim is to find the best solution without degrading any objective functions. Table 1 shows the list of literature reviewed for this research work.

A multi-objective optimization problem can be converted to a single objective optimization problem by multiplying each objective with weight and summing them together. Multi-Objective Mathematical Programming (MOMP) methods can be categorized as a-priori, interactive, and posteriori (Mavrotas, 2009). In the a-priori method, the decision-maker expresses their preferences before the solution process (e.g., setting goals or weights to the objective functions). In the interactive methods, phases of dialogue with the decision-maker are interchanged with phases of calculation, and the process usually converges, after a few iterations, to the most preferred solution. The decision-maker progressively drives the search toward the most preferred solution. In the posteriori method, the efficient solutions for the problem are generated after which the decision-maker takes a decision based on the needs and requirements.

Table 1. Literature Review Table

Literature Source	Discrete or Batch processing	Machine Environment				Objective function					Constraints					Solution method
		Single machine	Parallel machine		Flow shop	C_{max}	T_{max}	ΣU_j	ΣE_j and/ or ΣT_j	ΣC_j	r_j	s_j	d_j	p_j	S	
			P_m	Q_m												
Akker et al. (2010)	Discrete		✓				✓				✓		✓	✓		Column generation
Bulbul et al. (2003)	Discrete				✓				✓		✓		✓	✓		Column generation
Lee and Jung (1989)	Discrete				✓		✓						✓	✓		Goal programming
Pei et al. (2019)	Discrete				✓	✓								✓		Column generation
Sellen and Hott (1986)	Discrete				✓	✓								✓		Goal programming
Chen and Powell (1998)	Discrete			✓						✓			✓	✓		Column generation
Moghaddam et al. (2010)	Discrete	✓							✓	✓	✓		✓	✓		Fuzzy goal programming
Deliktas et al. (2014)	Discrete	✓				✓			✓	✓			✓	✓		Goal programming
Woo et al. (2019)	Discrete	✓				✓								✓		Column generation

(Continued on following page)

Table 1 (Continued)

Literature Source	Discrete or Batch processing	Machine Environment				Objective function					Constraints					Solution method
		Single machine	Parallel machine		Flow shop	C_{max}	T_{max}	ΣU_j	ΣE_j and/ or ΣT_j	ΣC_j	r_j	s_j	d_j	p_j	S	
			P_m	Q_m												
Wiechman and Damodaran (2015)	Batch	✓				✓					✓			✓	✓	Column Generation
Li and Wang (2018)	Batch	✓				✓						✓		✓	✓	MMAS
Kondakci and Bekiroglu (1997)	Discrete	✓						✓		✓			✓	✓		Moore and SPT
Ronconi and Kawamura (2010)	Discrete	✓							✓				✓	✓		Branch and bound
Damodaran and Gallego (2012)	Batch		✓			✓					✓	✓		✓	✓	Simulated annealing (SA)
Belder and Costa (2018)	Batch	✓								✓	✓	✓		✓	✓	Tabu search and PSO
Ghrayeb (2020)	Batch	✓				✓	✓				✓	✓	✓	✓	✓	SA and GRASP
Kashan et al. (2010)	Batch	✓				✓	✓					✓	✓	✓	✓	MOGA
Rezaeian and Zarook (2018)	Batch	✓				✓	✓				✓	✓	✓	✓	✓	BOGA
Sabouni and Jolai (2010)	Batch	✓				✓	✓					✓	✓	✓	✓	BIOO heuristics

(Continued on following page)

Table1 (Continued)

Literature Source	Discrete or Batch processing	Machine Environment				Objective function					Constraints					Solution method
		Single machine	Parallel machine		Flow shop	C_{max}	T_{max}	ΣU_j	ΣE_j and/ or ΣT_j	ΣC_j	r_j	s_j	d_j	p_j	S	
			P_m	Q_m												
Trindade et al. (2018)	Batch	✓				✓					✓	✓		✓	✓	Symmetric Breaking
Srinivasan Sampathi (2020)	Batch	✓				✓	✓				✓	✓	✓	✓	✓	Goal programming and Column generation

The widely used method is the weighted residual method. In this method, a multi-objective problem is converted into a single objective by assigning a non-negative weight to each objective function and minimizing their sum (Ghrayeb, 2020). Another common method is the ε-constraint method, which minimizes one objective, and redefines the other objectives as constraints less than or equal to some ε. The Pareto-optimal front can be approximated by varying the level(s) of ε. Consider this example,

$$\boldsymbol{min}\ f_1(x), f_2(x), .. f_n(x)$$

$$subject\ to$$

$$x \in S$$

In the ε-constraint method, we optimize one of the objective functions using the other objective functions as constraints, incorporating them in the constraint part of the model.

$$\boldsymbol{min}\ f_1(x)$$

$$subject\ to$$

$$x \in S$$

$$f_2(x) \leq \varepsilon_2$$

.

.

$$f_n(x) \leq \varepsilon_n$$

In the case of the maximization objective, the ≤ sign changes to ≥ sign. One issue with this approach is that it is necessary to preselect which objective to minimize and setting ε_j values. This is problematic because for many values of ε_j there will be no feasible solution (Mavrotas, 2009).

The third method is Goal programming. “Goal or target levels specify the values of the criteria functions in an optimization model that decision-makers consider sufficient or satisfactory” (Rardin, 2017). Goal or soft constraint uses deficiency variables to achieve the target. These deficiency variables are non-negative. For example, the equal to format goal constraints are modeled as (criterion function) – (over satisfaction deficiency variable) + (under satisfaction deficiency variable) = target value.

The section below describes the different solution approaches for scheduling jobs on a single machine with multiple objectives. The solution approach is divided into three categories: the heuristics approach, column generation (CG), and goal programming (GP). Since this research focuses on improving the Pareto front and reducing the computational time, CG and GP are discussed in detail.

2.1 Heuristics and other scalarization techniques

Beldar and Costa (2018) considered the $1|r_j$, batch$|\ \Sigma C_j$ and proposed several metaheuristics approach to solve the problem based on Tabu search and Particle Swarm Optimization (PSO). Damodaran and Gallego (2012) studied $P_m|\ r_j$, batch$|C_{max,}$ and developed a Simulated Annealing (SA) algorithm. It was concluded that the SA solution approach outperforms the Modified Delay (MD) heuristic and Greedy Randomized Adaptive Search Procedure (GRASP). Li and Wang (2018) developed a Max-Min Ant System (MMAS) algorithm to solve a single batch processing machine problem. A local search method, known as the Multiple Jobs Exchange, is proposed to improve the algorithm’s performance by adjusting

jobs between batches. The performance of the MMAS algorithm is compared with CPLEX, as well as several other algorithms. For large population sizes, MMAS outperformed other algorithms.

Kashan et al. (2010) proposed a hybrid genetic algorithm for multi-objective single machine scheduling that quickly converges to Pareto optimal solution. The results prove that this hybrid genetic algorithm outperforms the genetic algorithm. Rezaeian and Zarook (2018) proposed a bi-objective genetic algorithm (BOGA) for the NP-hard problem. They found that the BOGA result was more efficient and faster than the ε-constraint method in generating the Pareto front. Sabouni and Jolai (2010) used dynamic programming and proposed a Pareto approach. This method gave the optimal solution in the case of irreconcilable job groups or boundless batches. Ghrayeb (2020) proposed a methodology for solving the problem of scheduling jobs with unequal ready times, unequal processing times, and unequal sizes on a single batch processing machine, with the objectives of minimizing makespan and maximum tardiness using SA and GRASP. A comparison between SA, GRASP, and CPLEX results were made and concluded that SA seems to be the best approach for the proposed problem.

Kondakci and Bekiroglu (1997) addressed the scheduling problem on a single machine considering total flow time and the number of tardy jobs measures simultaneously and developed several precedence theorems that identify various efficient solution properties. Ronconi and Kawamura (2010) used the branch and bound algorithm for solving the NP-hard problem. The proposed method outperformed the CPLEX result for instances greater than 15 jobs. Pei et al. (2019) studied FJ2| pre-emption | C_{max} no-wait job shop scheduling problems and used column generation to solve the problem.

2.2 Column generation

According to Rardin (2017), “Column generation approaches deals with complex combinatorial problems by first enumerating a sequence of columns representing viable solutions to parts of the problem, and then solving a set partitioning (or covering or packing) model to select an optimal collection of these alternatives fulfilling all problem requirements.” According to Akker et al. (2010), the CG technique works well to minimize total weighted tardiness in identical parallel machines. They proposed a hybrid algorithm that solves all instances with 160 jobs and ten machines. Chen and Powell (1998) solved the single-machine scheduling problem with a CG approach where each column represents a schedule on one machine. They also found that the combination of the Dantzig- Wolfe decomposition method with the CG approach is promising and capable of solving large problems. Pei et al. (2019) constructed an effective CG method by combining tailored dynamic programming and a dedicated branch and bound method and found that CG can solve large-scale instances in acceptable time while CPLEX can handle relatively small cases. Woo et al. (2019) proposed a CG approach using Dantzig- Wolfe decomposition and introduced three approaches for reducing CG computational time.

Bulbul et al. (2003) studied flow shops in settings with penalties for delay in delivering customer orders and costs for holding both finished goods and work-in-process inventory and developed heuristics. The results compared two different reformulations in which the CG scheme is enhanced by the solution of an equivalent Lagrangian relaxation formulation and artificial variables in the LP master problem. Wiechman and Damodaran (2015) used the CG approach for scheduling a batch processing machine with minimization of makespan. The problem can be

represented as $1|\ s_j,\ p_j,$ batch$|C_{max}$. They compared the standard CG method and CG with solution pooling and concluded that CG with solution pooling consistently provides better solutions.

2.3 Goal Programming

Goal program modeling of a multi-objective optimization starts by demanding decision-makers to specify new data: goal or target levels for each criterion used to evaluate solutions. The GP model's objective expresses the desire to satisfy all goals as nearly as possible by minimizing a weighted sum of the deficiency variables (Rardin, 2017). Sellen and Hott (1986) developed a GP model for the general flow-shop case that allows the scheduler to find a more comprehensive optimal solution. According to Sellen and Hott (1986), incorporating both the makespan and flow-time criteria allows the scheduler to consider the effects on machine and job idle-time, as these are equivalent criteria under the standard flow-shop framework. Moghaddam et al. (2010) developed a fuzzy GP model for solving a multi-objective single-machine scheduling problem. The objective was to minimize the total weighted tardiness and completion time. Lee and Jung (1989) studied the production planning of a flexible manufacturing environment, and various superior aspects of the GP model have been discussed over the other models when solving the production planning problems.

Lee and Jung (1989) showed how various objectives could be incorporated into the model. Deliktas et al. (2014) proposed a three-stage solution methodology to solve the multi-objective scheduling problem. A multistage GP is the third stage of the solution methodology to construct a satisfactory schedule. Moghaddam et al. (2010) proposed a fuzzy multi-objective linear programming model for single-machine scheduling problems. It was concluded that the

proposed method produced better results when compared to Wang and Liang's approach for 6 to 7 job instances.

2.4 Research Justification

In MOMP, there is more than one objective function. In general, there is no single solution that simultaneously optimizes all the objective functions. Batch processing machines are standard in the industry, especially in electronics manufacturing. Solving the single-machine problem is vital as solution approaches applied to single-machine problems can be adapted for multiple machines. When considering constraints, the industry's typical constraints are job size, job release time, job processing time, and job due date, so including these in the proposed problem is essential. Additionally, the proposed bi-objective is minimizing makespan and maximum tardiness. These two objectives ensure that machines are not running longer than necessary, and jobs are delivered on time to customers.

From the literature reviewed, it is evident that none of the journals covered solving the proposed problem using column generation and goal programming. Ghrayeb (2020) considered the constraints and objective function similar to the proposed problem, but SA and GRASP were used to solve it. Past research mainly focused on the heuristics approach in solving multi-objective single-machine scheduling problems. This, research aims in developing the goal programming and column generation formulation for the single BPM to improve the solution quality and computational time.

CHAPTER 3. MATHEMATICAL FORMULATION

The Mixed-Integer Linear Programming model (MILP) for the $1|p\text{-}batch, s_j, r_j| C_{max}, T_{max}$ is discussed in this chapter. Below is a description of the notation used:

Sets

$\{j \in J\}$	set of jobs
$\{b \in B\}$	set of batches

Parameters

p_j	processing time of job *j*
s_j	size of job *j*
r_j	ready time of job *j*
d_j	due date of job *j*
S	capacity of the machine
α	weight for makespan objective
β	weight for maximum tardiness objective ($\beta = 1 - \alpha$)*Decision Variables*
C_{max}	makespan or completion time of the last job or batch

T_{max}	maximum tardiness
C_b	completion time of batch *b*
P_b	processing time of batch *b*
R_b	ready time of batch *b*
T_b	tardiness of batch *b*
X_{jb}	a binary variable – 1, if job *j* is processed in batch *b*; 0, otherwise.

3.1 Weighted Residual Method (MILP-W)

The following is the mathematical formulation using the weighted residual method given by Ghrayeb (2020).

Minimize $\alpha C_{max} + \beta T_{max}$ (1)

Subject to

$$\sum_{b \in B} X_{jb} = 1 \qquad \forall j \in J \tag{2}$$

$$\sum_{j \in J} s_j X_{jb} \leq S \qquad \forall b \in B \tag{3}$$

$$P_b \geq p_j X_{jb} \qquad \forall j \in J, b \in B \tag{4}$$

$$R_b \geq r_j X_{jb} \qquad \forall j \in J, b \in B \tag{5}$$

$$C_1 = R_1 + P_1 \tag{6}$$

$$C_b \geq C_{b-1} + P_b \qquad \forall j \in J, b \in B/\{1\} \tag{7}$$

$$C_b \geq R_b + P_b \qquad \forall j \in J, b \in B \tag{8}$$

$$T_b \geq C_b - d_j X_{jb} - M(1 - X_{jb}) \qquad \forall j \in J, b \in B \tag{9}$$

$$X_{jb} \in \{0,1\} \qquad \forall j \in J, b \in B \tag{10}$$

$$C_{max} \geq C_b \qquad \forall b \in B \tag{11}$$

$$T_{max} \geq T_b \qquad \forall b \in B \tag{12}$$

$$C_b, P_b, R_b, T_b \geq 0 \qquad \forall b \in B \tag{13}$$

The objective function (1) is to minimize the weighted sum of makespan (the completion time of the last batch) and maximum tardiness (the maximum tardiness of any job). α and β are the weights given to each objective.

Constraint (2) ensures that each job is processed in only one batch. Constraint (3) ensures that the total size of a batch (i.e., the sum of sizes of all jobs within that batch) does not violate machine capacity. Constraint (4) defines that the processing time of a batch is at least equal to the largest processing time of a job within that batch. Constraint (5) defines that the ready time of a batch is at least equal to the largest ready time of a job within that batch. Constraint (6) defines that the completion time of the first batch is simply the sum of the ready and processing times of that batch. Constraint (7) applies if the ready time of the b^{th} batch is less than the completion time of the $(b\text{-}1)^{st}$ batch; thus, the completion time of the b^{th} batch is the sum of the completion

time of the (b-1)st batch and the processing time of the b^{th} batch. Constraint (8) defines that the completion time of the b^{th} batch is at least the sum of its ready and processing times. Constraint (9) defines that the makespan is at least equal to the completion time of all batches. Constraint (10) defines that the tardiness of a batch is at least equal to the difference between the completion time of that batch and the due date of any job within that batch. Constraint (11) defines that maximum tardiness is at least equal to the tardiness of all batches. Constraints (12) and (13) define binary and non-negativity constraints on decision variables. This mixed-integer linear program can be solved using a commercial solver such as IBM ILOG CPLEX.

Table 2 shows the data for a fifteen-job instance. For different values of α and β, the example problem is solved using IBM ILOG CPLEX. The results of the weighted residual method are shown in Table 3. Based on the weights, the decision-maker has a set of solutions to choose from.

Table 2. Sample problem

p_j	9	10	2	3	3	14	4	10	20	14	8	10	18	11	5
r_j	8	3	2	6	3	0	5	3	5	3	2	6	6	2	0
s_j	5	7	7	7	5	12	15	9	9	3	12	1	5	12	12
d_j	23	16	15	19	21	9	13	24	29	20	23	19	22	20	11

Table 3. Results of the weighted residual method

Alpha	Beta	Objective	Cmax	Tmax
0	1	27	50	27
0.25	0.75	32.75	50	27
0.5	0.5	38.50	50	27
0.75	0.25	43.75	49	28
1	0	48	48	39

Ghrayeb (2020) showed that the above formulation is very computationally intensive, and the commercial solver was not effective to solve larger problem instances. Consequently, a goal programming formation is proposed below.

3.2 Goal Programming Formulation (MILP-G)

In the GP model, the under and over satisfaction variables are introduced. The objective function is the minimization of the weighted sum of the deficiency variables. The new variable introduced in the GP formulation is given below.

T^-, T^+ *is under and over satisfaction deficiency variables for* T_{max}

C^-, C^+ *is under and over satisfaction deficiency variables for* C_{max}

The GP formulation is given below.

Minimize $\alpha\, C^+ + \beta\, T^+$ (14)

Subject to

(2)-(10)

$0 \geq C_b + C^- - C^+ \quad \forall b \in B$ (15)

$0 \geq T_b + T^- - T^+ \quad \forall b \in B$ (16)

$C_b, P_b, R_b, T_b, C^-, C^+, T^-, T^+ \geq 0 \quad \forall b \in B$ (17)

Except for constraints (15) and (16), all others are similar to the main formulation (MILP-W) given by Ghrayeb (2020). In constraints (15) and (16) the completion time and tardiness of the batch are made to zero with the help of under satisfaction and over-satisfaction variables. The objective function (14) minimizes the over satisfaction variables, i.e., C^+ and T^+. The fifteen-job instance shown in Table 2 was solved using this goal programming formulation. Table 4 shows the results. The closer a given algorithm's Pareto front is to (0,0), the higher the quality of its solutions (Ghrayeb,2020).

Table 4. Example Results – MILP-G

Alpha	Beta	Objective	Cmax	Tmax
0	1	27	50	27
0.25	0.75	43.75	50	27
0.5	0.5	38.50	50	27
0.75	0.25	43.75	49	28
1	0	48	48	39

3.3 Goal Programming with Symmetry (MILP-G$^+$)

In any MILP, the generation of the symmetric solution is very common. Trindade et al. (2018), considered two types of symmetries.

1. Two solutions are set to be symmetric if the construction of the batches is equal on both solution and batches processed in the same order.
2. Two solutions are set to be symmetric if the construction of the batches is equal on both solutions, but the batches are processed in a different order which does not modify the makespan.

They proposed a symmetry-breaking constraint for four formulations including $1|s_j, r_j, p\text{-}batch|C_{max}$. According to Trindade et al. (2018), the jobs are first ordered by non-decreasing release times. In particular, we consider that the jobs are indexed satisfying:

$$r_{(1)} \leq r_{(2)} \leq \cdots \leq r_{(n)} \tag{18}$$

In this research, the same concept is applied to MILP-G$^+$ to mitigate the effects of symmetry in MOMP. By eliminating symmetry, the number of constraints and variables can be reduced when compared to the original formulation. The symmetry breaking constraints may alter the solution space, but they help to solve the formulation in a shorter time and in some cases help to find a better solution.

Minimize $\alpha\, C^+ + \beta\, T^+$

Subject to

$$\sum_{b \in B: b \geq j} X_{jb} = 1 \qquad \forall j \in J \tag{19}$$

$$\sum_{j \in J: j \leq b} s_j\, X_{jb} \leq S\, X_{bb} \qquad \forall b \in B \tag{20}$$

$$P_b \geq p_j X_{jb} \qquad \forall j \in J, b \in B : j \leq b \tag{21}$$

$$R_b \geq r_j X_{bb} \qquad \forall j \in J, b \in B \tag{22}$$

$$X_{jb} \leq X_{bb} \qquad \forall j \in J, b \in B : j \leq b \tag{23}$$

(6)-(10)

(15)-(17)

Constraint (19) determines that each job *j* is assigned to a batch *b*, such that $b \geq j$. For example, job 2 can be in batch 2 or any of the higher numbered batches. Constraint (20) determines the batches do not exceed the machine capacity. They also ensure that each batch *b* is used only if job *b* is assigned to it. Constraint (21) determines the processing time of the batch. Constraint (22) determines the ready time of the batch. The remaining constraints are similar to goal programming formulation. Any feasible solution for the MILP-G$^+$ is also feasible for MILP-G with the same objective function (Trindade et al., 2018). An example of fifteen job problem instances arranged in increasing order of ready date is shown in Table 5. The results obtained by solving the formulation for the example instance are summarized in Table 6.

Table 5. Example 15 Job Instance – Increasing order of ready time.

$\boldsymbol{p_j}$	14	5	2	8	11	10	3	10	14	4	20	3	10	18	9
$\boldsymbol{r_j}$	0	0	2	2	2	3	3	3	3	5	5	6	6	6	8
$\boldsymbol{s_j}$	12	12	7	12	12	7	5	9	3	15	9	7	1	5	5
$\boldsymbol{d_j}$	9	11	15	23	20	16	21	24	20	13	29	19	19	22	23

Table 6. Example Results- MILP-G$^+$

Alpha	Beta	Objective	Cmax	Tmax
0	1	29.00	49	29
0.25	0.75	34.00	49	29
0.5	0.5	39.00	49	29
0.75	0.25	44.00	49	29
1	0	48.00	48	37

3.4 Goal Programming with Symmetry Modified (MILP-GM$^+$)

Reducing the effect of symmetry in MILP is an intense area of research, where different strategies are suggested to mitigate the effect of symmetry. In this research, a new symmetric

formulation with Goal programming is proposed. In this formulation, job j is placed in batch b such that $b \leq j$. Similar to MILP-G$^+$, jobs are first ordered by non-decreasing release times. More specifically, we consider that the jobs are indexed satisfying equation (18). The solution for example problem instance of Table 5 is shown in Table 7. In the modified symmetry formulation, a job can be in any batch, which is numbered smaller than the job number. For example, job 2 can be either in batch 1 or 2.

Minimize $\alpha\, C^+ + \beta\, T^+$

Subject to

$$\sum_{b \in B, b \leq j} X_{jb} = 1 \qquad \forall j \in J \qquad (24)$$

(3)-(10)

(15)-(17)

Table 7: Example Result-MILP-GM$^+$

Alpha	Beta	Objective	C_{max}	T_{max}
0	1	27	50	27
0.25	0.75	43.75	50	27
0.5	0.5	38.50	50	27
0.75	0.25	43.75	49	28
1	0	48	48	39

3.5 Column Generation Formulation (MILP-CG)

In the column generation approach, the original formulation is decomposed into a restricted master problem and one or more subproblem(s). The restricted master problem is then linear relaxed and iteratively solved to optimality by utilizing improving columns generated by solving the subproblem. Finally, the restricted master problem is resolved as an integer program to obtain a final feasible solution. This section provides the reformulation of the mathematical model (MILP-G) through decomposition. Dantzig-Wolfe decomposition (Dantzig and Wolfe, 1960) is a method that transforms the original problem into a restricted master problem (RMP) and one or more subproblems (SPs). Therefore, the linear solution can act as a lower bound on the actual solution, or an upper bound for a maximization problem.

Constraints (2)-(5) of MILP-W display a block diagonal structure. Consequently, the MILP-G formulation also displays the block diagonal structure and hence the formulation can be decomposed into one restricted master problem and several subproblems. Solving the subproblem gives a set of jobs that can be processed simultaneously in a batch without violating the machine capacity constraint. Solving the subproblem thus yields the jobs in a batch, the batch processing time, and the batch ready time.

Sets

$\{t \in T\}$ set of Position

Parameters

a_{jt} binary – 1, if job *j* is processed in position *t*; 0, otherwise.

Decision Variables

λ_{tb} a binary variable – 1, if batch *b* is in position *t* is selected; 0, otherwise.

The restricted master problem is,

Minimize $\alpha\, C^{+} + \beta\, T^{+}$

Subject to

$$\sum_{b\in B, t\in T} a_{jt}\, \lambda_{tb} \geq 1 \qquad \forall\, j \in J \qquad (25)$$

$$C_b \geq (C_{b-1} + P_b)\, \lambda_{tb} \qquad \forall j \in J, t \in T, b \in B/\{1\} \qquad (26)$$

$$C_b \geq (R_b + P_b)\, \lambda_{tb} \qquad \forall j \in J, t \in T, b \in B \qquad (27)$$

$$0 \geq C_b + C^{-} - C^{+} \qquad \forall b \in B \qquad (28)$$

$$T_b \geq (C_b - d_j X_{jb} - M(1 - X_{jb}))\, \lambda_{tb} \qquad \forall j \in J, t \in T, b \in B \qquad (29)$$

$$0 \geq T_b + T^{-} - T^{+} \qquad \forall b \in B \qquad (30)$$

$$\sum_{b\in B} \lambda_{tb} \leq 1 \qquad \forall\, t \in T \qquad (31)$$

$$\sum_{t\in T} \lambda_{tb} \leq 1 \qquad \forall\, b \in B \qquad (32)$$

$$\lambda_{tb} \in \{0,1\} \qquad \forall j \in J, b \in B \qquad (33)$$

$$C_b, P_b, R_b, T_b, C^{-}, C^{+}, T^{-}, T^{+} \geq 0 \qquad \forall b \in B \qquad (34)$$

Sub Problem

$$Minimize\ -\sum_{j\in J} \pi_j\ \mathrm{Y}_j \qquad (35)$$

$$\sum_{j \in J} s_j Y_j \leq S \qquad \forall b \in B \qquad (36)$$

$$R \geq r_j Y_j \qquad \forall j \in J \qquad (37)$$

$$P \geq p_j Y_j \qquad \forall j \in J \qquad (38)$$

$$Y_j \in \{0,1\} \qquad \forall j \in J \qquad (39)$$

$$P, R \geq 0 \qquad (40)$$

Constraint (25) ensures that each job is present in at least one position. Constraint (26) and (27) determines the completion time of the batch. Constraints (28) and (30) are the goal programming constraint in which the completion time and tardiness are made to zero using the under and over satisfaction variables. Constraint (29) determines the tardiness of the batch. Constraints (31) and (32) ensure each position has at most one batch and one batch is present at most in one position. The subproblem objective is the minimization of duals of constraint (25). Each column generated from the subproblem represents a batch. The ready time, processing time, and size capacity of the machine are ensured by the subproblem using constraints (36), (37), and (38). Table 8 shows the result of MILP-CG for the example 15 job instance shown in Table 2.

Column generation procedure is briefly described below.

1. Generate an initial basic feasible solution.
2. Solve the relaxed restricted master problem (RMP).
3. Obtain the dual values of the solved RMP.
4. Update the cost coefficients of the subproblem with the dual variable values.
5. Solve the subproblem.

6. Add columns to the set of columns in the RMP.
7. Repeat the steps 2-5, until it exceeds to subproblem limit.
8. Use the generated columns to solve the integer master problem.

Table 8. Example result - MILP-CG

Alpha	Beta	Objective	C_{max}	T_{max}
0	1	112	141	112
0.25	0.75	57	75	51
0.5	0.5	49.50	59	40
0.75	0.25	63	69	45
1	0	68	68	53

Table 9 shows the results of different methods for the sample instance shown in table 2. The quality of the solution depends on the number of times the subproblem (Sub problem limit) is solved. Since this is an NP-hard problem to limit the solving time, we set the subproblem limit to 25 for 15, 30, and 50 job instances and 10 for 100 and 150 job instances. This is because for each iteration the subproblem and the relaxed inter problem takes more time when the number of jobs increases.

Table 9. Comparison between different Methods

Alpha	Beta	MILP-W			MILP-G			MILP-G+			MILP-GM^{+}			MILP-CG		
		Objective	C_{max}	T_{max}	Objective	C_{max}	T_{max}	Objective	C_{max}	T_{max}	Objective	C_{max}	T_{max}	Objective	C_{max}	T_{max}
0	1	27	50	27	27	50	27	29	49	29	32	49	32	112	141	112
0.25	0.75	32.75	50	27	32.75	50	27	34	49	29	37.25	53	32	57	75	51
0.5	0.5	38.50	50	27	38.50	50	27	39	49	29	42.50	53	32	49.50	59	40
0.75	0.25	43.75	49	28	43.75	49	28	44	49	29	47.25	52	33	63	69	45
1	0	48	48	39	48	48	39	48	48	37	52	52	39	68	68	53

From Table 9, MILP-CG has higher solution value when compared to other methods irrespective of alpha. This is because we limit the number of times the sub problem is solved which directly affects the solution quality.

CHAPTER 4. EXPERIMENTATION

To evaluate the performance of the MILP, a set of instances must be generated. Ghrayeb (2020) generated a data set to experiment with their solution approaches. The same data set is used in this study. The data generation is discussed in section 4.1. The MILP goal programming formulation given in chapter 3 is solved with IBM ILOG CPLEX 12.10.0 software. As shown in the MILP formulation, a weighted sum approach is used to evaluate the objective function. The values of α are chosen from set [0, 0.25, 0.5, 0.75, 1] and β = 1- α. All experiments are conducted on an Intel Core i7 processor with 1.8GHHz and 12 GB RAM.

4.1 Data Generation

The processing and ready times were sampled from discrete uniform (DU) variable such that p_j~DU [1, 20] and r_j~DU[0, ρZ], where $Z = \Sigma\, p_j$ and ρ is a positive real number such that $0 \leq \rho \leq 1$. Job sizes are split into three levels, where s_1~DU [1, 15], s_2~DU [15, 30], and s_3~DU[1, 30]. Due dates are calculated as $d_j = r_j + p_j$ + DU [0, γZ], where γ is a positive real number such that $0 \leq \gamma \leq 1$.

Five sets of problem instances were generated by varying the number of jobs (i.e., n = 15, 30, 50, 100, and 150). Similarly, different values of $\rho \in$ {5%,10%, 40%}, and $\gamma \in$ {5%,10%, 40%} were considered while generating the data sets. For each combination of n, ρ, γ, and s_j, five

instances are generated, resulting in a total of 675 instances. For experimentation, one-third of the total set is randomly chosen, for a total of 45 instances per level of *n* (225 total). The machine capacity *S* was assumed to be equal to 30.

4.2 Experiments

After the MILP for the different goal programming models is coded in IBM ILOG CPLEX IDE, all instances are solved. Because the problem under study is NP-hard, a commercial solver takes a prohibitively long time to converge to optimality, so the model is allowed to run for 1800 seconds (30 minutes) or until an optimal solution is found, whichever happens first. The solution found by CPLEX (C_{max}, T_{max}, Objective value), as well as the run time and percentage gap, are recorded. The percentage gap is calculated using (41).

$$\%Gap = \frac{|Best\ Integer\ Solution - Best\ Objective\ value\ of\ Linear\ Problem|}{1 \times e^{-10} + |Best\ Integer\ Solution|} \times 100 \qquad (41)$$

One metric to evaluate the quality of the solution from MILP-G is the percent improvement when compared to MILP-W, as shown in (42). The objective from MILP-W is denoted as *obj*(MILP-W) in the equation. If the value of the percent improvement is positive, then the GP formulation outperforms weighted; otherwise, weighted has found a better solution.

$$\%\ Improvement = \frac{[Obj(MILP-W) - Obj(MILP-G)]}{Obj(MILP-W)} \qquad (42)$$

However, as this is a multi-objective problem, other performance metrics must be considered. In a bi-objective problem, there is no guarantee of a global minimum or maximum concerning both objectives considered. There exists a set of solutions that are superior to the rest of the solutions in the solution space when all objectives are considered, but these solutions may be inferior to other solutions if only one objective is considered (Jolai, 2012).

CHAPTER 5. RESULTS

In any multi-objective problem, no single solution exists that simultaneously optimizes each objective. All MILP formulations were solved using CPLEX for all 225 job instances. The results of the weighted residual method were taken from Ghrayeb (2020). One way to assess the quality of the proposed algorithm is to compare the objective of the different algorithms. Figures 1, 3, 5 and 7 summarizes the comparison of different formulations - Weighted vs Goal, Weighted vs Goal Symmetry, Weighted vs Goal Symmetry modified, respectively. The blue portion of the graph indicates the number of problem instances for which the weighted method had better objective value. The orange part indicates the number of problem instances for which the goal programming formulation had a better objective value and the grey portion indicate the number of instances for which the two objective values are equal.

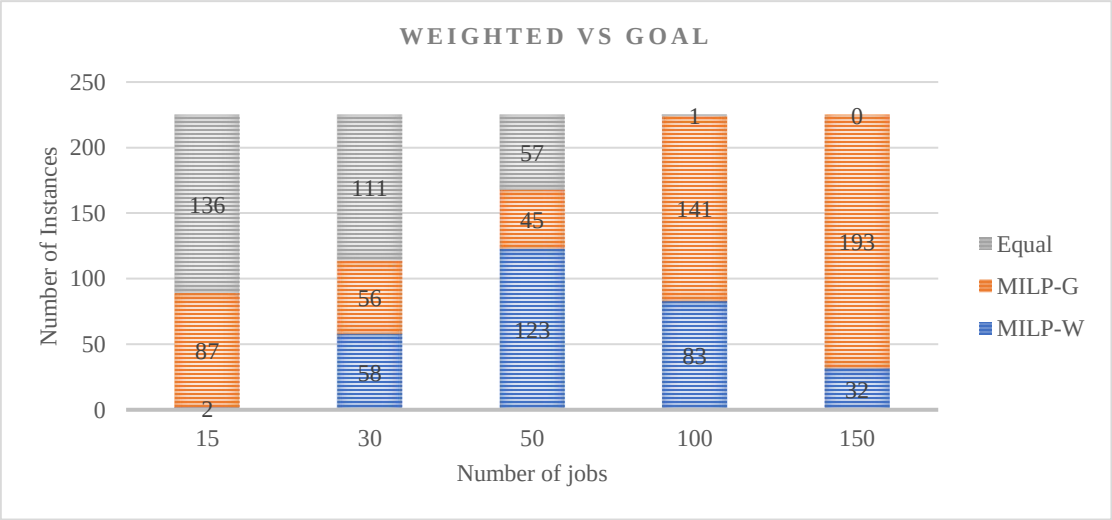


Figure 1. Weighted vs Goal

Figure 1, it is evident that the values were equal, or MILP-G dominated for a 15-job instance. For 30 and 50 job instances, the MILP-G were not dominating but, for higher job instances (100 and 150) MILP-G has better results than MILP-W. Figure 2 shows the result for different values of alpha. These trends hold good when broken down into different values of alpha (Figure 2). For 15 job instance (alpha =0 and 1) majority of the values were equal.

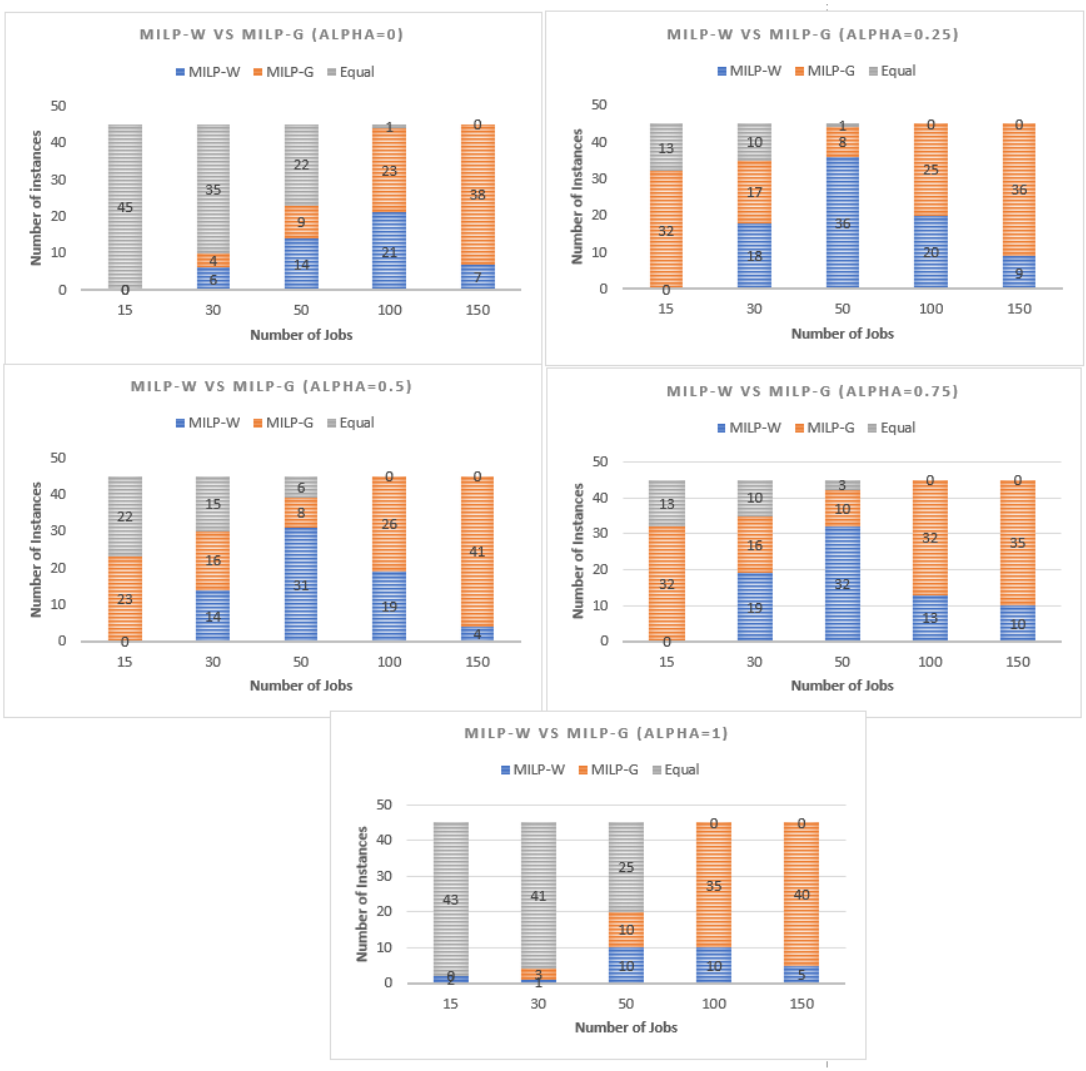


Figure 2. MILP-W vs MILP-G (Different values of alpha)

Figure 3, MILP-G^{+} dominated when the job instances are higher (100 and 150). For 15 and 50 job instances, the MILP-W had better solutions than MILP-G+. In 30 jobs instance, it is equally distributed between two methods. These trends hold good when broken down into different values of alpha (Figure 4). Figure 4, the number of equal values is higher for (alpha =1) for 30 jobs.

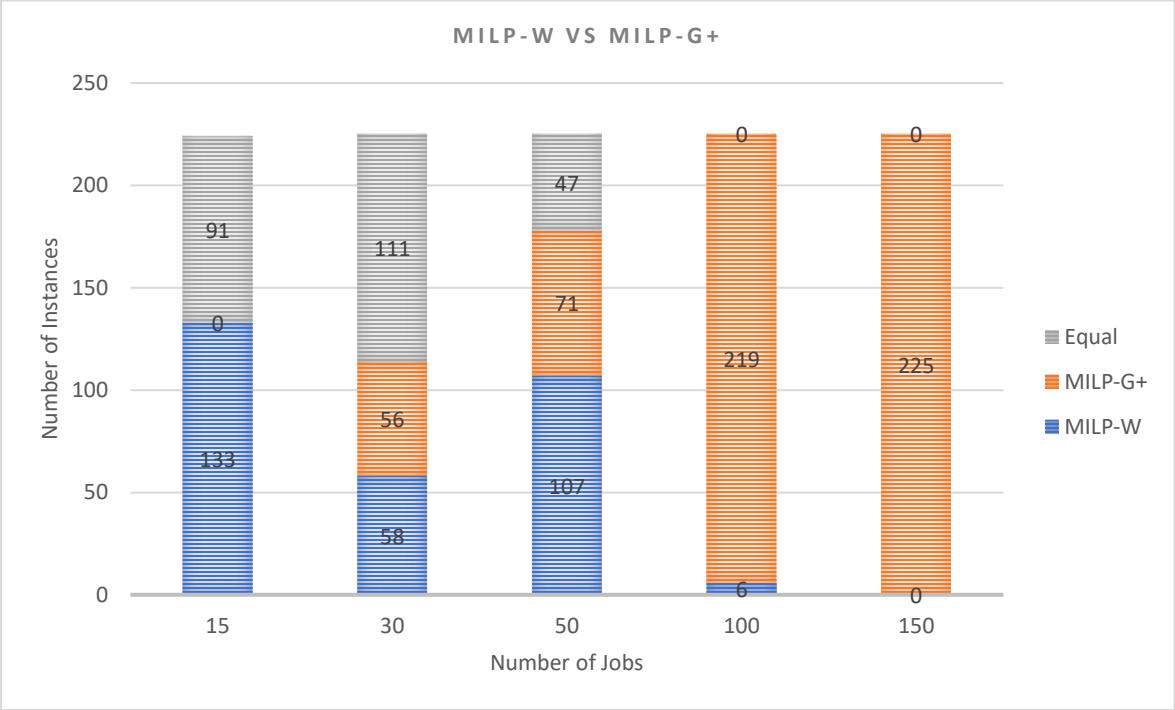


Figure 3. Weighted vs Goal Symmetric

Figure 5, MILP-GM^{+} also dominates for 100 and 150 jobs. For 15, 30 and 50 Jobs MILP-W has better solution in majority of instances than MILP-GM^{+}. Figure 6 shows results of MILP-W vs MILP-GM^{+} for different values of alpha.These trends holds good when broken down into different values of alpha (Figure 6).

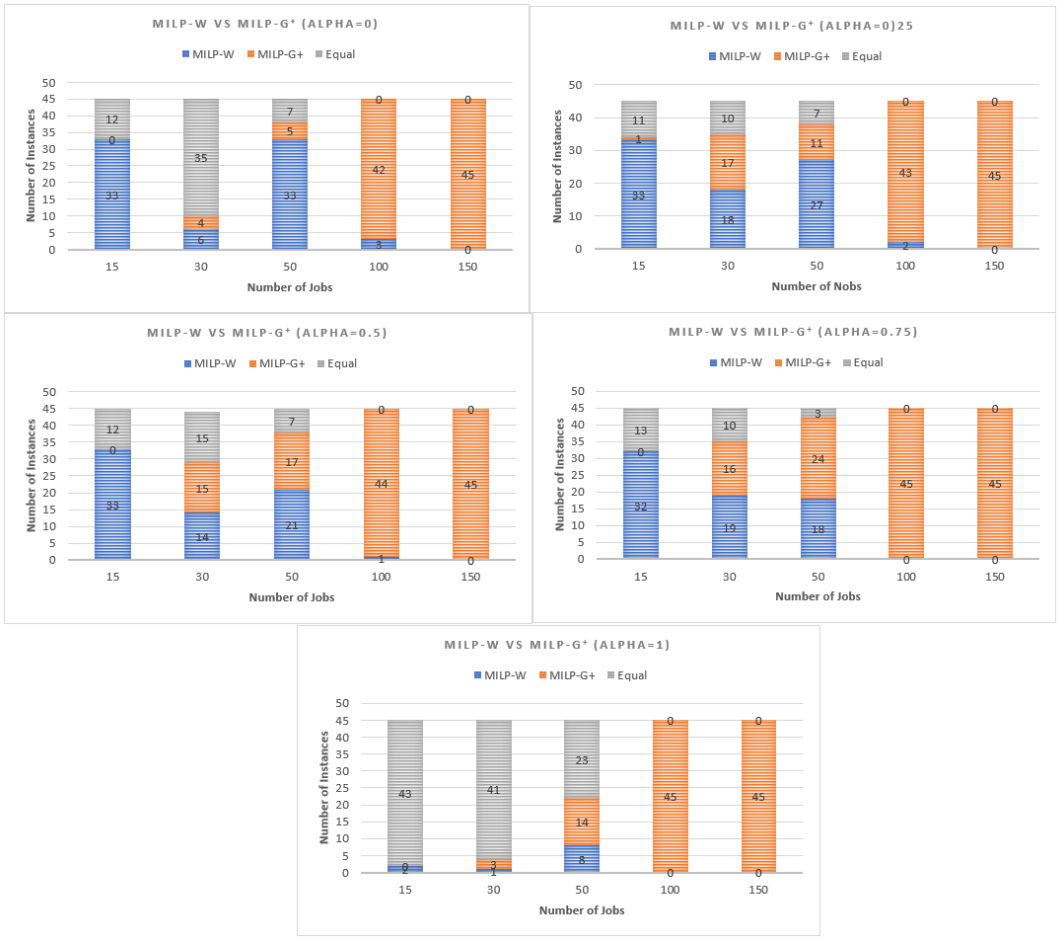


Figure 4. MILP-W vs MILP-G$^+$ (Different values of alpha)

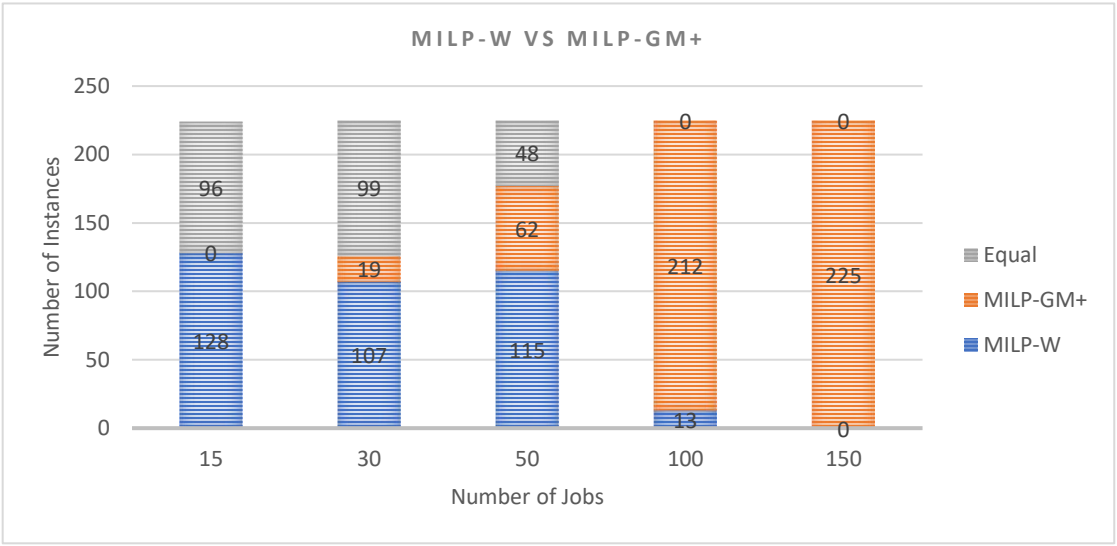


Figure 5. Weighted vs Goal Symmetric Modified

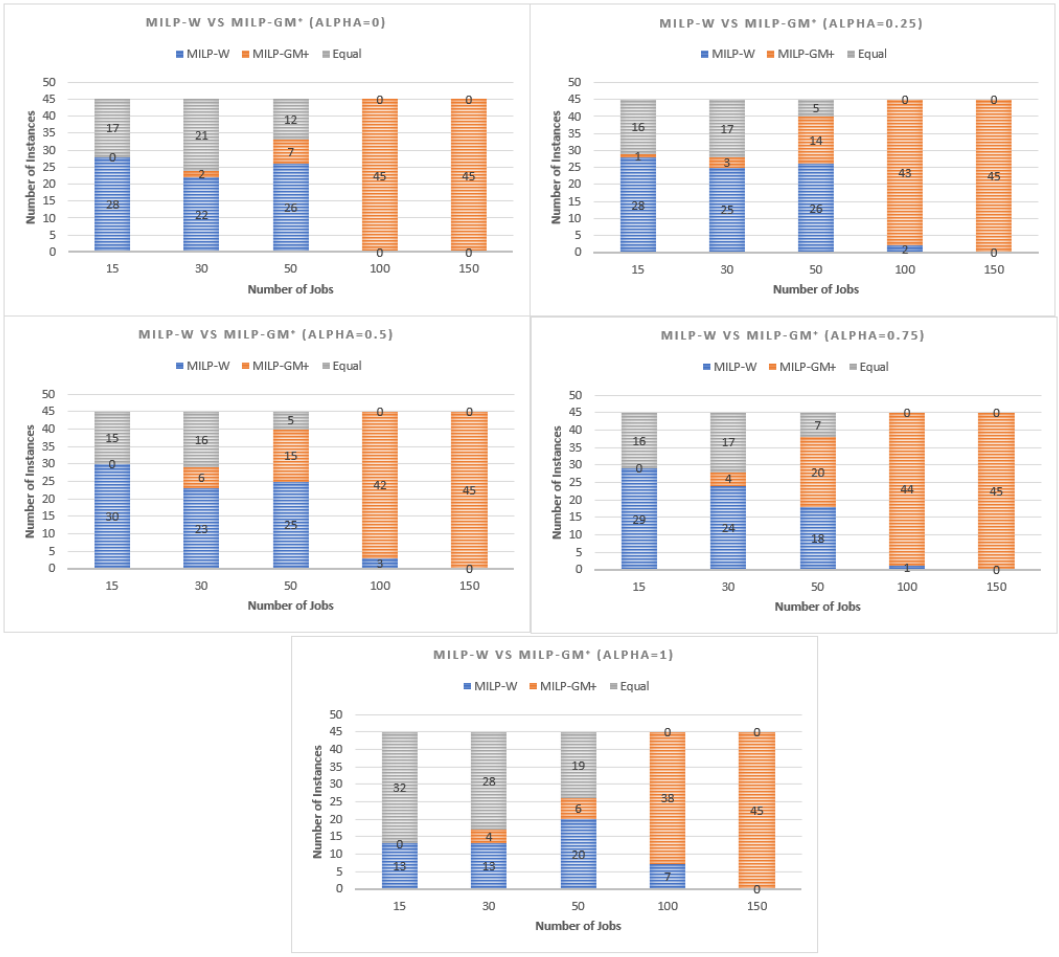


Figure 6. MILP-W vs MILP-GM+ (Different values of alpha)

Figure 7, MILP-CG is completely dominated by MILP-W. The solution MILP-CG depends on the number of sub problem solved. This could be the reason for poor solution of MILP-CG. Figure 8 shows the breakdown of results for different values of alpha. These trends hold good when broken down into different values of alpha (Figure 8).

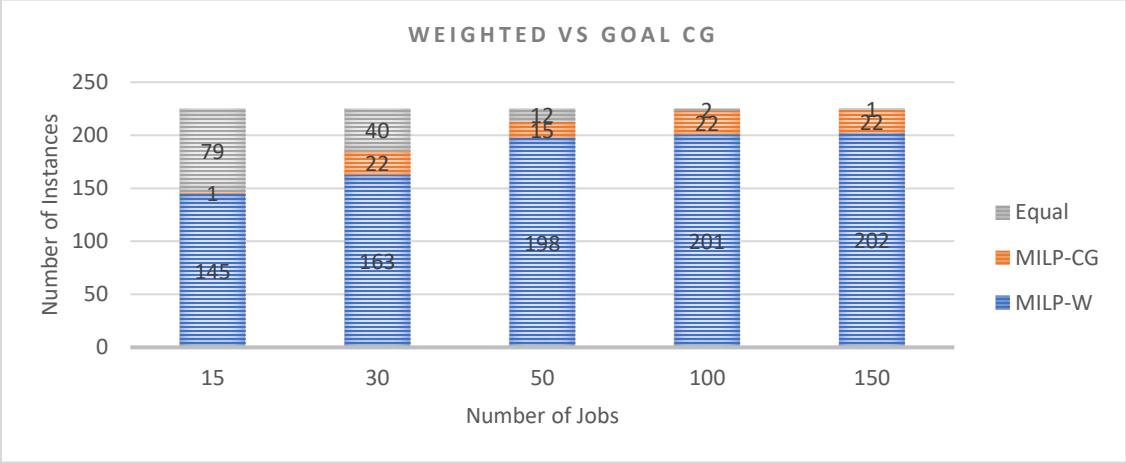


Figure 7. Weighted vs Goal CG

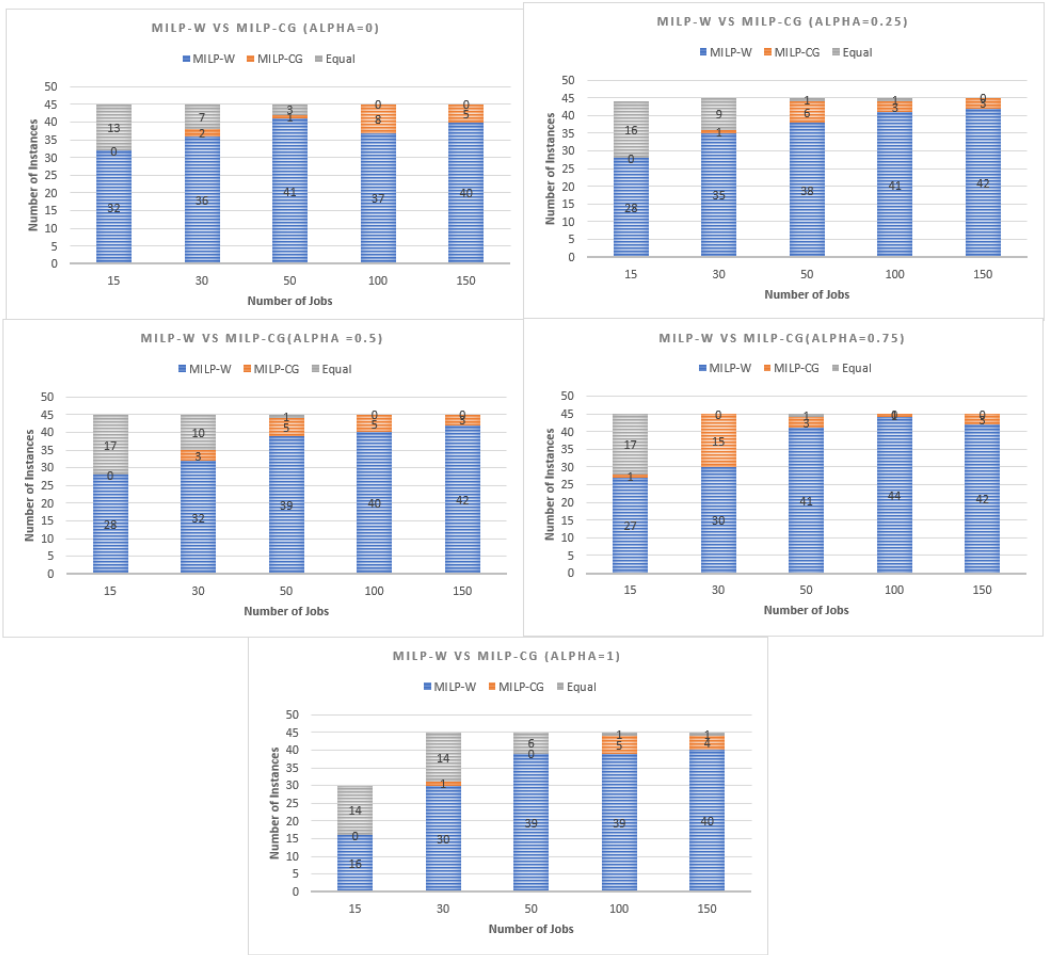


Figure 8. MILP-W vs MILP-CG (Different values of alpha)

5.1 Percentage of a Better solution

Figure 9 shows the percentage of instances each algorithm finds better or the same solution than MILP-W. In general, MILP-G$^+$ and MILP-GM$^+$ find the better or same solution for almost all 100 and 150 job instances. For 15 job instances, MILP-G had a better or same solution for almost all the instances. For 30 job instances, MILP-G and MILP-G$^+$ had better or the same results for almost 75% of the time. For 50 job instances, all three methods had better or the same instances almost 50% of the time. Figure 9, MILP-CG has the lowest percentage of better or same solution instance.

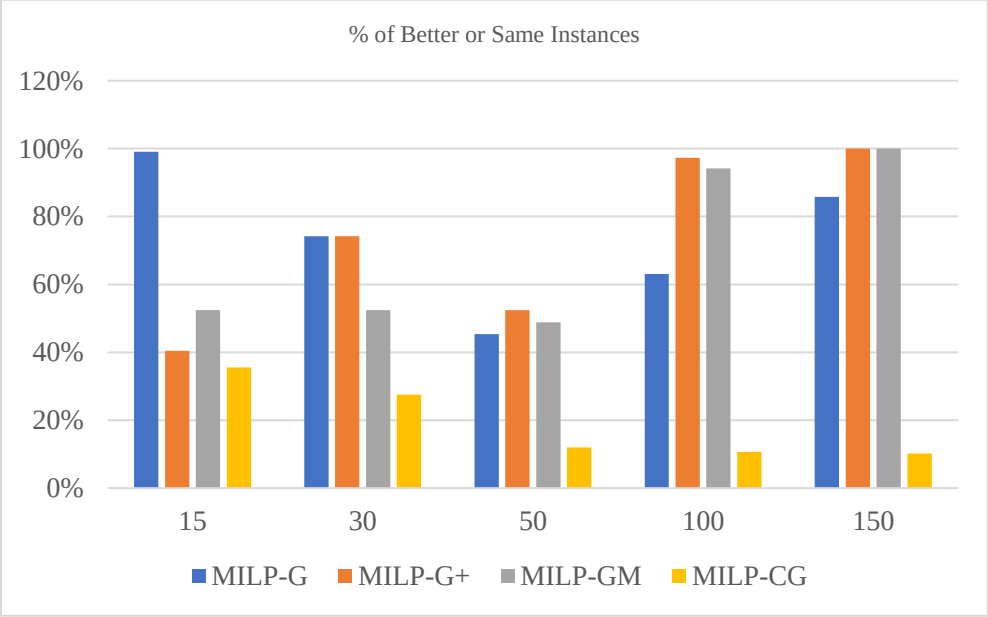
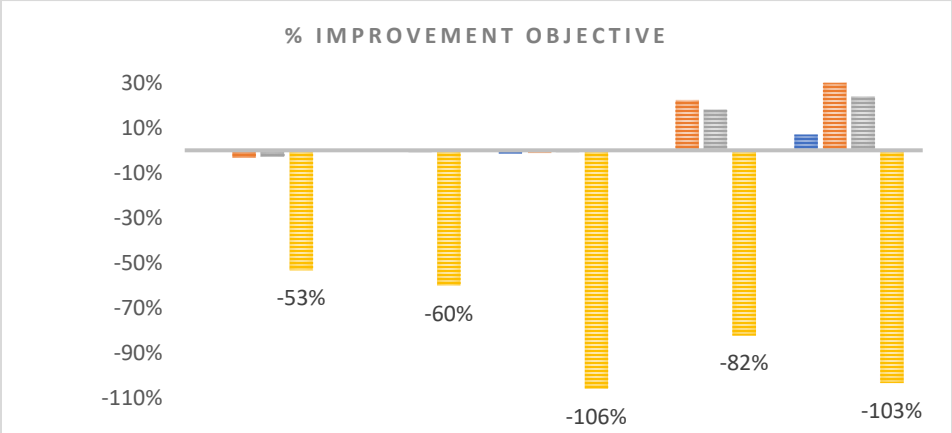


Figure 9. Percentage of a better or same solution

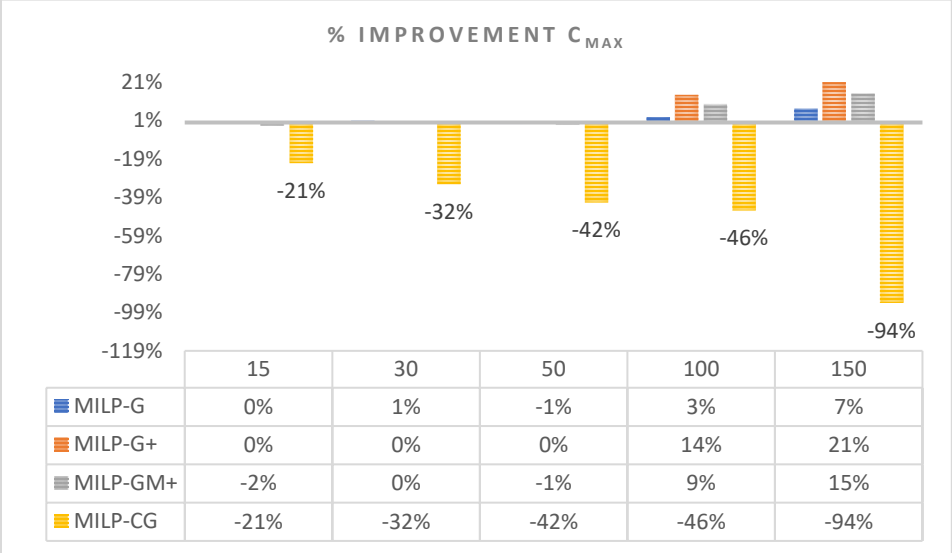
5.2 Percentage Improvement

The percentage improvement is calculated using the equation (42). The comparison of the different formulations using percentage improvement as a metric are summarized in Figures 10, 11, and 12. Figure 10 shows the percentage improvement for the objective function. Figure 6, the % improvement was higher for higher job instances. MILP-G^{+} and MILP-GM^{+} showed higher percentage improvement than MILP-G for 100 and 150 Jobs. For 15 and 50 jobs, the percentage improvement was negative showing MILP-W had a better objective value. For 30 job instances, the percentage improvement was very negligible. Figure 11 and 12 shows the percentage improvement for C_{max} and $T_{max.}$



	15	30	50	100	150
MILP-G	0%	0%	-1%	1%	7%
MILP-G+	-3%	0%	-1%	22%	30%
MILP-GM+	-3%	-1%	-1%	18%	24%
MILP-CG	-53%	-60%	-106%	-82%	-103%

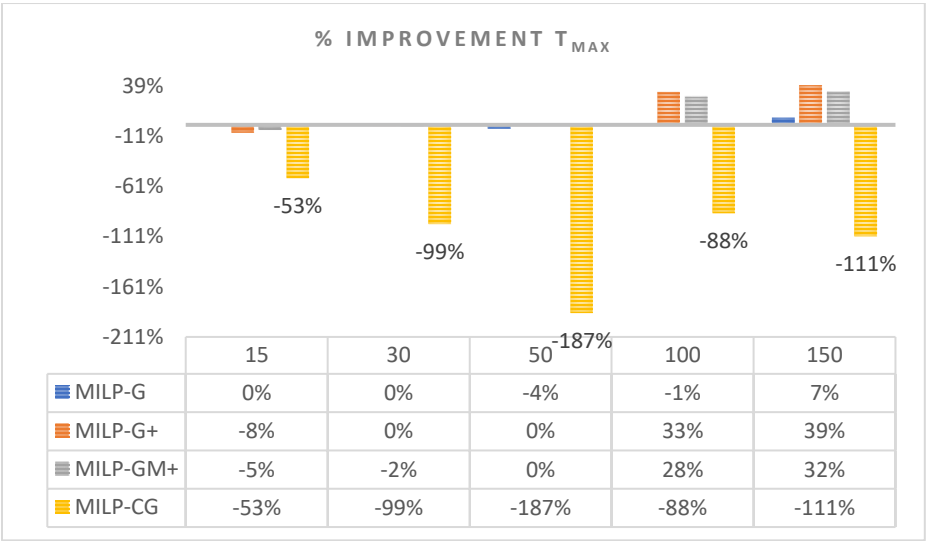
Figure 10. Percent Improvement - Objective



	15	30	50	100	150
MILP-G	0%	1%	-1%	3%	7%
MILP-G+	0%	0%	0%	14%	21%
MILP-GM+	-2%	0%	-1%	9%	15%
MILP-CG	-21%	-32%	-42%	-46%	-94%

Figure 11. Percent Improvement - C_{max}

From Figure 11, the percentage improvement is higher for 100 and 150 jobs. For lower job instances, the MILP-W had a better C_{max} value because the value is negative. From Figure 12, percentage improvement holds good for 100 and 150 jobs. Similarly, the percentage improvement is negative for lower job instances (15,30, and 50). In general, MILP-W had better C_{max} and T_{max} for lower job instances.



	15	30	50	100	150
MILP-G	0%	0%	-4%	-1%	7%
MILP-G+	-8%	0%	0%	33%	39%
MILP-GM+	-5%	-2%	0%	28%	32%
MILP-CG	-53%	-99%	-187%	-88%	-111%

Figure 12. Percent Improvement - T_{max}

MILP-CG has high negative percentage improvement but still has few instances dominating MILP-W. So we plot percentage improvement for the instances where MILP-CG had better solution. Figure 13 shows the percent improvement for the instances were MILP-CG had better solution than MILP-W.

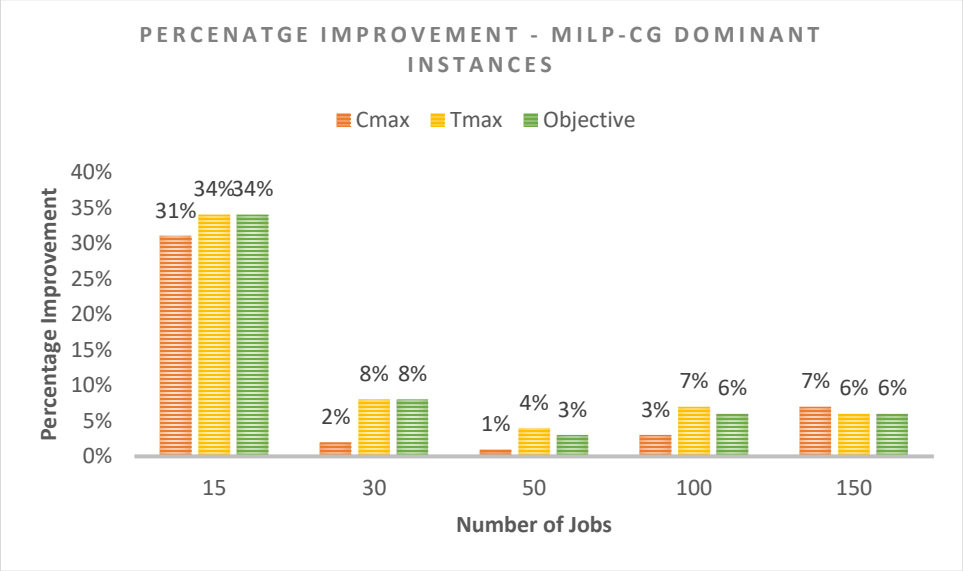


Figure 13. Percent Improvement- MILP-CG Dominant Instances

Figure 13, there is significant improvement interms if C_{max}, T_{max} and objective. So, further research on decomposition of the model will help to improve the solution quality.

5.3 Run time Comparison and Gap Percentage

An important factor to consider is the run time required by CPLEX to solve the different formulations. The average computation times for weighted and goal methods are shown in Figure 14. There is not much of a difference in runtime between MILP-W and MILP-G. The average

runtime of MILP-G^{+} is less for 50, 30, and 100 job instances. MILP-GM^{+} had less time compared to MILP-W and MILP-G. in 15, 30, and 50 job instances. Consequently, it can be concluded that the new formulations not only report good solutions compared to MILP-W, but they also require less computational time. In addition to comparing the formulations with the solution, a comparison between different MILP methods is made using a gap percentage. The gap percentage is calculated using the formula (41). Gap percentage is defined as the percentage difference between the best integer objective and best linear objective, Gap is an indication of how close CPLEX is to find an optimal solution. The smaller the gap, the closer CPLEX is to the optimal value. The more difficult an instance is to solve, the larger the gap value. Figures 15 to 19 shows the average gap vs runtime of different formulations discussed.

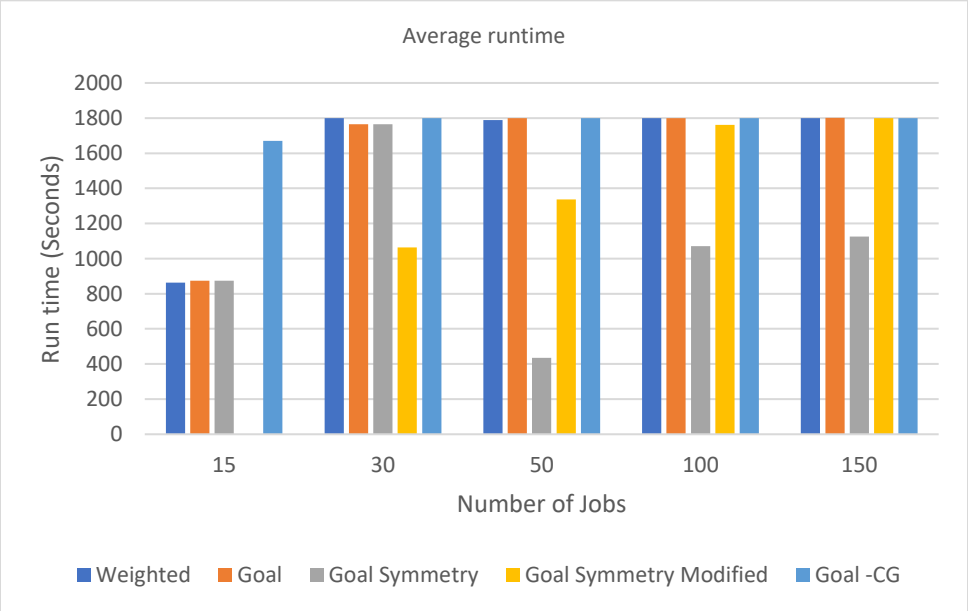


Figure 14. Average run time (in seconds)

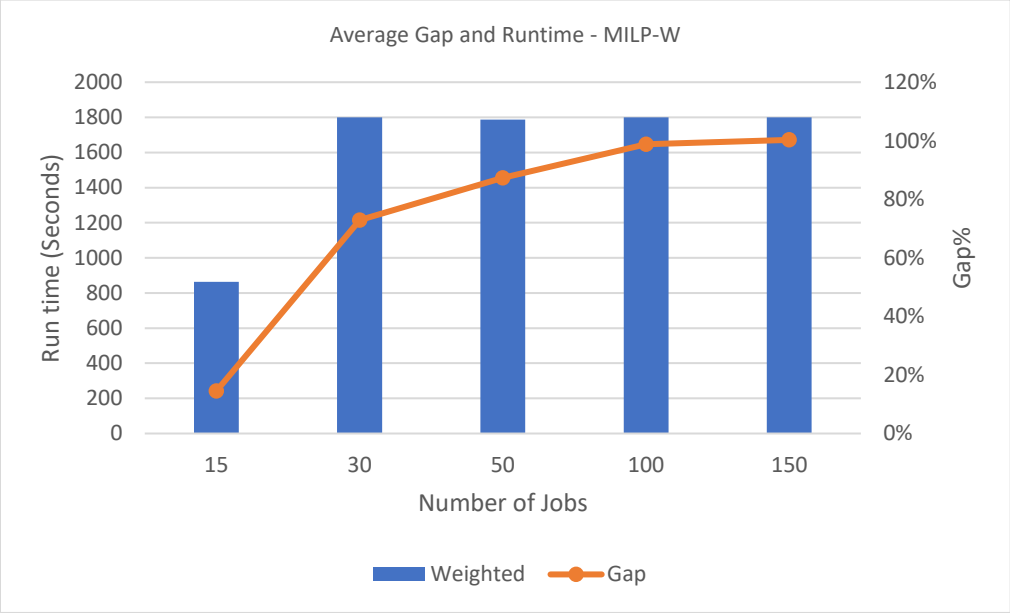


Figure 15. Average Gap vs Runtime-MILP-W

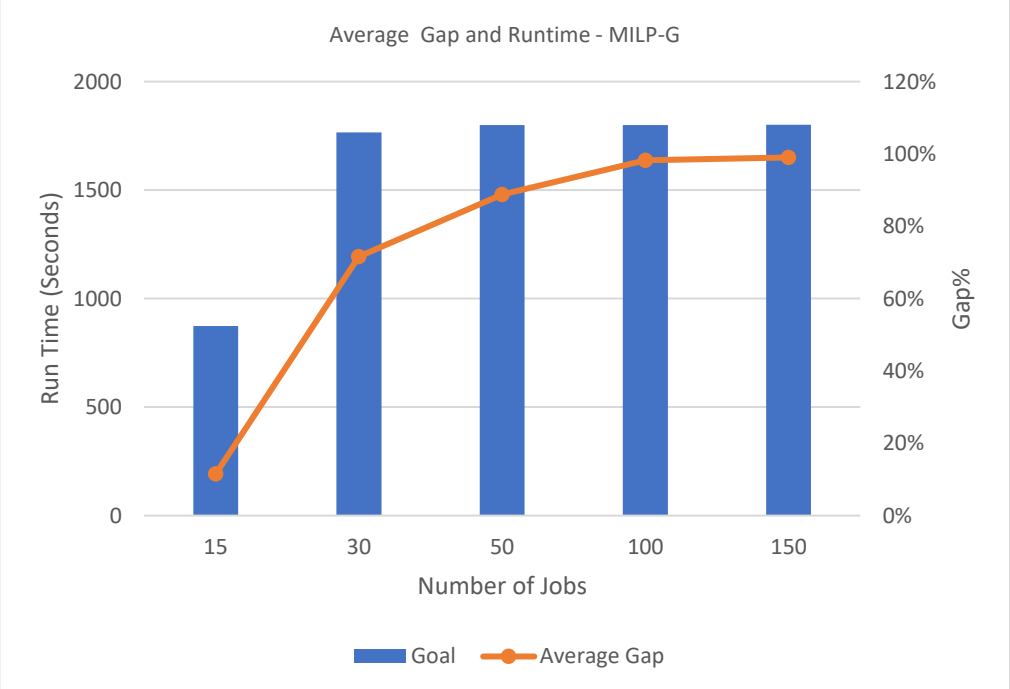


Figure 16. Average Gap vs Runtime - MILP-G

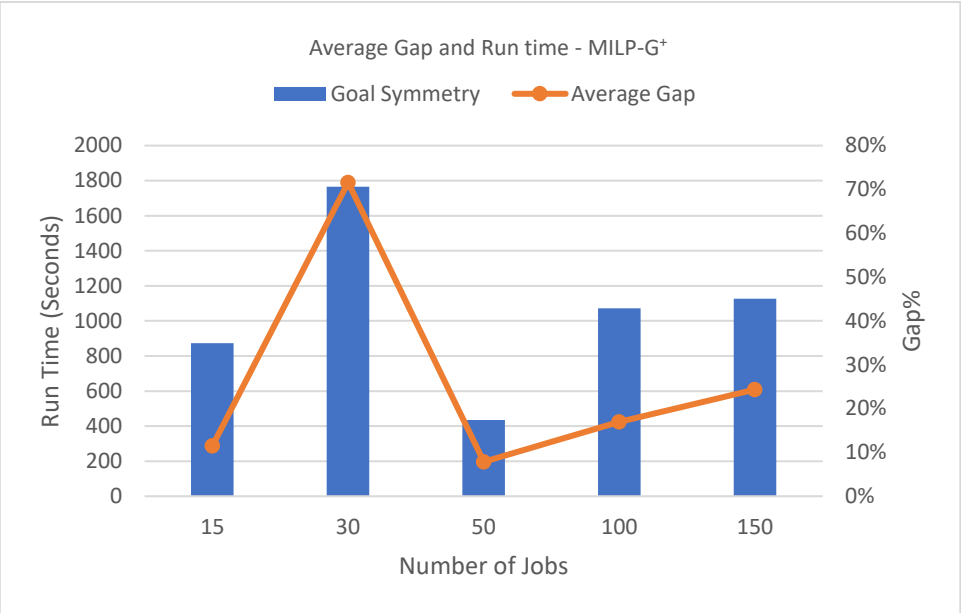


Figure 17.Average Gap vs Runtime - MILP - G^+

There is not much of a difference between MILP-W, MILP-G, and MILP-GM^+ in terms of Average runtime vs Gap. The Gap and runtime are comparatively less for MILP-G^+. As the number of jobs increases, the gap also increases. In Figure 15, 100 and 150 job instances have average gap values 100%, meaning MILP-W is not able to converge to an optimal solution.

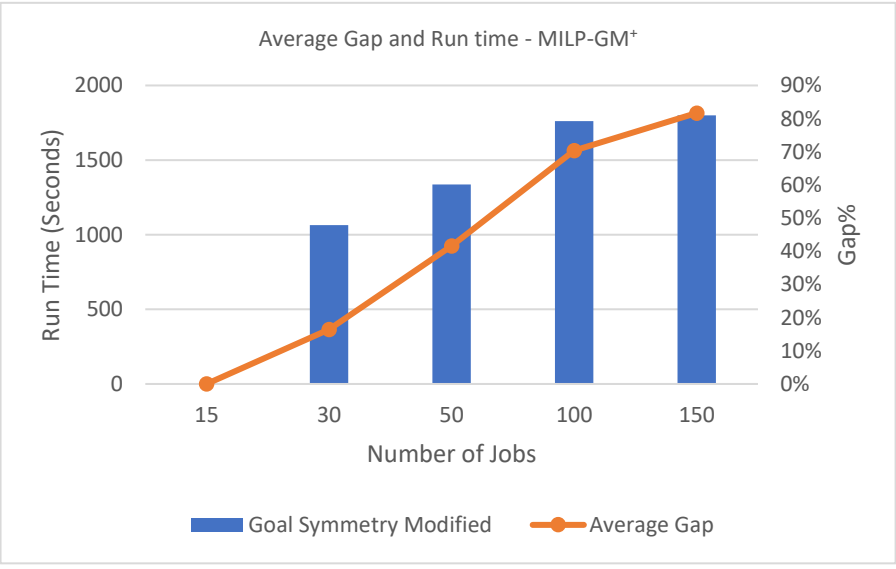


Figure 18. Average Gap vs Runtime - MILP-GM+

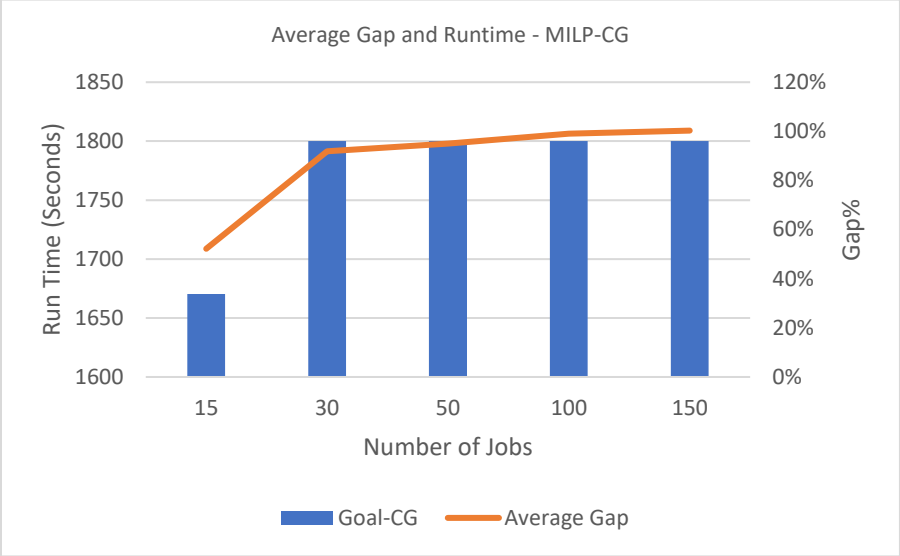


Figure 19. Average Gap vs Runtime - MILP-CG

From figure 19, MILP-CG had the average runtime of 1800 seconds and gap of 100% for jobs 30 to 100. Overall, MILP-G^{+} has comparatively less gap and runtime.

5.4 Pareto Front Example

The Pareto frontier for each MILP can be constructed by graphing the points included in the Pareto set. The x-axis is defined using $C_{max,}$ and the y-axis is defined using T_{max}. The closer the given algorithm's Pareto front is to (0,0), the higher the quality of its solution (Ghrayeb, 2020). An example Pareto front is shown for all jobs instances from figure 20 to figure 24. From figure 20, The Pareto front of MILP-CG is far away from (0,0) when compared to other methods proving that the solution quality of CG depends on the subproblem limit. In figure 20, MILP-G and MILP-W overlap each other because the values are equal. In figure 21 and figure 22, MILP-G, MILP-G^{+}, MILP-W and MILP-GM^{+} concentrated in a single region showing not much of a

difference in solution value. In figure 21-24, the Pareto front of MILP-CG is far away from (0,0) showing the solution quality of column generation is bad.

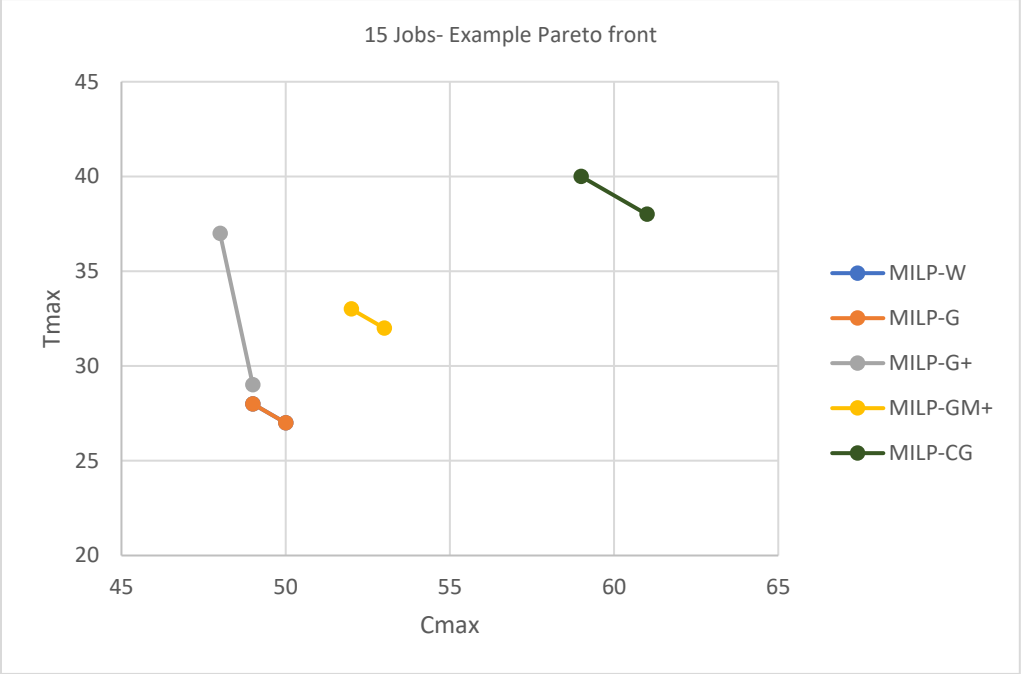


Figure 20. 15 Jobs- Example Pareto front

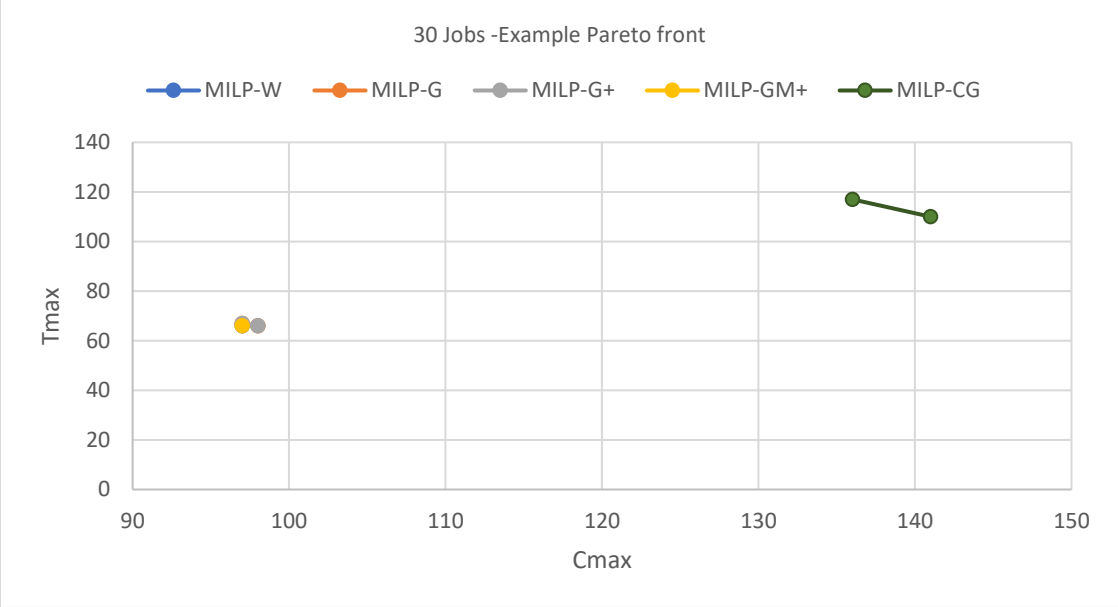


Figure 21. 30 Jobs -Example Pareto front

Figure 22. 50 Jobs- Example Pareto front

From Figures 23 and 24, the Pareto front of MILP-G+ and MILP-GM^{+} is close to (0,0) providing evidence that it has a better solution than any other method. MILP-W is away from (0,0) shows it finds it difficult to solve the problem to optimality for higher job instances. Overall MILP-G+ produces better solution quality than all other methods.

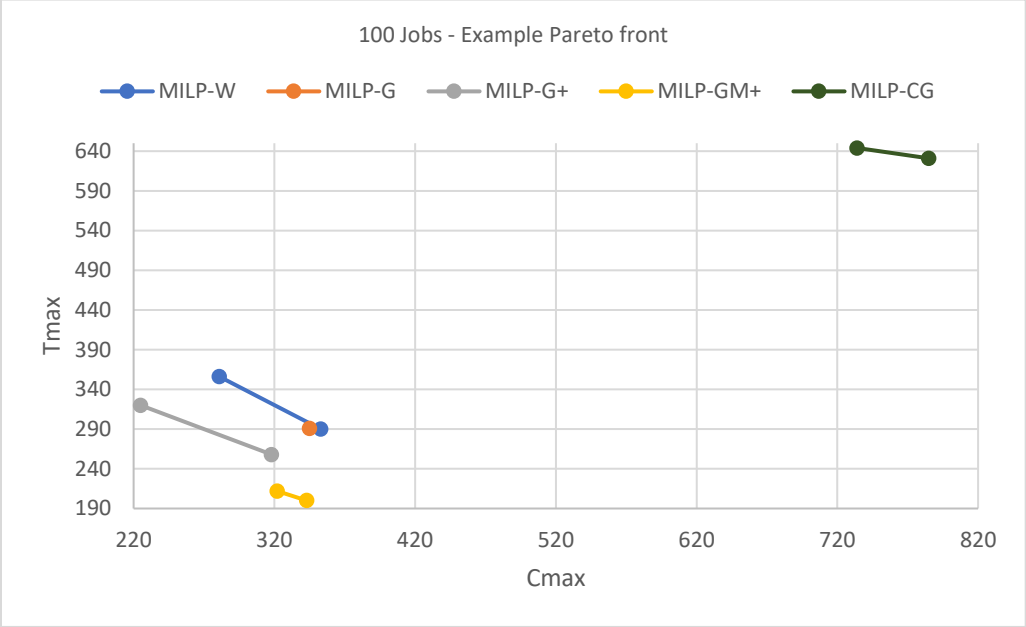


Figure 23. 100 Jobs - Example Pareto front

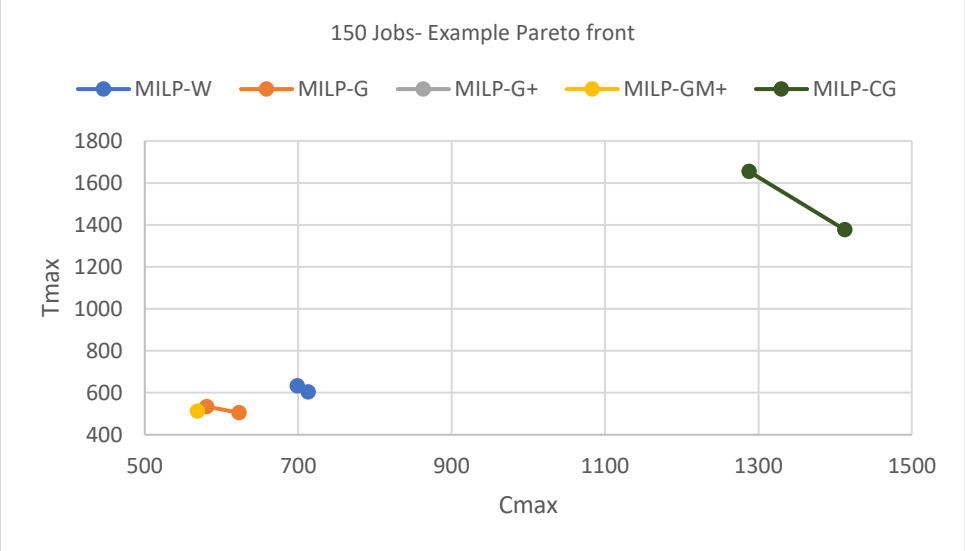


Figure 24. 150 Jobs- Example Pareto front

CHAPTER 6. CONCLUSIONS AND FUTURE WORK

The problem under study can be denoted as $1|p\text{-}batch, s_j, r_j| C_{max}, T_{max}$. The main objective is to minimize the makespan and maximum tardiness such that it does not exceed the size capacity of the machine. MILP-G, MILP-G$^+$, MILP-GM$^+$ was developed and solved using CPLEX. One way to evaluate the proposed formulations is to compare their solution quality with the solution from MILP-W (Ghrayeb, 2020). The different formulations are compared based on their objective values, % gap, run time, and % improvement. The following sections summarize the main findings of this research as well as potential future work.

6.1 Conclusions

MILP-G, MILP-G$^+$, MILP-GM$^+$ was used to solve the generated two hundred and twenty-five problem instance. Based on the comparison with MILP-W, both MILP-G$^+$ and MILP-GM$^+$ dominated on larger problem instances for all values of alpha. The percentage improvement in objective value for MILP-G$^+$ is 30% and 22% for 150 and 100 job instances, respectively. For MILP-GM$^+$ the percentage improvement is 24% and 18% for 150 and 100 job instances, respectively. The results of percentage improvement are summarized in Table 10.

Table 10. Percentage Improvement

% Improvement-Objective Value	Number of Jobs	
	100	**150**
Goal	1%	7%
Goal Symmetry	22%	30%
Goal Symmetry Modified	18%	24%
Column Generation	-82%	-101%

From Table 10, it is evident that MILP-G$^+$ and MILP-GM$^+$ had the better improvement in terms of objective value. MILP-CG had the negative percentage improvement meaning MILP-W had better solution value. Table 11 shows the percentage of times the new formulations found better or the same solution as MILP-W. MILP-G$^+$, MILP-GM$^+$ finds better or the same solution as MILP-W for 100% of the 150-job instances. For 100 job instances the value is 97% and 94% for MILP-G$^+$ and MILP-GM$^+$. On average MILP-G provides better or same solution as MILP-W on 74% of the instances. Looking at the runtime MILP-G$^+$, MILP-GM$^+$ do not have significant differences in runtime between them. The average run time MILP-G$^+$ is less for higher job instances when compared with MILP-W.

Table 11. Number of better or same solution

Number of Better or same solution	Number of Jobs				
	15	30	50	100	150
Goal	99%	74%	45%	63%	86%
Goal Symmetry	40%	74%	52%	97%	100%
Goal Symmetry Modified	52%	52%	49%	94%	100%
Goal -Column Generation	42%	28%	12%	11%	10%

Based on all results, MILP-G$^+$, MILP-GM$^+$ seems to be the best approach for larger job instances (100 and 150). For smaller job instances (15, 30, 50) MILP-W had a better solution than the proposed methods. Depending on the size of the problem at hand, MILP-G$^+$ and MILP-GM$^+$ may be chosen over MILP-W for their high-quality solutions and shorter computational time.

6.2 Future Work

In this research, the weights were provided between two objective functions to solve the proposed problem. Five levels were considered for α and β, from 0 – 1 at intervals of 0.25, so it could be interesting to test smaller intervals of values to see if the Pareto-optimal front changes. Additionally, as discussed in Chapter 2, different solution approaches exist for solving multi-objective formulations. It would be worthwhile to explore methods like the ε – constraint method to see if different, higher quality Pareto-optimal fronts are obtained. Moreover, the Symmetric breaking method can also be tried with MILP-W to compare the runtime and the solutions obtained.

Fuzzy goal programming is an extension of traditional goal programming to solve multi-objective problems with notably defined model parameters in a decision-making environment. The original MOLP is converted into a Fuzzy Multi-Objective Linear Programming (FMOLP) using the piece-wise linear function (Moghaddam et al., 2010).

Although a column generation approach was developed the experimental study did not show no evidence of its superiority over other formulations. A further study on the decomposition approach would be beneficial.

Overall, the new formulations proposed and evaluated show their useful to solve larger problem instances efficiently. This research would benefit schedulers who have the daunting task of scheduling batch processing machines with the constraints discussed in this research.

REFERENCES

Abedi, M., Seidgar, H., Fazlollahtabar, H., & Bijani, R. (2014). Bi-objective optimization for scheduling the identical parallel batch-processing machines with arbitary job size, unequal job release times and capacity limits. *International Journal of Production Research, 53*(6), 1680-1711. doi:10.1080/00207543.2014.952795

Akker, V., Hoogeveen J.A., & Kempem , V. (2010). Using column generation to solve parallel machine scheduling problems with minmax objective functions. *Journal of Scheduling, 15*(6), 801-810. doi:10.1007/s10951-010-0191-z

Beldar, P., & Costa, A. (2018). Single machine batch processing problem with release dates to minimize total completion time. *International Journal of Industrial Engineering Computations*, 331-348. doi:10.5267/j.ijiec.2017.8.003

Bulbul, K., Kaminsky, P., & Yono, C. (2004). Flow shop scheduling with earliness, tardiness, and intermediate inventory holding cost. *Naval Research Logistics, 51*(3), 407-445. doi:10.1002/nav.20000

Chen, Z.-L., & Powell, W. (1999). Solving parallel machine scheduling problems by column generation. *INFORMS Journal on Computing, 11*(1), 78-94. doi:10.1287/ijoc.11.1.78

Damodaran, P., & Velez-Gallego, M. C. (2012). A simulates annealing algorithm to minimize makespan of parallel batch processing machines with unequal job ready times. *Expert Systems with Application, 39*(1), 1451-1458. doi:10.1016/j.eswa.2011.08.029

Deliktas, D., Torkul, O., Ustun, O., & Kiris, S. (2014). Single machine scheduling with sequence-dependent setup times by using AHP and multi-choice goal programming. *International Symposium of the Analytic Hierarchy Process 2014.* Washington D.C. doi:10.13033/isahp.y2014.153

Ghrayeb, L. (2020). *Bi-objective optimization for single batch processing machine.* DeKalb: Northern Illinois University; ProQuest Dissertations & Theses Global. Retrieved from https://www.proquest.com/dissertations-theses/bi-objective-optimization-single-batch-processing/docview/2452101430/se-2?accountid=12846.

Graham, R. L.; Lawler, E. L.; Lenstra, J.K.; Rinnooy Kan, A.H.G. (1979). "Optimization and Approximation in Deterministic Sequencing and Scheduling: a Survey". *Proceedings of the Advanced Research Institute on Discrete Optimization and Systems Applications of the Systems Science Panel of NATO and of the Discrete Optimization Symposium*. Elsevier. 5, 287–326.

Groover, M. (2010). *Fundamentals of modern manufacturing* (4 ed.). John Wiley & Sons.

Jin, M., Liu, X., & Luo, W. (2020). Single-machine parallel-batch scheduling with nonidentical job sizes and rejection. *Mathematics, 8*(2), 258-265. doi:10.3390/math8020258

Kashan, A. H., Karimi, B., & Jolai, F. (2010). An effective hybrid multi-objective genetic algorithm for bi-criteria scheduling on single batch processing machine with non-identical job sizes. *pplications of Artifician Intelligence, 23*(6), 911-922. doi:10.1016/j.engappai.2010.01.031

Kiran, D. (2019). *Production planning and control.* Elsevier.

Kondakci, S. K., & Bekiroglu, T. (1997). Scheduling with bicriteria: total floetime and number of tardy jobs. *International Journal of Production Economics, 53*(1), 91-99. doi:10.1016/S0925-5273(97)00099-6

Lee, S. M., & Jung, H.-J. (1989). A multi-objective production planning model in a flexible manufacturing environment. *International Journal of Production Research, 27*(11), 1981-1992. doi:10.1080/00207548908942668

Li, X., & Wang, L. (2018). Scheduling batch proccessing machine using max-min ant sysytem algorithm improved by loacal search method. *Mathematical Problems in Engineering*, 1-10. doi:10.1155/2018/3124182

Mavrotas, G. (2009). Effective implementation of the e-constraint method in Multi-Objective. *Applied Mathematics and Computation*, 455-465. doi:10.1016/j.amc.2009.03.037

Melouk, S., Damodaran, P., & Chang, P.-Y. (2004). Minimizing makespan for single machine batch processig with non-identical job sizes using simulated annealing. *International Journal of Production Economics, 87*(2), 141-147. doi:10.1016/S0925-5273(03)00092-6

Moghaddam, R. T., Javadi, B., Jolai, F., & Ghodrarnama, A. (2010). The use of fuzzy multi-objective linear programming for solving a multi-objective single machine scheduling problem. *Applied Soft Computing, 10*(3), 919-925. doi:10.1016/j.asoc.2009.10.010

Pei, Z., Zhang, X., Zheng, L., & Wan, M. (2019). A column generation-based approch for prportionate flexible two stage no-wait job shop scheduling. *International Journal of Production Research*, 1-23. doi:10.1080/00207543.2019.1597291

Pinedo, M. (2009). *Planning and scheduling in manifacuring and services* (2 ed.). Springer-Verlag New York. doi:10.1007/978-1-4419-0910-7

Rardin, R. l. (2017). *Optimization in Operations Research.* Hoboken: Pearson Higher Education.

Research and reviews. (2020). *Batch productions*. Retrieved October 05, 2020, from Research and reviews: https://www.rroij.com/scholarly/batch-productions-journals-articles-ppts-list.php#:~:text=There%20are%20several%20advantages%20of,common%20process%20in%20pharmaceutical%20industry.

Rezaeian, J., & Zarook, Y. (2018). An efficient bi-objective genetic algorithm for single batch processing machine scheduling problem with sequence dependent family setup time and non-idnetical job size. *Journal of Optimization in Industrial Engineering, 11*(2), 65-78. doi:10.22094/joie.2018.792.1505

Ronconi, D. P., & Kawamura, M. S. (2010). The single machine earliness and tardiness scheduling problem: lower bounds and a branch and bound algorithm. *Computational and Applied Mathematics, 29*, 107-124.

Sabouni, Y. M., & Jolia, F. (2010). Optimal methods for batch processing problem with makespan and maximum lateness objectives. *Applied Mathematical Modelling, 34*(2), 314-324. doi:10.1016/j.apm.2009.04.007

Selen, W. J., & Hott, D. D. (1986). A mixed-integer goal programming formulation of standard flow-shop scheduling problem. *The Journal of the Operational Research Society, 37*(12), 1121-1128. doi:10.2307/2582302

Singh, R. (2006). *Introduction to basic manufacturing and workshop technology.* New Delhi: New age international limited.

Trindade, R., Bassi de Araújo, O., Costa Fampa, M., & Müller , F. (2018, Jan). Modelling and symmetry breaking in scheduling problems on batch processing machines. *International Journal of Production Research*, 7031-7048. doi:10.1080/00207543.2018.1424371

Wiechman, N., & Damodaran, P. (2015). A column generation approach for scheduling a batch processing machine with makespan objective. *Int. J. Industrial and Systems Engineering, 21*(3), 334-355.

Woo, Y.-B., Kim, B. S., & Moon, I. (2019). Column generation algorithms for a single machine problem with deteriorating jobs and deteroration maintenance activities. *Procedia Manufacturing, 39*, 1119-1128. doi:10.1016/j.promfg.2020.01.358