Taylor & Francis
Taylor & Francis Group

Applied Artificial Intelligence
An International Journal

ISSN: 0883-9514 (Print) 1087-6545 (Online) Journal homepage: https://www.tandfonline.com/loi/uaai20

Problem Specific Variable Selection Rules for Constraint Programming: A Type II Mixed Model Assembly Line Balancing Problem Case

Hacı Mehmet Alakaş & Bilal Toklu

To cite this article: Hacı Mehmet Alakaş & Bilal Toklu (2020) Problem Specific Variable Selection Rules for Constraint Programming: A Type II Mixed Model Assembly Line Balancing Problem Case, Applied Artificial Intelligence, 34:7, 564-584, DOI: 10.1080/08839514.2020.1731782

To link to this article: https://doi.org/10.1080/08839514.2020.1731782

APPLIED ARTIFICIAL INTELLIGENCE
2020, VOL. 34, NO. 7, 564–584
https://doi.org/10.1080/08839514.2020.1731782

Problem Specific Variable Selection Rules for Constraint Programming: A Type II Mixed Model Assembly Line Balancing Problem Case

Hacı Mehmet Alakaş [a] and Bilal Toklu [b]

[a]Faculty of Engineering, Department of Industrial Engineering, Kırıkkale University, Yahşihan/Kırıkkale, Turkey; [b]Faculty of Engineering, Department of Industrial Engineering, Gazi University, Ankara, Turkey

ABSRACT

The main idea of constraint programming (CP) is to determine a solution (or solutions) of a problem assigning values to decision variables satisfying all constraints. Two sub processes, an enumeration strategy and a consistency, run under the constraint programming main algorithm. The enumeration strategy which is managing the order of variables and values to build a search tree and possible solutions is crucial process in CP. In this study problem-based specific variable selection rules are studied on a mixed model assembly line balancing problem. The 18 variable selection rules are generated in three main categories by considering the problem input parameters. These rules are tested with benchmark problems in the literature and experimental results are compared with the results of mathematical model and standard CP algorithm. Also, benchmark problems are run with two CP rules to compare experimental results. In conclusion, experimental results are shown that the outperform rules are listed and also their specifications are defined to guide to researchers who solve optimization problems with CP.

Introduction

Constraint programming (CP) is widely used for solving constraint satisfaction and optimization problems. Researchers have successfully achieved the solution of various problems with CP such as scheduling (Sel et al. 2015; Serra, Nishioka, and Marcellino 2012), manufacturing (Banaszak, Zaremba, and Muszyński 2009; Soto et al. 2012), supply chains (Lee and Lee 2013; Rodrigues and Leandro 2007), rostering (He and Qu 2012; Qu and He 2009), vehicle rooting (Ozfirat and Ozkarahan 2010) and allocation (Bui, Pham, and Deville 2013). Mainly, related problems in CP are modeled and solved as constraint satisfaction problems (CSP). The CSP consists of an n-tuple of variables which are related to their domain di and m-tuple of constraints. The CSP has a solution when all variables take value in their

CONTACT Hacı Mehmet Alakaş hmalagas@kku.edu.tr Faculty of Engineering, Department of Industrial Engineering, Kırıkkale University, Ankara Yolu 7. Km, Yahşihan/Kırıkkale, 71451

domains. The values must satisfy all constraints. Backtracking, branch, and bound algorithm or local searches are generally used to explore and obtain the CSP problem's solution. Two main phases are processed to explore the solution. First one is an enumeration strategy which composed of variable and value selections. The second one is the propagation which is used to filter the variable domains by eliminating the inconsistent values.

As mentioned by Soto et al. (2015), determining of "which variable selection strategy combined with which value selection strategy" is a hard and crucial phase in constraint programming. Instead of using default constraint programming variable or value selection rules, problem-specific rules may be more effective than default rules. Based on this hypothesis, answer the following questions are examined in this study:

(1) Are the variable selection rules based on problem specifications more efficient than CP method rules?
(2) Which types of rules are more efficient in solving the problem?
(3) What are the main characteristics of effective rules?

In this study, problem-based variable selection rules are investigated for type 2 mixed-model assembly line balancing problem (MMALBP). The problem is modeled as the constraint optimization problem and 18 specific variable selection rules that are classified into three main generated categories. The solutions of these models are compared with solutions of a mathematical model, the constraint programming model with the default rule, the constraint programming model with the minimum domain rule and the constraint programming model with the maximum domain rule. Consequently, effective rules that reach a better solution in a shorter time are determined to answer the research questions mentioned above.

The rest of the paper is organized as follows; after the introduction, the constraint programming algorithm is presented in Sect. 2. In Sect. 3, the mixed model assembly line problem is introduced, and variable selection rules of MMALBP are given and explained in Sect.4. Numerical experiments and discussions about experiments are presented in Sect. 4. Conclusions and suggestions for future studies are summarized in Sect. 5.

Background: Constraint Programming (CP)

Constraint programming is an alternative programming technique generated by combining effectiveness in achieving the optimal solution of linear programming and easy definition of logical expressions of computer programming

methods. In constraint programming, difficult definable constraints in linear programming are defined easily using logical expressions. (Apt 2003)

The constraint programming model is defined as a triple notation (X, D, C) where X is a tuple of variables $X = (X_1, X_2, X_3, ..., X_n)$, D is a tuple of the domain $D = (D_1, D_2, D_3, ..., D_n)$ and C is a tuple of constraints $C = (C_1, C_2, C_3, ..., C_n)$. Solving a $CSP\,(X, D, C)$involves assigning values to variables in such a way that all constraints are satisfied. The solution involves a tuple set of variables and their values $A = (X_1, V_1), (X_2, V_2), (X_3, V_3), ..., (X_n, V_n)$.

Constraint Solving

CP problems are generally solved by backtracking based algorithms. The solution procedure is given in Algorithm 1 that includes three main steps, variable selection, value selection, and propagation. In the variable selection step, which variable is selected first to assign a determined value. In the next step of the value selection, the value that is assigned to the selected variable is determined. When a decision variable is assigned as a value, the algorithm recomputes the possible value sets of all its dependent variables in the propagation step.

Algorithm 1. Solve (X; D; C)
While (success) **or** (failure) **do**
    Variable Selection ();
    Value Selection ();
    Propagate_C ();
    If empty domain in future variable **then**
        Shallow Backtrack ();
    End if
    If empty domain in current variable **then**
        Backtrack ();
    End if
End while

Variable selection and value selection in the search algorithm are two key processes to improve the search performance. These two steps are named as an enumeration strategy in common. Actually, if we could develop an efficient enumeration strategy, it can be possible to achieve the best solution for performing fewer backtracks and requiring a shorter solving time. On the other hand, the main problem in the CSP solving is to decide which strategy is better depending on the problem types (Soto et al. 2015).

In recent years, researchers proposed various approaches about enumeration strategies for solving several combinatorial problems. As an example,

Christodoulou and Stamatopoulos (2002) studied crew assignment problem which is a subproblem of crew scheduling problem. They have modelled the problem as a constraint programming and used the rule of selecting the highest flight time of flight first as a variable selection strategy. In the value selection, the crew member was selected that had the smallest flight time until that time. Rousseau et al. (2004) dealt with vehicle routing problem with a time window. They found routes with constraint programming and used problem-based variable selection and value selection rules. Qu and He (2009) proposed constraint programming method that had specific variable and value selection strategy for nurse scheduling. Nurses with heavier workload were selected first in variable selection strategy and nurses were assigned night shift firstly. Siala, Hebrard, and Huguet (2015) modeled a constraint programming model for a car sequencing problem and suggested an enumeration search strategy. They determined rules for variable and value selection in four main categories. These were branching, searching, selecting and combining. Researchers have developed problem-based specific enumeration strategies in these mentioned studies. Some researchers have proposed enumeration strategies to improve constraint programming search procedure (Balafoutis and Stergiou 2010; Crawford et al. 2011; Grimes and Wallace 2007; Soto et al. 2013, 2015; Wallace and Grimes 2008). Based on the literature research about the enumeration strategies generally, researchers study about CP-based rules to improve the solution efficiency. In this study, the variable selection rules that consider problem specifications are investigated in addition to CP model-based rules.

Mixed Model Assembly Line Balancing Problem

Assembly line is a manufacturing process which subparts of products assembled to produce the final products. Workers should perform specified tasks to produce the final products in a series of stations which are connected together by material handling system. Assignment of the given tasks with a set of precedence relations to the stations for optimizing a performance measure is defined as the Assembly Line Balancing problem (ALBP) (Özcan and Toklu 2009). Assembly line balancing problem is classified in terms of the layout, objectives, variability of the task times, etc. ALBP can be classified into four types in terms of the objective functions: Type 1 is the minimization of the number of stations for a given cycle time, Type 2 is the minimization of the cycle time for a given number of stations, Type E (Effectiveness) is the maximization of the effectiveness value, and Type F (Feasible) is the absence of any objective function.

In the early times of assembly lines in production systems, a single model was produced due to high demand and line balancing was relatively easy. Because of diversified customer demands, establishing separate assembly lines for each model are no longer economical. For this reason, different models are produced on the same assembly line and several new situations have emerged. When lot sizes of models are equal to one, this type of assembly line is named as a mixed model line. If lot sizes of models are greater than one, this type of assembly line is named as multi-model line (Figure 1) (Scholl 1999).

The mixed-model assembly line is a type of assembly line that similarly m models are assembled simultaneously on the same line. Each model has its own precedence diagram, but these diagrams can be combined into only one precedence diagram with N tasks. The tasks have processing times that can vary between models. In the combined precedence diagram, if the task time is equal to zero for a model, this task is not processed for this model. The problem place in the ALBPs literature is given in Figure 2. Assumptions about to the problem are as follows:

- Each task must be assigned to a station.
- Serial assembly line layout.
- Task duration is deterministic and known.
- Tasks are not divisible.
- A number of station is fixed and known.
- The models are produced on the line simultaneously.

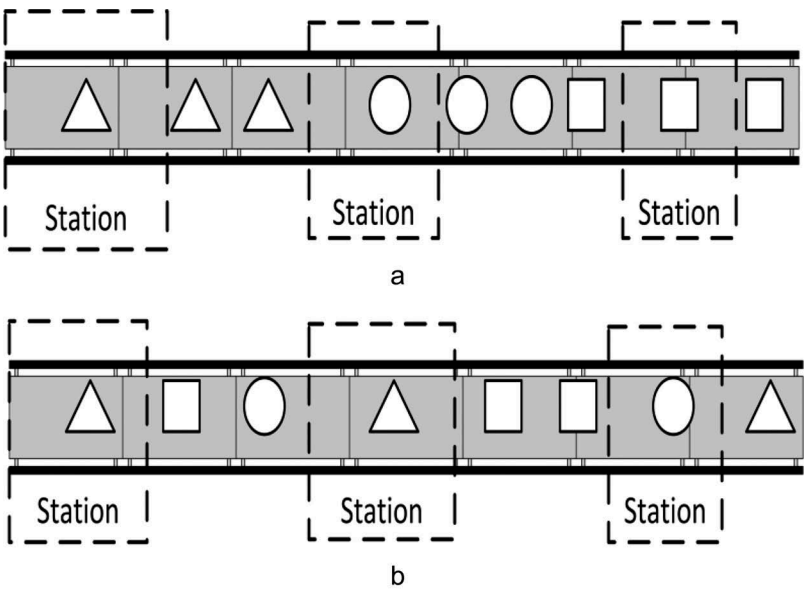

Figure 1. a) Multi model assembly line b) mixed model assembly line.

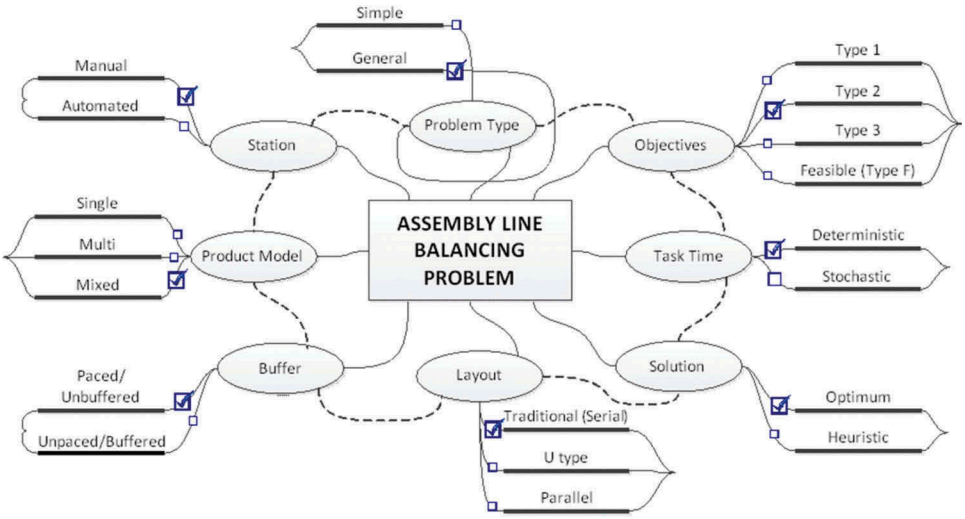

Figure 2. The problem place in the ALBPs literature.

- Precedence constraints are known and must be stable on task-station assignments.
- At least one task must be assigned to each station.

The literature research about MMALBP is restricted by serial, multi/mixed model and one-sided types, although there are numerous studies about the simple ALBPs. Arcus (1965) has firstly addressed the MMALBP. In this study, he proposed the COMSOAL method as a solution and aimed to minimize the station number. He used a cycle time which is weighted by the demand percentages for assigning the tasks. Gokcen and Erel (1998)'s study is the one of the studies that achieved the optimal solution and they modeled the problem as 0–1 integer programming firstly.

The studies in which researchers solved the problem with various meta-heuristics: genetic algorithm (Haq, Rengarajan, and Jayaprakash 2006; Hwang and Katayama 2010; Mamun et al. 2012; Manavizadeh et al. 2012; Rekiek, De Lit, and Delchambre 2000; Simaria and Vilarinho 2004; Venkatesh and Dabade 2008; Yang, Gao, and Sun 2013). In addition to these, simulated annealing (McMullen and Frazier 1998; Mendes et al. 2005), ant colony (McMullen and Tarasewich 2003; Yagmahan 2011), tabu search (Bock 2008), beam search (Matanachai and Yano 2001). And also there are some studies using simulations (McMullen and Frazier 1997; Mendes et al. 2005; Tiacci 2012). The researchers used simulations to find performance values of assignments that are obtained by using other methods. Gokcen and Erel (1997) proposed goal programming and Erel and Gokcen (1999) proposed shortest path algorithm as a solution method.

ALBP is firstly formulated as a CP model by Bockmayr and Pisaruk (2001) and a combined method with integer programming and CP is offered. Pastor, Ferrer, and García (2007) presented a comparative study of the performance of CP and MIP for simple ALBP type 1 and 2 problems. Also, Topaloglu, Salum, and Supciller (2012) have proposed a solution procedure with rule-based CP modeling to solve ALBP. Alağaş, Yüzükırmızı, and Türker (2013) dealt with type-2 ALBP with stochastic process time. They proposed a new algorithm which gives the optimum solution using Constraint Programming and Queueing Network. In their algorithm, the possible combinations are determined by Constraint Programming, and then, the performance measures are evaluated by Queueing Network. They tested the method with several numerical experiments from literature. Öztürk et al. (2013) solved the MMALBP as scheduling problem and aimed to minimize a total completion time of all orders. They suggested a methodology with integrated MIP and CP model to solve task assignment, task sequencing, and model sequencing problems together. Also, they suggested a CP model for flexible assembly line balancing problem with parallel stations (Özturk et al. 2015). Alağaş et al. (2016) suggested CP model for MMABLP-2. The proposed model minimizes the cycle time for a given number of stations. They showed that the performance of the CP model performed well and can be a choice as an alternative solution method. Bukchin and Raviv (2018) established constraint programming models for SALBP-1 and SALBP-2 and showed that it was an effective method by making comparisons. They also proposed the constraint programming model for U-ALBP. Pınarbaşı, Yüzükırmızı, and Toklu (2016) studied variability of task times. They modeled stochastic SALBP-2 problem with constraint programming and also used queuing network methods for performance evaluation.

According to the results of the literature research, most of the studies about MMALBP are addressed the type 1 problem. The number of study about type 2 problems is less than the number of study about type 1 problem. Furthermore, Sivasankaran and Shahabudeen (2014) reached this conclusion in the literature review paper, too.

Mixed Model Assembly Line Balancing Problem Type II Constraint Programming Model

Constraint programming model for MMALBP 2 is developed based on the mathematical model. When we describe the problem as CP,$SN_i$is a decision variable and it can take all positive integer values. Domains of decision variables are restricted by constraint 3 and 4 on CP model. As mentioned before, the solution of the problem is obtained when all of the variables take

values which provide all of the constraints. Constraint programming model is as follows:

Notations of CP model.

$i,\ j$	Task indexes
k	Station index
m	Model index
N	Number of tasks in the combined precedence diagram
M	Number of models
S	Number of station
C	Cycle time
E_i	Earliest station number that task i can be assign
L_i	Earliest station number that task i can be assign
t_{im}	Processing time of task i for model m
$(i,j) \in Pr$	Precedence relationship between tasks. Task i must be finished before task j start.
SN_i:	station number of task i to be assigned.(Decision variable)
Y_{ik}:	= 1, If SN_i take "k" value = 0, otherwise

Objective:

$$Min\, z = \sum_{m=1}^{M} C_m \tag{1}$$

Constraints:

$$SN_i \leq SN_j \quad (i,j) \in \Pr \tag{2}$$

$$SN_i \leq L_i \quad i = 1.....N \tag{3}$$

$$SN_i \geq E_i \quad i = 1.....N \tag{4}$$

$$\exists [SN_i] = k \quad k = 1.....S\, ve\, i = 1.....N \tag{5}$$

$$ct_m \geq \sum_{i=1}^{|T|} \tau_{im} * Y_{ik} \quad k = 1.....S\, ve\, m = 1.....M \tag{6}$$

$$SN_i \in [E_i,\ L_i] \text{ and integer} \tag{7}$$

In this model, with the objective Obj. 1, minimization of the sum of models' cycle time is aimed. Const. (2) guarantees that precedence requirements are satisfied, i.e., if task j predecessor of the task i, task i decision variable SN_i must be equal or small than task j decision variable SN_j. Const. (3) and (4) are defined domains of SN_i decision variables. E_i represents the value of the lowest station number that task i can be assigned in Const. (4). The highest station number task i can be assigned and denoted by the value of L_i in

Const.(3). Const. (5) provides at least one task assign to each station. Const. (6) guarantees that the cycle time is not smaller than any station time. Const. (7) defines the decision variable SN_i which represents the station number of the tasks to be assigned.

E_i and $L_i$are found in equations 8 and 9. In this formulation, C_m indicates the cycle time of model m. Desired cycle time is taken as the cycle time and calculations are made according to this value (Gokcen and Erel 1997).

$$E_i = \max_{m=1,\ldots M} \left[\frac{t_{im} + \sum_{j \in PR_i} t_{jm}}{C_m} \right]^+ \tag{8}$$

$$L_i = \min_{m=1,\ldots M} \left(K + 1 - \left[\frac{t_{im} + \sum_{j \in S_i} t_{jm}}{C_m} \right]^+ \right) \tag{9}$$

Pseudo code of CSP algorithm for MMABLP-2 is given in Algorithm 2:

Algorithm 2*: Solve* Z (SN, D, C), A

Inputs:

Task=N
Station= S
Model=M
Task times for each model and task
Precedence relations of tasks
Assigned task set=AT
Unassigned task set =UT

Run *Domain reduction*
While $UT \neq \emptyset$ **do**
Select SN_i according to A selecting rule
Select Value v in domain SN_i randomly
Run *Propagation*
If $D_j = \emptyset$; $j \in UT$ **then**
Run Shallow *Backtrack*
If $D_j = \emptyset$; i selecting task **then**
Run *Backtrack*
End If
End If
End While
Outputs:
Station number sets of tasks
Station times
Sum of cycle times

Variable Selection Rules for Mixed Model Assembly Line Balancing Problem Type II

In the MMALBP, a branching order of variables is imported for shorter solution time and obtaining good solutions in the search tree. In addition to the general variable selection rules of constraint programming, problem-specific rules can also be used. These rules, which are generated for the MMALBP-2, are grouped into three main categories according to input parameters: task times based, precedence relations based and both task time and precedence relations based rules. Addition to these rules, CP method-specific rules can be used for variable selection. The rules in these groups as follow:

Task Time Based Rules (TTBR): Task times are a primary factor in determining station times because the station time is the sum of times of tasks that are assigned that station. Six rules are generated based on the task times. The rules are generated by task times can be different for each task and changing depending on the models. These rules are as follow:

TTBR I: Maximum task times are found depending on models for each time and these values are recorded in an array whose element number to be equal the number of tasks. Again, the element of the array which has maximum value is selected for branching. $\underset{i=1\ldots N}{Max}\left(\underset{m=1..M}{Min}\ t_{im}\right)$

TTBR II: Minimum task times are found depending on models for each time and these values are recorded in an array which is element number to be equal the number of tasks. The element of the array which has maximum value is selected for branching. $\underset{i=1\ldots N}{Max}\left(\underset{m=1..M}{Min}\ t_{im}\right)$

TTBR III: The average task times are computed for each task depending on models and these values are recorded in an array. The element of the array which has maximum value is selected for branching. $\underset{i=1\ldots N}{Max}\left(\sum_{i=1}^{M} t_{im}\Big/M\right)$

TTBR IV: The average task times are computed for each task depending on models and these values are recorded in an array. The element of the array which has minimum value is selected for branching. $\underset{i=1\ldots N}{Min}\left(\sum_{i=1}^{M} t_{im}\Big/M\right)$

TTBR V: The sum of task times are computed for each task depending on models and these values are recorded in an array. The element of the array which has maximum value is selected for branching. $\underset{i=1\ldots N}{Max}\left(\sum_{m=1}^{M} t_{im}\right)$

TTBR VI: The sum of task times are computed for each task depending on models and these values are recorded in an array. The element of the array which has minimum value is selected for branching. $\underset{i=1\ldots N}{Min}\left(\sum_{m=1}^{M} t_{im}\right)$

Precedence Relations Based Rules (PRBR): The precedence relationships among tasks are important factors for determining station numbers that the tasks can be assigned and can limit these station numbers for each task depending on other tasks. The eight rules in this category are as follow:

PRBR I: All successor numbers of tasks are computed and these values are recorded in an array. The element of the array which has maximum value is selected for branching. $Max(NumAll\text{Successors(i)})\ i = 1\ldots N$

PRBR II: All successor numbers of tasks are computed and these values are recorded in an array. The element of the array which has minimum value is selected for branching. $Min(NumAll\text{Successors(i)})\ i = 1\ldots N$

PRBR III: All predecessor numbers of tasks are computed and these values are recorded in an array. The element of the array which has maximum value is selected for branching. $Max(NumAll\text{Predecessors(i)})\ i = 1\ldots N$

PRBR IV: All predecessor numbers of tasks are computed and these values are recorded in an array. The element of the array which has minimum value is selected for branching. $Min(NumAll\text{Predecessors(i)})\ i = 1\ldots N$

PRBR V: A task which has a maximum E value (computed with Eq. 14) is selected for branching. $\underset{i=1\ldots N}{Max}\ E_i$

PRBR VI: A task which has a minimum E value (computed with Eq. 14) is selected for branching. $\underset{i=1\ldots N}{Min}\ E_i$

PRBR VII: A task which has a maximum L value (computed with Eq. 15) is selected for branching. $\underset{i=1\ldots N}{Max}\ L_i$

PRBR VIII: A task which has a minimum L value (computed with Eq. 15) is selected for branching. $\underset{i=1\ldots N}{Min}\ L_i$

Rules Based on Precedence Relationships and Task Time Together (PR_TTBR): These rules take into account precedence relationships among tasks and task time together. Especially, task times which can change model by model are used for generating rules. The rules in this category are as follow:

PR_TTBR I: For each task, sum of its own time and all successor task times of the task is computed model by model. Again for each task, the maximum value in the computed values is selected and record in an array. In the last step, the element of the array which has maximum value is selected for branching. $\underset{j=1\ldots N}{Max}\left(\underset{m=1\ldots M}{Max}\left(t_{jm} + \sum_{i\in AllSucc(j)} t_{im}\right)\right)$

PR_TTBR II: For each task, sum of its own time and all successor task times of the task is computed model by model. Again for each task, the maximum value in the computed values is selected and record in an array. In the last step, the element of the array which has maximum value is selected. $\underset{j=1\ldots N}{Max}\left(\underset{m=1\ldots M}{Min}\left(t_{jm}+\sum_{i\in Allsucc(j)}t_{im}\right)\right)$

PR_TTBR III: For each task, sum of its own time and all predecessor task times of the task is computed model by model. Again for each task, the maximum value in the computed values is selected and record in an array. In the last step, the element of the array which has maximum value is selected for branching. $\underset{j=1\ldots N}{Max}\left(\underset{m=1\ldots M}{Max}\left(t_{jm}+\sum_{i\in Allpred(j)}t_{im}\right)\right)$

PR_TTBR IV: For each task, sum of its own time and all predecessor task times of the task is computed model by model. Again for each task, the maximum value in the computed values is selected and record in an array. In the last step, the element of the array which has maximum value is selected. $\underset{j=1\ldots N}{Max}\left(\underset{m=1\ldots M}{Min}\left(t_{jm}+\sum_{i\in Allpred(j)}t_{im}\right)\right)$

Constraint Programming Rules: These rules are generated using features of the constraint programming model. The rules in this group as follow:

CP I: The task which has a maximum domain size is selected for branching.
CP II: The task which has a minimum domain size is selected for branching.

Computational Experiments

In order to investigate the advantages of CP rules in section 4, experiments are made with 12 test problems from the literature. The results are compared with a mathematical model and standard constraint programming model of MMABLP 2. The comparisons are made on the basis of the rules categories. The mathematical and constraint programming models are solved with ILOG CP/Cplex 12.6 and each of the models is run for the each instance with an hour time limit. Six problem instances consisting of 28 to 297 tasks and 2 to 4 models are selected from the literature. Task times are different for each instance. In Table 1 include a list of the model, task and station numbers for each instance.

Results of Computational Experiments

The experimental results are discussed separately according to the rule categories.

Task Time Based Rules Results and Discussions: Results of TTBR experiments are given in Table 2 According to these results, when MP and TTBRs

Table 1. Experimental sets.

	# Model	# Task	# Station
Heskiaoff	2	28	8
Heskiaoff	3	28	8
Sawyer	2	30	8
Sawyer	3	30	8
Kilbridge	2	45	12
Kilbridge	3	45	12
Tongue	2	70	20
Tongue	3	70	20
WeeMag	2	75	20
WeeMag	4	75	20
Scholl	2	297	45
Scholl	4	297	45

solution results are compared, TTBR rules generally reach the better solution in a shorter time. TTBR I, II, III and V rules are efficient to find better solutions but TTBR IV and VI are not. Same or better solutions are obtained 10 of 12 experiment results with TTBR 1, 2, 3 and 5 rules. When standard CP solution results are compared with TTBRs results, all rule-based CP models reach a better solution (except TTBR 4 and 6 results of Tongue_2 problem). Consequently, task time-based rules are efficient to solve the MMALBP-2.

Precedence Relations Based Rules (PRBR): Results of PRBR experiments are given in Table 3. As the results in Table 3. CP models with precedence relations based rules are insufficient to solve the MMALBP-2. Although these CP models can reach better results in some experiments, MP model performs better most of the experiments. Besides, CP models with PRBRs generally reach same or better results than standard CP model except Scholl_2, Scholl_4, and Tongue_2 problems. For these reasons, these rules must not be used for MMALBP 2.

Rules Based on Precedence Relationships and Task Time Together (PR_TTBR): Results of PR_TTBR experiments are given in Table 4. These rules are established combining the precedence relations and task times together. PR_TTBR I and II rules at five experiments and PR_TTBR III and IV rules at four experiments cannot be reached a better solution. PR_TTBRs generally find the best solutions in long CPU times. Therefore, these rules must not be used to solve the MMALBP 2.

Constraint Programming Rules: CP rules experimental results are given in Table 5. MP model performance is better than CP I rule model performance. But CP II rule model performance generally better than MP model performance. Better results are obtained at 9 of 12 experiments with CP II rule model in shorter time. For these two models, only CP II model can be used to solve the MMALBP 2.

Discussions

The experimental results can be summarized as follow:

Table 2. TTBR rules solution results.

	MP		Standard CP		TTBR I		TTBR II		TTBR III	
	Best Solution	CPU time	Best Solution	CPU time	Best Solution	CPU time	Best Solution	CPU time	Best Solution	CPU time
Heskiaoff 2	488	610	490	42	488	43	488	128	488	452
Heskiaoff 3	664	2	665	180	664	11	664	18	664	8
Sawyer 2	367	0.7	367	7	367	0.35	367	0.4	367	0.2
Sawyer 3	658	2.5	658	17	658	0.45	658	0.8	658	0.4
Kilbridge 2	375	460	376	1040	375	11	375	3	375	98
Kilbridge 3	597	240	602	123	597	3590	598	61	598	96
Tongue 2	403	1460	400	2980	397	25	399	13	398	68
Tongue 3	615	100	612	524	603	190	600	105	604	265
WeeMag 2	154	235	159	28	156	5	155	53	155	437
WeeMag 4	311	290	322	2240	312	304	311	186	311	488
Scholl 2	4242	3550	3199	2918	3170	1438	3165	366	3162	1616
Scholl 4	8021	2150	6495	2659	6396	1551	6388	1300	6393	777
	MP		Standard CP		TTBR IV		TTBR V		TTBR VI	
	Best Solution	CPU time	Best Solution	CPU time	Best Solution	CPU time	Best Solution	CPU time	Best Solution	CPU time
Heskiaoff 2	488	610	490	42	489	72	488	472	489	230
Heskiaoff 3	664	2	665	180	665	775	664	9	664	129
Sawyer 2	367	0.7	367	7	367	3.2	367	0.15	367	3
Sawyer 3	658	2.5	658	17	658	2.5	658	0.3	658	2.5
Kilbridge 2	375	460	376	1040	376	325	375	64	376	120
Kilbridge 3	597	240	602	123	600	555	598	90	602	245
Tongue 2	403	1460	400	2980	406	10	400	26	405	118
Tongue 3	615	100	612	524	608	222	604	248	608	222
WeeMag 2	154	235	159	28	158	271	156	10	158	17
WeeMag 4	311	290	322	2240	315	28	310	197	315	350
Scholl 2	4242	3550	3199	2918	3198	323	3168	1700	3199	182
Scholl 4	8021	2150	6495	2659	6447	1702	6401	1086	6450	600

Table 3. PRBR rules solution results.

	MP		Standard CP		PRBR I		PRBR II		PRBR III		PRBR IV	
	Best Solution	CPU time	Best Solution	CPU time	Best Solution	CPU time	Best Solution	CPU time	Best Solution	CPU time	Best Solution	CPU time
Heskiaoff 2	488	610	490	42	489	224	488	508	489	41	489	27
Heskiaoff 3	664	2	665	180	664	32	664	240	664	130	664	158
Sawyer 2	367	0.7	367	7	367	1.3	367	0.95	367	0.3	367	1.6
Sawyer 3	658	2.5	658	17	658	1.7	658	2.5	658	3	658	8
Kilbridge 2	375	460	376	1040	376	180	376	115	376	8	376	105
Kilbridge 3	597	240	602	123	599	440	599	523	600	170	599	123
Tongue 2	403	1460	400	2980	400	27	404	65	406	557	401	555
Tongue 3	615	100	612	524	603	425	610	105	609	87	606	38
WeeMag 2	154	235	159	28	158	356	158	343	156	42	160	33
WeeMag 4	311	290	322	2240	314	376	314	310	312	285	312	172
Scholl 2	4242	3550	3199	2918	3277	931	3260	332	3254	1073	3286	2504
Scholl 4	8021	2150	6495	2659	6651	563	6516	1592	6487	687	6658	1146
	MP		Standard CP		PRBR V		PRBR VI		PRBR VII		PRBR VIII	
	Best Solution	CPU time	Best Solution	CPU time	Best Solution	CPU time	Best Solution	CPU time	Best Solution	CPU time	Best Solution	CPU time
Heskiaoff 2	488	610	490	42	489	63	489	4	489	9	489	88
Heskiaoff 3	664	2	665	180	664	10	664	44	664	61	664	367
Sawyer 2	367	0.7	367	7	367	0.6	367	0.6	367	0.5	367	0.5
Sawyer 3	658	2.5	658	17	658	3	658	2.5	658	0.4	658	1
Kilbridge 2	375	460	376	1040	376	50	377	13	375	180	377	25
Kilbridge 3	597	240	602	123	600	546	599	100	599	70	601	170
Tongue 2	403	1460	400	2980	405	210	400	175	404	25	404	58
Tongue 3	615	100	612	524	609	50	603	125	604	230	608	225
WeeMag 2	154	235	159	28	158	27	159	428	156	54	158	72
WeeMag 4	311	290	322	2240	313	482	316	194	313	313	314	174
Scholl 2	4242	3550	3199	2918	3199	1237	3289	511	3201	1056	3258	386
Scholl 4	8021	2150	6495	2659	6507	1200	6646	1453	6409	980	6639	1035

Table 4. PR_TTBR rules solution results.

	MP		Standard CP		PR_TTBR I		PR_TTBR II		PR_TTBR III		PR_TTBR IV	
	Best Solution	CPU time	Best Solution	CPU time	Best Solution	CPU time	Best Solution	CPU time	Best Solution	CPU time	Best Solution	CPU time
Heskiaoff 2	488	610	490	42	489	16	489	9	488	505	489	7
Heskiaoff 3	664	2	665	180	664	1	664	27	664	53	664	51
Sawyer 2	367	0.7	367	7	367	1.69	367	0.53	367	0.38	367	0.36
Sawyer 3	658	2.5	658	17	658	9	658	7	658	25	658	5
Kilbridge 2	375	460	376	1040	376	568	376	180	375	528	375	65
Kilbridge 3	597	240	602	123	599	350	599	105	599	358	599	17
Tongue 2	403	1460	400	2980	400	145	400	139	404	7	400	565
Tongue 3	615	100	612	524	601	380	600	325	609	73	607	529
WeeMag 2	154	235	159	28	156	219	157	476	156	103	156	32
WeeMag 4	311	290	322	2240	313	544	314	62	312	323	311	433
Scholl 2	4242	3550	3199	2918	3304	1307	3282	200	3276	210	3242	725
Scholl 4	8021	2150	6495	2659	6642	477	6636	1264	6535	893	6487	1089

Table 5. CP rules solution results.

	MP		Standard CP		CP I		CP II	
	Best Solution	CPU time	Best Solution	CPU time	Best Solution	CPU time	Best Solution	CPU time
Heskiaoff 2	488	610	490	42	489	189	488	362
Heskiaoff 3	664	2	665	180	664	324	664	56
Sawyer 2	367	0,7	367	7	367	8,5	367	0,5
Sawyer 3	658	2,5	658	17	658	3,5	658	1,2
Kilbridge 2	375	460	376	1040	376	120	375	89
Kilbridge 3	597	240	602	123	601	340	598	220
Tongue 2	403	1460	400	2980	404	335	400	15
Tongue 3	615	100	612	524	612	61	601	85
WeeMag 2	154	235	159	28	157	74	156	18
WeeMag 4	311	290	322	2240	317	358	312	379
Scholl 2	4242	3550	3199	918	3278	1556	3175	1451
Scholl 4	8021	2150	6495	2659	6470	1634	6418	1082

(A) The rules in TTBR and CP rule categories are efficient to solve the MMALBP 2. Performances of the rules in TTBR and CP categories are better due to obtaining better results and achieve these results in shorter CPU time than the other rule categories.

(B) When the rules of TTBR and CP categories are analyzed, TTBR I, II, III, V and CP II rules are more efficient than others. Especially, these rules perform well to solve large-scale problems that have number of task, model and station number (Scholl 2 and Scholl 4, etc.).

(C) MP model and CP models results are compared based on the number of tasks of the experiments. The experiments that have 28 and 30 tasks are solved in shorter CPU times and optimum solutions are obtained with MP model. MP and CP models perform closer in 45 tasks experiments. When problem sizes are larger, e.g., 70 and 75 tasks experiments, CP models reach better results than MP model. These better results are obtained with CP models in shorter CPU times. The largest problems in experiments are Scholl 2 and Scholl 4 problems that have 297 tasks. Better results are obtained with CP models in shorter CPU times. According to all these results, when a problem size is larger, smaller cycle times are obtained with CP models.

(D) The rules that select the variable considering maximum value of selecting parameters reach smaller cycle times than the rules considering minimum values of selecting parameters (except minimum domain rule). So that, while a rule is generated for MMALBP 2, a variable that selecting parameter is maximum is selected firstly.

Conclusion

The variable and value ordering are a key element in CP and are named together enumeration strategy. These steps are responsible for selecting

a variable that search algorithm branches firstly from and which value assign to this variable. Rules can be generated based on problem characteristics or CP algorithm specifications. In this study, problem-based variable selection rules and their impacts on problem solutions are investigated.

In this study, The MMALBP 2 is considered to give answers to main questions that are mentioned in the introduction. As the result of computational experiments, the rules in task time-based category are effective and solve the problem in shorter time. The rules in other two categories are not more effective than CP rules.

The common feature of the effective rules (mentioned in remark B) is that these rules are related directly to the objective function. In MMALBP 2, the objective function is the minimization of cycle time which is the maximum value of station times. Station times are calculated depending on task times. As seen in computational experiments, the rules in task time-based rules category are produced better solution than other rules category. Finally, according to the results of the experimental study, better solutions can be obtained to select a variable firstly that has a maximum selecting parameter.

A straightforward direction for future work is to determine value selection rules based on problem specifications. Variable selection rules can be studied for other problem types (scheduling, vehicle routing, production planning, etc.) based on problem specifications taking into consideration remarks of this study.

ORCID

Hacı Mehmet Alakaş http://orcid.org/0000-0002-9874-7588
Bilal Toklu http://orcid.org/0000-0001-5094-1501

References

Alağaş, H. M., M. Pınarbaşı, M. Yüzükırmızı, and B. Toklu. 2016. Karma modelli tip-2 montaj hattı dengeleme problemi için bir kısıt programlama modeli. *Pamukkale Üniversitesi Mühendislik Bilimleri Dergisi* 22 (4):340–348.

Alağaş, H. M., M. Yüzükırmızı, and A. K. Türker. 2013. Stokastik Montaj Hatlarinin Kisit Programlama Ve Kapali Kuyruk Ağlari Ile Dengelenmesi. *Gazi Üniversitesi Mühendislik-Mimarlık Fakültesi Dergisi* 28 (2), 231-240.

Apt, K. 2003. *Principles of constraint programming*. Cambridge University Press, Amsterdam, The Netherlands.

Arcus, A. L. 1965. A computer method of sequencing operations for assembly lines. *International Journal of Production Research* 4 (4):259–77. doi:10.1080/00207546508919982.

Balafoutis, T., and K. Stergiou. 2010. Evaluating and improving modern variable and revision ordering strategies in CSPs. *Fundamenta Informaticae* 102 (3–4):229–61. doi:10.3233/FI-2010-307.

Banaszak, Z. A., M. B. Zaremba, and W. Muszyński. 2009. Constraint programming for project-driven manufacturing. *International Journal of Production Economics* 120 (2):463–75. doi:10.1016/j.ijpe.2008.12.016.

Bock, S. 2008. Using distributed search methods for balancing mixed-model assembly lines in the automotive industry. *Or Spectrum* 30 (3):551-578. doi:10.1007/s00291-006-0069-9.

Bockmayr, A., and N. Pisaruk. 2001. Solving assembly line balancing problems by combining IP and CP. *arXiv Preprint Cs/0106002*.

Bui, Q. T., Q. D. Pham, and Y. Deville. 2013. *Solving the agricultural land allocation problem by constraint-based local search*. Paper presented at International Conference on Principle and Practice of Constraint Programming, Uppsala, Sweden.

Bukchin, Y., and T. Raviv. 2018. Constraint programming for solving various assembly line balancing problems. *Omega* 78:57–68. doi:10.1016/j.omega.2017.06.008.

Christodoulou, G., and P. Stamatopoulos. 2002. *Crew assignment by constraint logic programming*. Paper presented at proceedings of the 2nd hellenic conference on artificial intelligence SETN-2002, Thessaloniki, Greece.

Crawford, B., R. Soto, C. Castro, and E. Monfroy. 2011. A hyperheuristic approach for dynamic enumeration strategy selection in constraint satisfaction. Paper presented at the International Work-Conference on the Interplay Between Natural and Artificial Computation, La Palma, Canary Islands.

Erel, E., and H. Gokcen. 1999. Shortest-route formulation of mixed-model assembly line balancing problem. *European Journal of Operational Research* 116 (1):194–204. doi:10.1016/S0377-2217(98)00115-5.

Gokcen, H., and E. Erel. 1997. A goal programming approach to mixed-model assembly line balancing problem. *International Journal of Production Economics* 48 (2):177–85. doi:10.1016/S0925-5273(96)00069-2.

Gokcen, H., and E. Erel. 1998. Binary integer formulation for mixed-model assembly line balancing problem. *Computers & Industrial Engineering* 34 (2):451–61. doi:10.1016/S0360-8352(97)00142-3.

Grimes, D., and R. J. Wallace. 2007. Sampling strategies and variable selection in weighted degree heuristics. In *Principles and practice of constraint programming–CP*. Springer, 831-838, Rhoda Island, USA.

Haq, A. N., K. Rengarajan, and J. Jayaprakash. 2006. A hybrid genetic algorithm approach to mixed-model assembly line balancing. *International Journal of Advanced Manufacturing Technology* 28 (3–4):337–341. doi:10.1007/s00170-004-2373-3.

He, F., and R. Qu. 2012. A constraint programming based column generation approach to nurse rostering problems. *Computers & Operations Research* 39 (12):3331–43. doi:10.1016/j.cor.2012.04.018.

Hwang, R., and H. Katayama. 2010. Integrated procedure of balancing and sequencing for mixed-model assembly lines: A multi-objective evolutionary approach. *International Journal of Production Research* 48 (21):6417–41. doi:10.1080/00207540903289755.

Lee, J. E., and K. D. Lee. 2013. Modeling and optimization of closed-loop supply chain considering order or next arrival of goods. *International Journal of Innovative Computing, Information and Control* 9 (9):3639–54.

Mamun, A. A., A. A. Khaled, S. M. Ali, and M. M. Chowdhury. 2012. A heuristic approach for balancing mixed-model assembly line of type I using genetic algorithm. *International Journal of Production Research* 50 (18):5106–16. doi:10.1080/00207543.2011.643830.

Manavizadeh, N., M. Rabbani, D. Moshtaghi, and F. Jolai. 2012. Mixed-model assembly line balancing in the make-to-order and stochastic environment using multi-objective evolutionary algorithms. *Expert Systems with Applications* 39 (15):12026–31. doi:10.1016/j.eswa.2012.03.044.

Matanachai, S., and C. A. Yano. 2001. Balancing mixed-model assembly lines to reduce work overload. *IIE Transactions* 33 (1):29–42. doi:10.1080/07408170108936804.

McMullen, P. R., and G. V. Frazier. 1997. A heuristic for solving mixed-model line balancing problems with stochastic task durations and parallel stations. *International Journal of Production Economics* 51 (3):177–90. doi:10.1016/S0925-5273(97)00048-0.

McMullen, P. R., and G. V. Frazier. 1998. Using simulated annealing to solve a multiobjective assembly line balancing problem with parallel workstations. *International Journal of Production Research* 36 (10):2717–41. doi:10.1080/002075498192454.

McMullen, P. R., and P. Tarasewich. 2003. Using ant techniques to solve the assembly line balancing problem. *Iie Transactions* 35 (7):605–17. doi:10.1080/07408170304354.

Mendes, A. R., A. L. Ramos, A. S. Simaria, and P. M. Vilarinho. 2005. Combining heuristic procedures and simulation models for balancing a PC camera assembly line. *Computers & Industrial Engineering* 49 (3):413–31. doi:10.1016/j.cie.2005.07.003.

Özcan, U., and B. Toklu. 2009. Balancing of mixed-model two-sided assembly lines. *Computers & Industrial Engineering* 57 (1):217–27. doi:10.1016/j.cie.2008.11.012.

Ozfirat, P. M., and I. Ozkarahan. 2010. A constraint programming heuristic for a heterogeneous vehicle routing problem with split deliveries. *Applied Artificial Intelligence* 24 (4):277–94. doi:10.1080/08839511003715196.

Öztürk, C., S. Tunalı, B. Hnich, and M. A. Örnek. 2013. Balancing and scheduling of flexible mixed model assembly lines. *Constraints* 18 (3):434–69. doi:10.1007/s10601-013-9142-6.

Özturk, C., S. Tunali, B. Hnich, and A. Ornek. 2015. Cyclic scheduling of flexible mixed model assembly lines with parallel stations. *Journal of Manufacturing Systems* 36:147–58. doi:10.1016/j.jmsy.2015.05.004.

Pastor, R., L. Ferrer, and A. García. 2007. Evaluating optimization models to solve SALBP. In *Computational Science and Its Applications–ICCSA*. Springer, 791-803, Kuala Lumpur, Malaysia.

Pınarbaşı, M., M. Yüzükırmızı, and B. Toklu. 2016. Variability modelling and balancing of stochastic assembly lines. *International Journal of Production Research* 54 (19):5761–82. doi:10.1080/00207543.2016.1177236.

Qu, R., and F. He. 2009. A hybrid constraint programming approach for nurse rostering problems. In *Applications and innovations in intelligent systems XVI*. Springer, 211-224, London, United Kingdom.

Rekiek, B., P. De Lit, and A. Delchambre. 2000. Designing mixed-product assembly lines. *Ieee Transactions on Robotics and Automation* 16 (3):268–80. doi:10.1109/70.850645.

Rodrigues, L. C. A., and M. Leandro. 2007. Enhancing supply chain decisions using constraint programming: A case study. In *MICAI 2007 advances in artificial intelligence*, 1110-1121, Aguascalientes, Mexico.

Rousseau, L. M., M. Gendreau, G. Pesant, and F. Focacci. 2004. Solving VRPTWs with constraint programming based column generation. *Annals of Operations Research* 130 (1–4):199–216. doi:10.1023/B:ANOR.0000032576.73681.29.

Scholl, A. 1999. *Balancing and sequencing of assembly lines*. Heidelberg: Physica-Verlag.

Sel, C., B. Bilgen, J. M. Bloemhof-Ruwaard, and J. G. A. J. van der Vorst. 2015. Multi-bucket optimization for integrated planning and scheduling in the perishable dairy supply chain. *Computers & Chemical Engineering* 77:59–73. doi:10.1016/j.compchemeng.2015.03.020.

Serra, T., G. Nishioka, and F. J. M. Marcellino. 2012. *The offshore resources scheduling problem: Detailing a constraint programming approach*. Paper presented at Principles and Practice of Constraint Programming, Springer, 823-839, Quebec City, Canada.

Siala, M., E. Hebrard, and M. J. Huguet. 2015. A study of constraint programming heuristics for the car-sequencing problem. *Engineering Applications of Artificial Intelligence* 38:34–44. doi:10.1016/j.engappai.2014.10.009.

Simaria, A. S., and P. M. Vilarinho. 2004. A genetic algorithm based approach to the mixed-model assembly line balancing problem of type II. *Computers & Industrial Engineering* 47 (4):391–407. doi:10.1016/j.cie.2004.09.001.

Sivasankaran, P., and P. Shahabudeen. 2014. Literature review of assembly line balancing problems. *International Journal of Advanced Manufacturing Technology* 73 (9–12):1665–-94. doi:10.1007/s00170-014-5944-y.

Soto, R., B. Crawford, S. Misra, W. Palma, E. Monfroy, C. Castro, and F. Paredes. 2013. Choice functions for autonomous search in constraint programming: Ga Vs. Pso. *Tehnicki Vjesnik-Technical Gazette* 20 (4):621–27.

Soto, R., B. Crawford, W. Palma, E. Monfroy, R. Olivares, C. Castro, and F. Paredes. 2015. Top-k based adaptive enumeration in constraint programming. *Mathematical Problems in Engineering* 2015:1–12. doi:10.1155/2015/580785.

Soto, R., H. Kjellerstrand, O. Durán, B. Crawford, E. Monfroy, and F. Paredes. 2012. Cell formation in group technology using constraint programming and boolean satisfiability. *Expert Systems with Applications* 39 (13):11423–27. doi:10.1016/j.eswa.2012.04.020.

Tiacci, L. 2012. Event and object oriented simulation to fast evaluate operational objectives of mixed model assembly lines problems. *Simulation Modelling Practice and Theory* 24:35–48. doi:10.1016/j.simpat.2012.01.004.

Topaloglu, S., L. Salum, and A. A. Supciller. 2012. Rule-based modeling and constraint programming based solution of the assembly line balancing problem. *Expert Systems with Applications* 39 (3):3484–93. doi:10.1016/j.eswa.2011.09.038.

Venkatesh, J. V. L., and B. M. Dabade. 2008. Evaluation of performance measures for representing operational objectives of a mixed model assembly line balancing problem. *International Journal of Production Research* 46 (22):6367–88. doi:10.1080/00207540701383164.

Wallace, R. J., and D. Grimes. 2008. Experimental studies of variable selection strategies based on constraint weights. *Journal of Algorithms* 63 (1–3):114–29. doi:10.1016/j.jalgor.2008.02.009.

Yagmahan, B. 2011. Mixed-model assembly line balancing using a multi-objective ant colony optimization approach. *Expert Systems with Applications* 38 (10):12453–61. doi:10.1016/j.eswa.2011.04.026.

Yang, C. J., J. Gao, and L. Y. Sun. 2013. A multi-objective genetic algorithm for mixed-model assembly line rebalancing. *Computers & Industrial Engineering* 65 (1):109–16. doi:10.1016/j.cie.2011.11.033.