Assembly line balancing: general resource-constrained case

Albert Corominas, Laia Ferrer, Rafael Pastor

Assembly line balancing: general resource-constrained case

Journal:	*International Journal of Production Research*
Manuscript ID:	TPRS-2009-IJPR-1067.R1
Manuscript Type:	Original Manuscript
Date Submitted by the Author:	21-Mar-2010
Complete List of Authors:	Corominas, Albert; Technical University of Catalonia, IOC (Institute of Industrial and Control Engineering) Ferrer, Laia; Technical University of Catalonia, IOC Research Institute Pastor, Rafael; Universidad Politécnica de Cataluña, Instituto de Organización y Control de Sistemas Industriales
Keywords:	ASSEMBLY LINE BALANCING, ASSEMBLY LINES
Keywords (user):	

Assembly line balancing: general resource-constrained case

Albert Corominas, Laia Ferrer*, Rafael Pastor

IOC Research Institute, Technical University of Catalonia Barcelona, Spain

Avda. Diagonal 647, pl. 11. 08028 Barcelona,
Tel. (+ 34) 93 401 1615 / 6579 / 1701; Fax. (+ 34) 93 401 66 05
albert.corominas@upc.edu, laia.ferrer@upc.edu, rafael.pastor@upc.edu

(Received xxxxx; final version received xxxxx)

The problem of designing and balancing assembly lines has been widely studied in the literature. A recently introduced issue is the efficient use of constrained resources with specific assumptions, in which a task needs a resource type (A) or one of two resources (A $\vee$ B). This paper presents a more general resource-constrained case, in which each task needs resources that may be simple or multiple, alternative and/or concurrent: for instance, (3A), (A $\wedge$ 4B $\wedge$ 3C), (3A $\vee$ 2B $\vee$ C), (A $\wedge$ B) $\vee$ (2C $\wedge$ D) or (A $\vee$ B) $\wedge$ (2C $\vee$ D). We also introduce an upper bound on the number of available resources. Finally, we present a computational experiment using the mathematical models that we develop, showing the instances that can be efficiently solved.

Keywords: assembly line balancing; resource constraint.

1 Introduction

Assembly lines are an important part of many production systems, including those used in the automobile industry. The design and balancing of assembly lines is a hard problem to solve optimally due to its combinatory nature—it is NP-hard (see e.g. Wee and Magazine 1982)—and to the high number of tasks and constraints involved in industrial problems. Basically, balancing consists of assigning indivisible tasks to workstations in such a way that a certain efficiency objective function is optimised (for instance, the number of work stations, the cycle time, or the cost per unit of product). The assignment usually must be solved under constraints such as precedence or incompatibilities between tasks and different and/or limited processing resources.

The problem of designing and balancing assembly lines has been widely studied in the literature and various reviews have been published, the most recent of which include Erel and Sarin (1998), Rekiek *et al*. (2002), Becker and Scholl (2006), Scholl and Becker (2006) and Boysen *et al*. (2007). To solve this problem, exact (e.g. Scholl and Klein 1997 and Pastor and Ferrer 2009), heuristic (e.g. Ponnambalam *et al*. 1999 and Martino and Pastor 2009) and metaheuristic procedures (e.g. Pastor *et al*. 2002) have been developed. Most papers assume that resources (workers, equipment, transportation means, etc.) are homogeneous and available without limits and that all the tasks require the same resources to be processed and can be assigned to any workstation. Only a few papers consider heterogeneous resources in terms of time, cost and/or the tasks to be

* Corresponding author: Laia Ferrer, IOC Research Institute, Av. Diagonal 647 (edif. ETSEIB), p.11, 08028 Barcelona, Spain; Tel. (+ 34) 93 401 65 79; Fax. (+ 34) 93 401 66 05; laia.ferrer@upc.edu

processed (see, for instance, Graves and Withney 1979, Faaland *et al*. 1992, Falkenauer 1997, Pinnoi and Wilhelm 1998, Nicosia *et al*. 2002, Bukchin and Rubinovitz 2003, Amen 2006). These papers solve a double assignment problem: they simultaneously assign resources to tasks and tasks to workstations. Even fewer studies consider limited resources (for instance, Pinnoi and Wilhelm 1997). In addition, Agpak and Gökçen (2005) have defined the resource-constrained assembly line balancing (RCALB) problem.

Agpak and Gökçen (2005) present an industrial problem of assembly line balancing with simultaneous assignment of equipment and tasks to workstations: the resource-constrained case (RCALB). The objective consists in minimising the total number of used resource units "to establish balance of the assembly line with minimum number of stations and resources" (p. 129). Each task can only be processed by a resource A or by a resource B (the RCALB problem, Type 1), or by either of them (A $\vee$ B) (RCALB problem, Type 2). For this purpose, 0-1 integer programming models are developed. The possibility of limited resources is mentioned at the end of Agpak and Gökçen's paper.

We generalise the RCALB problem by defining the general resource-constrained assembly line balancing (GRCALB) problem, which takes into account the following characteristics:

- The resources required to process one task may be more than one resource of one type (e.g. 3A: to process the task 3 units of A resource are required), more than one resource type simultaneously (A $\wedge$ 4B $\wedge$ 3C), or combinations of alternative and/or concurrent resources, e.g. (3A $\vee$ 2B $\vee$ C), (A $\wedge$ B) $\vee$ (2C $\wedge$ D) or (A $\vee$ B) $\wedge$ (2C $\vee$ D).
- The objective function is the total cost. This includes the cost of the stations, which is introduced as usual (e.g. Graves and Lamar, 1983; Pinnoi and Wilhelm, 1997), and the cost of the resources.
- There is an upper bound on the number of available units of each type of resource. It could also be possible to bound the total number of resource units or their total cost (Pinnoi and Wilhelm, 1997).

As some authors have pointed out, the NP-hardness of the balancing line problem implies that algorithms for solving mathematical programming models may not be efficient enough for industrial size problems. However, the computing power is becoming more and more powerful, thanks to the available hardware and software. Consequently, it is important to assess the maximum problem size that can be solved exactly. According to Bixby (2002), in the decade preceding the paper, the problem-solving speed of mathematical programs had increased by a factor of approximately 1,000,000, due to hardware and software improvements. Atamtürk and Savelsbergh (2005, p. 69) stated that "integer programming is rapidly gaining acceptance as a powerful computational tool that can provide optimal or near-optimal solutions to real-life strategic and operational planning problems", as has already been achieved, for example, in Corominas *et al*. (2008). Other exact procedures, such as ad hoc branch and bound, may be in some cases more powerful and able to solve larger instances (this hypothesis should be verified, which is proposed as future research) but branch and bound algorithms are more expertise demanding (for instance, is necessary to define a tight bound, what may be difficult for

this problem) and its development is highly time consuming. From the industrial point of view, developing mathematical programming models that could be easily replicated and efficiently solved using standard optimization software may be more useful.

Mathematical programming models for the resource selection problem can be found in the literature. However, on most occasions, their efficiency has not been studied and heuristic procedures or other exact solution methods have been used. Graves and Lamar (1983) present an integer model that focuses on workstation selection, develops an approximate solution procedure based on relaxations, and tests its effectiveness in a limited set of test problems. Pinnoi and Wilhelm (1997) formulate a family of hierarchical models that incorporate many aspects of assembly design problems (e.g. resource requirement and limited availability) and propose their future exploitation by cutting plane methods. A comparison of the effectiveness of different heuristic methods is presented in Amen (2001). Concerning exact solution methods, Pinnoi and Wilhelm (1998) present an analysis of the effectiveness of a branch and cut to solve the assembly system design problem. Bukchin and Tzur (2000) consider that several resource alternatives are available to process each task and they develop a branch and bound that is able to solve problems of moderate size.

This paper presents a more general resource-constrained case in which each task needs resources that may be simple or multiple, alternative and/or concurrent, and an upper bound on the number of available resources is introduced. Three variations for modelling the problem are presented and their solution performance using mathematical programming is compared. The results identify the most effective model and state the basis for future developments of heuristic procedures.

The rest of the article is organised as follows: the general resource-constrained case is presented and modelled in Section 2; Section 3 presents the computational experiment; and Section 4 presents the conclusions and suggestions for future work.

2 The general resource-constrained case: a model

Next, the general resource-constrained assembly line balancing (GRCALB) problem is presented and modelled as a generalisation of the resource-constrained assembly line balancing (RCALB) problem proposed in Agpak and Gökçen (2005). The aim of this general model is to increase the applicability and bring the theoretical problem closer to the industrial reality.

The input data for the problem are:

- The set of tasks to process and their characteristics: precedence relations, processing times and the resources required (specific requirement of concurrent and/or alternative resources).
- The upper bound on the cycle time.
- The cost of the stations and that of each resource unit.
- The number of available units of each type of resource.

The GRCALB problem consists of simultaneously assigning resources and tasks to workstations, minimising the cost and respecting the existing constraints: precedence relations, resource availability and the processing time in the workstations.

Figure 1 presents the terminology used in this paper for the data and the variables.

Indexes: i (tasks), j (stations), r (types of resources), p (clauses) and q (resource units).

Data:

N	Number of tasks $(i = 1,...,N)$.
$m_{\max}$	Upper bound on the number of stations $(j = 1,...,m_{\max})$.
$m_{\min}$	Lower bound on the number of stations.
R	Number of types of resources $(r = 1,...,R)$.
t_i	Processing time of task i.
CT	Upper bound on the cycle time.
E_i, L_i	First and last station, respectively, where task i may be assigned.
P	Set of pairs of tasks with an immediate precedence relation.
W_j	Set of tasks that may be assigned to station j.
MW_j	Maximum number of tasks that may be assigned to station j, respecting CT.
K_{rj}	Set of tasks that may be processed in j using resource units of type r.
NM_r	Number of available units of resource of type r.
CE_j	Cost of the existence of station j ($CE_j = CE\ (\forall j)$).
CR_r	Cost of a resource unit of type r.
β	Elementary proposition: pair number of units-resource type.
λ	Clause: set of one or more than one elementary propositions β connected by the binary logical operators conjunction $\wedge$ or disjunction $\vee$.
RL_i	Logical expression of task i. RL_i has C_i clauses connected by conjunctions or disjunctions.
C_i	Number of clauses in the logical expression of task i $(p = 1,...,C_i)$.
α_{rpi}	Number of resource units of type r in clause p of task i. $(r = 1,...,R; p = 1,...,C_i; i = 1,...,N)$.
τ_{pi}	Number of types of resources with $\alpha_{rpi} > 0$ in clause p of task i $(p = 1,...,C_i; i = 1,...,N)$.
MR_{rj}	Maximum number of resource units of type r that any task i, which can be assigned to j $(i \in K_{rj})$, may require to be processed: $MR_{rj} = \max_{i \in K_{rj}, p=1,...,C_i} (\alpha_{rpi})$.

Variables:

$x_{ij} \in \{0,1\}$	1, *if* task i is assigned to station j $(\forall i; j = E_i,...,L_i)$.
$y_j \in \{0,1\}$	1, *if* any task is assigned to station j $(j = m_{\min}+1,...,m_{\max})$.
s_{rj}	Number of resource units of type r assigned to station j $(\forall j, \forall r \mid K_{rj} \neq \varnothing)$.
$v_{rqj} \in \{0,1\}$	1, *if* resource unit number q of type r is assigned to station j. $(\forall j; \forall r \mid K_{rj} \neq \varnothing, q = 1,...,MR_{rj})$.
qc_{rpi}	0, *if* the elementary proposition of the resource type r of clause p of task i in CNF is fulfilled $(i = 1,...,N; p = 1,...,C_i; r \mid \alpha_{rpi} > 0)$.
qd_{pi}	0, *if* clause p of task i in DNF is fulfilled $(i = 1,...,N; p = 1,...,C_i)$.

Figure 1. Terminology

Next the three characteristics that define the GRCALB problem are introduced:

- First, the fixed cost corresponding to the existence of the stations $\left(CE_j\right)$ and the cost of one unit of the different types of resources r $\left(CR_r\right)$ are considered. Thus, the objective function minimises the overall assignment cost. The costs CE_j and CR_r allow the manager to balance the number of stations and the number of resources used. Minimisation of the number of resource units (for instance, Agpak and Gökçen, 2005) is a particular case of cost minimisation, $CE_j = 0 \ \ \forall j$ and $CR_r = CR \ \ \forall r$.
- Secondly, heterogeneous resources are considered, in terms of costs (as mentioned above) and the tasks that can be processed. The number of available resource units of each type, r, has an upper bound, NM_r.
- The most important general characteristic of the GRCALB problem is that a wide range of situations are considered with a variety of required resources to process a task i: more than one resource unit of the same type, 3A; more than one resource in a concurrent way, A $\wedge$ 4B $\wedge$ 3C; and other options with alternative and/or concurrent resources, for instance, 3A $\vee$ 2B $\vee$ C, (A$\wedge$B) $\vee$ (2C$\wedge$D) or (A$\vee$B) $\wedge$ (2C$\vee$D). Proposition algebra is used to tackle these situations by modelling the resulting logical expressions through linear constraints and binary variables.

Let us define an elementary proposition β as the pair number of units-resource type, for example: 1A, 3B. Let us define a clause λ as a set of one or more than one elementary propositions β connected by either the binary operator of conjunction $\wedge$ (concurrent resources required) or disjunction $\vee$ (alternative resources required), for instance, $\lambda_1 \equiv \left(\beta_1 \wedge \beta_2\right)$ or $\lambda_2 \equiv \left(\beta_3 \vee \beta_4 \vee \beta_5\right)$. Each task i has an associated logical expression $\left(RL_i\right)$ composed of C_i clauses λ connected by conjunctions or disjunctions. Thus, the expression of resource requirements to process the tasks is generalised. Then, the elementary proposition β of clause p of task i for resource r is $\beta_{rpi} \equiv \left(\alpha_{rpi}, r\right)$.

The transformation of a logical expression with any binary operators into an equivalent one is well known, and can be carried out in one of the following ways (Friedman, 1986): i) conjunctive normal form (CNF), in which elementary propositions β are related to disjunctions, whereas clauses λ are related to conjunctions, for instance, $\lambda_1 \wedge \lambda_2 \wedge \lambda_3 \equiv \left(\beta_1 \vee \beta_2\right) \wedge \left(\beta_3 \vee \beta_4\right) \wedge \left(\beta_5\right)$; ii) disjunctive normal form (DNF), in which elementary propositions β are related to conjunctions, whereas clauses λ are related to disjunctions, for instance, $\lambda_1 \vee \lambda_2 \equiv \left(\beta_1 \wedge \beta_2\right) \vee \left(\beta_3 \wedge \beta_4 \wedge \beta_5\right)$. The transformation from CNF to DNF and vice versa is also well known and consists of applying the following equivalences:

$$\left(\beta_1 \vee \beta_2\right) \wedge \left(\beta_3 \vee \beta_4\right) \equiv \left(\beta_1 \wedge \beta_3\right) \vee \left(\beta_1 \wedge \beta_4\right) \vee \left(\beta_2 \wedge \beta_3\right) \vee \left(\beta_2 \wedge \beta_4\right)$$

$$\left(\beta_1 \wedge \beta_2\right) \vee \left(\beta_3 \wedge \beta_4\right) \equiv \left(\beta_1 \vee \beta_3\right) \wedge \left(\beta_1 \vee \beta_4\right) \wedge \left(\beta_2 \vee \beta_3\right) \wedge \left(\beta_2 \vee \beta_4\right)$$

In the proposed problem, CNF and DNF are used to define the resource requirements and, without any loss of generality, it is assumed that the logical expressions (RL_i) are given in one of these two options.

Next, we present the linear equivalent constraints for the GRCALB problem, for a logical expression in CNF or DNF.

Let RL_i be the logical expression for task i in CNF:

$$\lambda_1 \wedge ... \wedge \lambda_p \wedge ... \wedge \lambda_{C_i} \equiv (\beta_{11i} \vee ... \vee \beta_{r1i} \vee ... \vee \beta_{R1i}) \wedge ... \wedge (\beta_{1C_i i} \vee ... \vee \beta_{RC_i i}) \equiv \bigwedge_{p=1}^{C_i} \bigvee_{r=1}^{R} (\alpha_{rpi}, r)$$

The linear expressions that may model it are, alternatively, constraint sets (1) and (2) or constraint sets (3) and (4):

$$\alpha_{rpi} \cdot x_{ij} \le s_{rj} + \alpha_{rpi} \cdot qc_{rpi} \qquad \left(p = 1,...,C_i; \forall r \middle| \alpha_{rpi} > 0; j \in [E_i,...,L_i]\right) \tag{1}$$

$$\sum_{r|\alpha_{rpi}>0} qc_{rpi} \le \tau_{pi} - 1 \qquad (p = 1,...,C_i) \tag{2}$$

(1) if task i is assigned to station j $(x_{ij} = 1)$, then the number of resource units r assigned to j (s_{rj}) must be greater than or equal to the number of units of r required according to clause p of task i, when the variable (qc_{rpi}) is equal to 0, which indicates the fulfilment of the elementary proposition of the type of resource r of clause p of task i. (2) forces at least one variable qc_{rpi} to be equal to 0 in clause p of the logical expression of task i.

Next, we illustrate constraints sets (1) and (2) with an example, in order to clarify them. Lets assume an instance with $R = 3$ (A, B and C) and the following data concerning task number 9: $RL_9 = (6A \vee 7B) \wedge (8C)$, $E_9 = 4$ and $L_9 = 5$. Then, the parameters concerning task 9 and constraints sets (1) and (2) are:

$$C_9 = 2; \tau_{19} = 2; \tau_{29} = 1; \alpha_{119} = 6; \alpha_{129} = 0; \alpha_{219} = 7; \alpha_{229} = 0; \alpha_{319} = 0; \alpha_{329} = 8;$$

$$\begin{aligned}
&6 \cdot x_{94} \le s_{14} + 6 \cdot qc_{119} && (\text{for } p = 1, r = 1 \text{ and } j = 4)\\
&6 \cdot x_{95} \le s_{15} + 6 \cdot qc_{119} && (\text{for } p = 1, r = 1 \text{ and } j = 5)\\
&7 \cdot x_{94} \le s_{24} + 7 \cdot qc_{219} && (\text{for } p = 1, r = 2 \text{ and } j = 4)\\
&7 \cdot x_{95} \le s_{25} + 7 \cdot qc_{219} && (\text{for } p = 1, r = 2 \text{ and } j = 5)\\
&8 \cdot x_{94} \le s_{34} + 8 \cdot qc_{329} && (\text{for } p = 2, r = 3 \text{ and } j = 4)\\
&8 \cdot x_{95} \le s_{35} + 8 \cdot qc_{329} && (\text{for } p = 2, r = 3 \text{ and } j = 5)
\end{aligned}$$

$$qc_{119} + qc_{219} \le 1 \quad (\text{for } p = 1)$$
$$qc_{329} \le 0 \quad (\text{for } p = 2)$$

Notice $qc_{329} = 0$, therefore previous fifth and sixth constraints become:

$$8 \cdot x_{94} \le s_{34} \quad (\text{for } p = 2, r = 3 \text{ and } j = 4)$$
$$8 \cdot x_{95} \le s_{35} \quad (\text{for } p = 2, r = 3 \text{ and } j = 5)$$

The alternative constraints sets sets (3) and (4) to model logical expression for task i in CNF are:

$$x_{ij} \le \sum_{r|\alpha_{rpi}>0} v_{r,\alpha_{rpi},j} \qquad \left(p = 1,...,C_i; j \in [E_i,...,L_i]\right) \tag{3}$$

$$v_{r,q,j} \le v_{r,q-1,j} \qquad \left(p = 1,...,C_i; \forall r \left| \alpha_{rpi} > 1; 2 \le q \le \alpha_{rpi}; j \in [E_i,...,L_i]\right.\right) \tag{4}$$

(3) if task i is assigned to station j $\left(x_{ij} = 1\right)$, for each clause p of the logical expression of the task i, RL_i, the number of resource units of at least one of the resources r assigned to j must be at least the amount required $\left(\alpha_{rpi}\right)$ to process task i. (4) establishes coherence in the value of the binary variables v_{rqj} and eliminates symmetries.

Next, we illustrate constraints sets (3) and (4) with the same example used for constraint sets (1) and (2). The parameters concerning task 9 are the same presented before and constraints sets (3) and (4) are:

$$x_{94} \le v_{164} + v_{274} \quad (\text{for } p = 1 \text{ and } j = 4)$$
$$x_{95} \le v_{165} + v_{275} \quad (\text{for } p = 1 \text{ and } j = 5)$$
$$x_{94} \le v_{384} \quad (\text{for } p = 2 \text{ and } j = 4)$$
$$x_{95} \le v_{385} \quad (\text{for } p = 2 \text{ and } j = 5)$$

$$v_{124} \le v_{114}; v_{134} \le v_{124}; v_{144} \le v_{134}; v_{154} \le v_{144}; v_{164} \le v_{154}; \quad (\text{for } p = 1, j = 4 \text{ and } r = 1)$$
$$v_{125} \le v_{115}; v_{135} \le v_{125}; v_{145} \le v_{135}; v_{155} \le v_{145}; v_{165} \le v_{155}; \quad (\text{for } p = 1, j = 5 \text{ and } r = 1;$$
$$v_{224} \le v_{214}; v_{234} \le v_{224}; v_{244} \le v_{234}; v_{254} \le v_{244}; v_{264} \le v_{254}; v_{274} \le v_{264}; \quad (\text{for } p = 1, j = 4 \text{ and } r = 2)$$
$$v_{225} \le v_{215}; v_{235} \le v_{225}; v_{245} \le v_{235}; v_{255} \le v_{245}; v_{265} \le v_{255}; v_{275} \le v_{265}; \quad (\text{for } p = 1, j = 5 \text{ and } r = 2)$$
$$v_{324} \le v_{314}; v_{334} \le v_{324}; v_{344} \le v_{334}; v_{354} \le v_{344}; v_{364} \le v_{354}; v_{374} \le v_{364}; v_{384} \le v_{374}; \quad (\text{for } p = 2, j = 4 \text{ and } r = 3)$$
$$v_{325} \le v_{315}; v_{335} \le v_{325}; v_{345} \le v_{335}; v_{355} \le v_{345}; v_{365} \le v_{355}; v_{375} \le v_{365}; v_{385} \le v_{375}; \quad (\text{for } p = 2, j = 5 \text{ and } r = 3)$$

Let RL_i be the logical expression in DNF:

$$\lambda_1 \vee \ldots \vee \lambda_p \vee \ldots \vee \lambda_{C_i} \equiv \left(\beta_{11i} \wedge \ldots \wedge \beta_{r1i} \wedge \ldots \wedge \beta_{R1i}\right) \vee \ldots \vee \left(\beta_{1C_i i} \wedge \ldots \wedge \beta_{RC_i i}\right) \equiv \bigvee_{p=1}^{C_i} \bigwedge_{r=1}^{R} \left(\alpha_{rpi}, r\right)$$

The linear expressions that may model it are (5) and (6):

$$\alpha_{rpi} \cdot x_{ij} \leq s_{rj} + \alpha_{rpi} \cdot qd_{pi} \qquad \left(p = 1, \ldots, C_i; r \middle| \alpha_{rpi} > 0; j \in \left[E_i, \ldots, L_i\right]\right) \qquad (5)$$

$$\sum_{p=1}^{C_i} qd_{pi} \leq C_i - 1 \qquad (6)$$

(5) if task i is assigned to station j $\left(x_{ij} = 1\right)$, then the number of resource units r assigned to j $\left(s_{rj}\right)$ must be greater than or equal to the number of resource units of r required according to clause p of task i, when the variable $\left(qd_{pi}\right)$ is equal to 0, which indicates fulfilment of clause p of task i. (6) establishes that at least one of the variables of qd_{pi} must be equal to 0, in the logical expression RL_i.

Next, we illustrate constraints sets (5) and (6) with the an example analogous to the one presented before, but with the logical expression RL_i for task i in DNF: $R = 3$, $RL_9 = (6A \wedge 7B) \vee (8C)$, $E_9 = 4$ and $L_9 = 5$. The parameters concerning task 9 are the again the same and constraints sets (5) and (6) will be:

$$6 \cdot x_{94} \leq s_{14} + 6 \cdot qd_{19} \quad (\text{for } p = 1, r = 1 \text{ and } j = 4)$$
$$6 \cdot x_{95} \leq s_{15} + 6 \cdot qd_{19} \quad (\text{for } p = 1, r = 1 \text{ and } j = 5)$$
$$7 \cdot x_{94} \leq s_{24} + 7 \cdot qd_{19} \quad (\text{for } p = 1, r = 2 \text{ and } j = 4)$$
$$7 \cdot x_{95} \leq s_{25} + 7 \cdot qd_{19} \quad (\text{for } p = 1, r = 2 \text{ and } j = 5)$$
$$8 \cdot x_{94} \leq s_{34} + 8 \cdot qd_{29} \quad (\text{for } p = 2, r = 3 \text{ and } j = 4)$$
$$8 \cdot x_{95} \leq s_{35} + 8 \cdot qd_{29} \quad (\text{for } p = 2, r = 3 \text{ and } j = 5)$$

$$qd_{19} + qd_{29} \leq 1$$

DNF seems to be the natural form of expressing the possibility of an alternative resource for processing tasks. The possibility of transforming a logical expression from CNF to DNF, and vice versa, allows using three sets of constraints: (1) and (2), or (3) and (4), or (5) and (6).

Next, the mathematical programming model that considers CNF with constraint sets (1) and (2) (CNF-1) or DNF is presented. Then, the changes to be undertaken to model CNF with constraints (3) and (4) (CNF-2) are indicated.

Model:

$$[MIN]\, z = \sum_{j=m_{\min}+1}^{m_{\max}} CE_j \cdot y_j + \sum_{j=1}^{m_{\max}} \sum_{\forall r \mid K_{rj} \neq \varnothing} CR_r \cdot s_{rj} \qquad (7)$$

$$\sum_{j=E_i}^{L_i} x_{ij} = 1 \qquad (i = 1,...,N) \qquad (8)$$

$$\sum_{i \in W_j} t_i \cdot x_{ij} \leq CT \qquad (j = 1,...,m_{\max}) \qquad (9)$$

$$\sum_{i \in W_j} x_{ij} - MW_j \cdot y_j \leq 0 \qquad (j = m_{\min}+1,...,m_{\max}) \qquad (10)$$

$$\sum_{j=E_i}^{L_i} j \cdot x_{ij} \leq \sum_{j=E_k}^{L_k} j \cdot x_{kj} \qquad \forall (i,k) \in P \qquad (11)$$

$$\sum_{\forall j \mid K_{rj} \neq \varnothing} s_{rj} \leq NM_r \qquad (r = 1,...,R) \qquad (12)$$

Constraints (1) and (2) or (5) and (6) regarding RL_i $\forall i$

The objective function (7) minimises the total assignment cost: the cost of the additional stations plus the cost of the resources required. (8) establishes that each task must be assigned to one and only one station. The cycle time of the used stations cannot be greater than the upper bound on the cycle time (9) and (10). (11) establishes the precedence relations between pairs of tasks. The bound on the number of resource units is introduced by (12). Finally, (1) and (2) or (5) and (6) model the logical expressions RL_i, which establish the resource requirements to process task i alternatively and/or concurrently.

The changes required to model CNF with (3) and (4) are to replace (7) with (13); to replace (12) with (14); and to consider constraints (3) and (4) regarding RL_i:

$$[MIN]\, z = \sum_{j=m_{\min}+1}^{m_{\max}} CE_j \cdot y_j + \sum_{j=1}^{m_{\max}} \sum_{\forall r \mid K_{rj} \neq \varnothing} CR_r \cdot \sum_{q=1}^{MR_{rj}} v_{rqj} \qquad (13)$$

$$\sum_{\forall j \mid K_{rj} \neq \varnothing} \sum_{q=1}^{MR_{rj}} v_{rqj} \leq NM_r \qquad (r = 1,...,R) \qquad (14)$$

3 Computational experiment

A computational experiment with various objectives was carried out to study the effectiveness of the models. First, we wanted to identify the most efficient model of either the CNF (CNF-1 or CNF-2) or the DNF. Second, we aimed to study the maximum instance size that can be exactly solved by mathematical programming, in a reasonable computing time for an industrial environment. Finally, a second experiment was performed to analyse the influence of resource limitations.

The computational experiment included parameters and values that are commonly used in the literature (Dar-El 1973, Bukchin and Tzur 2000, Amen 2001, Levitin *et al.* 2006, Pastor and Ferrer 2009).

- Maximum calculation time: 3,600 seconds.
- Number of tasks, N: 20, 30, 40, …, 80
- Order strength, OS: 0.7 and 0.9 (which is the number of existing direct and indirect precedence relations divided by the theoretical maximal number, Mastor 1970).
- Maximal to minimal task processing time, MMT: 5, 10 (task processing time from 5 to 25 and from 5 to 50, respectively).
- Cycle time, CT, to maximal task processing time, $CTMT$: 2 and 3.
- Lower bound on the number of workstations, $m_{\min}$: $\left\lceil \sum_{i=1}^{n} t_i \Big/ ct \right\rceil$.
- Upper bound on the number of workstations, $m_{\max}$: $\min\left(2 \cdot m_{\min}; N\right)$.
- Number of different resource types, R. Two levels are evaluated: 2 and 4 resource types.
- Logical expressions for the tasks, RL_i. For each task i the probability that its logical expression is given in CNF or in DNF is 50%. To solve the instance with a specific model, some RL_i should be converted from CNF to DNF or vice versa. The logical expression of each task i is generated as follows: the tasks do not require any resource (therefore, they do not have a logical expression RL_i) with a probability equal to 0.50; the tasks require only one unit of one type of resource (therefore, RL_i has only one clause with only one elementary proposition) with a probability of 0.30; finally, the remaining 0.20 of the probability corresponds to tasks that have one of two levels of the number of clauses C_i in RL_i, an integer number generated by a uniform distribution from 1 to 2, and from 3 to 4. The number of elementary propositions of each clause (the number of resource types with $\alpha_{rpi} > 0$, τ_{pi}) is an integer generated from a uniform distribution between 1 and R. Finally, the number of resource units of each elementary proposition (value of α_{rpi}) is 1, 2 or 3 with a probability of 0.85, 0.10 and 0.05, respectively.
- Initially, there was no limit on the available resource units of type r, NM_r, so constraints 12 and 14 could be eliminated. In the second experiment the influence of these constraints on the performance of the proposed models was tested.
- Cost $CE_j = CE = 100\ (\forall j)$ and CR_r takes a value from a uniform distribution between 1 and 20.

For each possible combination of the first parameters (number of tasks, N; order strength, OS; maximal to minimal task processing time, MMT; and cycle time to maximal task processing time, $CTMT$), 10 instances were generated, giving a total of 560 instances (80 for each value of N). For each of these instances, two values of the number of resource types were proposed, and their logical expression combinations (with

the 2 possible parameters) were calculated, giving a total of 2,240 instances (320 for each value of N).

Mathematical programs were solved using the ILOG CPLEX 11.0 Optimizer, on a PC 3 GHz Intel Pentium D with 1 GB of RAM.

Table 1 summarises the results of the computational experiment, showing the number of instances with a proved optimal solution (Opt), the number of instances with a feasible solution (Fea), the number of instances with no feasible solution $(\overline{Fea})$ and the average number of variables (Var) and constraints ($Cons$) obtained by each of the three models (CNF-1, CNF-2, DNF) for each number of tasks (N).

Please insert Table 1

Table 1 shows that CNF-2 obtained the best results. With CNF-2, a greater number of instances were solved optimally (1,302 vs. 1,278 for CNF-1 or 1,288 for DNF) and there were fewer instances with no feasible solution (105 vs. 146 for CNF-1 and 131 for DNF).

Table 1 also shows that the proposed models were able to solve significantly more instances than Agpak and Gökçen (2005), which only solved an instance of 11 tasks. In particular, CNF-2 solved all of the instances up to 50 tasks and almost all the instances of 60 tasks (only one instance with no feasible solution).

The average number of variables and constraints increases with the number of tasks; for each N value, the number of variables is very similar for the three models and the number of constraints is similar for CNF-2 and DNF and slightly lower for CNF-1. Although compared with nowadays standards the number of variables and constraints may not be considered very high, the results of the instances from 70 tasks on are not satisfactory enough, which confirms the high difficulty of solving this problem.

An analysis of the different characteristics of the instances was undertaken. Table 2 shows the solutions of the instances ((Opt), (Fea) or $(\overline{Fea})$) obtained by each of the three models (CNF-1, CNF-2, DNF), depending on order strength (OS, 0.7 or 0.9), the maximal to minimal task processing time (MMT, 5 or 10), the cycle time to maximal task processing time ($CTMT$, 2 or 3), the number of resource types (R, 2 or 4), and the number of clauses C_i in RL_i for those tasks that required more than one unit of one type of resource (C_i, 1-2 or 3-4).

Please insert Table 2

As expected, the results were better for instances with higher OS. Although there were a few more instances with no solution among the instances with OS 0.9 vs. 0.7, the number of optimal solutions in the instances with higher OS was significantly greater

(735 vs. 543 with CNF-1, 750 vs. 552 with CNF-2 and 741 vs. 547 with DNF). Results were also better for instances with higher MMT, for example, regarding the number of optimal solutions: 674 vs. 604 for CNF-1, 679 vs. 623 for CNF-2 and 680 vs. 608 for DNF. The influence of $CTMT$ is even clearer: a solution was obtained in almost all instances with high $CTMT$ (only 1 instance with CNF-1 did not obtain a feasible solution), whereas with low $CTMT$ there were 145, 105 and 131 instances with no solution for CNF-1, CNF-2 and DNF, respectively. The different levels of R and C_i did not have a clear influence on the results.

This comparison of the performance of the three models depending on the characteristics of the instances confirms that CNF-2 obtains the best results. CNF-2 always obtains more optimal and more feasible solutions than CNF-1 for each of the characteristics. DNF obtains slightly better results than CNF-2 in some specific cases: with an OS value of 0.9 DNF obtained one feasible solution more than CNF-2; and DNF obtained one optimal solution more than CNF-2 for an MMT value of 10. Even with these minor exceptions, we can recommend the use of CNF-2, regardless of the characteristics of the instance to solve.

An ANOVA (STATGRAPHICS Plus, Statistical Graphics Corp.) analysis was undertaken to evaluate relative behaviour between the three models and the influence of the characteristics of the problem instances—in particular, the number of resources (R) and the number of tasks (N). Next, we summarise the main conclusions obtained by means of the Fisher Test Graphics provided by ANOVA. Figure 2 confirms the results shown in Table 1: the model with the best overall behaviour was CNF-2, in terms of the number of optimal solutions and the total number of solutions. As we can see in Figure 3, CNF-2 had robust behaviour for the characteristics R and N. Concerning the number of optimal solutions (Opt), Figure 3a shows CNF-2 had the best performance for low and high values of R, in terms of time (Figure 3b). The three models had similar performance for lower R, but CNF-2 was significantly the quickest model in the instances of higher R. Concerning the total number of solutions obtained $(Opt\ Fea)$, CNF-2 also had the best results for low and high levels of R (Figure 3c). Finally, Figure 3d shows that this superiority increased with the number of tasks of the instances.

Please Insert Figure 2

Please Insert Figure 3

Next, an additional computational analysis was carried out to study the influence of limitations in resource availability. Thus, the maximum number of units of a resource type r was set, NM_r, and constraint sets (12) and (14) were added.

First, we analysed the influence of limitations of resource availability on the optimum value of the objective function. We took as an example an instance of 40 tasks with the following characteristics: OS 0.7, MMT 10, $CTMT$ 2, R 2, C_i 1-2. With no

bounds on the number of resource units, an optimal solution for this instance was obtained, with a value of the objective function equal to 127, using 6 units of resource 1 (R1') and 7 units of resource 2 (R2'). We ran variations of the instance by changing the bounds on the available resource units of each type (R1 and R2, respectively). Table 3 shows the results: the values of the optimal solutions, the variations in which no feasible solution was obtained (NS), and those in which it was proved that no feasible solution existed (-1). The bound of each type of resource started at 10 units and diminished until it was proved that the solution of the instance was infeasible. Of course, when $R1 \geq R1'$ and $R2 \geq R2'$ (-), the optimal solution was 127 (the value obtained with no resource limits).

Please insert Table 3

Table 3 illustrates how the value of the solution of the instance varied when the available resource units (R1 and/or R2) were reduced, the value of the cost was increased, or it became infeasible. When $R2 < R2'$ and $R1 \geq R1'$, first, the instance was still feasible and the value of the solution kept on increasing; next, no solution was found; and, finally, the instance became infeasible. However, when $R1 < R1'$, the instance found no solution and became infeasible.

Finally, a wider computational experiment was carried out to analyse the influence of the resource limits on the effectiveness of the proposed models. We took as an example the instances of 30 and 40 tasks and we focused the analysis on CNF-2, which had already been identified as the most effective model. In order to ensure that the constraints were active, the value NM_r of the most used resource was set to one unit less than the value obtained with no resource limits. The maximum computing time was set to 3600 s. Table 4 compares the solutions of the instances ((Opt), (Fea) or $(\overline{Fea})$) obtained for the instances of 30 and 40 tasks, with no resource limits (CNF-2) and with resource limits (CNF-2_res-lim).

Please insert Table 4

Due to the complexity increase, the model with active resource limits (CNF-2_res-lim) was not as effective as the model without these constraints (CNF-2): CNF-2 obtained a feasible solution in all instances whereas CNF-2_res-lim did not. However, the percentage of instances with a feasible solution obtained with CNF-2_res-lim was high: a solution was obtained in 84.4% of the instances of 30 tasks and in 80% of the total instances (30 and 40 tasks).

4 Conclusions and future works

This paper presents the general resource-constrained assembly line balancing (GRCALB) problem as a generalisation of the resource-constrained assembly line balancing problem

(RCALB) proposed in Agpak and Gökçen (2005). This generalisation expands the applicability of the theoretical problem and brings it closer to the industrial reality. The main innovation in the GRCALB problem is that it considers simple or multiple, alternative and/or concurrent resource requirements to process the tasks. Moreover, the cost of station existence and of the resources is considered, and the limits on the number of resource units of each type are introduced.

The GRCALB problem is modelled through mathematical programming and exactly solved considering tools and computing times that are acceptable in an industrial environment. The results of this wide computational experiment are presented: the exact solution is obtained in small and medium sized instances; all instances up to 60 tasks are solved, with only one instance exception. This size is comparable to that of real industrial problems dealt with in the literature. For instance, Cortés *et al*. 2009 solves a real life assembly line balancing problem for a motorcycle manufacturing company with 57 tasks and *CTMT* equal to 1.6; Corominas *et al.* 2008 solves another real life example of a motorcycle manufacturing company with 107 tasks, *OS* equal to 0.7 and *CTMT* equal to 1.28. An additional computational experiment also illustrates how the limits on resource availability may lead to an increase in the value of the solution and shows how models with active resource limits are effective. For the solution of larger instances, further research could focus on developing ad hoc branch and bound or heuristic procedures based on the mathematical models presented (Wolsey 1998; for instance, Fix and Relax, Escudero and Salmeron 2005, or Dive and Fix, Hoffman 2000) and metaheuristic procedures.

Acknowledgements

This paper was supported by the Spanish MCyT project DPI2007-61905 and co-financed by FEDER.

References

Agpak, K. and Gökçen, H., 2005. Assembly line balancing: Two resource constrained cases. International Journal of Production Economics, 96, 129–140.

Amen, M., 2001. Heuristic methods for cost-oriented assembly line balancing: A comparison on solution quality and computing time. International Journal of Production Economics, 69, 255–264.

Amen, M., 2006. Cost-oriented assembly line balancing: Model formulations, solution difficulty, upper and lower bounds. European Journal of Operational Research, 168, 747–770.

Atamtürk, A. and Savelsbergh, M.W.P., 2005. Integer-programming software systems. Annals of Operations Research, 140, 67–124.

Becker, C. and Scholl, A., 2006. A survey on problems and methods in generalized assembly line balancing. European Journal of Operational Research 168, 694–715.

Bixby, R.E., 2002. Solving real-world linear programs: a decade and more of progress. Operations Research, 50, 3–15.

Boysen, N., Fliedner, M. and Scholl, A., 2007. A classification of assembly line balancing problems. European Journal of Operational Research, 183, 674–693.

Bukchin, J. and Tzur, M., 2000. Design of flexible assembly line to minimize equipment cost. IIE Transactions, 32, 585–598.

Bukchin, J. and Rubinovitz, J., 2003. A weighted approach for assembly line design with station paralleling and equipment selection. IIE Transactions, 35, 73–85.

Corominas, A., Pastor, R. and Plans, J., 2008. Balancing assembly line with skilled and unskilled workers. OMEGA, 36, 1126–1132.

Cortés, P., Onieva, L., and Guadix, J., 2009, Optimising and simulating the assembly line balancing problem in a motorcycle manufacturing company: a case study. International Journal of Production Research (first published on 10 June 2009, doi: 10.1080/00207540902926522).

Dar-El, E.M., 1973. MALB - A heuristic technique for balancing large single-model assembly lines. AIIE Transactions, 5, 343–356.

Erel, E. and Sarin, C.S., 1998. A survey of the assembly line balancing procedures. Production Planning & Control, 9, 414–434.

Escudero, L.F,. and Salmeron, J. 2005. On a Fix-and-Relax Framework for a Class of Project Scheduling Problems Annals of Operations Research 140, 163–188,

Faaland, B.H., Klastorin, T.D., Schmitt, T.G. and Shtub, A., 1992. Assembly line balancing with resource dependent task times. Decision Sciences, 23, 343–364.

Falkenauer, E., 1997. A grouping genetic algorithm for line balancing with resource dependent task times. Proceedings of the Fourth International Conference on Neural Information Processing, 464–468.

Friedman, A., 1986. Fundamentals of Logic Design and Switching Theory. Computer Science Press.

Graves, S.C. and Lamar, B.W., 1983. An integer programming procedure for assembly system design problems. Operations Research, 31, 522–545.

Graves, S.C. and Withney, D.E., 1979. A mathematical programming procedure for equipment selection and system evaluation in programmable assembly. Proceedings of the IEEE Decision and Control, 531–536.

Hoffman, K.L., 2000. Combinatorial optimization: Current successes and directions for the future. Journal of Computational and Applied Mathematics, 124, 341-360.

Levitin, G., Rubinovitz, J. and Shnits, B., 2006. A genetic algorithm for robotic assembly line balancing. European Journal of Operational Research, 168, 811–825.

Martino, L. and Pastor, R., 2009. Heuristic procedures for solving the general assembly line balancing production with setups. International Journal of Production Research (first published on 11 February 2009, doi: 10.1080/00207540802577979).

Mastor, A.A., 1970. An experimental investigation and comparative evaluation of production line balancing techniques. Management Science, 16, 728–746.

Nicosia, G., Pacciarelli, D. and Pacifici, A., 2002. Optimally balancing assembly lines with different workstations. Discrete Applied Mathematics, 118, 99–113.

Pastor, R., Andrés, C., Durán, A. and Pérez, M., 2002. Tabu search algorithms for an industrial multi-product and multi-objective assembly line balancing problem, with reduction of the task dispersion. Journal of the Operational Research Society, 53, 1317–1323.

Pastor, R. and Ferrer, L., 2009. An improved mathematical program to solve the simple assembly line balancing problem. International Journal of Production Research 47, 2943–2959.

Pinnoi, A. and Wilhelm, W.E., 1997. A family of hierarchical models for assembly system design. International Journal of Production Research, 35, 253–280.

Pinnoi, A. and Wilhelm, W.E., 1998. Assembly system design: A branch and cut approach. Management Science 44, 103–118.

Ponnambalam, S.G., Aravindan, P. and Mogileeswar, G., 1999. A comparative evaluation of assembly line balancing heuristics. International Journal of Advanced Manufacturing Technology, 15, 577–586.

Rekiek, B., Dolgui, A., Delchambre, A. and Bratcu, A., 2002. State of art of optimization methods for assembly line design. Annual Reviews in Control 26, 163–174.

Scholl, A. and Klein, R., 1997. SALOME: A bidirectional branch and bound procedure for assembly line balancing. INFORMS J Comp, 9, 319–334.

Scholl, A. and Becker, C., 2006. State-of-the-art exact and heuristic solution procedures for simple assembly line balancing. European Journal of Operational Research, 168, 666–693.

Wee, T.S. and Magazine, M.J., 1982. Assembly line balancing as generalized bin packing. Operations Research Letters, 1, 56–58.

Wolsey, L.A. 1998. Integer Programming. Wiley.

TABLES

	CNF-1					CNF-2					DNF				
N	*Opt*	*Fea*	$\overline{Fea}$	*Var*	*Cons*	*Opt*	*Fea*	$\overline{Fea}$	*Var*	*Cons*	*Opt*	*Fea*	$\overline{Fea}$	*Var*	*Cons*
20	320	0	0	196	175	320	0	0	189	153	320	0	0	191	153
30	317	3	0	388	330	320	0	0	381	288	318	2	0	382	286
40	279	41	0	624	517	281	39	0	620	457	280	40	0	617	448
50	194	126	0	927	761	203	117	0	930	670	196	124	0	917	649
60	103	209	8	1286	1017	107	212	1	1290	898	103	212	5	1274	874
70	43	241	36	1677	1323	49	250	21	1684	1174	50	238	32	1664	1130
80	22	196	102	2152	1680	22	215	83	2170	1489	21	205	94	2136	1428
Total	**1,278**	**816**	**146**			**1,302**	**833**	**105**			**1,288**	**821**	**131**		

Table 1. Results of the computational experiment depending on the number of tasks of the instances.

		CNF-1			CNF-2			DNF		
		Opt	*Fea*	$\overline{Fea}$	*Opt*	*Fea*	$\overline{Fea}$	*Opt*	*Fea*	$\overline{Fea}$
OS	0.7	543	514	63	552	537	31	547	515	58
	0.9	735	302	83	750	296	74	741	306	73
MMT	5	604	428	88	623	435	62	608	436	76
	10	674	388	58	679	398	43	680	385	55
CTMT	2	495	480	145	507	508	105	500	489	131
	3	783	336	1	795	325	0	788	332	0
R	2	639	409	72	646	418	56	643	409	68
	4	639	407	74	656	415	49	645	412	63
C_i	1-2	648	398	74	652	414	54	649	408	63
	3-4	630	418	72	650	419	51	639	413	68

Table 2 Results of the computational experiment depending on the characteristics of the instances.

		$R2 \geq R2'$				$R2 < R2'$				
		10	9	8	7	6	5	4	3	2
$R1 \geq R1'$	10	-	-	-	-	146	165	224	NS	-1
	9	-	-	-	-	146	165	224	NS	-1
	8	-	-	-	-	146	165	224	NS	-1
	7	-	-	-	-	146	224	224	NS	-1
	6	-	-	-	127	224	224	224	NS	-1
$R1 < R1'$	5	NS	NS	NS	NS	NS	NS	NS	NS	-1
	4	-1	-1	-1	-1	-1	-1	NS	-1	-1

Table 3 Influence on solution of the limitations on the available resources.

Model	**CNF-2**			**CNF-2_res-lim**		
Tasks	*Opt*	*Fea*	$\overline{Fea}$	*Opt*	*Fea*	$\overline{Fea}$
30	320	0	0	265	5	50
40	281	39	0	217	25	78
Total	**601**	**39**	**0**	**482**	**30**	**128**

Table 4 Comparison of the results with and without resource limits.

FIGURE CAPTION

Insert Figure 2a	**Insert Figure 2b**
a) Number of optimal solutions	b) Number of optimal and feasible solutions

Figure 2. Means and 95.0% LSD interval graphic for models.

Insert Figure 3a	**Insert Figure 3b**
a) Number of optimal solutions depending on the resources	b) Computing time to get the optimal solutions depending on the resources
Insert Figure 3c	**Insert Figure 3d**
c) Number of optimal and feasible solutions depending on the resources	d) Number of optimal and feasible solutions depending on the tasks

Figure 3. Interaction plots for the number of resources R and the number of tasks N.

Figure 2a:

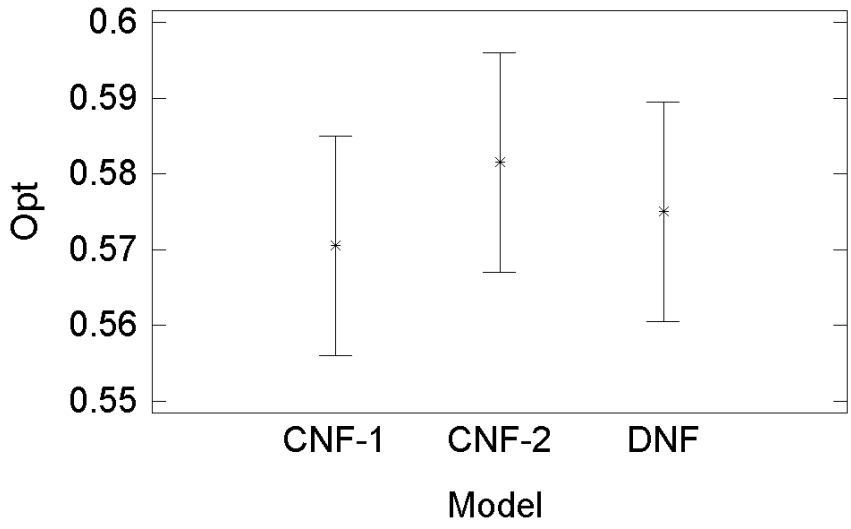

Figure 2b:

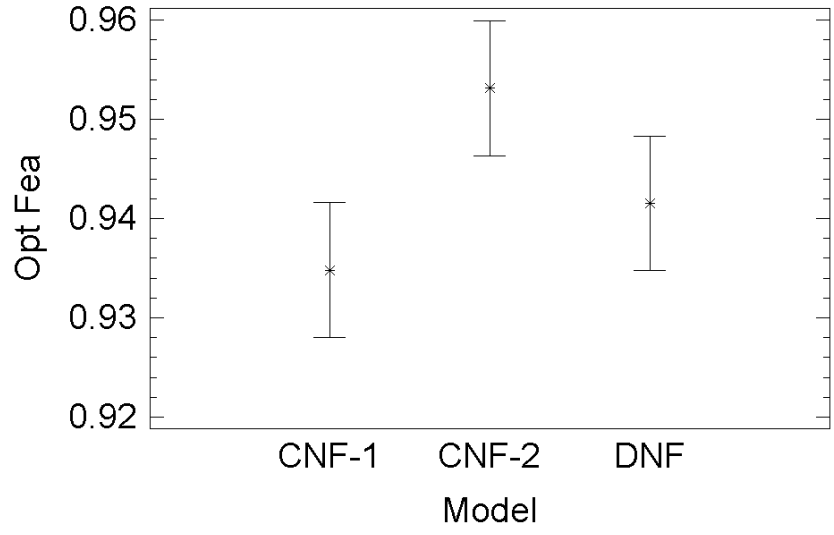

Figure 3a:

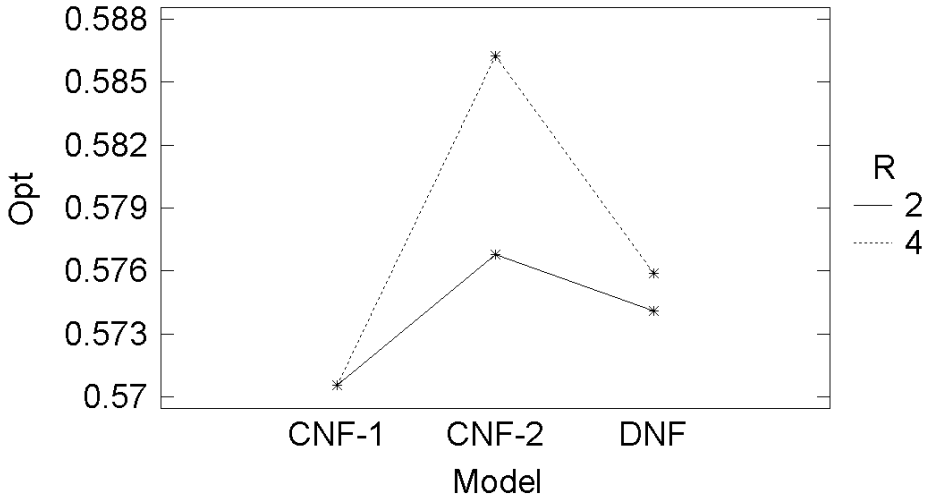

Figure 3b:

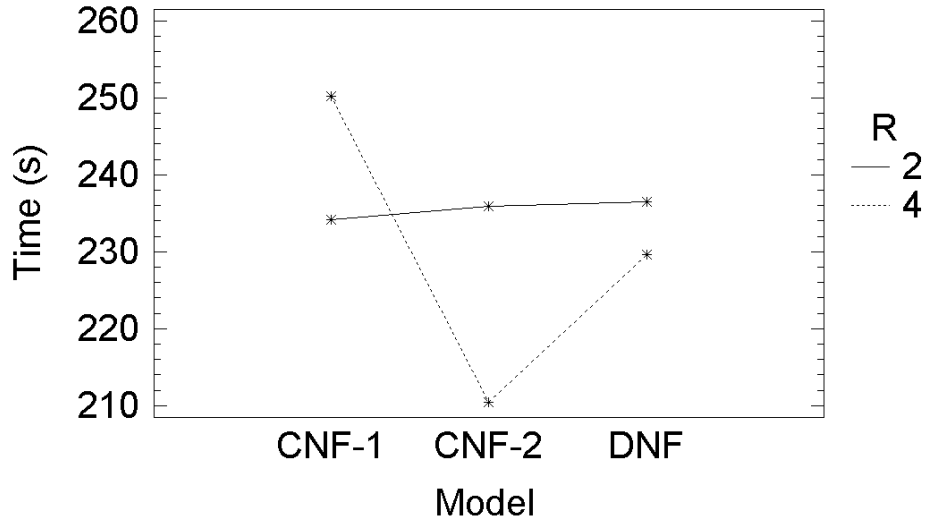

Figure 3c:

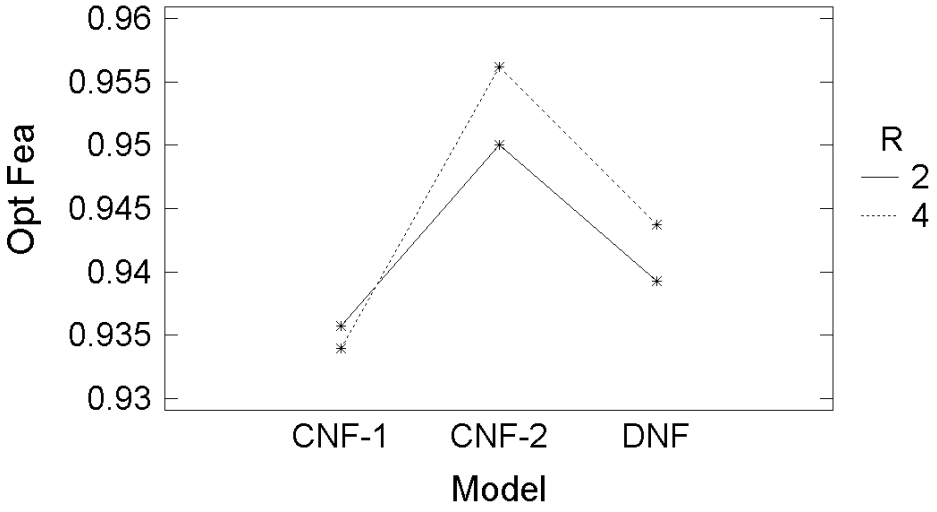

Figure 3d: