NTIGen: a Software for Generating Nissan Based Instances for Time and Space Assembly Line Balancing

Chica M[1], Cordón O[2], Damas S[3], Bautista J [4]

Abstract The time and space assembly line balancing problem (TSALBP) is a realistic multiobjective version of assembly line balancing industrial problems involving the joint optimization of conflicting criteria such as the cycle time, the number of stations, and the area of these stations. For this family of problems there is not any repository where researchers and practitioners can obtain realistic problem instances also containing information on mixed products plans. In this contribution we introduce a new TSALBP instance software generator that can produce problem instances having industrial real-like features. This generator is called NTIGen (Nissan TSALBP Instance GENerator) since it is developed from the information and real data of the assembly line and production planning of the Nissan plant of Barcelona. The NTIGen software as well as some benchmark instances are publicly available on Internet and could be used by researchers to carry out general TSALBP experiments and to also discriminate between different assembly line configurations when future demand conditions vary.

Keywords: Time and Space Assembly Line Balancing, Problem Instance Generator, Mixed Products, Nissan, Optimization

[1]Manuel Chica Serrano (✉ e-mail: manuel.chica@softcomputing.es)
European Centre for Soft Computing, Gonzalo Gutiérrez Quirós s/n 33600 Mieres (Asturias).

[2]Óscar Cordón García (✉ e-mail: oscar.cordon@softcomputing.es)
DECSAI and CITIC-UGR, Universidad de Granada, 18071 Granada
European Centre for Soft Computing, Gonzalo Gutiérrez Quirós s/n 33600 Mieres (Asturias).

[3]Sergio Damas Arroyo (✉ e-mail: sergio.damas@softcomputing.es)
European Centre for Soft Computing, Gonzalo Gutiérrez Quirós s/n 33600 Mieres (Asturias).

[4]Joaquín Bautista Valhondo (✉ e-mail: joaquin.bautista@upc.edu)
Dpto. Organización de Empresas, ETSEIB, UPC Catalunya, 08028 Barcelona.

* This work has been supported by Ministerio de Economía y Competitividad under project SOCOVIFI2 (TIN2012-38525-C02-01 and TIN2012-38525-C02-02), and under PROTHIUS-III: DPI2010-16759, both including EDRF funding.

1 Introduction

An assembly line is made up of a number of workstations, arranged either in series or in parallel. Since the manufacturing of a production item is divided into a set of tasks which require an operation time for their execution, a usual and difficult problem, called assembly line balancing (ALB), is to determine how these tasks can be assigned to the stations fulfilling certain restrictions such as precedence relations. The final aim of ALB is to get an optimal assignment of subsets of tasks to the stations of the plant (Boysen et al. 2007). A well-known family of ALB problems is the simple assembly line balancing problem (SALBP) (Baybars 1986, Scholl and Becker 2006). The SALBP only considers the assignment of each task to a single station in such a way that all the precedence constraints are satisfied and no station workload time is greater than the line cycle time.

As a result of the observation of the ALB operation in an automotive Nissan plant from Barcelona (Spain), Bautista and Pereira (2007) recently proposed a SALBP extension aiming to design a more realistic ALB model. They considered an additional space constraint to get a simplified but closer version to real-world situations, defining the time and space assembly line balancing problem (TSALBP). The TSALBP presents eight variants depending on three optimization criteria: m (the number of stations), c (the cycle time), and A (the area of the stations). The multicriteria nature of the TSALBP favoured the application of multiobjective meta-heuristics such as multiobjective ant colony optimization (Chica et al. 2010), evolutionary multiobjective optimization (Chica et al. 2011), and memetic algorithms (Chica et al. 2012).

However, we noticed the absence of an available dataset with real-like instances for the TSALBP. And what is more, there is not any instance containing mixed product plans when the demand is uncertain. Therefore, we have implemented real-like Nissan TSALBP instance generator software (NTIGen) in order to let researchers to validate their models and methods in a diverse set of TSALBP instances and production plans. The design and implementation of NTIGen is done with the use of real data and industrial features of the Nissan industry plant of Barcelona. The software is freely available on-line to be used for future research works. Using this tool, a set of eight instances has been generated as a benchmark to show the different features of the instances.

The rest of the paper is structured as follows. In Section 2 the TSALBP formulation is explained. The description of the NTIGen software is shown in Sections 3 and 4. A comparison of the eight instances generated by NTIGen is shown in Section 5. Some concluding remarks are given in Section 6.

2 Time and Space Assembly Line Balancing Problem

The manufacturing of a production item is divided into a set J of n tasks. Each task j requires an operation time for its execution $t_j > 0$ that is determined as a function of the manufacturing technologies and the employed resources. Each station k ($k = 1, 2,..., m$) is assigned to a subset of tasks S_k ($S_k \subseteq J$), called workload. Each task j can only be assigned to a single station k.

Each task j has a set of direct "preceding tasks" P_j which must be accomplished before starting it. These constraints are normally represented by means of an acyclic precedence graph, whose vertices stand for the tasks and where a directed arc (i, j) indicates that task i must be finished before starting task j on the production line. Thus, task j cannot be assigned to a station that is ordered before the one where task i was assigned. Each station k also presents a station workload time $t(S_k)$ that is equal to the sum of the tasks' processing time assigned to the station k. SALBP focuses on grouping tasks in workstations by an efficient and coherent way.

In this simplistic model there is a need of introducing space constraints in assembly lines' design based on two main reasons: (a) the length of the workstation is limited in the majority of the situations, and (b) the required tools and components to be assembled should be distributed along the sides of the line. Hence, an area constraint may be considered by associating a required area a_j to each task j and an available area A_k to each station k that, for the sake of simplicity, we shall assume it to be identical for every station and equal to $A = max_k=1,2,...,m_{Ak}$. Thus, each station k requires a station area $a(S_k)$ that is equal to the sum of areas required by the tasks assigned to station k.

This leads us to a new family of problems called TSALBP (Bautista and Pereira 2007). It may be stated as: given a set of n tasks with their temporal t_j and spatial a_j attributes ($1 \leq j \leq n$) and a precedence graph, each task must be assigned to a single station such that: (i) every precedence constraint is satisfied, (ii) no station workload time ($t(S_k)$) is greater than the cycle time (c), and (iii) no area required by any station ($a(S_k)$) is greater than the available area per station (A).

TSALBP presents eight variants depending on three optimization criteria: m (the number of stations), c (the cycle time) and A (the area of the stations). Within these variants there are four multiobjective problems depending on the set of criteria to be minimized (m, c and/or A). For more information about the problem we refer the interested reader to Chica et al. (2010; 2012).

3 Basics of the Nissan TSALBP Instance Generator Software (NTI-Gen)

The main goal of the NTIGen software is to create real-like TSALBP instances with different features to serve as a benchmark for any future research work. Although there are ALB instances available online and even a SALBP instance generator (Otto et al. 2011), there is not any existing source where TSALBP instances can be generated and referred. Also, as pointed out in the Introduction section, there is no instance containing production plans information.

Assembly lines in the automotive industry present a set of industrial features which condition the task and graph distribution of the problem instance. The user must be allowed to incorporate these industrial real-like features to the generated instances and these instances should be similar to the original Nissan instance context (Chica et al. 2012). Concretely, the developed NTIGen software includes the following features, which are illustrated in Figure 1:

- *Checkpoints:* They are assembly line points in which workers test the quality and completeness of a set of operations previously finished. If we consider these checkpoints as new tasks, the representation of a checkpoint in an assembly line graph is given by a task having a high number of preceding tasks (for instance, task 11 in Figure 1).
- *Tasks without precedence*: In real industrial scenarios, such tasks are justified if there are operations unconditioned by other operations. They are commonly found in the engine and trim lines of the car manufacturing. In Figure 1, tasks 1, 3, 8, 7, and 10 have no precedence.
- *Final tasks:* Tasks in an assembly line which are associated to the most external and final operations of the product. They are represented as tasks with no successors in the precedence graph (tasks 12, 13 and 14 in Fig 1).
- *Isolated tasks:* They can be performed at any part of the assembly of an item. An example of these kinds of tasks is this related with additional parts of a product which can be incorporated to the global product at any station. Task 4 in Figure 1 is an isolated task as it has no precedence.

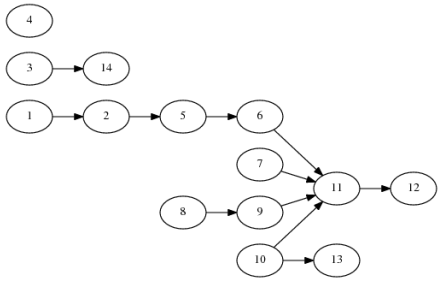

Fig. 1 A precedence graph with 13 tasks showing examples of different kinds of tasks in an industrial context: chains of tasks, initial and final tasks, isolated tasks, and checkpoints.

- *Operations aggregation*: This process comes up when some operations need the same tools or are done by the same worker. In this case, several tasks of the same stage are put together in just one task.
- *Operations breaking up*: If possible, it is used in the industrial context to detail the implementation of an operation in different operating tasks. It is useful for balancing an assembly line when the cycle time is reduced.
- *Chains of tasks*: They appear when there are strongly linked operations, normally in the same station or stage. A chain of tasks represents natural sequences of operations within the assembly process (see tasks 1, 2, 5, and 6 in Figure 1).

4 Tuneable Parameters of NTIGen

The features introduced in the previous sub-section can be parameterized by the NTIGen user to generate a customizable instance. NTIGen is also fed by a set of stages with some initial tasks. By default, these stages and tasks correspond to the original Nissan instance with 140 tasks and 21 workstations (Chica et al. 2012) although they can be modified by the user before launching the application. The user can set all the desired features by changing the parameters of an XML file. The most important input parameters are the following:

- *Number of tasks (n).* This is an important parameter of the instance that enormously conditions its complexity. From the initial set of tasks, new operating tasks are generated by breaking up them until reaching the user needs. If we need fewer tasks than the original ones, they are merged at random. The new generated tasks are required to belong to the same or close stages than their original ones.
- *Processing times (t_j).* The processing time of each task t_j is randomly disrupted by a normal distribution within a user-defined interval. When creating or merging tasks, the processing times for the resulting tasks are reduced or duplicated, respectively. This is done to maintain the original situation of the Nissan instance.
- *Production plans.* The production plans are always set to the NSIO original plans. The processing times of the tasks for the different engine products are created by randomly modifying the original processing time t_j within the range $[0.9t_j, 1.1t_j]$.
- *Cycle time (c).* It is also disrupted independently from the processing times of the tasks. As done with t_j, the disruption is created within a user-defined interval. In our case, the new cycle time is set to a value within $[0.75c, 1.25c]$.
- *Required operation area (a_j).* Task areas are specified by two-dimensional units, i.e. length (a_j) and width (b_j). The first dimension, a_j, is the truly useful variable for the TSALBP optimization. In the original instance, b_j is always set

to one distance unit. To generate a new instance, the squared area of each task is always maintained by the generator but b_j is randomly changed to a set value. In our case, the set is given by *{0.5, 0.75, ..., 2.25}*. This set of possible b_j values can be modified by the user of the NTIGen software. Therefore, the length of each task a_j, used for the optimization, is different for each generated TSALBP instance. As done with the processing times, a_j is reduced or duplicated when increasing or decreasing the number of tasks to try to maintain the original Nissan situation.

Apart from the operating tasks and their corresponding processing times and areas, NTIGen generates the precedence graph of the instance. These precedence relations are created between tasks of the same stage (generating chains) or different stages within a maximum window, set by the user, in order to link tasks which are industrially close. The minimum and maximum number of preceding tasks for a checkpoint in a problem instance can be set prior the instance generation. The same definition can be done for the number of initial, final, and isolated tasks.

NTIGen creates precedence relations until it reaches the required complexity of the graph which is another important feature of an ALB instance (Bhattacharjee and Sahu 1990). This complexity of the precedence graph is also a user parameter and it is measured by the order strength (*OS*) of the graph (Dar-El 1975). The *OS* is calculated from the graph in transitive closure. The transitive closure of a set of direct precedences E is given by $E^T = \{(i, j) | i \in V, j \in F_i^T\}$, with V being the set of nodes and F_i^T the set of indirect successors of the task i. The *OS* represents the number of ordering relations of the graph in a transitive closure with respect to all possible ordering relations: $OS = \frac{|E^T|}{\frac{n(n-1)}{2}}$.

The *OS* varies between *[0, 1]*. If *OS* is equal to 0 the instance has no precedence relations but if *OS* takes value 1, there is just one feasible sequence of tasks. The result after running the NTIGen software is a structured text file describing the generated instance with the list of tasks, their operating times and area, and their precedence relations. The precedence relations form the transitive reduction of the graph in order to minimize computational resources.

In addition, by changing the number of tasks, their processing time and area we can generate instances having different time variability (TV) and area variability (AV). Descriptors about the generated instance are listed after its creation to show the complexity of the graph, TV, AV, and the number of checkpoints, isolated, initial, and final tasks.

5 Examples of some Generated TSALBP Instances

By using the NTIGen software, a set of eight new TSALBP real-like instances have been created (Table 1). The NTIGen software and this set of TSALBP instances are publicly available at http://www.prothius.com/TSALBP.

Table 1 Main characteristics of the generated TSALBP instances

Features	P1	P2*	P3	P4	P5	P6	P7	P8
Random seed	24151	N/A	117017	21277	113683	56399	5869	73553
No. of tasks	100	140	190	220	280	320	376	420
Cycle time	199.97	180	207.07	222.42	221.62	169.552	186.65	137.751
OS	0.5	0.9	0.7	0.5	0.3	0.6	0.25	0.95
Precedences	156	293	314	304	407	435	548	608
Precs. window	5	N/A	5	1	2	1	3	2
TV	35.95	24	41.75	151.45	224.29	2742.28	901.34	1003.77
AV	500	513.86	266.67	300	400	200	300	133.33
Initial tasks	14	1	6	33	59	32	87	6
Final tasks	8	5	7	20	42	31	49	8
Isolated tasks	2	0	5	3	0	5	0	3
Checkpoints	3	N/A	0	6	7	1	12	0

6 Concluding Remarks

The existing TSALBP formulation and previous ALB works do not cover an important real scenario where the same assembly line is devoted to produce mixed products and their demand is not fixed. Furthermore, the TSALBP instances of the literature were created by modifying ALB instances. The NTIGen software presented in this work allows researchers to create realistic TSALBP instances and production plans for future research. The generated TSALBP instances contain many real-like industrial features, e.g. checkpoints, isolated tasks, initial and final tasks, chains of tasks, or stages, which make the NTIGen software a practical tool for simulating the industrial conditions of an assembly line. Also, the NTIGen user can generate instances with production plans, having different operation time for each task of the assembly line.

7 References

Bautista, J., Pereira, J., 2007. Ant algorithms for a time and space constrained assembly line balancing problem. European Journal of Operational Research 177, 2016–2032.

Baybars, I., 1986. A survey of exact algorithms for the simple assembly line balancing problem. Management Science 32, 909–932.

Bhattacharjee, T.K., Sahu, S., 1990. Complexity of single model assembly line balancing problems. Engineering Costs and Production Economics 18, 203– 214.

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

Chica, M., Cordón, O., Damas, S., 2011. An advanced multi-objective genetic algorithm design for the time and space assembly line balancing problem. Computers and Industrial Engineering 61, 103–117.

Chica, M., Cordón, O., Damas, S., Bautista, J., 2010. Multiobjective, constructive heuristics for the 1/3 variant of the time and space assembly line balancing problem: ACO and random greedy search. Information Sciences 180, 3465– 3487.

Chica, M., Cordón, O., Damas, S., Bautista, J., 2012. Multiobjective memetic algorithms for time and space assembly line balancing. Engineering Applications of Artificial Intelligence 25, 254–273.

Dar-El, E.M., 1975. Solving large single-model assembly line balancing problems - A comparative study. IIE Transactions 7, 302–310.

Otto, A., Otto, C., Scholl, A., 2011. SALBPgen - a systematic data generator for (simple) assembly line balancing. Jena Research Papers in Business and Economics 5.

Scholl, A., 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.