View metadata, citation and similar papers at core.ac.uk brought to you by CORE
provided by Research Papers in Economics

The Annals of "Dunarea de Jos" University of Galati
Fascicle I – 2009. Economics and Applied Informatics. Years XV - ISSN 1584-0409

A Business Process Management System based on a General Optimium Criterion

Vasile MAZILESCU
vasile.mazilescu@ugal.ro
Daniela ŞARPE
daniela.sarpe@ugal.ro
Mihaela NECULIȚĂ
mihaela.neculita@ugal.ro
Angela Eliza MICU
angelaelizamicu@yahoo.com
Dunărea de Jos University of Galati

Abstract. Business Process Management Systems (BPMS) provide a broad range of facilities to manage operational business processes. These systems should provide support for the complete Business Process Management (BPM) life-cycle [16]: (re)design, configuration, execution, control, and diagnosis of processes. BPMS can be seen as successors of Workflow Management (WFM) systems. However, already in the seventies people were working on office automation systems which are comparable with today's WFM systems. Recently, WFM vendors started to position their systems as BPMS. Our paper's goal is a proposal for a Tasks-to-Workstations Assignment Algorithm (TWAA) for assembly lines which is a special implementation of a stochastic descent technique, in the context of BPMS, especially at the control level. Both cases, single and mixed-model, are treated. For a family of product models having the same generic structure, the mixed-model assignment problem can be formulated through an equivalent single-model problem. A general optimum criterion is considered. As the assembly line balancing, this kind of optimisation problem leads to a graph partitioning problem meeting precedence and feasibility constraints. The proposed definition for the "neighbourhood" function involves an efficient way for treating the partition and precedence constraints. Moreover, the Stochastic Descent Technique (SDT) allows an implicit treatment of the feasibility constraint. The proposed algorithm converges with probability 1 to an optimal solution.

Keywords: BPMS, control assembly system, stochastic optimisation techniques, TWAA, SDT

Jel Code: D83

1. Introduction

We define BPM as follows: supporting business processes using methods, techniques, and software to design, enact, control, and analyze operational processesinvolving humans, organizations, applications, documents and other sources of information. This definition restricts BPM to operational processes, i.e.processes at the strategic level and processes that cannot be made explicit are excluded. It also follows that systems supporting BPM need to be "process aware". After all, without information about the operational processes at hand little support is possible. The BPM life-cycle has four phases (W.M.P. van der Aalst, A.H.M. ter

Hofstede, and M. Weske, 2003): (1) *Process design.* Any BPM effort requires the modeling of an existing ("as-is") or desired ("to-be") process, i.e., a *process design*. During this phase process models including various perspectives (control-flow, data-flow, organizational, sociotechnical, and operational aspects) are constructed. The only way to create a "process-aware" enterprise information system is to add knowledge about the operational processes at hand. (2) *System configuration.* Based on a process design, the process-aware enterprise information system is realized. In the traditional setting the realization would require a time-consuming and complex software development process. The traditional software development process is replaced by a configuration or assembly process. Therefore, we use the term *system configuration* for the phase in-between process design and enactment. (3) *Process enactment.* The *process enactment* phase is the phase where the process-aware enterprise information system realized in the system configuration phase is actually used. (4) *Diagnosis.* Process-aware enterprise information system have to change over time to improve performance, exploit new technologies, support new processes, and adapt to an ever changing environment. Therefore, the *diagnosis* phase is linking the process enactment phase to a new design phase.

Assembly lines are flow-line production systems which are of great importance in the industrial production of high quantity standardized commodities and more recently even gained importance in low volume production of customized products. Due to high capital requirements when installing or redesigning a line, configuration planning is of great relevance for practitioners. Accordingly, this attracted the attention of research, who tried to support practical configuration planning by suited optimization models. In spite of the great amount of extensions of basic assembly line balancing there remains a gap between requirements of real configuration problems and the status of research. This gap might result from research papers focusing on just a single or only a few practical extensions at a time. Real-world assembly systems require a lot of these extensions to be considered simultaneously. Open research challenges are identified and the practitioner is provided with hints on how to single out suited balancing procedures for his type of assembly system.

For simplicity, an assembly line is specified by a finite set of tasks, the processing time for each task, and the precedence relationships which define the permissible orderings of the tasks. Given a required output rate, the assembly line balancing (ALB) problem involves finding the minimum number of workstations and assigning assembly tasks to workstations so as to minimize the balance delay without violating the precedence relationships imposed on the problem. Since the ALB problem is a combinatorial optimization problem, the problem-solving time increases progressively with the problem size, making it impractical to find an optimal solution for large-sized problems. Alternatively, several researchers have developed heuristic and meta-heuristic algorithms to obtain *near-optimal* solutions. A fundamental capacity of an assembly line is to react and control to the occurrences of unpredictable events. The flexibility at the control level depends on tasks-to-workstations assignments, which are available, when such an event happens (Donath and Graves, 1988).

A new assignment is implemented and a new schedule of products is recalculated. Thus, the assembly line is reconfigured (He and Kusiak, 1997). The assignment problem consists in dispatching the assembly tasks to the available workstations. A Task to Workstation Assignment (TWA) must be conforming to the precedence constraints between tasks, involved by the assembly process and it must be feasible, i.e. each task can be performed by the assigned workstation. Because there are many possible assignments, the selected assignment has to satisfy an optimum criterion. An automated assembly line is controlled by a computer system. The assembly line is comprised of a plurality of machines which are each segmented intoits basic unit operations providing work stations. The work stations are then controlled by the computer system and operated asynchronously with respect to the other work stations of the assembly line. Workers in the manual assembly line have high risk of cumulative trauma disorders due to the following reasons: (1) they are required to manually perform tasks by maintaining the same working posture on a prolonged basis, (2) their body postures tend to be inappropriate especially if the assembly workstations are poorly designed or are non-adjustable, (3) they are constrained by the required output rate to produce a large quantity of products daily, and (4) they have to use specific muscle(s) repetitively. It has been reported that ergonomics is one of the important issues that are normally not considered when assigning assembly tasks to workstations.

2. The economic impact of assembly lines in Romania

In Romania, for example Unio, a manufacturer of technology for the mining and energy industry, is to team up with German company Kuka to produce assembly lines for a new Mercedes car plant in Germany. The partnership will begin next January, and the contract will be worth more than 2 million euros. The assembly lines will be built entirely and the beneficiary, Kuka, will act only as consultant. This is a big step for Unio because, through Kuka, it has entered an exclusive and demanding market. It is not easy to move from manufacturing for the mining industry to industrial robots in a mere five years. Unio currently supplies Kuka with Volvo, Opel and Ford assembly lines under contracts worth more than a million euros a year. Kuka is the number one industrial robot maker in Europe and one of the top companies in the world in its field with factories in ten countries.

Customized applications for assembly of the components of the automotive transmission is another good example for applications of assembly lines in Romania. For more than 25 years, Marposs has dedicated to its customers a combination of advanced products, technical know-how and commitment to the design, installation and support anywhere in the world, of the automatic assembly lines for automotive transmissions. The typical measurement needs are: shims determination for components positioning, pre-load on differential housing and gearboxes, meshing stations, back-lash verification and a variety of dimensional and functional inspection on the single elements of the powertrain. Usefull applications are: (1) Manual transmissions assembly line (semi-automatic and automatic), (2) Differential gear-unit assembly line (semi-automatic and automatic), (3) Differential hypoid gear set semi-automatic assembly, (4) Differential side gears/pinions automatic assembly, (5) CVT assembly line, (6) Balance shaft cassette assembly line.

Romania has a long tradition of car manufacturing: it has produced motor vehicles for about half a century, but small-scale operations and assembly lines existed even before the second world war. Today a number of motor vehicle manufacturers as well as a network of suppliers working both for local assembly lines and export are present in the market, with Renault as the main operator. The years 2002-2004 marked the recovery of the local automotive production, backed by the strong growth of the national economy, and by a significant foreign investments rate. Although it will still take time to raise income levels and for the automotive sector to fully exploit its potential, the sector has already become one of the fastest growing in the country. It is foreseen that a continued economic growth, unique tax level (16%), revenues from work abroad and a low concentration of automobiles (around 200 auto per 1000 people) will keep boosting the demand. Similarly, an elevated growth rate in construction and services should stimulate further more the demand for commercial vehicles.

A population of 22 million ranks Romania as the second largest of the new EU member states, behind Poland. Local motor vehicles production has faced an extremely important growth over the last years: in 2005 it rose up to 194,616 units, up 59.4% as compared with 2004. Passenger car production increased by 76.3%, to 174,538 units. Despite a lower pace, this positive trend was kept in 2006, when 213,597 cars were manufactured (+ 9.6%). The export of passenger cars went up by 245% to almost 59,000 units from 2004 to 2005, raising four-fold in total between 2004 and 2006. The industry employs around 90,000 people including both motor vehicle manufacturers and component producers. One should add a significant number of people working for the tier two or tier three suppliers (raw materials, sheet metal, ball bearings, assembly elements, etc). Forecasts concerning the Romanian market and national production evolution in 2007 and 2008 are quite promising. Sales on the domestic market will continue to grow by approx. 10-15% per year in 2007, 2008. The diversifying trend for the demand will continue, favorising an increase in the import's market share for vehicles and light commercial vehicles.

From 1993 on, the western component manufacturers started to discover the advantages offered by Romania as a European manufacturing location, mainly concerning work force skills and costs. Therefore, a growing number of investors came in, producing mainly for export, with little connection to the local motor vehicle assemblers. Lately, with the advent of new manufacturers starting operations in Romania (and mainly after the arrival of Renault in 1999), a number of international suppliers decided to follow the vehicle manufacturer and to invest in order to supply directly the local assembly lines (e.g. Valeo, Johnson Controls and others). Today the domestic automotive component manufacturers are split in two main groups, the first one being composed by the traditional local suppliers in operation from the '70s and '80s (the combined turnover of these companies has been € 622.9 million in 2005, with a work force of around 36,500 people), and the second one by the companies with foreign ownership (there are 73 companies of this kind in operation, with a combined turnover in 2005 of € 1083.68 million, and around 52,500 people employed). It is expected that in the next years the national network of auto parts suppliers will expand and fortify, animated especially by the Logan program. There is a considerable room for improvement in terms of R&D performances.

The national R&D system in the autoparts sector is subdimensioned and insufficiently modernised, and access for the local parts research to the regional or international research networks is at the present stage very difficult. However, the national authorities have recently picked up these concerns, planning to dedicate a special attention to the issue. There are good indicators that the local component of R&D is starting to develop, sustaining the durable modernisation of the sector.

Another important example of assembly lines in Romania is successful installation of PC assembly line. A member of the Altex Group, has held the official opening ceremony of its newly installed PC assembly line on 2006. The line, designed and built in cooperation between FlexLink and Mach Foreign Trade Distribution SRL, is based on FlexLink's Dynamic Assembly System (DAS). The turn key delivery included the complete logistics around PC production from component unpacking, kitting, case assembly, PC flashing and testing, and packaging of the complete PC system. All PC's are made to customer order, and the line, which was financed by Raiffeissen Leasing Romania SRL, has the capacity to produce one PC per minute. The decision by CES to go for the FlexLink solution was made in only a little over a month. The main reason why CES decided to invest in this line was the quality improvement opportunity and the track and tracing possibility offered by the sophisticated DAS software, as well as the flexibility in terms of volume and product mix. The FlexLink solution makes it possible for CES to easily switch to production of other similar products, such as DVD players or mobile phones.

3. Our problem

Intuitively, for any product sequence given by a scheduling procedure, a well-balanced assembly line will diminish the waiting time of products in the buffers of workstations. That is why, generally, the optimum criterion is the minimization of the idle time. In this case, the assignment problem is very similar to the Assembly Line Balancing (ALB) problem, despite the fact ALB is used for designing the assembly line, i.e. to produce a first rough layout of the line. Technically speaking, as optimisation algorithm, ALB problem is equivalent to a TWA problem whose optimum criteria is the best line balancing. The main difference between ALB and TWA is concerning the number of workstations. For ALB, this number is unknown and the algorithm will try to minimize it, because, when the idle time is minimized, the number of workstations is also minimized implicitly. Nevertheless, the two problems use the same optimization techniques. In (Ghosh and Gagnon, 1989) we have a very good review concerning the approaches and the optimization techniques used by ALB both for Single-Model (SMALB) and Mixed-Model (MMALB) assembly lines. For single-model ALB problem, the assembly plan is represented generally by a precedence graph. The MMALB problem involves a further difficulty, that is the construction of an appropriate assembly plan representing all the product models. Thompoulos (Thomopoulos, 1970) constructs a precedence graph for the entire family of models and transforms a MMALB problem into a SMALB problem. Holmes (Holmes, 1987) considers the product models as variants and use a linear assembly plan for each of them. A global plan is obtained by merging these plans.

For the mixed-model case, the TWA problem needs also an assembly plan for the entire family of models. We have proposed in (Mazilescu, 2003) the assembly graph that describes the assembly process for a single product. In this paper we generalize this concept for a family of product models and we construct, under some hypothesis, a precedence graph for the entire family. That allows us to express the mixed-model TWA problem through an equivalent single-model TWA problem. Thus, the same algorithm treats the two cases. As the ALB problem, the TWA problem is NP-hard because it is a graph partitioning problem with an optimum criteria. An algorithm, which guarantees a global minimum for our problem, should have an exponential computational complexity. Such an algorithm cannot be used in a real time scheduling system, especially for large assembly systems.

In practice, a "good solution" given by an approximation algorithm may be satisfactory. The stochastic descent methods have already been used for solving ALB problem. For example, genetics algorithms are used for the single-model case in the papers (Falkenauer, 1996) and for mixed-model case in (Rekiek et al., 1997) and simulated annealing technique is used in (Jung, 1997) for solving a stochastic ALB problem. In this paper we propose a special implementation of stochastic descent technique called "Kangaroo" (Fleury, 1995). In order to define the "neighbourhood" functions, elementary shift operators are introduced. They allow an efficient way to meet the partition and the precedence constraints.

The feasibility constraint is treated implicitly, because the unfeasible solutions are penalized by the objective function. The accessibility of any optimal solution from any initial solution is proved. This is the main property, which guarantees the convergence with probability 1 of the proposed algorithm to an optimal solution.

4. The case study

4.1. Tasks-to-workstations assignment problem for single-model assembly lines

In this section, we show that a tasks-to-workstations assignment is a graph-partitioning problem that, moreover, has an optimality criterion. We consider an assembly line formed by M operational workstations: W_i, i=1, ... , M. The assembly process for a given product is characterised by a task set $Q = \{t_1, t_2, \dots, t_N\}$ made up of N assembly tasks. The assembly process involves some precedence constraints between tasks. So, the task set is partially ordered by a precedence relation described by a digraph G=(Q, U), $U \subset Q \times Q$. Nodes represent generic assembly tasks and arrows represent precedence relations between tasks. An arrow from node n_1 to node n_2 indicates that task n_1 must be completed before task n_2 begins.

The time needed by the workstation W_i to perform the task t_j is τ_{ij}. It is possible that $\tau_{ij} = \infty$; that means that the workstation W_i can not perform the task t_j. In order to define a tasks-to-workstations assignment, let P_i be the task set assigned to the workstation W_i. Obviously, the sets P_i, i=1,...,M determine a partition of the task set Q. Such a partition is denoted by u:

$$u = \{P_1, P_2, \ldots, P_M\}, \text{ with } P_i \subset Q,\ i = 1, \ldots, M$$

A tasks-to-workstations assignment has to meet the precedence constraints, i.e.

$$\forall\, t_{j_1}, t_{j_2} \in Q,\ j_1 \neq j_2, \text{ with } t_{j_1} \in P_{i_1}, t_{j_2} \in P_{i_2}, (t_{j_1} \text{ precedes } t_{j_2} \text{ in G}) \Rightarrow i_1 \leq i_2$$

Obviously, an assignment has also to meet the feasibility constraint, i.e.

$$\tau_{ij} < \infty,\ \forall i \in \{1, \ldots, M\},\ \forall j \in P_i$$

A proper assignment is a partition u which meets the precedence and feasibility constraints. Generally, there are many proper assignments. One may choose the assignment that minimises an objective function f(u):

$$\min_u\ f(u) \qquad (1)$$

A possible optimality criteria is to minimise the cycle time of the assembly line. The assembly line's cycle time t_c is the maximum work content of a workstation. In this case, the objective function is

$$f_1(u) = t_c(u) = \max_{i=1,\ldots,M} \sum_{j \in P_i} \tau_{ij} \qquad (2)$$

Another objective function specific for ALB problem is

$$f_2(u) = \sqrt{\sum_{i=1}^{M} (S_{max} - S_i)^2} \quad (3)$$

$$S_{max} = t_c(u)\ ;\ S_i = \sum_{j \in P_i} \tau_{ij}$$

So we have an optimization problem: the search of a proper assignment which meets the optimality criteria (1). This problem is obviously NP-hard, because it is a graph partitioning which satisfies the optimality criteria (1). One may notice that, if the processing time of a task t_j does not depend on the assigned workstation W_i, that is:

$$\tau_{ij} = \tau_j,$$

and f(u) = f_2(u). We have the same optimality criteria as for the SMALB problem. In order to emphasize the generality of the proposed algorithm, generally in the next sections we shall not set the objective function.

4.2. Tasks-to-workstations assignment problem for mixed-model assembly lines

A mixed-model assembly line produces a family of product models, each of them following a different assembly process. Our general approach, in the case of mixed-model assembly lines, is to find an equivalent single-model TWA problem. This approach is inspired from a technique used for solving MMALB (Thomopoulos 1970, Macaskill 1972, Macaskill 1973). The most precise representation of an assembly process for a single product model is the assembly tree (Henrioud and Bourjault, 1992), because it provides the both components that define an assembly task. The assembly tree can be transformed in assembly graph having a tree structure. Generally speaking, this representation is not possible for a family of product models. In the case of iso - structural family, which means that each product model in the family has the same basic structure formed by generic parts, one may generalise this representation.

4.2.1. Mixed-model assembly plan for iso - structural family of product models

We propose an assembly plan for mixed-model assembly lines which is, in the same time, a precedence graph. An instance of a generic part in a product model may be a part, a subassembly or an empty part. A generic part will induce a generic task of the assembly process. In this case, one can define a *generic assembly graph*. An example of generic assembly graph is provided in figure 1. The nodes A, B, C, S, D, E represent generic assembly tasks. A generic task having 2 predecessors (for example the task S) represents the joining of a generic subassembly with a generic base constituent.

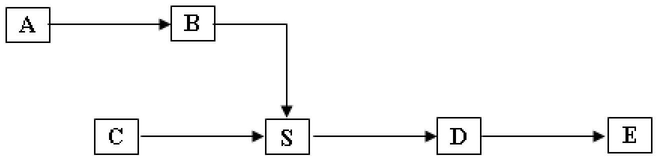

Figure 1. Example of generic assembly graph for a family of product models

The generic assembly graph can be obtained by a good analysis of the family of product models, which will establish the generic parts and the precedence constraints between parts. Assembly graph for each product model in the family can be generated from the generic assembly graph, replacing the generic tasks by their instances. Considering a family formed by 3 product models, having the generic assembly graph presented in figure 1, the figure 2 provides the assembly graph for each of them.

The generic tasks A and E are constant tasks, because their instances are the same for all products. On the other hand, the generic tasks B, C, D, S are variant tasks, because their instances are different. For example, the generic task B is replaced in model 1 by a structure (b^1_1, b^1_2, b^1_3, b^1_4) corresponding to a subassembly, and by a single task in models 2 and 3. Empty task is considered, like in the case of generic task D, for model 3. This task corresponds to an optional generic constituent whose instance may be the empty part. We consider the assembly graph of product model i is represented by a digraph $G_i=(Q_i, U_i)$ with a set of nodes Q_i and a set of arrows U_i.

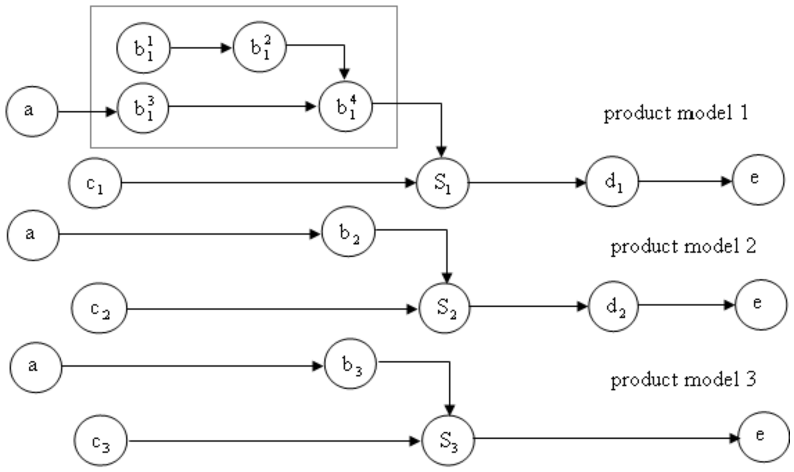

Figure 2. Assembly graphs for each product model of the family

The precedence constraints for all models of the family can be represented by a superimposed assembly graph G=(Q,U) where:

$$Q=Q_1 \cup Q_2 \cup ... \cup Q_K, \quad U=U_1 \cup U_2 \cup ... \cup U_K \qquad (4)$$

where K is the number of product models in the family. So, the task set Q defined by (4) is partially ordered by a precedence relation which is described by the superimposed assembly graph G. For the product family presented in figure 2, the superimposed graph is provided in figure 3.

The relationship between the structure of the family of product models and the assembly plan, represented by the superimposed graph G, is now explicit.

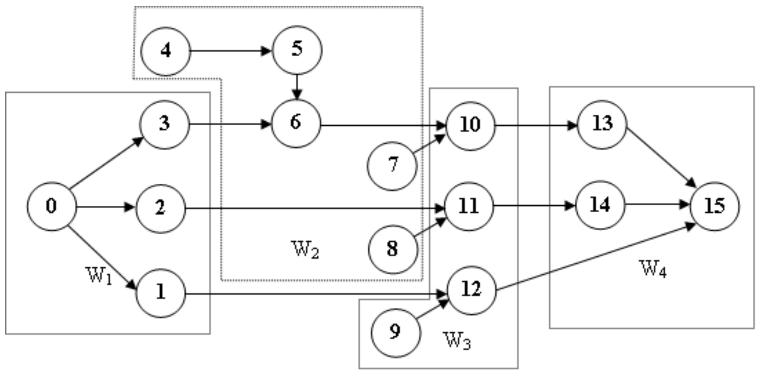

Figure 3. The superimposed graph corresponding to the family product models

4.2.2. The formulation of the assignment problem for mixed-model

Let's consider an assembly line for a family of product models, formed by M given operational workstations. A tasks-to-workstations assignment depends upon a given structure of assembly line's production; in a given time horizon, the assembly line must produce, for each product model, a given number of products. The following notation are used:

q_k = the number of products for model k, k=1,...,K
τ_j = the processing time of task t_j

$$d_{kj} = \begin{cases} \tau_j \text{, if task } t_j \text{ is required by model k} \\ 0 \text{, otherwise} \end{cases}$$

The total aggregation hypothesis states that all instances of a generic task are performed by the same workstation. More explicitly, for all product models and for all products the same assembly operation is performed in the same workstation. It is a realistic hypothesis, because, generally, workstations have specialised operators. To each task j, one may associate the aggregated processing time T_j, computed as follows:

$$T_j = \sum_{k=1}^{K} q_k . d_{kj} \text{ , } j = 1,..., N \quad (5)$$

A proper assignment is a partition u which meets the precedence constraints corresponding to the superimposed assembly graph G=(Q,U) and the feasibility constraint. Now, one can ask to find a proper assignment, which minimises the maximum work content, or satisfies another optimality criteria. So, we have a problem equivalent to a single-model TWA problem. It has, as a precedence graph, the superimposed assembly graph and, as processing times, the aggregated times given by formula (5). This result can by extended to assembly lines for which the total aggregation hypothesis is not adopted. In this case, the number of tasks and the complexity of the precedence graph are increasing (see Thomopoulos 1970, Macaskill 1973).

4.3. A general stochastic descent technique

An algorithm, which guarantees a global minimum for our problem, should have an exponential computational complexity. Hence sub-optimal solutions are generally preferred to optimal solutions. That is why we propose to use an approximation technique based on stochastic descent, called "Kangaroo" (Fleury, 1995), inspired by the simulated annealing method, but having a quite different searching strategy. As in the case of simulated annealing, the "Kangaroo" method is implemented by an iterative procedure which minimises an objective function f(u). A current solution u of the considered problem is replaced by a better one situated in its neighbourhood N(u), using a random selection. If a new improvement is no longer possible, a "jump" procedure is performed, in order to escape from the attraction of a local minimum. This procedure can use a different neighbourhood definition N' (u) for the random jumping. The "Kangaroo" algorithm is described hereafter.

```
algorithm Kangaroo
begin
Initialise A;
Choose an initial solution u;
        $c \leftarrow 1$ ; $u^* \leftarrow u$ ;
        repeat
        if c<A then descent(u, u*, c) ;
                else jump(u, u*, c) ;
        until "stop criteria";
end ;

procedure descent(u, u*, c)
begin
```

```
          Random generation of v in N(u) ;
          If f(v) ≤ f(u)
                    then {If f(v)<f(u) then {c ← 0 ;
                              if f(v)<f(u*) then u* ← v}
                    u ← v ;}
          c ← c +1 ;
end descent ;
procedure jump(u, u*, c)
          Random generation of v in N'(u);
          If f(u) ≠ f(v) then {c ← 0;
                              if f(v) < f(u*) then u* ← v}
          c ← c+1;
          u ← v;
end jump;
```

The parameter A is the maximum number of iterations without improvement of the current solution and u* is the best solution found until the current iteration. The variable c counts the number of iterations between two improvements of the objective function. The stop criterion is either a maximum iteration number or a bottom bound of the objective function f(.). If some conditions presented in section 6 are fulfilled, then the best solution u* converges with probability 1 to a global minimum.

4.4. Neighbourhood's definition and precedence constraints

For the TWA problem, the current solution u of the iterative process is a proper assignment. Hence, it must be a partition of task set Q meeting the precedence constraints described by the precedence graph G. The feasibility constraint will be treated implicitly, because a partition u, which does not meet this constraint, is penalized by the objective function. If $□_{ij}$ = □ for some pairs (i, j), we have to choose the form of objective function such that f(u)=□. Such is the case with the objective functions (2) and (3). So, a non-feasible partition will never be assigned to u*, but it is accepted as a current solution of the iterative process, in order to assure the convergence of the algorithm (see section 6). An important advantage of the "Kangaroo" method is the fact that it works even for objective function with infinite values. This is not the case, for example, with simulated annealing method. The procedures **descent** and **jump** use, in principle, two different definitions for the neighbourhoods N(u) and N'(u). But N(u) and N'(u) may be the same in some implementations of the "Kangaroo" algorithm. Firstly, we shall consider that

$$N(u)=N'(u).$$

Hereafter, we propose a specific definition of the neighbourhoods N(u) and N'(u) for TWA problem. When one uses a deterministic optimisation algorithm (like as exhaustive search, dynamic programming, branch and bound), the partition constraint is the reason of the exponential complexity. Families of task set, that are not partitions, are generated as intermediary states and partition tests are necessary. The proposed algorithm generates only partitions as intermediary solutions, because N(u) contains only partitions of the task set Q. With the notation of section 2, a partition is:

$$u=\{P_1, P_2,..., P_M\}.$$

Let us consider a shift-left operator:

$$(u, t_j) \xrightarrow{shl} u',$$

which generates a partition u' when it is applied to a partition u for a task t_j. This operator moves the task $t_j \in P_{io}$, $i_o>1$ from the set P_{io} to the set P_{io}-1. If we denote $u'=\{P'_1, P'_2,...,P'_M\}$ it holds:

$$P'_{io-1}=P_{io-1} \cup \{t_j\} \ ; \ P'_{io}=P_{io} - \{t_j\}$$

$$P'_i = P_i, \text{ for } i \neq i_o-1 \text{ and } i \neq i_o \ ;$$

If a partition u meets the precedence constraints and if all the direct predecessors of the task t_j belong to sets P_i, $i<i_o$, then u' will also meet the precedence constraints. Moreover, u' is obviously a partition of Q. If this condition is not met, i.e. if there is a direct predecessor of the task t_j belonging to P_{io}, we consider that the shift-left operator is not well defined. For example, an assembly line with 4 workstations has to process the tasks described by the superimposed graph presented in fig.3. Let us consider the assignment corresponding to the partition

$$u = \{P_1, P_2, P_3, P_4\}, \text{ with } P_1 = \{0, 1, 2, 3\},$$

$$P_2 = \{4, 5, 6, 7, 8\}, P_3 = \{9, 10, 11, 12\}, P_4 = \{13, 14, 15\}.$$

The task 10 can be shifted left from workstation 3 to workstation 2, because this moving meets the precedence constraints, i.e. tasks 6 and 7 belong to workstation 2. The resulting partition is $u'=\{\{0,1,2,3\}, \{4, 5, 6, 7, 8, 10\},\{9, 11, 12\},\{13, 14, 15\}\}$, $u' \in N(u)$.

For the task 6 the shift left operator is not well defined. In the same way, a shift-right operator,

$$(u, t_j) \xrightarrow{shr} u',$$

can be defined as below:

$$t_j \in P_{io}, i_o<M \ ; \ P'_{io+1}=P_{io+1} \cup \{t_j\} \ ; \ P'_{io}=P_{io} - \{t_j\}$$

$$P'_i = P_i \ , \text{ for } i \neq i_o+1 \text{ and } i \neq i_o \ ;$$

If partition u meets the precedence constraints and if all the direct successors of the task t_j belong to sets P_i, $i>i_o$, then u' will also meet the precedence constraints. Moreover, u' is obviously a partition of Q. If there is a direct successor of the task t_j belonging to P_{io}, we consider that the shift-right operator is not well defined.

From now on, a shift operator is either a shift-left, or a shift-right operator. To apply such an operator to a partition u for a task t_j, one has to verify the constraints concerning the direct predecessors or the direct successors of t_j. The main idea of the proposed algorithm is that the next solution is obtained from the current solution by a well defined shift operator. *If the initial partition of the iterative process meets the precedence constraints, then the algorithm will generate only partitions meeting these constraints, without additional tests.* Hence, this algorithm will have a weak computational complexity. In the current partition u, there are several tasks which can be shifted.

The proposed algorithm forms a list L_u composed by these tasks. A task may be present two times in L_u, when it can be shifted left and right (the both operators being well defined). The algorithm constructs also the list D_u, which contains the index of the corresponding destination workstation.

For example, if u is the partition illustrated in figure 3, it holds:

$$L_u=[1, 2, 3, 4, 7, 8, 6, 7, 8, 9, 10, 11, 10, 11, 12, 13, 14]$$

$$D_u=[2, 2, 2, 1, 1, 1, 3, 3, 3, 2, 2,2, 4, 4, 4, 3, 3]$$

So, the task $L_u(k)$ can be shifted from its workstation to workstation $D_u(k)$, $1\leq k \leq card(L_u)$. The neighbourhood N'(u) is the set of partitions u' obtained by shifting all tasks belonging to L_u:

$$N'(u) = \{u' \mid (u,\ L_u(k)) \to u',\ 1 \leq k \leq card(L_u)\} \qquad (6)$$
$$N(u) = N'(u)$$

So, to generate a partition $u' \in N(u)$ is equivalent to choose a task in the list L_u. The random selection of the task that will be shifted is done in accordance with an uniform distribution law. With the elements described above, the **descent** procedure for our assignment problem becomes:

```
procedure descent_TWA(u, u*, c)
begin
- Construct the list $L_u$ of tasks which can be shift and the corresponding $D_u$ list ;
//possible integration of heuristic elements in order to reduce the length of $L_u$ ;
- Random selection of a task $t_j$ in $L_u$;
- Construction of the partition v by shifting $t_j$ : $(u,\ t_j) \to v$
If $f(v)\leq f(u)$
    then {If $f(v)<f(u)$ then {$c\leftarrow 0$ ;
                               if $f(v)<f(u^*)$ then $u^* \leftarrow v$ }
        $u \leftarrow v$ }
$c\leftarrow c+1$ ;
end descent_TWA ;
```

4.5. Convergence of the algorithm

The series $\{u_n\}$ of the current solutions in the iterative process, realised by a particular execution of the presented algorithm, is a realization of a Markov chain (Fleury, 1993). At iteration n, u^*_n is the best encountered solution. One may consider the series $\{u^*_n\}$ of the best solutions of the iterative process; that is also a realization of Markov chain. In this section, we shall prove that the stochastic process (u^*_n) converges with probability 1 to a global minimum. The main point is the accessibility property of the neighbourhood function N'(u). Let X be the partition set meeting the precedence constraint (i.e. the proper assignment set) and let X_{min} be the optimal partition set (i.e. the set of proper assignments which minimize the objective function). Obviously, X and X_{min} are finite sets.

Proposition 1. (Accessibility) If $u_i, u_f \in X$ there is a finite sequence of tasks $t_1, t_2,..., t_k$ such that

$$u_i \xrightarrow{sh(u_i,t_1)} u_1 \xrightarrow{sh(u_1,t_2)} u_2 \cdots$$
$$\cdots \xrightarrow{sh(u_{k-1},t_k)} u_k = u_f,$$

where sh(.) is either shl(.)or shr(.) operator (for the proof see Mazilescu 2003).

Notice that the probability of the transfer mentioned in proposition 1 is not equal to zero, because the probability of a single transition is determined by an uniform random selection over a finite number of possibilities. As we have stated in section 5, sometimes, the partitions obtained in the iterative process are not feasible assignments. We must consider such partitions, in order to assure the accessibility property of the neighbourhood function N'(u).

Proposition 2. The TWA algorithm with the neighbourhood functions N'(.) and N(.) defined by equation (6) is convergent with probability 1 to an optimal solution (see Mazilescu, 2003).

A very important aspect is the fact that the convergence with probability 1 requires accessibility in the sense of neighbourhood N'(.), which is used by jump procedure. Hence, deterministic heuristics may be integrated in descent procedure in order to guide the search of an optimum solution. This is equivalent to modifying the definition of neighbourhood N(u) such that

$$N(u) \subset N'(u).$$

4.6. Computational aspects

The iterative procedure begins with any initial assignment. Obviously, for the large assembly systems, one may choose an initial solution that corresponds to a real tasks-to-workstations assignment. So, the proposed algorithm will find faster a good or an optimal solution. A good initial solution can be obtained by the Sequential Assignment Procedure (SAP) [KUS 94]. This procedure aims at maximizing equality of average processing time, among all stations. An implementation of the assignment algorithm was realised in accordance with the elements presented in section 5. It is a general variant of the algorithm, because any objective function can be used. One can improve the convergence speed of the assignment algorithm by adding some deterministic heuristics in descent procedure. So, the definition of the neighbourhood N(u) will be modified in order to guide the search of an optimum solution. Obviously, the heuristics added in descent procedure depend on the objective function.

For example, if equation (2) or (3) gives the objective function, the list L_u may contain only the tasks belonging to the maximum work content workstations. Intuitively, to decrease the line's cycle time, one have to shift a task belonging to such a workstation. In a second variant of the proposed algorithm, we have implemented this heuristic. In the same time, the complexity of each iteration has been improved by reducing the number of tasks of the list L_u. The two variants of assignment algorithm have been coded in C++ and executed on a PC workstation.

Let us consider a school example whose precedence graph is illustrated in figure 3:

$$f(u)=t_c(u)\ ;\ M=4\ ;\ N=16\ \tau_o=45,\ \tau_1=10,$$
$$\tau_2=20,\ \tau_3=15,\ \tau_4=10,\ \tau_5=10,\ \tau_6=20,\ \tau_7=10,\ \tau_8=20,\ \tau_9=15,\ \tau_{10}=20,\ \tau_{11}=40,\ \tau_{12}=30,\ \tau_{13}=10,\ \tau_{14}=20,$$
$$\tau_{15}=45$$

Considering an initial assignment given by SAP:

$$P_1=\{0, 1, 2, 4\},\ P_2=\{3, 5, 6, 7, 8, 9\},$$
$$P_3=\{10, 11, 12, 13\},\ P_4=\{14, 15\},$$

and the parameter A set to 6, the proposed algorithm finds, in 18 up to 74 iterations, the optimal solution :

$$P^*_1=\{0, 2, 8\},\ P^*_2=\{3, 4, 11, 14\},\ P^*_3=\{1, 5, 6, 7, 9, 10\},$$
$$P^*_4=\{12, 13, 15\};\ t^*_c=85.$$

A good solution (t_c=90) was found in 10-34 iterations. An example of the objective function's evolution during the iterative process is presented in Figure 4.

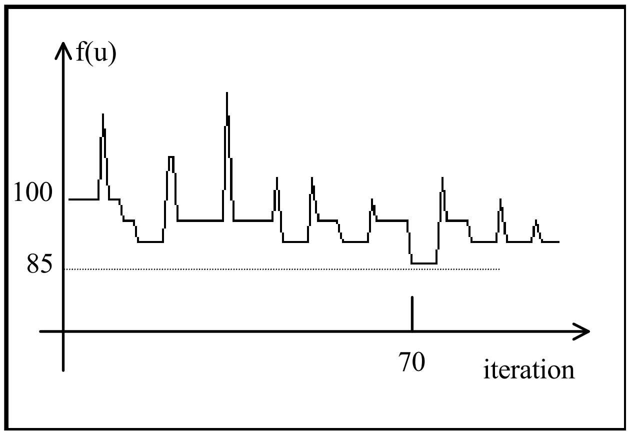

Figure 4. Evolution of the cycle time in 100 iterations

The two variants of the algorithm have been used to solve tasks-to-workstations assignment problem for large assembly systems. The test data generated are characterized by the following parameters:

(1) K, number of product models in the family: 3,...,6;
(2) q_k, k=1,...,K, production quantity for model k: 100,...,200;
(3) N, number of aggregated tasks: 16,...,128 ;
(4) T_j, j=1,...,N, aggregated processing time: 100,...,450;
(5) M, number of stations: 4,...,16;

We have considered the objective function given by equation (3). For the same value of parameter A, the second variant of the proposed algorithm is obviously faster then the first one. That means that a solution with the same value of the objective function is found, generally, in a smaller number of iterations. The results presented hereafter are obtained with this variant of the proposed algorithm.

Table 1 provides some computational results obtained for a set of problems whose optimal solutions are known. When the algorithm finds a good solution u_o instead of the optimal one u*, the approximation error is

$$\frac{f(u_O) - f(u^*)}{f(u^*)}.$$

Concerning the choice of the number of iterations without improvement of the objective function, the computational experience proved that a good value for A is:

$$A = a \cdot \left[\frac{N}{M}\right],\ a \in \{1,\ 2\} \qquad \text{([x] is the greater integer less or equal to x)}$$

For all the problems, the second variant of the algorithm has found the optimal solutions in a acceptable iteration number.

N	M	K	iteration	approximation error
32	8	3	9-17	5.8%
64	16	3	11-34	5.8%
128	16	6	9-164	5.8%
128	16	6	13-304	2.9%

Table 1. Computational results

For example, the optimal solution of a mixed-model assignment problem with 128 tasks, 16 workstations and 6 product models was found in 16394 iterations. But, after up to 164 iterations, the best solution is at 5.8% from an optimal one. The tests proved a good behaviour of the proposed assignment algorithm for real problems concerning large assembly lines. All the tests have proved that the algorithm may be used in real time scheduling module.

5. Conclusions

We have proposed an algorithm for solving TWA problem for single or mixed-model, whatever the optimal criteria is, in the context of BPMS. It is a particular implementation of a stochastic descent technique. After the formulation of the TWA problem for single-model assembly lines, we have proved that, for the iso-structural mixed-model case, there is an equivalent single-model TWA problem. The definition of the "neighbourhood" function based on elementary shift operations allowed a very efficient way for treating the partition and precedence constraints. If the initial solution is a proper assignment, the test of partition and precedence constraints is no longer necessary. Because the algorithm manages with objective functions having infinite values, the feasibility constraint is treated implicitly. Hence, the algorithm's complexity is very good. The solution space has the property of accessibility in the sense of "neighbourhood" function. The algorithm converges with probability 1 to a global minimum. This is not the case with other heuristic search methods. Its convergence speed can be improved by adding some deterministic heuristics in the descent procedure. The proposed algorithm was tested for solving assignment problems for large assembly systems. We have adopted two objective functions used also by ALB problem. The results have shown that this approach is realistic and efficient.

References

1. *Donath M., and Graves R., (1988). Flexible Assembly Systems : an approach for near real-time scheduling and routing of multiple products. International Journal of Production and Research. Vol.26, no.12, p1903-1919*
2. *Falkenauer E., (1996). A Hybrid Grouping Genetic Algorithms for Bin Packing. Journal of Heuristics, 2 : 5 - 30 Kluwer Academic Publisher, 1996*
3. *Fleury G., (1993). Méthodes stochastiques et déterministes pour les problèmes NP - difficiles. PhD Thesis, Université Blaise Pascale , Clermont - Ferrand*
4. *Fleury G., (1995). Applications des méthodes stochastiques inspirées du recuit simulé à des problèmes d'ordonnancement. Automatique Productique Informatique Industrielle, Vol. 29 - no 4 -5/1995, p445-470*
5. *Ghosh R.J., and Gagnon S., (1989). A comprehensive literature review analysis of the design, balancing and scheduling of assembly systems, International Journal of Production and Research, Vol.27, no. 4, p637-670*
6. *He D.W. and Kusiak A., (1997). Design of Assembly Systems for Modular Product. IEEE Transactions on Robotics and Automation, Vol. 13, No. 5, p646-655*
7. *Henrioud J. M., and Bourjault A., (1992). Computer aided assembly process planing. Journal Engineering Manufacture, 206 (B1), p61 - 66*
8. *Holmes C.A., (1987). Equipment Selection and Task Assignment for Multi-product Assembly System Design. Master thesis, M.I.T, USA*

9. *Jung L., (1997). A Single-Run Optimization Algorithm for Stochastic Assembly Line Balancing Problems. Journal of Manufacturing Systems, Vol.16, No. 3, p204-210*
10. *Kusiak A., (1994). Concurrent Engineering: Issues, Models, and Solution Approaches. Chapter 3, R. Dorf and A. Kusiak (Eds) , John Wiley, New York, pp.35-49*
11. *Macaskill J., (1972). Production line balancing for mixed-models lines. Management Science, 19(4), p 423-433*
12. *Macaskill J., (1973). Computer simulation for mixed-models production lines. Management Science, 20(3), p 341-348*
13. *Mazilescu V., (2003). A Processing Algorithm for an Intelligent Production System, Economy Informatics, Volume VIII, Nr 1, p. 72 - 77*
14. *Rekiek, B., Falkenauer, E., and Delchambre, A. (1997). Multi-Product Resource Planning. Proceedings of the 1997 IEEE International Symposium on Assembly and Task Planning, USA, Marina del Ray, August 7-9, 1997, p115-121*
15. *Thomopoulos, N. T. (1970). Line balancing-sequencing for the mixed-model assembly. Management Science, 14(2) B59-B7*
16. *W.M.P. van der Aalst, A.H.M. ter Hofstede, and M. Weske, (2003). Business Process Management: A Survey. In W.M.P. van der Aalst, A.H.M. ter Hofstede, and M. Weske, editors, International Conference on Business Process Management (BPM 2003), volume 2678 of Lecture Notes in Computer Science, pages 1–12. Springer-Verlag, Berlin*