An Ant-bidding Algorithm for Multistage Flowshop Scheduling Problem: Optimization and Phase Transitions

Alberto V. Donati[1], Vince Darley[2], and Bala Ramachandran[3]

[1] Joint Research Center, European Commission, Via E. Fermi 1, TP 267 21020 Ispra (VA), Italy.
Corresponding author:
alberto.donati@jrc.it

[2] Eurobios UK Ltd., 26 Farringdon Street, London EC4A 4AB, UK.
vince.darley@eurobios.com

[3] IBM T.J. Watson Research Center, Route 134, Yorktown Heights, 10598 NY, US.
rbala@us.ibm.com

Abstract

In this chapter we present the integration of an ant-based algorithm with a greedy algorithm for optimizing the scheduling of a multistage plant in the consumer packaged goods industry. The multistage manufacturing plant is comprised of different stages: mixing, storage, packing and finished goods storage, and is an extension of the classic Flowshop Scheduling Problem (FSP). We propose a new algorithm for the Multistage Flowshop Scheduling Problem (MSFSP) for finding optimized solutions. The scheduling must provide both optimal and flexible solutions to respond to fluctuations in the demand and operations of the plants while minimizing costs and times of operation. Optimization of each stage in the plant is an increasingly complex task when considering limited capacity and connectivity of the stages, and the constraints they mutually impose on each other.

We discuss how our approach may be useful not just for dynamic scheduling, but also for analyzing the design of the plant in order for it to cope optimally with changes in the demand from one cycle of production to the next. Phase transitions can be identified in a multidimensional space, where it is possible to vary the number of resources available. Lastly we discuss how one can use this approach to understand the global constraints of the problem, and the nature of phase transitions in the difficulty of finding viable solutions.

Key words: Scheduling, Optimization, Ant Colony System, Phase Transitions

This work was co-funded in a Consortium Agreement between Bios Group Inc. and Unilever during 1998–1999. The authors would also like to thank D. Gregg, S. Kauffman and R. MacDonald for useful discussions and support of this work.

1 Introduction and Problem Description

We focus in this chapter on the scheduling of operations of a multistage manufacturing plant in the consumer packaged goods industry, the core of a multistage supply chain. The production plant considered here has three main parts: making (mixers), intermediate storage (tanks), and packing lines; each stage has inputs and/or outputs, with limited connectivity with the adjacent stage. Each finished product or SKU (Stock Keeping Unit) – the final output of the production cycle – differs in terms of variant and pack size.

Given the demand for the period considered (usually one week) with the type and quantity of each SKU required for the period, the optimization consists in finding the most efficient schedule of the resources of each stage, to minimize the latest completion time on the packing lines (also called the *makespan*). In this way, the approach proposed here will globally optimize the schedule of the factory. The plant details are the following.

Making. Raw materials are subject to a number of parallel chemical processes to obtain different variants, each characterized by a number of attributes, such as color, base formulation, and dilution level. The variants produced in a mixer in batches are temporarily stored in tank facilities, and then directed to packing lines. The Making section in the plant is characterized by the number of mixers. Each mixer is then characterized by the variants it can make, the processing duration for every variant, the batch size, the cleaning/changeover times between different variants (which depend on the exact variant sequence). Finally mixers are characterized by the connectivity with the tanks, that is the maximum number of simultaneous connections with tanks for the transfer of the variant made, and their respective flow rate into the tanks.

Intermediate Storage. Storage facilities are temporary storage/caching tanks or silos, connecting the mixers to the packing lines. This stage is characterized by the number of tanks, and for each tank the variants that can be stored, its capacity with respect to the variant, the maximum number of simultaneous connections they can receive from the mixers, the maximum number of simultaneous connections for the transfer of the variant stored to the packing lines, and the setup/changeover times in changing the variant stored.

Packing Lines. The packing lines are the stage where the packing process is completed, which results in the finished product. The plant is characterized by the number of packing lines active (briefly PL). Each PL is characterized by which SKU can be packed and the relative packing rate, by the setup times due to changes in the pack size, and by the setup times due to changes in variant attributes. They are also characterized by the number, time and duration of maintenance operations, and finally by a "preferred attribute sequence" (for example, a color sequence), that might be preferred or required to follow under a specified policy. The maximum number of connections they can receive from the tanks are also specified.

Finished Goods Storage. Forecasted demand and actual demand can sometimes present differences which cannot be covered during the production span. This can be due in part to forecast errors, but mostly due to short notice changes to customer or-

ders, or manufacturing problems may lead to time/quantity problems with the availability of certain products in sufficient quantity. This stage allows for the storage of a small buffer of goods to handle those situations. Nevertheless finished goods storage is wasteful in terms of excess inventory and physical requirements, but allows one to relax the constraints on the rest of the manufacturing plant. Hence, manufacturing operations continuously endeavor to keep the finished goods inventory at a minimum, while satisfying the supply chain and end customer requirements.

Supply Chain. In principle the optimization process could be continued outside the factory to consider interactions between different factories (which may be producing the same or different products), transport of finished goods and raw ingredients. Addressing these interactions would require the analysis of the corresponding supply chain networks. We have limited ourselves here to the scheduling aspects of a given manufacturing plant in the supply chain.

For the plant type considered here, the connectivity between the three stages is limited. Taking into account all the problem details and constraints makes Multistage Factory Scheduling an increasingly complex task, besides its nature as an NP-complete problem.

Flowshop/jobshop scheduling has been widely studied in recent years and there is an abundant literature about all the possible variants/constraints of such problems.

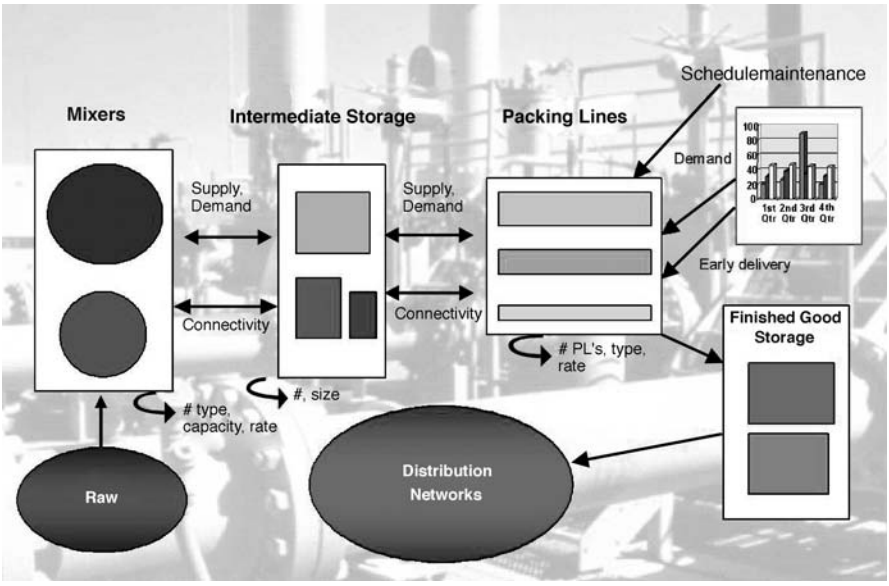

Fig. 1 Supply chain and plant representation. From left to right: mixers, intermediate storage (tanks), and packing lines. Each stage is characterized by number of resources (#), their type/capacity/rate and connectivity with the adjacent stage. The weekly demand profile of the SKUs to be done, is the input of the problem

In some cases a two-objective hierarchical minimization is carried out, the first with respect to the maximum makespan (the time from start to completion of the last job), the other minimizing the total usage time.

Mathematical programming approaches have been studied for model based planning and scheduling in the process industries. The flexible flowshop scheduling problem with more than two stages was first studied by Lee and Vairaktarakis in [1] presenting a worst-case heuristic method. More recently [2] theoretical aspects of a 3-stages flexible flowshop to complete finished products are considered and solved with an agent based model to deal with unforeseen changes in the system. In [3], mathematical programming models and solution methods are presented and strategies for implementing model-based integration of process operations are reviewed to address scheduling of multiproduct plants, integration of planning and scheduling across plant sites and design of multipurpose batch plants under uncertainty.

In general, there are two classes of process scheduling formulations – one based on discretizing time into uniform or non-uniform intervals and another based on treating time in a continuous manner. The advantages and disadvantages of these approaches are reviewed in [4]. Various heuristic/metaheuristics methods have been considered as well: a simulated annealing algorithm for a multistage manufacturing system is proposed in [5], a fast tabu search in [6], simulated annealing and genetic algorithm [7], and various other heuristics like the NEH (polynomial heuristic) presented in [8] and in the survey in [9] of machine scheduling problems. This list is limited, given the abundance of variety of methods and model/problems for these types of scheduling problems and their possible variants.

In this chapter, we present a new version of the permutation flowshop scheduling problem with limited temporary storage and with the presence of a making stage that is intimately related to the schedule. The model has been formulated to suit the process industries with liquid goods, so while on one side there is a possible loss of generality in the problem considered, on the other it constitutes an application to a real case.

We will also analyze the robustness of schedules and related phase transitions. Phase transitions in optimization problems have received considerable research attention recently in that they constitute an essential aspect when dealing with changes that might increase utilization of the system resources, such as spikes in the demand profiles from one period to another. Phase transition in problem difficulty in the general NK search problem has been studied in [10]. Phase transitions have been studied for the jobshop problem [11] and multiprocessor scheduling [12]. This work differs from the earlier phase transition research in that our focus is not on the phase transition with respect to problem difficulty, but the phase transition related to operations management in a production facility.

The chapter is organized as follows. In Sect. 2, we introduce the use of ant-based algorithms for the multistage scheduling flowshop problem. We provide the algorithm details in Sect. 3 and present computation results in Sect. 4. In the remainder of the chapter, we study the design and phase transition aspects of the problem.

2 Ants and Multistage Scheduling Flowshop Problem (MSFSP)

Recently a variety of new distributed heuristics have been developed to solve hard, NP-complete optimization problems. An ant-based algorithm uses a colony of ants, or cooperative agents, to find and reinforce optimal solutions. A variety of Ant Colony Optimization (ACO) algorithms have been proposed for discrete optimization, as discussed in [14], and have been successfully applied to the traveling salesman problem, symmetric and asymmetric ([15] and [16]), the quadratic assignment problem [17], graph-coloring problem [18], sequential ordering problem [19], flowshop [20], jobshop [21], and vehicle routing problem [24]. In particular in [20] a comparison is made with a previous ACO algorithm, the min-max ant system (MMAS, presented in [22]) for the scheduling of a single-stage flowshop problem, and an improved version is also discussed, for the standard Taillard's problems presented in [23].

The use of ants-based algorithms rather than other heuristics or metaheuristics methods, such as Genetic Algorithms, Simulated Annealing or Taboo Search, is motivated by several facts. First, it has been shown that ants algorithms can very effectively explore the search space, thus finding very good solutions in short times for many NP complete problems when combined with local search heuristics. A comparison of several strategies and metaheuristics and theoretical aspects is discussed in [25]; the effects of local search procedures have been discussed in [19] to show that the combination of an ACO and local search methods gives better results with respect to other heuristics combined with the same procedures; in particular for the sequential ordering problem (SOP), which is an asymmetric TSP with precedence constraints, it is reported that the Ant Colony System (ACS) is better than a genetic algorithm in combination with the same local search. This is due to the fact that ACS produces starting solutions that are easily improved by the local search while starting solutions produced by the genetic algorithm quickly bring the local search to a local minimum.

Besides this, ants-based algorithms can deal with a high number and variety of system constraints and it can easily be adapted to new situations where additional constraints are added to the problem. Finally, the fact that they can deal with changes in the system in a very effective way, so that they can be applied in a dynamic situation such as plant scheduling, where changes (e.g. glitches, breakdowns, additional delays, etc.) are likely to happen, so a quick reschedule is needed.

Ant-based optimization is a parallel distributed optimization process. A population of ants collectively discover and combine good pieces of different solutions. The mechanism is to encode the information not in a population of individual solutions, but in a modification of the environment: a *shared* space of pheromones. Artificial ants, searching for solutions, construct a solution step by step, and examine at each step the pheromones in the neighborhood. What step to do next will be determined by weighting the immediate reward of that move with the pheromone distribution, left by other ants in situations similar to the one they find themselves in. Once an ant has completed a solution,it calculates the fitness of that solution, and then retro-

spectively adjusts the strength of pheromone on the path it took proportionally to the fitness.

A procedure such as this, iterated over a population of ants, will provide two things: (a) a set of good solutions, and (b) a space of pheromones encoding global information.

Clearly (a) is very important. However, we will show that for real-world optimization of dynamic, uncertain, changing problems, (b) is of equal importance. It is for this reason that we have highlighted the ant algorithm as an important new approach which we believe is of great practical use. Briefly the importance of the space of pheromones is the following: if we change the problem in some small way (to reflect a change in demand, or a breakdown, delay or glitch in the manufacturing plant, or an addition to the plant), the information contained in the pheromones is still mostly relevant to the new problem. Hence a solution to the new/modified problem can be found considerably quicker. More traditional optimization procedures cannot do this and need a restart on the new problem.

The difficulty, as with applying all optimization algorithms, is in determining a suitable abstract representation: we need to define the space of pheromones and problem steps and choice procedure appropriately. It is known that for many algorithms (genetic algorithms, simulated annealing, etc.) the choice of representation is crucial in producing an effective optimization procedure. We do not expect ant algorithms to be any less representation-dependent in this regard.

An advantage of the ant algorithm, in this regard, is the relatively natural way in which expert rules of thumb may be encoded into the 'immediate reward' mentioned above. Many algorithms find it difficult to incorporate such hints or guidance into the search process. For ants algorithms this is quite straightforward, where the rules of thumb were encoded into the heuristic component. The bidding algorithm [13] is thus used for the calculation of the transition probabilities used by the ants in the solution construction procedure. This will be described in detail in the next section.

The algorithm presented here has also a number of enhancements with respect to the classic ACO. On one hand, to improve the search in the neighborhood of good solutions, a boost of pheromones is performed for high fitness solutions found; on the other, to enhance the exploration of very low probability steps in the solution construction, a random exploration is introduced.

Using these enhancements, the process of learning is constant and faster, and with proper choice of the parameter set, the optimal solutions are found within a few hundred iterations. We found that ants devote their resources very effectively to exploring a variety of high fitness solutions, while it is crucial to maintain the balance between learning and reinforcement.

3 Algorithm Scheme

The following section describes in detail the ant-bidding algorithm.

The optimization problem can be formulated as follows. Given:

1. M mixers, each having a production profile $\{V\}$ of the variants that can be done, producing each variant v of $\{V\}$ in a continuous manner at a rate $r_{\mathrm{M}}(v)$, $v = 1, \ldots, V$ (we approximate batch mixing operations by assigning a rate equal to batch size and batch duration); we also define the variant changeover/cleanup times $t^{\mathrm{M}}_{\mathrm{variant}}(i, j)$ to change to variant j after variant i, $t^{\mathrm{M}}_{\mathrm{conc}}(i, j)$ and an additional changeover time due to some variant attributes (such concentration, or white/not white type); the maximum number of simultaneous connections to the tanks are also defined;
2. T storage tanks, each having a capacity of Cv, $v = 1, \ldots, V$, the changeover times $t^{\mathrm{T}}_{\mathrm{setup}}(i, j)$ to store variant j after variant i, the maximum number of simultaneous incoming connections from the mixers, and the maximum number of simultaneous connections to the packing lines;
3. PL packing lines, each having a packing profile $\{S\}$ of the SKU that can be packed, and for each a packing rate $r_{\mathrm{PL}}(s)$, $s = 1, \ldots, S$, the variant changeover time $t^{\mathrm{PL}}_{\mathrm{variant}}(i, j)$ of packing a SKU having variant j after one having variant i, the pack-size changeover time $t^{\mathrm{PL}}_{\mathrm{size}}(i, j)$ of packing SKU j after i having a different pack-size, the maximum number of simultaneous connections from the tanks, the number n of scheduled maintenance operations with their intervals $[t_i, t_f]_k$, $k = 1, \ldots, n$ and the preferred attribute sequence of length m, $\{A^{PL}_1, \ldots, A^{PL}_m\}$, which is also PL dependent;
4. the demand profile of the S SKUs to complete for the period, where for each is specified a variant and pack size and the quantity to be done, $D(s)$, $s = 1, \ldots, S$;

find the assignment and sequence of the V variants on the mixers and of SKUs on the packing lines that satisfy all the constraints (in particular limited number and capacity of the tanks, limited connectivity, preferred attribute sequence, clean/up changing times) and such to minimize the makespan, calculated as the time when the last SKU to be done is packed, since the beginning of the operations. In the following we will refer to *task* or *job* to indicate either the making of a variant or to the packing of an SKU.

In the ant algorithm scheme, each step is the completion of an assignment task. There are then two types of steps, 1. the making of a variant in a mixer (M-step), 2. the packing of a SKU on a packing line (P-step).

The problem is represented by two acyclic diagraphs $G(\mathbf{n}, \mathbf{e})$ where the nodes $\mathbf{n}$ represent the tasks to be completed and edges express a temporal sequence relation: $\mathbf{e}(i, j)$ = task j follows task i. The pheromones φ are represented with two distinct matrices relative to the two different types of tasks (making and packing). The element $\varphi(i, j)$ on each edge represents at any time the pheromone relative to the convenience of doing the job j immediately after job i on any resource. Moreover we have introduced at the beginning of the schedule on each resource (M mixers and PL packing lines) *virtual* tasks, to be able to store information also about the first (real) job, so that the first tasks scheduled are also associated with the pheromones (those connecting with the *virtual* tasks).

The dimension of the M-pheromones matrix is then $(V + M) \cdot V$ and that for the P-pheromones is $(S + PL) \cdot S$.

A. Initialization

The problem data is read in, the pheromones are set to a uniform value. The level of pheromones during the iterations is never allowed to drop below a minimum threshold and never be higher than a maximum level. This procedure allows the ants, on one hand, to never exclude zero-pheromone edges, and on the other, to never accumulate high amounts of pheromones on one edge, avoiding the formation of high concentration-pheromone trails. This may slow down the algorithm for very large problems, where it can be reasonable to remove those edges that are rarely used, being relative to task that are topologically distant (such as cities in the TSP).

B. Propagation of Ants

An iteration of the algorithm consists in the initialization and propagation of a number of artificial ants (usually 5 to 10). Each ant of the group completes a solution by choosing a sequence of steps, until no SKU is left to be done; since the problem size is $N = V + S$, this will involve N such steps for each ant, that is V M-steps and S P-steps during the construction procedure. Each ant maintains an updated taboo list, a Boolean list of $V + S$ elements to know what has been done and what needs to be completed. During propagation the pheromones are not changed, and only when the last ant of the group has finished, are the pheromones are updated.

Each artificial ant completes the following constructive procedure:

· **Choose a resource**

The ant progressively chooses which *resource* is available for the scheduling of the next job; with the term *resource* we indicate one of the following two:

1. a mixer and an empty tank,
2. a packing line and a not-empty tank.

The resource is chosen by examining among all the resources and jobs to complete, which one is available or will be available first, in consideration of the constraints; besides some tasks can be done on some *resource* but not on others (e.g. a mixer is available but it cannot handle any of the variants left to be done), in this case the resource is not considered as an available one. So, if a mixer and an empty tank are available, an M-step will be performed, or if a packing line and a not-empty tank are available, a P-step will be performed.

· **Choose a task**

M-step. Once a mixer and an empty tank have been identified, the bids for the variants (*MBids*) that still need to be done and that can be done on this resource, are calculated.

The bids of the variants are given by:

$$MBid(v) = \sum_{s=1,\ldots,S} PBid(s_v) \tag{1}$$

where v is the variant considered, s is the index for the SKUs s_v having this variant and *PBid* is the bid of SKU s over all the possible packing lines where the SKU can

be done, finding the one with the highest value, as calculated by:

$$PBid(s_v) = \frac{D(s_v)}{T'} \tag{2}$$

where $D(s_v)$ is the demand of the SKU s, and the term T' includes the delay to wait for that PL to be available, the setup times, and the time to process the SKU: $D(s_v)/r_{PL}(s)$ where $r_{PL}(s)$ is the packing rate of SKU s on the packing line.

Note that in the term *PBid* we have not taken into account any setup/changeover times for the mixer or tank, or other delays due to mixer and tank availability and the setup times for the packing lines: these quantities are not known at this time, being dependent on the schedule and on which packing line the specific SKU that will be scheduled.

The *MBids* are finally scaled by a proper factor to be in the order of magnitude of the pheromones. The transition probabilities of making variant j after making the last variant i are given by:

$$P(j) \propto \frac{\varphi_V(i,j)^{\alpha} \cdot MBid(v_j)^{\beta}}{(1 + T_{\text{setup}}(i,j))^{\gamma}} \tag{3}$$

where φ_V are the M-pheromones, $T_{\text{setup}}(i,j)$ is the setup/changeover time due to the mixer and tank and it is computed as the maximum of the setup times $t^M_{\text{variant}}(i,j)$ + $t^M_{\text{conc}}(i,j)$ and $t^T_{\text{setup}}(i,j)$, considering the time when the mixer and tank become available respectively (they are usually not available at the same time, so overlapping in the setup times occurs); the factors α, β and γ are introduced to adjust the relative weights of these three components, the pheromones versus the heuristic factors (the bids and setup times).

The transition probabilities are used in the following ways:

1. Greedy step: choose the j that has the maximum value of probability
2. Probabilistic step: choose the j with a probability distribution given by (3)
3. Random step: choose j randomly.

Which type of step to make next is determined by two fixed cutoff parameters q_o and r_o with $q_o < r_o$. At each step, a random number r is generated in $[0, 1]$; the ant makes a greedy step if $r < q_o$, else if $r < r_o$ the ant makes a probabilistic step, otherwise it makes a random step.

Once the step has been made (that is a task has been chosen) and the next task j has been selected, the necessary system variables are updated and the mixer starts to make the variant j, filling the tank(s) selected.

The parameters q_o and r_o are used to regulate the process exploration versus exploitation, that is to say, intensification versus diversification.

P-step. If the next available resource is a packing line and a not empty tank is found, the task will be to make a SKU on that packing line. In other words, the packing line has been chosen at this point, but the tank not yet, because there might be several not

empty tanks. Before choosing the tank it is then necessary to calculate the probabilities for the SKUs, which will determine which variant (and then which tank) to use. Of course for the SKU to be considered in the bids, its corresponding variant must be present in some tank. Then the value of the bid is given by:

$$PBid(s) = \frac{D(s)}{T} \tag{4}$$

where $D(s)$ is the demand of the SKU s for the period, and T here is the resource allotment time, that is the total time required on the chosen resources to complete the SKU, and includes setup times of the packing lines, due to changes in the pack size, the presence of maintenance cycles (that will block the packing and shift its termination at a time subsequent to the end of the maintenance), the processing rates of the packing line for the SKU. In the calculation are included also the delays due to the following fact: another SKU with the same variant might have already been scheduled on one or more different packing line(s) and the variant quantity in the tank may not be enough to start packing an SKU using that variant. In the worst situation, it is necessary to introduce a delay to the start of the packing. The calculation of the exact delay is relatively expensive computationally (when there is multiple inflow and outflow to a single tank, for example), so in this calculation we only consider an approximate delay D, that guarantees the feasibility of the SKU.

The resource allotment time then given by:

$$T = \max(t_{\text{available}} - t, 0) + T_{\text{setup}} + t(s) + \Delta \tag{5}$$

where t is the current time, $t_{\text{available}}$ is the time when the packing line will finish the previous job and will be available, T_{setup} is the setup time (for changing pack size and/or variant), $t(s) = D(s)/r_{PL}(s)$ is the processing time of SKU s on the chosen PL; finally Δ is the approximate estimation of extra delays (approximate in the sense that the real delay can change due to availability of liquid in the tank and it will be computed only when the SKU has been chosen), and it includes also possible additional penalties due to the presence of maintenance during the packing task, since in this case, the packing has to be interrupted.

Then the transition probability for doing SKU j after SKU i, is then given by:

$$P(j) \propto \frac{\varphi_{SKU}(i,j)^{\alpha} \cdot PBid(s_j)^{\beta}}{(1 + \mathrm{T}_{\text{setup}}(i,j))^{\gamma}} \tag{6}$$

where φ_{SKU} are the P-pheromones and $T_{\text{setup}}(i,j)$ is the adjusted setup time for doing j after i (that is, it is calculated considering variant and size changeovers from i to j). Again, the factors α, β and γ are introduced to adjust the relative weights of the heuristic components. And like before, which type of step to make next (that is if greedy, probabilistic or random) is set by the two parameters q_o and r_o.

C. Calculation of the Fitness

When all the SKUs in the demand profile have been completed (assigned to the packing lines), the schedule is complete. A complete solution is obtained and the fitness is calculated.

The fitness is the inverse of the maximum makespan MS over the packing lines, that is the time when the last packing line ends the last SKU:

$$f = \frac{q}{\max\limits_{i=1,\ldots,PL}(MS)} \tag{7}$$

where q is a scaling factor depending on the size of the problem, in order to maintain the fitness function (used to update the pheromones) in the appropriate order of magnitude.

D. Update of the Pheromones

The pheromone matrices are updated by evaporation and deposition processes: all pheromones evaporate at a rate $(1-\rho)$, while the pheromones on the solutions found (the paths of the ants) are augmented proportionally to the fitness (the goodness) of the solution found, according to the following equation:

$$\varphi(i,j) \leftarrow \rho \cdot \varphi(i,j) + \varepsilon \cdot \sum_{a=1\ldots nAnts} f(a)\big|_{(i,j)\in S_a} \tag{8}$$

where *nAnts* is the number of ants circulating at a time (between pheromones updates), S_a is the solution found by the ant a, $f(a)$ is the fitness of the solution S_a,

- Read the problem input data
- Initialize pheromones φ_{SKU} and φ_V
- for *N_it* times repeat:
  - Initialize and propagate the colony, repeating the following (nAnts) times:
    - initialize one ant a, and its taboo list.
      Repeat the following steps (tasks) completing the schedule:
      - Which step-type? Choose what resource will be available first.
        Do a M-Step if it is a (mixer+tank), do a P-Step if it is a packing (line+tank).
      - Calculate the bids, and then the transition probabilities.
      - Chose the step type, throwing a random number r:
        1. greedy step, if r< p_0
        2. probabilistic step, if r< r_0
        3. random step, otherwise.
      - Choose the task.
      - Take the step and update the system state.
    - Calculate the fitness f(a)
    - If fitness is greater than the best fitness, best fitness = f(a), store the solution.
  - evaporate the pheromones everywhere
  - increment the pheromones of the solution found.
- return the best solution found.

Fig. 2 Outline of the ants-bidding algorithm

which contributes to the increment on the edges $(i, j) \in S_a$. Here ϵ is constant, and ρ is also referred to as the pheromone persistency constant. If the fitness is close enough (usually 5%) or greater than the best fitness found so far, the pheromones on the edges of S_a are incremented by ϵ_{boost} instead of ϵ in Equation (8), a parameter that is usually set to one or two orders of magnitude larger than ϵ. In particular, if the fitness of a solution is larger than the fitness of the best solution found so far, the solution and its fitness are stored as the new best. The iteration then continues from **b**. with a new colony, until all the $\mathbf{N_{it}}$ iterations are completed. The scheme of the algorithm is presented in Fig. 2.

4 Further Enhancements

In this section we discuss some additional aspects/enhancements that have been included in the model.

A. Attribute Sequence

We may encounter additional constraints in production scheduling in the real world. This may arise due to the need to account for aspects of the problem that have not been explicitly modeled, such as to minimize clean up procedures in order to reduce waste and environmental impact. For example, we may impose an additional constraint that the SKU must follow a specified color sequence and/or pack size sequence, called attribute sequence, which depends on the packing line considered. In such cases, only the SKUs whose attribute is the correct one for the current attribute sequence of that packing line will have a non zero bid. When all the SKUs with that attribute are completed the attribute is allowed to change to the next value in the sequence. For the variant there is no direct attribute sequence on the mixers, but they will inevitably depend on the attribute sequence on the packing lines. In this way the only non-zero *MBids* calculated in Equation (1) are those for the variants associated with SKUs that are feasible in the current attribute sequence on at least a packing line. This is done simply setting to zero the *PBids* in Equation (2) that violate the sequence, so if there is no SKU in the actual sequence the resulting *MBid* will be zero.

B. Maintenance Times

On the packing lines periodic maintenance operations are usually scheduled on an ongoing basis. In the current implementation, we assume that the maintenance schedule is known at the beginning of the optimization. Maintenance is encoded in the following manner: 1. in the computation of the available resources, only those that are not under maintenance are considered; 2. if a line is selected with a maintenance upcoming, if the packing is not finished when the maintenance starts, the remainder of the packing continues after the maintenance has been completed. This will affect the factor *PBid* in Equation (2) because the time T' to complete the job will be increased by the duration of the maintenance.

If other maintenance cycle(s) are present, analogous checks are made and further interruption in the packing might happen.

C. Turning on/off the Mixers

During the production of a variant, it might happen that the mixer has a production rate that is not properly compensated on the other side by the packing lines, either because the overall packing rate is too low, or because the system is in a state where some or all the packing lines cannot pack the variant being made in the mixer. In this situation, since the tanks have a limited capacity, if production is not stopped, tank(s) can overflow. For this reason we have introduced a mechanism to turn off the mixers when this is about to happen. The mixer is turned back on when the level in the tank has dropped below a reasonable level, or there is a packing activity with an overall rate greater than the production rate.

Mixers are turned on and off several times. The estimation of the precise turn on/off times has proven to be an issue requiring some quite complex heuristic calculations as part of the optimization process.

In particular we have to compute the earliest possible time at which a mixer can be turned on while ensuring that all packing lines will be able to run and no tanks will overflow or become empty. Similarly we have to compute the earliest possible time at which a given job can begin on a packing line, again while ensuring that no tanks will overflow or become empty. In the general case, determining the *earliest time* is an optimization problem in its own right. Because of the time-criticality of this routine to the performance of the overall optimization, carefully constructed heuristics and a limited search are used for this calculation. These were designed and observed to yield good results (as will be seen), and not to create nonviable solutions. However, it is possible, under some circumstances, that the resulting solutions might be suboptimal. These algorithms must deal with mixers of differing speeds, potentially feeding a tank which feeds more than one packing line simultaneously (each of which may run and hence drain the tank at different speeds). Finding the earliest time such a job can start might require turning a mixer on earlier than before, but might also require turning off a mixer that was previously on. Such complicated permutations are very time-consuming to explore, given that these questions are asked in the innermost step of the optimization algorithm (i.e. are asked many thousands or millions of times). Therefore a number of heuristics were developed which considered separate cases such as that of a mixer being faster versus slower than the packing line(s) it feeds, and that of small versus large time ranges when the mixer is currently off, and that of packing jobs which extend across long versus short horizons. These heuristics were carefully evaluated and tested by statistically generating large numbers of artificial mixer/tank/packing-line scenarios. This gave us good confidence that the heuristics perform well. It is still theoretically possible that the heuristics don't return the optimal earliest time, however.

Split of the Making of the Variants

So far we have assumed that each variant is made at once and is completed in full, once its production starts, until the total necessary quantity (the sum of the demands of the SKUs having that variant) has been produced, even after an arbitrary number of on/off operations on the mixers. Nevertheless it could be more convenient to complete the production of a variant in two or more distinct phases.

We have implemented this enhancement by simply considering this fact. The attribute sequence might impose on the packing lines a sequence that will result in a suboptimal schedule if, for the SKU demand profile considered, there is no possibility of using the variant for contiguous/subsequent SKUs on the packing line. In other words, because of the attribute sequence, SKUs with another variant need to be packed before it is possible to complete the SKUs having the same variant. This situation will result in making and storage resources that are not fully exploited, since they are in a waiting state.

We then introduced the possibility of splitting a variant when this situation happens. The splitting is done at the beginning of the optimization by creating an additional variant in the optimization problem, whenever the presence is detected of one or more SKUs that will never be contiguous under the attribute sequence constraints on some packing line. A new variant is created relative to those SKUs, while the demand of the initial variant is adjusted.

With this enhancement, improvements with respect to the previous results are 2.91% on average, for the problems considered below in the next section (problems W19 to W22).

Finally one might want to consider also splitting the packing of the SKUs. We believe this procedure could also increase the quality of the best schedule found, but no implementation has been done at this time, since more complex criteria need to be introduced in this regard.

5 Results of the Optimization

We ran the model presented on problems of the following size: 30–40 SKUs, 10–12 variants, with usually 3 mixers, 5 tanks, 4–6 packing lines. These problems were given by Unilever for the scheduling of a real factory, based on data for fabric conditioner plants. Several tests were conducted. We present in this section only those for which we had an existing best solution as a benchmark. For each of these prob-

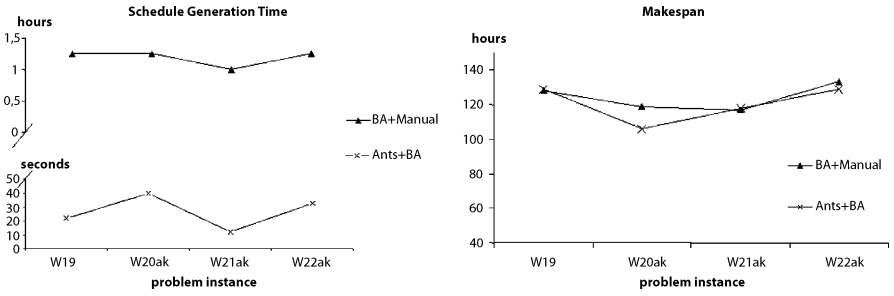

Fig. 3 Benchmark comparison of the new algorithm with the bidding algorithm combined with the best known. On the left, computation times are shown, while the right side shows the quality of the solutions found. The new algorithm finds solutions about two orders of magnitude faster (in minutes), and usually of better quality

lems, 5 optimization runs were repeated (that is starting from scratch, with uniform pheromones), each with 10,000 iterations, that are usually completed in about 20 seconds on a 2.66 GHz Pentium IV, using J2SE Runtime Environment 5.0. In most problems the best solution is found within the first 1000 iterations, and within 10,000 iterations the same absolute best solution is found in all 5 runs (except with problems W21ak and W22ak). The optimization was also run for 100,000 iterations in a search for further improvements.

We compared the results obtained in terms of computation time and quality with existing benchmarks of the bidding algorithm combined with human adjustments, that is manual methods based on visualization of the solution and adjustments by skilled trials moving/splitting jobs. This included the possibility of splitting SKUs as well, a possibility not implemented in the current version of the ants-bidding algorithm (in the figures, indicated Ants+BA). This method, however, takes times in the order of hours to generate the scheduling.

As can be seen, the Ants+BA algorithm can find optimized solutions much faster, with an improvement of about two orders of magnitude. Most importantly, the solutions are also improved in terms of optimization objective (shorter maximum makespan). The results of the runs conducted are compared with the known benchmark, and are shown in Table 1.

On average over all runs considered, an improvement of $< \Delta\% > = 3.29\%$ for the 10,000 iterations runs and $< \Delta\% > = 3.83\%$ for the 100,000 iterations runs is reached, which is significant over a week of operations on a single plant. Some negative signs are present for D, mainly due to the fact that the benchmark times given were rounded to the hour.

Besides this, it is also worth noticing that the new algorithm behaves in a quite different way with respect to the number of changeovers and setup times, as shown in Fig. 4. In particular variant changeovers on mixers and pack size changeovers are reduced, while variant changeovers on packing lines are increased. This result might be interpreted as due to the fact that usually pack size changeover durations are greater than variant changeover durations on packing lines.

Table 1 Results of optimization in terms of maximum makespan (in hours). The ant-bidding algorithm (Ants+BA) is compared with the best known solutions after 5 runs of 10,000 and 100,000 iterations, and the best solution found is adopted. The percentual improvement $\Delta\%$ is then calculated

Instance	Benchmark	Ants+BA (1.e5 iter.)	$\Delta\%$	Ants+BA (1.e6 iter.)	$\Delta\%$
W19	131	129.02	1.52	129.02	1.52
W20	101	99.36	1.64	99.36	1.64
W21	112	112.06	−0.05	112.06	−0.05
W22	103	96.79	6.22	96.79	6.22
W20ak	119	106.11	11.45	106.11	11.45
W21ak	117	117.15	−0.13	113.80	2.77
W22ak	133	129.88	2.37	128.76	3.24

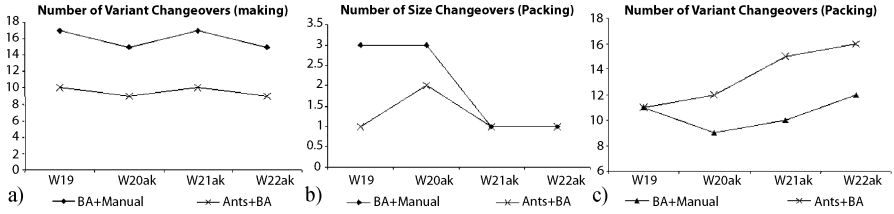

Fig. 4 Comparison of the number of changeovers (**a**) making, (**b**) and (**c**) variant and SKU size on PL

The overall average improvement in the number of changeovers is (for the problems considered here) equal to 20.54%. We notice that there are several solutions having the same makespan, because minimizing the makespan leaves some degeneracy on the other packing lines, as they might finish at any time earlier than the maximum makespan and provide a viable solution of equal quality. This aspect could be useful when alternative ways of scheduling are needed.

A. Sensitivity to the Parameters

As in other optimization problems, ant algorithms demonstrate robustness with respect to changes in the instance of the problem with little or no tuning of the parameters required. The values for the parameters are set on some test data, to maximize the quality of the solutions found and the speed of their search. Values of the parameters that gave the best results are: $q_o = 0.4-0.5$ and $r_o = 0.99/0.98$ (one/two random step each 100 steps), $\alpha = 0.1$, $\beta = 1.0$, $\gamma = 2.0$, $r = 0.7$, $\epsilon = 1.0$ and $\epsilon_{boost} = 10.0$, with an offset of 0.01% from the best solution. In particular we observe that with respect to other optimization problems solved with ACO, the value of q_o used here seems quite low. This is due to the fact that the problem considered here is highly constrained, so it is better to favor exploration instead of exploitation.

The number of ants per iteration is always set to $nAnts = 1$, and does not seem to be a crucial parameter for these small–medium size problems.

B. Glitches on Packing Lines

In this section we analyze the robustness of the production schedule with respect to the effects of unforeseen glitches on the packing lines. This is particularly important, since there are many unexpected events in the real world that may require production rescheduling. The capacity to cope with glitches depends mainly on the *flexibility* of the plant. Notice that there are two situations that might affect the impact of a glitch:

a. the glitch is on a PL that has little utilization.
b. the glitch is on the PL that has high utilization.

To visualize the plant schedule and the effects of the glitch, we have developed a graphical user interface. In Fig. 5, two schedules are shown as a result of the optimization process. The horizontal axis is time, the top panel shows the mixer schedule, the middle panel the liquid quantities in the tanks and the bottom panel shows the packing line schedule. The maximum makespan is also indicated at the very top with the red line.

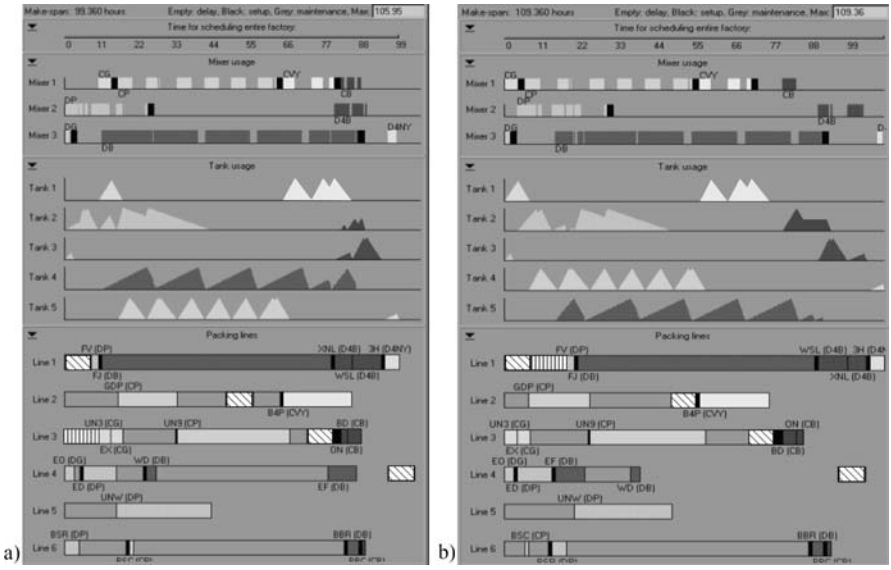

Fig. 5 (**a**) The glitch on packing line 3, is from 0 to 10 (in red) and happens to be on a PL which has some slack. In this case, there is no change in the makespan, since the duration of the glitch is less than the available slack. (**b**) Here the glitch (from 8 to 18) is on the *critical* packing line 1, the one with the maximum make-span and no flexibility to move other packing tasks to other lines. Hence the overall makespan increases

Variants (on mixers and tanks) and SKUs (on packing lines) are labeled with their names and colored with the respective color of the variant/SKU. The black rectangles represent setup times, while dark gray rectangles on the packing lines are the scheduled maintenance operations. In case **a.** the glitch (in red) occurs on a line with low utilization, so has no remarkable effects on overall production completion, while in case **b.** the glitch is on a line that is highly utilized and is the only one that can accomplish the long production of FJ(DB) (in blue) and the following packing tasks, so all production must shift, increasing the final makespan by the duration of the glitch.

The impact of the glitches are then essentially related to the flexibility of the plant. With dynamic scheduling the loss of time is never greater than the duration of the glitch.

C. Other Considerations

Because of the stochastic nature of this algorithm, to gather statistical consistency data the algorithm was run several times over repeated runs, each starting from scratch, with uniform pheromones distribution.

Over the runs, the average best fitness and the average system fitness as well as their standard deviations were calculated, to analyze the convergence process. By *system fitness* here we indicate the fitness of each ant (sampled every 10 iterations) averaged over the number of ants that are being considered in that interval, which is

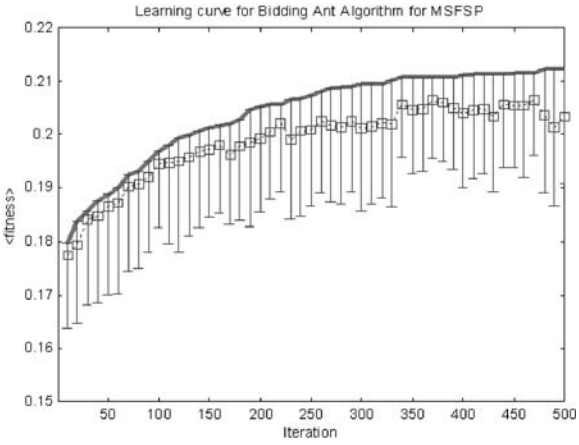

Fig. 6 Learning curve: the red solid line represents the best fitness found during iteration of the algorithm, the blue squares and bars represent the system fitness and its variation

related to the good solution discovery process. In Fig. 6, the red lines represent the average best fitness found so far (averaged over all runs), the blue squares the average system fitness (over the runs). The vertical bars represent the standard deviation of the average system fitness, to indicate how much variation has been present from one experiment to another. If, for example, the greediness is high (q_o) the squares will lie very close to the red line, with smaller error bars, indicating that the ants are performing less exploration.

It is also useful to evaluate the distribution of the average fitness and its cumulative distribution C, which are related to the probability P of finding a solution for

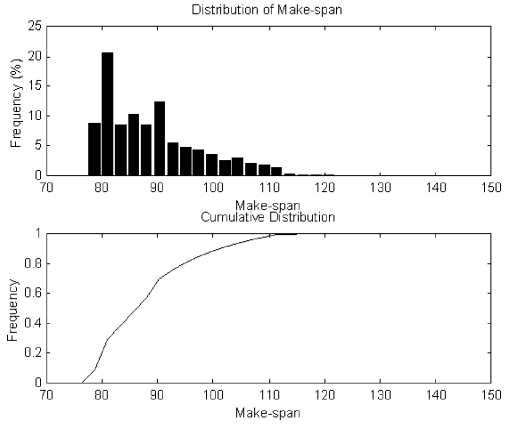

Fig. 7 Makespan distribution and its integral, of every agent that generates a solution during iteration of the algorithm

a given value of the fitness ($P = 1 - C$) and to phase transitions in the system, as shown in Fig. 7.

Another relevant issue is how ants, once they have explored the space of solutions, are able to react to changes in the system. One type of change is to add, remove or split a job, perhaps as a consequence of demand changes. Others relate to plant operation, such as glitches or machine breakdowns. We can show that the pheromone memory allows the discovery of new solutions in a faster way. In Fig. 8, the red (lower) curve represents the fitness when the optimization is started as usual with a uniform distribution of pheromones. After 300 iterations the optimization is stopped and the problem is slightly changed by removing some tasks to be scheduled. The pheromones in this case were initialized with the distribution obtained at the end of the previous run. The blue (upper) curve shows not only a higher fitness (which actually depends on the fact that some jobs were removed, hence shortening the makespan), but most importantly a steeper rise.

6 Design

The optimization can also be run to test different plant configurations. Aided by data generated by an optimization system such as this, on the relative constrainedness of different parts of the manufacturing plant, a designer could understand and experiment with the implications of different types of design changes, and thereby understand which design features contribute most heavily to the creation of a factory which is *both efficient and robust* from the point of view of the operations/planning management.

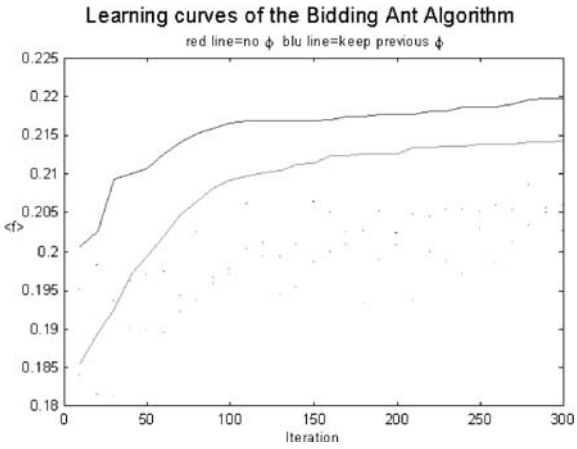

Fig. 8 Learning curves: the Y axis represent the fitness function (arbitrary unit), the red (lower) curve is a standard run of the algorithm. When the pheromones of the last iteration are used to initialize those for a modified problem, learning is much faster, as shown in the blue (upper) curve

It is possible to create several sets of experiments by changing each time the value of the demand for each SKU and by initializing the pheromones to the average calculated over the first set of experiments (i.e. those with unchanged demand). We conducted an analysis of the factory capability to absorb variations in the demand by introducing a uniform variation of 20% from period to period, in the demand of the SKUs and spread between multiple SKUs.

Figure 9 shows nine possible configurations for a plant, derived by variations of an original problem with 3 mixers ($M = 3$), 3 tanks ($T = 3$) and 5 packing lines ($PL = 5$). For the first row of plots, each plot is relative to an increase in the number of mixers, starting with $M = 2$, up to $M = 4$. The second row, each plot is relative to an increase in the number of tanks, starting from $T = 2$, up to $T = 4$. The third and last row, each plot is relative to the increase in the number of packing lines, starting from $PL = 4$ up to $PL = 6$.

In all the plots the z-axis is the makespan necessary to complete the schedule, while the x and y axes represent the incremental variation in the number of the other resources of the plant, e.g. in the first row the number of mixers is progressively increased, and each plot shows the resulting makespan as a variation of the number of PL versus the number of tanks, T. The green bars are proportional to the standard deviation, or variation for the time to complete the schedule.

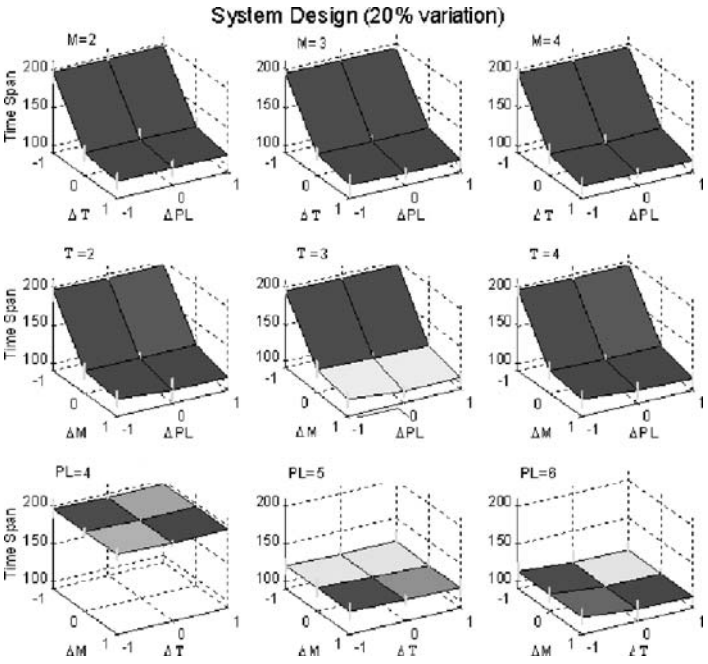

Fig. 9 Plots of the average time to complete a schedule with different plant designs: in each plot, we vary the number of resources available. Each row represents a different resource varying. First row mixers, second tanks, and third packing lines

We make two important observations here: on row 3, plot 1, we have that with $PL = 4$, even increasing the number of mixers to $M = 4\ (\Delta M = 1)$ and the number of tanks $T = 4(\Delta T = 1)$, no significantly better schedules are obtained, than just having $M = 2(\Delta M = -1)$, and $T = 2(\Delta T = -1)$. The situation of having just 4 packing lines is evidently a bottleneck for the system.

In other plots, as in row 2, plot 2, we notice that increasing PL and M is really beneficial if we move from $PL = 4(\Delta PL = -1)$ and $M = 2(\Delta M = -1)$, to $PL = 4(\Delta PL = -1)$ and $M = 3(\Delta M = 0)$, but not convenient at all to add an extra line, that is to $PL = 5(\Delta PL = 0)$ and $M = 4(\Delta M = 0)$ or from the initial point $PL = 4(\Delta PL = -1)$ and $M = 2(\Delta M = -1)$ to $PL = 5(\Delta PL = 0)$ and $M = 2(\Delta M = -1)$. We have added a packing line without obtaining any significant improvement.

The examples cited here support the following observations:

1. Tradeoffs exist between cost of design and robustness of design, in terms of the ability to robustly handle demand variations. Hence, robustness criteria need to be considered in deciding plant capacity.
2. Even if it is decided to increase capacity to improve design robustness to deal with demand uncertainty, the capacity increase needs to be decided on the basis of an analysis of plant bottlenecks.

A. Dynamic Capacity

We now examine phase transitions related behavior in this problem with a specific focus on capacity/flexibility related issues. We define in this section, a measure of the dynamic capacity of a plant based on the idea of considering the possible different ways (processes) to complete a finished product/job/task/object. We consider all the possible processes and distinguish among parallel and serial processes, respectively those that can happen at the same time, and those that follow one and other, like a sequence of operations, or sub-path.

The idea is to calculate for each finished product (here the SKUs) a measure of efficiency/flexibility or *utility* of the parallel and serial processes involved by first identifying each distinct way that the final product can be achieved with the corresponding sequence of operations (serial processes), and evaluating the speed at which they can complete the process. Then, those that represent alternative ways to accomplish the final product are added in a proper way. The calculation is repeated for all products. Finally all the utilities for all the products are combined taking into account also additional correction factors due to constraints on the maximum parallel processes present at any time.

In this way, for each possible final product, we consider all the feasible paths from start to completion. Each of these paths can be regarded as one set of operations or indefinitely many, and different finished products may share the same set, as in a typical multistage flowshop process, or have a quite different deconstruction into sets or simpler processes, as in a complex job-shop type process, or complex heterogeneous supply chain.

For each path we calculate, given the sequence of operations required, the minimum value of the utility U along the path, over either discrete steps or a continuous

path, that is the minimum rate at which the task can be accomplished on this path. We repeat this calculation for all the pathways for this task, properly adding them together (some weighting might be needed). Adding up all the utilities for different paths for this task might not be sufficient, given that parallel processes might not be possible in reality because of constraints, such as limited simultaneous connections or limitations on resources. We need to examine for each of those paths how many of them can be effectively simultaneously possible and calculate the fraction of those that may be carried out at the same time. We can call this term the simultaneity constraint S, which corrects the utility U for the task t considered. The corrected utility is thus given by:

$$\tilde{U}(t) = S(t) \cdot \sum_{p \in \text{paths}} \min_{n \in \text{nodes}} U(p, n, t) \tag{9}$$

where the nodes represent the serial operations to complete a product on the path p, each characterized by a processing rate, and $\tilde{U}$ represents the corrected utility when taking into account constraints on simultaneous processing of tasks. The dynamic capacity is then given by:

$$C = \sum_{t \in \text{tasks}} \tilde{U}(t) \tag{10}$$

We have carried out the above procedure for the multistage flowshop problem (MS-FSP), which involes finding the dynamic capacity when the system is a plant with M mixers, producing V variants, each with different production profile and rate (which depends on the variant), T tanks, for temporary storage of different variants (one at a time), PL packing lines, characterized by packing profile (with SKU) and rate for each SKUs (a variant in a certain pack size). We assume that the first two stages are fully connected, while from tanks to packing lines there is a maximum number of simultaneous connections Nc at a time. A scheme of these settings is represented in Figure 10.

For each SKU s and each feasible path p (a combination of mixer and packing line) the utility can be defined as: $U(p, s) = \min r_M, r_{PL})$, where r_M and r_{PL} are respectively the mixing rate (for the variant) and the packing rate (for the SKU). The simultaneity constraint factor is taken into account in the following:

$$S(t) = \frac{\min(PL(s), N_c)}{\max(PL(s), N_c)} \tag{11}$$

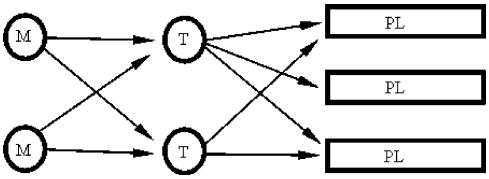

Fig. 10 Schematic representation of the liquid plant considered for the phase transition analysis

where *PL(s)* is the number of packing lines where the SKU s can be done and N_c the maximum number of connections at a time between the tank and the packing lines. Due to full connectivity from mixers to tanks, more correction factors are not necessary; we do not need this assumption in this context, but we take into account this aspect. Indeed the time to complete a task is governed by the minimum rate along the serial process, as explained in the text.

The total dynamic capacity is then given by:

$$C = \sum_{v \in \text{Variants}} S(s) \cdot \sum_{p \in \text{paths}} U(p, s(v)) \tag{12}$$

It should be noted that the dynamic capacity has been defined in this way to capture the flexibility of the plant, which is namely related to the various possible paths available to manufacture a product.

We generated a number of plant instances with varying degrees of connectivity and a number of demand profiles which are constantly increased by a finite amount. The results are shown in Fig. 11, where the z-axis is the time to complete the schedule (the makespan), the green bars represent the standard deviation over the demand variation, having added for each demand profile a 20% uniform random variation, and repeating this calculation over five runs. The red dots are missing data for plant profiles that have been interpolated.

We note that a phase transition is present in this case for $C = 35{,}000$, where a sudden drop occures in the time required to complete the schedule to a value that is fairly constant even with further increases of the capacity. The interpretation of this

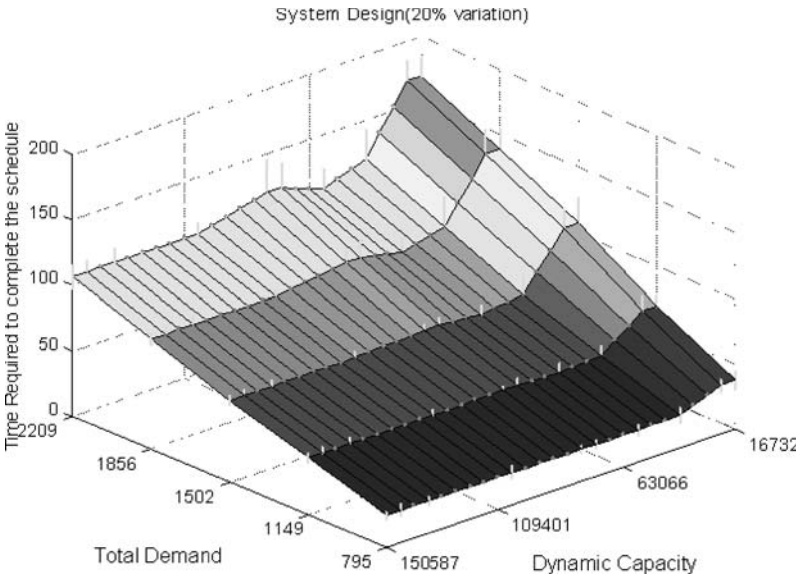

Fig. 11 Study of the factors affecting the time required to complete a production cycle, based on the total demand and dynamic capacity defined in Equation (12)

result is that, irrespective of the demand, the plant dynamic capacity needs to exceed a certain threshold to keep the plant makespans acceptable and robust to variable demands. This situation is represented also in Fig. 12, where the dynamic capacities of two existing plant configurations are computed and indicated by the two vertical lines. A great improvement could be accomplished if only these plants could increase their dynamic capacity and move after the phase transition, gaining about a 30% improvement. This improvement could be accomplished by simply increasing the number of connections between stages, with very little additional cost.

Interestingly enough, Unilever has anecdotal stories that relate to this analysis. Over the years a given factory (say a toothpaste factory) has changed from making 12 kinds of toothpaste, to 15 kinds, and then to 22, all without any substantial change in the efficiency of production. However, when the 23rd kind was introduced, suddenly the factory just couldn't cope with the production any more. Efficiency dropped drastically.

Further analysis and experiments need to be done varying the number of SKUs under different demand scenarios with a fixed plant capacity to examine any phase transition related effects. This is the subject of future work, although the order parameter we have introduced, and its phase transitions appear to be a likely explanation for these observations.

7 Conclusions

We have examined a number of aspects related to the scheduling of a multistage manufacturing plant. The use of an adaptive algorithm provides a better and more effective search in the solution space and copes with rescheduling in the case of malfunctioning or glitches, and variations in the demand. The improved speed in the

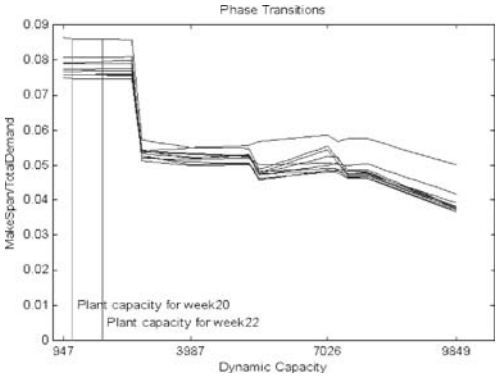

Fig. 12 Phase transition in the time necessary to complete a production cycle. An increase of the dynamic capacity can correspond a relevant decrease of the production times. The two vertical lines represent two plants from the data. Here it is shown that if bottlenecks are eliminated by improving connectivity among stages (e.g. little additional investment), the throughput of the plant can significantly increase

search for solutions allows one not only to find optimal solutions, but also to study several system configurations under many different conditions. The discovery of an order parameter and a deep theoretical analysis of the phase transitions improves the understanding of system design and its robustness and capability to respond to a changing environment.

Potential extensions of this work include applications to other flowshop and job-shop problems, but also to other complex/adaptive scheduling problems.

References

1. C. Y. Lee, G. Vairaktarakis (1994) Minimizing makespan in hybrid flowshops. Operations Research Letters 16:149–158
2. A. Babayan, D. He (2004) Solving the n-job 3-stage flexible flowshop scheduling problem using an agent-based approach. International Journal of Production Research 42(4):777–799
3. M. H. Bassett, P. Dave, F.J. Doyle, G.K. Kudva, J.F. Pekny, G.V. Reklaitis, S. Subrahmanyam, D. L. Miller, M.G. Zentner (1996) Perspectives on model based integration of process operations. Computers and Chemical Engineering 20(6-7):821
4. C. A. Floudas, X. Lin (2004) Continuous time versus discrete time approaches for scheduling of chemical processes – a review. Computers and Chemical Engineering 28(11):2109–2129
5. C. Charalambous, T. Tahmassebi, K. Hindi (2000) Modeling multi-stage manufacturing systems for efficient scheduling. European Journal of Operational Research 122(2):329
6. E. Nowicki, C. Smutnicki (1996) A fast tabu search algorithm for the permutation flow-shop problem. European Journal of Operational Research 91:160–175
7. T. Aldowaisan, A. Allahverdi (2003) New heuristics for no-wait flowshops to minimize makespan. Computers and Operations Research 30(8):1219–1231
8. M. Nawaz, E.E. Enscore Jr, I. Ham (1983) A heuristic algorithm for the m-machine, n-job flowshop sequencing problem. The International Journal of Management Sciences 11:91–95
9. N. G. Hall, C. Sriskandarajah (1996) A survey of machine scheduling problems with blocking and no-wait in process. Operations Research 44(3):510–525
10. V. Darley (1999) Towards a Theory of Autonomous, Optimizing Agents. PhD Thesis, Harvard
11. J. C. Beck, W. K. Jackson 1997, Constrainedness and the phase transition in job shop scheduling. Technical report TR 97-21, School of Computing Science, Simon Fraser University
12. H. Bauke, S. Mertens, A. Engel (2003) Phase transition in multiprocessor scheduling. Physical Review Letters 90(15):158701–158704
13. B. Ramachandran (1998) Automatic scheduling in plant design and operations. Internal Report, Unilever Research, Port Sunlight Laboratory
14. M. Dorigo, G. Di Caro, L. M. Gambardella (1999) Ant algorithms for discrete optimization. Artificial Life 5(2):137–172
15. M. Dorigo, V. Maniezzo, A. Colorni (1996) The ant system: optimization by a colony of cooperating agent. IEEE Transactions on Systems, Man and Cybernetics part B 26(1):29–41

16. L. M. Gambardella, M. Dorigo (1995) AntQ: A reinforcement learning approach to the traveling salesman problem. Proceedings of the Twelfth International Conference on Machine Learning ML95 252–260
17. L. M. Gambardella, E. D. Taillard, M. Dorigo (1999) Ant colonies for the quadratic assignment problem. Journal of the Operational Research Society 50:167–176
18. D. Costa, A. Hertz (1997) Ants can colour graphs. Journal of the Operational Research Society 48:295–305
19. L. M. Gambardella, M. Dorigo (2000) An ant colony system hybridized with a new local search for the sequential ordering problem. INFORMS Journal on Computing, 12(3):237–255
20. C. Rajendran, H. Ziegler (2004) Ant-colony algorithms for permutation flowshop scheduling to minimize makespan/totalflowtime of jobs. European Journal of Operation Research 155:426–438
21. A. Colorni, M. Dorigo, V. Maniezzo, M. Trubian (1994) Ant system for job-shop scheduling. Belgian Journal of Operations Research, Statistics and Computer Science 34:39–53
22. T. Stuetzle (1998) An ant approach for the flow shop problem. Proceedings of the 6th European Congress on Intelligent Techniques and Soft Computing (EUFIT 98), Vol. 3:1560–1564
23. E. Taillard (1993) Benchmarks for basic scheduling problems. European Journal of Operational Research 64:278–285
24. L. M. Gambardella, E. Taillard, G. Agazzi (1999) MACS-VRPTW: vehicle routing problem with time windows. In: M. Dorigo and F. Glover (eds) New Ideas in Optimization. Corne, McGraw-Hill, London
25. C. Blum, A. Roli (2003) Metaheuristics in combinatorial optimization: Overview and conceptual comparison. ACM Computing Surveys 35:268–308