

An Ant-Bidding Algorithm for Generalized Flow Shop Scheduling Problem: Optimization and Phase Transitions

Alberto V. Donati^{*†}, *Vince Darley*[‡], *Bala Ramachandran*[§]

albertodonati@inwind.it, vince.darley@eurobios.com, rbala@us.ibm.com

November 2004

Abstract

In this paper we present the integration of an ant-based algorithm with a greedy algorithm for solving the scheduling of a multi-stage plant. A multi-stage manufacturing plant is comprised of different stages: mixing, storage, packing and finished goods storage, and is an extension of the classic Flow Shop Scheduling Problem (FSP). The algorithmic details for the Multi Stage Flow Shop Scheduling Problem (MSFSP) are introduced 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 optimization, but also for analyzing the design of the plant in order for it to optimally cope 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 constrainedness of the problem, and the nature of phase transitions in the difficulty of finding viable solutions.

Keywords: scheduling, optimization, bids, ant colony system.

This work was co-funded in Consortium Agreement between Bios Group and Unilever during 1998-1999, and was referenced in Scientific American, *Swarm Smarts*, by E. Bonabeau and G. Theraulaz, Mar. 2000.

1. Introduction and problem description

* Corresponding author.

† IDSIA.

‡ Eurobios.

§ IBM T.J. Watson Research Center, Yorktown Heights.

We focus in this paper on the scheduling of operations of a multi-stage manufacturing plant, which is the core of a multi-stage supply chain. The production plant has three main parts: making (mixers), intermediate storage (tanks), and packing lines; each stage has inputs and outputs, with limited connectivity, represented by connecting lines.

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, the type and quantity of each SKU, for the period to be scheduled (usually one week), the optimization consists in finding the schedule of the resources on each stage, to minimize the latest time of completion time on the packing lines (also called the *makespan*). In this way, the approach proposed will attempt to globally optimize the schedule of the whole factory.

Unilever possesses about 10,000 of such plants worldwide.

In details the plant is characterized in the following way:

Making: raw materials are subject to a number of parallel chemical processes to obtain different variants, each characterized by a number of attributes, like color, base formulation, and dilution level. The variants produced are temporarily stored in tank facilities, and then directed to packing lines. Making is characterized by the number of mixers in the plant. Each mixer is then characterized by the variants it can make, the rate of operation (for every variant), the batch size, the cleaning/changeover times between different variants (which depends on the exact variant sequence). Finally mixers are characterized by the connectivity with the tanks, that is the maximum number of simultaneous connections at a time with tanks, and a 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 have with the mixers, the maximum number of simultaneous connections to the packing lines, and the setup/change over times in changing the variants 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, and each PL is characterized by connectivity with the tanks, by which SKU can be packed (and at which 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 could be preferred (or required) to follow under a specified policy.

Finished Goods Storage: forecasted demand and actual demand can sometimes present differences which can not be covered during the production span. This can be due in part to forecast errors, but mostly due to short notice changes to customer orders, 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 handling properly 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.

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. In this paper we will not discuss the wider implications of examining the position of a given manufacturing plant in the supply chain.

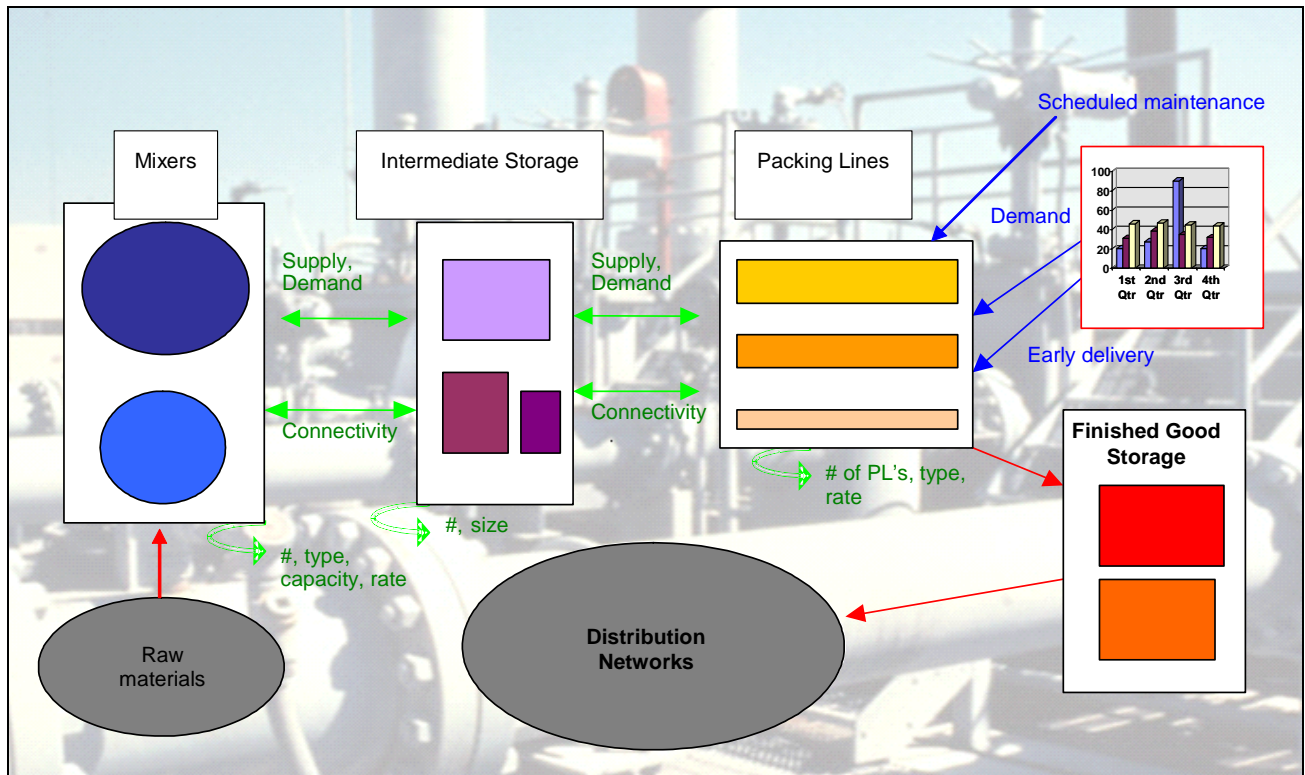


Figure 1: Supply chain and plant representation. From left to right: mixers, tanks, and packing lines.

In the most general case the connectivity between the three stages is limited, and this adds more complications to the scheduling. It is easy to recognize that the Multistage Factory Scheduling is then an increasingly complex task, which is a NP-complete problem not naturally solvable with the aid of linear programming techniques.

2. Ants and Multistage Scheduling Flow Shop 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 has been proposed for discrete optimization, as discussed in [2], and have been successfully applied to the traveling salesman problem, symmetric and asymmetric ([3], [4]), the quadratic assignment problem ([5]), graph-coloring problem ([6]), sequential ordering ([7]), job-shop ([8]), flow-shop ([9]), and vehicle routing ([10]).

Ant-based optimization is a parallel distributed optimization process. A population of ants mutually discovers and combines 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, constructing a solution step by step, and examining at each step the pheromones in the neighborhoods. 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 retrospectively adjusts the strength of pheromone on the path it took proportionally to the fitness.

A procedure such as this, iterated many times by a population of ants, will create two things: (a) good solutions will be found, 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 the ants can immediately discover a good solution to the new problem. 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 the Ant algorithm it is quite straightforward, and these rules of thumb were encoding into the bidding algorithm described below.

The algorithm presented has a number of enhancements. On one hand, the search for solutions has a *reinforcing* component, done by boosting high fitness solutions found, and on the other, by an enhanced exploring component by promoting more random exploration.

The bidding algorithm is used for the calculation of the greedy part of the probabilities (this will be seen in the algorithm details), used by the ants to decide which step to take. In other words, a step is the completion of an assignment task (that is making a variant in a mixer, called M-step, or packing a SKU on a packing line, called P-step).

Using these enhancements, the process of learning is constant and faster, and with the proper choice of the parameter set, the optimal solutions are found within 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 the details of the ant-bidding algorithm.

Initialization

The pheromones are of two different types and are therefore represented with 2 matrices, one for the variants (M-Step) and the other for the SKUs (P-steps). The matrices are set to a uniform values at the beginning. An element i, j of any of the matrix represent at any time the pheromone on the virtual “edge” connecting job i with job j , that is when job j immediately follows job i on *any* resource. This scheme is called job-pair implementation. Note that it is useful to introduce at the beginning of the schedule on each resource (M mixers and PL packing lines) virtual jobs, to be able to store information in the pheromone matrices about the first (real) jobs of the schedule; otherwise the first jobs of the schedule would result indifferent for the pheromones.

The dimension of the M-Pheromone matrix is then $(V+M)*V$, where V is the number of variants to make, and the one of the P-Pheromone matrix is $(SKU+PL)*SKU$, where SKU is the number of SKUs to pack.

The level of pheromones during the iterations is never allowed to drop under a minimum threshold and never be higher than a maximum level. This procedure allows the ants on one side to never exclude zero-pheromone “edges”, 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 might be a reasonable assumption to remove some edges from the graph during the iterations of the algorithm (for the TSP this is like removing the edge between distant cities).

Propagation of ants

An iteration of the algorithm consists in the initialization and propagation of a number of artificial ants. Each ant of the group completes a solution, by choosing a sequence of steps until no SKU is left to be done;

here, since the problem size is $N=V+SKU$, this will involve N such steps for each ant. Each ant maintains an updated taboo list containing the information about steps carried out so far, that is, a list of assignments already done. During propagation the pheromones are not changed, and only when the last ant of the colony has finished, the pheromones are updated.

The propagation of each ant consists in repeating the following steps.

• **Choose a resource**

The ant has to choose at first which *resource* is available for the scheduling after the last step. With the term of *resource* we indicate one of the following two:

1. a mixer and a tank,
2. a packing line and a tank.

This will determine the type of step to take, if a mixer assignment, M-Step, else a packing line assignment, P-Step.

The resource is chosen by considering the one that will be first available, given the state of the system for that ant, with all the constraints; for example, given that not all jobs are feasible everywhere, if a mixer is available but it cannot handle any of the variants left to be done, the mixer is not consider an available resource.

So if 1. is available, the ant will perform a M-Step, otherwise a P-Step.

• **Choose a task**

M-Step

Once a mixer and a tank have been identified, the bids (M-Bids) for each variant that still need to be done, and that can be done on this resource, have to be calculated.

The bids of the variants are given by the following:

$$(1) \quad MBid(v) = \sum_{s=1, \dots, SKU} PBid(s | v)$$

where v is the variant, s is the index for the SKU, $s | v$ is the SKU having v , $PBid()$ is the bid of the SKU s and all the possible packing lines where the SKU can be done, finding the highest value of the bid, and is calculated as:

$$(2) \quad PBid(s | v) = \frac{D(s | v)}{R'}$$

where $D(s | v)$ is the demand of s , and the term R' includes the delay to wait for that PL to be available, the setup times, and the time of completion of the SKU: $D(s | v) / r_{PL}(s)$ where $r_{PL}(s)$ is the packing rate of SKU s on the packing line (very similar to eq. (5)).

In the term $PBid()$, we do not take into account any setup/changeover times for the mixer and the tank, or delays due to mixer and tank availability and the setup times for the packing lines.

The MBids are finally scaled by a proper factor to be in the order of magnitude of the pheromones.

The probabilities of making variant j after making the last variant i , are then:

$$(3) \quad P[j] = \frac{1}{N} \cdot \frac{\varphi_V [i][j]^\alpha \cdot MBid[j]^\beta}{(1 + Setup[j])^\gamma}$$

where:

N is a normalization factor for the probability,

φ_V are the pheromones for the variants,

$Setup[j]$ is the setup/changeover time for the mixer, for doing j after i ,

α, β, γ are the relative weights of these factors.

Once the probability distribution has been calculated, it can be used in different ways.

A probabilistic step consists in picking the j with a probability given by (3), while a greedy step consists in picking the j that has the maximum value of probability. Finally a random step is just picking randomly among the tasks feasible on that resource.

Which type of step to use is determined by 2 cutoff parameters g_o and p_o with $g_o < p_o$. At each step a random number r is generated, so the ant take a greedy step if $r < g_o$ (with $0 \leq g_o \leq 1$); else if $r < p_o$ the ant takes a probabilistic step, otherwise a random step. Once the step has been chosen a number of variables are updated, like the mixer available time, tank available time, variant that the tank is storing and so on.

P-Step

If the next available resource is **2**, the task to do is to assign a SKU to a packing line. The packing line has been chosen at this point, but the tank not yet, because in each given moment, there are several tanks containing liquid. Before choosing the tank it is then necessary to calculate the probabilities for the SKUs, that will determine which variant (and then which tank) to use. Of course for the SKU to be considered in the bids, its variant must be present in some tank. Then the value of the bid is given by:

$$(4) \quad PBid(s) = \frac{D(s)}{R}$$

where $D(s)$ is the demand of the SKU s for the period, and R 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 shift the termination of the SKU that is not complete by the time the maintenance starts), the processing rates of the packing line for the

SKU, and finally the delays due to the following: other packing jobs with the same variant can be already started on one or more different packing line(s) and the mixer rate (feeding that tank) for the variant, compared to the quantity in the tank, may not keep up with an additional packing line using the variant. In the worst situation, we will need to introduce a delay for the start of the packing. The calculation of the exact delay can sometimes become relatively expensive computationally (when there is multiple inflow and outflow to a single tank, for example), so we only consider in the calculation of the bids and approximate delay Δ , the one that would guarantee the feasibility.

We have then that the resource allotment time is:

$$(5) \quad R = \max(T_{\text{Available}} - t, 0) + T_{\text{setup}} + T_{\text{complete}} + \Delta$$

where t is the current time, $T_{\text{Available}}$ is the time when the packing line will finished the previous job and will be available, T_{setup} is the setup time (for changing pack size and/or variant), $T_{\text{complete}} = D(s)/r_{PL}(s)$ is the completion time of the SKU on that PL, $r_{PL}(s)$ is the packing rate for s , and Δ is the approximate estimation of the delay, and also includes possible additional penalties due to the presence of maintenance during the packing task (in this case the packing needs to be interrupted).

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

$$(6) \quad P[j] = \frac{1}{N} \cdot \frac{\varphi_{SKU}[i][j]^\alpha \cdot PBid[j]^\beta}{(1 + T_{\text{setup}}[j])^\gamma}$$

where:

N is a normalization factor for the probability,

φ_{SKU} are the pheromones for the SKUs

$T_{\text{setup}}[j]$ is the setup time for the packing line for doing j after i .

Similarly the type or step to take (greedy, probabilistic, or random) is set by the two constants g_O and p_O .

• Update the taboo list

The taboo list stores which tasks have already been completed. The first V steps are always M-Steps, because at the beginning the mixers are all available, and the tanks all empty. When all the SKUs in the demand for the period have been completed (assigned to the packing lines), the schedule is complete.

At this point the ant calculates its fitness in the following way:

Calculation of the fitness

The fitness is the inverse of the maximum make-span over the packing lines, that is the time when the last packing line ends the last SKU:

$$(7) \quad f = \frac{q}{\max(MS)}$$

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

d. Update the pheromones

The pheromones matrices are updated by an evaporation and deposition process: all pheromones evaporate at a rate $1-\rho$ and those pheromones on ant's paths (the solution S) are augmented according to the fitness of those ants, according to the following equation:

$$(8) \quad \varphi[i][j] \leftarrow \rho \cdot \varphi[i][j] + \varepsilon \cdot \sum_{a=1 \dots nAnts} f(a) \Big|_{(i,j) \in S_a}$$

where: $f(a)$ is the fitness of ant a , and is incremented only on the edges $(i, j) \in S_a$ so that all pheromones evaporate, but just those pheromones which contribute to individual ant's paths are incremented in proportion to the fitness of those paths. Here ε is constant, and ρ is the pheromone persistency constant (equal to 1 less the evaporation constant). If f is close enough (usually 5%) or greater than the best fitness so far, we may choose to boost the pheromones by using ε_{boost} instead of ε .

Each ant's fitness $f(a)$ is compared to the best fitness found so far, and if it is greater, this becomes the new best fitness, and the solution is stored.

The iteration then continues from **b.** with a new colony, until all the N_{it} iterations are completed.

The scheme of the algorithm is presented in the following Figure 2.

- Read the problem input data
 - Initialize pheromones ϕ_{SKU} and ϕ_V
 - for N_{it} 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:
 - What step-type? Choose what resource will to be available first.
 - Do a M-Step if it is a mixer, do a P-Step if it is a packing line.
 - Calculate the bids, and then the probability distribution.
 - Chose the step type, throwing a random number r :
 1. greedy step, if $r < g_o$
 2. probabilistic step, if $r < p_o$
 3. random step, otherwise.
 - Chose the task.
 - Take the step and update the taboo list and the necessary quantities.
 - Calculate the fitness $f(a)$
 - If fitness is greater than the best fitness, best fitness = fitness, and store the solution.
 - evaporate the pheromones everywhere
 - increment the pheromones of the solution found.
- return the best solution found.

Figure 2: outline of the Ants-Bidding algorithm.

4. Some other issues

Attribute Sequences.

We can required as an additional constraint, that the SKU must follow a determined color and/or pack size constraint, called attribute sequence, which depend on the packing line considered. Then, 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 of the sequence having at least one SKU associated. For the variant there is no direct attribute sequence on the mixers, but of course they also depend on the attribute sequence on the packing lines. In this way the only non-zero M-Bids are for the variants which have at least one SKU feasible in the current attribute sequence on a PL, and because they are the result of the addition of the P-Bids, again we consider in the sum just those for the SKUs feasible under these constraints. Finally we observe that for some problems it is not possible to complete a schedule under these constraints.

The maintenance time constraints.

On the packing lines the maintenance is sometimes necessary and it is at a known time, and for a known duration. A packing line can have as many maintenance interventions as desired in this model. In the current implementation we proceed in the following way: we consider in the computation of the available resources the ones that can start first, being able to complete the job, including the presence of a maintenance time. If the packing ends after the maintenance starts, the job gets shifted to when the maintenance finishes, This check is repeated for all the maintenance cycles following.

Turning on or off the mixers.

During the production of a variant on a mixed, it might happen that the mixer has a production rate that is not compensated on the other side by the packing lines, either because their rate is too low, or the system is in a state where some (or all) the packing lines can not pack that variant. Since the tanks have a limited capacity, if production is not adjusted, they might overflow. For this reason we have introduced a mechanism to turn off the mixer when this is about to happen. The mixer can also be turned back on when the level on the tank has dropped, or there is a packing activity with a (total) rate greater than the production rate. There could be the possibility that mixers are turned on and off several times. The estimation of the precise times is also a very complicated issue requiring some quite complex sub-routines as part of the optimization process.

Computational times

We note that for problems with about 30 SKUs, 10 variants, with 3 mixers, 5 tanks, 4-6 packing lines, optimal solutions are found within 100 iterations. The code was developed in Java, and the program ran on a Pentium II at 233 MHz, completing its run in 30-60 seconds. Naturally one might want to search as much as possible for optimal solutions, by leaving the algorithm running for at least 1,000 iterations or more, that in these computational conditions which are completed in average in about 10 minutes.

Sensitivity to the parameters

As in other optimization problems, the algorithm demonstrates 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 speed in their search. Typical values of the parameters are like the following: $g_0=0.7$ and $p_0=0.99$ (one random step each 100 steps), $\alpha=0.1$ $\beta=1$ $\gamma=1$ $\rho=0.7$ $\epsilon=1$ $\epsilon_{boost}=10$. The number of ants per iteration is always 1, and does not seem to be a crucial parameter for these small size problems.

5. Results of the optimization

In this section we compare the results among the efficiency and quality of the solutions found over 4 different instances (or weeks) of the Ants-bidding algorithm with the previous system used at Unilever for the scheduling of the plant.

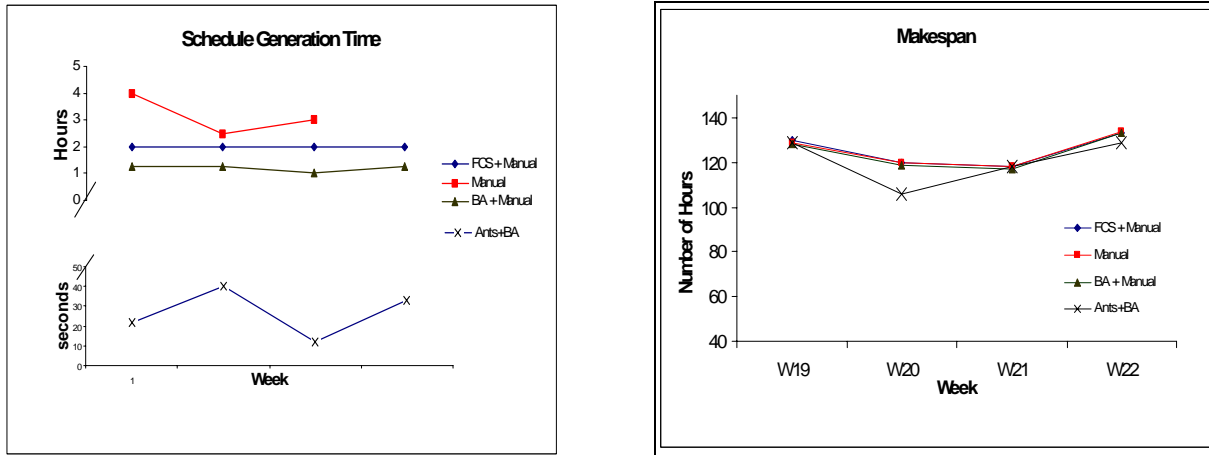


Figure 3: comparison of several algorithms used at Unilever (PCS, Manual, BA) with the new Ants and Bidding Algorithm. In a) the comparison is on the efficiency, in figure b) in quality of the solutions found.

As one can notice the Ants+BA algorithm can find optimized solutions extremely faster, of an order of magnitude of 10^2 and these are usually also better in quality. On average the improvement is 3.4%, which is significant on a week of operation.

It is also worth noticing that the number of changeovers is improved.

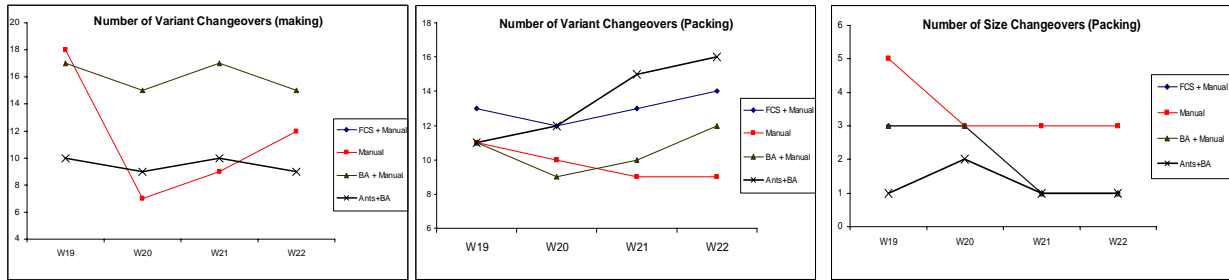


Figure 4: the comparison of the number of changeovers (a. making, b. and c. variant and SKU size on PL).

Also changeovers are improved as a secondary, implicit objective.

The overall average improvement in number of changeovers reaches 36% when compared to the BA+manual (that is the best algorithm in terms of makespan).

We notice that there is in general not only one optimal solution, but often several of them, given the degeneracy of having the maximum makespan as objective.

6. Glitches on packing lines

In this section we analyze the effects of having a glitch on a packing line. The capacity to cope with glitches depends mainly the “flexibility” of the plant. There are two opposite situations:

- the glitch falls on a PL that has a make-span shorter than the maximum,
- the glitch falls on the PL with the maximum make-span.

To visualize the scheduling and the effects of the glitch, we have developed an interface.

The figure shows the results of the optimization process. In Figure 5 (a. and b.), the first panel at the top shows the schedule for the mixers, turning on and off; in the middle part it is shown the corresponding quantities in the tanks, and in the bottom part the schedule for the packing lines, which gives the final maximum make span, also shown at the very top with the red line.

Variants (on mixers) and SKUs (on packing lines) are labeled with their names, and the ones with the same colors have the same variant. The black spots represent setup times, while dark gray are the maintenance times for the packing lines.

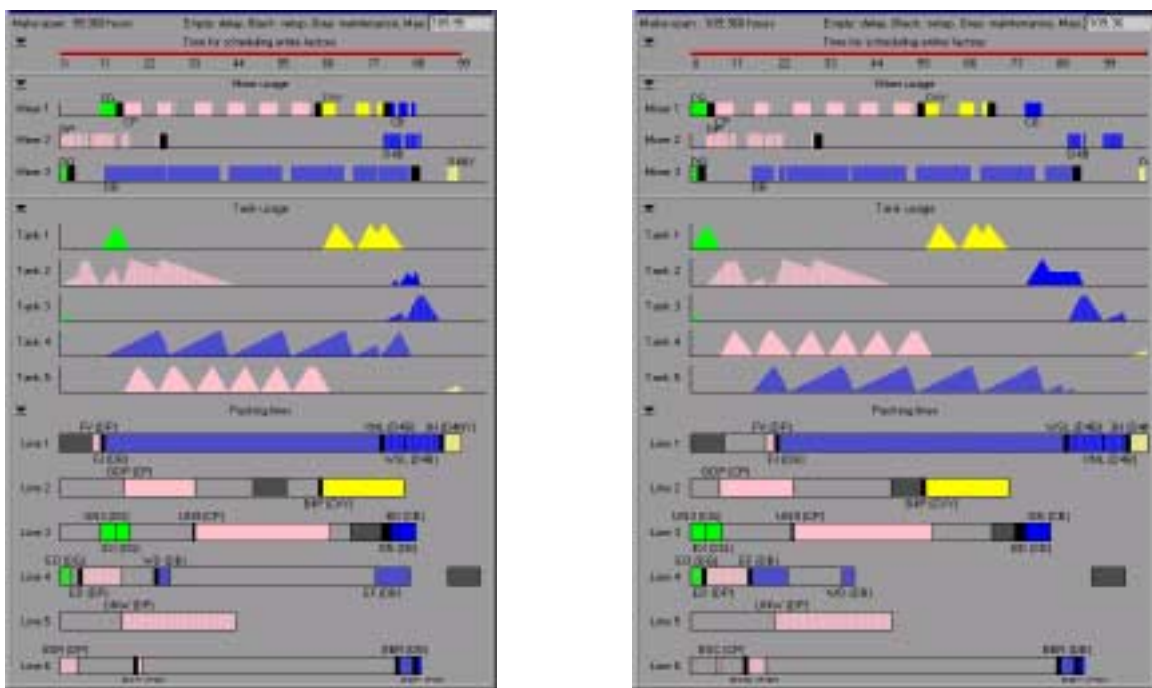


Figure 5: a) The glitch on packing line 3, is from 0 to 10, happen to be on a PL which presents already some delays and it is shorter than the maximum make-span. In this case there is no loss of time. b) Here the glitch is on the “critical” packing line (1), the one with the maximum make-span and no flexibility. The glitch is from 8 to 18.

In the first case there could be a minimum effect if not at all, and eventually the consequences of the glitch are contained if the length of the glitch is reasonably comparable to the empty times plus the difference with the maximum. If the glitch happens to be on the PL with the maximum make-span, the impact of the glitch is

essentially related to the flexibility of the plant, but with a dynamic scheduling eventually the loss of time is never greater than the duration of the glitch.

7. Other considerations

To gather statistical consistence, the algorithm is run several times over repeated experiments. This is useful for example for calculating the average best fitness or the average system fitness (the fitness of the average ant), as long as standard deviations of the fitness and the system fitness plus a number of other significant quantities one might analyze on the convergence process. The system fitness is the fitness of each ant (sampled every 10 iterations) and averaged over the run of the experiment. It is intimately related to the learning process. In the following figure the red lines represents the average best fitness over the runs of the best fitness found so far in the run, the blue squares the system fitness, and bars its standard deviation. If for example the greediness is high (g_0) the squares will lay very close to the red line, with smaller error bars.

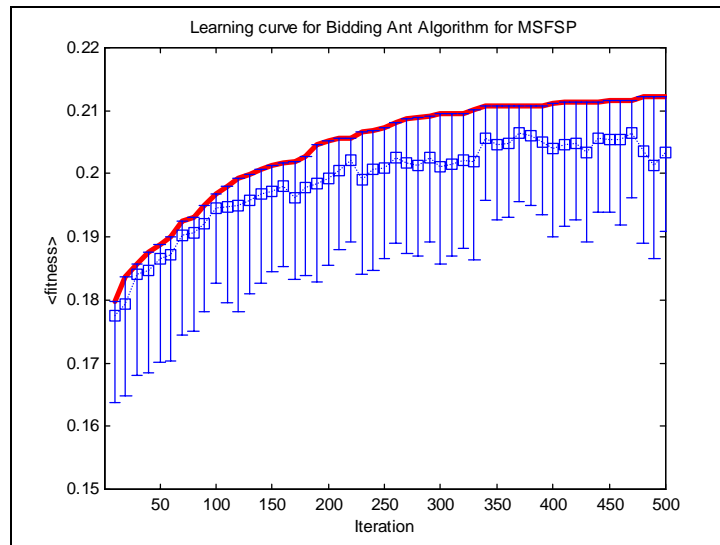


Figure 6: Learning curve: the (red) solid line represent the best fitness found during the iteration of the algorithm, the (blue) squares is the system fitness, and the bars its error. represent how good are the solutions found in a generic iteration, and how close are to the best solution (learning).

It is also useful to evaluate the distribution of the average fitness and its cumulative distribution C , related to the probability P of finding a solution for a given value of the fitness ($P=1-C$) and to phase transitions in the system, like is shown in the following figure.

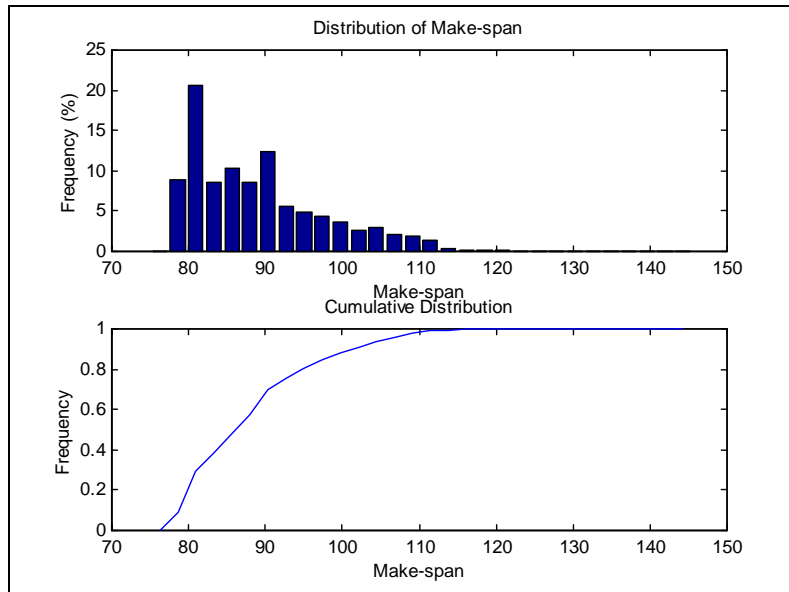


Figure 7: Make span distribution and its integral, of every agent that generate a solution during the iteration of the algorithm.

We note that the fitness distributions present a long right tail (right side), revealing the existence of a number of nearly optimal solutions.

Another relevant issue is how ants, once they have learned the optimal 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-breakdown. We can show that the pheromone memory allows the discovery of new solutions in a faster way. In the following figure, the optimization started as usual with a uniform distribution of pheromones is shown in red (lower curve), while the optimization when the pheromones are initialized with the distribution obtained at the end of the previous run, is shown in blue (upper curve).

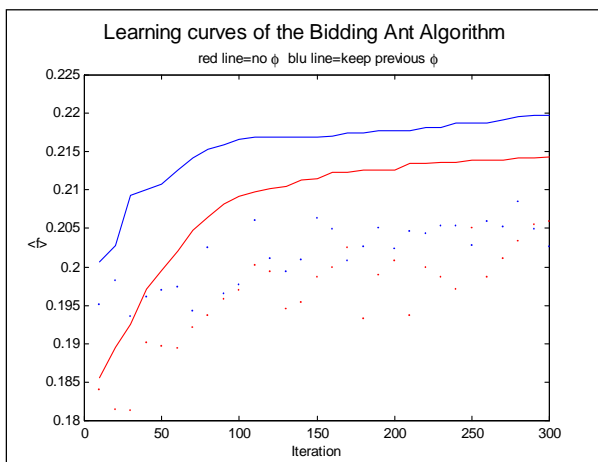


Figure 8: Learning curves: the (red) lower curve is a standard run of the algorithm. When the pheromones of the last run are used for the new run, the learning is much faster, as in the (blue) upper curve.

8. Design

The optimization can be run also at the scope of testing 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*.

It is possible to repeat the second set of experiments by changing each time the value of the demand uniformly for each SKU, and by initializing each time the pheromones to the average calculated over the first set of experiments. In the following 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. The following figure shows nine possible configurations for the plant, derived by variations of an original problem with 3 mixers ($M=3$), 3 tanks ($S=3$), and 5 packing lines ($PL=5$). For the first row of plots, each plot is relative to an increase of the number of mixers, starting by $M=2$, up to $M=4$. The second row, each plot is relative to an increase of the number of tanks, starting from $S=2$, up to $S=4$. The third and last row, each plot is relative to increase 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 axis represent the incremental variation in the number of the other resources of the plant (e.g. for the first row where the mixers are changing, in each plot the plane is variation of the number of PL versus variation of the number of tanks, S). The green bars are proportional to the standard deviation, or variation for the time to complete the schedule.

Here two important observations: for example on row 3, plot 1, we have that with $PL=4$, even increasing to $M=4$ ($\Delta M=1$) the number of mixers and to $S=4$ ($\Delta S=1$) the number of tanks we will not have a better schedule, than just having $M=2$ ($\Delta M= -1$), and $S=2$ ($\Delta S= -1$). This situation of having just 2 packing lines then is evidently is a bottle neck for the system.

In other plots, like row 2, plot 2, we can notice that increasing PL and M is really convenient 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 no convenient at all to go from here 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 not obtaining any improvements.

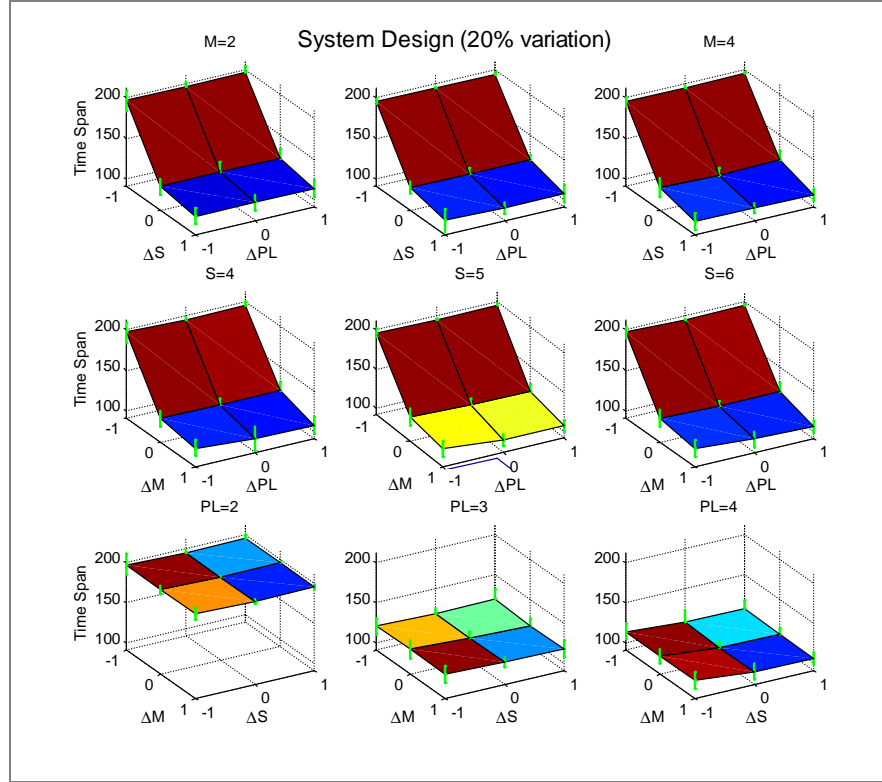


Figure 9: Plots of the average time to complete a schedule with different plants: in each plot we vary the number of resources available. Each row represents a different resource varying. first row mixers, second tanks, and third packing lines.

Dynamic Capacity

We can define a measure of the capacity of a plant by using a notion of number of possible different ways to manufacturing each SKU. Given that for each way there will be a time associated, in relation to the different processing rates of each resource, we can keep those into account. The total capacity is defined as:

$$(9) \quad C = \sum_{sku} \sum_R \min(rate(SKU))$$

R are the resources on which the SKU can be done where and the minimum is calculated among the packing line rate and the mixer rate where the SKU can be done. In this way also the connectivity between mixers and tanks is kept into account, as well the connectivity between tanks and PLs which being a dynamic connectivity, is kept into account in the following factor:

$$(10) \quad F = \frac{\min(nPLs, \maxFeed)}{\max(nPLs, \maxFeed)}$$

where $nPLs$ is the number of PL where the SKU can be done and $maxFeed$ the maximum number of connections at a time between the tank and the PL.

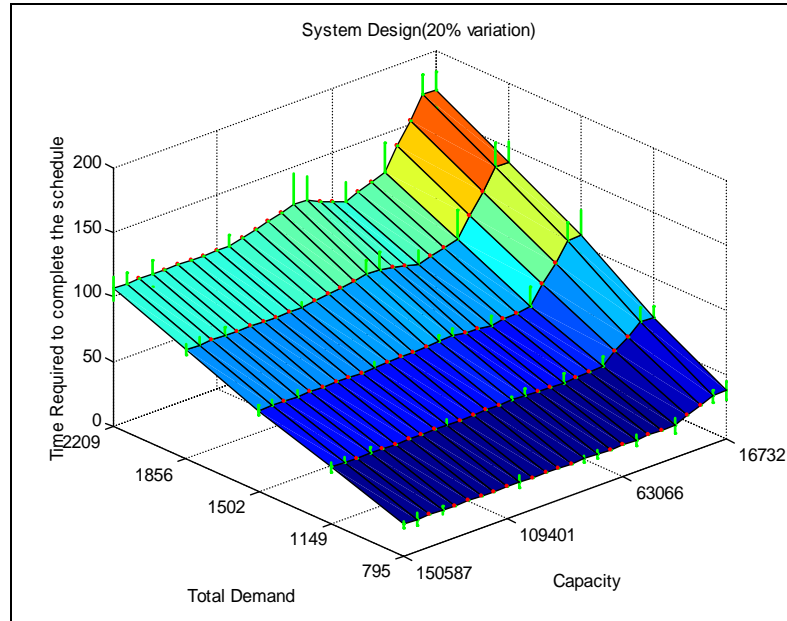


Figure 10: Study of the factors affecting the time required to complete a production cycle, based on the total demand and the capacity defined in eq. (9).

We generated a number of plant examples and a number of demand profiles (here the total demand is constantly increased by a finite amount). The z-axis is the make span, or time to complete the schedule, the (green) bars is the variation in the time when we also add, for each demand profile, a 20% uniform random variation in the demand (the red dots are missing data for plant profiles that has been interpolated). We note that a phase transition is present in this case for $C=35,000$ where we can notice a sudden drop in the time required to complete the schedule, to remain fairly constant after that when further increasing the capacity.

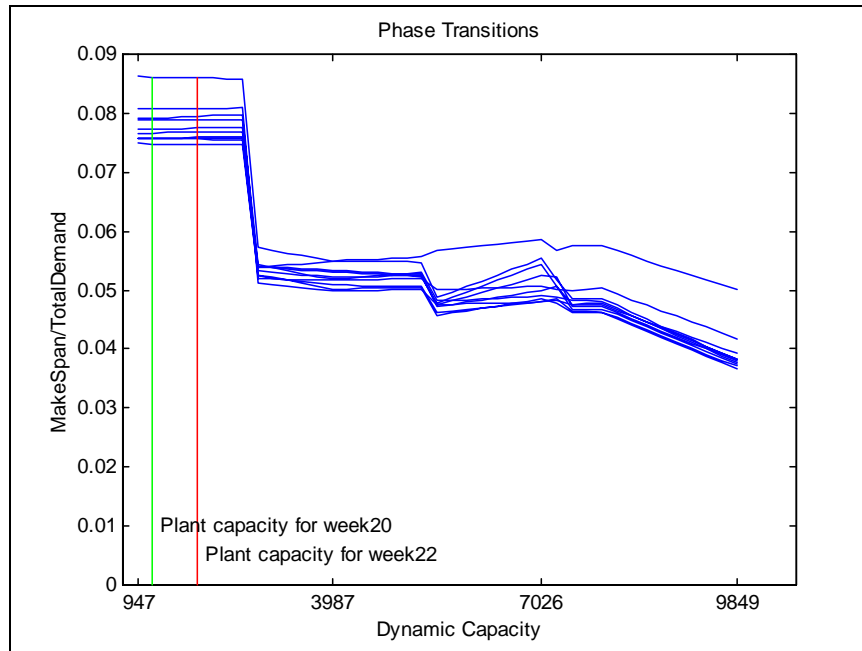


Figure 11: Phase transition in the time necessary to complete a production cycle. This figure shows that at an increase of the dynamic capacity, can correspond a relevant decrease of the production times. The 2 vertical lines represent 2 Unilever plants. This figure shows how to eliminate bottlenecks, so that with little investments (like improving connectivity among stages), the throughput can highly increase.

Interestingly enough, Unilever have anecdotal stories which agree with this picture. Over the years a given factory (say a toothpaste factory) has changed from making 12 kinds of toothpaste, to 15 kinds of toothpaste to 20, to 22, all without any substantial change in efficiency of production. However, one day the 23rd kind of toothpaste is introduced, and suddenly the factory just can't cope any more. Efficiency drops through the floor. This phase transition analysis is one very good explanation for exactly how that can happen.

Conclusions

We have introduced a number of aspects related to the scheduling of a multi stage liquid plant.

The use of an adaptive algorithm provides a better and more effective search in the solution space, and copes with rescheduling in case of malfunctioning or glitches, and variations in the demand.

The improved speed in the search of the solutions also allows studying many configurations of the system, under many different conditions, and to identify an order parameter and phase transitions, which might be crucial in the system design.

References

- [1] D. Gregg, “*Supply Chains Design and Operations: Basic Models for Investigating Generalized Flow Shop Scheduling and Design*”. Unilever Research, Port Sunlight Laboratory, September 1997.
- [2] M. Dorigo, G. Di Caro, L. M. Gambardella, “Ant Algorithms for Discrete Optimization”, in *Artificial Life*, vol. 5, no. 2, 137-172, (1999).
- [3] M. Dorigo, V. Maniezzo, A. Colorini, “The Ant System: Optimization by a Colony of Cooperating Agent”, *IEEE Transactions on Systems, Man and Cybernetics*, part B, vol. 26, no. 1, 29-41, (1996).
- [4] L. M. Gambardella and M. Dorigo, “Ant-Q: A reinforcement learning approach to the traveling salesman problem”, Proceedings of the Twelfth International Conference on Machine Learning, ML-95, pp. 252-260. Palo Alto, CA: Morgan Kaufmann, 1995.
- [5] L. M. Gambardella, E. D. Taillard, and M. Dorigo, “Ant colonies for the Quadratic Assignment Problem”, *Journal of the Operational Research Society* 50, 167-176, (1999).
- [6] D. Costa and A. Hertz, “Ants can colour graphs”, *Journal of the Operational Research Society* 48, 295-305, (1997).
- [7] L. M. Gambardella and M. Dorigo, “HAS-SOP: An hybrid ant system for the sequential ordering problem”, Technical Report 11-97, IDSIA, Lugano, 1997.
- [8] A. Colorni, M. Dorigo, V. Maniezzo, and M. Trubian, “Ant system for job-shop scheduling”, *Belgian Journal of Operations Research, Statistics and Computer Science (JORBEL)*, 34, 39-53 (1994).
- [9] V. Darley, A. V. Donati, “Ant Based Algorithms and problem difficulty for Generalized Flow Shop Scheduling and Design for Multi Stage Manufacturing”, BIOS Group, Internal Report, 1998.
- [10] L. M. Gambardella, E. Taillard, G. Agazzi, “MACS-VRPTW: Vehicle Routing Problem with Time Windows”, in *New Ideas in Optimization*, D. Corne, M. Dorigo and F. Glover (eds), 63-76, McGraw-Hill, London, 1999.