SIMULATION IN THE TEXTILE INDUSTRY: PRODUCTION PLANNING OPTIMIZATION

Gianluigi Ferraris
University of Turin
Email: ferraris@econ.unito.it

Matteo Morini, corresponding author
University of Turin and LABORatorio 'R. Revelli'
Email: matteo.morini@unito.it

Abstract— **The work being introduced is aimed at supporting the crucial activity of deciding what is to be done, and when, within an industrial, applied, real-world situation. More specifically: matching assorted tasks to applicable production units, and deciding the priority every job is to be given. The problem, common to many different industries, arises when a considerable amount of different articles must be produced on a relatively small number of reconfigurable units. Similar issues have a strong impact on an essential concern, eminently in the textile industrial domain: satisfying the always-in-a-rush customers, while keeping accessory production costs (set-up costs, machinery cleaning costs, ...) under control, keeping at a minimum the losses related to wasteful resource-management practices, due to "under pressure" decision making.**

Given the real-world situation, where human planners tend to be the only ones considered able to tackle such a problem, the innovation hereby suggested consists of an automated, artificial intelligence based, system capable of objectively driving the search and implementation of good solutions, without being influenced by pre-existing knowledge, mimicking a powerful lateral-thinking approach, so difficult to accomplish when management pressure impedes and daunting tasks bound the human rationality.

Ranking the effectiveness of a candidate solution, where path-dependency and unexpected complex effects may bias the final outcome, is not a matter trivially manageable by traditional operational research-style systems where no dynamics (recursive phenomena, feedbacks, non-linearity) appear. In order to overcome the limitations that an analytical specification of the problem imposes, the Agent-Based Modelling paradigm had to be taken into consideration.

Thanks to ABM we're provided with the opportunity of "in-silico" experimenting every imaginable scenario, by executing the planning in a virtual lab, where the production events happen instead of simplistically being computed. In this way we avoid following a reductionist approach, clumsily based on the usage of a static representation of the enterprise world, squashed into a cumbersome system of equations.

The model have been built resorting to the Swarm toolkit (see [Bur94], [JLS99], [MBLA96]); the underlying programming language (Objective-C) made the procedure of mapping the agents involved in the process onto software objects a plain and consistent task.

The problem presented belongs to the "shop problems" family in general, although many peculiarities make it an unconventional and distinguished one. When referring to "production planning", the authors have in mind the scheduling problem rather than ERP/MRP issues. In fact, the stage of the production on which the work is focused gives the availability of raw and semi-finished materials for granted. The up- and down-streams of the supply chain are normally performed by significantly oversized equipment, in the textile industry. On the other side, "core" processes, spinning and weaving in particular, require peak exploitation of the available production units.

KEYWORDS:

Production, Scheduling, Optimization, Industrial Processes, Manufacturing

I. THE PROBLEM

Matching tasks to units, under additional constraints, is the key issue. While certain constraints are to be regarded as "hard" (let's think of a technical issue rendering some of the production units useless in working on particular a (sub)task, thus reducing the set of available units), others are "soft" constraints: different units perform better on certain tasks, whereas others can suboptimally do, maybe with worse (yet acceptable) results, or take a longer time.

The sequencing of tasks is, on the other hand, one of the degrees of freedom of the problem, being the choice of giving priority to one task driven by timely delivery constraints.

For the sake of readability in this paper the words "order", "task", "job" will be used interchangeably.

A. Minimizing the production overall cost

Different production plans result in varying (aggregate) production costs. Each evaluation in terms of costs is made by adding several components: some of them are costs in a proper sense, others are more like abstract values by which we try to capture the economical impact of undesirable situations. Examples of the first kind are the setup costs; on the other hand delayed deliveries are certainly unwanted, even if not directly expressible as economical losses. Being considered an unreliable supplier because of repeated delays, in the long run, leads to unsatisfied customers being lost. This is, of course, an hardly economically quantifiable loss: it depends on how the firm's management perceives the importance of reliability, and how strongly is feared the risk of losing a repeatedly "deluded" customer.

B. Textile technicalities explained

The simulation is performed on and limited to, for the sake of simplicity, one of the production chain tasks only: proper spinning. Previous and successive operations can be overlooked, since they normally take place in oversized departments. Warping and combing, for instance, require relatively inexpensive machinery to be completed: it is common practice

to buy extra units 'just in case', since most of the plants value comes from spinners. The department where extreme care must be taken in avoiding any bottleneck effect is the spinning room.

We may confidently say that, should a good production plan be found for the spinning, the raw materials availability could be taken for granted, and the operations due to be performed up- and downwards the production chain could be arranged easily, not acting as constraints.

Finding a good production plan often implies dealing with mutually exclusive goals, in situations ridden with trade-offs. The only reasonable way to manage so many different aspects simultaneously is to reduce everything to its economical meaning, and it is hardly a straightforward task.

1) Production units setup: Spinners are complicated machines that can be adapted to produce many different kinds of yarns: apart from technical-mechanical parameters that can be tuned (speed, crossing angle, twisting…), each head (*see Glossary*) can be set up, by physically substituting some parts, to make for a wide range of technical specifications. Every kind of yarn features specific technical parameters and may require different parts to be mounted. At least three families (each one made by three types or more) of mechanical parts must be kept into account: cards, rotors, nozzles (*see Glossary*).

The act of setting up a spinning unit in order to have it ready to produce a certain kind of yarn may take a considerable amount of time: up to three hours may be spent removing and re-inserting a big amount of different mechanical parts, apart from trimming the appropriate software controls.

Of course putting similar products in sequence saves setup time: the least different two lots put in sequence are, the simplest and quicker the setup operations will be.

$$SC_{i,j} = f(\dot{p}_{1,i,j}, \dot{p}_{2,i,j}, \ldots, \dot{p}_{N,i,j})$$

The setup cost SC for order i placed after order j (or on a stopped production unit) depends on the dummy variables $p_{\{1<n<N\},i,j}$: each of them expressing the fact that the spinner part enumerated as the n-th (out of N) needs being exchanged when order j comes after order i (regardless of the spinner involved).

This seems a good enough reason to keep similar, if not identical lots, together, sticking them one after another. We'll see later why it's not that simple.

Nevertheless, the cost of setup can simply and accurately be accounted for in terms of man/hours spent performing the operation: after all, it consists of a sort of opportunity cost.

2) Timely delivery: Each order the firm is asked to produce is labelled with an "expected delivery date": customers are promised their yarn will be ready to ship by an approved calendar date (sometimes stringent conditions are imposed by "big" buyers), which they expect to be reliable. Should the delivery constraints be missed, a disappointed customer would, to say the least, complain bitterly. We have a situation which is very difficult to express in economical terms; very seldom a penalty is contractually established, rather the firm reputation is at stake, and the risk is to lose customers.

In order to keep into account, besides of the setup constraint ("less is better"), this additional constraint, a figurative cost has been introduced. It consists of an amount of money associated with the delay and the importance, positively correlated with both: the longer the delay and the bigger the order, the higher the (not-so-metaphorical) cost to be charged. Expressed in symbols:

$$DC_i = f(d_i^+, w_i^+)$$

where the delay cost DC for order i grows as the delay d and weight w (in kilograms) grow.

It becomes clear that sequencing similar orders on the same spinner is not an option: the freedom to save setup costs is at odds with the need to satisfy the timely delivery condition. A simplified example is presented (see Appendix, gantt sample).

3) Simultaneous setups, patrolling: To make things even worse (and almost impossible to deal with "by hand", which is nowadays the only viable way available to enterprises) further constraints are to be kept into account.

Production units setups, for instance, are performed by specialized workers; the number of setup teams available is limited, thus limiting the amount of setup operations which can happen at the same time. The effect of a missed setup (because of the unavailability of a team) on the production is simply a delay in the production of the order: no setup can be performed until one of the busy setup teams is available again. The total production time, and the time the order will be ready to ship, will be determined by the actual production time plus the initial delay.

Other employees are committed to the so-called spinners "patrolling": they are required to follow the ongoing production, ready to fix any problem should occur. A patroller is normally assigned 4 to 6 spinners to watch; the complication here arises from heterogeneity in the behaviour of different spinners: every different yarn features a specific likelihood to create (generically speaking) problems, that is to draw more or less attention from the patrollers. A patroller will be able to follow productions that are problematic up to a certain point: the average must be kept below this critical point. Above the limit, production times will grow (in a more or less foreseeable way) for all of the spinners under the overloaded patroller.

An index of "problematicity" is needed in order to manage such a subtle issue. The patroller load PL corresponds (for the n-th patroller) to the sum of the "problematicity index" p for each order i multiplied by the number of heads, h, available on the spinner j.

$$PL_n = \sum_{i=1}^{S} h_i p_j$$

Index PL is normalized in order to have 1 as the maximum tolerable patrolling load. Above this load, orders production times increase by empirically determined amounts:

PL	ΔPT	
$0 < PL \leq 1$	0	normal load
$1 < PL \leq 1.2$	$+10\%$	slight overload
$1.2 < PL \leq 1.5$	$+25\%$	severe overload
$PL > 1.5$	$> +25\%$	unacceptable overload

$$PT = production\,time$$

C. Evaluation by simulation

In order to evaluate the alternative candidate production plans, being able to rank them by "goodness", it takes a metric: a measurement of their own figurative cost. Such an operation needs to take into account the intrinsic complexity of executing the plan: each decision taken with regard to the assignment of a certain task conditions the subsequent decisions. While executing a plan two dimensions come into place: time and space; its evaluation cannot overlook this crucial assumption. Setup teams, for instance, may grant a total availability, compatibly with daily timetables, yet this can be suboptimal if compressed in a limited amount of time: queues tend to form.

A simulation was introduced, based on the enterprise design, which let us overcome the hard - if not impossible - problem of keeping track of such effects in the accounting. By simulating, all the production events are "made happen": formation of queues, delays, interaction among entities emerge spontaneously and are accounted for, when evaluating the total cost. This way avoids introducing tricks and approximations such as assigning pre-digested costs to unforeseeable events, using average (yet reliable?) values that render the accounting less accurate.

Exploiting a simulation also gives the advantageous chance to experiment with unlikely settings, or hard to observe in real-world situations. The need to evaluate by traditional computational techniques a production unit breakdown, for instance, one would be compelled to resort to an average "expected time between failures": this implies accepting two unrealistic assumptions, that we deal with a continuous phenomenon, and that the events are evenly distributed. By simulating, randomly occurring (and randomly lasting) events can be generated, while keeping probabilities within a pre-defined range: instead of an unrealistic continuous distribution we are correctly working on discrete events, with different durations.

An accurate cost tracking and accounting is instrumental to a good final result: the figurative cost of each plan enters the solutions generator (the genetic algorithm), where it is used to evolve subsequent generations of solutions. Even small distortions may disrupt the search process towards inefficient regions of the solutions space, prolonging computational times and considerably worsening the quality and reliability produced solutions.

II. Experimenting solutions

A. Agents: a local definition for an umbrella-term

The wealth of definitions and interpretations that coexist when "agents" come into play calls for a clarification: the agents hereby presented are to be intended as interacting no-minded software objects (in the Object-Oriented programming sense), whose main role is to encapsulate data, to make (mostly basic) computations and to pass informations back and forth. There is no communication protocol specification apart from the well-known getters/setters; the Swarm toolkit is used as an useful framework (see also [LS00a], [Ter98]) where software agents perform actions in a (perhaps sophisticated) time sequence by means of a scheduler triggering events, in this specific case in a deterministic way.

B. An Enterprise to experiment upon

The Enterprise Simulator is the module where solutions are experimented, that is where the simulation takes place. A model of the supply chain under scrutiny is used in order to watch candidate plans 'happen': the production process is represented in abstract, resorting to representative agents. Production units agents, setup agents, patrollers agents have been developed with the aim of giving simple yet exhaustive representations of their respective roles. Even production orders are embodied by dumb agents: objects encapsulating all the informations pertaining to the tasks to be performed, which are bounced between proper agents that act based on the informations they achieve from the orders themselves.

Presenting how the process takes place in the model is out of the scope of this paper; the steps - in a way absolutely adherent to the real process - implemented are: orders reception (in batch), orders dispatching to production units (filling queues), PUs setups, involving setup time computation after setup teams gathering, patrollers capacity reservation, production, repeat.

Ongoing and predetermined orders, already loaded on PUs and/or already due, are completed before initiating the candidate plan evaluation.

III. Inventing solutions (enter the Golem)

To find a good planning solution, given the enormous[1] set presenting itself, a Genetic Algorithm has been implemented, based on the well known AI paradigm first introduced by J. Holland (in [Hol75]).

The idea was to emulate the natural evolutionary process performing reproduction and death of structures that are representing a strategy. Provided that a whole set of structures is normally called "the population" of the GA, each of them is analogously named "an individual"; each one encodes a strategy into a binary string called "a genome". After having created an initial random set of structures, each of them is evaluated, one item at a time, by performing the strategy it represents, encoded, into an appropriate simulated environment. In this way a serial[2] evaluation of each structure can be

[1] An average spinning mill needs to plan about 50 jobs onto about 15 spinners at a time, which results in circa 10^{67} different feasible schedules. The weaving industry involves even bigger figures: up to 100-120 jobs to plan on 50-60 weavers, giving 10^{120} schedules.

[2] The process of evaluating populations is intrinsically parallel, being the population refresh step the only "pivot" operation which needs to wait for the completion of the individual-by-individual fitness assignments. For an in-depth presentation of the authors' works in this direction, see [Mor04], and thereafter in this article.

performed, in order to assign every strategy a value measuring its goodness: the so-called fitness of the individual. When the whole set has been evaluated, an evolution step can be taken: each individual is assigned a probability to reproduce itself (give birth to "offspring") and a probability to die, according to its fitness value: better-fitted genomes are assigned a higher probability to reproduce and a lower probability to die, and vice versa. Reproduction is made by copying and crossing two individual's genomes to obtain a couple of new structures to put into two new individuals; these newborn individuals will replace two old structures selected - from the previous generation - to die. By performing this algorithm in a loop the population becomes more and more fitted and the better types tend to spread into the population. The GA method is very useful when a wide set of alternatives has to be explored: it is general-purpose, it does not require any previous coded knowledge about the problem and it allows finding reasonable solutions in a short time.

To face the scheduling problem a special, but general, implementation of a GA has been employed. The goal was to set up a boosted GA, able to handle individuals composed by more than one structure, and structures defined on a very large alphabet. Another requirement was that this special implementation of a GA, the Golem, needed to handle special structures where all the alphabet symbols appeared only once[3].

The decision to write a special GA was due to the peculiarities of the problem to tackle. Each candidate strategy aimed to solve it can be split into two parts:

1) which machine will have to make an order
2) which priority will be assigned to each order

The two parts interact between each other in a complex way so the goodness of a solution depends on the goodness of each of them, but it is not possible to determine the contribution of each part to the performance of the solution. Both have to be evaluated simultaneously. Unless that, the contents of each part are very different and they could be coded in a highly different way. The first part could be expressed by a sequence of numbers, each of them identifies a unit, whereas the position of each code number identifies the order to be made. Adopting the same structure for the priorities the problem to assign univocal values to each order has to be faced. In addition the code numbers are defined on a set which cardinality is given by the number of machines the enterprise owns, while the cardinality of the priority set is defined by the number of orders the enterprise is going to plan. Resorting to the standard two-symbol $(0, 1)$ alphabet would have caused an ineffective representation of the solutions space, given the problem to represent each number in binary code every time the number of orders, or the number of machines, is not a power of two.

The Golem tackles the aforementioned issues by allowing the user:

1) to decide independently for each genome how many symbols need to be used by the coding alphabet, i.e. how many different values will be used in it
2) to decide a different length for each genome, i.e. how many positions it will include
3) to handle genomes where each symbol of the related alphabet will appear only once.

In addition the Golem was written taking into consideration:

1) the robustness of the methods exposed to the user, who can hardly misuse them
2) the efficiency (performance-wise) of the program

The Golem features methods to let the users' applications smoothly handle and control the search process. The user has simply to define the structure of the strings/individuals by coding the number and specific parameters for each of them: type (univocal or random), length, alphabet cardinality. The application (the *Enterprise Simulator* in this case) can conveniently interact with the Golem, demanding for an individual to evaluate and, after having performed the evaluation, returning the fitness value to the Golem. When all the population's individuals have been evaluated, the Golem automatically performs the evolutionary step. The Golem code has been optimized to ensure a high performance level, and has been regression–tested versus the earlier, more readable versions.

IV. Experimenting invented solutions, ergo suggesting the good ones

The evolution process performed by the Golem is driven by evaluating each single candidate solution appearing in the GA population. The production plans require an estimation as accurate as possible, incorporating every element of the dynamic interaction characteristic of the enterprise operations. It is the existence of such relationships among the intervening parts which distinguishes the problem as one of a complex kind: the aggregated outcome differs from what is obtained by the single components.

Keeping in mind the facts mentioned above, the unfeasibility of operating by decomposing the problem in parts is self-evident: the interactional effect would be totally missed; likewise, resorting to mathematical functions, static by their own nature, would imply neglecting all the time-related features, which are fundamental when it comes to plan actions intended to happen over time, being themselves subject to scheduling.

Computer simulation, by allowing management facts to happen in an artificial laboratory (the enterprise model), permits to quantify and express costs, whether figurated or not, generated by each candidate schedule, accurately and significantly, in order to promote the search for the best solution to the given problem.

The very same tool can be exploited in performing what-if analyses driven by human decisions, in order to rank GA-made solutions; this allows comparing what's produced by the human heuristics versus what's suggested by AI techniques, in a straightforward way. Plausibly it's the only viable method to provide a shared metric which permits, given the amplitude

[3]The so-called "univocal" genomes, where every symbol representing a job must not be repeated nor left out of every perspective solultion.

of the problem, to decide whether the search direction is a productive one or not.

In order to exploit the enterprise simulation to these purposes, the modelled objects are required to act as a bridge between the (scheduling) plan from the inferential method (the Golem) to the enterpreneurial metric.

In designing the Golem, that concerning this activity is just one of the advantages aimed at: the chance to use an extended (symbolic) alphabet solved some coding issues that during the first trials performed by standard AGs hindered the search process. An alphabet restricted to binary digits forces production units and orders number to be expressed by grouped symbols (as many as needed in order to the maximum value in the definition domain to fit); wherever the defined domain is less dense than the set of the natural numbers (when dealing with orders classified by differentiating their number by thousands or tenths of thousands, for instance), several non-significant solutions may appear. In such circumstances translation algorithms need to be employed, which, keeping such unfavorable factors into account, operate extraneous transformations (i.e. back-and-forth remapping) unknown to the AG; in the worst cases the same value gets assigned to formally different structures. Such behaviours can sensibly mislead the solutions learning and refinement process, keeping effective results from being efficiently achieved: execution times may stretch considerably.

A further issue emerged from the orders execution priorities. A standard GA in this case tended to produce non-univocal outcomes: the same priority may have been assigned to several different orders. Artificially differentiating equal values, based on the position within the structure for instance, might have impaired the GA abilities also in this situation. The system would have somehow been "deceived" by such artifacts. Providing the ability to opt between different operators, applicable to different kinds of genomes, the Golem could solve this issue too.

Achieving reasonable solutions quickly is fundamental to the enterprise: by analysing the experimental results a logarithmic trend of the solutions goodness have emerged clearly, functional to the number of evolutions performed. Practically speaking, the Golem is able to rapidly improve the solutions during the early stages of learning, while its productivity decreases as the optimum is approached. Going for population convergence appeared a suboptimal behaviour: halting the system after a certain number of evolutions seems way better than comsuming a long time in exchange for marginal improvements.

V. Preliminary Results

Although the system hereby presented is, from a developement standpoint, mature, its adoption at a production stage by the pilot plants involved is at its early phase. Nevertheless, batteries of tests on real-world data have been thoroughly performed.

The typical set-up involved sampling batches of orders from a random date in the past (picking up real-time fresh data very seldom, for reasons to be explain afterward). The results obtained were measured against random plans (averaged over multiple runs with different random seeds), against an ingenuous strategy[4], against human solutions.

The system has been evaluated at various stages of the search: although only after a 5-minutes run on an ordinary desktop PC (details in Appendix) the proposed solution is already better then the human-made, the performance level (the costs saved) improves quickly, yet asymptotically (see fig. 1).

A systematic comparison between results is hard to perform: historical data on previous production plans isn't always available; asking human planners to re-evaluate prior data sets is very likely to lead to biased solutions; the same happens with "live" data. Additional problems associated with hard-to-extract implicit knowledge are very likely to arise, when dealing with real-world situations. This has been kept into account, and comparisons have been performed both against ad-hoc solutions on datasets expressly and silently submitted to human planners and historical data, when available, hoping to level out bias.

Early - yet consistent - results have been presented and discussed with managers and experts, and they clearly show the superiority of the system presented. In the following table, results are shown as an indicator normalized versus the human performance (made equal a hundred), and represent the overall costs kept into account by the system, which of course neglects exogenous costs.

random	100.00
pseudo-FIFO (see note 4)	92.05
human	82.27
5' run	68.75
30' run	62.25
6h run	60.62

VI. Conclusions

Production planning constitutes a typically complex problem: the interacting parts taking part in the process makes impossible the application of traditional search procedures, based for most part on the decomposability of the problem as a prerequisite.

Given an (although limited) number of tasks to schedule, even the plain enumeration of the possible solution becomes practically unfeasible, given the combinatorial explosion implied. In this scenario the limits of applying heuristics based on human experience have appeared: the human mind attempts to solve the problem operating on limited subsets at a time, implicitly decomposing the complex problem, thus missing an

[4]The strategy, which is an oversimplification of the human way of scheduling jobs, consists of a sort of modified and refined "First-In-First-Out" method: jobs are appended to jobs with similar set-ups requirements that are already in queue on a given production unit; jobs with different set-up requirements are scheduled either on free PU's, if any, or the first PU expected to become available.

overall view on it. Every single decision taken on the assignment of a task onto a production unit constitutes a sensible "cut-off" on the solutions space, resulting in neglecting the exploration of large areas.

Implementing GAs let us exploit their implicit parallelism, both from a computational and an investigative point of view: starting from randomly generated solutions, avoiding predigested strategies, the GA also considers solutions that would be rejected by a human solver as absurd ones; not seldom innovative ideas are found among such apparently suboptimal candidates, and they are the ones that give superior results.

Apparently this is the main reason for the superiority of the system with respect to the human approach. It demonstrates itself far superior both in computation duration - efficiency - and final results - efficacy.

The system put into place constitutes, though, just a starting point: ways to improve the efficiency are being investigated and experimented, by distributing the "thinking" part of the work, the simulation, on several distributed nodes of a computer network, drastically incrementing the degree of parallelism of the computational process. At the same time work is being done on making the inferential engine (the Golem) more powerful, by introducing even more dramatic variations with respect to the standard GA's. The ongoing tests concern: clustered, cooperating GA's, and GA's featuring varying populations and variable-length individuals.

GLOSSARY

A brief list of technical terms relevant to the textile industry.

head: one of the (tenths to hundreds of) elements working on a single thread, constituting a spinning mill.

card: a toothed brush used to disentangle fibers.

rotor: a rotating device used in transporting fibers.

nozzle: a v-shaped element through which air flows.

APPENDIX

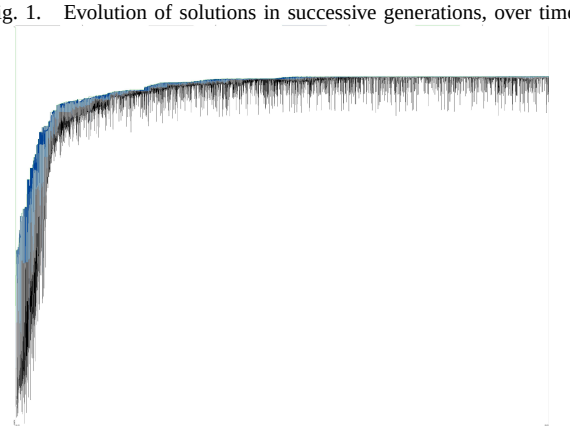
- Gantt example:

Customers *a* and *b* demanded, respectively, for [A1, A2] and [B1, B2]. Orders A1 and A2 are, from a technical standpoint, similar, and require a negligible setup time between them. B1 and B2 are also very similar. Ignoring (by now) the delivery constraints the obvious plan is to sequence similar orders on the same spinner (solution i):

spinner #	t_0	t_1	...	t_n
1	A1	q-A2		
2	B1	q-B2		
...				

The two customers, on the other hand, have different timing requests: a needs A1 and A2 as soon as possible; b is not pressing very much for a quick delivery and is fine for him to receive B1 and B2 by a later date. The most appropriate plan in this case would appear as follows (solution ii, grid entries changed from solution i have been italicized in order to let them stand out):

Fig. 1. Evolution of solutions in successive generations, over time



spinner #	t_0	t_1	...	t_n
1	A1	*l-B1*		
2	*A2*	l-B2		
...				

The small letters preceding the second orders are meant to show the different setup times required in both situations: as expected, q stands for 'quick' setup, l for 'long' setup.

Even in an oversimplified situation like the one described above, the complicated management of incompatible constraints appears; what makes solution i preferable over ii are the actual setup and delivery "costs", which must be accounted for as accurately as possible.

- Experimental set-up: technical details

The experimental gear used consisted of a rather aged desktop PC equipped with a single 800-Mhz Pentium-III CPU and 256 MB RAM. The amount of available memory becomes relevant when the GenomaBucket solutions caching system comes into play. It is beyond the scope of this article to present it; refer to [Mor03] for details.

REFERENCES

[AE94] R. L. Axtell and J. M. Epstein. Agent-based modelling: Understanding our creations. Bulletin of the Santa Fe Institute, 9(2), 1994.

[Axt99] R. Axtell. The Emergence of Firms in a Population of Agents. Brookings Institution, Washington, 1999.

[Axt00] R. Axtell. Why Agents? On the Varied Motivations for Agent Computing in the Social Sciences. Center on Social and Economic Dynamics, November 2000. Working Paper No. 17.

[BMVf] S. Bandini, S. Manzoni and G. Vizzari. Multi Agent Systems in Computer Science: Focusing on MAS Based Modelling and Agent Interaction, EXYSTENCE Thematic Institute for Complexity and Innovation, forthcoming.

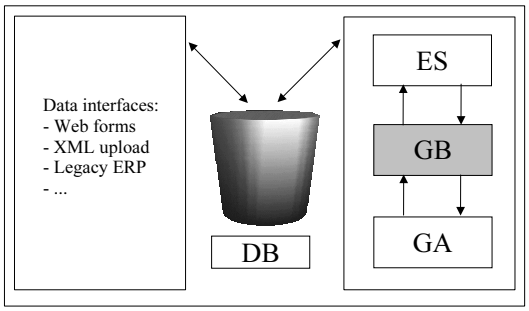
[Bur94] R. Burkhart. The Swarm Multi-Agent Simulation System, Position Paper for OOPSLA '94 Workshop on "The Object Engine", `http://www.swarm.org/archive/oopsla94.html`

[DG88] J.H. Holland D.E. Goldberg. Genetic algorithms and machine learning. Machine Learning, 3:95 104, 1988.

[Eps96] J. M. Epstein. Growing Artifi cial Societies. Brookings Institution Press,Washington, D. C., 1996.

[Eps99a] J. M. Epstein. Agent-based computational models and generative social science. Complexity, 4(5):41 60, 1999.

Fig. 2. Architectural overview



[Eps99b] J. M. Epstein. Learning To Be Thoughtless: Social Norms and Individual Computation. Center on Social and Economic Dynamics, September 1999. Working Paper No. 6.

[Fer01] G. Ferraris. GAMES: Algoritmi Genetici per l'Economia. Number 51 in Quaderni del Dipartimento di Scienze Economiche e Finanziarie G. Prato . Universit degli studi di Torino, Facolt di Economia, March 2001.

[GT00] N. Gilbert and P. Terna. How to build and use agent-based models in social science. Mind & society, 1(1), 2000.

[Hol75] J. H. Holland. Adaptation in Natural and Artificial Systems. MIT Press, Cambridge, MA, 1975.

[Hol98] B. Holmstrom. The firm as a subeconomy. In Bureaucracy: Issues and Apparatus, October 1998.

[HR99] M. Harris and A. Raviv. Organization Design. University of Chicago, July 1999.

[JLS99] P. Johnson, A. Lancaster, and B. Stefansson. Swarm User Guide. Swarm Development Group, November 1999.

[LS00a] Francesco Luna and Benedikt Stefansson, editors. Economic Simulations in Swarm: Agent-Based Modelling and Object Oriented Programming. Kluwer Academic Publishers, 2000.

[LS00b] Francesco Luna and Benedikt Stefansson, editors. Economic Simulations in Swarm: Agent-Based Modelling and Object Oriented Programming, chapter 9. Kluwer Academic Publishers, 2000. F.-R. Lin, T. J. Strader, M. J. Shaw, Using Swarm for Simulation the Order Fulfillment Process in Divergent Assembly Supply Chains.

[LS00c] Francesco Luna and Benedikt Stefansson, editors. Economic Simulations in Swarm: Agent-Based Modelling and Object Oriented Programming, chapter 10. Kluwer Academic Publishers, 2000. C. Schlueter-Langdon, P. Bruhn, M. J. Shaw, Online Supply Chain Modelling and Simulation.

[LTS96] F.-R. Lin, G.W. Tan, and M. J. Shaw. Multi-Agent Enterprise Modelling. University of Illinois at Urbana-Champaign, October 1996. Office of Research Working Paper 96-0134.

[MBLA96] N. Minar, R. Burkhart, C. Langton, and M. Askenazi. The Swarm Simulation System: A Toolkit for Building Multi-agent Simulations. Santa Fe Institute, June 1996. `http://www.swarm.org/`

[MT00a] J. P. Marney and H. F. E. Tarbert. Why do simulation? toward a working epistemology for practitioners of the dark arts. Journal of Artificial Societies and Social Simulation, 3(4), October 2000.

[Mor03] M. Morini, *Penelope Project: Web-Based Textile Production Planning*, SwarmFest 2003, University of Notre Dame, IN. `http://www.nd.edu/swarm03/`

[Mor04] M.Morini, *Penelope Meets NEMOTE: Distributed Production Planning Optimization*, SwarmFest 2004, University of Michigan, Ann Arbor, MI. `http://cscs.umich.edu/swarmfest04/`

[PCG99] M. J. Prietula, K. M. Carley, and L. Gasser, editors. Simulating Organizations, Computational Models of Institutions and Groups. AAAI Press The MIT Press, 1999.

[SLS96] T. J. Strader, F.-R. Lin, and M. J. Shaw. Information infrastructure for electronic virtual organization management. University of Illinois at Urbana-Champaign, October 1996. Office of Research Working Paper 96-0135.

[SLS98] T. J. Strader, F.-R. Lin, and M. J. Shaw. Simulation of order fulfillment in divergent assembly supply chains. Journal of Artificial Societies and Social Simulation, 1(2), March 1998.

[Ter98] P. Terna. Simulation tools for social scientists: Building agent based models with swarm. Journal of Artificial Societies and Social Simulation, 1(2), 1998. `http://www.soc.surrey.ac.uk/JASSS/1/2/4.html`