A Fuzzy-Evolutionary Approach to the Problem of Optimisation and Decision-Support in Supply Chain Networks

Sven Schellenberg, Arvind Mohais, Maksud Ibrahimov,
Neal Wagner, and Zbigniew Michalewicz

Abstract. This chapter deals with the problem of balancing and optimising the multi-echelon supply chain network of an Australian ASX Top 50 company which specialises in the area of manufacturing agricultural chemicals. It takes into account sourcing of raw material, the processing of material, and the distribution of the final product. The difficulty of meeting order demand and balancing the plants' utilisation while adhering to capacity constraints is addressed as well as the distribution and transportation of the intermediate and final products. The aim of the presented system is to minimise the time it takes to generate a factory plan while providing better accuracy and visibility of the material flow within the supply chain. The generation of factory plans within a short period of time allows for what-if-scenario analysis and strategic planning which would not have been possible otherwise. We present two approaches that drive a simulation to determine the quality of the generated solutions: an event-based approach and a fuzzy rule-based approach. While both of them are able to generate valid plans, the rule-based approach substantially outperforms the event-based one with respect to convergence time and quality of the solution.

1 Introduction

Understanding and managing a company's supply chain is one of the hardest tasks procurement planners and supply chain managers face in today's

Sven Schellenberg · Arvind Mohais · Neal Wagner
SolveIT Software, Pty Ltd., 99 Frome Street, Adelaide, SA 5000 Australia
e-mail: {ss,am,nw}@solveitsoftware.com

Maksud Ibrahimov · Zbigniew Michalewicz
School of Computer Science, University of Adelaide,
South Australia 5005, Australia
e-mail: {maksud.ibrahimov,zbigniew.michalewicz}@adelaide.edu.au

business environment. Ideally, a supply chain is driven by demands generated from customers placing orders. An order may include one or many order items composed of processed and unprocessed raw materials and components. In addition to the customers' orders which, generally speaking, draw final products out of the supply chain network (i.e., pull factors), supply chain networks are often subject to push factors which are caused by suppliers feeding raw material into the supply chain [1]. In most cases, the supply of unfinished goods cannot be synchronised with the demand generated at the other end of the supply chain, or an adjustment is delayed. Suppliers constantly, periodically, or spontaneously deliver raw material or components into the supply chain network. An arbitrary imbalance is created by customers and suppliers which the supply chain network tries to even out or trade-off (see Figure 1).

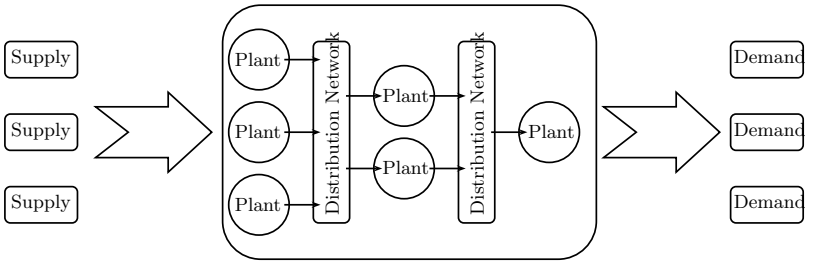


Fig. 1. Schematic view of multi-echelon supply chain network (3-echelon in this case)

Reconciling supply and demand by determining the amount of finished goods to produce becomes a labour-intense and time-consuming endeavour. It involves sourcing decisions when there are more than one internal or external suppliers of raw material, factory management decisions to define production plans and to determine maintenance outages, and decisions on how to effectively distribute the finished goods within the supply chain and to the end customer. Generating an optimal plan which takes all the previous considerations into account becomes virtually impossible for human operators who are in most cases only equipped with spreadsheet tools.

This chapter presents ways to synchronise and reconcile the drivers of multi-echelon supply chain networks demonstrated on a real-world example of an Australian manufacturer of agriculture chemicals. Although the model presented in this chapter aims to represent a specific supply chain model of a particular manufacturer, the components the network is based on represent supply chain entities as they can be found in many other businesses. We present two approaches which generate optimised production and material procurement plans. The system spawned off this project is currently evaluated and fine-tuned, and final completion and total business integration are expected to be finished in a few months. When deployed, the system

will facilitate the planning process. The planning includes determining the production across the supply chain (type and quantity of final and intermediate products), scheduling trains and trucks, making sourcing decisions based on contractual obligations, internal supply, and potential supplement deliveries. The operators will have the opportunity to regenerate plans as soon as the environment changes, obtaining a response within a few minutes. The prompt generation of plans allows for strategic planning which could not be achieved by the previous mode of operation. The power of what-if-scenarios can be harnessed to benefit from early structural decisions of the supply chain, such as added production, storage or transportation capacities.

This chapter details two approaches on how to balance the output of producing entities such as plants and factories with storage facilities like tanks, stock piles, or silos. The aim is to generate an optimised production plan for each site including decisions such as sourcing of raw material and production rate while honouring storage capacity constraints.

Although both approaches employ a simulation as part of the solution evaluation, the kind of simulation differs. In the first approach, an Evolutionary Algorithm (EA) tries to find a sequence of events. An event is defined by the date it occurs and an impact it has to the simulation state. The events are stored in a priority queue and executed in chronological order. An event could for instance cause a material changeover or a wind down of the production rate of a plant. An actual simulation only occurs at these discrete events (Discrete Event Simulation, DES); in between any two adjacent sample points (events), the system's state is assumed to be linearly changing, which means a tank's level can be inferred at any given time between two sequential events. In this particular implementation, the reduction of plant's utilisation in order to comply with capacity constraints is performed deterministically, by taking the excess production of a storage and propagating back to its supplying nodes, reducing their production proportionally to their share of the excess amount.

The second approach presented in this chapter does not use events to change the state of the system. As opposed to the previous approach, it tries to find the underlying rules that will balance the supply chain's producers and product changeovers. During the optimisation process, an EA generates a rule base and compares its performance on the simulation run at a predefined interval.

The two above approaches are compared in terms of their overall performance comprising violation of hard constraints such as storage capacity, total final product yield at the end of the run, and satisfaction of demand.

This chapter is structured as follows: After this introduction, the description of the problem is presented in detail followed by the description of the approaches to tackle the stated problem. Experimental results are given in the following Section 4. Before finishing the chapter with a conclusion and outlook of future work, the result of a literature review on related problems is briefly summarised in Section 5.

2 The Problem

In many industries which base their operations on multi-echelon production systems, procurement planners and factory operators often face the same sort of problems when they create production plans for a short or medium term planning horizon:

- What is the best supplier for raw materials (in terms of reliability, contractual bindings, availability, costs)?
- How much of it should be sourced in a given period?
- How much finished or intermediate product should be produced?
- When is the best time to schedule maintenance outages?

There are many more questions that can be considered. The bottom line is that the problem is rather complex, involves trade-offs which cannot be made in isolation, that is disregarding processes that happen further down the supply chain, and all decisions are heavily interdependent. Applying rule-of-thumb reasoning will not lead to an optimal or not even to a feasible plan if hard-constraints such as capacity limits are violated.

In order to understand the complexity of the presented problem and the manual interaction involved, let us consider the following example: A factory plan is supposed to be developed that balances the production and storage capacities of the supply chain presented in Figure 2.

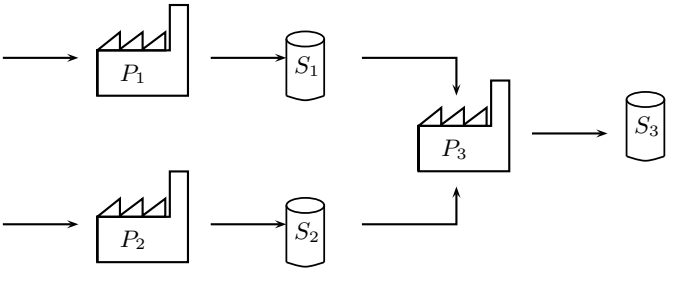


Fig. 2. Simple supply chain example to demonstrate complexity of balancing the components involved.

The network is composed of 3 plants ($P_1 \ldots P_3$) producing material M_1, M_2 and M_3 (P_1 produces M_1, P_2 M_2, and P_3 M_3) at a rate of 50 units per time unit t, and 3 storage tanks (S_1, S_2, and S_3). Assume that all the tanks have the same capacity of 100 units, and S_1 and S_3 are already filled to 80% of their maximum capacity. The planner tries to run the plants as hard as possible, that is, it is desired to run them on maximum production capacity. Running P_1 hard within a period of t means it produces 50 units of M_1 which are stored into S_1. Since there is a constant consumption of

M_1 by plant P_3 an overflowing of the tank does not occur. P_2 also produces its material M_2 and conveys it into its tank S_2. P_3 is sourcing its two input products and converting it at a ratio of 1:1 (which means we obtain 50 units of M_3 at the end of t from P_3). Since S_3 is already filled by 80 units (assume no consumption at this point), an excess amount of 30 units has been produced at the end of the time period. As the excess amount cannot be stored elsewhere, the production of the plant feeding into S_3 has to be reduced.

In this example, a utilisation of 40% would avoid the excess capacity to be created and storage S_3 to contain 100 units at the end of the period. Winding down the production in P_3 however reduces the demand of M_1 which is produced by P_1. The planner has to go back to the producer of M_1 and also reduce the production rate of this plant, as the material cannot be stored in S_1 (which is also filled to 80% of its capacity). This process of propagating back the plant utilisation has to be done for every storage facility that is exceeding its capacity. In this case, only two plants were affected, but for multi-echelon networks, it can be easily conceived that the amount of work that has to be done once a storage constraints are violated is enormous. The dynamics of the environment the plan is going to be implemented in often force the planners to re-create a plan multiple times (imagine unplanned outages of production plants as an example).

In essence, the process of creating such a plan can be very labour-intense, particularly in a real-world dynamically changing environment. Expanding the planning horizon to generate long-term plans (for instance yearly plans) is even more prone to change, as uncertainties and the level of their impact have implications on larger parts of the plan. A benefit of long term planning, the ability to plan strategically by evaluating what-if-scenarios, becomes virtually impossible when plans are generated manually. Analysing the impact of an event and generating a sufficient number of contingency plans is only possible if the time to generate those plans is very low and involves minimal amount of user interactions.

The proposed system automates the planning process to the extent that constraints are entered via an intuitive user interface and a plan is delivered within minutes after the optimisation process has been started. This allows for strategic planning as well as a prompt update as soon as new orders are placed which impact on the production schedule.

3 The Approach

In this section we describe two approaches that we developed to optimise the supply chain of the given business. Both approaches perform a simulation in order to determine the quality (or fitness) of the solution. The first approach generates a sequence of events and applies these to a discrete event simulation (DES). We call this one "Event-Based Optimisation" approach. A simulation is performed at any event occurring (next-event time advance [2]). In contrast, the second approach uses rules to make decisions for factory utilisation and

sourcing. The "Rule-Based Optimisation" approach uses also a simulation to evaluate evolved solutions, but advances the time at a fixed interval (fixed-increment time advance [2]). In addition to the supply chain model being used and the common parts of the EA, this section describes both approaches in detail and lists advantages and disadvantages.

3.1 The Supply Chain Model

When modelling a real-world system, careful attention has to be paid to the level of abstraction of the model. On the one hand, a high level of details improves the fidelity of the examined properties of the system, but on the other hand, the simulation process becomes computational expensive which prolongs the run time of the actual optimisation. A trade-off has to be made between reflecting as much as possible to draw the required conclusions from the system and minimisation of details. The model we employ for both of our optimisation approaches is described as follows:

The supply chain network can be thought of as a directed graph $G = (V, E)$ with $V = \{0, \ldots, n\}$ as a set of vertices and $E = \{0, \ldots, m\}$ as the set of edges. Each vertex/node can be of any of the three following types:

1. Plant
2. Storage
3. Switch

Although nodes of different type process material differently, the three node types have common properties. Each node has a set of predecessors which they depend on with respect to their source products. These predecessors are defined by the incoming edges of the vertex. Figure 3 illustrates a very simple supply chain network. *Plant 1* produces two products, stores both of them in separate storages and *Plant 2* converts *Product A* into *Product C* which is finally stored into *Storage 3*. Note that in this simplified version, *Plant 1* has no predecessors, whereas *Plant 2*'s predecessor is *Storage 2*. The conversion from a raw material into an intermediate or final product can be thought of as a chemical reaction with arbitrary input and output products.

A switch, the third kind of supply chain node, is not a physical entity, but it serves to route the material through the supply chain network. Due to the internal design and the paradigm to keep functional units as simple as possible, plants and storages are only allowed to source a material from one predecessor node. In case there is more than one predecessor nodes which output the same product, a switch has to be inserted which controls the material flow (see Figure 4). A switch can either implement a local heuristic to determine which successor to deliver to or it may be controlled by the optimiser which evolves the routing as part of its usual individual reproduction process (the switch becomes part of an individual's genotype). An example for the former may be that *Switch A* is implemented in a way such that it always exhausts the capacity of a tank before it starts filling up another tank

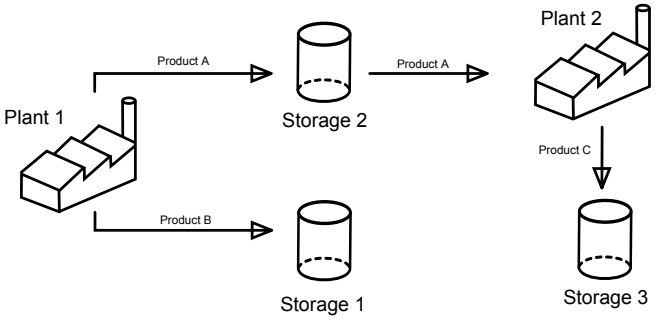


Fig. 3. Supply Chain model showing relationship between supply chain nodes

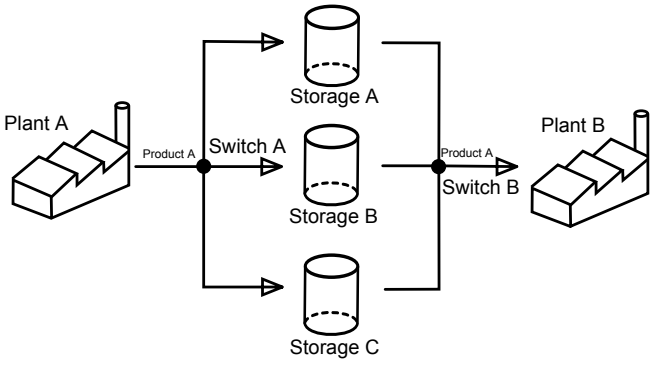


Fig. 4. Supply Chain illustrating material flow routed by Switches

(see Figure 4, *Switch A*). The local deterministic heuristic incorporated into *Switch B* may reduce the storage's level evenly draining all tanks at the same time.

In addition to nodes having predecessors, nodes also contain output buffers (one per product) in which the products they produce or store and their quantity are temporarily kept. Switches do not contain output buffers, but delegate requests for material to their predecessors depending on their implemented routing logic.

The previously described node-predecessor relationship only works for continuous material flow such as liquids that are pumped from a producing plant into a storage tank. In many supply chain networks, the distribution of material is done via transportation means with limited capacity or infrequent transportation times (trucks, ships, air planes or railway). Therefore, an additional property of the edge between two nodes is a schedule defining the availability of the transportation means and its capacity. Only when the transportation means is available at the plant or storage, it can empty the

node's buffers and store the material temporarily in its own buffer. Plants either have to shut down or store their products into adjacent tanks at times at which the transportation means is not available.

Given the supply chain network of the particular business, a simulation process can be performed that is mostly similar for both optimisation approaches. The only difference is in the way the decisions are inferred that change the state of the supply chain network.

Before the simulation is started, the supply chain nodes have to be initialised and a list of the sample points (event queue) will be created. During the initialisation process of the nodes, the buffers of some storage entities are filled to simulate an opening stock. The next step is to determine the points of the planning horizon at which the system is sampled. These points are the set of all events that occur during the planning horizon, such as

- Change of availability of a plant (plants may run at reduced run rate due to partial maintenance or outages).
- Product changeovers.
- Arrival/departure of transportation means travelling between two nodes.

It is of paramount importance to the validity of the simulation to add all events that cause a discontinuous change of the supply chain network's state, that is, a change that causes nonlinearity which would prevent inference of nodes' properties in between two adjacent sample points, to this event queue. If desired, additional sample points can be added to verify the result of the simulation.

Once the event queue is filled, the nodes are sorted by precedence. Nodes without predecessors occur first whereas terminal nodes that base their buffer's material and quantity on the result of all previous components' production are located last in this list. The next step is to iterate through the list of predecessors (beginning with the least dependant node) and process the node. Processing a node breaks down into three steps:

1. Pull resources (raw material, intermediate material, final product, etc) from predecessor.
2. Apply the conversion rule.
3. Push the converted material into output buffer(s).

The first step is straightforward: Since each node knows about its predecessors and the material it required, it pulls the maximum amount of this material from each of its predecessor nodes. Switches forward the call to their pull method to the appropriate predecessor. Essentially, the pull step boils down to copying the content of the predecessor's output buffers and passing it on to the conversion step.

In the conversion step, the actual business logic for each plant is implemented. If we consider a chemical formula like $1A + 2B \rightarrow 2C + 1D$ with hypothetical elements A, B, C and D, the factory that processes this formula would source all the material it could get for product A and B and determine the quantities of the resulting product C and D (taking into account

the ratio of A:B=1:2). The amount of input material, which has actually been used during the conversion process, is reduced from the predecessor's output buffers to account only for the actual consumption and to determine overproduction. While *switches* do not implement a conversion routine, they pass on the incoming materials to their output buffers.

Finally in the last step, the produced material is stored into its output buffers to be available for the successor node's processing procedure. At the end of the sample cycle, that is, once all nodes have been processed, the simulator captures the state of the system by storing the buffers which contain the quantity/products tuple for each node. The final buffer capacity allows determining the plants' utilisation or storages' remaining capacity. The simulation terminates once the planning horizon's end date is reached.

3.2 Features of the Optimiser

The meta-heuristic used to optimise the supply chain network is a steady-state EA, that is, only one individual alteration occurs at each generational step, replacing a parent. EAs are well understood, reliable and they can be customised to solve a variety of problems of different domains. They have been applied to other problems of similar level of difficulty and performed satisfactory. This section recapitulates the general working principle of the EA and elaborates on the specific implementation used here.

The EA operates as follows: The population contains a number of individuals (approximately 100 individuals) which encode a solution in their genotype. After initialising and evaluating the individuals of the population, the recombination process is started. As part of it, an operator from the set of available operators is chosen to alter the individual. These operators are application-specific as their operation strongly depends on the encoding chosen as well as the particular business problem they try to solve. Some operators may have the ability to consider certain business constraints to operate in the feasible search space which reduces the search space and leads to faster convergence. The specific operators used for both optimisation approaches are explained in the following sections.

After the operator changed the individual, the new individual is immediately evaluated. A self-tuning procedure compares the fitness value of the individual before and after the operation and adjusts operator weights according to whether the new individual yields a better solution or not. The weight is taken into account the next time an operator is chosen. In case an operator performed poorly, it is less likely to be selected for evolving the next individuals (similar to roulette-wheel selection). After a defined number of generations, the weights are reset, so dominating operators have to proof again their performance. This is particularly important if the optimiser prematurely converges and the optimisation process gets stuck at a local optimum. The previously preferred operator would not contribute to

improving the solution's quality which means other (potentially more explorative) operators come into play.

The evaluation procedure uses application-specific measures to determine the fitness of the individual. For this particular optimisation problem, both optimisation approaches use an identical fitness evaluation function. Part of the evaluation is the simulation. Once finished, the system state is evaluated. The objectives of the optimisation process for supply chains include, but are not limited to, (a comprehensive list of measures of supply chains can be found at [3]):

- Maximisation of product yield.
- Minimisation of product changeovers.
- Adherence to a specified product ratio for the remaining planning period.
- Satisfaction of demand.
- Minimising transportation, inventory, backlogging costs.

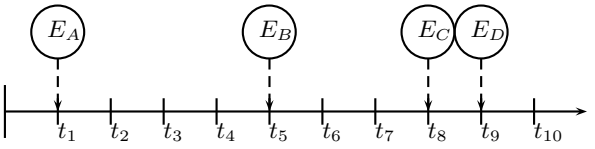
The optimisation process is terminated once a pre-defined number of generations is reached or if the search stagnates over a certain period of generations. The best individual that was found throughout the optimisation process is returned (keep-the-best strategy [4]).

Both techniques used in this chapter leverage of the same optimisation engine described in this section (and in fact applications for other customers do as well). Different business rules are encoded within the individual and as operators which makes the EA reusable for other optimisation problems.

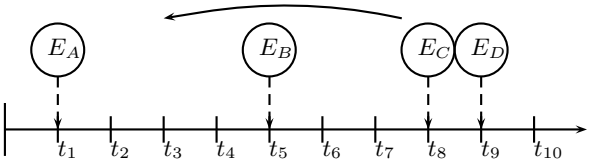
3.3 Event-Based Optimisation

The event-based optimisation approach tries to determine a sequence of events that, when applied to the simulation, results in an optimal solution. Events are defined by the date they occur and the action they perform, i.e., the state change they cause to the system. Examples are a changeover event in a factory which causes a factory to produce a different product, a change in the factory's utilisation, a change of the availability of the factory or a factory specific event such as a cleanout of storage tanks. The event-based approach has a very shallow hierarchy of representations. From the event sequence (which can be understood as the genotype) a conversion into the final solution is made by means of the simulation.

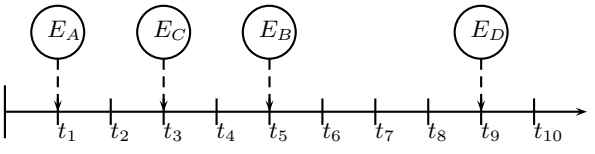
At the start of the optimisation run, each individual is initialised with a random sequence of events. The operators alter the event queue in different ways. They change the type of event (which changes the action they perform) or the date of their occurrence as illustrated in Figure 5. Some of them insert a delay at a specific time which causes all subsequent events to be delayed. A crossover operator randomly determines an event of the event queue of two individuals and swaps all of the following events with the other individual, similar to the classical crossover operation for genetic algorithms.



(a) Event queue before mutation



(b) Mutation operator in action changing date of E_C to t_3



(c) Event queue after mutation. E_C now alters the state at t_3 according to its encoded type.

Fig. 5. Timeline of the simulated horizon illustrating the impact of the date mutation operator on the event queue (dashed lines indicate the time events occur, i.e. when they cause a change of the state of the simulated system).

One part of the evaluation process is to transform the genotype of the individual (the representation of the solution we apply the operators to) into a representation that can be evaluated (the other part is to aggregate the fitness value components). The transformation is done by running the simulation according to the encoded sequence of events. At each of the event dates, a simulation is executed which samples the state of the system (e.g., fill level of storage tanks, excess level, utilisation, amount of raw material sourced, etc). After the event date, the system state will change non-linearly (for instance, a different product is produced; a plant shuts down its operation, etc). Once the first simulation pass is completed a repair function is triggered. Before the repair, each plant's produced products are consumed by their subsequent storages, disregarding the current available storage capacity.

The repair function deterministically winds down the producers (the plants) in order to balance the production with the available storage capacity. This process starts at the last storage in the supply chain, i.e., the one that relies on all of the previous nodes, and works back to the first node

in the supply chain (i.e., the one without any predecessor nodes). The excess amount of an overflowing storage is proportionally reduced from the previous producers in order to determine the appropriate production rate (planned utilisation). Each time a plant's utilisation is adjusted, a partial simulation has to be performed again as the minimised production has implications to other supply chain nodes downstream in the supply chain network. The advantage of this deterministic method to compute the utilisation is that these kinds of events do not fall into the search process and, therefore, decrease it.

On the other hand, this way of adjusting the plant's utilisation is rather expensive as the simulation is rerun every time the utilisation is changed. An optimisation process not honouring the storage constraints, that is, with disabled repair function, was able to evaluate about 1500 individuals per minute, whereas a normal optimisation process (repair-enabled) could only evaluate a maximum of 150 to 200 individuals on the same machine and the same supply chain network (this comparison of course is not exact as violated storage constraints result in more raw material being available which skews the production plan and the obtained product yield).

3.4 Rule-Based Optimisation

The approach proposed in this section is based on a combination of fuzzy logic and EAs. Rather than directly evolving decisions made during the simulation of the supply chain (in form of events), rules are generated which drive the decision making process.

Since this book is mainly about EAs, a very short introduction into Fuzzy Logic is given in this section. Thereafter, the application of fuzzy logic to the subject of this chapter, optimising supply chains, is demonstrated and the particular implementation is discussed by providing examples on encoding and decoding of the individual's genotype.

3.4.1 Fuzzy Logic

Fuzzy logic is based on the fuzzy set theory in which values are expressed as degree of membership rather than crisp inputs like discrete measurements. If we were to classify the age of a person having two categories (young and old), classical Boolean logic would either map a given age to young or old. Let's say the cut-off point separating old from young would be exactly 40 years of age. In Boolean logic, every person below 40 years would be young, whereas everyone with an age greater or equal to 40 is classified as old. Instead, in fuzzy logic, an age can belong to multiple categories (or degrees of membership). The degree of membership is denoted by a number between 0 and 1. Given two membership functions *young* and *old*, a person aged 30 years could be young to the degree of 0.8 and old to the degree of 0.3 (depending on the shape of the membership functions). Fuzzy terms such as

"age is young" are called linguistic terms ("age" is a linguistic variable and "young" a linguistic value). Those terms can be combined by Fuzzy-And or Fuzzy-Or (or other fuzzy operators, or T-Norms [5]) and further extended to IF-THEN rules of the form "IF age IS old AND health IS bad THEN health-insurance-premium IS high" (The IF part is called "antecedent" and THEN part is named "consequent"). Many of these rules can make up a rule base.

The type of fuzzy logic system (FLS) as it is used for this approach is called Mamdani-type FSL [6]. The fuzzy inference process can be decomposed into three major steps. First of all, the crisp input data obtained by measurements is transformed into fuzzy sets (fuzzification step) by determining the degree of membership of the fuzzy terms specified in the fuzzy rule applied. The next step combines the degrees of the fuzzy sets by the given fuzzy-operators and summarises each term into a rule weight. This rule weight is used to determine the output of the rule by limiting the shape of the output function. The last steps, the defuzzification, combines all reshaped output functions and applies a defuzzification function in order to obtain a crisp output value. This function is usually the centre of mass or a weighted average function.

3.4.2 Application of FL to SC Optimisation

Translated to our actual problem of generating decisions to drive a supply chain simulation, the fuzzy rules determine in their consequent part when to schedule product changeovers (that is which products are produced at which time) or the utilisation of the factories (utilisation of the facility). The advantage for using fuzzy logic to encode the rules driving the supply chain decisions is that these natural language rules facilitate diagnostics and audit features of the system. A planner controlling the system can deduct the reasoning of the system by analysing the rule bases which is more intuitive than the previous event-based approach.

Another downside of the event-based approach is also that the EA has to find the correct event for each time, albeit the conditions may be similar to a previous point in the plan at which the correct decision was made. Say a product changeover has to be triggered every time a storage tank is about to overflow, as the product is consumed from the overflowing tank as soon as the changed-over product is produced. This changeover may not be triggered at another point in time as the EA has not yet generated such an event at this time. It may or may not randomly generate such an event at the particular time. The rule-based system however would have had developed a rule that triggers a changeover upon reaching of maximal storage capacity of the tank, which means every time the condition holds (storage tank is high), the changeover is triggered. The application of rules makes the generated plan become more predictive and coherent. In addition, the underlying logic can be investigated and manually fine-tuned.

The individuals that are evolved in the evolutionary process encode the rule base. We decided to generate the rules by the EA as they may differ for different settings and different products. Evolving the rule base by the EA allows adaptation to the current constraints and environment. There are many other means to combine EAs with fuzzy logic. They differ in the part of the fuzzy inference system that is subject to the optimisation. Genetic tuning for instance changes the database (shape or number membership functions, linguistic terms) of the fuzzy inference system. Other methods evolve the knowledge base or generate new knowledge base components. An elaborate taxonomy and survey on methods to combine genetic algorithms and fuzzy logic systems (GFS) can be found at [7].

A fuzzy rule (consider this example for explanatory purposes: `IF Level_Of_Tank IS high THEN Utilisation_Of_Plant is low`) of the structure consists of an antecedent part (`IF ...`) and a consequent part (`THEN ...`). The antecedent can contain multiple linguistic terms (`Level_Of_Tank IS high`) which can be combined by different operators (T-Norm). For our purposes, Fuzzy-And and Fuzzy-Or are sufficient. A linguistic term has two parts, the linguistic variable (`Level_Of_Tank`) and the linguistic value (`high`). The number of linguistic values can be arbitrary, but in order to reduce the complexity, we opted for 5 linguistic values and triangle membership functions (see Figure 7).

The possible linguistic variables are the level of each storage tank, the currently produced product of a plant and previous utilisation. A set of rules forms a rule base. For the proposed system, a number of rule bases is created, one rule base per possible consequent part. Since the linguistic variables for all the rules of a rule base are the same (that is, all rules in one rule base pertain to the utilisation of a specific plant), the only additional information

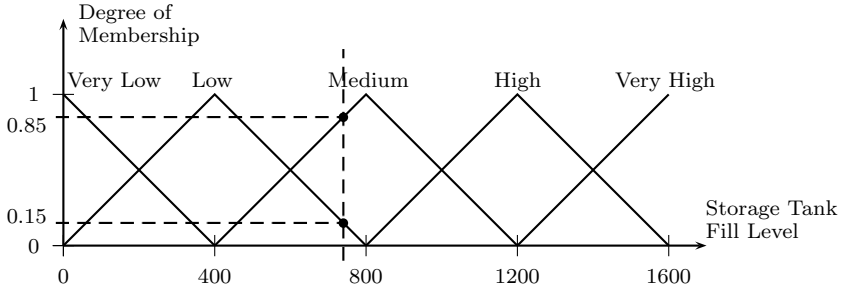


Fig. 6. Overlay of 5 membership functions denoting the fill level of a tank. At a level of 740l, the tank level is member of the "low" function by 0.15 and member of the "medium" function by 0.85. For each storage, the parameters for each membership function are different as the maximum capacity may differ (and hence "very high" would relate to a different maximal level).

that needs to be encoded in the chromosome is the linguistic value of the consequent part. Figure 6 displays the integer number vectors encoding each rule. A rule base to n rules encoded as sketched in Figure 9: The first index represents whether the rule is enabled, the second specifies the fuzzy operation which combines the linguistic terms (Fuzzy-And or Fuzzy-Or) and the following pairs of integer values contain a pointer to a look-up-table of linguistic variables and linguistic values. The last cell holds the linguistic value for the rule base's consequent part.

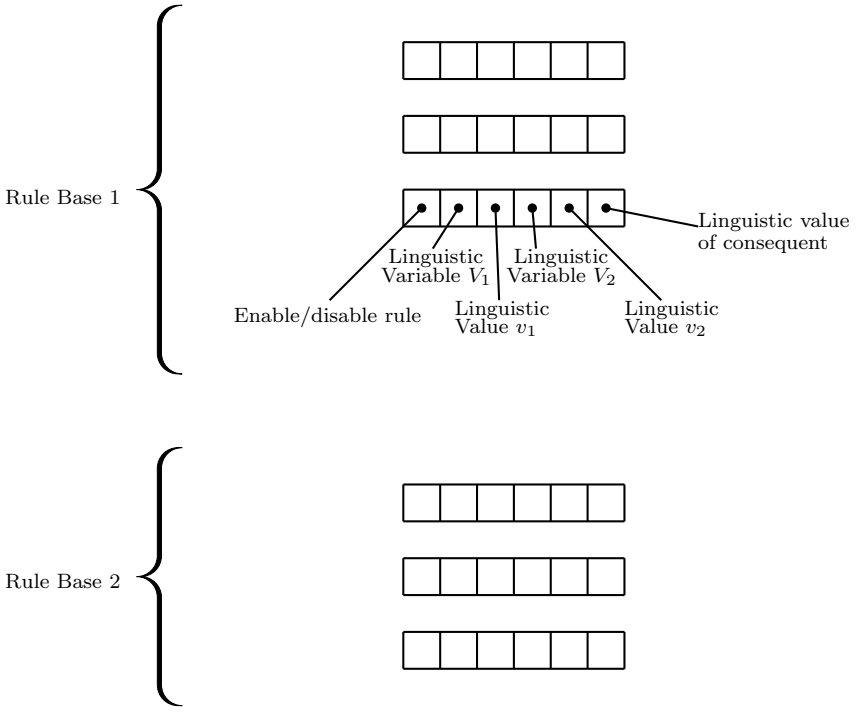


Fig. 7. Encoding the rule bases. Elements in squares are of type integer number and stand either for booleans or indices in look-up-tables that hold linguistic variables and values.

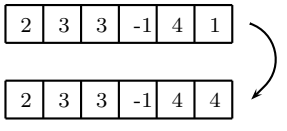
3.4.3 Generation of Rules

As in the previous section, the same EA with its self-calibrating capabilities is employed. The only differences lie in the method of evaluation, that is, how the simulation is carried out, and the set of operators used to evolve the individuals (as the encoding of the individuals). The evaluation function itself is exactly the same. It takes into account the maximal production yield, a

penalty is deducted in case storage constraints are violated and also a penalty in case of a delay in providing enough final product to satisfy firmed orders is applied.

Unlike the event-based optimisation, the rule-based approach samples the system at pre-defined intervals. At these sample points, all properties of the simulated system are evaluated and actions are derived from the current state. These actions are triggered by the rules described above. A rule may pertain to the fill level of a storage shed and cause a reduction of the feeding plant upon reaching of a "high" fill level.

Different operators modify the genotype of the individual at each recombination step. A mutation operator randomly changes bits of the genes by honouring the feasible maximal possible integer number value at the position in the chromosome. The meaning of such a mutated chromosome changes in the decoding step which leads to different rules and thus different decisions in the simulation step (see Figure 8).



(a) Mutation of chromosome (before and after mutation)

```
Before: IF Level(TankC) IS high
THEN Util.(PlantA) IS low

After : IF Level(TankC) IS high
THEN Util.(PlantA) IS very high
```

Index	Ling. Variable	Ling. Term
0	Level(TankA)	very low
1	Level(TankB)	low
2	Level(TankC)	medium full
3	Level(TankD)	high
4	Util(PlantA)	very high
5	Util(PlantB)	
7		
...		

(b) Phenotype decoding

Fig. 8. Impact of mutation on decoded phenotype ('-1' means the term is not considered in the rule)

Another typical operator for genetic algorithms, the crossover operator, was also adapted and implemented. Two flavours of single point crossover are employed. One of them cuts individual rules of two parent individuals into two pieces (at a random crossover point) and swaps its right-hand part with the other parent. The other crossover operator swaps entire rules at once by determining again a crossover point and replacing one part with the parent's rule set.

4 Experimental Results

We have tested the two approaches on two identical supply chain networks. The aim was to maximise production while honouring storage constraints. The EA was configured to terminate its search after a maximum of 5000 generations, or prematurely if the search would not yield any improvement within 1000 generations. By virtue of the system, the event-based approach runs a simulation whenever an event occurs. The rule-based approach was configured to sample the system at a fixed interval of one day and change the system state by applying its rules.

Both algorithms were able to generate feasible solutions without violating constraints. The quality of the averaged solutions of each approach, however, differed significantly. While the event-based approach managed to fill up the final product storage shed to 172,000 tonnes (see Figure 9), the rule-based approach exceeded this value by 37% (233,000 tonnes). In addition, the search procedure of the latter one terminated much earlier (on average at about 1300 generations) while the event-based algorithm used up the full span of available generational cycles. Another interesting observation is that the time it takes to evaluate an individual is much less for the rule-based approach (approx. 800 individuals per minute vs. 150 i/min). This and the premature termination caused the rule-based approach to find an (even better) solution after a few minutes of run time only.

As already stated in the introduction of this chapter, the system presented is applied to a real-world problem and, as such, it is difficult to compare it to synthetic problems as they are usually used as a baseline in academia. The only plausible baseline can be obtained by comparing the factory planner's schedule and the expected product yield with the schedule generated by the system. Preliminary test results indicate a high degree of similarity between human and system generated factory schedules with respect to the length of product runs (or in other words, the date of scheduled product changeovers) and accumulated product yield at the end of the planning horizon. Taking only these few measures into account, one can conclude that the model adequately represents the supply chain. The time it takes to generate a yearly plan by the planning personnel is tremendous. This means a what-if-scenario analysis becomes virtually impossible. Unless we deal with strategic what-if-scenarios, the result of such a scenario would become obsolete by the time it is obtained. Using the proposed system, a near-optimal plan for an entire year could be created in less than 10 minutes for the event-based approach and about 2 minutes for the rule-based approach respectively (carried out on a standard Dual Core 1.6GHz computer optimising a 5-echelon supply chain). This allows for what-if-scenario analysis especially for short term planning horizons.

Another observation worth mentioning is that in some instances, the optimiser made decisions that were, by a human operator, hard to justify. These decisions dealt with production trade-offs that were done early in the planning

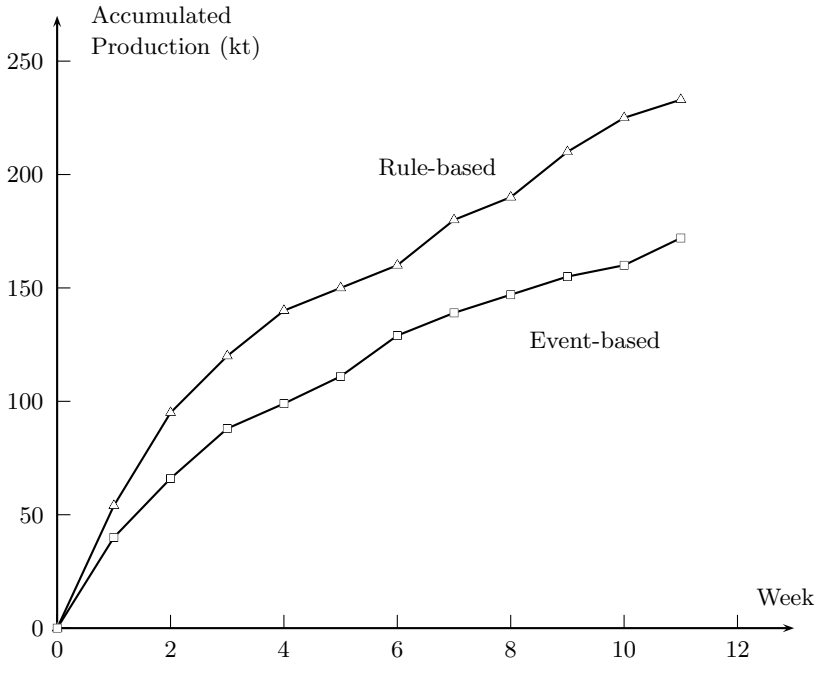


Fig. 9. Total product yield of event-based and rule-based approach (in kilotonnes)

period in order to benefit from an event that happened later with the aim to increase the overall production. This may be a valid decision with respect to the evaluation function (higher production means fitter individual), but, since we deal with a real-world environment, one has to consider the uncertainties that the future may bear (especially for long-term plans). The future benefit the optimiser was speculating for may in reality never materialise which would result in an overall inferior plan (compared to one that would not have made the trade-off decision in the first place). As a consequence, we introduced a staged optimisation that partitions the planning horizon in periods which are optimised in isolation. Once a period has been optimised, the resulting stock is carried over into the next period serving as opening stock.

We ran a trial to determine the impact of partitioning the optimisation. To obtain a normalised result we limited each period's runtime according to its share on the overall planning horizon. Optimising the whole period at once took about 8 minutes. For the test case with two periods, a maximum optimisation time of 4 minutes was allocated for each of the periods. Essentially, this means we allocated the processing time evenly, as opposed to allowing the EA to exhaust the maximum of 2000 generations for each period (in which case the result would be skewed as the search space is only

half the size, but the same amount of computational resources are applied to optimise). The simulation was run four times over the whole period and the planning horizon split in 2, 4 and 8 periods. Figure 10 confirms our initial assumption. The total yield drops by about 25% when comparing the optimisation over the entire period to the 8-segments-run. The segmented solutions are of lower quality in terms of the individual's fitness, but when audited by human planners they are much more viable as they exploit short term opportunities while ignoring higher, yet more unlikely, long-term gains.

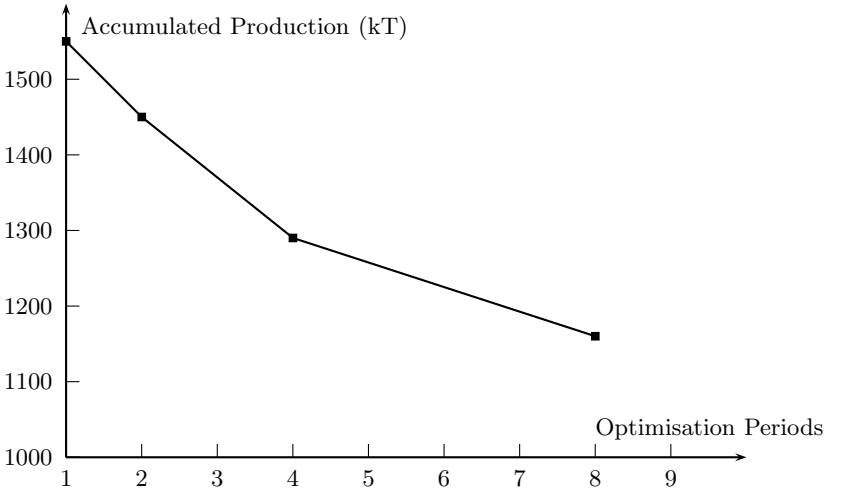


Fig. 10. Decline of total yield when optimising periods in isolation

5 Related Work

The optimisation of supply chain networks has been an ongoing research topic for many years. The research work can be categorised into two major streams. Either a supply chain has to be built from scratch with optimal location of production facilities, storage and distribution centres, or an existing supply chain has to be managed in order to make decisions that pertain sourcing of raw materials, amount of production, etc. A combination of both is also possible in which case an existing supply chain is supposed to be structurally altered (e.g., by adding new sites or negotiating new supply contracts). Structural or management decisions can also be considered with regards to their planning horizon. The former decisions are of strategic and the latter ones of tactical and operational nature. This section identifies and discusses a collection of related work undertaken in the area of optimising supply chains focussing on both streams: Operational/tactical as well as tactical planning.

An approach that addresses long and short term planning questions is presented in [8]. In this paper, the authors describe a system that uses a hybrid technique of mixed integer programming, a genetic algorithm, and discrete event simulation to make optimal decisions of where to produce (internally or externally), production planning, transportation as well as strategic decisions on location and capacity of facilities. While the genetic algorithm is employed to optimise sourcing policies and qualitative variables, mixed integer programming reduces computational costs by solving quantitative variables. The effectiveness of the obtained supply chain configuration is evaluated by a simulation.

In [9], an discrete-event simulation facilitates the evaluation of supply chain scenarios of a food supplier. The results obtained from the simulations suggest ways to improve the supply chain by changing inventory strategies. As a result, the stock level could be reduced which was beneficial to the freshness of the products, and additional products could be introduced as shelf space was freed.

Other researchers concentrate only on isolated parts of the supply chain. [10] describes a hybrid simulation-optimisation approach with the objective to select the best supplier of a supplier portfolio (in a strategic way rather than for daily sourcing: the supplier found is used throughout the planning horizon). A genetic algorithm is employed to search for possible configurations of suppliers. As with the previous approach, a discrete-event simulation determines the key performance indicators (KPIs) that form the input for the evaluation function.

Another approach concentrating on parts of the supply chain in isolation is discussed by Xie and Petrovic in [11]. Their approach defines a new decision-making system for stock allocation that is based on fuzzy IF-THEN rules. Initially, the rule base is generated by domain experts, but they allow for alteration by changing the rule's weights. A simulation on a 2-echelon supply chain (1 warehouse, and multiple retailers) is run to demonstrate the effectiveness of this approach.

Fuzzy set theory is also applied in [12]. Unlike [11], Wang and Shu use fuzzy logic to model uncertainty such as demand, processing time and delivery of supplies. The objective of their work is to develop a supply chain model that minimises the inventory costs by meeting the demands. A genetic algorithm tries to find the optimal order-up-to levels for all stock-keeping units. Again, this approach only looks at one objective (minimisation of inventory costs) and one method to achieve this objective (reduction of inventory).

The common denominator of the above papers is the application of a simulation (mostly DES) in order to obtain properties of the supply chain and evaluate the performance of the optimisation method. This observation is also backed by several surveys such as [13, 14, 15]. Methods used to obtain inputs for the simulation are mainly genetic algorithms (GA). However, even though GAs seem to dominate, recent publications indicate an advent of fuzzy logic systems as drivers for the simulation process.

Another aspect worth considering when building a solution that is meant to be reusable is the structure of the building blocks of the supply chain, that is, the model. Many attempts have been made to develop a unified supply chain model and terminology. Some of them like [16] represent common terminology or [17], which emphasises configurability and proposes event-discrete simulation as means to analyse supply chains, but this work has its emphasis on business processes rather than the definition of reusable component as they are desirable to create a programming model. Others like [18, 19, 20] use special modelling languages to express the complexity of business processes and automatically generate simulation models. Although all of them suggest employing simulations to analyse supply chain networks, they require expert knowledge of modelling languages like Rockwell Software's ARENA, one of the prevalent languages for modelling supply chain networks. The resulting models may be generated in a programming language that is incompatible to the rest of the system. We believe that a generic supply chain model can be developed that supports both, flexibility and ease of use when creating the model without the necessity to acquire expert knowledge on simulation languages. The ideas presented in this chapter are implemented in a framework which is employed to model the supply chain operations of a real-world business.

6 Conclusion and Future Work

In this chapter, we looked at the application of an EA to the problem of optimising the key decision points in the supply chain network of a major agricultural chemicals company. Modelling the highly complex nature of that company's operations was the first part of the challenge of successfully accomplishing this endeavour. A balanced mix of discrete and continuous event simulation had to be used. Furthermore, the model was designed in such a way as to be amenable to use within the framework of an EA.

Of key importance was developing the ability to represent the decision points in the supply chain network simulation as entities that could be manipulated by evolutionary operators. This was achieved in different ways. For the event-based approach, firstly by allowing the timing of decision events to be determined by values within the candidate individual representation, and also the types of those decision events. Thus, for example, a candidate individual could specify that the operation of "change from product A to product B" could be scheduled to happen in a particular part of the network on a given date and time. Secondly, the processing logic of "switching" nodes of the network could be manipulated by evolutionary operators. For example, the production of a particular chemical could involve a number of ingredients which have to be drawn from a variety of sources. In some cases, it might be easy to determine a set of rules to express the correct routing logic to use. In other situation, it may not be obvious and thus it would be preferable to let the individual represent decision making logic, as part of the encoding, and

then evaluate the performance of evolved logic as a component of the overall fitness evaluation process.

The rule-based approach abandons the idea of having isolated nodes that handle the sourcing. In contrast to the event-based system, a decision is made at each sample point based on the current state of the system. The resulting plan produces consistent and traceable decisions. Despite the fact that this approach adds the process of balancing supply and storage capacities to the search space (as opposed to running a deterministic repair function), it is able to find a solution much faster while generating even better solutions. Reasons for this observation may be that the search space of the event-based approach, that is all combinations of type and date of events, seems to be larger than the permutation of rules used in the rule-based approach. In addition, many infeasible solutions seem to be generated which demand for repair by running a costly re-evaluation.

Using rules to make decisions constricts the search space. However, only those potential solutions seem to be neglected that are less viable, as indicated by the better solutions obtained. The reason is less surprising when considering the nature of the problem. Changeovers, reduced plant production and sourcing of material are based on rules. A changeover occurs once a tank is reaching its maximum capacity, the production is reduced upon downstream bottlenecks and a sourcing decision depends on minimal procurement costs. Trying to find these decisions without understanding their natural cause is more expensive and bears many lost opportunities compared to a supply chain that is driven by a condition-decision scheme as we presented it.

The results presented in this chapter also illustrate the trade-off that is frequently accepted by business managers in practice. By running the software in a global mode over a large time-frame, an excellent result could be achieved, but the validity of such a result could be doubted by human managers who would rightly point out that the further out into the future that assumptions are made about supply chain conditions, the less reliable those assumptions would be. Hence we chose to apply the simulation/evolution algorithm over the whole time-frame in phases, starting with a short term phase of a few months, and then looking further into the future. This gave our approach the benefit of seeking higher levels of optimisation in the short term, wherein knowledge of conditions is quite firm, and then freezing those results and progressively expanding the scope of inclusion to seek out optimisation further into the future.

As constructed, the software application arising from the rule-driven simulation model and the EA presented in this chapter was able to provide invaluable insight and speculative modelling ("what-if") capabilities to managers of the client company, allowing them to find ways to optimise their supply chain network, and of course maximise production. This is the litmus test of the value of this application, and it is a rewarding application of the power of evolutionary computation to a real-world business.

In addition to our work presented in this chapter, several promising ideas may improve the results obtained. As an example, we aim to expand the linguistic terms that can be taken into account by adding future availability of plants as well as past utilisation to smoothen out the plant's overall utilisation.

A promising method to improve the evolution of the rule bases is to employ a co-evolutionary approach in which rule bases would be developed in isolation. An instance of an EA would only concentrate on its designated rule base (i.e. one EA instance could be employed per plant to evolve rules for its utilisation, one EA instance for sourcing, etc.) passing on the best of its rule bases to form the overall solution rule bases.

This chapter has presented only the current state of our endeavour to find a common model and methodology for optimising supply chain networks which caters for many business cases and industries. We expect the current method to be fine-tuned and extended to allow maximal generalisation and applicability.

References

1. Hinkelman, E.G.: Dictionary of International Trade Handbook of the Global Trade Community, 6th edn. World Trade Press (2005)
2. Law, A.: Simulation Modeling and Analysis (McGraw-Hill Series in Industrial Engineering and Management). McGraw-Hill Science/Engineering/Math. (2006)
3. Gunasekaran, A., Patel, C., Tirtiroglu, E.: Performance measures and metrics in a supply chain environment. Internation Journal of Operations & Production Management 21(1), 71–87 (2001)
4. Michalewicz, Z. (ed.): Genetic algorithms + data structures = evolution programs, 2nd edn. Springer-Verlag New York, Inc., New York (1996)
5. Hájek, P.: Mathematics of Fuzzy Logic (Trends in Logic), 1st edn. Springer, Heidelberg (1998)
6. Mamdani, E.H., Assilian, S.: An experiment in linguistic synthesis with a fuzzy logic controller. International Journal of Man-Machine Studies 7(1), 1–13 (1975)
7. Herrera, F.: Genetic fuzzy systems: taxonomy, current research trends and prospects. Evolutionary Intelligence 1(1), 27–46 (2008)
8. Truong, T.H., Azadivar, F.: Simulation optimization in manufacturing analysis: simulation based optimization for supply chain configuration design. In: WSC 2003: Proceedings of the 35th Conference on Winter Simulation, pp. 1268–1275 (2003)
9. van der Vorst, J.G.A.J., Beulens, A.J.M., van Beek, P.: Modelling and simulating multi-echelon food systems. European Journal of Operational Research 122(2), 354–366 (2000)
10. Ding, H., Benyoucef, L., Xie, X.: Simulation optimization in manufacturing analysis: a simulation-optimization approach using genetic search for supplier selection. In: WSC 2003: Proceedings of the 35th Conference on Winter Simulation, pp. 1260–1267 (2003)

11. Xie, Y., Petrovic, D.: Fuzzy-logic-based decision-making system for stock allocation in a distribution supply chain. Intelligent Systems in Accounting, Finance and Management 14(1-2), 27–42 (2006)
12. Wang, J., Shu, Y.-F.: Fuzzy decision modeling for supply chain management. Fuzzy Sets and Systems 150(1), 107–127 (2005)
13. Semini, M., Fauske, H., Strandhagen, J.O.: Applications of discrete-event simulation to support manufacturing logistics decision-making: a survey. In: WSC 2006: Proceedings of the 38th Conference on Winter Simulation, pp. 1946–1953 (2006)
14. Simulation Study Group. Simulation in the uk manufacturing industry (1991)
15. Terzi, S., Cavalieri, S.: Simulation in the supply chain context: a survey. Computers in Industry 53(1), 3–16 (2004)
16. Supply Chain Council. Supply-Chain Operations Reference-model Version 9.0 (2008), `http://www.supply-chain.org/` (January 15, 2010)
17. Rabe, M., Jaekel, F.-W., Weinaug, H.: Reference models for supply chain design and configuration. In: WSC 2006: Proceedings of the 38th conference on Winter Simulation, pp. 1143–1150 (2006)
18. Mackulak, G.T., Lawrence, F.P., Colvin, T.: Effective simulation model reuse: a case study for amhs modeling. In: WSC 1998: Proceedings of the 30th Conference on Winter Simulation, pp. 979–984. IEEE Computer Society Press, Los Alamitos (1998)
19. Ding, H., Benyoucef, L., Xie, X., Hans, C., Schumacher, J.: One a new tool for supply chain network optimization and simulation. In: WSC 2004: Proceedings of the 36th Conference on Winter Simulation, pp. 1404–1411 (2004)
20. Ganapathy, S., Narayanan, S., Srinivasan, K.: Logistics: simulation based decision support for supply chain logistics. In: WSC 2003: Proceedings of the 35th Conference on Winter Simulation, pp. 1013–1020 (2003)