Proceedings of the 2018 Winter Simulation Conference
M. Rabe, A. A. Juan, N. Mustafee, A. Skoogh, S. Jain, and B. Johansson, eds.

SPEEDING UP SIMULATION-BASED OPTIMIZATION OF SUPPLY NETWORKS BY MEANS OF A MULTI-POPULATION GENETIC ALGORITHM AND REUSE OF PARTIAL SOLUTIONS

Kai Gutenschwager
Bastian Wilhelm

Ostfalia University of Applied Sciences
Am Exer 2
Wolfenbüttel, 38302, GERMANY

Sven Völker

Ulm University of Applied Sciences
Prittwitzstraße 10
Ulm, 89075, GERMANY

ABSTRACT

Supply network design is an important strategic operations management problem. The paper discusses the use of simulation-based optimization for solving this task in a performant manner. A three-tier hierarchical approach is presented: On the first tier, a multi-population genetic algorithm decides about the active nodes of the network and assigns product groups to nodes. On the second tier, the production capacities and sourcing paths are determined. On the third tier, a simulation model calculates an objective function value that describes the quality of the network design. While the genetic algorithm acts as a central control instance, the evaluation of the individuals on the second and third tiers are parallelized in a computer network. In the course of the optimization process, partial solutions are saved and reused in order to reduce computational cost. The approach is tested on a real-world example of a European supply network.

1 INTRODUCTION

A supply network consists of a set of nodes (suppliers, manufacturing plants, warehouses, and customers) that are connected via transport relations (Simchi-Levi et al. 2009). Raw materials and product components are provided by suppliers and refined to products by manufacturing plants. Products are sent to the customers according to their specific demand via warehouses. Designing supply networks is a challenging strategic management task: Decisions about opening or closing facilities strongly effect distribution costs and have long-term consequences.

Many facets of supply network design can be grasped quantitatively. A supply network design problem (SNDP) can be formulated as a mixed integer problem in various ways, which differ in considered objectives, included constraints, and level of detail. Examples of such models are given in Ng et al. (2015) and Sampat et al. (2017). They can not only be solved analytically but also by numerical methods, especially meta-heuristics: Most often, evolutionary algorithms like genetic algorithms (GA) are applied (Altiparmak et al. 2006; Eskandarpour et al. 2017). Recent research investigates the use of ant colony optimization (Lagos et al. 2015) and reinforcement learning (Rabe and Dross 2015) to solve SNDPs.

An important aspect of supply network design is uncertainty. Uncertainty in SNDPs can be handled by probability theory and fuzzy set theory (Govindan et al. 2017). However, supply chain simulation seems to be a more powerful and flexible approach since it enables modelling of stochastic processes at an arbitrary level of detail. Hence, simulation studies are common in the context of supply network design, see e.g., Fioroni et al. (2015). The combination of simulation for evaluating supply network configurations and meta-heuristics for controlling the search for good configurations, i.e., simulation-based optimization, has been successfully used several times, e.g., by Juan and Rabe (2013) and Peirleitner et al. (2016). Based on this principle, the architecture of a decision support system for logistics networks has been proposed by Rabe et al. (2017).

Simulation-based optimization of supply networks is computationally expensive, since each simulation run requires considerable computing time and many replications are needed in an optimization run. A promising approach to reduce computing times for simulation runs is parallelization across several processors or computers. This can be done on the level of single replications as discussed in numerous papers, e.g., Mustafee et al. (2014) and Sarli et al. (2016). In the context of simulation-based optimization, however, a simpler and more efficient approach is possible – the parallel execution of independent replications. Especially GAs are easy to parallelize since many possible network configurations (individuals) have to be analyzed independently of each other in each generation.

In this paper, two techniques to accelerate simulation-based optimization for supply network design are applied: Firstly, the evaluation of several individuals of the same generation is distributed in a computer network. Secondly, we take advantage of the fact that similar configurations of supply networks induce the same logistics processes. Therefore, it is possible to store partial solutions found in one simulation run and reuse them in other simulation runs.

The remaining sections are organized as follows: Section 2 specifies the supply network design problem, Section 3 explains the architecture of the proposed solution, which is detailed in Sections 4 to 6. Empirical results for a real-world use case are given in Section 7.

2 PROBLEM STATEMENT

In this paper, the SNDP is defined as follows: Given is a set of customers with their respective locations and a set of products. Each customer has a yearly demand for one or several of the products (given on a detailed level as daily order lines for each product). Products are produced in manufacturing plants, which are equipped with resources of different types. Products that can be produced by the same type of resources are aggregated to product groups. The production rate for each resource unit and product group is known. Furthermore, there is a set of candidate locations for plants and warehouses. Cost rates are given for installation and operation of resources, for transports between plants and warehouses, for transports between plants or warehouses and customers, and for stock holding. Production lead times and transportation times are random variates with known distributions and parameters. All other parameters are assumed to be deterministic. The SNDP is to find a configuration of the supply network so that the overall costs are minimal while a required service level is maintained. In more detail, supply network design has to answer the following questions on a strategic level:

1. Which candidate locations should be chosen to open manufacturing plants?
2. Which candidate locations should be chosen to open warehouses?
3. Which product groups are produced in which plants?
4. Which manufacturing plant should be chosen for supplying a customer with a product group?
5. Which sourcing path should be chosen for the transport of a product group to a customer?
6. How many resource units of which resource type have to be installed in which plant?

The problem instance used in our empirical study is a large European production and distribution network consisting of 428 customers, 25 (candidate) warehouses, 11 plants and 4,388 products, organized in 11 product groups. Customer orders are given as real-world data on a daily basis for a simulation period of one year. The orders comprise approximately 2,305,600 order lines. Delivery dates are specified to the day. The basic structure of the supply network is given in Figure 1.

3 GENERAL APPROACH

We solve the SNDP by simulation-based optimization where a GA controls the global search, and a stochastic simulation model is used to evaluate different configurations of the supply network. The performance issue of simulation-based optimization is addressed by parallelizing the search: The simulation runs for evaluating different configurations of the supply network are distributed accross several computers since these runs

Figure 1: Layout of the supply network example.

are the computationally expensive part of the optimization process. The parallelization is achieved by a three-tier architecture with a blackboard communication layer (see Figure 2).

On the first tier, alternative configurations of the supply network are created and managed by a GA (see Section 4). These configurations (or individuals) are relatively simple and only define which candidates for manufacturing plants and warehouses are used, and to which plants the product groups are allocated. The transport relations are not part of the high-level configuration of the supply network. In order to transfer the individuals to the simulators, their chromosomes are written to a relational database that acts as central communication layer. Each chromosome corresponds to a simulation task.

In order to utilize all available simulator instances, enough open simulation tasks have to be available at all times during an optimization run. This cannot be ensured by a GA with a single population, because the GA has to wait for the evaluation of all offspring individuals of one generation before the individuals of the next generation can be created. Depending on the relation between the number of offspring individuals and the number of available simulator instances, the percentage of unused calculation capacity can grow to nearly 50% in the worst case. For this reason, a multi-population approach is implemented. This ensures that a sufficient number of simulation tasks is permanently available during the optimization run.

The solution architecture supports an arbitrary number of simulators. Each simulator realizes the second and third tier of the architecture and operates on its own computer in a local area network. Simulators with no assigned task observe the blackboard in the communication layer. Once a simulator detects a new individual provided by the GA, it registers for the task of evaluating this individual. Afterwards, the simulator reads the chromosome and prepares the configuration for simulation. This preparation is performed on the second tier of the architecture and includes two tasks: Calculating the number of required machines for each manufacturing plant, and finding the most cost-efficient sourcing paths for transporting products from the manufacturing plants to to the customers (see Section 5).

Once this is done, a simulation run is performed to estimate the objective function value. In order to do this in an efficient manner, partial solutions are stored in the central database. Reusing these solutions enables exchange of information between different simulators and helps to avoid repeated calculations. Details of this mechanism are explained in Section 6.

After the objective function value has been determined, the simulator writes it to the database that is observed by the GA on the first tier. The GA reads the objective function value and uses it subsequently in the optimization process (see Section 4).

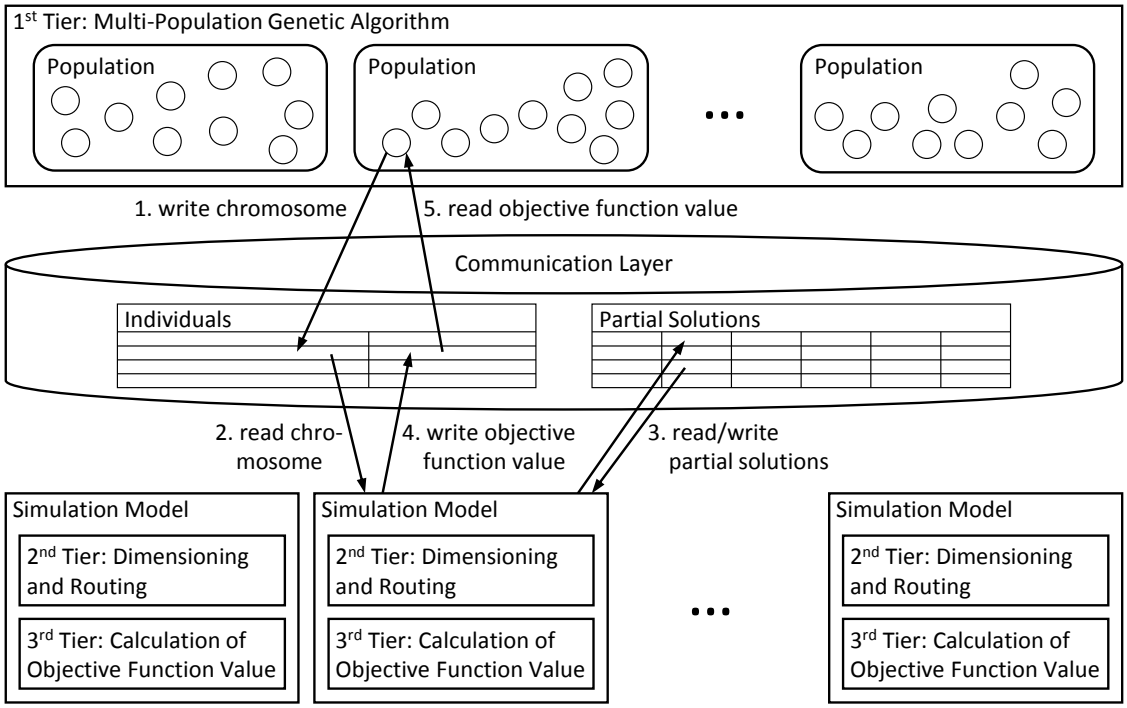

Figure 2: Solution architecture.

4 FIRST TIER: MULTI-POPULATION GENETIC ALGORITHM

4.1 Representation

An appropriate representation of individuals is crucial for the efficiency of any GA. Different representations of supply networks have been proposed: edge-based encoding (Michalewicz et al. 1991), vertex-based encoding (Altiparmak et al. 2009) and edge-and-vertex encoding (Gen and Cheng 2000). These representations are able to encode not only the information about active plants and warehouses, but also about the active transport relations, i.e. the edges of the supply network.

The proposed hierarchical approach defers the definition of transport relations from the top-level GA to the second tier of the architecture. Hence, only the information which plants and warehouses are open and the allocation of product groups to plants have to be encoded in a chromosome. This leads to a simple representation where chromosomes are defined as a triple $(\mathbf{m}, \mathbf{w}, \mathbf{A})$. Element m_i of vector $\mathbf{m}$ indicates, whether manufacturing plant i is open ($m_i = 1$) or closed ($m_i = 0$). Similarly, element w_j of vector $\mathbf{w}$ specifies, whether warehouse j is open ($w_j = 1$) or closed ($w_j = 0$). Finally, element a_{pi} of matrix $\mathbf{A}$ indicates, whether plant i produces product group p ($a_{pi} = 1$) or not ($a_{pi} = 0$).

4.2 Crossover, Mutation, and Selection

Crossover operators play an important role in GAs: A crossover operator combines properties of two parent individuals to create a new individual. In this paper, two crossover operators are considered. *Simple Crossover* performs a one-point crossover (Bäck et al. 2000) separately on plant vector $\mathbf{m}$ and warehouse vector $\mathbf{w}$. The crossover on matrix $\mathbf{A}$ is linked to the treatment of plant vector $\mathbf{m}$: Column vector $a_{\cdot i}$ of the offspring is taken from the same parent that determines the value of plant vector element m_i.

Although easy to realize, simple crossover has an obvious disadvantage: It depends on the original sorting of the vectors $\mathbf{m}$ and $\mathbf{w}$. For instance, these vectors may be given as alphabetically sorted lists of city names. As a consequence, adjacent elements of $\mathbf{m}$ and $\mathbf{w}$ do not necessarily represent neighboring

cities. Therefore, it cannot be assumed that good partial solutions for supplying customers in a certain geographic region are described in compact form and are preserved by simple crossover. According to the building-block hypothesis (Forrest and Mitchell 1993), simple crossover should be an inefficient operator.

Regional Crossover uses geographical information and provides a more promising approach. This operator divides the map of all locations by a randomly placed intersecting line, thus creating two half-planes. Information about open plants and warehouses as well as the allocation of product groups to plants is taken from one parent for the locations in one half-plane and from the other parent for the locations in the other half-plane.

Both simple and regional crossover may lead to invalid solutions. A solution is valid only if each single product group is allocated to at least one manufacturing plant. Therefore, validity of newly created individuals is checked immediately after performing the crossover. Product groups that are not allocated to a plant are assigned randomly to one of the open plants.

Afterwards, offspring individuals are mutated. Two different mutation operators are used: *Flip Location* changes the state of a random warehouse from "open" to "closed" or vice versa. *Reallocate Product Group* randomly reassigns a product group from one manufacturing plant to another open plant. Both operators preserve validity.

The chromosome of the offspring individual is written to the database that serves as a central communication platform (see Section 3). The transport relations are determined decentrally on the second tier of the software system, and the objective function value is calculated by a simulation system on the third tier. Once the entire offspring of a generation has been created and evaluated, individuals for the next generation have to be selected. The implemented GA uses elitism, i.e., a fixed number of best individuals of the parent generation is included into the offspring generation. The remaining individuals of the next generation are selected from the offspring using roulette-wheel selection (Lipowski and Lipowska 2012). This selection is based on the reciprocal of the objective function value since the objective function is to be minimized and cannot be used as fitness value directly.

5 SECOND TIER: DIMENSIONING AND ROUTING

Once an individual is read from the database, the basic structure of the supply network is set up with respect to the sourcing of all product groups for all customers in the network. As this model does not include actual transports, a correction factor f_{ij} is introduced to model different cost rates for transports between locations i and j. In our case study, f_{ij} is set to 1 if location j is a final customer, and to an initial value of 0.5 otherwise. These values indicate that the transport of a loading unit from a plant to a warehouse costs on average only half as much as the last mile transport from a warehouse (or plant) to the final customer (e.g., due to the usage of bigger or better utilized trucks).

We implemented an enumeration procedure to identify the best sourcing path for each combination of product group and customer with respect to the estimated transport costs: Starting with all plants that are able to produce the product group (defined in tier 1), the direct connections to the customer are evaluated. In a next step, using the (open) warehouses from each possible plant to the customer are evaluated. Note that also open plants that are not assigned to a specific product group can serve as a warehouse for the respective product group. The best overall sourcing path is selected for each product group of each customer location. With S_{jp} being the set of successors of location j for product group p, the costs for each transport relation between locations i and j for product group p are computed as:

$$c_{ijp} = \left(\sum_{k \in S_{jp}} demand_{kp} \right) \times distance_{ij} \times f_{ij}$$

Note that $demand_{kp}$ of product group p for customer k is an aggregation of (real-world) customer order lines, which are also used as input data of the underlying simulation model. The distance between two

locations is computed as the orthodromic distance (with latitude and longitude as input) times a detour factor of 1.28 for modeling street distances.

An example is given in Figure 3, where the sourcing path for a product group for customers 1, 2, and 3 is given with plant P1 and warehouse W1, whereas customers 4 and 5 are directly assigned to P1 and customers 6 and 7 are sourced from P1 via warehouse W2. The demand of W1 – defined as the sum of the demands of all assigned customers (given next to the respective node) – is 500 loading units. The demand of W2 is 100 loading units, respectively.

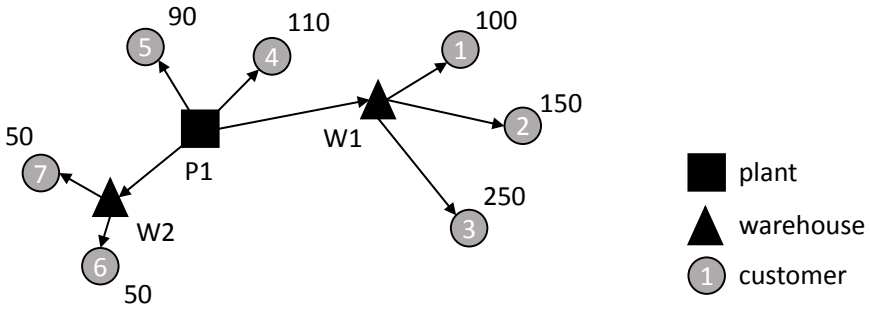

Figure 3: Example for sourcing paths with customer demands [number of loading units per year].

Once a sourcing path is defined for each combination of a product group and a customer, the number of loading units of each product group per year can be directly derived for each plant. Combining this information with the given production rates per product group, the number of required resource units per product group and plant is computed, such that an average utilization of about 80% is achieved. In the example of Figure 3, W1 has to handle 500 loading units, W2 needs to handle 100 loading units, and P1 needs to produce 800 loading units of the given product group per year in total.

All assignments of products to plants and warehouses are now given. These allocations and the sourcing paths are input for the simulation model of tier 3.

6 THIRD TIER: CALCULATION OF INVENTORY LEVELS AND FITNESS FUNCTION

6.1 Basic Simulation Model

The underlying simulation model is realized on the basis of SimChain. SimChain is a discrete event simulation tool, which has originally been developed as a class library for the simulation system Plant Simulation (Gutenschwager and Alicke 2004). All relevant data are stored in a MySQL database, which is used to automatically generate the simulation model utilizing a building block library (implemented within Plant Simulation) and to store all result data. Due to its flexible and data-driven structure, SimChain is suitable for the task at hand.

For our approach, the demand of all final customers needs to be deterministic, i.e., orders need to be created at the same customer location at the same date with the same amount in different simulation runs. SimChain is able to import fixed order data. However, in order to speed up the model generation phase, we do not import the customer order data anew for each instance of the model we solve, but instead define all customer locations along with the customer orders in a basic model file. For each configuration of the network, we first adopt the basic model only with respect to open plants and warehouses. Secondly, we generate the resources of the plants within the given model objects of the plants according to the assignment of product groups to plants anew.

In the last step of model generation, the transport relations are created according to the sourcing paths determined in tier 2. Four different means of transport are defined along with the maximum capacity in terms of the number of loading units per vehicle and the cost rate per kilometer. Within the simulation model, the number of available vehicles is not limited.

The fitness of each individual (network configuration) is computed as the sum of fixed costs for opening plants and warehouses, fixed costs for resources (depending on the number of resource units in each plant), already computed in tier 2, as well as the costs for transport and the capital binding costs, which are computed using the simulation model. Further handling and variable production costs are not considered.

6.2 Concept of Partial Solutions

In this section, we present an approach that stores partial solutions and makes use of them if the respective partial model is part of another network again. Partial solutions are defined for subtrees of the network, each for a single product group as given in Figure 3: Starting from a plant for a product group, a subtree includes all warehouses and customers that are supplied by the respective plant directly or indirectly. For such a subtree, average resource utilization and average stock levels of all products of the respective product group are independent from the remaining network. Furthermore, related transports occur at the same point of time with the same transport amounts, if no (different) backlogs of transport orders occur when simulating else-wise different networks. This is a main assumption of our approach: There are no capacity restrictions for the transport relations, i.e., if products are on stock, they are transported at the next available date of the respective transport schedule.

If a subtree occurs in another network configuration again, we do not need to simulate the production and order processes of the respective locations of the subtree again. Instead, we can reuse the result for average stock levels and individual service levels for all relevant products from previous simulation runs. However, the transport planning process and transport process between the locations need to be simulated again. Here, the only interdependency with the processes for the other product groups occur. An example is given in Figure 4: The subtree for product group PG1 is indicated by solid lines in both network layouts. A second product group, PG2 (indicated by dashed lines), is originally produced at plant P2 and is delivered to customers 1 and 3 via warehouse W1. The transports between W1 and customer 1 are carried out with the same trucks for both product groups. In a second configuration of the supply network (Figure 4b), P1 produces PG2 as well as PG1, and plant P2 is not open. For this network structure, we combine the transports for PG1 and PG2 for all transports between P1 and W1 as well as W1 and customer 1 and customer 3 respectively.

In order to simulate the transport process correctly, we simply need to include all transport amounts as they occurred in the original simulation run for the subtree of PG1. This means, one part of the transports starting in P1 is the (dynamically computed) result of the regular order and production planning process (for PG2) and the other part is defined as a static list of given transports with the day (period) of the transport, the number of loading units and the number of loading units sent late, which are needed to calculate the service levels correctly. The respective partial solutions are stored in a single table, where each entry represents a transport relation (as observed for a specific replication number) for the individual under consideration, a given start and destination location, the product group, the average stock of the product

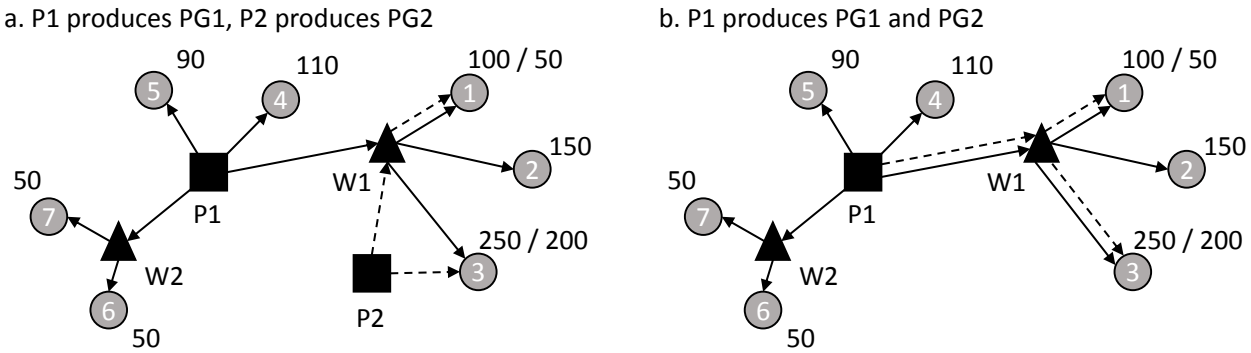

Figure 4: Example for sourcing paths with customer demands [number of loading units per year].

group (in the start location), and all transport quantities of the product group between the two locations. The transport quantities are stored as a string representing a list of tuples (period of transport, number of loading units, and number of loading units delivered late). In order to indicate to which subtree a partial solution belongs, the following notation is used: Starting with the product group and the number of the replication, the entire sourcing path is part of the ID of the partial solution along with the total amount of loading units for each location on the sourcing path. For the example given in Figure 3, ten partial solutions are stored for product group PG1 (Table 1). The first record specifies the transports between W1 and C1 as part of a sourcing path where P1 produces 800 loading units and W1 handles 500 loading units. The first transport between W1 and C1 takes place on day 1 with a transport volume of 3 loading units, none of them delayed.

Table 1: Example for storing partial solutions of a subtree.

id_partial_solution	**replication number**	**product group**	**start**	**dest**	**avg. stock (start)**	**transports**
PG1:1:P1/800/W1/500/C1	1	PG1	W1	C1	105.3	(1:3,0)(2:1,0), ...
PG1:1:P1/800/W1/500/C2	1	PG1	W1	C2	105.3	(1:8,0)(4:4,1) ...
PG1:1:P1/800/W1/500/C3	1	PG1	W1	C3	105.3	(3:2,0)(11:1,0) ...
PG1:1:P1/800/W1/500	1	PG1	P1	W1	401.8	(2:15,0)(8:20,0) ...
PG1:1:P1/800/C4	1	PG1	P1	C4	401.8	(4:2,0)(10:1,0) ...
PG1:1:P1/800/C5	1	PG1	P1	C5	401.8	(2:2,0)(12:1,0) ...
PG1:1:P1/W2/100/C6	1	PG1	W2	C6	95.2	(1:4,0)(2:4,0) ...
PG1:1:P1/W2/100/C7	1	PG1	W2	C7	95.2	(1:10,0)(12:7,0) ...
PG1:1:P1/W2/100	1	PG1	P1	W2	401.8	(5:20,0)(12:24,0) ...

Setting up a generic model based on the results of tier 2 of the overall solution approach now includes a further step: We need to identify all subtrees of the new network configuration and check for each transport relation whether the respective partial solution is already known from previous simulation runs. If so, the respective transports and all other relevant result data (as the average stock) are imported for each transport relation and location. As a consequence, the production planning and sourcing of products for the product groups of all locations of the subtree are not part of the model. After finishing a simulation run, the IDs of all new partial solutions are computed using a back-propagation algorithm starting with the customers, and all relevant data are stored in the database if not already given.

7 EMPIRICAL RESULTS

The solution approach has been tested on the European production and distribution network introduced in Section 2. The simulation period is one year, and the planning of production orders and transports are performed on a daily basis. The computational time for the operations of the GA (tier 1) are within a few milliseconds per iteration and can be neglected. Also, the time for writing individuals into the database and retrieving the fitness values from the database can be neglected, i.e., there is no relevant overhead for using a communication layer between the GA and the evaluation of solutions. As a consequence, using more than one computer on tier 2 and 3 leads to a linear reduction of the total time needed, which has the highest effect to reduce the time to complete GA runs. The computational time for the algorithm on tier 2 (sourcing of products and dimensioning of resources) is about 17.2 seconds on average. For further evaluating the behavior of our approach, we set up two experiments. The first one was to show by what factor the computational time can be reduced using partial solutions. In the second experiment, we analyzed the influence of the two different crossover operators presented in Section 4.

7.1 Experiment 1: Influence of Using Partial Solutions

As first experiment, we have carried out a GA run for 50 generations with a population of 10 individuals. We have restricted the evaluation to one simulation run per individual, because we are only interested in the portion of reused partial solutions within a GA run. As we can only use partial solutions for runs with the same replication number (each corresponding to the identical seed values for the respective random variates), carrying out more simulation runs per individual does not lead to different results concerning the portion of reused partial solutions. In each iteration, 20 new individuals are created as offspring. Both crossover operators are applied with the same probability. The GA run starts with an empty database for partial solutions. On average, a solution consists of about 4,700 partial solutions. The results are given in Figure 5.

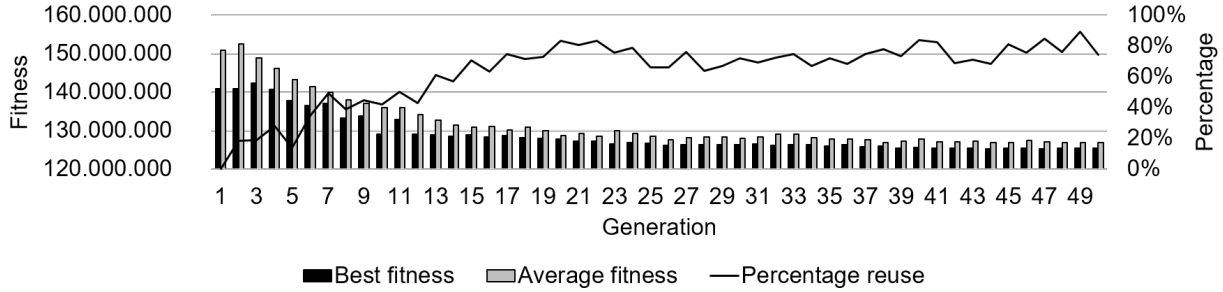

Figure 5: Development of minimal and average fitness, and percentage of reuse of partial solutions.

The reuse of partial solutions increases rather fast. After 12 generations, the percentage of reused partial solutions is already above 50%. After 45 generations, the average percentage of reused partial solutions is higher than 80%. After the GA run has finished, more than 700,000 partial solution data sets (each one representing a transport relation between two locations) have been stored in the database.

The time needed for reading and writing data to setup the model and to store new partial solutions is nearly constant with an average of 7.4 seconds per individual. In the beginning of the GA run, writing partial solutions is more expensive, whereas towards the end of a GA run, the computational time for reading (and initializing) partial solutions is dominating (Figure 6).

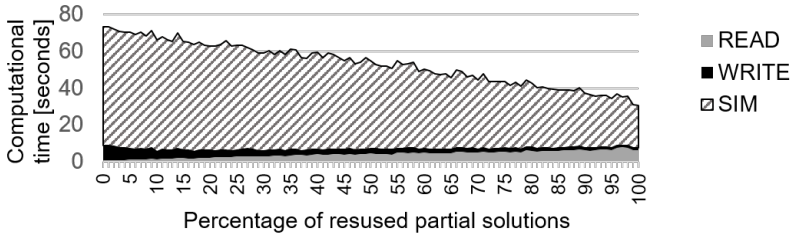

Figure 6: Computational times.

If no partial solutions can be used, the time for performing a single simulation run is about 64.5 seconds. If, on the other hand, the entire model can be generated using partial solutions (and only the transports are dynamically planned within the simulation run), the computational time is reduced to an average of 23.1 seconds (see Figure 6). In the example given, the average percentage of reused partial solutions is 64%, with an average computation time for the simulation runs of 37.6 seconds, i.e., a reduction of 26.9 seconds per simulation run.

As a consequence, using partial solutions leads to an overhead of about 7.4 seconds per simulation run. For the nearly 1,000 individuals analyzed in the course of the GA run, the total computation time for evaluating all individuals with only one simulation run per individual (including the database access times

when using partial solutions) has been reduced from 17.6 hours to 12 hours (32,2%) for a GA run. Note that the total time needed can be reduced linearly by using more than one simulator instance.

If further runs of the GA are executed on the basis of stored partial solutions, the average number of reused solutions increases only slightly in subsequent GA runs. We have executed three further GA runs, and the percentage of reused partial solutions and the overall runtime remains nearly the same, even though the number of stored partial solutions increases from 700,000 to over 3,000,000 solutions. Figure 7 gives the respective results. When starting a new GA run, usually hardly any partial solutions from previous GA runs can be reused (due to the random selection of the first generation of individuals). The increasing usage of partial solutions within a GA run goes along with the schema theorem of Holland (1975). A schema is defined as a template, consisting of specific alleles for a set of genes and "wildcards" for all other genes. Chromosomes belong to a schema if they fit into the template having identical alleles for the genes specified explicitly in the template. Schemas correspond to our partial solutions, and the occurrence of the same good parts of solutions increases during the evolution of a population, and these can be reused more often as the population converges.

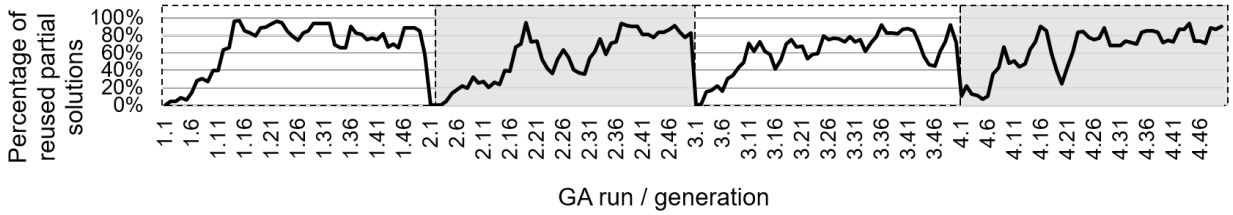

Figure 7: Development of percentage of reused partial solutions.

At the first glance, the results imply that it is sufficient to store the partial solutions for each GA run only locally within the simulation model without additional operations to store partial solutions in the database. The respective 7.4 seconds for database access and setting up the "fixed" transport orders defined in the respective partial solutions could then be reduced significantly. However, it is more efficient to stick to a shared database of partial solutions if more than one simulator instance work on the same GA run.

7.2 Experiment 2: Influence of Different Crossover Operators

In the second experiment, the influence of the two presented crossover operators has been analyzed. Starting with the same initial population defined by the random seed for GA runs (on tier 1), four pairs of GA runs have been conducted. In one run, the simple crossover operator is used, and in the corresponding GA run the regional crossover is used. The results for the best solution found (best fitness) are given in Figure 8. In two runs, the simple crossover operator has been superior. On average, the two crossover operators lead to nearly the same solution quality, with a difference of less than 0.2%. Hence, the expexted superiority of regional crossover (see Section 4.2) could not be empirically proved.

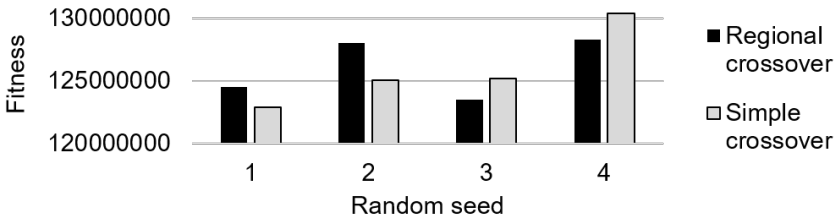

Figure 8: Comparison of simple crossover and regional crossover.

8 CONCLUSION AND OUTLOOK

We have presented an approach to solve the SNDP by simulation-based optimization. The approach uses a multi-population genetic algorithm, where a number of computers work in parallel and share information about the solutions found. Our results show that a parallelization of the solution process leads to an almost linear reduction of the total time needed. The concept of storing and reusing partial solutions is very promising as well and leads to a reduction of the computational time of approximately 32%, even if each GA run starts with an empty list of partial solutions.

Since our work is focused on the reduction of computational cost instead of the improvement of solution quality, there is room for the latter. This can be achieved by further analyzing and improving crossover and mutation operators, and optimizing the parameter settings of the GA. Also, the algorithm on tier 2 could be further optimized. In this context, a deterministic estimation of the solution quality of individuals could be carried out in order to exclude complete individuals from further cost-intensive analysis.

ACKNOWLEDGEMENT

This work is partially based on the "Entwicklung eines Supply Chain Simulationswerkzeugs in der Cloud unter Berücksichtigung dynamischer Verschlüsselungstechnologien" (SimChain MONSTER) project, which is funded by the Federal Ministry for Economic Affairs and Energy, Germany.

REFERENCES

Altiparmak, F., M. Gen, L. Lin, and T. Paksoy. 2006. "A Genetic Algorithm for Multi-objective Optimization of Supply Chain Networks". *Computers & Industrial Engineering* 51(1):196–215.

Altiparmak, F., M. Gen, L. Lin, and I. Karaoglan. 2009. "A Steady-state Genetic Algorithm for Multi-product Supply Chain Network Design". *Computers & Industrial Engineering* 56(2):521–537.

Bäck, T., D. B. Fogel, and Z. Michalewicz. 2000. *Evolutionary Computation 1: Basic Algorithms and Operators*. Bristol: Institute of Physics Publishing.

Eskandarpour, M., P. Dejax, and O. Péton. 2017. "A Large Neighborhood Search Heuristic for Supply Chain Network Design". *Computers and Operations Research* 80:23–37.

Fioroni, M. M., L. A. G. Franzese, I. R. de Santana, P. E. P. Lelis, C. B. da Silva, G. D. Telles, J. A. S. Quintáns, F. K. Maeda, and R. Varani. 2015. "From Farm to Port: Simulation of the Grain Logistics in Brazil". In *Proceedings of the 2015 Winter Simulation Conference*, edited by L. Yilmaz et al., 1936–1947. Piscataway, New Jersey: IEEE.

Forrest, S., and M. Mitchell. 1993. "Relative Building-Block Fitness and the Building-Block Hypothesis". In *Foundations of Genetic Algorithms*, edited by L. D. Whitley, Volume 2, 109–126. San Mateo: Morgan Kaufmann Publishers.

Gen, M., and R. Cheng. 2000. *Genetic Algorithms and Engineering Optimization*. New York: Wiley.

Govindan, K., M. Fattahi, and E. Keyvanshokooh. 2017. "Supply Chain Network Design under Uncertainty: A Comprehensive Review and Future Research Directions". *European Journal of Operational Research* 263(1):108–141.

Gutenschwager, K., and K. Alicke. 2004. "Supply Chain Simulation mit ICON-SimChain". In *Logistik Management*, edited by T. Spengler et al., 161–178. Heidelberg: Physica.

Holland, J. H. 1975. *Adaption in Natural and Artificial Systems*. Ann Arbor: University of Michigan Press.

Juan, A. A., and M. Rabe. 2013. "Combining Simulation with Heuristics to Solve Stochastic Routing and Scheduling Problems". In *Simulation in Produktion und Logistik 2013*, edited by W. Dangelmaier et al., 641–650. Paderborn: HNI Verlag.

Lagos, C., F. Paredes, S. Niklander, and E. Cabrera. 2015. "Solving a Distribution Network Design Problem by Combining Ant Colony Systems and Lagrangian Relaxation". *Studies in Informatics and Control* 24(3):251–260.

Lipowski, A., and D. Lipowska. 2012. "Roulette-wheel Selection via Stochastic Acceptance". *Physica A: Statistical Mechanics and its Applications* 391(6):2193–2196.

Michalewicz, Z., G. A. Vignaux, and M. Hobbs. 1991. "A Nonstandard Genetic Algorithm for the Nonlinear Transportation Problem". *ORSA Journal on Computing* 3(4):307–316.

Mustafee, N., K. Katsaliaki, and S. J. E. Taylor. 2014. "A Review of Literature in Distributed Supply Chain Simulation". In *Proceedings of the 2014 Winter Simulation Conference*, edited by A. Tolk et al., 2872–2883. Piscataway, New Jersey: IEEE.

Ng, W. P. Q., M. A. Promentilla, and H. L. Lam. 2015. "An Algebraic Approach for Supply Network Synthesis". *Journal of Cleaner Production* 88:326–335.

Peirleitner, A. J., K. Altendorfer, and T. Felberbauer. 2016. "A Simulation Approach for Multi-stage Supply Chain Optimization to Analyze Real World Transportation Effects". In *Proceedings of the 2016 Winter Simulation Conference*, edited by T. M. K. Roeder et al., 2272–2283. Piscataway, New Jersey: IEEE.

Rabe, M., and F. Dross. 2015. "A Reinforcement Learning Approach for a Decision Support System for Logistics Networks". In *Proceedings of the 2015 Winter Simulation Conference*, edited by L. Yilmaz et al., 2020–2032. Piscataway, New Jersey: IEEE.

Rabe, M., F. Dross, D. Schmitt, M. Ammouriova, and C. Ipsen. 2017. "Decision Support for Logistics Networks in Materials Trading Using a Simheuristic Framework and User-generated Action Types". In *Simulation in Produktion und Logistik 2017*, edited by S. Wenzel and T. Peter, 109–118. Kassel: kassel university press.

Sampat, A. M., E. Martin, M. Martin, and V. M. Zavalaa. 2017. "Optimization Formulations for Multi-product Supply Chain Networks". *Computers and Chemical Engineering* 104:296–310.

Sarli, J. L., H. P. Leone, and M. D. los Milagros Gutiérrez. 2016. "Ontology-Based Semantic Model of Supply Chains for Modeling and Simulation in Distributed Environment". In *Proceedings of the 2016 Winter Simulation Conference*, edited by T. M. K. Roeder et al., 1182–1193. Piscataway, New Jersey: IEEE.

Simchi-Levi, D., P. Kaminsky, and E. Simchi-Levi. 2009. *Designing and Managing the Supply Chain: Concepts, Strategies and Case Studies*. 3rd ed. Boston: Mcgraw-Hill Education.

AUTHOR BIOGRAPHIES

KAI GUTENSCHWAGER is head of the Institute of Information Engineering at the Ostfalia University of Applied Sciences, Wolfenbüttel. He studied Business Informatics and received his doctoral degree from the Technical University of Braunschweig. He worked for SimPlan as a head of the office being a simulation expert in the field of logistics and supply chain management. From 2009 he was a professor for information systems in logistics at Ulm University of Applied Sciences and since 2013 he is a professor at Ostfalia University of Applied Sciences. His research is focused on IT-based methods for the simulation and optimization of production and logistics systems. His e-mail address is k.gutenschwager@ostfalia.de.

SVEN VÖLKER studied Business Informatics and received his doctoral degree from Ilmenau Technical University. He worked for Tecnomatix, UGS, and Siemens as a solution architect and project manager in the field of Digital Manufacturing. Since 2010, he is Professor for Logistics Planning and Digital Manufacturing at Ulm University of Applied Sciences. His research is focused on IT-based methods for planning, simulation, and optimization of production and logistics systems. His e-mail address is voelker@hs-ulm.de.

BASTIAN WILHELM (M.Sc.) is research assistant at the Ostfalia University of Applied Sciences. He received his bachelor as well as his master degree at Ostfalia. His research focuses on finding solutions for an encryption and right management model as well as optimization models in the context of multi-organizational simulation applications. His e-mail address is ba.schulten@ostfalia.de.