Application of the Fish School Search for the Supply Chain Network Planning Problem

Bernd Hellingrath, Dennis Horstkemper, Diego de S. Braga, Luis Filipe de A. Pessoa

Westfälische Wilhelms-Universität Münster
Leonardo-Campus 3, D-48149 Münster, Germany
{bernd.hellingrath, dennis.horstkemper,diego.siqueira, filipe.pessoa}@wi.uni-muenster.de

Fernando Buarque de Lima Neto, Marcelo Gomes Pereira de Lacerda

University of Pernambuco (UPE)
Rua Benfica, 455, 50750-410 – Recife, Brasil
{fbln, mgpl}@ecomp.poli.br

Abstract— **The Fish School Search (FSS) is a recently proposed meta-heuristic, which is inspired by the collective behavior of fish schools during their search for food. This optimization technique has been showing promising results when applied to benchmarking problems. Moreover, the FSS has been especially designed to tackle complex problems, especially ones with large search spaces like they are found in the area of Supply Chain Management. As such, the Supply Chain Network Planning (SCNP) problem was chosen as an NP-hard optimization problem to evaluate the applicability of the FSS in this problem domain. Additionally, other state-of-the-art meta-heuristics used for optimization problems (Particle Swarm Optimization and Differential Evolution), as well as a state-of-the art mathematical optimization technique were applied to solve the same planning problem in order to determine the relative performance of the FSS algorithm for this class of problems.**

Keywords—Fish School Search, Meta-heuristics, Computational Intelligence, Supply Chain Management, Supply Chain Network Planning, Bio-inspired Methods

I. MOTIVATION

In the research domain of meta-heuristics new variants and algorithms are proposed and published at a consistently high rate. This is motivated by the need for improved solution techniques for NP-hard planning problems, to which several real world applications belong to. Conceived by Bastos and Lima Neto in 2007, the Fish School Search (FSS) is such a novel meta-heuristic, first published in 2008 [1]. It has been developed as a general optimization technique and incorporates a new way to deal with the tradeoff between exploration and exploitation of a search space, based on the contracting and expanding behavior of a fish school on the search for food. To test the applicability of this algorithm on real world problems, the SCNP was chosen as a problem, which is NP-hard and contains a high number of local optima. Exact mathematical solution techniques can be used to solve such a problem, but can entail runtimes unsuitable for the timespans, in which real world planning tasks have to be performed in.

A supply chain is defined as a network of organizations, which are connected by the flow of materials, information and financial resources in different processes and activities, which produce values in form of products and services and are driven by the demands of the final customer [2][3]. Modern supply chains show an ever increasing complexity due to several influences such as heterogeneity in customer needs, shorter product lifecycles and a globalization of the supplier base [4]. The management of such complex inter-organizational networks is a key capability for today's companies. In this context, SCNP aims at the determination of an optimal allocation of demands and capacities in inventory, transportation and production facilities for a tactical time horizon, consisting out of several planning periods, aiming to fulfill customer demands timely and at minimal costs [5]. It is assumed, that all information, which is required to create a plan for the whole supply chain, is centrally available for the decision maker. The main decision variables in SCNP are the planned production and inventory volumes in each facility and the transportation volumes between the different facilities. Here, a facility is e.g. a production plant or a warehouse. The outcome of SCNP is a master plan, which is a framework for the tactical and operational supply, production and distribution planning of each organization within the supply chain. Solution methods for the SCNP are available in commercial information systems, such as Advanced Planning Systems [3].

SCNP is performed within a planning horizon of typically months and as such aggregated data is used for the planning decisions in each facility, e.g. each production facility is described by a single capacity value which is determined by the bottleneck capacity in its internal production network. Therefore, the decisions for a single production facility within the SCNP (excluding the transportation volume decisions) can be broken down to a Capacitated Lot-Sizing Problem (CLSP) with setup costs The CLSP with setup costs has been proven to be NP-hard [6]. As the SCNP contains several facilities, which can be broken down to such problems, it is NP-hard as well.

To assess the relative performance of the FSS algorithm, two more established meta-heuristics, the Particle Swarm Optimization and the Differential Evolution algorithms, are also applied onto the SCNP. To assess the total quality of the generated solutions, they are compared to the optimal solution generated by an exact mathematical solution technique.

This paper is structured as follows: firstly, a mathematical problem formulation of a specific SCNP is described. In Section III, the four different solution techniques, which are applied to the SCNP, are shortly depicted. Within Section IV

the implementation approach for the application of the three different meta-heuristics for the SCNP is shown. Section V presents the results of the application upon a small test scenario. Finally, the paper will be summarized in Section VI and future research tasks will be outlined.

II. Mathematical Model of the Planning Problem

A common way to model planning problems such as the SCNP is a mathematical formulation. As most variables within the SCNP represent real world objects, which are indivisible, the SCNP is a mixed-integer problem, which also constitutes it as NP-hard in this formulation [7]. Furthermore, the general SCNP, as described in the introduction, is often customized to fit to a real world decision scenario. With the final purpose of applying the FSS on a problem of real world size in mind, such a customized model was chosen for further investigation. It considers an automotive supply chain, in which several suppliers, who produce according to the Built-to-Order (BTO) principle, deliver intermediate goods to an Original Equipment Manufacturer (OEM), who in turn produces final goods which are requested by its customers. This decision situation focuses on the alignment of production and inventory capacities, which are regarded as the main bottleneck in such a supply chain, as well as demands between all members of the supply chain. Therefore, capacitated transportation links, warehouses and distribution centers are in a first step omitted from this model, as they are not the in the center of investigation in this decision situation. The following mathematical model formulation, as it was presented by Hellingrath and Küppers [8], is adapted from the works of Pibernik and Sucky [9]. It is assumed, that the production, storage and transportation processes are occurring on different levels of the supply chain. The final level of the supply chain represents the customers, which have specific demands for final products. Table 1 describes the meaning of the used variables and parameters.

The objective function (1) of the model describes a minimization of all costs that occur as a result of the determination of the optimal allocation of demands and capacities within the supply chain: It includes production costs, which are split into product and facility specific setup costs for each production lot and volume, product and facility specific manufacturing costs for each produced item, periodic costs for adjustment measures in each facility, with which the capacities in a facility can be adapted to the existing demand, as well as transportation and inventory holding costs. Furthermore, the model contains a factor called “demand fulfillment”. It is used to determine the relative amount of fulfilled customer demand and can therefore also be used to calculate the unfulfilled demands. The amount of the latter is multiplied with fictional backorder costs as a penalty. The constraint (2) and (3) ensure the correct flow of materials throughout the network, i.e. products can only be delivered to a customer, when they are either already stored in inventory or can be produced within the same period, which in turn can only be done when the required amount of intermediate products is available is well. Constraint (4) represents the need to fulfill final customer demand by the last stage of the node and is used to determine the “demand fulfillment” factor. Constraint (5) provides that maximum capacities for the production of goods have to be met in each facility. A similar formulation is considered in constraint (6) and describes that inventory capacities have to be met as well. To simplify the model description, further constraints for the non-negativity of variables and the correct setting of helping variables have been omitted from this representation.

$$\begin{aligned} min \sum_{t=1}^{T}\sum_{i=1}^{I}\sum_{k\epsilon K_i}\sum_{n\epsilon N_i}\Bigg[& PQ_{i,k,t}^{n}\cdot ProdCst_{i,k}^{n} + z_{i,k,t}^{n}\cdot FixCst_{i,k}^{n} \\ & + \sum_{f\epsilon F_{i,k}} y_{i,k,t}^{f}\cdot PCst_{i,k}^{f} \\ & + \sum_{k\epsilon K_{i+1}} TQ_{i,k,l,t}^{n}\cdot TransCst_{i,k,l}^{n} + IQ_{i,k,t}^{n} \\ & \cdot InvCst_{i,k}^{n} + \sum_{m\epsilon N_{i-1}} IQ_{i,k,t}^{m}\cdot InvCst_{i,k}^{m}\Bigg] \\ & + \sum_{t=1}^{T}\sum_{n\epsilon N_I}(1-df_{n,t})\cdot BOCst_{n} \end{aligned} \quad (1)$$

Subject to:

$$IQ_{i,k,t}^{n} = IQ_{i,k,t-1}^{n} + PQ_{i,k,t}^{n} - \sum_{l\epsilon K_{i+1}} TQ_{i,k,l,t}^{n} \quad \forall\, i, n\epsilon N_i, k\epsilon K_i, t \quad (2)$$

$$IQ_{i,k,t}^{m} = IQ_{i,k,t-1}^{m} - \sum_{n\epsilon N_i} PQ_{i,k,t}^{n}\cdot BOMF_{i,m,n} + \sum_{l\epsilon K_{i-1}} TQ_{i-1,l,k,t}^{m} \quad \forall\, i, m\epsilon N_{i-1}, k\epsilon K_i, t \quad (3)$$

$$\sum_{k\epsilon K_I} TQ_{i,k,l,t}^{n} = df_{n,t}\cdot DM_{I,t}^{n} \quad \forall\, n\epsilon N_I, l\epsilon K_{I+1}, t \quad (4)$$

$$\sum_{n\epsilon N_i} PQ_{i,k,t}^{n}\cdot pc_{k}^{n} \le BC_{i,k,t}^{prod} + \sum_{f\epsilon F_{i,k}} y_{i,k,t}^{f}\cdot incC_{i,k}^{f} \quad \forall\, i,k,t \quad (5)$$

$$\sum_{n\epsilon N_i} IQ_{i,k,t}^{n}\cdot ic_{k}^{n} + \sum_{m\epsilon N_{i-1}} IQ_{i,k,t}^{m}\cdot ic_{k}^{m} \le BC_{i,k,t}^{inv} \quad \forall\, i,k,t \quad (6)$$

TABLE I. Descriptions of variables.

Var.	***Description***
t	Period
i	SC tier (i=1,…,I+1)
K_i	Set of nodes on tier i
Ni	Set of products produced on tier i
$PQ_{i,k,t}^{n}$	Production quantity of product n on node k (tier i) in period t
$ProdCst_{i,k}^{n}$	Production costs of product n (tier i) and node k
$z_{i,k,t}^{n}$	Binary variable indicating that product n is produced on node k (tier i) in period t
$FixCst_{i,k}^{n}$	Changeover costs for the production of product n and node k (tier i)
$F_{i,k}$	Set of adjustment measures available on node k (tier i)
$y_{i,k,t}^{f}$	Binary variable indicating that adjustment measure f is implemented on node k (tier i) in period t
$TQ_{i,k,lt}^{n}$	Transport quantity of product n from node k (tier i) to node l (tier i+1)
$TransCst_{i,k,l}^{n}$	Transport costs for product n from node k on (tier i) to node l (tier i+1)

(Continued..)

(Continued..)

M	Large number
$IQ^{m}_{i,k,t}$	Inventory quantity of intermediate product m on node k (tier i) in period t
$InvCst^{m}_{i,k}$	Inventory holding costs for intermediate product m on node k (tier i)
$df_{n,t}$	Fulfillment of the final customer demand of final product n in period t (in percent)
$BOCst_{n}$	Costs for unfulfilled demand of product n
$IQ^{n}_{i,k,t}$	Inventory quantity of final product n on node k (tier i) in period t
$BOMF_{i,m,n}$	Required quantity of intermediate product m (from tier i-1) for the production of final product n (bill-of-material factor)
$DM^{n}_{l,t}$	Ultimate customer demand for product n in period t
$pc^{n}_{i,k}$	Required production capacity for the production of product n on node k (tier i)
$BC^{prod}_{i,k,t}$	Production capacity of node k (tier i) in period t (basic capacity)
$incC^{f}_{i,k}$	Production capacity increase adjustment measure f on node k (tier i)
$ic^{n}_{i,k}$	Required inventory capacity of product n on node k (tier i)
$ic^{m}_{i,k}$	Required inventory capacity of intermediate product m on node k (tier i)
$BC^{inv}_{i,k,t}$	Inventory capacity of node k (tier i) in period t (basic capacity)

III. Applied Solution Methods

A. Fish School Search

Fish School Search (FSS), as proposed in 2008 [1], is a meta-heuristic search algorithm that was designed with inspiration in the collective foraging behavior of natural fish schools. The success of the search process is represented by the weight of the fishes, in a way that the heavier the individual is, the better the search results it made so far are. The weight of the fish is updated throughout the feeding process. A second means to encode success in FSS is the radius of the school (smaller radiuses mean better results). All the mechanisms are explained in the following sections.

Individual Movement Operator

In the Individual Movement Operator, each fish moves randomly and independently towards localities of improved solutions. This operator is executed only if the new position is better than the previous one. This movement is described by (7), where $x_{ij}(t+1)$ is the new value in the position vector of the individual i within the dimension j, $x_{ij}(t)$ is the old value, r is a random value between 0 and 1 and $step_{ind}(t)$ is the step size on time t. The new step size is calculated through (8), where $step_{ind_{init}}$ and $step_{ind_{final}}$ are the initial and final step sizes and $iterations$ is the maximum number of iterations.

$$\vec{x_i}(t+1) = \vec{x_i}(t) + r\, step_{ind}(t), \quad (7)$$

$$step_{ind}(t+1) = step_{ind}(t) - \frac{step_{ind_{init}} - step_{ind_{final}}}{iterations} \quad (8)$$

Feeding Operator

As mentioned before, the feeding operator is responsible for the weight update of all fishes. This operator is executed only if the fitness of a given fish increases during the individual movement. This process is defined by (9) where Δf_i is the fitness variation after the individual movement of the fish i, and $\max(\Delta f)$ is the maximum fitness variation in the whole population.

$$W_i(t+1) = W_i(t) + \frac{\Delta f_i}{\max(\Delta f)}. \quad (9)$$

Collective Instinctive Movement Operator

The Collective Instinctive Movement Operator is the first collective movement in the algorithm (i.e. it endows coherence for the search). This operator is defined by (10), where N is the population size, Δx_k and $\Delta f(x_k)$ is the position variation and the fitness variation of the individual of index in the individual movement.

$$\vec{x_i}(t+1) = \vec{x_i}(t) + \left(\frac{\sum_{k=1}^{N} \Delta\vec{x_{kj}} \Delta f(\vec{x_k})}{\sum_{k=1}^{N} \Delta f(\vec{x_k})}\right). \quad (10)$$

Collective Volitive Movement Operator

In this step, the population must contract or expand, using the barycenter of the fish school as a reference. It is calculated according to (11), where $W_i(t)$ is the weight of the fish i at the time t. The total weight of the whole population must be calculated in order to decide if the fish school will contract or expand. If the total weight increased after the last individual movement, the school as a whole will contract. Otherwise, the population will expand. This mechanism automatically switches between exploration and exploitation behaviors. The contraction and expansion processes are defined by (12) and (13), respectively, in which $step_{vol}$is the volitive step size constant.

$$\vec{B_j}(t) = \frac{\sum_{i=1}^{N} \vec{x_i}(t) W_i(t)}{\sum_{i=1}^{N} W_i(t)}, \quad (11)$$

$$\vec{x_i}(t+1) = \vec{x_i}(t) - step_{vol}\, rand(0,1) \frac{\left(\vec{x_i}(t) - \vec{B_j}(t)\right)}{distance\left(\vec{x_i}(t), \vec{B_j}(t)\right)}, \quad (12)$$

$$\vec{x_i}(t+1) = \vec{x_i}(t) + step_{vol}\, rand(0,1) \frac{\left(\vec{x_i}(t) - \vec{B_j}(t)\right)}{distance\left(\vec{x_i}(t), \vec{B_j}(t)\right)}. \quad (13)$$

B. Other Solution Methods

Two different and established meta-heuristics were also applied on the same planning problem to evaluate the relative effectiveness of the FSS. Differential Evolution (DE) is a stochastic population-based optimization algorithm, developed by Rainer Storn and Kenneth Price [10]. It was developed as a part of evolutionary computing in order to optimize nonlinear and non-differentiable functions in a continuous space. The parameterization of the algorithm can be self-adaptive [11]. A great advantage of DE is the use of low parameters (i.e. population size, differential weight and crossover probability). The Particle Swarm Optimization (PSO) algorithm, proposed by Kennedy and Eberhart 0, is an adaptive meta-heuristic based on the behavior of several individuals (i.e. particles). In PSO, each particle aims at to find solutions to a given problem

from their social interactions, based on the collective knowledge of the swarm and its own experience. PSO is fast and capitalizes on information shared within the neighboring strategy used. Additionally, the mathematical model was solved by CPLEX, a state of the art commercial solver, using well known algorithms such as the simplex algorithm and branch & bound [3][14][15]. It was used to determine the optimal solution value of the planning problem to measure the quality of the solutions generated by all meta-heuristics.

IV. Implementation Approach

The PSO, DE and FSS algorithms, just like all meta-heuristics, have certain limitations. They were all initially designed to solve continuous, unconstrained, static and single solution problems and do not natively include constraints into their search space. Since the SCNP is a discrete and constrained problem, there are several challenges that must be overcome when applying such algorithms onto this problem.

The first challenge is the determination of the information represented in the solution vector of each individual within the meta-heuristics. When all variables are included into the vector the algorithm would need to account for e.g. inventory, transport and production volumes to be aligned towards each other (i.e. inventory volumes in one period have to be consistent with production and transportation volumes of prior periods). The algorithms do not know the interdependencies of these variables, so they can create many infeasible solutions. Therefore, only production (pq) and transportation (tq) volumes were included into the solution vector. All other variables are calculated in dependence of these variables and the known parameters using the equations presented in Section II. Such an approach was also chosen in the work of Posignon and Mönch [16] and proofed to be applicable. Figure 1 shows exemplary the solution vector of an individual of a network with 4 production facilities for 4 different products containing 4 transportation links (it is assumed, that the transportation amounts towards the final customer equal its demand). This vector is repeated for each period in the planning problem.

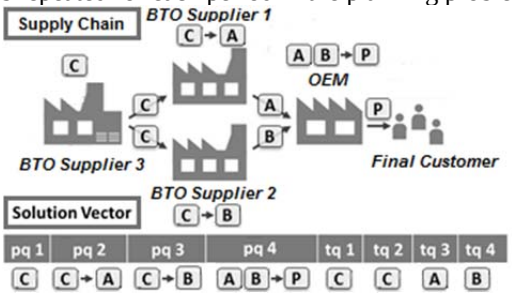

Figure 1: Solution Representation

All decision variables not represented in the solution vector of the meta-heuristic are heuristically calculated, i.e. inventory volumes for final products in a certain facility can be calculated by adding production volumes and incoming transportation volumes to the inventory volumes of the same product in the same facility of the previous period while also subtracting demands and outgoing transport volumes.

The second challenge is related to the aforementioned fact that the three algorithms were designed to handle continuous problems, while the SCNP is a discrete one. In order to satisfy the need for integer values, although the variables from the solution representation within the meta-heuristic are used as continuous variables during the optimization process, they were rounded to the nearest valid size of non-splittable items in the production network before being passed to the fitness function (i.e. objective), which determines the value of a given solution similar to the objective function presented in Section II.

The third challenge regards the fact that the algorithms can find invalid solutions during the optimization process. For example, a lack of production volumes for a final product, combined with an outgoing transportation of said product to satisfy a customer demand leads to a negative inventory level.

None of the algorithms were designed to naturally include constraints in their search space. Therefore, another way to ensure the creation of solutions which adhere to these constraints has to be chosen. This issue was approached by including additional penalty costs, which are applied in case of constraint violations. These penalty costs were not included in the mathematical model as presented in Section II. There are three types of constraint violations: A production capacity violation occurs if the total used production capacity set by the meta-heuristic is higher than the maximum production capacity of a given node. An inventory capacity violation results if the required inventory capacity calculated is higher than the maximum inventory capacity of a given node. Finally, a demand violation surfaces if customer's demand is not fulfilled. For all three types of violation, firstly, the degree of infeasibility is calculated, e.g.. the exact amount of overutilization of a capacity. Afterwards, the final penalty value is calculated by summing up the products of each measured degree of infeasibility and a respective penalty cost factor for each type of constraint violation. Therefore, a positive gradient is formed within the infeasible areas, in which the costs of the solutions increase as they get more and more distant from the borders between the feasible and infeasible areas. Thus, the individuals will avoid these areas. The penalty costs have to be sufficiently high, so that the algorithms will rate the value of a valid solution higher than that of an infeasible solution.

However, a problem of this approach is the vast amount of local optima as a result of the additional penalty costs, which can be seen in the following example: A final customer demand cannot be satisfied, because a capacity bottleneck occurs in a facility further upstream in the supply chain. A solution, in which all facilities in the network create all required items for this final product was found. However, one facility cannot produce the needed volumes of an intermediate product due to capacity constraints and penalty costs occur, as the inventory volumes become negative. The ideal solution would be to acknowledge this bottleneck at the last production node and produce less final products, so that no production and transportation costs occur for products, which cannot be produced in the real production system. However, this solution

would be represented by an individual with a vastly different solution vector. Therefore, the algorithm might be stuck in a local optimum, while the global optimum is at a completely different position within the search space.

In general, all costs which are represented in the mathematical model are also calculated within the implementation of the fitness function. This way, comparable solutions can be created and evaluated.

V. EXPERIMENTS

For testing purposes, we used a simple test scenario, which adheres to the structure of the supply chain shown in Figure 1 in Section IV. The scenario describes a planning problem for 10 periods, in which 4 different products are produced within 4 different facilities with varying final customer demands. These demands exceed the supply chains capacity in certain periods in order to force the creation of inventories. The final customers are directly delivered from the last production facility. Given that the representation of the solution vector shown in Figure 1 considers only one period, the individual/particle needs 10 times more information to represent 10 periods: A total of 80 dimensions are needed in order to determine the ideal solution of this optimization problem. It also contains a vast number of local optima, based on the reasoning in Section IV as well as on the combinatorial nature of the problem itself.

The experiment for the meta-heuristics consisted of varying the number of individuals and iterations (for each algorithm) and running each solution generation process 30 times. The search space boundaries were defined as 0 and 500 for every dimension. The Table II shows the configuration for each simulation. However, since the FSS calls the fitness function twice per iteration, the maximum number of iterations allowed for this algorithm amounts to half of the values of the same parameter for the PSO and DE algorithms. Additionally, the planning problem has been solved with CPLEX to determine the optimal solution value.

TABLE II. EXPERIMENTS DESCRIPTION.

Experiment	Population	Iterations
Exp_1	10	10000
Exp_2	10	1000
Exp_3	10	100
Exp_4	100	10000
Exp_5	100	1000
Exp_6	100	100
Exp_7	1000	1000
Exp_8	1000	100
Exp_9	10000	1000
Exp_10	10000	100

The two PSO parameters used in the experiments, the social and cognitive accelerations, were set to 1.3 and 2.8, based on suggestions from studies performed in [17]. Moreover, the update strategy is synchronous, since it is more appropriate for Global PSO [17].

The DE algorithm is quite robust with respect to the parameters differential weight (dw) and crossover probability (cp). In these experiments the dw and cp parameter values were set respectively to 0.35 and 0.2, based on studies performed in [18] which show this configuration gives a generally good convergence on a wide range of problems.

In [1] the maximum individual and volitive step sizes of the FSS were chosen according to the search space length. In this case, considering that the search space size for each dimension was set to 500, some experiments showed that the values 50 and 10 for these parameters showed the best results.

All parameters were set according to the results acquired in previous experiments. These values were chosen in an attempt to reach the best values in the final experiments.

VI. RESULTS

Table III shows the results for each approach. The best results for each experiment are highlighted. The FSS reached better results in most of the experiments. However, the PSO outperformed both algorithms in the cases in which there was a lower value for the maximum number of iterations (i.e. 100 for PSO and DE and 50 for FSS). The fast convergence power of this algorithm explains this behavior.

Due to the characteristics of the FSS, which allows it to avoid being trapped in local minima through the expansion and contraction mechanism, it was able to find the best results among all the tested algorithms. However, this mechanism causes a relatively slow convergence, necessitating a higher amount of iterations to reach good results. Observing the results in Table III, with more iterations and/or fitness function calls, the FSS algorithm is able to reach better results than the other algorithms. Smaller standard deviations also showed that the FSS is able to have more consistent results.

TABLE III. SIMULATIONS RESULTS. THE BEST RESULTS ARE IN BOLD.

Experiment	PSO		
	Best Fitness	**Mean**	**SD**
Exp_1	1.01E+07	2.01E+07	6.53E+06
Exp_2	9.49E+06	2.16E+07	6.29E+06
Exp_3	**1.79E+07**	**3.06E+07**	6.97E+06
Exp_4	1.51E+06	4.43E+06	2.63E+06
Exp_5	2.00E+06	5.68E+06	2.59E+06
Exp_6	**3.41E+06**	**9.37E+06**	**3.61E+06**
Exp_7	1.59E+06	4.32E+06	1.69E+06
Exp_8	**2.42E+06**	**4.96E+06**	**1.68E+06**
Exp_9	1.64E+06	3.54E+06	1.89E+06
Exp_10	**1.79E+06**	**3.29E+06**	1.69E+06
Experiment	**DE**		
	Best Fitness	**Mean**	**SD**
Exp_1	1.48E+06	2.16E+06	6.77E+05
Exp_2	2.35E+06	4.13E+06	1.23E+06
Exp_3	3.75E+07	6.36E+07	1.24E+07
Exp_4	1.39E+06	2.33E+06	6.23E+05
Exp_5	2.04E+06	3.98E+06	1.30E+06
Exp_6	3.42E+07	6.52E+07	1.18E+07
Exp_7	1.89E+06	4.38E+06	1.61E+06
Exp_8	2.99E+07	6.63E+07	1.15E+07
Exp_9	1.85E+06	4.37E+06	1.59E+06
Exp_10	2.79E+07	6.59E+07	1.10E+07

(Continued..)

(Continued..)

Experiment	FSS		
	Best Fitness	**Mean**	**SD**
Exp_1	**9.67E+05**	**1.14E+06**	**1.07E+05**
Exp_2	**2.06E+06**	**3.23E+06**	**5.00E+05**
Exp_3	4.20E+07	5.79E+07	**6.01E+06**
Exp_4	**5.35E+05**	**5.41E+05**	**3.87E+03**
Exp_5	**8.29E+05**	**1.04E+06**	**7.47E+04**
Exp_6	3.62E+07	4.44E+07	3.65E+06
Exp_7	**5.37E+05**	**5.73E+05**	**2.98E+04**
Exp_8	3.22E+07	3.83E+07	2.13E+06
Exp_9	**5.33E+05**	**5.61E+05**	**2.61E+04**
Exp_10	1.48E+07	1.79E+07	**1.63E+06**

The DE algorithm found results of substandard quality and was unable to overcome the other algorithms in the experiments. Even though the PSO outperformed the other algorithms in all experiments with low iterations, the results were still far from optimal. When comparing the best results of each meta-heuristic with the optimal solution calculated by the mathematical solver as shown in Table IV, it becomes obvious that both the PSO and DE algorithms became stuck in local optima. Therefore, the FSS was the only technique able to find good solutions, which adhere to all constraints.

TABLE IV. COMPARISON OF THE BEST RESULT FROM EACH APPROACH.

Approach	Best Solution	Gap to optimal solution
CPLEX	5.27E+05	-
FSS	5.33E+05	1%
PSO	1.51E+06	286.43%
DE	1.39E+06	264%

VII. CONCLUSION AND FUTURE RESEARCH

In this work, the applicability of the FSS algorithm on real world planning problems in the logistics domain was shown using a small test scenario. The FSS has demonstrated the ability to circumvent the high number of local optima in this problem domain. In contrast, the PSO and DE algorithms get stuck in local optima quickly and are unable to find good solutions. This proofs that the novel way to deal with the exploration versus exploitation tradeoff of the FSS allows it to outperform comparable meta-heuristics.

In future work, the FSS algorithm needs to be applied to real scenarios with test data from practice, including vast supply chain networks with explicitly modeled storage and distribution facilities as well as additional constraints like capacitated transportation links: While the FSS does not outperform exact mathematical solvers in solution time or quality in small problems, it should compare better solving larger problems, because mathematical optimization algorithms scale at least exponentially with the size of a problem, while a meta-heuristic such as the FSS should scale in a much more favorable manner. Additionally, several improvements of the implementation can be performed. For example, better ways to initialize the variable values at the start of the optimization can be determined. Another interesting approach is the inclusion of constraints into the search space of the meta-heuristic to avoid using penalty costs.

REFERENCES

[1] C. J. A. B Filho., F. B. de Lima Neto, A. J. C. C.. Lins, A. I. S. Nascimento., and M. P. Lima, , "A novel search algorithm based on fish school behavior," Systems, Man and Cybernetics, SMC 2008. IEEE International Conference on, 2008, pp. 2646-2651.

[2] Christopher, M., "Logistics and supply chain management, creating value-adding networks", Financial Times Prentice Hall, Harlow, 3rd ed., 2005.

[3] Stadtler, H., Christoph K., eds. "Supply chain management and advanced planning: concepts, models, software and case studies." Springer, 2008.

[4] Bozarth, C. C., Warsing, D., Flynn, B.B., Flynn, E.J., "The impact of supply chain complexity on manufacturing plant performance." Journal of Operations Management, 2009, 27. Jg., Nr. 1, pp. 78-93.

[5] Simchi-Levi, D., Kaminsky, P., Simchi-Levy, E. "Designing and Managing the Supply Chain - Concepts, Strategies and Case Studies", Mc graw-Hill, 2008.

[6] Florian, M., Jan K. L, Kan, A.R.. "Deterministic production planning: Algorithms and complexity." Management science 26.7 1980, pp. 669-679.

[7] Nemhauser, G. L., Wolsey, L. A., "Integer and combinatorial optimization" ,Vol. 18, New York: Wiley, 1988.

[8] Hellingrath, B., Küppers, P., "Multi-Agent Based Evaluation of Collaborative Planning Concepts in Heterarchical Supply Chains." In: Proceedings of the Logistikmanagement 2011 *Bamberg, Germany*, 2011, pp. 1-22.

[9] Pibernik, R., Sucky, E., "Master Planning in Supply Chains." in:Supply Chain Management und Logistik. Physica-Verlag HD, 2005. pp. 69-93.

[10] Storn, R., Price, K., "Differential Evolution – A simple and efficient adaptive scheme for global optimization over continuous spaces." International Computer Science Institute, 1995.

[11] Engelbrecht A., "Computational intelligence: An introduction." 2nd ed. John Wiley & Sons, 2007. 597p.

[12] Kennedy, J. ,Eberhart, R. "Particle Swarm Optimization", IEEE International Conference on Neural Networks (Perth, Australia), IEEE Service Center, Piscataway, NJ, IV, 19985, pp. 1942-1948.

[13] Clerc, M. "The swarm and the queen: towards a deterministic and adaptive particle swarm optimization". Proceedings, 1999 ICEC, Washington, DC, pp 1951-1957.

[14] Suhl, L, Mellouli, T. "Optimierungssysteme: Modelle, Verfahren, Software, Anwendungen". Springer, 2009.

[15] Van Hentenryck, P. "The OPL Optimization Programming Language." MIT Press, 1999.

[16] Ponsignon, T., Mönch, L. "Heuristic approaches for master planning in semiconductor manufacturing." Computers & Operations Research 39(3), 2010, p 479–491.

[17] Carlisle, A.. Dozier, G. "An off-the-shelf PSO. Proceedings of the Workshop on Particle Swarm Optimization". Purdue school of engineering and technology, Indianapolis, IN, 2001a.

[18] Ursem, R. K, Vadstrup, P. "Parameter Identification of Induction Motors using Differential Evolution." In Proceedings of the Fifth Congress on Evolutionary Computation (CEC-2003), 2003, pp 790–796.