Comparison of cooperative and classical evolutionary algorithms for global supply chain optimisation

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

***Abstract*— This paper discusses global optimisation from a business perspective in the context of the supply chain operations. A two-silo supply chain was built for experimentation and two approaches were used for global optimisation: a classical evolutionary approach and a cooperative coevolutionary approach. The latter approach produced higher quality solutions due to its use of communication between silos. Additionally, a second problem was presented involving an existing Australian multi-factory sheet steel business.**

I. Introduction

For the last few years most production-based businesses have been under enormous pressure to optimise their supply chains. Several studies have investigated optimisation techniques for various supply chain components. Some common components studied include the job-shop scheduling problem [1], [2], [3], planning and cutting problems [4], [5], routing problems [6], allocation and distribution problems [7], [8]. Supply chain problems are generally non-linear, heavily constrained, and can involve many variables. Due to the high level of complexity, it becomes virtually impossible for deterministic systems or even (human) domain experts to find an optimal solution. Moreover, what-if analysis, which requires calculating and recalculating solutions for different scenarios is an expensive exercise.

Large businesses typically breakdown their operations into components such as purchasing, production, and distribution. Each component operation can be referred to as a silo and for true global optimisation all silos must be taken into account. Optimisation of each individual silo in isolation may not lead to the global optimum. Thus, large businesses tend to be more interested in optimisation of their whole system rather than optimisation of single components of the system. This leads to the concept of *global optimisation from a business perspective*.

This paper compares two techniques for global supply chain optimisation. One technique employs classical evolutionary optimisation of silos, that is one silo is optimised and its output is then used as an initial input parameter for the optimisation of the second silo. The other optimises both silos in parallel with periodic communication between the two optimisation executions. Each technique is tested on a simulated supply chain.

Results show that parallel optimisation with communication yields higher *quality* solutions. In this paper, the *quality* of a solution is determined by its proximity to the global optimum.

In recent years, there has been an increased interest in solving supply chain management problems using evolutionary algorithms. David Naso et al. [9] look at the problem of coordination of just-in-time production and transportation in a network of partially independent facilities of ready-mixed concrete. They optimize the network of independent and distributed production centres serving a number of customers distributed across a certain geographical area. This problem, which has high complexity and strict time delivery constraints, is approached with a meta-heuristic based on a hybrid genetic algorithm with combined constructive heuristics.

Altiparmak et al. [10] proposes algorithms using mixed-integer, non-linear programming model for multi-objective optimization of a supply chain network based on the real world problem of a company that produces plastic products. They compare three approaches to find the set of Pareto-optimal solutions and discuss the pros and cons of each of them.

Zhou et al. [11] present a novel genetic algorithm to solve bi-criteria, multiple warehouse allocation problem. Proposed method finds the Pareto-front of wide range of non-dominated solutions without the arbitrary determination of weighting coefficients.

Even much earlier, researchers had been working on this type of problem as can be seen in the following references. Lee and Choi [12] apply genetic algorithms to solve a single machine scheduling problem with distinct dates and attempt to minimize all penalties. This method produces near optimal solutions, which they prove by comparison with an exact algorithm. Lee et al. [13] address the problem of inventory management of a refinery that imports several types of crude oil and proposes a mixed-integer linear programming model. Martin et al. [14] create a linear programming model to optimize flat glass production.

A. Coevolution

Coevolution is a simultaneous evolution of several genetically isolated subpopulations of individuals that exist in

Maksud Ibrahimov is with the School of Computer Science, University of Adelaide, South Australia 5005, Australia, email: maksud.ibrahimov@adelaide.edu.au.

Neal Wagner is with the SolveIT Software, Pty Ltd., 99 Frome Street, Adelaide, SA 5000 Australia, email: nw@solveitsoftware.com.

Arvind Mohais is with the SolveIT Software, Pty Ltd., 99 Frome Street, Adelaide, SA 5000 Australia, email: am@solveitsoftware.com.

Sven Schellenberg is with the SolveIT Software, Pty Ltd., 99 Frome Street, Adelaide, SA 5000 Australia, email: ss@solveitsoftware.com.

Zbigniew Michalewicz is with the School of Computer Science, University of Adelaide, South Australia 5005, Australia. Institute of Computer Science, Polish Academy of Sciences, ul. Ordona 21, 01-237 Warsaw, Poland, Polish-Japanese Institute of Information Technology, ul. Koszykowa 86, 02-008 Warsaw, Poland, email: zbigniew.michalewicz@adelaide.edu.au.

978-1-4244-8126-2/10/$26.00

common ecosystem. Each subpopulation is called species and mate only within its species. In evolutionary computation coevolution can be of two types: competitive and cooperative.

In *competitive* coevolution, multiple species coevolve separately in such a way, that fitness of individual from one species is assigned based on how good it is competes against individuals from the other species. One of the early examples of competitive coevolution is the work by Hillis [15], where he applied competitive predator-prey model to for the evolution of sorting networks. Rosin and Belew [16] used competitive model of coevolution to solve number of game-learning problems including Tic-Tac-Toe, Nim and small version of Go.

Cooperative coevolution uses divide and conquer strategy: all parts of the problem evolve separately, fitness of individual of a particular species is assigned based on the degree of collaboration with individuals of other species. Individual of each species evolves to contribute to its part within the single ecosystem. Two different models of cooperative coevolution were proposed: model by Potter [17] and model by Moriarty and Miikkulainen [18]. This paper mainly focuses on the Potter model which is more general model from the model proposed by Husbands and Mill [19]. Also somewhat similar model of evolution was proposed previously by Holland [20]. Potter applied cooperative coevolution artificial neural networks [21], to concept learning for construction of immune system [22], and optimisation of some standard functions [23].

Wiegand analysed cooperative coevolution on several pseudo-boolean functions, showed relationship between cooperative coevolution and symmetric games, proposed hierarchical categorisation of design choices for coevolutionary algorithms, performance effects on update timing, and analysis methods using evolutionary game theory and dynamic systems [24].

Popovici in her work used dynamical systems theory to analyse cooperative coevolutionary algorithms (CCEAs). She looked at the dynamics of individuals using fitness landscapes, explained the problem of convergence to the Nash Equilibrium with best response curves [25], analysed sequential and parallel CCEAs [26], investigated relationships between internal and external metrics [27]. She used two dimensional fitness landscapes with the parameter α to vary landscape surface in most of her work.

The rest of the paper is organised as follows: in section II the simulated supply chain is presented. Section III describes algorithms that address it. Experiments and results are described in section IV. The following section V will present a case study of production optimisation of an existing Australian company with several plants across the country. This paper will be finalised with the discussion of future work and conclusions in section VI.

II. Two-silo experimental supply chain

This section describes a two-silo simulated supply chain used for testing and comparing the two global optimisation techniques studied in this paper. These techniques are detailed in sections 3 and 4. The first silo is a production silo based on the scheduling problem, the second silo is a distribution silo based on the vehicle routing problem. Products made in the first silo are moved to the second silo for distribution. The goal of optimisation for this supply chain is to minimise overall expenses that occur with production and distribution.

A. Scheduling

Scheduling is an important part of real-world supply chain problems, especially in manufacturing and production systems. Scheduling addresses the problem of allocating resources to tasks with specific time intervals and constraints. Several types of scheduling problems exist, including job-shop scheduling problems (JSSP), flow-shop scheduling problems, open-shop scheduling problems and others. All these scheduling problems are NP-complete in nature and most of them involve multiple objectives such as maximising profit, and minimising production time. In this paper we are considering JSSP.

B. Vehicle routing problem

The Vehicle Routing Problem (VRP) was first proposed by Dantzig and Ramser [28] and was known as the Truck Dispatching Problem. Since then it has became one of the most famous combinatorial problems in operation research as it has many applications in practice especially in distribution management. This problem is a combination of the *Travelling Salesman Problem* and the *Bin Packing Problem* which are both known to be NP-hard. Thus, deterministic algorithms for VRP are only practical for small instances of the problem. For bigger instances non-deterministic methods should be applied.

C. The Problem

A simulated supply chain with two silos has been built to investigate the issues connected with its global optimisation. The first silo is a production silo that assembles goods and then ships these finished products for consideration in to the second silo. The second silo serves as a distribution component. It transports goods to customers using a fleet of trucks. This simulates processes in real-world supply chains. Below is more detailed description of functionality of each silo.

The *first* component in the supply chain is a production scheduling silo, and it is represented in the literature as *job-shop scheduling problem* (JSSP). Recall, that this problem consists of a set of jobs $J = \{J_1, \ldots, J_n\}$ and a set of machines $M = \{M_1, \ldots, M_n\}$ which these jobs can be executed on. Each job has a set of operations O and a specific order of execution on each of the existing machines. Each operation i of the job J_j on machine M_k takes certain processing time and denoted by $P_{i,j,k}$. Two operations can not be executed on a machine at the overlapping intervals of time. The goal is to find such ordering of these jobs on the

machines to minimise the total time needed to produce all of them [1].

The *second* component of the system is a distribution silo and it represents the *vehicle routing problem* (VRP). The basic VRP is defined as follows. Given undirected graph $G = (V, E)$ where $V = \{0, \ldots, n\}$ is the set of vertices. Each vertex $v \in V \backslash \{0\}$ represents a customer destination with non-negative demand q_i and vertex 0 represents a depot. $E = \{(i, j) : i, j \in V, i < j\}$ is a set of edges which serve as routes between destinations with travel cost c_{ij}. Depot has a fleet of m identical vehicles each with maximum vehicle capacity of Q and maximum route length of L. The goal of a problem is to find a set of m or at most m routes with the minimal total travel cost such that each customer is visited exactly once and each route starts and ends at the depot. Solution needs to satisfy maximal capacity and maximal route length constraints.

Combination of these two components yields our simulation of the supply chain system: one silo is producing goods and the other one is responsible for distributing them to customers. The overall supply chain works as follows. After an order is produced at the production silo it is moved to the distribution centre's depot of the second silo. The depot then assigns a truck and selects a route for delivery to the customer. To make this model closer to the real world, trucks deliver goods to customers by the end of each production period of length τ. For example if $\tau = 7$ then delivery of goods start at the end of the week for products produced during the week. If during production week, products a and b were produced, then both are shipped to customers at the end of the week. Later in the paper we will refer to orders produced within one τ-period as a *bucket*.

The performance of the first silo is measured in time units, while the performance of vehicle routing problem is measured in kilometres. In order to view the global performance of the supply chain, all this is converted to money equivalent:

$$Cost = \alpha X + \beta \sum Y_i \tag{1}$$

where X is a makespan of the produced schedule in time units, and Y_i is a total trip length of all trucks during production period i. Thus αX is a production cost and $\beta \sum Y_i$ – transportation cost. The goal of the problem on this supply chain is to minimise $Cost$.

For optimisation of VRP silo, we use Prins' algorithm [29] and a customised algorithm based on common techniques cited in the literature [1] for optimisation of JSSP silo. Using these algorithms as *building blocks*, two different approaches of solving the problem are discussed in the next section: classical evolutionary approach and approach based on cooperative coevolution [17].

III. Algorithms

Two local algorithms to address JSSP and VRP are described in this section. Later in the section, we explain how to use these algorithms as building blocks to construct algorithms that solve problem of global optimisation from the business perspective of this supply chain.

The main point of this section is to show that combination of optimal local solutions does not guarantee the quality of the global solution.

A. Algorithm for the JSSP

The following version of evolutionary algorithm was used for this problem.

Solution Representation

Each solution in chromosome is encoded as a permutation in the following way:

$$X = (x_1, \ldots, x_{nm}) \tag{2}$$

where n is the number of jobs and m is the number of machines with total of $n \times m$ genes. This representation is operation-based representation which means that each gene in the chromosome is one operation and each job appears in the chromosome exactly m times. An advantage of this representation is that all permutations make a valid schedule. This representation was proposed by Gen et al. [30]. Example of such representation is chromosome

$$(1, 2, 1, 2, 3, 3, 2, 1, 3).$$

which mean that operations will be executed in the following order: O_{11}, O_{21}, O_{12}, O_{22}, O_{31}, O_{32}, O_{23}, O_{13}, O_{33}. Here O_{ij} is j-th operation of the job J_i. To build a schedule from this sequence we start from m empty machines with no orders on them and then add operations in the order specified by the chromosome. Each operation is assigned greedily to its machine in such a way that it's executed as soon as possible without violating constraints: operation $O_{i,j+1}$ can not be executed before operation O_{ij} is finished and no two jobs can be executed at the same machine at the same time.

Operators

The following operators were used for the algorithm:

- *Partially mapped crossover (PMX).* This operator can be thought of as a crossover for permutations that as it preserves count of each element in the permutation. This operator has been proposed by Goldberg and Lingle to address blind travelling salesman problem. For details see [31].
- *InsertionMutation.* This operator selects a gene at random and then inserts it to another random position.
- *InversionMutation.* This operator [32] randomly selects two points in the chromosome and swaps the sequence of numbers in between these points.
- *SwapMutation.* This operator, also known as reciprocal exchange mutation operator, selects two genes in the chromosome at random and then swaps them.

Main algorithm loop

The first step of the algorithm is population initialisation which creates population with random individuals. Each individual is assigned a fitness value by an evaluation function which is designed so that its value is indicative

[1] more details on JSSP can be found in [1]

of how well the individual solves the problem. For this problem evaluation function is a makespan of the schedule created from the chromosome taken with the negative sign: $FitnessSched = -Makespan(schedule)$

During the main algorithm loop, individuals with higher fitness values are given more chances to reproduce. Reproduction process is made with the help of operators which take one or two individuals and produce offsprings. Every generation each of the operators are executed with specified probability. This algorithm uses steady state population and the tournament selection of size two.

B. Algorithm for the VRP

An evolutionary algorithm was used to solve the VRP problem. This algorithm is based on a memetic algorithm created by Prins [29].

Solution Representation

Each chromosome of the population is the permutation of n clients. In contrast of some algorithms [33] for VRP this permutation does not contain trip delimiters. Order of genes in the chromosome represents order in which these customers would be visited by a single truck. Optimal splitting procedure *Split* is then used to produce best partitioning of trips. It is based on the finding shortest path on the auxiliary graph, similar to the work of Beasley [34] for route-first cluster-second heuristic.

This weighted auxiliary graph $H = (X, A, Z)$ contains set of $n+1$ nodes $X = \{0, \ldots, n\}$, set of edges between nodes A and trip cost of the edge z_{ij}. Here, edge (i, j), where $i < j$, exists if trip from customer $i + 1$ to j is feasible i.e., is does not violate capacity and route length constraints.

An optimal route partition corresponds to the shortest path from depot, which is node 0, to the customer n in the auxiliary graph. Shortest path can be computed fast enough using Dijkstra's algorithm. Good description of this algorithm can be found in [35].

Current implementation does not actually build H in memory but rather builds it online getting the list of all neighbouring edges only when needed.

Operators

The following operators were used for the algorithm:

- *Order Crossover (OX)*. Since each chromosome of the population is a simple permutation classic permutation operator such as OX can be applied. It was first proposed by Davis [36]. Basically, it builds offspring by choosing a subsequence from one parent and preserving the relative order of genes from the other. Firstly, two cut points are selected at random, and segments between these points are copied to the offsprings. Next, starting from the second cut point of one parent, genes from the second parent are copied in the same order, omitting the symbols already present in the first parent. After reaching the end of the sequence, we continue from the start of the string.
- *Local Search Operator (LS)*. This operator, using *Split* procedure, first converts genotype (permutation) to the phenotype which is a set of truck routes. Then, it looks at certain $O(n^2)$ neighbourhoods generated with 9 different trip modifications. For details of these modifications see [29]. Each iteration of LS stops at the first improving modification. After this modification applied the process repeated again until no modification can be found.

Main algorithm loop

In order to avoid premature convergence a special *dispersal property* is maintained in the population. The basic idea of the dispersal property, is to have the population sorted in increasing order of fitness values and the fitness difference between two consecutive individuals set at least $\Delta > 0$.

$$\forall P_1, P_2 \in \Pi : P_1 \neq P_2 \Rightarrow |F(P_1) - F(P_2)| \geq \Delta \qquad (3)$$

where Π is a population and F is the evaluation function for the chromosome.

The initial population consists of σ chromosomes generated at random in such a way that they comply with the dispersal property 3. They generated one at a time, and if the new chromosome has difference with one of the chromosomes already in population of less than Δ it gets dropped and a new one is generated, otherwise it is inserted into the specific position in population to keep population sorted.

During the main algorithm loop, individuals with higher fitness values are given more chances to reproduce. Selection of parents is made with tournament selection of size two. Every generation two parents are selected, run trough OX operator and then one of the offsprings produced selected. This offspring is improved by LS operator with certain probability p_m. After this, if created offspring does not violate dispersal property 3, it replaces an individual from the second half of the population, i.e individual with index greater or equal than $\lfloor \sigma/2 \rfloor$. Steady state population is used for this evolutionary algorithm.

C. Classical approach

The two algorithms discussed above (for JSSP and VRP) produce reasonably good solutions to the problems. They were checked against the standard data sets. These algorithms are used to optimise proposed model of the supply chain.

For the actual supply chains, the production silo has a manager who is in charge of optimal scheduling of operations and the distribution silo has another manager that tries minimise transportation costs of the goods received from the first silo. Both managers try to optimise their work thinking only about their own silo without much exchange of information.

In this experiment a simulated supply chain is created that mimics this process. The JSSP algorithm will be run on the dataset of the first silo producing an optimal production schedule. The produced schedule will then be fed into the algorithm of the second silo. In the production silo, the whole makespan is divided into intervals of length τ. These

intervals correspond to the production periods when trucks start deliver goods produced during this period. For orders of each such period (bucket) VRP algorithm is run and results of each run are summed. The total performance of the supply chain is computed using equation 1. This method will produce global solution of the supply chain and will be referred further in the paper as *sequential approach*.

D. Cooperative coevolutionary approach

Another approach to tackle the aforementioned supply chain problem is based on cooperative coevolution. In this method two algorithms, each corresponding to a single silo, are run in parallel. Communication between silos occurs during evaluation of individual solutions. Solutions from one silo are evaluated based on their performance when combined with representative solution from the other silo and vice versa. JSSP and VRP algorithms described above are for silos 1 and 2 respectively.

Fitness evaluation of individuals

The main difference of cooperative coevolutionary algorithm is the way it evaluates individuals. In order to evaluate fitness of the individual, several *partner representatives* from another species are selected and combined with the individual, thus forming the global solution for the supply chain. The fitness of the best combination is then assigned to the individual and this partner is saved in the chromosome.

The question here is how to combine the two individuals? A scheduling individual produces a product order in which goods are made. From this product order job order buckets are extracted. Each job order in a bucket has a customer and each gene of the VRP individual also corresponds to a customer. Therefore job order buckets can be combined with a VRP individual to produce truck routes. To get routing information from a VRP individual corresponding to a particular bucket, only genes that refer to customers from job orders of that bucket need to considered. All other genes are ignored. Suppose, we have bucket with orders that need to go to customers 2, 5 and 7 and our VRP chromosome is (5, 2, 3, 4, 6, 1, 7, 8). Modified chromosome for this bucket is (5, 2, 7), so to these orders will be distributed in this exact order. $Split$ is then applied to form the routes. The same combining procedure is applied to all the buckets, then distances of all routes are added and multiplied by coefficient β to get the total transportation cost. The global cost of the solution is transportation cost plus production cost computed with the equation 1.

Main algorithm loop

In the beginning, each of the species is initialised with the random individuals by procedure $initialise()$. As with VRP algorithm, random initialisation in VRP species should be done in such a way that population of VRP species comply with the dispersal property 3.

During the main algorithm loop, JSSP algorithm evolves scheduling species and VRP algorithm evolves VRP species exactly the same way as in the single silo optimisation, with a few exceptions. One difference is that fitness evaluation is done in cooperative way as was described above rather than in the classical way. Each species use the same operators as were used in the original algorithms, except additional *InsertionMutation*, *InversionMutation* and *SwapMutation* are applied and *LS* operator improves routes only after solutions are combined and routes created. Procedure $applyOperators(species)$ applies operators relevant to each species.

Selection of partner representatives is done by the procedure $selectPartnerIndivudals()$ in the following way. For representatives from scheduling species 3 individuals are selected: the best and two random ones. For partner representatives from VRP species, population is divided into three equal intervals and random individual from each interval is chosen. Since individuals in the population are sorted by fitness values, this approach makes sure that diverse range of individuals are selected. In addition to 3 random individuals the best individual (first individual in population) is also selected. Due to the high dependency between the two silos strictly greedy algorithm of choosing just best individuals will not produce good results. This dependency effect of was studied by Potter in [17] and may be explained that with best partner strategy algorithm gets frozen in the Nash equilibrium.

To give each species some time to adapt to representatives of the other species, partner representatives are not selected every generation, instead they selected every certain number of generations called *freezing time* and stored in memory. Each species has its own representative freezing time.

During the evolutionary loop both species use steady state populations and tournament selection of size two for parent selection, the same as in the original JSSP and VRP algorithms.

Algorithm 1 Coevolutionary global optimisation

```
gen ← 0
for species ∈ allSpecies do
  initialise(species)
end for
for species ∈ allSpecies do
  evaluate(species)
end for
while gen < maxGen do
  gen ← gen + 1
  selectPartnerIndivudals()
  for species ∈ allSpecies do
    applyOperators(species)
    evaluateNewIndividuals()
  end for
end while
```

IV. Experiments and results

This section presents the results of experimental runs of sequential and coevolutinary approaches to the two silo supply chain problem.

The evolutionary algorithm for this problem was implemented in Java. A set of tests were developed based on the

standard data sets for vehicle routing and job-shop scheduling problems. JSSP data sets were taken from instances proposed by Taillard in [37] in particular ta11, ta35, ta51, ta70 and ta71 were used. For VRP test cases instances proposed by Christofides et al. [38] were taken, in particular CMT-1, CMT-2 and CMT-3. In table I column *Size* corresponds to the number of customers and hence the number of orders in the experiment.

In order to make data set for the supply chain model one instance of JSSP data set for first silo and one instance from VRP dataset for the second silo were taken. The following combinations were used: ta11 with CMT-1, ta35 with CMT-1, ta51 with CMT-2, ta70 with CMT-2 and ta71 with CMT-3. This defines job orders, depot location, customers and customer coordinates. The missing link between two silos is the mapping between job orders and customers was developed for each data set.

Due to stochastic nature of evolutionary algorithms, each experiment was executed 50 times so as to allow for statistically accurate results. Results average, standard deviation, minimal and maximum of all runs were recorded. For the total price evaluation from equation 1 $\alpha = 1.0$ and $\beta = 2.0$ were taken. Parameters for the sequential approach are the following:

- JSSP algorithm:
  - Number of generations = 1000000
  - Population size = 50
  - PMX operator probability = 0.7
  - InsertionMutation operator probability = 0.3
  - InversionMutation operator probability = 0.3
  - SwapMutation operator probability = 0.3
- VRP algorithm:
  - Number of generations = 10000
  - Population size = 30
  - Crossover operator was executed every generation
  - Local search probability = 0.2
  - Δ = 0.5

Below are parameters for the coevolutionary approach:

- Number of generations = 200000
- Parameters were used for scheduling species:
  - Species size = 200
  - Freeze time = 50
  - PMX operator probability = 0.7
  - InsertionMutation operator probability = 0.5
  - InversionMutation operator probability = 0.5
  - SwapMutation operator probability = 0.5
- The following parameters were used for VRP species:
  - Species size = 200
  - Freeze time = 50
  - Order crossover operator probability = 1.0
  - InsertionMutation operator probability = 0.5
  - InversionMutation operator probability = 0.5
  - SwapMutation operator probability = 0.5
  - Δ = 0.5

TABLE I
RUN RESULTS

Experiment	Size	Average	Min	Max	StdDev
Sequential	20	2603.28	2407.01	2800.69	93.57
Coevolutionary	20	**2505.59**	2365.35	2722.77	69.18
Sequential	30	3761.86	3451.21	4127.81	105.98
Coevolutionary	30	**3514.18**	3389.71	3597.83	68.16
Sequential	50	5941.21	5453.55	6390.89	205.68
Coevolutionary	50	**5652.39**	5372.54	5969.93	133.99
Sequential	50	6387.10	6046.69	6729.65	156.76
Coevolutionary	50	**6095.35**	5732.16	6336.52	149.86
Sequential	100	11859.45	11205.47	12532.39	295.91
Coevolutionary	100	**10762.09**	10312.97	10976.93	171.37

These parameters were chosen by the series of manual exploratory experiments. Table I shows the comparison of the results. In all the data sets we can see that coevolutionary approach produce better results on average with smaller standard deviation. Number of fitness evaluations in coevolutionary approach is about 1.4 times larger than in the classical one. However, increasing number of fitness evaluations in the latter method would not produce significantly better results as after number of generations used in the current algorithm it converges to the local optimum. Another issue with cooperative approach is that fitness evaluation of each pair of individuals is computationally expensive simulation. With classical approach simulation needs to be run only for evaluation of VRP individuals.

In the following section a practical approach for global optimisation is implemented on a real-world supply chain and based on software that was written to solve real-world business problem for actual industrial company.

V. REAL-WORLD STEEL COMPANY OPERATIONS

An existing Australian company that specialises in producing sheet steel is considered. Proposed solution and experiments conducted are presented. To protect the confidence of the company it will be called Global Steel in this paper.

A. Problem Statement

Global Steel has several plants across Australia and specializes in producing sheet steel. Sheet steel products are basically thin sheets of steel that have been coiled into rolls. There are many different types of products that the company can manufacture, that differ in the chemical composition, density and width. Furthermore, the rolls can be configured in different diameters and cores. When a plant manufactures a roll of steel, the width of the roll cannot be changed. However, customers typically place orders for smaller width, and so, large rolls must be cut into pieces along their width to satisfy these orders. This requires orders of similar configuration to be grouped together before cutting the sheets.

The constraints and business rules of this problem are listed below; many of which are similar to the previous problem.

- Each plant's daily work hours are limited.
- Each plant can produce a subset of all possible goods. Some products can be produced at multiple plants, but

the cost and rate of production is different at different plants.
- Each plant is equipped with a particular number of 'knives', which are used to cut large rolls into smaller ones. Each plant can only produce sheets of fixed width and has special knives to cut to the width needed.
- Each plant has a fixed daily operating cost which is the cost of running the plant, and does not depend on the actual products produced.
- After production, products are delivered to customers, and this incurs a transportation cost. There are defined transportation costs between each customer and plants. Some plants cannot ship to some customer due to specific business rules.
- Due to shipping constraints, export orders must be produced and shipped from a single plant; whereas for others, part of the order can be produced at one plant and other parts at other plants.
- Orders are produced in batches by the plants, which will be explained in detail later in this section.

Plants can produce rolls of sheet steel of a certain width only, which is generally much larger than the widths typically required by customers, and therefore, ordered items that have the same type, core, and diameter must be batched into a single roll. That is, a large roll is cut into pieces to satisfy these orders.

This process of batching sometimes results in wastage. To illustrate, suppose that customers C_1 and C_2 need rolls of product I_1, of the same diameter and with the same type of core. C_1 needs 2 rolls of width 500 mm, and C_2 needs 4 rolls of width 600 mm. Let's assume that the only plant that can produce I_1, must produce rolls of steel of width 2000 mm. The plant can produce two rolls of I_1 of width 2000mm. One can be cut into 4 pieces: 3 pieces of 600mm and 1 piece of 200mm. The 3 pieces of 600mm can be used for orders, but the remaining 200mm is wasted. It may be recycled, but a loss is incurred nevertheless. The second roll of 2000mm can be cut into widths of: 600mm, 500mm, 500mm, 400mm, the first 3 of which would go towards satisfying orders, and the last piece ends up as wastage. Some wastage is unavoidable, but clearly it is desirable to have as little wastage as possible in order to maximize production efficiency and hence profit. Intelligent grouping of orders can minimize wastage.

Plants use cutting tools called knives, physically arranged in parallel, to cut large rolls into smaller ones. Different plants are equipped with different numbers of knives, and this presents another constraint to consider when batching orders. A plant equipped with n knives can cut a roll into at most n+1 pieces.

$$Profit = S - C_m - C_t \tag{4}$$

where S is *selling price* of the goods produced, C_m is the *cost of manufacturing* those goods, and C_t is the *cost of transporting* the goods to customers. Selling prices may vary for the same product depending on the market which it is being sold. The cost of manufacturing a particular order item is the sum of the cost per hour of producing that grade of sheet steel, and the fixed hourly cost of running the relevant plant for the time it takes to produce the required tonnage. Transportation costs vary with the destination address of the customer. Export orders naturally incur higher transportation costs. These last two components are the negative components of the profit; the bigger they are, lesser is the profit. The selling price is the positive component of profit; it is known before hand, and does not depend on which plant an order is allocated to. Therefore, the goal of the problem is to minimize the transportation and manufacturing costs.

B. Experiments and Results

The evolutionary algorithm in this study was implemented in Java, and was tested on real world industrial data. The purpose of the software is to provide decision support to business managers who approve production allocation.

In this subsection we present the results of experimental runs of this algorithm with a view to gaining an understanding of how the algorithm solves the problem.

An experiment was performed on a set of real-world data for a particular month. Due to the stochastic nature of evolutionary algorithms, the experiment was executed 50 times so as to allow for statistically accurate results. The specific parameters used for the experiment are shown below.

- Population size = 250
- Tournament selection of size 2
- A steady state population is used

A 90 run execution shows median 1.18E+07 and 52110.52404 standard deviation of profit. The solution generated by algorithm produced financial results that are 4% better (which is equivalent to several hundred thousands of dollars) than those produced manually by the companys experienced team of planners.

VI. Conclusion and future works

In this paper we considered two supply chain problems. The first is a simple two-silo supply chain with production and distribution components. Two approaches were employed on it: a classical evolutionary and a cooperative coevolutionary approach. The latter produced higher quality solutions due to its use of communication between supply chain silos.

Additionally a second supply chain problem was presented involving an existing Australian multi-factory sheet steel business. This application illustrated how a broader perspective can be taken in such a way as to optimize the operations of all the factories at the same time, as well as some aspects of operations of each individual factory.

Our future work will focus on refining the cooperative coevolution technique described in this paper as well as investigating alternative methods of addressing the global supply chain optimisation problem. The refinements we plan to consider include techniques to handle supply chain constraints and multiple objectives.

VII. Acknowledgements

This work was partially funded by the ARC Discovery Grant DP0985723 and by grant N516 384734 from the Polish Ministry of Science and Higher Education (MNiSW).

References

[1] R. Cheng, M. Gen, and Y. Tsujimura, "A tutorial survey of job-shop scheduling problems using genetic algorithms—i: representation," *Comput. Ind. Eng.*, vol. 30, no. 4, pp. 983–997, 1996.

[2] P. J. M. Van Laarhoven, "Job shop scheduling by simulated annealing," *Operations research*, vol. 40, p. 113, 1992.

[3] L. Davis, "Job shop scheduling with genetic algorithms," in *Proceedings of the 1st International Conference on Genetic Algorithms*. Hillsdale, NJ, USA: L. Erlbaum Associates Inc., 1985, pp. 136–140.

[4] K.-H. Liang, X. Yao, C. Newton, and D. Hoffman, "A new evolutionary approach to cutting stock problems with and without contiguity," *Computers & Operations Research*, vol. 29, no. 12, pp. 1641 – 1659, 2002.

[5] J. Levine and F. Ducatelle, "Ant colony optimization and local search for bin packing and cutting stock problems," *The Journal of the Operational Research Society*, vol. 55, no. 7, pp. 705–716, 2004.

[6] Z. Michalewicz, *Genetic algorithms + data structures = evolution programs (3rd, revised and extended ed.)*. New York, NY, USA: Springer-Verlag New York, Inc., 1996.

[7] D. W. Coit and A. E. Smith", "Solving the redundancy allocation problem using a combined neural network/genetic algorithm approach," *Computers & Operations Research*, vol. 23, no. 6, pp. 515 – 526, 1996.

[8] K. Zielinski, P. Weitkemper, R. Laur, and K.-D. Kammeyer, "Parameter study for differential evolution using a power allocation problem including interference cancellation," in *Evolutionary Computation, 2006. CEC 2006. IEEE Congress on*, 2006, pp. 1857–1864.

[9] D. Naso, M. Surico, B. Turchiano, and U. Kaymak, "Genetic algorithms for supply-chain scheduling: A case study in the distribution of ready-mixed concrete," *European Journal of Operational Research*, vol. 177, no. 3, pp. 2069 – 2099, 2007. [Online]. Available: http://www.sciencedirect.com/science/article/B6VCT-4J4B98K-2/2/a297b8cd49d17f23f789c1e29ff13c9a

[10] F. Altiparmak, M. Gen, L. Lin, and T. Paksoy, "A genetic algorithm approach for multi-objective optimization of supply chain networks," *Computers & Industrial Engineering*, vol. 51, no. 1, pp. 196 – 215, 2006, special Issue on Computational Intelligence and Information Technology: Applications to Industrial Engineering. [Online]. Available: http://www.sciencedirect.com/science/article/B6V27-4KPN9S4-1/2/35690ee7a766281941b552296f63309d

[11] G. Zhou, H. Min, and M. Gen, "A genetic algorithm approach to the bi-criteria allocation of customers to warehouses," *International Journal of Production Economics*, vol. 86, no. 1, pp. 35–45, October 2003. [Online]. Available: http://ideas.repec.org/a/eee/proeco/v86y2003i1p35-45.html

[12] C. Y. Lee and J. Y. Choi, "A genetic algorithm for job sequencing problems with distinct due dates and general early-tardy penalty weights," *Computers & Operations Research*, vol. 22, no. 8, pp. 857 – 869, 1995. [Online]. Available: http://www.sciencedirect.com/science/article/B6VC5-3YCMMBN-10/2/b73b269d3d4271724952e0f412dc5d71

[13] H. Lee, J. M. Pinto, I. E. Grossmann, and S. Park, "Mixed-integer linear programming model for refinery short-term scheduling of crude oil unloading with inventory management," *Industrial & Engineering Chemistry Research*, vol. 35, no. 5, pp. 1630–1641, 1996. [Online]. Available: http://pubs.acs.org/doi/abs/10.1021/ie950519h

[14] C. H. Martin, D. C. Dent, and J. C. Eckhart, "Integrated production, distribution, and inventory planning at libbey-owens-ford," *Interfaces*, vol. 23, no. 3, pp. 68–78, 1993.

[15] W. D. Hillis, "Co-evolving parasites improve simulated evolution as an optimization procedure," *Phys. D*, vol. 42, no. 1-3, pp. 228–234, 1990.

[16] C. D. Rosin and R. K. Belew, "Methods for competitive co-evolution: Finding opponents worth beating," in *Proceedings of the 6th International Conference on Genetic Algorithms*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1995, pp. 373–381.

[17] M. Potter, "The design and analysis of a computational model of cooperative coevolution," Ph.D. dissertation, George Mason University, 1997.

[18] D. E. Moriarty and R. Mikkulainen, "Efficient reinforcement learning through symbiotic evolution," *Mach. Learn.*, vol. 22, no. 1-3, pp. 11–32, 1996.

[19] P. Husbands and F. Mill, "Simulated coevolution as the mechanism for emergent planning and scheduling," in *Proceedings of the Fourch International Conference on Genetic Algorithms*, R. Belew and L. Booker, Eds. Morgan Kaufmann, 1991, pp. 264–270.

[20] J. H. Holland, "Properties of the bucket brigade," in *Proceedings of the 1st International Conference on Genetic Algorithms*. Hillsdale, NJ, USA: L. Erlbaum Associates Inc., 1985, pp. 1–7.

[21] M. A. Potter and K. A. D. Jong, "Evolving neural networks with collaborative species," in *Proceedings of the 1995 Summer Computer Simulation Conference*, 1995, pp. 340–345.

[22] ——, "The coevolution of antibodies for concept learning," in *Proceedings from the Fifth Parallel Problem Solving from Nature*. Springer-Verlag, 1998, pp. 530–539.

[23] ——, "A cooperative coevolutionary approach to function optimization," in *Proceedings from the Third Conference on Parallel Problem Solving from Nature*. Springer-Verlag, 1994, pp. 249–257.

[24] R. P. Wiegand, "An analysis of cooperative coevolutionary algorithms," Ph.D. dissertation, George Mason University, Fairfax, VA, USA, 2004, director-Jong, Kenneth A.

[25] E. Popovici and K. De Jong, "Understanding cooperative coevolutionary dynamics via simple fitness landscapes," in *GECCO '05: Proceedings of the 2005 conference on Genetic and evolutionary computation*. New York, NY, USA: ACM, 2005, pp. 507–514.

[26] E. Popovici, "Sequential versus parallel cooperative coevolutionary algorithms for optimization." IEEE Press, 2006.

[27] E. Popovici and K. A. D. Jong, "Relationships between internal and external metrics in co-evolution," in *Congress on Evolutionary Computation*, 2005, pp. 2800–2807.

[28] G. B. Dantzig and J. H. Ramser, "The truck dispatching problem," *Management Science*, vol. 6, no. 1, pp. 80–91, 1959. [Online]. Available: http://dx.doi.org/10.2307/2627477

[29] C. Prins, "A simple and effective evolutionary algorithm for the vehicle routing problem," *Computers & Operations Research*, vol. 31, no. 12, pp. 1985 – 2002, 2004. [Online]. Available: http://www.sciencedirect.com/science/article/B6VC5-48NX3CG-6/2/59c8437348cefff8e0ebfc72bf3f8dba

[30] Y. T. M. Gen and E. Kubota, "Solving job-shop scheduling problem using genetic algorithms." in *16th Int. Conf. on Computer and Industrial Engineering*, 1994, pp. 576–579.

[31] D. E. Goldberg and J. R. Lingle, "Alleles, loci, and the traveling salesman problem," in *First International Conference on Genetic Algorithms and Their Applications*, 1985.

[32] D. E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1989.

[33] P. Machado, J. Tavares, F. B. Pereira, and E. Costa, "Vehicle routing problem: Doing it the evolutionary way," in *In GECCO 2002: Proceedings of the Genetic and Evolutionary Computation Conference*. Morgan Kaufmann Publishers, 2002, p. 690.

[34] J. Beasley, "Route first–cluster second methods for vehicle routing," *Omega*, vol. 11, no. 4, pp. 403–408, 1983.

[35] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, "Introduction to algorithms, second edition," 2001.

[36] L. Davis, "Applying adaptive algorithms to epistatic domains," in *IJCAI'85: Proceedings of the 9th international joint conference on Artificial intelligence*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1985, pp. 162–164.

[37] E. Taillard, "Benchmarks for basic scheduling problems," *European Journal of Operational Research*, vol. 64, no. 2, pp. 278–285, January 1993.

[38] A. M. Christofides, N. and P. Toth, "The vehicle routing problem," *Combinatorial Optimization*, p. 431448, 1979.