Optimization of the Multiple Retailer Supply Chain Management Problem

Caio Soares
Dept. of Comp. Sci. & Soft. Eng.
3137 Shelby Center
Auburn University, AL 36849-5347
(334) 844-6333
soarecv@auburn.edu

Gerry Dozier
Dept. of Computer Science
North Carolina A&T State University
Greensboro, NC 27411
(336) 334-7245 ext. 467
gvdozier@ncat.edu

Emmett Lodree
Dept. of Industrial & Systems Eng.
3312 Shelby Center
Auburn University, AL 36849-5346
(334) 844-1433
lodreej@eng.auburn.edu

Jared Phillips
Dept. of Comp. Sci. & Soft. Eng.
Auburn University, AL 36849-5347
philljr@auburn.edu

Katie Nobles
Dept. of Comp. Sci. & Soft. Eng.
Auburn University, AL 36849-5347
noblekl@auburn.edu

Yong won Park
Dept. of Comp. Sci. & Soft. Eng.
Auburn University, AL 36849-5347
parkyon@auburn.edu

ABSTRACT

With stock surpluses and shortages representing one of the greatest elements of risk to wholesalers, a solution to the multi-retailer supply chain management problem would result in tremendous economic benefits. In this problem, a single wholesaler with multiple retailer customers must find an optimal balance of quantities ordered from suppliers and acceptable lead time costs, while taking into account limiting factors such as the time each retailer will wait for a backorder. The following four evolutionary computations (EC) are utilized to find a solution: evolutionary programming (EP), genetic algorithms (GA), particle swarm optimizers (PSO), and estimation of distribution algorithms (EDA). In addition, problem-specific modifications to each are created. Of the 32 attempted algorithms, the following proved to be best with respect to the client-mandated test-suite: Probabilistic Dual-Topology Full-Model PSO, Star-Topology Full-Model PSO using dynamically-adjusting learning rates, Out-of-the-Box Star-Topology Full-Model PSO, and a Gaussian-based Star-Topology Full-Model PSO with the Constriction Coefficient. A secondary test-suite was also developed to test the effectiveness of the best algorithms on the problem. With respect to the client-mandated and the developed test suite's fitness threshold and maximum number of function evaluations, the best algorithm had an 87% and 90% success rate, respectively. Considering the flexibility and high performance of the solution and the generality of the problem, these results represent a significant contribution to commercial wholesaling.

Categories and Subject Descriptors

D.3.3 [**Artificial Intelligence**]: Problem Solving, Control Methods, and Search.

General Terms

Algorithms and Performance.

1. INTRODUCTION

Regarded as a relatively new field, Evolutionary Computation (EC) finds its earliest manifestations in the 1950's through the work of Friedberg, Bremermann, and Box, among many others. Their EC contributions ranged from work in automatic programming (Friedberg), numerical optimization problems (Bremermann), and industrial analysis and design experiments. Due to considerable skepticism, however, it was not until the mid-1960s that we find the basis for what we today consider EC. During this time, the foundations for EC were laid by Lawrence Fogel (Evolutionary Programming – EP), John Holland (Genetic Algorithms – GA), and the combined work of Bienert, Rechenberg, and Schwefel (Evolutionary Strategies - ES) [5]. Emerging from these three principle roots, a plethora of problem-solving methodologies have entered into the EC canon. Among the newer techniques, Estimation of Distribution Algorithms (EDA) and Particle Swarm Optimizers (PSOs) have proven to be particularly useful. Our research spans the scope of Evolutionary Computation, but this paper focuses its main attention on the best performing ECs specific to this problem.

In the remainder of this section, a high-level introduction to each of the techniques used in this research is provided. Next, Section 2 explains the Multiple Retailer Supply Chain System Problem in detail along with a breakdown of two test suites used to assess each EC. Sections 3 and 4 detail the implementations of the best performing ECs and the results attainted, respectively. A discussion then highlights these results in Section 5, followed by a conclusion in Section 6. Finally, a list of references is provided in Section 7. To represent the broad spectrum of EC, four techniques are chosen in an attempt to solve this problem.

1.1 Evolutionary Programming

Early researchers in Artificial Intelligence were primarily focused on heuristic methods and the simulation of neural networks in an effort to model human intelligence rather than addressing the practical goal of finding working solutions to problems [3]. With the meaning of intelligence under consideration, EPs were devised by Lawrence Fogel in order to evolve finite state machines to predict events on the basis of former observations. Prediction was then viewed as a crucial ingredient to intelligent behavior [6]. In addition, EPs also proved useful as a stochastic optimization technique for discovering the extrema of nonlinear functions.

1.2 Genetic Algorithms

Deriving from cellular automata research, GAs arose as a theoretical interest point rather than solution to a real-world problem. Popularized by John Holland and his colleagues, GAs have traditionally employed crossover operators as their main means of variation [7]. This use of non-uniform mutation provides GA's with search capabilities that are less susceptible to the premature convergence associated with more directed mutation techniques. This very lack of direction, however, often limits GAs in their ability to quickly converge to an optimal solution, which often excludes their use in optimization problems. Perhaps the most well-known EC, the GA is notable due its flexibility and robustness.

1.3 Estimation of Distribution Algorithms

Selecting suitable parameter values for an EC can be an onerous optimization problem in and of itself. Another problem of highly parameterized ECs is the interaction of multiple parameter settings. In an effort to eliminate these problems, EDAs were devised so as to "predict movements of the population in the search space" by calculating the distribution of the previous population [10]. EDAs do not use the operators found in many conventional forms of evolutionary computation, but instead utilize correlation information among variables. In particular, each new generation is sampled from a joint probability distribution that approximates the discrete empirical distribution of selected individuals in the previous population. This process enables EDAs to capture the structure of variables of a problem. For these reasons, EDAs can be a more efficient approach than other ECs for some problems.

1.4 Particle Swarm Optimization

Particle Swarm Optimizers were devised and developed to embody the fundamental concept that social sharing of information provides an evolutionary advantage [9]. In early PSO work, particles explored multi-dimensional, non-linear space using a simple strategy, in which the velocity of a particle depends solely on a cognition and a social factor. Many variations of PSO have been developed to improve performance, and much effort has been focused on finding optimal settings for various algorithm parameters. The effectiveness of a PSO banks upon its use of collective knowledge of the fitness landscape and its ability to quickly explore the search space.

2. THE MULTIPLE RETAILER SUPPLY CHAIN SYSTEM PROBLEM

2.1 Problem Overview

In order for a product to reach the customer, it must go through a supplier, a wholesaler, and a retailer. The Retailer Supply Chain problem arises when a wholesaler tries to efficiently provide for its retailers and their customers while balancing total cost. In the past, the quantities a wholesaler would supply to its retailers were based on educated guessing and trial and error with simulation models. The objective of the multiple retailer supply chain system problem is to minimize the total cost for a wholesaler. Here, this is attempted by using evolutionary computation. The total cost for a wholesaler is calculated as the sum of the ordering costs, supply costs, and holding (or storage) costs taken across all the retailers. These are a function of the following variables:

- Number of retailers – R
- Ordering Cost of a single item – C_o
- Item storage cost – C_H
- Average number of items requested – q_{avg}
- Std. deviation of number of items requested – q_{sd}
- Quantity ordered by each retailer – Q
  (Note that Q_{min} is 0 whereas Q_{max} is the corresponding q_{avg} plus two times the corresponding q_{sd}.)
- Maximum time retailer will wait for back order – M
- Minimum cost associated with back order lead time proposed by supplier – C_{min}
- Maximum cost associated with back order lead time proposed by supplier – C_{max}
- Acceptable lead time of the supplier for particular retailer – L
- Minimum viable lead time of the supplier for retailer – L_{min}
- Maximum viable lead time of the supplier for retailer – L_{max}
- Cost of lost sales in case retailer chooses to no longer do business with supplier – $C_{B,LS}$
- Actual number of items requested by the retailer – x

Although based on total cost, the client-specified fitness function actually weighs the ordering cost more heavily than the other two expenses. In particular, each fitness value is the sum of the total ordering cost (*OCost*) and the average of the sums of holding (*HCost*) and shortage (*SCost*) costs over all retailers R (Eq. 2.1).

$$Fitness = OCost + \frac{1}{R}\sum_{k=1}^{R}(HCost_k + SCost_k) \qquad \text{Eq. 2.1}$$

The ordering cost is simply the product C_0 and total quantity order by all the retailers (Eq. 2.2).

$$OCost = C_o \sum_{j=1}^{n} Q_j \qquad \text{Eq. 2.2}$$

The holding cost is C_H times the excess quantity ordered (Eq 2.3).

$$HCost = C_H \max\left\{\sum_{j=1}^{n} Q_j - \sum_{j=1}^{n} x_j, 0\right\} \qquad \text{Eq. 2.3}$$

The shortage cost is the sum of all *individual* shortage costs associated with each retailer. An *individual* shortage cost is maximum of 0, and $C_{\mathrm{B,LS}}$, which is defined in Eq. 2.4.

$$C_{B,LS} = (x-Q)\cdot\left[\left(\frac{C_{\max}-C_{\min}}{L_{\min}-L_{\max}}\right)\cdot(L-L_{\min})+C_{\max}\right]\cdot\left[1-\frac{L}{M}\cdot(x-Q)\right]+\frac{L\cdot C_{LS}}{M}\cdot(x-Q)^2$$

Eq. 2.4

The supplier distributes excess items among retailers whose order amount exceeds the supplier's initial estimate. These parameters are set to values in accordance with the client-specified test suite in order to find the best performing algorithm. Also, a second test suite is devised to further analyze the best performing algorithms.

2.2 Client-Specified Test Suite

In this primary test suite, the premise of a textbook supplier is employed to determine the test suite parameter values. The fitness threshold has been set at 174400 by the client. Thus, a wholesaler supply configuration, represented as either a chromosome or particle location, is considered a solution when the fitness maps to a value less than 174400. Configurations mapped to smaller values are said to be more fit than those that produce larger values. In this test suite, R = 5, C_o = 75, and C_H = 10. Moreover, Table 2.1 displays the initial values for the remaining variables.

Table 2.1 Client-Specified values

R#	q_{avg}	q_{sd}	*M*	C_{min}	C_{max}	L_{min}	L_{max}	$C_{B,LS}$
1	375	45	14	200	700	2	11	825
2	565	85	5	375	1025	2	4	1477
3	725	75	7	325	985	2	7	1250
4	165	30	10	275	825	2	10	830
5	50	30	7	275	325	2	6	325

2.3 Secondary Test Suite

While the first test suite is applied to all 32 ECs, the secondary test suite is applied only to the four best performing ECs. In contrast, the secondary test suite draws upon the premise of a toothpaste supplier. One of the reasons is that toothpaste is vended by diversely-sized retailers ranging from "Mom and Pop" stores to Super Wal-Marts. The uniform price across brands brought on by the flat demand curve associated with toothpaste allows for reasonably setting the item cost to a single, non-varying value. The retailers in Table 2.2 are matched to the following:

1. Gas station close to supplier
2. Small store close to supplier
3. Grocery store far from supplier
4. Medium retail store close to supplier
5. Large retail store far from supplier

The fitness threshold for this particular test suite is set at 24200000. Don't forget that algorithm configurations mapped to smaller values are said to be more fit than those that produce larger values. In this test suite, $R = 5$, $C_o = 10$, and $C_H = 1$. Also, Table 2.2 displays the initial values for the remaining variables.

Table 2.2 Secondary Test Suite values

R#	q_{avg}	q_{sd}	*M*	C_{min}	C_{max}	L_{min}	L_{max}	$C_{B,LS}$
1	10	6	8	26	35	1	8	40
2	29	15	6	24	50	1	7	116
3	84	34	5	32	65	1	6	336
4	247	74	3	21	88	1	3	988
5	734	147	2	28	80	1	4	2936

3. IMPLEMENTATION

Section 3 begins by outlining the implementation of the ECs employed on this particular problem, followed by a more thorough description of the implementation of PSOs, the top performing EC, and then the implementation of the four best performing algorithms. Note: The list of parameter values mentioned in each subsection is only fully tested on the out-of-the-box algorithms, while each modification tests simply the best parameter set combination from each out-of-the-box algorithm.

3.1 ECs Overview

3.1.1 EP

Four different EP algorithms are implemented in an attempt to solve the Supply-Chain Management Problem:

1. Standard EP (SEP)
2. Continuous Standard EP (CSEP)
3. Meta EP (MEP)
4. Continuous Meta EP (CMEP)

In an effort to improve performance, a modified algorithm for each EP is created. Next, are the characteristics employed in each.

Selection strategy: $(\mu + \lambda)$ for all EPs ($\lambda = 1$ for CSEP and CMEP and $\lambda = \mu$ for SEP and MEP). Also, competition-determined subjective fitness for population size ≥ 10 for SEP and MEP.

Parent selection: random parent selection is used for CSEP and CMEP, while every individual in the population servers as a parent in the remaining two EPs.

Variation: mutation with Gaussian variation is used by all EPs. Moreover, MEP and CMEP utilize strategy parameters η and σ, which are evolved throughout the algorithm, to adapt the amount of mutation to best fit the particular problem.

Parameter Values:

μ: [5, 9, 10, 50, 100]

C: [100, 200, 500, 550, 1000, 2000, 3000, 3500, 4000, 4500, 5000, 6000, 10000, 13000, 20000, 50000, 100000, 200000, 500000, 1000000]

# of competitors: [1, 7, 2/3μ]

η :[0.1, 0.2, 0.3, 0.4]

σ: [10%, 30%, 50%].

The *C* value is used as a divisor to the square root of the parent's fitness in SEP and CSEP because the mutation originally yielded by the square root of the parent's fitness almost always landed outside the allele bounds. Note that the aforementioned parameters do not apply to all EPs, but are used where applicable.

3.1.2 GA

This research showed that GAs can efficiently search for a solution to the Supply Chain Management System Problem. Four different genetic algorithms are implemented:

1. Binary-Coded Elitist Generational GA (BCEG)
2. Real-Coded Elitist Generational GA (RCEG)
3. Binary-Coded Steady State GA (BCSS)
4. Real-Coded Steady State GA (RCSS)

As with the EPs, four modifications are devised in an attempt to achieve superior performance. The characteristics of each GA are as follows.

Selection strategy: Generational Strategy is used with BCEG and RCEG, while always keeping the best found individual (elitism). A Steady State Strategy is used for BCSS and RCSS.

Parent Selection: Binary Tournament selection is used for GAs.

Variation: Uniform Crossover is used for the two Binary-Coded GAs, while BLX-0.5 is used for the Real-Coded algorithms.

Parameter Values:

μ: [5, 10, 20, 50, 100]

BMR: [1%, 3%, 5%, 7%, 10%]

Resolution: 20 bit gene representation (Binary-Coded)

σ: [1%, 3%, 5%, 7%, 9%, 10%, 30%, 50%, 70%, 90%]

3.1.3 EDA

Although numerous EDA variants exist, this research is limited to the following four:

1. Binary-Coded EDA (BCE)
2. Real-Coded EDA (RCE)
3. Elitist Binary-Coded EDA (EBCE)
4. Elitist Real-Coded EDA (ERCE)

Once again, four more variations are created. The main aspects of the four out-of-the-box algorithms are listed below:

Selection strategy: A Generational Strategy is used for all EDAs, while EBCE and ERCE also always keep the best found individual (elitism).

Parent Selection: The number of parents selected (μ) is set at 50% of the population size.

Variation: A probability distribution function is used to create offspring in Binary-Coded EDAs, whereas, a probability density function is used for their Real-Coded counterparts.

Parameter Values: λ: [6, 10, 20, 30, 40, 50, 100]

3.1.4 PSO

This work implements four canonical PSO algorithms:

1. Ring-Topology Full-Model PSO (OOBRT)

2. Star-Topology Full-Model PSO (OOBST)
3. Ring-Topology Full-Model PSO (with Constriction Coefficient) (OOBRTK)
4. Star-Topology Full-Model PSO (with Constriction Coefficient) (OOBSTK)

Each of these satisfies the Full-Model definition by having positive-valued learning rates. More specifically, the learning rates are set to 2.05 to comply with the requirements of Clerc's Constriction Coefficient K [2]. The other influential characteristics are as follows:

Social Influence: Both Ring-Topology algorithms utilize a neighborhood best location in calculating their social component, whereas, both Star-Topology algorithms use the global best location in computing the social component.

Variation: Velocity vectors supply PSOs with variation, and are initialized to 0.

Parameter Values:

μ: [10, 20, 30, 40, 50, 100, 125]

$V_{max} = X_{max}$

$V_{min} = -X_{min}$

3.2 PSO Overview

Each technique made use of standard equations found in [1]. In Eq. 3.1, a particle's velocity V_{id} is updated using the particle's cognition factor P_{id} and social factor P_{gd}. The cognition factor is the best location a particle has ever occupied, whereas the social factor is the best location ever occupied by one of its neighbors. Each component X_{id} of the particle's current location is updated using Eq. 3.2. This update occurs asynchronously; in other words, the neighborhood best or global best location (depending on algorithm) is updated after every one of the x-vectors updates.

$$V_{id} = [V_{id} + \varphi 1 \cdot rand() \cdot (p_{id} - x_{id}) + \varphi 2 \cdot rand() \cdot (p_{gd} - x_{id})] \quad \text{Eq. 3.1}$$

$$x_i = x_i + v_i \quad \text{Eq. 3.2}$$

The learning rate parameters φ_1 and φ_2 are each set to positive values in accordance with the Full-Model specification. The PSOs use various methods for controlling the magnitude of the velocity. Moreover, a v-vector's component is set to 0 when the corresponding x-vector component goes outside one of its boundaries. Finally, all algorithms utilize asynchronous updates. That is, topological information is updated during the course of each iteration, rather than in between iterations.

3.2.1 PSO Ring

As with all PSOs, the ring-based Full-Model PSO searches a fitness landscape using a collection of particles, each of which encapsulates vectors representing its current location, best found location, velocity, and the fitness values corresponding to the two locations [9]. The trajectory of any given particle over time is determined by three factors: the location of its best fitness value, a certain degree of (pseudo) random behavior, and the influence of their neighbors. A neighborhood of a particle P is a predefined grouping of particles containing P, such that P "knows" the best location ever occupied by every particle in the set. Such knowledge can be limited to neighborhoods of a specific size. In each ring-based PSO, these neighborhoods are overlapping, fixed-length arcs of a predefined and static circle formed by the particles (hence the name ring topology). In this implementation, each neighborhood consists of three particles. Each ring-based PSO first initializes its particle population. Recall that each particle *i* contains three vectors x_i, p_i, and v_i that respectively denote the particle's current location, best location, and velocity. Initially, the components of x_i are randomly generated values that lie within problem specific bounds. Vector p_i is set equal to x_i, and v_i is initialized to the zero vector. The current location vectors are then evaluated using the fitness function, and the iterative process begins. During every iteration, each particle changes its location using the update equations (Eq. 3.1 and Eq. 3.2).

3.2.2 PSO Star

A Star Topology PSO differs from a Ring Topology PSO very little. Using a star topology means that every particle is connected to every other particle in such a way that it knows where the best found particle is, no matter how far. This knowledge leads to its second and final difference: how the particle is updated. Again, the updating formula used is very similar to ring topology, with one slight difference, the use of a global best solution, instead of a neighborhood best solution. When computing the social component of the equation, the global best solution is used instead of the neighborhood best as in a ring topology.

3.2.3 Constriction Coefficient (K)

[2] defines a constriction coefficient, K, which is designed to limit the velocity vector's magnitude. The definition of K is provided in Eq. 3.3 and utilized to compute the velocity in Eq. 3.4.

$$K = \frac{2}{\left|2 - \varphi - \sqrt{\varphi^2 - 4\varphi}\right|} \quad \text{Eq. 3.3}$$

$$V_{id} = K[V_{id} + \varphi 1 \cdot rand() \cdot (p_{id} - x_{id}) + \varphi 2 \cdot rand() \cdot (p_{gd} - x_{id})] \quad \text{Eq. 3.4}$$

3.3 Best Performers

Among all algorithms implemented in the client-mandated test suite, the best performing four are three PSO modifications and one out-of-the-box PSO. The implementation of each of these four algorithms is outlined in the following subsections.

3.3.1 Probabilistic Dual-Topology Full-Model PSO (PDT)

[4] concludes that using small neighborhoods prevents premature convergence, whereas the use of global neighborhoods increases convergence speed. Since both of these properties are desirable, PDT incorporates both small and large neighborhoods. In particular, PDT tracks the best locations in three-particle neighborhoods and a global neighborhood. When a particle updates, it has a 100 * λ % chance of using the neighborhood best location p_{id} shown in Eq. 3.1. It has a 1-100 * λ % chance of using the global best individual in place of the neighborhood best. The value of λ decreases with each update iteration through the population according to the equation

$$\lambda = \frac{3}{1 + \ln(t+1)}$$

where *t* is the iteration number initially equal to zero. Thus, this modification encourages a high level of variation initially, but gradually focuses the search as time progresses. In this manner, it continues to explore a large part of the fitness landscape but has faster convergence than with the out-of-the-box Ring-Topology Full-Model PSO.

3.3.2 Star-Topology Full-Model PSO (OOBST)

This is the same Star-Topology Full-Model PSO, described in [1].

3.3.3 Star-Topology Full-Model PSO using dynamically-adjusting learning rates (STLR)

This modification comes along the lines of earlier work done in [1]. It experiments with raising, lowering, and dynamically adjusting the social and cognitive learning rates φ_1 and φ_2 in the x-vector updating equation (refer to Eq. 3.1 and Eq. 3.2). Instead of using φ_1 and φ_2 by themselves, each are added to an adjusting ratio r_i, as seen in Eq. 3.5. Ratio r_i is computed by dividing the current number of function evaluations so far, f, by the maximum number of function evaluations allowed, m, then multiplying that quantity by φ_1 (Eq. 3.6), or by φ_2 (Eq. 3.7).

$$v_i = v_i + (\varphi_1 + r_1) \cdot U(0,1) \cdot (p_i - x_i) + (\varphi_2 + r_2) \cdot U(0,1) \cdot (pg_i - x_i) \quad \text{Eq. 3.5}$$

$$r_1 = (f / m) * \varphi_1 \quad \text{Eq. 3.6}$$

$$r_2 = (f / m) * \varphi_2 \quad \text{Eq. 3.7}$$

Experiments are done with φ_1 and φ_2 ranging from 1.0 to 3.0, as well as subtracting r_i instead of adding it to the corresponding φ. This means that the learning rate starts at a particular value and then either increases as the algorithm progresses, or decreases as the algorithm progresses. Furthermore, this leads the learning rate to either double by the end of the algorithm, or get to 0 by the end of the algorithm. The best results are achieved when $\varphi 1$ and $\varphi 2$ are set between 1.7 and 2.0, with 1.8 along with addition to φ_1 and φ_2, yielding the most successful results.

3.3.4 Gaussian-based Star-Topology Full-Model PSO along with the Constriction Coefficient (GBST)

In GBST, the uniform variates sampled in the velocity equation are replaced by Gaussian variates. The effects of this method are further detailed in Section 5.

4. RESULTS

All 32 algorithms are tested under the client mandated test suite, however, only the four best performing algorithms are also tested under the secondary test suite. For the brevity of this paper, only the results for the best performing algorithms are reported.

4.1 Results of Client-Mandated Test Suite: Textbooks

Among the PSOs considered in this paper, GBST clearly performed the best. In terms of success rate, it surpassed the second best PSO by nearly 7 percentage points as shown in table 4.1. Results also reveal that GBST achieved an average best fitness of 174068, which is 0.1% better than that of the second best performing algorithm. Finally, it required 10% fewer function evaluations than its closest competitor as shown in Table 4.2. These outcomes are explained by GBST's appropriate balance of selection pressure and variation. Because the best location ever found by the PSO serves as a focus for the entire swarm, the star topology guides GBST quickly toward a solution. Its use of the Gaussian distribution in evolving particle velocity, however, provides for an expansive search of the fitness landscape.

Table 4.1: Success Rates

Algorithm	PDT	STLR	OOBST	GBST
Success Rate	70.0%	80.0%	76.7%	86.7%

Table 4.2: Averages and Standard Deviations for function evaluations needed to get below the threshold (174400)

Algorithm	PDT	STLR	OOBST	GBST
Average	330.3	316.1	318.4	257.3
Standard Dev.	137.8	128.4	142.4	102.0

4.2 Results of Second Test Suite: Toothpaste

Again, GBST performed the best. In terms of success rate, it beat the second best PSO by 20 percentage points as shown in Table 4.3. Further results of this test suite also show that GBST achieved an average best fitness of 23806803, about 1.9% less than that of the second best performing algorithm. It also called for 22.9% fewer function evaluations than its closest competitor as shown in table 4.4. The explanation for these statistics mirrors that of the results associated with test suite 1.

Table 4.3: Success Rates

Algorithm	PDT	STLR	OOBST	GBST
Success Rate	16.7%	70.0%	53.3%	90.0%

Table 4.4: Averages and Standard Deviations for function evaluations needed to get below the threshold (24200000)

Algorithm	PDT	STLR	OOBST	GBST
Average	451.3	376.5	402.4	290.1
Standard Dev.	113.3	130.1	107.6	99.1

4.3 Comparison of Results

Fig. 4.1 shows that while the success rates varied in proportional distance, the algorithms performed relative to each other. In both test suites GBST performed the best with STLR, OOBST, and PDT following. This correlation is also seen for the average best fitness and the average number of function evaluations. Fig. 4.2 depicts the association between the average number of function evaluations for the two test suites.

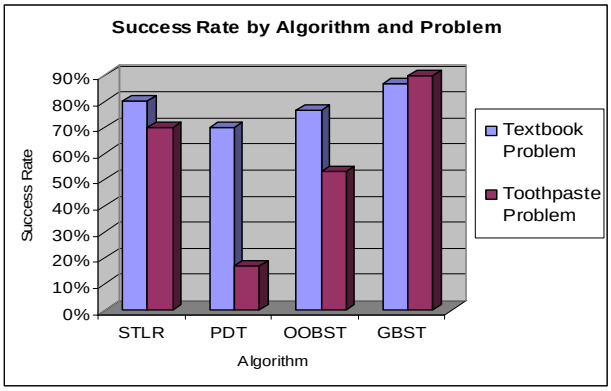

Figure 4.1: Success Rate

There is one case in which the algorithms rank differently and that is on the average best fitness between STLR and OOBST shown in Fig. 4.2. It is speculated that this discrepancy is due to the nature of OOBST and its high risk of "fly-overs" meaning it will bypass any good solution. With a larger search space in the toothpaste test suite, it runs a smaller chance of flying over a good solution.

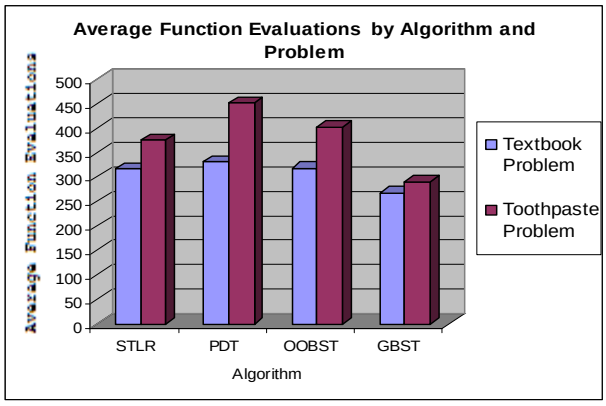

Figure 4.2: Average Function Evaluations (Lower is Better)

5. DISCUSSION

A single, universally-best EC does not exist. This research loudly echoes and supports this sentiment. Several algorithms from each of the EC paradigms performed well on this problem. GAs and EDAs, for example, came within 10 percentage points of the best algorithms. For this particular problem, however, PSOs proved most adept. Their use of social information provided an evolutionary advantage that proved imperative to solution discovery within the context of the supply chain management problem. In Kennedy and Eberhart's original paper on PSOs [9], the authors observed that particles which "overshoot" locations associated with best known fitness values are responsible for the greatest improvements in the swarm's global fitness values. That is, particles with high velocity discover new and better regions of the fitness landscape more often than slow moving particles. These particles, however, also benefit the population by "micro-exploring" known regions for optima. This model of PSO search is so germane to the author's conception of the strategy that they attempted to manually stimulate the evolution of "explorers" and "settlers". This approach seems to lend itself well to the Multiple Retailer Supply Chain Management Problem.

The use of Gaussian variates in the adaptation of particles' velocity vectors provides greater separation among particles. In the canonical swarm [1], the source of randomness is a uniform distribution on the interval [0,1). Thus, from a population perspective, velocities are not biased toward large or small values. In contrast, 66% of Gaussian variates have magnitudes that lie in this interval. These latter variates, however, are biased toward the origin – a characteristic affecting the emergence of "settlers". Given the infinite tails of Gaussian distribution, some velocities are randomized by exceptionally large values. The particles corresponding to such variates comprise the "explorers". This division of particles is appropriate to this problem because of the specific nature of the fitness landscape. While high particle velocity has been associated with poor performance in other contexts [1], it is quite beneficial in cases where optima exist along search space boundaries. Thus, GBST "explorers" quickly delineate particular bounds as "good" regions after which GBST "settlers" locate the best boundary locations. A nexus between these two groups is achieved via the global topology. This global influence, in fact, explains the high performance of each of our best performing algorithms. The benefits of using a global topology have been well documented [1]. The fact that PDT relies only partially on global information explain why this algorithm performed the worst among our best algorithms. STLR performed better than OOBST because its use of dynamically increasing learning rates accelerated particles. As with GBST, this property leads to early boundary exploration. STLR, however, lacks a second category of particle to micro-explore newly discovered regions. A lack of diversity, therefore, lends rationale to the observed inferiority of this algorithm to OOBST.

6. CONCLUSION AND FUTURE WORK

In this study, standard techniques from several EC paradigms as well as problem-tailored modifications are applied to an issue of tremendous economic and theoretic value: The Multiple Retailer Supply Chain Management Problem. Among the ECs considered, it is learned that PSOs provide the best high performance solution for this particular case. Since PSOs are highly parameterized, they provide many opportunities for future research. For example, additional population sizes can be considered. Instead of using Full Models, each PSO can be adapted to make use of the Social, Selfless, and Cognitive models as described in [8]. Different neighborhood size can be utilized. The application of Evolution Strategies to this problem could provide further insights. Finally, the problem itself could be extended to model additional features of the real world that have an impact on commercial wholesaling.

7. REFERENCES

[1] Carlisle, A. and Dozier, G. 2001. An Off-The-Shelf PSO. In Proceedings of the 2001 Workshop on Particle Swarm Optimization, pp. 1-6, Indianapolis, IN.

[2] Clerc, M. and Kennedy, J. 2002. The Particle Swarm – Explosion Stability and Convergence in a Multidimensional Complex Space. IEEE Transactions on Evolutionary Computation, Volume 6, #1.

[3] DeJong, K. & Spears, W. 1993. On the State of Evolutionary Computation. In Proceedings of the Fifth ICGA, 618-623. Kaufmann, San Mateo, CA.

[4] Eberhart, R. and Kennedy J. 1995. A New Optimizer Using Particle Swarm Theory, Proc. Sixth International Symposium on Micro Machine and Human Science (Nagoya, Japan), IEEE Service Center, Piscataway, NJ, 39-43.

[5] Fogel, D. B. 2000. Evolutionary Computation: Basic Algorithms and Operators (1st ed.). Philadelphia Institute of Physics Publishing, Bristol, PA.

[6] Fogel, D. and Chellapilla, K. 1998. Revisiting Evolutionary Programming. SPIE Aerosense98, Applications and Science of Computational Intelligence. pp.2-11, Orlando, FL.

[7] Goldberg, D.E. 1989. Genetic Algorithms in Search, Optimization & Machine Learning.

[8] Kennedy, J. 1997. The Particle Swarm: Social Adaptation of Knowledge. In the Proceedings of the 1997 International Conference on Evolutionary Computation, pp. 303-308, IEEE Press.

[9] Kennedy, J. and Eberhart, R. 1995. Particle Swarm Optimization. In the Proceedings of the 1995 IEEE International Conference on Neural Networks. pp. 1942-1948, IEEE Press.

[10] Lozano, J., Sagarna, R., and Larra~naga, P. 2005. Parallel estimation of distribution algorithms. In P. Larra~naga and J. A. Lozano, editors, Estimation of Distribution Algorithms. A New Tool for Evolutionary Computation.