Applying PSO and DE on Multi–Item Inventory Problem with Supplier Selection

G.S. Piperagkas[1], C. Voglis[1], V.A. Tatsis[1], K.E. Parsopoulos[1], K. Skouri[2]

[1] Department of Computer Science, University of Ioannina, GR–45110 Ioannina, Greece
gpiperag@cs.uoi.gr, voglis@cs.uoi.gr, tatsis.bill@gmail.com, kostasp@cs.uoi.gr

[2] Department of Mathematics, University of Ioannina, GR–45110 Ioannina, Greece
kskouri@uoi.gr

Abstract

We apply Particle Swarm Optimization (PSO) and Differential Evolution (DE) on a multi–item inventory optimization model with supplier selection, limited capacity and defective items. The model assumes an individual cost per transaction as well as a product–dependent holding cost. The goal is the determination of an optimal procurement strategy given the demand over a finite planning horizon. This is formulated as a highly–constrained mixed–integer optimization problem, which is solved using PSO and DE. In addition, the original mixed–integer model is reduced to a real–parameter optimization task and solved with the same algorithms. The results for both approaches are reported and statistically analyzed, offering useful insight regarding the most promising setting.

1 Introduction

Particle Swarm Optimization (PSO) [7] and Differential Evolution (DE) [16] are two of the most popular heuristic optimization algorithms. Their popularity can be attributed to the combination of high efficiency with minor implementation effort, which renders them accessible to researchers in diverse scientific fields. PSO and DE have apparent structural and operational similarities such as the use of an iteratively updated population of potential solutions. The updating process is based on combinations of difference vectors among existing population members. Selection of the best–performing solutions produces the necessary probabilistic pressure for sampling in the most promising regions of the search space.

Operations Research (OR) has offered a rich ground for application of Evolutionary Algorithms (EAs). Numerous works verified the effectiveness of EAs in solving various problems, including scheduling, routing, production planning, management and economic administration etc. Among others, *supplier selection* combined with *inventory management* is a central problem with a remarkable amount of relative work in literature. Supplier selection problems are usually formulated as mixed–integer optimization problems, incorporating purchasing, transportation and inventory costs over multiple periods, under the conditions of multiple sourcing, criteria and constraints. Extensions on lot–sizing with supplier selection for multi–period and multi–product cases have been studied, along with cases with limited capacities on suppliers [1, 5].

In many research works, the products are considered to be of perfect quality. However, in realistic production environments there is often a probability of imperfect quality. It is important for the optimal policy to take into account the relationship between quality imperfection and lot sizing. Such a model was studied in [14], where a probability that production goes out of control was considered. Further models have been considered, incorporating inspection policy [15]. The issue of non–shortages in models with proportional imperfect quality was evoked in [8], where the proportion of imperfect items was assumed to be a random variable. Other models considered the rework of defective products [6], flexible production processes [4], as well as multi–stage lot sizing for imperfect production processes [2].

Recently, a new model was proposed by Rezaei and Davoodi [13]. This model refers to the problem of lot sizing with supplier selection, considering crucial concepts such as imperfect items and limited storage capacity. The problem was formulated as a highly constrained mixed–integer optimization task and it was solved by using two different approaches: a deterministic one using the Lindo software [1] and a

[1] http://www.lindo.com

stochastic one based on Genetic Algorithms (GAs). The latter approach was shown to be very promising and triggered our interest in applying PSO and DE on the specific model, since both algorithms have proved to be very competitive to GAs. Also, the proposed model involves intimately related integer and real–valued variables. In simple words, the integer variables represent the decision of ordering a product from a specific supplier or not, while the real variables specify the exact quantity. However, the decision of not ordering can be equivalently represented solely by assuming a zero value of the corresponding real variable. This way, the model can be simplified by dropping all integer parameters, thereby reducing its dimension. However, the application of PSO and DE on the simplified problem requires some caution, as it will be analyzed in a later section. Finally, the constraints can be handled by using a penalty function approach, which has shown to offer a straightforward solution in constrained optimization cases [12].

In our study, we considered the Unified PSO (UPSO) [11] algorithm as well as the five standard DE operators [16]. UPSO generalizes the standard PSO, producing highly competitive schemes that combine its exploration/exploitation properties as reported in previous works [9, 12]. For both algorithms, initialization in feasible points was not required. The rest of the paper is organized as follows: UPSO and DE are briefly described in Section 2, while the considered optimization models are exposed in Section 3. Experimental setting and results are reported and discussed in Section 4 and the paper concludes in Section 5.

2 Background Information

In the following paragraphs, the UPSO and DE algorithms are briefly described.

2.1 Unified Particle Swarm Optimization

The original PSO algorithm was introduced in 1995 by Eberhart and Kennedy [7]. Its main concept includes a population, called a *swarm*, of potential solutions, called the *particles*, probing the search space. The particles iteratively move in the search space with an adaptable velocity, retaining in a *memory* the best positions they have ever visited, i.e., the positions with the lowest function values.

The exploration capability of PSO is promoted by *information exchange* among particles. More specifically, each particle is assigned a (usually index–dependent) neighborhood. In the *global* PSO variant, also known as *gbest*, the neighborhood of each particle is the whole swarm and the overall best position is the main information–provider for all particles. On the other hand, in the *local* PSO variant, also known as *lbest*, the neighborhoods are strictly smaller usually consisting of a few particles. In such cases, each particle may have its own leader that influences its velocity update. Perhaps the most common neighborhood topology is the *ring*, where each particle assumes as neighbors its mates with neighboring indices [12].

To put it formally, consider the minimization problem:

$$\min_{x \in V \subset \mathbb{R}^n} f(x).$$

Then, a swarm of N particles is a set S of n–dimensional search points, $x_i \in S$, $i = 1, 2, \ldots, N$. The i–th particle has a velocity (position shift), v_i, and retains in memory the best position, $p_i \in S$, it has ever visited. A ring neighborhood of radius m for the particle x_i, implies that the experience of the particles with indices from the set $B_i = \{i - m, \ldots, i, \ldots, i + m\}$ will be available to x_i at each iteration.

Assume that g_i is the index of the best position found so far in the neighborhood of x_i:

$$g_i = \arg\min_{j \in B_i} \{f(p_j)\},$$

and let t denote the iteration counter. Then, according to the *constriction coefficient* version of PSO [3], the swarm is updated as follows:

$$v_{ij}^{(t+1)} = \chi \left[v_{ij}^{(t)} + \varphi_1 \left(p_{ij}^{(t)} - x_{ij}^{(t)} \right) + \varphi_2 \left(p_{g_i j}^{(t)} - x_{ij}^{(t)} \right) \right], \quad (1)$$

$$x_{ij}^{(t+1)} = x_{ij}^{(t)} + v_{ij}^{(t+1)}, \quad (2)$$

where $i = 1, 2, \ldots, N$, and $j = 1, 2, \ldots, n$. The parameter χ is the constriction coefficient and it is used as a means to control the magnitude of the velocities. The other two parameters are defined as $\varphi_1 = c_1\, r_1$ and $\varphi_2 = c_2\, r_2$, where c_1 and c_2 are positive constants, also called the *cognitive* and the *social* parameter, respectively, and r_1, r_2, are random variables uniformly distributed in $[0, 1]$, different for each i, j and t. Based on the stability analysis of Clerc and Kennedy [3], the values, $\chi = 0.729$, $c_1 = c_2 = 2.05$, are considered as the default parameter set. If $x_i^{(t+1)}$ improves the best position $p_i^{(t)}$, it replaces it in $p_i^{(t+1)}$. Otherwise, the best position remains unchanged.

UPSO was proposed as an alternative PSO scheme that combines the different exploration/exploitation properties of the gbest and lbest PSO models [10, 11]. The original UPSO scheme is based on the constriction coefficient PSO variant defined in Eqs. (1) and (2), although it can be straightforwardly defined also for other variants. Putting it formally, let $G_i^{(t+1)}$ and $L_i^{(t+1)}$ denote the velocity update of the i–th particle for the gbest and lbest PSO, respectively:

$$G_{ij}^{(t+1)} = \chi \left[v_{ij}^{(t)} + \varphi_1 \left(p_{ij}^{(t)} - x_{ij}^{(t)} \right) + \varphi_2 \left(p_{gj}^{(t)} - x_{ij}^{(t)} \right) \right], \tag{3}$$

$$L_{ij}^{(t+1)} = \chi \left[v_{ij}^{(t)} + \varphi_1' \left(p_{ij}^{(t)} - x_{ij}^{(t)} \right) + \varphi_2' \left(p_{g_i j}^{(t)} - x_{ij}^{(t)} \right) \right], \tag{4}$$

where t denotes the iteration counter; g is the index of the overall best position; and g_i is the index of the best particle in the neighborhood of x_i. Then, UPSO updates the particle's position according to the scheme [10]:

$$U_{ij}^{(t+1)} = (1 - u)\, L_i^{(t+1)} + u\, G_i^{(t+1)}, \tag{5}$$

$$x_{ij}^{(t+1)} = x_{ij}^{(t)} + U_{ij}^{(t+1)}, \tag{6}$$

where the parameter u is called the *unification factor* and balances the influence (trade–off) of the global and local search directions. Note that the lbest and gbest PSO models can be obtained for $u = 0$ and $u = 1$, respectively.

The standard UPSO scheme was further extended by introducing a stochastic parameter to imitate mutation in EAs. Mutation can help towards the preservation of population diversity, which has a crucial impact on the swarm's exploration capability. Thus, depending on the variant of UPSO under consideration, Eq. (5) can be written either as:

$$U_i^{(t+1)} = (1 - u)\, L_i^{(t+1)} + r_3\, u\, G_i^{(t+1)}, \tag{7}$$

which is mostly based on the lbest PSO or as:

$$U_i^{(t+1)} = r_3\, (1 - u)\, L_i^{(t+1)} + u\, G_i^{(t+1)}, \tag{8}$$

which is mostly based on the gbest PSO. The mutation parameter, r_3, is a normally distributed variable. The convergence properties of these variants were studied in [10] and UPSO's competitiveness was experimentally verified in various problems [12].

2.2 Differential Evolution

The DE algorithm was introduced by Storn and Price [16] as a population–based, stochastic optimization algorithm for numerical optimization problems. DE employs a population, $P = \{x_1, x_2, \ldots, x_N\}$, of individuals to probe the search space. The population is randomly initialized, usually following a uniform distribution over the search space. As in PSO, each individual is an n–dimensional vector, serving as a candidate solution of the corresponding minimization problem. The population is evolved by applying two operators, *mutation* and *recombination*, for producing new candidate solutions. Then, *selection* is performed to construct the new population. The aforementioned operators are applied iteratively until a termination criterion is fulfilled.

The mutation operator produces a new vector, v_i, for each individual, x_i, $i = 1, 2, \ldots, N$, by combining x_i with some of its mates. Different operators have been developed for this task, with the following being the most common ones:

$$\text{DE1}: \quad v_i^{(t+1)} = x_g^{(t)} + F\left(x_{r_1}^{(t)} - x_{r_2}^{(t)}\right), \tag{9}$$

$$\text{DE2}: \quad v_i^{(t+1)} = x_{r_1}^{(t)} + F\left(x_{r_2}^{(t)} - x_{r_3}^{(t)}\right), \tag{10}$$

$$\text{DE3}: \quad v_i^{(t+1)} = x_i^{(t)} + F\left(x_g^{(t)} - x_i^{(t)} + x_{r_1}^{(t)} - x_{r_2}^{(t)}\right), \tag{11}$$

$$\text{DE4}: \quad v_i^{(t+1)} = x_g^{(t)} + F\left(x_{r_1}^{(t)} - x_{r_2}^{(t)} + x_{r_3}^{(t)} - x_{r_4}^{(t)}\right), \tag{12}$$

$$\text{DE5}: \quad v_i^{(t+1)} = x_{r_1}^{(t)} + F\left(x_{r_2}^{(t)} - x_{r_3}^{(t)} + x_{r_4}^{(t)} - x_{r_5}^{(t)}\right), \tag{13}$$

where t denotes the iteration counter; F is a fixed user–defined parameter; g denotes the index of the best individual in the population, i.e., the one with the lowest function value; and $r_i \in \{1, 2, \ldots, N\}$, $i = 1, 2, \ldots, 5$, are mutually different, randomly selected indices that differ also from i. Obviously, in order to be able to apply all DE mutation operators, it must hold that $N > 5$.

After mutation, recombination is applied on the generated vector, v_i, producing a *trial vector*, $u_i = (u_{i1}, u_{i2}, \ldots, u_{in})$, $i = 1, 2, \ldots, N$, as follows:

$$u_{ij}^{(t+1)} = \begin{cases} v_{ij}^{(t+1)}, & \text{if } R_j \leqslant CR \text{ or } j = RI(i), \\ x_{ij}^{(t)}, & \text{if } R_j > CR \text{ and } j \neq RI(i), \end{cases} \tag{14}$$

where $j = 1, 2, \ldots, n$; R_j is the j–th evaluation of a uniform random number generator in the range $[0, 1]$; $CR \in [0, 1]$ is a user–defined crossover constant; and $RI(i)$ is an index randomly selected from the set $\{1, 2, \ldots, n\}$. Finally, the produced trial vector, u_i, is compared against the corresponding individual, x_i, and the best between them is included in the population of the next generation.

3 Problem formulation

In this section, we describe the original model [13] as well as the simplified one, along with the penalty function used in our study. Prior to the descriptions, we shall state the following assumptions [13]:

(1) The transaction cost, o_j, for supplier j is independent of the variety and quantity of the ordered products.

(2) The holding cost, h_i, of product i is product–dependent.

(3) The demand, d_{it}, for product i at time period t is known over a planning horizon.

(4) Items of imperfect quality are kept in stock and sold prior to the next period in a single batch.

(5) Each lot of product i received from supplier j contains a percentage ρ_{ij} of defective items.

(6) Purchasing price of product i from supplier j is b_{ij}. Good quality items are sold in price s_{gi} per unit, while defective items are sold in a single batch at a discounted price, s_{di}.

(7) A screening process of the lot is conducted with a unit screening cost, c_i, for product i.

(8) Each supplier has a limited capacity.

(9) All requirements must be fulfilled in the period in which they occur. Backordering and shortage is not allowed.

(10) Product i needs a storage space, w_i, and the total storage capacity is W.

3.1 Original Model

Following the aforementioned assumptions, Rezaei and Davoodi [13] developed a mathematical model that refers to the scenario of supply chain with multiple products and multiple suppliers, all of which have limited capacity. The demand over a finite planning horizon is also known and an optimal procurement strategy for this multi–period horizon is to be determined.

Each of the products can be sourced from a number of approved suppliers. However, a supplier–dependent transaction cost applies whenever an order is placed. A product–dependent holding cost per period applies for each product in the inventory that is carried across a period in the planning horizon. Also a maximum storage space at each period is considered. In order to maximize the total profit, the decision maker needs to decide what products to order, in what quantities, by which suppliers, and at which time periods. Assuming that i denotes the product, j denotes the supplier and t denotes the time period, the required quantity is denoted as x_{ijt}.

The objective function, henceforth called the *original model*, is defined as follows [13]:

$$\max f\left(x_{ijt}, y_{jt}\right) = \left[\sum_{i}\sum_{j}\sum_{t} x_{ijt}\left(1-\rho_{ij}\right)s_{gi} + \sum_{i}\sum_{j}\sum_{t} x_{ijt}\rho_{ij}s_{di}\right] - \left[\sum_{i}\sum_{j}\sum_{t} x_{ijt}b_{ij}\right.$$

$$\left. + \sum_{j}\sum_{t} o_j y_{jt} + \sum_{i}\sum_{j}\sum_{t} x_{ijt}c_i + \sum_{i}\sum_{t} h_i\left(\sum_{k=1}^{t}\sum_{j} x_{ijk}\left(1-\rho_{ij}\right) - \sum_{k=1}^{t} d_{ik}\right)\right]. \qquad (15)$$

It consists of the sum of the revenues of selling good quality products and imperfect quality products, subtracting the purchase cost of the products, the transaction cost for the suppliers, the screening cost of the products and the holding cost for the remaining inventory at each time period. Obviously, since the problem is defined as maximization, the negative of the objective function defines the corresponding minimization task. The parameters y_{jt} are binary and they are defined as $y_{jt} = 1$ if an order is placed to supplier j at time period t, while $y_{jt} = 0$ otherwise. Also, the problem is highly constrained. More specifically, there are four types of constraints [13]:

Type I: $C_I(i,j,t) = \sum_{k=1}^{t}\sum_{j} x_{ijk}\left(1-\rho_{ij}\right) - \sum_{k=1}^{t} d_{ik} \geqslant 0, \quad$ for all i and t,

Type II: $C_{II}(i,j,t) = \left(\sum_{k=1}^{T} d_{ik}\right) y_{jt} - x_{ijt}\left(1-\rho_{ij}\right) \geqslant 0, \quad$ for all i, j and t,

Type III: $C_{III}(i,j,t) = \sum_{i} w_i\left(\sum_{k=1}^{t}\sum_{j} x_{ijk}\left(1-\rho_{ij}\right) - \sum_{k=1}^{t} d_{ik}\right) \leqslant W, \quad$ for all t,

Type IV: $0 \leqslant x_{ijt} \leqslant k_{ij}, \quad$ for all i, j and t,

where k_{ij} is the capacity of supplier j for product i. The following interpretations can be stated for the four types of constraints [13]:

(1) All requirements must be fulfilled in the period in which they occur.
(2) Backordering and shortage are not allowed (Type I).
(3) All orders are accompanied by the appropriate transaction cost (Type II).
(4) The total storage space is limited by W (Type III).
(5) Suppliers have limited capacities (Type IV).

If I, J and T denote the number of products, suppliers and time periods, respectively, then the number of constraints is equal to $M_c(I,J,T) = (I \times T) + (I \times J \times T) + T + 2 \times (I \times J \times T)$, while the number of variables y_{jt} and x_{ijt} (problem dimension) is equal to $M_v(I,J,T) = (J \times T) + (I \times J \times T)$. Obviously, even for small values of I, J and T, the corresponding problem constitutes a challenging task due to the large number of variables and constraints.

3.2 Simplified Model

The original model can be simplified by eliminating the integer parameters, thereby reducing its dimension. Indeed, based on the definition of the parameters y_{jt}, we can infer a dependence between them and x_{ijt}, as follows:

$$y_{jt} = \begin{cases} 1, & \text{if } x_{ijt} > 0 \text{ for at least one } i, \\ \\ 0, & \text{otherwise.} \end{cases} \tag{16}$$

Thus, the variables y_{jt} are set to 1 if an order is placed on supplier j at time period t, otherwise they are set to 0. If an order is not placed, i.e., $y_{jt} = 0$, then the quantities x_{ijt} of *all* products ordered from supplier j at time period t, shall also be equal to zero. In practice, we consider that x_{ijt} is equal to zero if it is actually smaller than a predefined small threshold, i.e., $0 < x_{ijt} \leqslant \varepsilon_z$.

Therefore, we may drop the integer parameters, y_{jt}, from our optimization problem and retain only x_{ijt}. It is not required to modify the form of the objective function in Eq. (15), as far as we determine the parameters y_{jt} by using Eq. (16) whenever a function evaluation is conducted. We will call this formulation the *simplified model*.

The gain of removing the variables y_{jt} is twofold. On the one hand, the problem's dimension is reduced by $J \times T$. On the other hand, the integer part of the problem is removed, hence it can be tackled as a pure real–valued optimization problem. Nevertheless, for both the original and the simplified model the number of constraints remains unchanged.

3.3 Penalty Function

In our approach, the constraints were handled by using a multi–stage penalty function. Assume that:

$$\tilde{C}_s(i,j,t) = \begin{cases} |C_s(i,j,t)|, & \text{if constraint } \ C_s(i,j,t) \ \text{ is violated,} \\ \\ 0, & \text{otherwise,} \end{cases}$$

where $s \in \{I, II, III\}$, $i = 1, 2, \ldots, I$, $j = 1, 2, \ldots, J$, and $t = 1, 2, \ldots, T$. Also, let P_{pen} be a fixed positive parameter. Then, the penalty function for the minimization problem is defined as follows:

$$\mathcal{F}\left(x_{ijt}, y_{jt}\right) = -f\left(x_{ijt}, y_{jt}\right) + \sum_{i,j,t,s} \tilde{C}_s(i,j,t) P_{\text{pen}}, \tag{17}$$

i.e., a penalty that depends on the degree of violation is added to the objective value for each violated constraint. Usually, in order to avoid false penalization due to approximation errors, a violated constraint is penalized only if its value exceeds a predefined violation tolerance, i.e.:

$$\tilde{C}_s(i,j,t) \ \geqslant \ \varepsilon_c \ > \ 0.$$

Also, we shall note that it is not required to include Type IV constraints in the penalty function, as they simply define the ranges of the variables and they can be explicitly handled by restricting the populations within these box constraints. If an individual violates such a constraint, it is either blocked on the violated limit or bounces back inside the search space.

4 Experimental Setting and Results

We considered the problem instance in [13] with 3 products, 3 suppliers and 4 time periods, i.e., $I = 3$, $J = 3$, and $T = 4$. This implies a mixed–integer original model of dimension $M_v(3,3,4) = 48$, while the corresponding real–valued simplified problem has dimension $M'_v(3,3,4) = 36$. In both cases, there are $M_c(3,3,4) = 124$ constraints from which, 12 are of Type I, 36 are of Type II, 4 are of Type III and the remaining are of Type IV. For the penalty function, the violation tolerance $\varepsilon_c = 10^{-6}$ and the fixed

Alg.	Original model (with integer variables)							Simplified model (without integer variables)						
	Feas.	Mean	StD	Min	Max	Mean-NF	StD-NF	Feas.	Mean	StD	Min	Max	Mean-NF	StD-NF
UPSOℓ	100	15421.64	1914.48	8951.05	20971.63	–	–	100	16759.17	3278.01	7509.59	24417.35	–	–
UPSO.1	100	15212.12	2127.72	10375.01	20151.81	–	–	100	18859.93	2238.09	11760.49	21770.91	–	–
UPSO.1m	100	15668.70	2659.13	10428.47	24292.16	–	–	100	16473.78	2618.19	8485.52	20836.61	–	–
UPSO.5	91	14500.79	2438.09	8607.04	21344.48	9468.36	8710.50	87	16455.72	3125.58	7345.19	23142.99	5351.26	5928.03
UPSO.5m	99	15045.86	2799.24	9358.54	23203.40	243.50	0.00	86	15294.78	2772.99	6973.88	20877.98	8886.35	11535.16
UPSO.9	56	13859.20	2223.46	8478.83	22280.95	85763.78	73177.19	45	15088.82	2780.00	10010.74	20087.92	54858.49	67414.87
UPSO.9m	57	13993.82	1934.43	9927.13	18307.67	92979.01	70864.78	53	14991.10	3460.81	7498.16	23208.42	78242.57	74742.21
UPSOg	45	13415.84	1418.20	9323.64	16255.25	79836.19	72462.18	40	14908.15	2973.21	10283.91	22636.29	78306.92	97846.59
DE1	91	14624.70	4093.28	4924.90	23278.21	86100.71	62665.31	96	18651.37	2720.45	10802.69	23491.00	5751.31	6651.73
DE2	92	8516.30	2349.54	3624.98	15511.66	1978.72	1095.68	96	11995.80	1764.43	6495.59	14921.55	1314.26	378.57
DE3	100	20142.90	1658.85	11280.75	22690.86	–	–	100	20191.47	1307.50	13530.06	22064.48	–	–
DE4	100	12999.85	3619.56	5792.37	22410.83	–	–	100	14869.66	2134.99	8000.83	19298.33	–	–
DE5	3	7233.88	2969.75	5078.29	10621.32	11953.12	6506.82	4	8706.46	1121.83	7967.51	10378.10	14405.69	6870.72

Table 1: Results for all algorithms and problem instances averaged over 100 experiments. Higher values correspond to better solutions.

penalty $P_{\text{pen}} = 10^3$ were used. Also, the parameter $\varepsilon_z = 10^{-6}$ was used to identify a zero component in a solution vector.

Regarding the two algorithms, UPSO was considered for the unification factor values $u = 0.0$ (lbest), 0.1, 0.5, 0.9 and 1.0 (gbest). These choices aimed at investigating the behavior of both lbest–oriented and gbest–oriented UPSO variants. The cases of $u = 0.1$, 0.5 and 0.9, were also considered in their mutated variants defined in Eqs. (7) and (8). Henceforth, we will denote the UPSO variants as UPSOℓ, UPSO.1, UPSO.1m, UPSO.5, UPSO.5m, UPSO.9, UPSO.9m, and UPSOg, respectively, where "m" denotes a variant with mutation. The default PSO parameter values defined in Section 2.1 were adopted in our study, while the Gaussian mutation term $r_3 \sim \mathcal{N}(0, 1)$ was used. Moreover, the five DE variants defined in Eqs. (9)–(13) were considered with the common parameter values $F = 0.5$ and $CR = 0.7$.

Both UPSO and DE are primarily designed to tackle real–valued variables. For this reason, the integer variables in the original problem were assumed to take real values in the range $[0, 1]$ for the swarm/population update, while they were rounded to the nearest integer (either 0 or 1) for the function evaluation. Contrary to this, the simplified model does not require such assumptions since all of its parameters are real–valued.

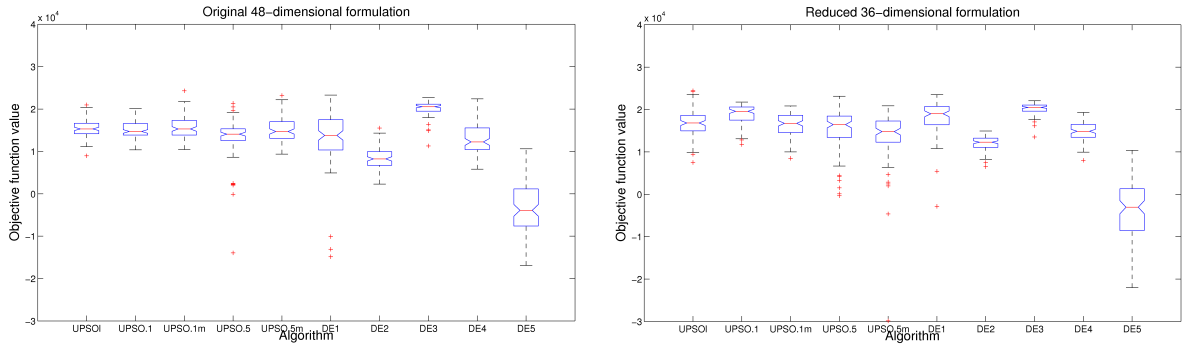
The populations in both algorithms were randomly initialized in the search space. In the original model, uniform initialization within the variables' ranges is adequate. However, the simplified model raises a crucial initialization issue. More specifically, in the original model the probability that an integer parameter is initialized either to 0 or 1 is equal, since the algorithms uniformly sample real numbers within $[0, 1]$. On the other hand, the simplified model samples only within the ranges of the real parameters x_{ijt}, and then computes the corresponding y_{jt} by using Eq. (16) and the relaxation parameter ε_z. Yet, this initialization almost surely assigns values $x_{ijt} > \varepsilon_z$, which correspond to $y_{jt} = 1$, since the volume (Lebesque measure) of the fraction of the search space that corresponds to $x_{ijt} < \varepsilon_z$ for all i (and hence to $y_{jt} = 0$) is very small, compared to the whole search space.

Therefore, performing a completely random initialization in the simplified model would be biased towards the values $y_{jt} = 1$, which correspond to solutions where all suppliers are getting product orders. To alleviate this deficiency, the initialization of the algorithms in the simplified model was conducted as follows:

$$x_{ijt} = \begin{cases} r_{ijt}, & \text{if } R_{ijt} > 0.5, \\ 0, & \text{otherwise}, \end{cases}$$

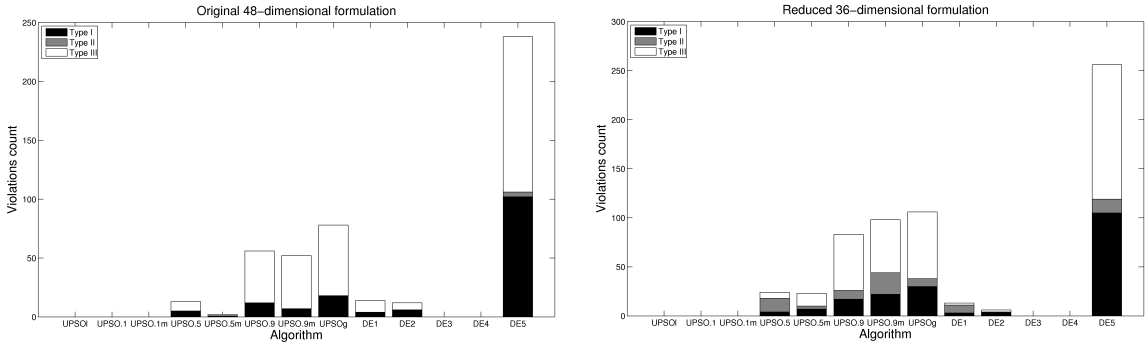
where R_{ijt} is a random value uniformly distributed in $[0, 1]$. Thus, each component of the initial population had equal probability of being zero or non–zero.

The performance of all UPSO and DE variants was tested on both the original and the simplified model. For each algorithm, 100 independent experiments were performed. At each experiment, the algorithm was allowed to perform 10^3 iterations using swarm/population size $N = 50$. The best solution detected throughout each experiment was recorded for each algorithm and problem instance, along with



(a) Original model (b) Simplified model

Figure 1: Boxplots of achieved solution values for the original and the simplified model.



(a) Original model (b) Simplified model

Figure 2: Constraints violation per algorithm for the original and the simplified model.

its feasibility. If a solution was infeasible, the corresponding penalty term was also recorded to reveal the degree of violation per case. Also, the total number of violated constraints over all experiments was recorded per algorithm and problem instance to study the possible tendency of an algorithm towards violation of constraints of specific types. All experiments were conducted using the data provided in [13].

The results were statistically analyzed and they are reported in Table 1 (notice that higher function values correspond to better solutions since the problem is maximization). The first column reports the corresponding algorithm. The next seven columns refer to the results for the original model, while the last seven columns refer to the simplified model. For each model, the first column (Feas.) reports the feasibility percentage of the obtained solutions in 100 experiments. The next four columns expose the mean, standard deviation, minimum and maximum of the obtained solution's value *only for the feasible cases*. For the infeasible cases, the next two columns (Mean-NF and StD-NF) report the mean and standard deviation of the corresponding penalty term.

The best feasible solution obtained with the GA approach in [13] had objective value 15266.8. As we can see in the Table 1, UPSO with small unification factor, mutated or not, provided always feasible solutions. Increasing the unification factor, i.e., moving closer to the gbest PSO model, resulted in a remarkable performance reduction with the penalty terms of the obtained solutions increasing with the value of the unification factor. On the other hand, two variants of the DE algorithm offered always feasible solutions. Both these two approaches, DE3 and DE4, involve the best individual of the population and two difference vectors. The best mean solution value was achieved by DE3 and it was equal to 20142.90. However, the best solution attained ever was achieved by UPSO.1m and it was equal to 24292.16. On the other hand, the worst average performance was achieved by the DE5, which could barely detect feasible solutions.

In the simplified model, the successful UPSO variants further improved their performance. The same held also for the efficient DE approaches. The variants that failed to provide feasible solutions in

Algorithm	UPSOℓ	UPSO.1	UPSO.1m	UPSO.5	UPSO.5m	UPSO.9	UPSO.9m	UPSOg	DE1	DE2	DE3	DE4	DE5
UPSOℓ	$*$	$*$	–	–	$*$	$*$	$*$	$*$	$*$	$*$	$*$	$*$	$*$
UPSO.1	–	$*$	$*$	$*$	$*$	$*$	$*$	$*$	–	$*$	$*$	$*$	$*$
UPSO.1m	–	–	$*$	–	$*$	$*$	$*$	$*$	$*$	$*$	$*$	$*$	$*$
UPSO.5	$*$	$*$	$*$	$*$	$*$	$*$	$*$	$*$	$*$	$*$	$*$	$*$	$*$
UPSO.5m	–	–	–	–	–	$*$	$*$	$*$	$*$	$*$	$*$	–	$*$
UPSO.9	$*$	$*$	$*$	$*$	$*$	–	–	–	$*$	$*$	$*$	$*$	$*$
UPSO.9m	$*$	$*$	$*$	$*$	$*$	–	–	–	$*$	$*$	$*$	$*$	$*$
UPSOg	$*$	$*$	$*$	$*$	$*$	–	–	–	$*$	$*$	$*$	$*$	$*$
DE1	$*$	$*$	$*$	–	–	$*$	$*$	$*$	$*$	$*$	$*$	$*$	$*$
DE2	$*$	$*$	$*$	$*$	$*$	–	–	–	$*$	$*$	$*$	$*$	$*$
DE3	$*$	$*$	$*$	$*$	$*$	$*$	$*$	$*$	$*$	$*$	–	$*$	$*$
DE4	$*$	$*$	$*$	$*$	$*$	$*$	$*$	$*$	–	$*$	$*$	$*$	$*$
DE5	$*$	$*$	$*$	$*$	$*$	$*$	$*$	$*$	$*$	$*$	$*$	$*$	–

Table 2: Wilcoxon rank–sum tests. The light gray area refers to the original model, while the darker gray area refers to the simplified model. The existence of statistical significance is denoted with "*", while the lack is denoted with "–". The main diagonal presents the algorithm's performance in the original model against itself in the simplified model.

the original model, retained their inferior performance also in the simplified model although their penalty terms (violation magnitude) were reduced. Also, we can notice that the improvement of the successful UPSO variants in the simplified model was higher than that of DE3 and DE4, although the latter exhibited smaller standard deviations.

The obtained solution values are also graphically illustrated for the most promising algorithms in Figs. 1(a) and 1(b), while the total number of violated constraints over all the 100 experiments per algorithm and problem instance are illustrated in Figs. 2(a) and 2(b), for the two models, respectively.

Compared to the solutions reported in [13], we can infer that UPSO and DE exhibit higher probability of providing feasible solutions, although, the infeasible solutions of the GA approach in [13] lie closer to the feasibility region, i.e., their violation magnitude is smaller than that of the infeasible solutions obtained by DE and UPSO. Figures 2(a) and 2(b) reveal also that the violation mainly concerns Type III and secondarily Type I constraints. An increase in Type II constraint violation is reported for the simplified model, due to the special interpretation of the variables y_{jt} as functions of x_{ijt}.

In addition to the aforementioned statistics, the performance of each pair of algorithms was tested for statistical significance by using the Wilcoxon rank–sum test. Thus, each pair of algorithms, A and B, was tested against the null hypothesis that the samples of the obtained solution values for A and B have the same median in a 95% level of significance. The results of the tests are reported in Table 2. As can be seen, in most cases there is statistically significant difference between the algorithms. Perhaps the most important observation is the existence of significant differences of the most successful approaches in the original model against themselves in the simplified model. This is also an indication that the simplified model can be advantageous. The only exception to this observation is the DE3 approach, which hardly exhibited any performance difference between the two models.

5 Conclusion

This paper constitutes an experimental investigation of the PSO and DE algorithms on a recently proposed model for supply chain with multiple items and suppliers, where the goal is the determination of an optimal procurement strategy given the demand for a finite planning horizon. In its original formulation, the problem was modeled as a highly–constrained mixed–integer optimization task. Besides the application of the two algorithms on the original model, a simplified model that reduces it to a real–valued optimization task was also proposed and tackled with the same algorithms. The obtained results suggest that the simplified model can be more advantageous for the successful algorithms than the original one. Also, it was shown that UPSO and DE are highly competitive to the GA–based approaches reported in the literature, constituting promising alternative solutions.

References

[1] C. Basnet and J. M. Y. Leung. Inventory lot–sizing with supplier selection. *Computers & Operations Research*, 32(1):1–14, 2005.

[2] M. Ben-Daya. Multi–stage lot sizing models with imperfect processes and inspection errors. *Production Planning and Control*, 10:118–126, 1989.

[3] M. Clerc and J. Kennedy. The particle swarm–explosion, stability, and convergence in a multidimensional complex space. *IEEE Trans. Evol. Comput.*, 6(1):58–73, 2002.

[4] K.-N. Francis Leung. A generalized geometric–programming solution to an economic production quantity model with flexibility and reliability considerations. *European Journal of Operational Research*, 176(1):240–251, 2007.

[5] E. Hassini. Order lot sizing with multiple capacitated suppliers offering leadtime–dependent capacity reservation and unit price discounts. *Production Planning and Control*, 19(2):142–149, 2008.

[6] P. A. Hayek and M. K. Salameh. Production lot sizing with the reworking of imperfect quality items produced. *Production Planning and Control*, 12:584–590, 2001.

[7] J. Kennedy and R. C. Eberhart. Particle swarm optimization. In *Proc. IEEE Int. Conf. Neural Networks*, volume IV, pages 1942–1948, Piscataway, NJ, 1995. IEEE Service Center.

[8] S. Papachristos and I. Konstantaras. Economic ordering quantity models for items with imperfect quality. *International Journal of Production Economics*, 100(1):148–154, 2006.

[9] K. Parsopoulos, K. Skouri, and M. Vrahatis. Particle swarm optimization for tackling continuous review inventory models. In *Lecture Notes in Computer Science (LNCS)*, volume 4974, pages 103–112. Springer, 2008.

[10] K. E. Parsopoulos and M. N. Vrahatis. UPSO: A unified particle swarm optimization scheme. In *Lecture Series on Computer and Computational Sciences, Vol. 1, Proceedings of the International Conference of Computational Methods in Sciences and Engineering (ICCMSE 2004)*, pages 868–873. VSP International Science Publishers, Zeist, The Netherlands, 2004.

[11] K. E. Parsopoulos and M. N. Vrahatis. Parameter selection and adaptation in unified particle swarm optimization. *Mathematical and Computer Modelling*, 46(1–2):198–213, 2007.

[12] K. E. Parsopoulos and M. N. Vrahatis. *Particle Swarm Optimization and Intelligence: Advances and Applications*. Information Science Publishing (IGI Global), 2010.

[13] J. Rezaei and M. Davoodi. A deterministic, multi–item inventory model with supplier selection and imperfect quality. *Applied Mathematical Modelling*, 32(10):2106–2116, 2008.

[14] M. J. Rosenblat and H. L. Lee. Economic production cycles with imperfect production processes. *IIE Transactions*, 18:48–55, 1986.

[15] M. K. Salameh and M. Y. Jaber. Economic production quantity model for items with imperfect quality. *International Journal of Production Economics*, 64(1–3):59–64, 2000.

[16] R. Storn and K. Price. Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces. *J. Global Optimization*, 11:341–359, 1997.