Solving Inventory Management Problems Using Metaheuristic Optimization Algorithms

Grigorios Piperagkas

MASTER THESIS

— ♦ —

Ioannina, June 2012

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ
ΠΑΝΕΠΙΣΤΗΜΙΟ ΙΩΑΝΝΙΝΩΝ

DEPARTMENT OF COMPUTER SCIENCE
UNIVERSITY OF IOANNINA

Επίλυση Προβλημάτων Διαχείρισης Αποθεμάτων με Χρήση Μεταευρετικών Αλγορίθμων Βελτιστοποίησης

Η ΜΕΤΑΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ ΕΞΕΙΔΙΚΕΥΣΗΣ

υποβάλλεται στην

ορισθείσα από την Γενική Συνέλευση Ειδικής Σύνθεσης

του Τμήματος Πληροφορικής Εξεταστική Επιτροπή

από τον

Γρηγόριο Πιπεράγκα

ως μέρος των Υποχρεώσεων για τη λήψη του

ΜΕΤΑΠΤΥΧΙΑΚΟΥ ΔΙΠΛΩΜΑΤΟΣ ΣΤΗΝ ΠΛΗΡΟΦΟΡΙΚΗ

ΜΕ ΕΞΕΙΔΙΚΕΥΣΗ

ΣΤΟΥΣ ΕΠΙΣΤΗΜΟΝΙΚΟΥΣ ΥΠΟΛΟΓΙΣΜΟΥΣ

Ιούνιος 2012

ΕΥΧΑΡΙΣΤΙΕΣ

Με την ολοκλήρωση της παρούσας εργασίας θα ήθελα να ευχαριστήσω ιδιαίτερα τον επιβλέποντά μου κ. Κωνσταντίνο Παρσόπουλο για τη στήριξη και την πολύτιμη βοήθειά του στην εκπόνησή της αλλά και σε ολόκληρη την πορεία των μεταπτυχιακών μου σπουδών.

Θα ήθελα επίσης να ευχαριστήσω τον κ. Ισαάκ Λαγαρή και την κ. Κωνσταντίνα Σκούρη, καθώς και τον κ. Ιωάννη Κωνσταντάρα για τις πολύτιμες συμβουλές τους και τις εποικοδομητικές συζητήσεις μας.

Επίσης ευχαριστώ τους κ. Κωνσταντίνο Βόγκλη, Δημήτριο Παπαγεωργίου και Αριστείδη Λύκα για τις εποικοδομητικές συζητήσεις και τη συνεργασία μας, καθώς και τον κ. Χρυσόστομο Στύλιο για τη συνεργασία και τη μερική χρηματική υποστήριξη των μεταπτυχιακών μου σπουδών.

Τέλος, να ευχαριστήσω την οικογένειά μου για την ηθική και υλική στήριξη της και τους καλούς μου φίλους για τις ωραίες στιγμές.

CONTENTS

List of Figures

List of Tables

Abstract

Piperagkas, Grigorios, Department of Computer Science,
University of Ioannina, Greece, June 2012,
Solving Inventory Management Problems Using Metaheuristic Optimization Algorithms
Thesis Supervisor: Konstantinos E. Parsopoulos

We investigate the solution of inventory management problems through metaheuristic optimization algorithms. More specifically, we focus on multi-item inventory problems with limited capacity, as well as on the dynamic lot-sizing problem with stochastic demand. Three quite popular optimization algorithms were selected, namely Particle Swarm Optimization (PSO), Differential Evolution (DE) and Harmony Search (HS).

Initially the multi-item inventory problem with supplier selection, limited capacity and defective items is presented. The model includes specific transaction and storage cost. The objective is the determination of a replenishment policy, given the demand over a planning horizon, while the problem is defined from a set of constraints and is formed as a mixed-integer optimization problem. The PSO and DE algorithms are used for solving the model, with appropriate modifications and assumptions for the specific model. At the next stage, the model is simplified in order to obtain the solution easier. Results are presented and analyzed statistically.

At the second part of the thesis, we examine the dynamic lot sizing problem under stochastic and non-stationary demand over a predefined finite planning horizon. The model is based on the dynamic Wagner-Whitin model, which was proposed in 1958 and was recently extended with stochastic demand. Here the solution is examined with three algorithms, namely PSO, DE and HS. It is the first attempt to solve this kind of problem with metaheuristic optimization algorithms. The methods are modified appropriately to accomodate the model' s specificities. Their efficiency is reported regarding the time and solution quality, for various problem instances. The algorithms performance is statistically analyzed along with their proper modifications. The results support the claim that they can be successfully considered as an alternative for the solution of the specific problems.

Περιληψη

Γρηγόριος Σ. Πιπεράγκας
Τμήμα Πληροφορικής, Πανεπιστήμιο Ιωαννίνων, Ιούνιος 2012.
Επίλυση Προβλημάτων Διαχείρισης Αποθεμάτων με Χρήση Μεταευρετικών Αλγορίθμων Βελτιστοποίησης.
Επιβλέπων: Κωνσταντίνος Ε. Παρσόπουλος

Στην παρούσα εργασία μελετάται η επίλυση προβλημάτων διαχείρισης αποθεμάτων με χρήση μεταευρετικών αλγορίθμων βελτιστοποίησης. Συγκεκριμένα, επικεντρώσαμε σε προβλήματα με πολλούς προμηθευτές και προϊόντα με περιορισμένο αποθηκευτικό χώρο, καθώς και στο δυναμικό πρόβλημα με στοχαστική ζήτηση. Επιλέχθηκαν τρείς ιδιαίτερα δημοφιλείς αλγόριθμοι για την επίλυσή τους, οι Particle Swarm Optimization (PSO), Differential Evolution (DE) και Harmony Search (HS).

Αρχικά παρουσιάζεται το πρόβλημα διαχείρισης αποθεμάτων με πολλά προϊόντα και πολλούς προμηθευτές, περιορισμένη αποθηκευτικό χώρο και ελαττωματικά προϊόντα. Το μοντέλο περιλαμβάνει συγκεκριμένο κόστος συναλλαγής αλλά και κόστος αποθήκευσης. Στόχος είναι ο καθορισμός του μεγέθους παραγγελιών, δεδομένης της ζήτησης για ένα συγκεκριμένο χρονικό ορίζοντα, ενώ το πρόβλημα καθορίζεται από μεγάλο πλήθος περιορισμών και διαμορφώνεται ως πρόβλημα μεικτού ακέραιου προγραμματισμού. Για την επίλυση χρησιμοποιούνται οι αλγόριθμοι PSO και DE, με κατάλληλες τροποποιήσεις και εκδοχές για το συγκεκριμένο είδος μοντέλου. Σε επόμενο στάδιο, το μοντέλο απλοποιείται για ευκολότερη επίλυση. Τα αποτελέσματα παρουσιάζονται και αναλύονται στατιστικά.

Στο δεύτερο μέρος της εργασίας, εξετάζεται ένα σύστημα προγραμματισμού και ελέγχου αποθεμάτων με ένα προϊόν και πεπερασμένο ορίζοντα σχεδιασμού, με στοχαστική μη στάσιμη ζήτηση. Το μοντέλο είναι βασισμένο στο δυναμικό μοντέλο των Wagner-Whitin που προτάθηκε το 1958 και επεκτάθηκε πρόσφατα για στοχαστική ζήτηση. Εδώ η επίλυση γίνεται με τρείς αλγορίθμους, τις μεθόδους PSO, DE και HS. Πρόκειται για την πρώτη απόπειρα επίλυσης του προβλήματος αυτού με μεταευρετικούς αλγόριθμους βελτιστοποίησης. Οι μέθοδοι τροποποιούνται κατάλληλα για να προσαρμοστούν στις απαιτήσεις του μοντέλου και καταγράφεται η επίδοσή τους, ως προς το χρόνο και την ποιότητα λύσης, για διαφορετικές εκδοχές και συνθήκες του προβλήματος. Η απόδοση των αλγορίθμων αναλύεται στατιστικά. Τα αποτελέσματα υποστηρίζουν την πεποίθηση ότι οι συγκεκριμένοι αλγόριθμοι μπορούν να θεωρηθούν ως εναλλακτικές μέθοδοι για την επίλυση αυτών των προβλημάτων.

CHAPTER 1

INTRODUCTION

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 [6], production planning, management and economic administration [5].

In the current thesis, three well studied algorithms have been employed to solve two interesting inventory management problems: *Particle Swarm Optimization (PSO), Differential Evolution (DE) and Harmony Search (HS)*. PSO [23], DE [37] and HS [15] are three of the most popular metaheuristic 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. Their development was inspired by the evolution and self–organization properties of living entities. PSO roughly resembles the swarming behavior observed in bird–flocks or fish schools, which is also intimately related to physical laws that characterize more fundamental systems such as gases in Particle Physics. On the other hand, DE is closely associated with evolutionary algorithms, resembling recombination and mutation procedures.

HS performs a procedure similar to the musical improvisation process, although its structure and operation have many commons with state–of–the–art evolutionary algorithms such as Evolution Strategies.

1.1 Multi-item inventory model with supplier selection

Among others, *supplier selection* combined with *inventory management* is a central problem with a remarkable amount of relative work in business management literature. Sup-

plier 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 [2, 18].

In many research works, the products are considered to be of perfect quality. However, in realistic production environments there is often a (usually small) 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 [35], where a probability that production goes out of control was considered. Further models have been considered, incorporating inspection policy [36]. The issue of non–shortages in models with proportional imperfect quality was evoked in [27], where the proportion of imperfect items was assumed to be a random variable. Other models considered the rework of defective products [19], flexible production processes [11], as well as multi–stage lot sizing for imperfect production processes [3].

Recently, a new model was proposed by Rezaei and Davoodi [34]. 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 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 [31].

In our study, we considered the Unified PSO (UPSO) [30] algorithm as well as the five standard DE operators [37]. UPSO generalizes the standard PSO, producing highly competitive schemes that combine its exploration/exploitation properties as reported in previous works [28, 31]. For both algorithms, initialization in feasible points was not required.

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

1.2 The stochastic dynamic lot–sizing problem

The *dynamic lot–size (DLS)* problem consists of determining the quantity of products to order or produce in each time period over a finite discrete planning horizon, in order to satisfy the demand for each period while minimizing the summation of setup and inventory holding costs. This model was first introduced by Wagner and Whitin in 1958 [43], who developed an $O(H^2)$ forward algorithm for a general dynamic version of the uncapacitated economic lot–sizing model, where H stands for the number of time periods.

DLS is embedded within many practical production planning problems. The zero–inventory ordering principle, which imposes that no production is undertaken if inventory is available, constitutes a key contribution for the uncapacitated cases. Zangwill [44] extended the Wagner–Whitin model by allowing complete backlogging of unsatisfied demand. This was the first work to highlight the importance of using networks to represent some production planning problems. More specifically, the problem was represented as a minimum cost flow problem in a network with concave arcs costs and a single source.

Although many alternative algorithms have been proposed, the dynamic programming method still remains the major analytical tool for solving lot–sizing problems. Federgruen and Tzur [10] presented a simple forward algorithm which solves the general dynamic lot–sizing model in $O(H \log H)$ time or in $O(H)$ under some assumptions on the cost data. This was the key improvement over the previously recommended well–known shortest path algorithm solution, which required $O(H^2)$ time.

Wagelmans et al. [42] extended the range of allowable cost data to permit coefficients with unrestricted signs. They developed an alternative algorithm to solve the resulting problem in $O(H \log H)$ time. Aggarwal and Park [1] developed an algorithm with complexity $O(H \log H)$, which solved the problem of H periods by solving two sub–problems of $H/2$ periods. Two recent review papers on the dynamic lot–sizing problem are those by Karimi et al. [21] and Jans and Degraeve [20]. The first one reviews single–level lot–sizing problems, their variants and solution approaches. The second one presents an overview of recent developments on the deterministic dynamic lot–sizing problem, focusing on the modeling of various industrial extensions rather than solution approaches.

All the above models assume that relevant data, such as the demand, are known and deterministic. However, this assumption is unrealistic in many situations. Guan and Miller [17] studied the stochastic version of the deterministic lot–sizing problem and proposed a polynomial time algorithm to obtain the optimal solution. Guan [16] studied a more general setting of the stochastic lot–sizing problem, assuming varying capacities and backlogging of unsatisfied demand.

Recently, Vargas [41] investigated the problem of planning dynamic order quantities, extending the Wagner–Whitin algorithm to the case of stochastic, time–varying demand with known density function. Safety stock requirements were implicitly included in planned order quantities whereby the objective was to minimize the sum of expected setup, backorder and inventory holding costs. A particularly elegant solution procedure was developed for the case of normally distributed periodic demands. In his analysis,

Vargas [41] stated that his work:

"...may serve as a basis for the development of improved production scheduling heuristics for the stochastic case and new heuristics can be directly incorporated in production scheduling systems."

Motivated by this proposal, we selected three metaheuristic optimization algorithms, still unstudied on the specific problem. The algorithms were selected on the basis of popularity, number of interdisciplinary applications, easy implementation and verified efficiency, which imply a prevailing position among similar approaches. (PSO) [23], DE [37] and HS [15] are three algorithms that perfectly matched our criteria. Although they were not the only candidate approaches, the ongoing interest of the research community on their properties and applications was the motive for our choice in the present study.

The aforementioned algorithms have been successfully applied also in other OR problems. For instance, we can refer to flow shop and machine scheduling problems [26, 29, 39], optimal scheduling of multiple dam systems and fluid–transport network design [13, 14], inventory optimization [28], dynamic lot–sizing problems [12] etc.

1.3 Thesis organization

The rest of this thesis is organized as follows: in Chapter 2, the three employed algorithms are presented and analyzed thoroughly, as well as their variants that were used in each model, in Chapter 3, the multi–item inventory model with supplier selection and the stochastic dynamic lot-sizing model with an introduction to the Wagner-whitin algorithm are presented. In Chapter 4 we present the experimental settings and all the results for both problems and all the instances that were examined. Finally, an epilogue is presented in Chapter 5.

CHAPTER 2

EMPLOYED ALGORITHMS

The main features of the employed algorithms, namely PSO, DE and HS, are presented. In our presentation, we assume that the optimization problem under consideration is defined as:

$$\min_{x \in X \subset \mathbb{R}^n} f(x), \tag{2.1}$$

where X is the search space. No additional assumptions on the objective function $f(x)$ are required, except of the availability of its value at any given point in X.

2.1 Particle Swarm Optimization

We present the PSO algorithm in its standard formation, as well as the recently proposed unified scheme.

2.1.1 Main scheme

The original PSO algorithm was introduced in 1995 by Eberhart and Kennedy [23]. The method utilizes a *swarm*, i.e., a population of search points that iteratively probe the search space for the global minimizer. The search points are called *particles*, and they concurrently move with an adaptable velocity (position shift) to new positions.

Moreover, each particle has a *memory* where it retains the best position it has ever visited, i.e., the position with the lowest function value. This can be considered as experience storage for the particle, which exploits this information to guide its search towards the most promising regions of the search space. The search stops as soon as a stopping criterion is achieved, usually related to the quality of the best solution found so far or to the number of function evaluations spent by the algorithm.

Apart from the personal memory, each particle has a neighborhood, i.e., a set of indices of other particles that share their memories (best positions) with it. Thus, the particle decides on its next move by aggregating its own discoveries with the best findings of its neighboring mates.

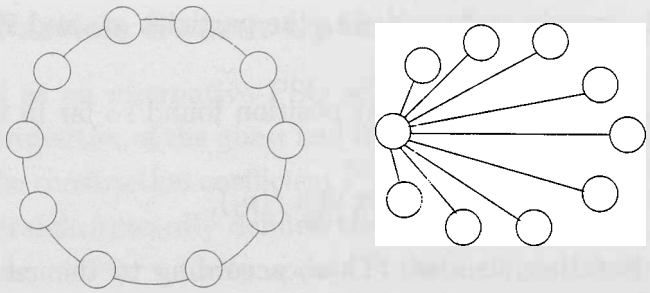

Figure 2.1: The ring (left) and star (right) neighborhood topologies of PSO.

Based on the concept of neighborhood, two main PSO schemes were proposed. The first one is called the *global* PSO (also known as *gbest*) model on account of the underlying global information–sharing scheme. According to this, all particles assume the whole swarm as their neighborhood and each particle takes into consideration its own memory as well as the overall best memory, i.e., the best position ever discovered by the whole swarm.

The second model is the *local* PSO (also known as *lbest*), where each particle is assigned a neighborhood usually consisting of a few particles. The organization of particles in neighborhoods rises the concept of *neighborhood topologies*, which refers to abstract representations of information–flow channels among the particles. Usually, the topologies are represented as graphs consisting of nodes (particles) and interconnections (communication channels) among them. The most common topology is the *ring*, where all particles are assumed to lie on a ring ordered according to their indices, such that each particle has two immediate neighbors with adjacent indices. Then, for a given particle, its neighborhood is completely defined by determining a *radius*, i.e., the number of particles with adjacent indices that constitute the neighborhood. More information on the effect of neighborhoods can be gained in [22, 38]. Figure 2.1 illustrates the aforementioned ring topology (left) as well as another popular scheme called *star* (right).

In general, the search procedure of heuristic algorithms such as PSO consists of two phases, namely *exploration* and *exploitation*. In the first, the swarm attempts to detect the most promising regions of the search space. In the latter, it promotes the faster convergence to the most promising regions detected so far. It has been experimentally verified that the neighborhood topology can influence the swarm's convergence dynamic. As can be easily inferred, the gbest model promotes the search around the overall best position in favor of exploitation. On the other hand, the lbest model with its regional and gradual information transmission between particles, leans effectively to exploration.

To put it formally, let:

$$S = \{x_1, x_2, \ldots, x_N\},$$

be a swarm of N particles, $x_i \in X \subset \mathbb{R}^n$, $i = 1, 2, \ldots, N$. The i–th particle has a velocity (position shift), v_i, and retains in memory the best position, $p_i \in X$, it has ever visited. A ring neighborhood of radius m for the particle x_i, is defined as the set of indices:

$$B_i = \{i - m, i - m + 1, \ldots, i, \ldots, i + m - 1, i + m\}.$$

The ring is assumed to recycle at its end, i.e., the particles x_N and x_2 are the immediate neighbors of x_1.

Assume that g_i is the index of the best position found so far in the neighborhood of x_i, i.e.,

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

and let t denote the iteration counter. Then, according to the *constriction coefficient* version of PSO [8], the swarm is updated as follows:

$$v_{ij}^{(t+1)} = \chi \left[v_{ij}^{(t)} + \varphi_1 \left(p_{ij}^{(t)} - x_{ij}^{(t)} \right) + \varphi_2 \left(p_{g_i j}^{(t)} - x_{ij}^{(t)} \right) \right], \tag{2.2}$$

$$x_{ij}^{(t+1)} = x_{ij}^{(t)} + v_{ij}^{(t+1)}, \tag{2.3}$$

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:

$$\varphi_1 = c_1 r_1, \qquad \varphi_2 = c_2 r_2,$$

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 $[0, 1]$, assuming different values for each i, j and t.

The constriction coefficient is needed to restrict the magnitude of the velocities, promoting convergence and alleviating the "swarm explosion" effect that has been shown to be detrimental for the search procedure [8]. In early PSO versions, the parameters were empirically determined based on trial runs. In more recent versions, the PSO stability analyses by Clerc and Kennedy [8] and Trelea [40] imply that parameters are selected such that the following equation holds:

$$\chi = \frac{2}{|2 - \varphi - \sqrt{\varphi^2 - 4\varphi}|} \tag{2.4}$$

where $\varphi > 4$ and $\varphi = c_1 + c_2$. Following this result, the values $\chi = 0.729$, $c_1 = c_2 = 2.05$, are considered as the default parameter set.

A full iteration of PSO is completed with the best positions update:

$$p_i^{(t+1)} = \begin{cases} x_i^{(t+1)}, & \text{if } f\left(x_i^{(t+1)}\right) < f\left(p_i^{(t)}\right), \\ p_i^{(t)}, & \text{otherwise.} \end{cases}$$

PSO was primarily designed to operate in continuous search spaces. However, experimental evidence have shown signs of efficiency also in integer and mixed–integer problems, without the need of radical modifications in the algorithm [29, 24]. The most straightforward approach to achieve this, is the use of its standard form with the particles been rounded to the closest integer prior to each function evaluation, while their position updates are performed in the continuous domain. This is the approach adopted also in the present paper. A typical example is provided in a later section.

2.1.2 Unified Particle Swarm Optimization

UPSO was proposed as an alternative PSO scheme that combines the different exploration/exploitation properties of the gbest and lbest PSO models [30]. The original UPSO scheme is based on the constriction coefficient PSO variant defined in Eqs. (2.2) and (2.3), 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:

$$G_{ij}^{(t+1)} = \chi \left[v_{ij}^{(t)} + \varphi_1 \left(p_{ij}^{(t)} - x_{ij}^{(t)} \right) + \varphi_2 \left(p_{gj}^{(t)} - x_{ij}^{(t)} \right) \right], \tag{2.5}$$

$$L_{ij}^{(t+1)} = \chi \left[v_{ij}^{(t)} + \varphi_1' \left(p_{ij}^{(t)} - x_{ij}^{(t)} \right) + \varphi_2' \left(p_{g_i j}^{(t)} - x_{ij}^{(t)} \right) \right], \tag{2.6}$$

where t denotes the iteration counter; g is the index of the overall best particle in swarm; 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 [30]:

$$U_{ij}^{(t+1)} = (1-u)\, L_i^{(t+1)} + u\, G_i^{(t+1)} \tag{2.7}$$

$$x_{ij}^{(t+1)} = x_{ij}^{(t)} + U_{ij}^{(t+1)}, \tag{2.8}$$

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 swarm's exploration capability. Thus, depending on the variant of UPSO under consideration, Eq. (2.7) can be written either as:

$$U_i^{(t+1)} = (1-u)\, L_i^{(t+1)} + r_3\, u\, G_i^{(t+1)}, \tag{2.9}$$

which is mostly based on the lbest PSO, or as:

$$U_i^{(t+1)} = r_3\, (1-u)\, L_i^{(t+1)} + u\, G_i^{(t+1)}, \tag{2.10}$$

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 [30] and the superiority of UPSO against the standard PSO was experimentally verified in various problems [31].

2.2 Differential Evolution

The DE algorithm was introduced by Storn and Price [37] as a population–based, stochastic optimization algorithm for numerical optimization problems. Similarly to PSO, DE employs a population, $S = \{x_1, x_2, \ldots, x_N\}$, of individuals to probe the search space.

The population is randomly initialized, usually following a uniform distribution over the search space. At each iteration, N competitions are held to determine the members of the population for the next iteration. This is achieved by iteratively applying three operations on each individual: *mutation*, *crossover* and *selection*.

The mutation operator produces a new vector, v_i, for each individual, x_i, $i = 1, 2, \ldots, N$, by combining it with some of the rest. Different operators have been proposed for this task. The following five operators are among the most common DE mutation schemes:

$$\text{DE1}: \quad v_i^{(t+1)} = x_g^{(t)} + F\left(x_{r_1}^{(t)} - x_{r_2}^{(t)}\right), \tag{2.11}$$

$$\text{DE2}: \quad v_i^{(t+1)} = x_{r_1}^{(t)} + F\left(x_{r_2}^{(t)} - x_{r_3}^{(t)}\right), \tag{2.12}$$

$$\text{DE3}: \quad v_i^{(t+1)} = x_i^{(t)} + F\left(x_g^{(t)} - x_i^{(t)} + x_{r_1}^{(t)} - x_{r_2}^{(t)}\right), \tag{2.13}$$

$$\text{DE4}: \quad v_i^{(t+1)} = x_g^{(t)} + F\left(x_{r_1}^{(t)} - x_{r_2}^{(t)} + x_{r_3}^{(t)} - x_{r_4}^{(t)}\right), \tag{2.14}$$

$$\text{DE5}: \quad v_i^{(t+1)} = x_{r_1}^{(t)} + F\left(x_{r_2}^{(t)} - x_{r_3}^{(t)} + x_{r_4}^{(t)} - x_{r_5}^{(t)}\right), \tag{2.15}$$

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.:

$$g = \arg \min_{j=1,\ldots,N} f(x_j),$$

and $r_i \in \{1, 2, \ldots, N\}$, $i = 1, 2, \ldots, 5$, are mutually different, randomly selected indices that differ also from the index i. Obviously, in order to enable all mutation operators, it must hold that $N > 5$. All vector operations in Eqs. (2.11)–(2.15), are performed componentwise.

After mutation, the recombination operator is applied on the generated vector, v_i, producing a *trial vector*:

$$u_i = \left(u_{i1}, u_{i2}, \ldots, u_{in}\right),$$

which is defined as follows:

$$u_{ij}^{(t+1)} = \begin{cases} v_{ij}^{(t+1)}, & \text{if } R_j \leqslant CR \text{ or } j = RI(i), \\ x_{ij}^{(t)}, & \text{otherwise,} \end{cases} \tag{2.16}$$

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 and the best between them is included in the population of the next generation, i.e.:

$$x_i^{(t+1)} = \begin{cases} u_i^{(t+1)}, & \text{if } f\left(u_i^{(t+1)}\right) < f\left(x_i^{(t)}\right), \\ x_i^{(t)}, & \text{otherwise.} \end{cases} \tag{2.17}$$

Apparently, DE does not require a separate memory as PSO, since it operates directly on the best solutions found so far (the corresponding best positions in PSO). This renders

DE a greedier algorithm than PSO. Also, there is no sound theoretical evidence on the proper parameter setting of the algorithm. Several different settings have been used in the literature [9] but their performance appears to be strongly dependent on the operator and problem at hand. Nevertheless, by their nature, the mutation operators that involve the best individual of the population are expected to be more exploitation–oriented than those that use randomly selected individuals.

2.3 Harmony search

HS was proposed by Geem, Kim and Longanthan in 2001 [15]. Inspiration was drawn from musical performance processes that occur when a musician searches for a better state of harmony, improvising the instrument pitches towards a better aesthetic outcome. In a similar manner, the algorithm seeks for the global optimum, maintaining and iteratively updating a memory, called the *Harmony Memory* (HM), of candidate solutions (harmonies). HM contains the best harmonies considered so far, i.e., candidate solutions with the smallest objective function values.

HM consists of N randomly initialized vectors (memory size is user–defined) along with their function values [25]:

$$HM = \left[\begin{array}{ccc|c} x_{11} & \cdots & x_{1n} & f(x_1) \\ x_{21} & \cdots & x_{2n} & f(x_2) \\ \vdots & \ddots & \vdots & \vdots \\ x_{N1} & \cdots & x_{Nn} & f(x_N) \end{array}\right] \tag{2.18}$$

The harmonies in HM are ordered in ascending order with respect to their values, i.e.:

$$f(x_1) \leqslant f(x_2) \leqslant \cdots \leqslant f(x_N).$$

Also, two additional parameters must be defined. The first one is the *Harmony Memory Considering Rate* (*HMCR*), which stands for the probability of selecting a vector component value among those already stored in HM. The second parameter is called the *Pitch Adjusting Rate* (*PAR*) and it defines the mutation probability of a selected value from HM. The role of these parameters is clarified below.

The algorithm works iteratively, exploiting the stored information in HM for the production of one or many new solutions at each iteration. The new solutions are built component by component, selecting at each step either a stored component or a random value. More specifically, let $x = (x_1, x_2, \ldots, x_n)^\top$ be a new solution to be built, with components:

$$x_j \in X_j, \qquad j = 1, 2, \ldots, n,$$

where $X_j \subset \mathbb{R}$ is the subspace of the search space X that corresponds to the j–th component. Then, x_j is probabilistically selected according to the scheme:

$$x_j = \begin{cases} x_{sj} \in \{x_{1j}, x_{2j}, \ldots, x_{Nj}\}, & \text{if } r_j \leqslant HMCR, \\ y \in X_j, & \text{otherwise,} \end{cases} \tag{2.19}$$

where r_j is a random variable uniformly distributed in $[0, 1]$; s is an index selected from the set $\{1, 2, \ldots, N\}$; and y is a random value in X_j.

The selection of s is probabilistic and can be either uniform over the set of indices or it can be a linear ranking selection scheme [7] of the stored harmonies in HM. In the latter, high selective pressure can impose strong selection bias towards the indices of the best solutions, in contrast to uniform selection which assigns equal selection probabilities to all indices. Consequently, uniform selection is expected to be more diversity–preserving in the produced harmonies.

After the construction of the new solution, $x = (x_1, x_2, \ldots, x_n)^\top$, each component is mutated as follows:

$$x_j = \begin{cases} x_j + q\,w, & \text{if } R_j \leqslant PAR, \\ x_j, & \text{otherwise,} \end{cases} \tag{2.20}$$

where R_j is a uniformly distributed random variable in $[0, 1]$; q is a uniformly distributed random variable in $[-1, 1]$; and w is a user–defined mutation magnitude.

The aforementioned operations can be repeated to produce a number, M_{prod}, of new harmonies. Each new harmony is evaluated with the objective function, and the best $M_{\text{rep}} \leqslant M_{\text{prod}}$ of them replace the worst M_{rep} harmonies stored in HM, if they improve them. Both parameters M_{rep} and M_{prod} are user defined. In the simplest case, $M_{\text{rep}} = M_{\text{prod}} = 1$ is used.

In contrast to PSO and DE, the HS operators with proper parameter setting can directly handle integer and mixed–integer problems without any modification. Also, we shall notice that HS has shown many structural similarities with the established Evolution Strategies (ES) approaches [4]. In fact, HS can be considered as an alternative ES variant, although with a different motivation and inspiration source. In view of this similarities, it is anticipated that the performance of HS can provide evidence also on the performance of standard ES variants.

2.4 Synopsis

We have thoroughly presented the three main algorithms used to solve the inventory management problems, namely PSO, DE and HS, as well as their variants that were used. In the following chapter the models that were solved are presented and analyzed.

CHAPTER 3

THE INVESTIGATED PROBLEMS

We present the model of multi-item inventory problem with supllier selection, limited capacity and defective items, with known demand over a planning horizon, along with the model of dynamic lot-sizing with stochastic demand.

3.1 Multi-item inventory problem with supplier selection: problem formulation

In this chapter, we describe the original model [34] 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 [34]:

(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.

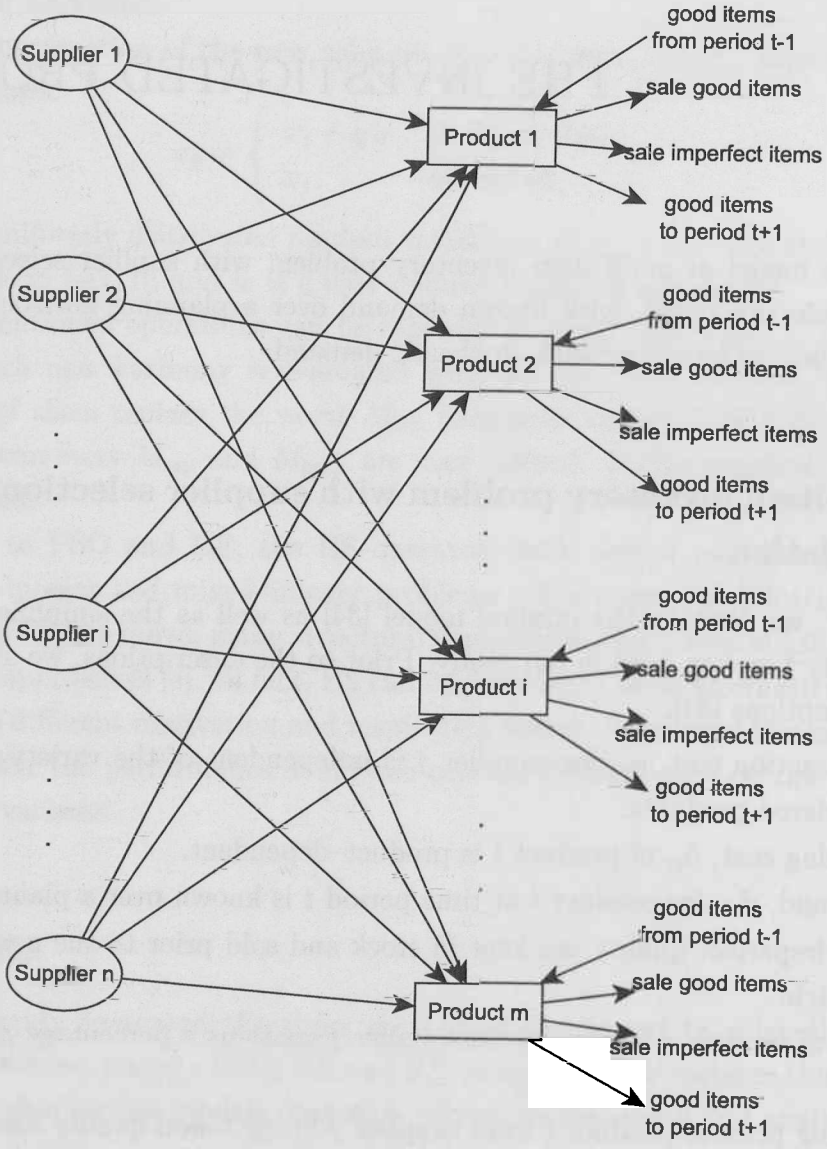

Figure 3.1: Diagram of the proposed inventory model

3.1.1 Original model

Following the above assumptions, Rezaei and Davoodi [34] 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 is defined as follows and henceforth it will be called the *original model* [34]:

$$\begin{aligned}\max f\left(x_{ijt}, y_{jt}\right) = &\left[\sum_i \sum_j \sum_t x_{ijt}\left(1-\rho_{ij}\right) s_{gi} + \sum_i \sum_j \sum_t x_{ijt}\rho_{ij} s_{di}\right] \\ &- \left[\sum_i \sum_j \sum_t x_{ijt} b_{ij} + \sum_j \sum_t o_j y_{jt} + \sum_i \sum_j \sum_t x_{ijt} c_i + \right. \\ &\left. + \sum_i \sum_t h_i \left(\sum_{k=1}^{t} \sum_j x_{ijk}\left(1-\rho_{ij}\right) - \sum_{k=1}^{t} d_{ik}\right)\right] \end{aligned} \tag{3.1}$$

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, otherwise $y_{jt} = 0$. Also, the problem is highly constrained. More specifically, there are four types of constraints [34]:

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

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

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

Type IV: $0 \leqslant x_{ijt} \leqslant k_{ij}$, 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 [34]:

(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.

3.1.2 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:

$$y_{jt} = \begin{cases} 1, & \text{if } x_{ijt} > 0 \text{ for at least one } i, \\ 0, & \text{otherwise.} \end{cases} \tag{3.2}$$

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. (3.9), as far as we determine the parameters y_{jt} by using Eq. (3.2) 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.

3.1.3 Penalty function

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

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

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:

$$\mathcal{F}(x_{ijt}, y_{jt}) = -f(x_{ijt}, y_{jt}) + \sum_{i,j,t,s} \tilde{C}_s(i,j,t) P_{\text{pen}}, \tag{3.3}$$

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.

3.2 The stochastic dynamic lot–sizing problem

We present the Wagner-Whitin model that was proposed in 1958, along with the recently proposed model that incorporates stochastic demand over the planning horizon.

3.2.1 The Wagner–Whitin model

The formula for an economic lot size under the assumption of a steady state demand is well studied and the calculation is based on the balance between the cost of holding inventory and placing an order. However, it is becoming more complicated when the demand in each period is known but with different value.

The mathematical model proposed by Wagner and Whitin [43] is an "one–way feasibillity" problem, meaning that an order can be placed in period t for period $t+i$ but not vice versa.

Mathematical model

We assume that, similarly to the standard lot size formulation, the buying or manufacturing costs and the selling price of an item are constant for all time periods and only the costs of the inventory management are varying. For period $t = 1, 2, \ldots, N$, we define the following quantities:

d_t : amount demanded

i_t : interest charge per unit of inventory carried forward to period $t+1$

s_t : setup cost

x_t : amount ordered or manufactured

It is obvious that all demands and costs are non–negative. The aim is to find a sequence of orders such that all demands are met at a minimum total cost. Such a sequence, unique or not, is called *optimal.*

A detrimental method to solve the problem is by exhaustive search on the sequences of possible solutions, the 2^{N-1} combinations of ordering or not in each period (for N total periods), assuming that we place an order in the first period.

Let K the inventory entering a period and K_0 the initial inventory. Therefore, for period t the inventory follows:

$$K = K_0 + \sum_{i=1}^{t-1} x_i - \sum_{i=1}^{t-1} d_i \geqslant 0. \tag{3.4}$$

The minimal cost policy then can be expressed as:

$$g_t(K) = \min_{(K+x_t \geqslant d_t)} \{i_{t-1}K + u(x_t)s_t + g_{t+1}(K + x_t - d_t)\}, \tag{3.5}$$

where,

$$u(x_t) = \begin{cases} 0, & \text{if } x_t = 0, \\ 1, & \text{if } x_t > 0. \end{cases} \tag{3.6}$$

For the last period N we have

$$g_N(K) = \min_{(K+x_t \geq d_t)} \{i_{N-1}K + u(x_N)s_N\}, \tag{3.7}$$

Computation of g_t starts at $t = N$, as a function of K, which is specified for period 1. Thus g_1 can be finally derived.

Complementary to the main equations of the model, the following theorems are presented (for each, the proof is stated extensively in [43]):

THEOREM 3.1. *There exists an optimal sequence such that* $Kx_t = 0 \forall t$ *(where* K *is the inventory entering period* t*).*

THEOREM 3.2. *There exists an optimal sequence such that for all* t, $x_t = 0$ *or* $x_t = \sum_{i=t}^{k} d_i$ *for some* k *and* $t \leq k \leq N$

THEOREM 3.3. *There exists an optimal sequence such that if a demand* $d_{t'}$ *is satisfied by some* $x_{t''}$, $t'' < t'$, *then* $d_t, t = t'' + 1, \ldots, t' - 1$ *is also satisfied by* $x_{t''}$.

THEOREM 3.4. *If* $K = 0$ *for a period* t, *it is optimal to solve the problem for periods* 1 *to* $t - 1$ *without considering the following periods.*

THEOREM 3.5. *If for a period* t' *the minimum of equation 3.5 occurs for* $i = t'' \leq t'$ *then in periods* $t > t'$ *we can only consider* $t'' \leq i \leq t$. *In the specific case where* $t' = t''$ *it is sufficient to consider only sequences with* $x_{t'} > 0$.

The Wagner–Whitin algorithm

As stated in [43], the algorithm steps can be listed as follows, for a period, $t' = 1, 2, \ldots, N$:

Step 1 Consider the policies of ordering at period t'', $t'' = 1, 2, \ldots t'$ satisfying the demands d_t, $t = t'', t'' + 1, \ldots, t'$, with this order.

Step 2 Determine the total cost of these t' different policies by adding the ordering and holding costs associated with placing an order at period t'', and the cost of acting optimally for periods 1 through $t'' - 1$. This cost has been determined previously in computations for the periods $t = 1, 2, \ldots, t' - 1$

Step 3 From all the t' alternatives, select the minimum cost policy for periods 1 through t' considered independently.

Step 4 Proceed to period $t' + 1$ or stop if $t' = N$

3.2.2 Stochastic model: problem formulation

The mathematical description of the stochastic lot–size problem is deployed in the following paragraphs, closely following the presentation of Vargas [41].

Assumptions and notation

The formal representation of the problem requires the following assumptions for the stochastic version of the Wagner–Whitin lot–size problem:

(1) The planning horizon is composed of H time periods.

(2) Demand in each period, t, is non–negative, independent and stochastic with known density.

(3) The production capacity is unlimited.

(4) Unsatisfied demand is fully backlogged and a backlogging cost is assessed at the end of each period per unit backlogged.

(5) A fixed lead time, L, is assumed and no disposal of inventory is allowed.

(6) At the conclusion of the horizon, holding or backlogging costs are assessed and any backlogged demand is left unfilled.

(7) An order must be placed in the first period, which arrives at the start of this period, and there are no pipeline lots.

Additional notation used in the problem formulation, follows below:

d_t: demand in period $t = 1, 2, \ldots, H$, with known density function, $g_t(x)$.

h_t: holding cost assessed at the end of period t per unit.

b_t: backlogging cost assessed at the end of period t per unit, assumed to be proportional to the holding cost, i.e., $b_t = p\,h_t$.

A_t: fixed production set up for each time period.

S_t: cumulative sum of all production lots to arrive up to and including period t (initially $S_0 = 0$).

$k(x)$: binary decision variable defined as:

$$k(x) = \begin{cases} 1, & x \neq 0, \\ 0, & x = 0. \end{cases}$$

Now we can give the general mathematical formulation of the problem, in the following section.

Mathematical model and objective function

The expected cost incurred in period $t = 1, 2, \ldots, H$, is given by the following expression:

$$\begin{aligned} E_t\,(S_1, S_2, ..., S_t) \;=\; & A_{t-L}\,k(S_t - S_{t-1}) \\ & + h_t \int_0^{S_t} (S_t - q)\,f_t(q)\,dq \\ & + b_t \int_{S_t}^{\infty} (q - S_t)\,f_t(q)\,dq, \end{aligned} \tag{3.8}$$

where $f_t(q)$ is the convolution of demand for periods $1 - L$ to t. The corresponding optimization problem is to specify the cumulative production amounts, S_t, $t = 1, 2, \ldots, H$, that produce the minimum total expected set–up, holding and backlogging costs [41], i.e.:

$$\min_{0 < S_1 \leqslant S_2 \leqslant \cdots \leqslant S_H} \sum_{t=1}^{H} E_t\,(S_1, S_2, ..., S_t)\,. \tag{3.9}$$

Actually, the problem is solved in two stages. At first the optimal replenishment quantities for any sequence of epochs is determined and, afterwards, the optimal sequence of replenishment epochs is identified.

More specifically, if S is the cumulative production quantity received up to period i, then the expected cost incurred in periods $i, i+1, \ldots, j-1$, is computed as:

$$\begin{aligned} K(S, i, j) \;=\; & A_{i-L} + \sum_{t=i}^{j-1} h_t \int_0^{S} (S - q)\,f_t(q)\,dq \\ & + \sum_{t=i}^{j-1} b_t \int_{S}^{\infty} (q - S)\,f_t(q)\,dq, \end{aligned} \tag{3.10}$$

where i and j are periods in which a production lot is available, with:

$$1 \leqslant i < j \leqslant H+1,$$

while no replenishment occurs for $j = H+1$.

As stated in [41], Eq. (3.10) is differentiable and convex in S so that we obtain the optimal solution by setting its derivative equal to zero. Consequently, if:

$$F_t(x) = \int_0^x f_t(q)\, dq,$$

then,

$$\sum_{t=1}^{j-1}(h_t + b_t)\, F_t(S) \;=\; \sum_{k=i}^{j-1} b_t,$$

which can be extended as:

$$\sum_{t=i}^{j-1}\left\{\frac{(h_t+b_t)}{\sum_{k=i}^{j-1}(h_k+b_k)}F_t(S)\right\} = \frac{\sum_{t=i}^{j-1} b_t}{\sum_{k=i}^{j-1}(h_t+b_t)} = \frac{p}{(1+p)}. \tag{3.11}$$

The first optimization stage is completed by computing the root, $S^*(i,j)$, of Eq. (3.11), which represents the optimal cumulative production amount to be received up to period i while no subsequent production lot is received before period j, and p is the parameter for which, $b_t = p\, h_t$, holds.

The second optimization stage consists of the computation of the optimal sequence of replenishment epochs, i.e., the sequence that gives the minimum value of the objective function defined in Eq. (3.9). This can be done by using different techniques and it constitutes the point of our interference in the solution process with the heuristic algorithms.

Exhaustive search is the most trivial algorithm for solving the problem and may even be effective in small problem instances. However, it becomes exponentially laborious with the number of time periods, since the number of all possible sequences becomes very large, requiring prohibitive computation time for their assessment. Nevertheless, it is still used by many practitioners for small problem instances.

Vargas [41] proposed a sophisticated technique that builds a tree–like structure and translates the problem to an equivalent one of finding the shortest path of the resulting acyclic network. The corresponding arcs of the network represent the options of replenishment occurring in period i with no subsequent replenishment until period j, and they are labeled with non–negative arc costs:

$$C_{ij} = K\Big(S^*(i,j), i, j\Big), \qquad 1 \leqslant i < j \leqslant H+1. \tag{3.12}$$

Further analysis on this technique can be found in [41].

LEMMA 3.1. *For any $i < j < k$, it holds that $S^*(j,k) \geqslant S^*(i,j)$.*

The proof of Lemma 3.1 can be found in [41]. The specific lemma presents that for any sequence of replenishment epochs, the cumulative production amounts are non-decreasing, meaning that the individual lot sizes are non-negative.

In the current work, we propose the solution of the second optimization stage by using the heuristic optimization algorithms mentioned in the previous chapter.

3.3 Synopsis

We have presented the two problems that were investigated: the multi-item inventory problem with supplier selection, defective items and limited capacity, along with the dynamic lot–sizing problem with stochastic demand. In the next chapter we present the experimental settings and the results for both problems.

CHAPTER 4

EXPERIMENTAL SETTINGS AND RESULTS

In the following sections we present the experimental settings for the two models and the corresponding results and analysis.

4.1 Settings for the multi-item inventory model with supplier selection

In our experimental setting, we considered the problem instance defined in [34] with 3 products, 3 suppliers and 4 time periods. Thus, following the notation used in the previous sections, we have:

$$I = 3, \qquad J = 3, \qquad T = 4.$$

The demands and the prices of the products considered in our experiments are reported in Tables 4.1 and 4.2, respectively. The dimension of the corresponding mixed–integer original model, as defined in Eq. (3.9), is equal to $M_v(3,3,4) = 48$. The corresponding-real–valued simplified problem has dimension $M'_v(3,3,4) = 36$. In both cases, the number of constraints is equal to $M_c(3,3,4) = 124$. In this number there are included 12 constraints of Type I, 36 of Type II, 4 of Type III, and the remaining are of Type IV. Regarding the penalty function, the parameters:

$$\varepsilon_c = 10^{-6}, \qquad P_{\text{pen}} = 10^3,$$

were used as violation tolerance and fixed penalty, respectively. Also, the parameter $\varepsilon_z = 10^{-6}$ was used to identify a zero component in a solution vector.

Regarding the employed algorithms, different parameter settings and operators were considered. Specifically, 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. (2.9) and (2.10). Henceforth, we will denote the UPSO variants as UPSOℓ, UPSO.1, UPSO.1m, UPSO.5,

Table 4.1: Demand of the three products over a planning horizon of four periods.

	Planning horizon (four periods)			
Products	1	2	3	4
1	170	155	160	140
2	85	90	80	105
3	280	255	290	300

Table 4.2: Prices of the three products supplied by the three suppliers and the corresponding transaction costs.

	Price (supplied from three suppliers)		
Products	1	2	3
1	25	27	24
2	30	32	33
3	54	50	49
Transaction cost	1000	900	1500

Table 4.3: Parameter sets for UPSO.

	UPSO Parameter Sets		
	A	B	C
χ	0.729	0.6	0.721
c_1, c_2	2.05	2.83	1.654

Table 4.4: Parameter sets for DE.

	DE Parameter Sets								
	A	B	C	D	E	F	G	H	I
F	0.3	0.3	0.3	0.5	0.5	0.5	0.7	0.7	0.7
CR	0.3	0.5	0.7	0.3	0.5	0.7	0.3	0.5	0.7

UPSO.5m, UPSO.9, UPSO.9m, and UPSOg, respectively, where "m" denotes a variant with mutation. All mutated variants assumed a normally distributed mutation term,

$r_3 \sim \mathcal{N}(0,1)$. Regarding the parameters χ, c_1 and c_2, the three parameter sets defined in Eq. (2.4) were tested with all the aforementioned UPSO variants.

Regarding DE, the five basic operators defined in Eqs. (2.11)–(2.15) were considered. Due to their sensitivity on the parameters F and CR, we tested all possible combinations with $F, CR \in \{0.3, 0.5, 0.7\}$. The parameter settings for both algorithms are summarized in Tables 4.3 and 4.4. The specific parameter set that is used with an algorithm will be denoted with a corresponding superscript. For example, UPSO.1$^{[B]}$ denotes UPSO with unification factor $u = 0.1$ and the parameter set B, i.e., $\chi = 0.6$, $c_1 = c_2 = 2.83$, while DE3$^{[H]}$ denotes the DE3 operator using the parameter set H, i.e., $F = 0.7$, $CR = 0.5$.

Both UPSO and DE were originally 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 the independent decision parameters are real–valued.

The populations in both algorithms were randomly initialized in the search space, based on the variables' ranges reported in [34]. In the original model, uniform initialization within the ranges is adequate. However, the simplified model raises a crucial initialization issue. Specifically, in the original model an integer parameter has equal probability of being initialized either to 0 or 1, 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. (3.2) 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 $y_{jt} = 0$) is very small, compared to the whole search space.

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

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

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

4.2 Results and discussion

The performance of all UPSO and DE variants under all parameter sets, 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

Table 4.5: Results for the original (48–dimensional) model.

Par. Set	Algorithm	Suc.	Mean	St.D.	Min	Max	Mean–NF	StD–NF
A	**UPSOℓ**	**100**	**15421.64**	**1914.48**	**8951.05**	**20971.63**	–	–
	UPSO.1	**100**	**15212.12**	**2127.72**	**10375.01**	**20151.81**	–	–
	UPSO.1m	**100**	**15668.70**	**2659.13**	**10428.47**	**24292.16**	–	–
	UPSO.5	91	14500.79	2438.09	8607.04	21344.48	9468.36	8710.50
	UPSO.5m	99	15045.86	2799.24	9358.54	23203.40	243.50	0.00
	UPSO.9	56	13859.20	2223.46	8478.83	22280.95	85763.78	73177.19
	UPSO.9m	57	13993.82	1934.43	9927.13	18307.67	92979.01	70864.78
	UPSOg	45	13415.84	1418.20	9323.64	16255.25	79836.19	72462.18
B	**UPSOℓ**	**100**	**16038.85**	**2304.36**	**11812.53**	**22502.33**	–	–
	UPSO.1	**100**	**16532.74**	**2448.75**	**12141.43**	**24239.41**	–	–
	UPSO.1m	**100**	**15803.60**	**2294.03**	**11113.09**	**22856.12**	–	–
	UPSO.5	93	14503.87	2787.31	9234.30	24330.26	67315.30	78975.54
	UPSO.5m	**100**	**14840.61**	**2564.40**	**10085.19**	**22944.63**	–	–
	UPSO.9	67	14431.56	2133.04	9961.11	21332.90	81346.06	74831.47
	UPSO.9m	50	14918.88	2000.13	10641.39	20022.54	96290.43	73952.45
	UPSOg	46	14020.47	1873.16	10122.79	18485.73	107048.28	72468.41
C	**UPSOℓ**	**100**	**15921.00**	**2231.32**	**10566.48**	**24651.15**	–	–
	UPSO.1	**100**	**15451.92**	**2324.91**	**11211.14**	**20934.17**	–	–
	UPSO.1m	**100**	**16074.32**	**2818.37**	**10819.31**	**23621.70**	–	–
	UPSO.5	91	13597.54	2302.49	9018.64	19260.53	23579.52	55674.28
	UPSO.5m	**100**	**14953.93**	**3005.48**	**8621.36**	**22315.34**	–	–
	UPSO.9	70	13698.67	2211.16	7510.87	19705.86	82698.38	79657.03
	UPSO.9m	57	14085.24	2091.52	8457.35	18474.54	58001.63	63954.17
	UPSOg	62	13906.82	2309.39	9486.54	20520.18	57271.92	66923.11
A	DE1	75	14559.15	2542.46	9594.67	24221.82	50181.80	58648.24
	DE2	95	12608.52	1941.84	6755.38	15970.42	2544.27	1511.81
	DE3	**100**	**16397.80**	**2336.46**	**12002.27**	**23445.69**	–	–
	DE4	86	15115.88	2496.64	7882.80	22409.93	58598.11	69066.23
	DE5	60	10385.70	2528.90	5754.32	15056.00	3814.12	2402.44
B	DE1	53	13963.80	2266.26	10670.87	20347.21	92550.63	76565.95
	DE2	95	13659.89	1745.42	6669.91	17712.97	14713.47	6589.16
	DE3	**100**	**16488.09**	**2782.94**	**10939.02**	**24328.46**	–	–
	DE4	79	15109.43	2789.85	8940.14	22365.54	115492.95	63699.07
	DE5	85	12030.76	2329.54	5351.02	18058.41	2083.88	2790.76
C	DE1	8	12970.13	1324.98	10904.66	14656.34	131506.73	95750.99
	DE2	94	14366.92	1545.68	10937.93	18795.41	10366.81	6519.84
	DE3	**100**	**16131.38**	**2543.41**	**12047.10**	**23310.21**	–	–
	DE4	63	15248.52	3029.65	10118.01	22890.32	131123.72	61536.85
	DE5	**100**	**13959.04**	**2394.37**	**8472.37**	**23030.60**	–	–
D	DE1	95	13220.86	3149.01	7136.59	23544.69	37328.77	49425.27
	DE2	35	6156.90	2293.36	1579.34	11442.20	3321.93	2930.96
	DE3	**100**	**14589.90**	**2614.71**	**8152.96**	**21897.24**	–	–
	DE4	**100**	**11264.49**	**3995.33**	**3628.60**	**23101.00**	–	–
	DE5	19	7024.98	2793.20	3143.21	14145.63	5307.43	3158.89
E	DE1	95	12690.39	3539.93	6073.78	21095.05	52324.84	50566.85
	DE2	35	6124.32	2169.77	3052.99	13021.80	3913.89	2691.21
	DE3	**100**	**18866.74**	**2742.57**	**9234.27**	**23780.44**	–	–
	DE4	**100**	**10795.47**	**3174.45**	**5136.31**	**20503.74**	–	–
	DE5	2	7307.34	1123.77	6512.72	8101.97	11858.66	5825.67
F	DE1	91	14624.70	4093.28	4924.90	23278.21	86100.71	62665.31
	DE2	92	8516.30	2349.54	3624.98	15511.66	1978.72	1095.68
	DE3	**100**	**20142.90**	**1658.85**	**11280.75**	**22690.86**	–	–
	DE4	**100**	**12999.85**	**3619.56**	**5792.37**	**22410.83**	–	–
	DE5	3	7233.88	2969.75	5078.29	10621.32	11953.12	6506.82
G	DE1	95	14335.34	1129.22	11682.55	19402.85	17865.09	8237.40
	DE2	9	10176.90	1519.37	8160.27	12993.36	8032.20	5857.71
	DE3	**100**	**14280.57**	**2095.88**	**9204.23**	**23947.20**	–	–
	DE4	38	11651.67	2168.63	6064.30	16483.25	4950.78	5303.61
	DE5	1	8584.26	0.00	8584.26	8584.26	13651.09	7355.91
H	DE1	79	14007.59	1246.80	10574.54	17376.06	31555.15	43556.65
	DE2	9	11267.39	2482.72	6995.72	14284.34	12981.05	7790.02
	DE3	**100**	**14504.43**	**1819.04**	**9950.94**	**21836.79**	–	–
	DE4	33	12550.27	1876.63	8305.99	17126.70	10283.64	7914.71
	DE5	0	–	–	–	–	30932.03	11213.42
I	DE1	80	14095.89	1622.62	10159.60	17604.50	38442.86	50921.62
	DE2	28	10365.33	2622.55	5851.03	14971.19	10284.19	6590.51
	DE3	**100**	**15057.91**	**2315.43**	**10441.52**	**21595.31**	–	–
	DE4	53	11534.51	2694.72	4193.69	16363.93	11097.23	9136.05
	DE5	0	–	–	–	–	38481.31	12186.73

10^3 iterations using a swarm/population size of $N = 50$. The best solution detected throughout each experiment was recorded for each algorithm along with its feasibility. If a solution was infeasible, the corresponding penalty term was recorded to reveal the

Table 4.6: Indices of the algorithms reported in the statistical test of Fig. 4.1.

1–UPSO$\ell^{[A]}$,	2–UPSO.1$^{[A]}$,	3–UPSO.1$m^{[A]}$,	4–UPSO$\ell^{[B]}$,	5–UPSO.1$^{[B]}$,	6–UPSO.1$m^{[B]}$,	7–UPSO.5$m^{[B]}$,	8–UPSO$\ell^{[C]}$,
9–UPSO.1$^{[C]}$,	10–UPSO.1$m^{[C]}$,	11–UPSO.5$m^{[C]}$,	12–DE3$^{[A]}$,	13–DE3$^{[B]}$,	14–DE3$^{[C]}$,	15–DE5$^{[C]}$,	16–DE3$^{[D]}$,
17–DE4$^{[D]}$,	18–DE3$^{[E]}$,	19–DE4$^{[E]}$,	20–DE3$^{[F]}$,	21–DE4$^{[F]}$,	22–DE3$^{[G]}$,	23–DE3$^{[H]}$,	24–DE3$^{[I]}$,

degree of violation per case. All experiments were conducted using the data provided in [34].

4.2.1 Results for the original model

The results for the original (48–dimensional) model are reported in Table 4.5 (notice that higher function values correspond to better solutions since the problem is defined as maximization). The first two columns of the table report the corresponding parameter set and algorithm, followed by seven columns that refer to the results. The first one (labeled as "Suc.") reports the percentage of success in detecting a feasible solution in 100 experiments. The next four columns expose the mean, standard deviation, minimum and maximum of the obtained solution's value *only for the successful runs*. For the infeasible cases, the last two columns (labeled as "Mean–NF" and "StD–NF") report the mean and standard deviation of the corresponding penalties of the final solutions. Finally, the best performing algorithms that achieved success of 100% are boldfaced.

The reported results draw a clear picture of the algorithms' performances. Regarding the PSO–based approaches, there is an evident superiority of exploration–oriented variants for all parameter sets. UPSOℓ, UPSO.1 and its mutated counterpart UPSO.1m, shown a remarkable consistency regardless of the selected parameter values. The UPSO.5m variant was also very promising but with marginally inferior average performance than the rest in the case of parameter set A. Finally, no clear correlation of specific parameter set and best performing UPSO variant is revealed by the reported values.

Regarding the DE variants, DE3 was clearly the less sensitive variant with respect to the different parameter settings, since it was completely successful in all cases. Less robust behavior was obtained for DE4 and DE5, which attained complete success only for a few parameter settings. The sensitivity of DE5 was remarkable since it was downgrading from 100% for DE5$^{[C]}$, to 0% for DE5$^{[H]}$ and DE5$^{[I]}$. Also, we shall notice that DE3$^{[F]}$ achieved the best average performance among all the algorithms for the original model. We must underline that the three best performing DE operators are all defined using two difference vectors (see Eqs. (2.13), (2.14) and (2.15)), while two of them (including DE3) take advantage of the best member, x_g, of the population. Especially for DE3, its structural similarities with PSO are more than obvious.

Despite the undoubted dominance of the aforementioned UPSO and DE variants over the rest, no sound comparisons can be straightforwardly inferred among them due to

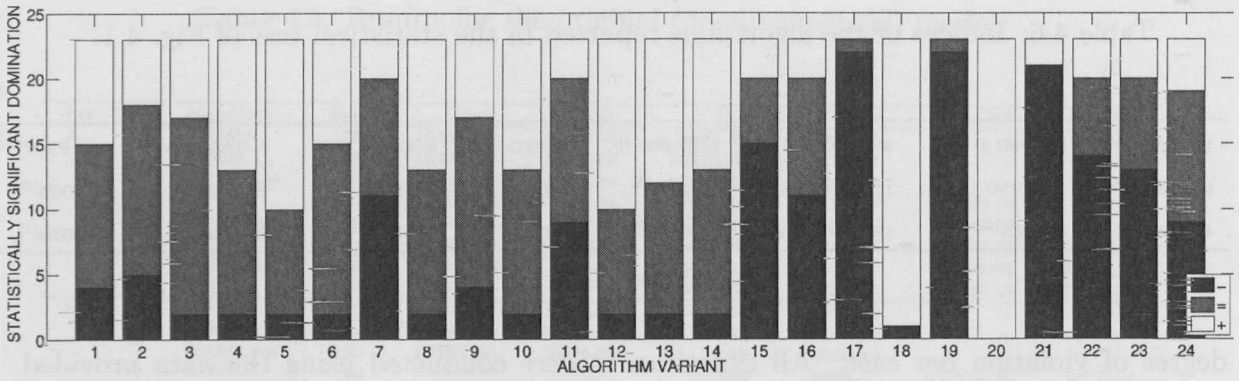

Figure 4.1: Statistical significance tests. The algorithms are indexed as in Table 4.6.

Table 4.7: Indices of the algorithms reported in the statistical test of Fig. 4.3.

Index	1	2	3	4	5	6
Algorithm	DE3$^{[E]}$	DE3$^{[E]}$	DE3$^{[E]}$	DE3$^{[F]}$	DE3$^{[F]}$	DE3$^{[F]}$
Iterations	1×10^3	2×10^3	3×10^3	1×10^3	2×10^3	3×10^3

frequent overlapping of the obtained data statistics reported in Table 4.5. To this end, the 24 completely successful variants that are boldfaced in the table, were compared through statistical significance tests. More specifically, pairwise comparisons of the performances of all 24 variants were conducted using the non–parametric Wilcoxon rank–sum test at 99% significance level. For each algorithm, we recorded the number of cases where it was better or worse with statistical significance, as well as the number of cases where its performance difference was statistically insignificant from the rest of the algorithms.

The results of these tests are graphically illustrated in Fig. 4.1, where the algorithms are indexed in order of appearance, as reported in Table 4.6. More specifically, for each algorithm there is a corresponding bar in Fig. 4.1. The bar is divided in three parts, colored with white, black and gray. The white part shows the number of other algorithms (out of 23) with which the algorithm was positively compared, i.e., it was dominant with statistical significance. The black part of the bar shows the number of other algorithms (out of 23) with which the algorithm was negatively compared, i.e., it was outperformed with statistical significance. Finally, the gray part shows the number of other algorithms with which there was no statistically significant difference in performance. Obviously, the larger the white part is, the better is the algorithm with respect to the rest of the tested variants.

The white parts of the bargraph in Fig. 4.1, clearly implies the superiority of DE3$^{[E]}$ (indexed as 18) and DE3$^{[F]}$ (indexed as 20), which can be fairly characterized as the best approaches for the original model. Nevertheless, this conclusion is based only on the specific setting of population size ($N = 50$) and 10^3 iterations, raising questions regarding

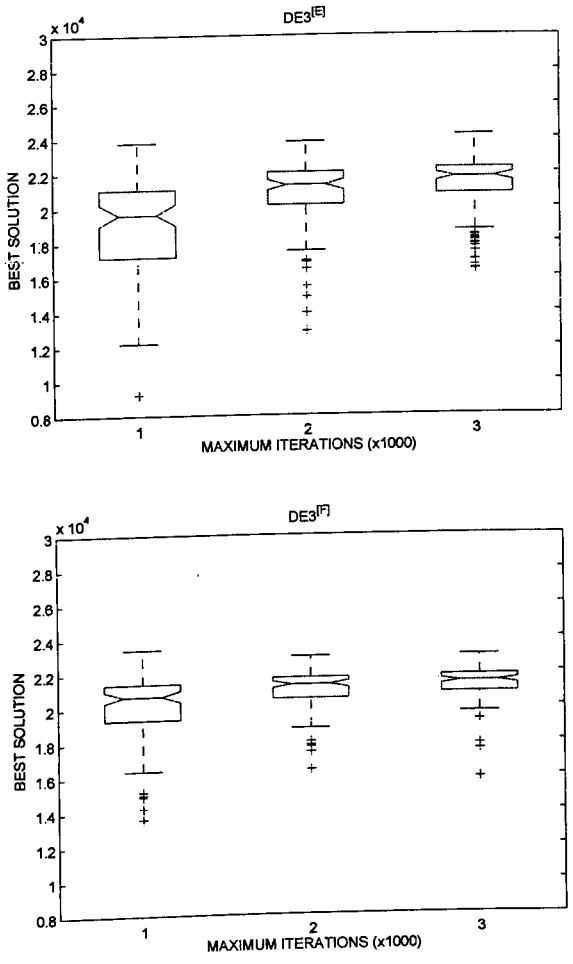

Figure 4.2: Performance of DE3$^{[E]}$ and DE3$^{[F]}$ under iteration number scaling.

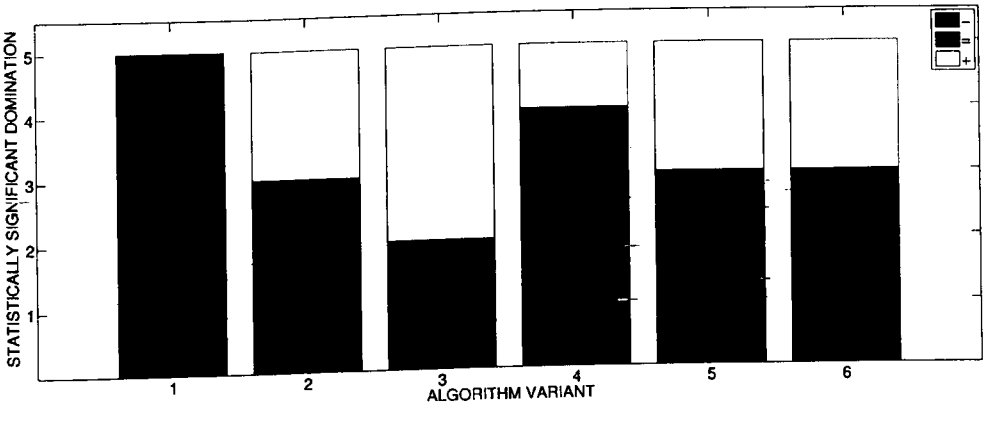

Figure 4.3: Statistical significance tests. The algorithms are indexed as in Table 4.7.

their behavior under different settings.

For this reason, further experiments were conducted for these two algorithms in order to discover additional evidence regarding their performance scaling. Thus, our experimentation was extended in two directions: firstly, we kept the population size fixed to 50

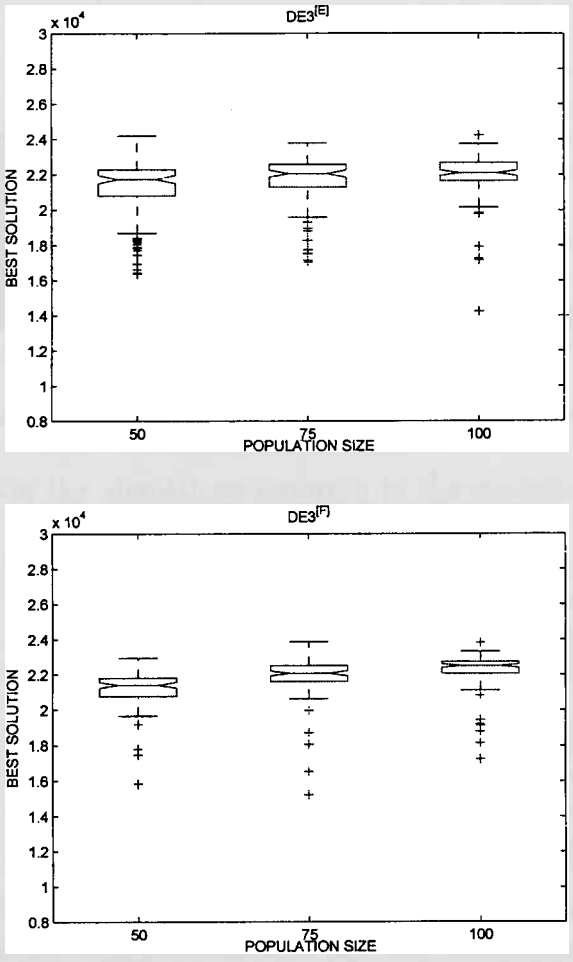

Figure 4.4: Performance of DE3$^{[E]}$ and DE3$^{[F]}$ under population size scaling.

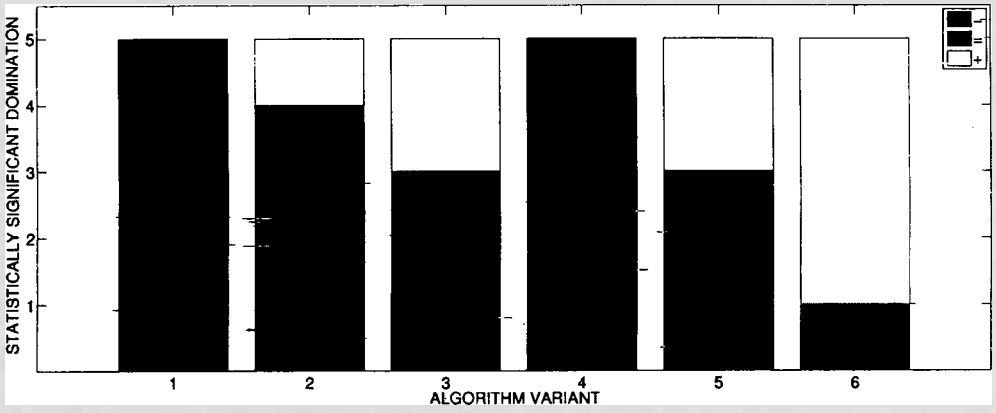

Figure 4.5: Statistical significance tests. The algorithms are indexed as in Table 4.7.

and linearly increased the available iterations to $k \times 10^3$ for $k = 2, 3$. This setting would help us to discover whether the attained behavior and relative performance of the most successful algorithms for the case of 10^3 iterations could be counterbalanced by additional computational budget. Secondly, we kept the number of iterations fixed to an adequately

high value, namely 3×10^3, and tested again the algorithms under population size scaling, i.e., $N = k \times 50$, $k = 1, 1.5, 2$.

The results for the experiments with iteration number scaling are graphically illustrated in Fig. 4.2, with respect to the value of the final solution found after 100 experiments. The corresponding statistical comparisons are illustrated in Fig. 4.3. From these figures, it is easily derived that both algorithms improved their performance, with DE3$^{[E]}$ being marginally ahead of DE3$^{[F]}$. Yet, there is a clear tendency of the boxplots in Fig. 4.2 for both algorithms to get narrower as the number of iterations increases. This verifies the general feeling that increasing the computational budget does not necessarily produce the same trend in performance.

The outcome for the experiments with population size scaling are graphically illustrated in Fig. 4.4, with respect to the value of the final solution found after 100 experiments, while the corresponding statistical comparisons are illustrated in Fig. 4.5. As we observe, the roles are reversed for the two algorithms in this case, with DE3$^{[F]}$ taking head as population size increases. This is derived by the white parts of the bars in Fig. 4.5. A possible explanation of this effect could be the different parameter sets used by the two variants. Indeed, DE3$^{[F]}$ has the same value, $F = 0.5$, with its mate variant, but differs at CR, which is higher for DE3$^{[F]}$ than DE3$^{[E]}$ (see Table 4.4). Thus, DE3$^{[F]}$ has higher probability of accepting components from newly produced vectors than retaining the existing components, as can be seen in Eq. (2.13), thereby favoring exploration. Remember that exploration–oriented variants were superior also for UPSO in our initial experiments. As expected, increasing the population size strengthens this effect further.

As a final comment for the original model, we shall mention that the best feasible solution obtained with the GA approach in [34] has objective value 15266.8. In the next section, the results for the simplified model are presented.

4.2.2 Results for the simplified model

The results for the simplified (36–dimensional) model are reported in Table 4.8, following the presentation motif of the previous section. A first inspection of the table offers some immediate conclusions. More specifically, the successful UPSO variants remarkably improved their performance, while the same is observed also for the most efficient DE approaches. However, we can see considerable differences for DE with parameter sets A, B and C. Indeed, the dominant DE3 for these cases was outperformed by other variants in the simplified model.

Also, there is a noticeable performance improvement for DE variants with one rather than two difference vectors, i.e., DE1 and DE2. This can be interpreted as a consequence of the reduced dimensionality of the simplified model. This can also account for the reduced penalty terms (violation magnitude) observed for the variants that retained their inferior performance also in the simplified model.

Further analysis of the results reveals a complete dominance of UPSO.1 against the rest PSO–based variants (mutated or not). Obviously, the reduced problem dimension-

Table 4.8: Results for the simplified (36–dimensional) model.

Par. Set	Algorithm	Suc.	Mean	St.D.	Min	Max	Mean–NF	StD–NF
A	**UPSOℓ**	**100**	**16759.17**	**3278.01**	**7509.59**	**24417.35**	–	–
	UPSO.1	**100**	**18859.93**	**2238.09**	**11760.49**	**21770.91**	–	–
	UPSO.1m	**100**	**16473.78**	**2618.19**	**8485.52**	**20836.61**	–	–
	UPSO.5	87	16455.72	3125.58	7345.19	23142.99	5351.26	5928.03
	UPSO.5m	86	15294.78	2772.99	6973.88	20877.98	8886.35	11535.16
	UPSO.9	45	15088.82	2780.00	10010.74	20087.92	54858.49	67414.87
	UPSO.9m	53	14991.10	3460.81	7498.16	23208.42	78242.57	74742.21
	UPSOg	40	14908.15	2973.21	10283.91	22636.29	78306.92	97846.59
B	**UPSOℓ**	**100**	**17136.33**	**2503.24**	**9955.92**	**22171.36**	–	–
	UPSO.1	**100**	**19804.36**	**1385.60**	**14157.15**	**21712.73**	–	–
	UPSO.1m	**100**	**17260.02**	**2537.20**	**10722.93**	**22125.78**	–	–
	UPSO.5	89	15757.12	2717.69	7908.57	20426.22	2152.33	3937.44
	UPSO.5m	89	15872.29	2732.52	9283.88	21427.43	13363.53	17319.03
	UPSO.9	67	16177.61	3362.40	8911.48	23527.84	41020.65	61171.56
	UPSO.9m	58	16558.38	3865.34	8265.28	23977.88	40028.42	64368.63
	UPSOg	59	14896.71	3240.53	8082.66	21081.66	58381.39	66775.30
C	**UPSOℓ**	**100**	**17510.62**	**2470.74**	**11599.80**	**22487.18**	–	–
	UPSO.1	**100**	**19075.84**	**2148.43**	**10947.89**	**22189.06**	–	–
	UPSO.1m	**100**	**17755.93**	**2010.20**	**10889.56**	**21172.55**	–	–
	UPSO.5	82	13508.11	3067.00	4299.81	20145.10	20308.60	58711.26
	UPSO.5m	40	13302.91	2315.85	8220.24	17000.62	23315.20	25042.97
	UPSO.9	47	14052.46	2765.54	7727.73	20750.58	61369.61	69703.66
	UPSO.9m	49	13707.07	3450.26	7459.08	21551.97	54792.29	97782.37
	UPSOg	37	16114.44	3432.76	8707.23	23744.53	76744.08	79459.57
A	**DE1**	**100**	**15984.49**	**2765.95**	**9388.31**	**21822.34**	–	–
	DE2	**100**	**11780.78**	**1901.86**	**7210.81**	**16944.50**	–	–
	DE3	88	16395.45	2164.90	10860.46	20440.12	9954.55	12194.72
	DE4	**100**	**18623.82**	**2470.09**	**9655.99**	**20927.67**	–	–
	DE5	93	10670.92	2182.30	5702.02	15630.71	1974.37	1870.77
B	DE1	70	13922.19	3133.28	5303.14	20277.95	18126.23	25327.49
	DE2	**100**	**12125.32**	**1634.36**	**6514.34**	**15649.21**	–	–
	DE3	6	13744.13	2388.96	11402.06	18206.96	75883.24	52511.67
	DE4	**100**	**18356.69**	**2360.10**	**11851.68**	**21870.83**	–	–
	DE5	65	10410.58	1661.34	7111.17	13532.27	2270.85	1484.78
C	DE1	3	10430.29	4537.52	7041.52	15585.35	143482.71	104973.00
	DE2	**100**	**16990.51**	**1677.95**	**12388.14**	**20902.19**	–	–
	DE3	0	–	–	–	–	206190.39	94357.79
	DE4	81	17329.37	3094.78	9567.92	22213.98	16762.81	22898.26
	DE5	**100**	**14064.60**	**1403.47**	**9223.34**	**17726.10**	–	–
D	**DE1**	**100**	**15793.91**	**2830.30**	**7556.60**	**22701.53**	–	–
	DE2	61	9599.96	2397.13	3359.51	15859.96	2293.09	1521.34
	DE3	**100**	**18296.31**	**1219.76**	**13717.56**	**20497.22**	–	–
	DE4	94	11703.96	1903.15	7686.59	16444.87	1588.12	1611.85
	DE5	7	8879.81	1826.91	6147.96	11762.54	5542.72	3598.24
E	**DE1**	**100**	**16904.64**	**3160.43**	**8519.29**	**22453.12**	–	–
	DE2	24	9095.77	2038.00	5916.90	13046.64	3237.87	2210.46
	DE3	**100**	**20579.86**	**702.24**	**17821.38**	**22004.01**	–	–
	DE4	96	11834.47	2385.84	6192.86	17526.98	1075.11	1136.44
	DE5	1	9501.12	0.00	9501.12	9501.12	14446.91	6224.22
F	DE1	96	18651.37	2720.45	10802.69	23491.00	5751.31	6651.73
	DE2	96	11995.80	1764.43	6495.59	14921.55	1314.26	378.57
	DE3	**100**	**20191.47**	**1307.50**	**13530.06**	**22064.48**	–	–
	DE4	**100**	**14869.66**	**2134.99**	**8000.83**	**19298.33**	–	–
	DE5	4	8706.46	1121.83	7967.51	10378.10	14405.69	6870.72
G	DE1	95	12402.31	3112.72	5370.04	18704.73	3794.68	3848.22
	DE2	6	7975.03	1770.66	5912.67	10151.68	6373.32	4466.43
	DE3	**100**	**14096.14**	**1615.40**	**10410.08**	**17182.87**	–	–
	DE4	1	8467.92	0.00	8467.92	8467.92	8831.10	5161.54
	DE5	0	–	–	–	–	21111.12	9222.42
H	DE1	96	12601.53	2838.62	6667.11	17973.07	3576.89	6110.49
	DE2	0	–	–	–	–	15438.37	6267.39
	DE3	**100**	**14202.29**	**1867.94**	**9269.24**	**17993.76**	–	–
	DE4	0	–	–	–	–	24633.79	9047.56
	DE5	0	–	–	–	–	53013.67	14042.80
I	**DE1**	**100**	**16521.72**	**2903.66**	**9958.85**	**23138.23**	–	–
	DE2	2	10714.10	953.53	10039.85	11388.34	13369.71	6866.40
	DE3	**100**	**17779.32**	**1499.10**	**14035.22**	**20761.56**	–	–
	DE4	1	6645.59	0.00	6645.59	6645.59	23390.84	10251.23
	DE5	0	–	–	–	–	86763.25	21377.91

ality along with the adequate search diversification attained by UPSO, offers a properly balanced search capability to the algorithm. On the other hand we can observe an exceptionally high failure rate of some DE variants, especially for the parameter sets G,

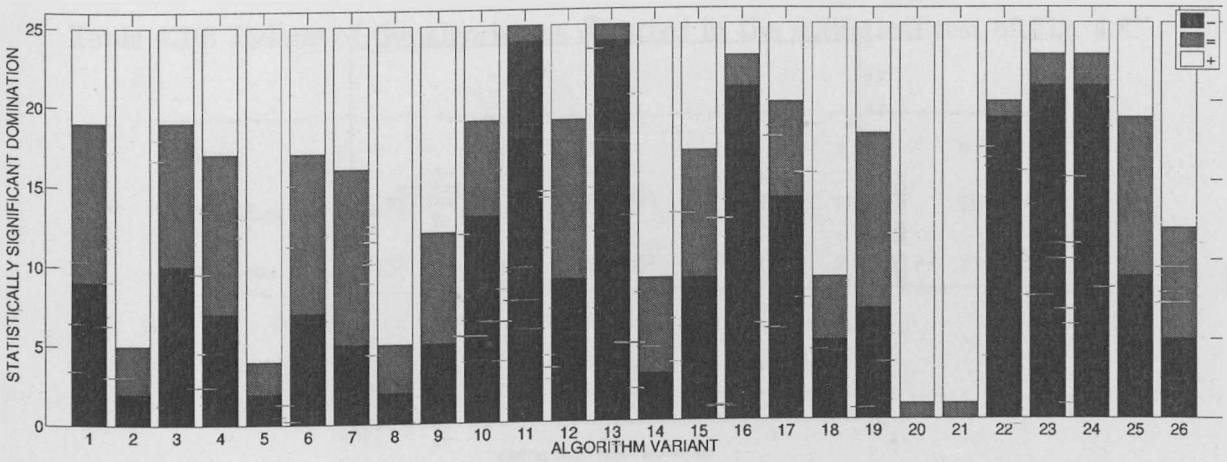

Figure 4.6: Statistical significance tests. The algorithms are indexed as in Table 4.9.

Table 4.9: Indices of the algorithms reported in the statistical test of Fig. 4.6.

1–UPSO$\ell^{[A]}$,	2–UPSO.1$^{[A]}$,	3–UPSO.1$m^{[A]}$,	4–UPSO$\ell^{[B]}$,	5–UPSO.1$^{[B]}$,	6–UPSO.1$m^{[B]}$,	7–UPSO$\ell^{[C]}$,	8–UPSO.1$^{[C]}$,
9–UPSO.1$m^{[C]}$,	10–DE1$^{[A]}$,	11–DE2$^{[A]}$,	12–DE4$^{[A]}$,	13–DE2$^{[B]}$,	14–DE4$^{[B]}$,	15–DE2$^{[C]}$,	16–DE5$^{[C]}$,
17–DE1$^{[D]}$,	18–DE3$^{[D]}$,	19–DE1$^{[E]}$,	20–DE3$^{[E]}$,	21–DE3$^{[F]}$,	22–DE4$^{[F]}$,	23–DE3$^{[G]}$,	24–DE3$^{[H]}$,
25–DE1$^{[I]}$,	26–DE3$^{[I]}$,						

H and I, which correspond to the parameter value $F = 0.7$. Even the successful DE variants (e.g. DE3) exhibited inferior performance for these cases, compared to the other parameter sets.

Similarly to the analysis in the previous section, the completely successful variants that are boldfaced in Table 4.8 were compared through statistical significance tests. Thus, pairwise comparisons of the performances of all 26 variants were conducted using the non-parametric Wilcoxon rank-sum test and the obtained results are graphically illustrated in Fig. 4.6, where the algorithms are indexed in order of appearance, as reported in Table 4.9. Again, the white part of each bar shows the number of other algorithms (out of 25) with which the algorithm was positively compared. On the other hand, the black part of the bar shows the number of algorithms (out of 25) with which it was negatively compared. Finally, the gray part shows the number of algorithms with which there was no statistically significant difference.

Despite the fact that UPSO$\ell^{[A]}$ achieved the best maximum value in all experiments, the better average performance with statistical significance was achieved by the same DE variants that prevailed also in the original model, namely DE3$^{[E]}$ (indexed as 20) and DE3$^{[F]}$ (indexed as 21), verifying their merit for the specific application. As in the previous model, we further expanded our experimental analysis by considering linearly increased number of iterations ($k \times 10^3$ for $k = 2, 3$) and population size scaling ($N = k \times 50$, $k = 1, 1.5, 2$).

All the obtained results and statistics are graphically illustrated in Figs. 4.7–4.10.

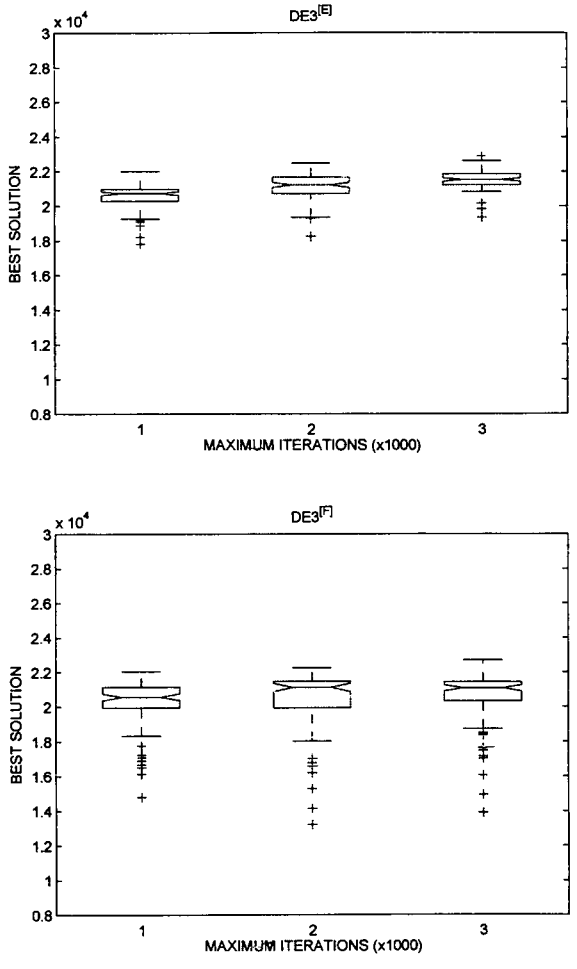

Figure 4.7: Performance of DE3$^{[E]}$ and DE3$^{[F]}$ under iteration scaling.

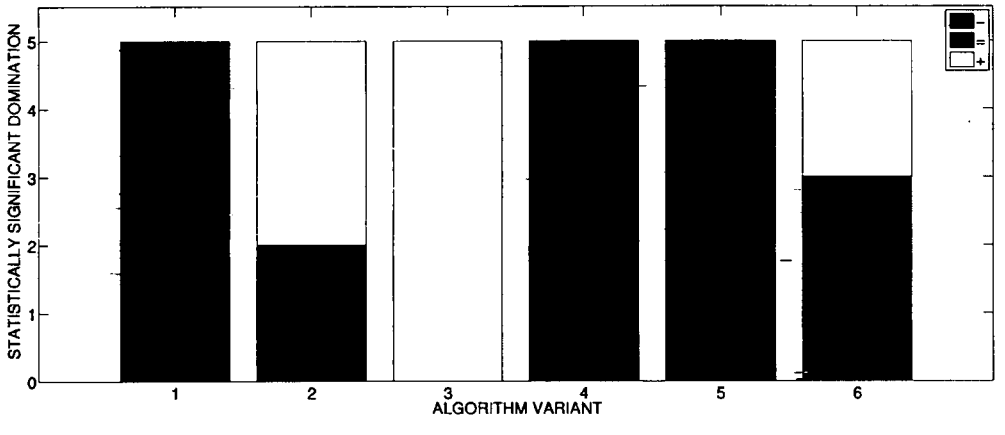

Figure 4.8: Statistical significance tests. The algorithms are indexed as in Table 4.7.

The results are completely aligned with these of the original model. More specifically, in the case of iteration scaling both algorithms improved their performance, with DE3$^{[E]}$ being significantly superior than the rest as the number of iterations increases, as derived

Table 4.10: Indices of the algorithms reported in the statistical test of Fig. 4.8.

Index	1	2	3	4	5	6
Algorithm	DE3$^{[E]}$	DE3$^{[E]}$	DE3$^{[E]}$	DE3$^{[F]}$	DE3$^{[F]}$	DE3$^{[F]}$
Iterations	1×10^3	2×10^3	3×10^3	1×10^3	2×10^3	3×10^3

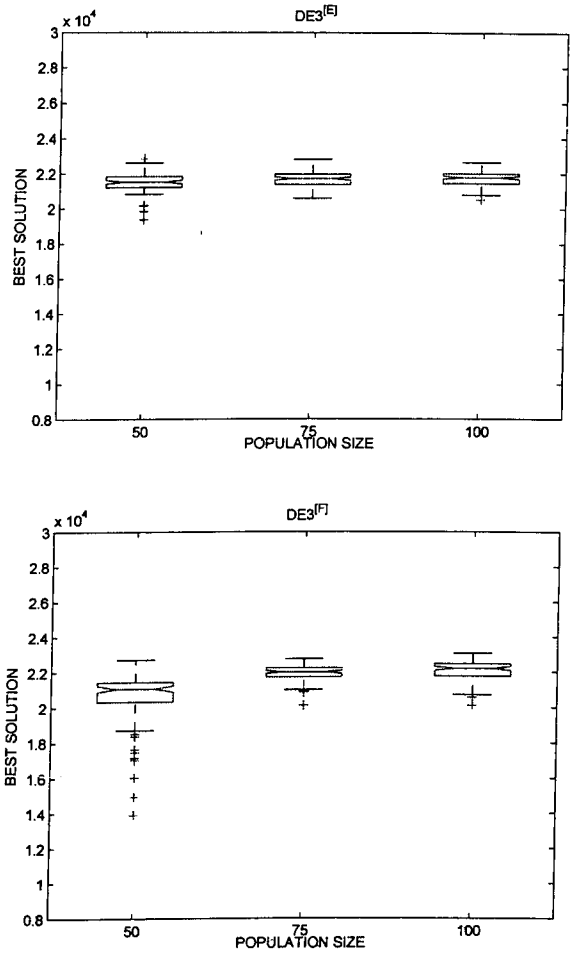

Figure 4.9: Performance of DE3$^{[E]}$ and DE3$^{[F]}$ under population size scaling.

from Figs. 4.7 and 4.8. On the other hand, under population size scaling, DE3$^{[F]}$ was particularly efficient, especially for higher values of the population size as derived from Figs 4.9 and 4.10. From these observations, we can infer that the simplified model does not modify the general performance profile and trend of the most successful algorithms in the specific problem. This is a significant property, since it allows the generalization of conclusions regarding the observed performances from the one model to the other.

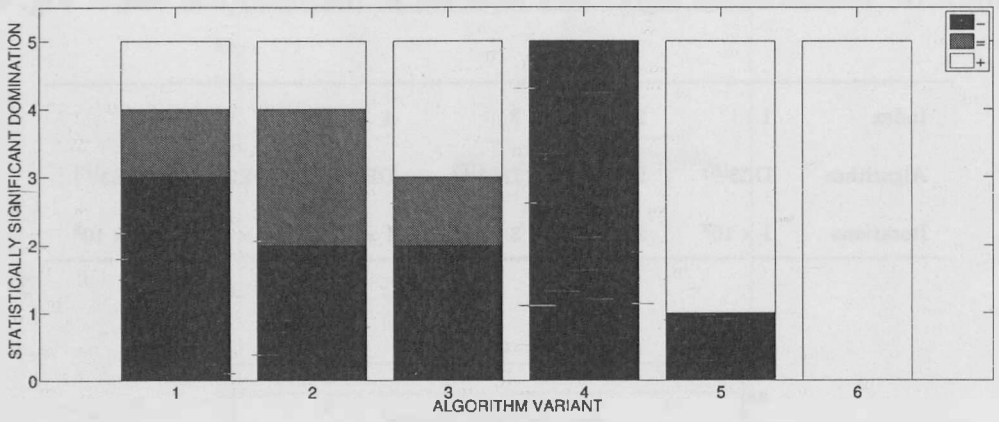

Figure 4.10: Statistical significance tests. The algorithms are indexed as in Table 4.10.

4.3 Stochastic dynamic lot–sizing model

In this section, we expose the exact settings used in our experiments as well as the obtained results, followed by the corresponding discussion.

4.3.1 Solution representation

As mentioned in Section 3.2.2, a production schedule for the entire planning horizon that minimizes the total cost defined in Eq. (3.9) can be determined through a two–stage optimization procedure. In the first stage, the optimal replenishment quantities for any sequence of replenishment epochs are analytically computed using Eq. (3.11) [41].

The second stage, consists of a binary optimization problem that aims at identifying the optimal sequence of replenishment epochs. For each epoch, a decision of placing an order (corresponding to 1) or not (corresponding to 0) must be made. If the decision is to place an order then the optimal quantity is already known from the first optimization stage and it is directly used for the computation of the total cost.

For a problem of H epochs, the employed algorithms need to work on the n–dimensional binary space $X = \{0, 1\}^n$ with $n = H$. However, as we already mentioned, the employed algorithms are primarily destined to work on real variables (with the exception of HS, which can alleviate this problem). For this reason, the tried–and–true rounding technique was adopted in the present study. More specifically, the algorithms were let to operate on the real search space $X = [0, 1]^n$ but, whenever the function evaluation of a particle (or individual) was required, its components were rounded to the nearest integer (0 or 1) without substituting the real components with the integers in the vectors. For example, the candidate solution:

$$x = (0.31,\ 0.74,\ 0.56,\ 0.91,\ 0.22)^\top,$$

would be mapped to the binary vector:

$$\bar{x} = (0,\ 1,\ 1,\ 1,\ 0)^\top,$$

and:

$$f(x) = f(\bar{x}).$$

This approach imposes only minimal intervention in the algorithms' dynamics. Only for the case of HS, the real components of the harmonies were replaced by their nearest integers both in HM and in the new harmonies produced by mutation.

4.3.2 The case of normally-distributed demand

Without restricting the applicability of the considered algorithms, stochastic demand was assumed to follow a normal distribution in our study, in accordance to the case study reported in [41]. Under this assumption, Eqs. (3.10) and (3.11) can be simplified to avoid multiple numerical integrations for determining the optimal cumulative production quantities.

In this manner, two functions are involved, namely the *cumulative normal distribution* and the *standard normal loss integral.* Let σ_t and μ_t denote the mean and standard deviation of cumulative demand in period t, with density function $f_t(x)$. Let also $\phi(x)$ denotes the standard normal density function with cumulative distribution function $\Phi(x)$, and let:

$$I_N(x) = \int_{z=x}^{\infty} (z - x)\, \phi(z)\, dz,$$

be the normal loss integral.

THEOREM 4.1. *With normally distributed demand, Eq. (3.10) is simplified as follows:*

$$K(S, i, j) = A_{i-L} + \sum_{t=i}^{j-1} h_t\, \sigma_t\, [z_t + (1 + p)\, I_N(z_t)], \tag{4.1}$$

where:

$$z_t = \frac{S - \mu_t}{\sigma_t}.$$

The proof of Lemma 4.1 can be directly derived, based on the following lemmas.

LEMMA 4.1. *It holds that* $\int_S^{\infty}(q - S)f(q)dq = \sigma I_N(\frac{S-\mu}{\sigma})$

LEMMA 4.2. *It holds that* $\int_{-\infty}^{S}(S - q)f(q)dq = \frac{S-\mu}{\sigma} + \sigma I_N(\frac{S-\mu}{\sigma})$

The proofs of Lemmas 4.1 and 4.2 are extensively presented in [41].

4.3.3 Test problems

Our aim in the present study was the investigation of the employed algorithms on various instances of the problem with respect to its dimension. The main interest behind this, is the scaling of the run–time required to find the optimal sequence of replenishment epochs. As we already mentioned in previous sections, the exhaustive inspection of all possible

combinations requires exponentially increasing time with the problem dimension (number of epochs), which becomes prohibitive even for modern computer systems. Therefore, efficiency with respect to both solution accuracy and run–time was in the center of our investigation.

For consistency, the test problems used in our experiments were based on the 12–dimensional problem presented in [41]. The data provided in this source includes the setup cost and cumulative demand for 12 epochs. We used this data and extended them for up to 48 epochs. For this purpose, we fitted a Gaussian distribution on the provided data and generated new setup cost and cumulative demand values by sampling the fitted distribution. The obtained values are reported in Table 4.11.

We considered the problem for dimensions 12, 18, 24, 30, 36, 42 and 48. The optimal sequence of the replenishment epochs for each instance was initially determined through exhaustive search. The total number of binary sequences per case as well as the required run–time[1] for their evaluation are reported in Table 4.12. As we can see, problem instances with more than 36 epochs need excessive computation time due to the huge number of sequences, which becomes larger than 10^{10}.

It must be stressed out that the actual number of binary variables for a problem instance of H epochs is equal to $H-1$, due to the model assumption that there is always an order decision in the first epoch [41], which implies that the corresponding binary variable is always fixed to 1. Thus, a problem with H epochs corresponds to 2^{H-1} binary sequences.

4.3.4 Experimental setup

We performed extensive experiments with the three employed algorithms under different parameter settings and variants. PSO was considered in both its global and local variant with ring neighborhoods of radius 1 and the default parameter set given in Section 2.1.1. DE was considered in its five basic operators and all possible combinations of its parameters, $F, CR \in \{0.3, 0.5, 0.7\}$. HS was considered under various harmony memory sizes and for both the uniform and the linear ranking selection schemes. Preliminary experiments provided clear evidence that for harmony memory size equal to N, the values $M_{\text{prod}} = N/2$ and $M_{\text{rep}} = N/5$ for the produced and replaced new harmonies (see Section 2.3), respectively, constitute appropriate choices.

The swarm size in PSO (equivalently the population size in DE and harmony memory size in HS) was set to $10 \times n$ for all problem instances, where n is the corresponding problem dimension. For each algorithm, 100 independent experiments were performed per problem instance. The stopping condition was the determination of the optimal sequence of replenishment epochs within a prescribed maximum number of function evaluations. For the smallest problem instances, this number was equal to the total number of sequences. For larger instances, it was limited to the value $T_{\max} = 5 \times 10^6$ as reported in

[1]The time refers to an Intel I7 machine with 8GB of memory.

Table 4.11: Setup cost and cumulative demand ($f_t(q)$) used in the test problems.

Time	Setup	Cumulative demand	
period	cost	Mean	St.D.
1	85	69	7.7
2	102	98	8.3
3	102	134	9.2
4	101	195	11.4
5	98	256	13.3
6	114	282	13.6
7	105	316	14.1
8	86	383	16.0
9	119	428	16.7
10	110	495	18.3
11	98	574	20.3
12	114	630	21.2
13	108	657	25.1
14	122	718	25.6
15	79	767	26.8
16	111	816	27.7
17	106	894	28.6
18	89	952	29.9
19	98	1008	30.9
20	106	1089	31.7
21	140	1127	33.0
22	132	1192	33.6
23	89	1259	34.5
24	135	1307	35.5
25	110	1363	36.2
26	102	1395	36.9
27	110	1427	37.3
28	101	1481	37.8
29	102	1546	38.5
30	119	1644	39.3
31	118	1685	40.5
32	118	1741	41.0
33	110	1792	41.7
34	90	1810	42.3
35	110	1855	42.5
36	120	1876	43.1
37	108	1943	43.3
38	114	1980	44.1
39	110	2034	44.5
40	100	2077	45.1
41	106	2135	45.6
42	95	2177	46.2
43	112	2238	46.6
44	91	2304	47.3
45	92	2387	48.0
46	94	2436	48.9
47	72	2450	49.4
48	118	2488	49.5

Table 4.12: Total number of sequences and run–time required for the exhaustive search per problem instance. The symbol "$\sim$" stands for "order of" and "$>$" denotes "higher than".

Dim.	Sequences	Time
12	2048	0.1 sec.
18	131072	0.5 sec.
24	$\sim 10^6$	~ 5 sec.
30	$\sim 10^8$	~ 312 sec.
36	$> 10^{10}$	> 7 hrs.
42	$> 10^{10}$	> 2 days
48	$> 10^{10}$	> 5 days

Table 4.13: Maximum function evaluations ($T_{\max}$) and swarm/population/harmony memory size (N) per problem instance.

Dim.	$T_{\max}$	N
12	2048	120
18	131072	180
24	5×10^6	240
30	5×10^6	300
36	5×10^6	360
42	5×10^6	420
48	5×10^6	480

Table 4.13.

All variants were extensively tested. The total number of independent experiments for all problem instances was higher than 26000. For each algorithm, variant, parameter set and problem instance, we recorded the success rate, i.e., the number of experiments (out of 100) where it succeeded to reach the optimal solution within the maximum number of function evaluations. Also, the mean, standard deviation, minimum and maximum value of the expected number of function evaluations, as well as the average required run–time (in seconds) were recorded per algorithm and problem instance.

4.3.5 Presentation of results and discussion

In view of the huge amount of the obtained results, it was necessary to make a selection of only the most interesting cases to report in our presentation. For this reason, we identified the most promising variant of each algorithm. For PSO, the lbest model was far the most

successful variant. The gbest model was prone to get stuck in suboptimal solutions, even for the low–dimensional problem instances. This can be attributed to its exploitation orientation in combination with the rounding scheme. Very often, the particles in the gbest model were rapidly clustered in very small ranges around the best solutions, also assuming very small velocities. This effect, combined with the fact that components in the range $[0, 0.5]$ were mapped to 0 while components in $(0.5, 1]$ were mapped to 1, offers a reasonable explanation for the low efficiency of gbest PSO. On the other hand, lbest PSO is clearly more exploration–oriented than gbest. Thus, the particles were able to retain sufficiently higher velocities that allowed them to move from the one half of the search space to the other, thereby exploring a higher number of binary sequences.

Regarding the DE algorithm, two operators were clearly distinguished among the five presented in Section 2.2, namely DE2 and DE5, while the most successful parameter values were $F = 0.7$ and $CR = 0.3$. A closer look at these two operators reveals that they both use only randomly selected individuals from the population, in contrast to the rest operators that exploit the best individual. This can be interpreted as an evidence that, on average, operators with higher diversity–preserving properties are related to the best observed performance. This is aligned with our observations reported above for the PSO algorithm.

Finally, for the HS algorithm, the uniform selection scheme was more efficient than the linear ranking scheme. It must be underlined that, since uniform selection represents the most fair scheme (equal selection probability for all vectors stored in memory), the linear ranking scheme was assigned high selective pressure, representing the completely biased selection towards the best harmonies. Obviously, uniform selection is more diversity–preserving than the linear ranking. Hence, its superiority aligns with the observations for the previous two algorithms, i.e., diversity–preserving variants are more successful in the specific problem.

In fact, this is the first interesting conclusion of the present work and it can be attributed to the nature of the binary optimization problem, with local minimizers that differ slightly (in one or two components) from the global one while their function values differ less than 0.6% from the global minimum.

Table 4.14 reports the detailed results for the aforementioned most successful algorithmic variants. More specifically, for each algorithm and problem instance, the success rate (successful experiments out of 100), mean, standard deviation, minimum, and maximum value of the required function evaluations for the successful experiments, as well as the required run–time (in seconds) are reported. The column that corresponds to the best algorithm is boldfaced. As best algorithm, we considered the one that primarily had the largest success rate and the smallest mean, and secondarily the smallest run–time.

In addition to Table 4.14, the results are also graphically illustrated in Figs. 4.11–4.12 to provide intuitive evidence of their performance and facilitate visual comparisons among them. Figure 4.11 illustrates the success rate per algorithm, with different colors denoting the different problem instances in ascending dimension order. In Figs. 4.13–4.19 boxplots

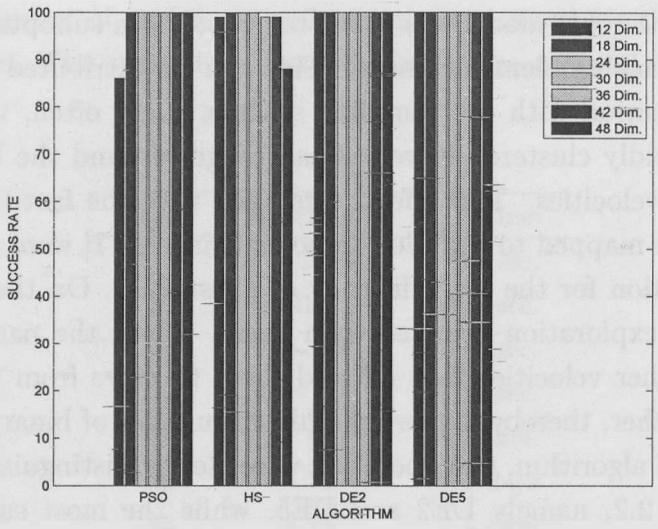

Figure 4.11: Success rate for each algorithm. Different colors denote the different problem instances.

are used to illustrate the distribution of the number of function evaluations required for each algorithm in the 100 independent experiments. On each box, the central mark is the median, the edges of the box are the 25–th and 75–th percentiles, the whiskers extend to the most extreme values, and the outliers are plotted individually (denoted with crosses). The notches define comparison intervals between medians. Two medians are significantly different at the 5% level if the corresponding intervals do not overlap. The interval endpoints are the extremes of the notches. Finally, Fig. 4.12 illustrates the scaling of the required run–time per algorithm as dimension increases.

A first inspection of Table 4.14 provides some immediate conclusions. Firstly, there is an undoubtful superiority of the DE variants against PSO and HS. Evidently, DE2 is the overall best performing variant, followed by DE5, PSO and HS. Secondly, the performance differences among them exhibit an increasing pattern with the problem's dimension. As we observe, even DE5 grows an exponentially increasing difference with DE2, although it has a slightly better mean in the 12–dimensional case. However, the latter problem instance needs special attention. A closer inspection of the reported data reveals that, despite the slightly lower mean of DE5, DE2 has slightly smaller standard deviation, which may suggest statistical insignificance between them. This is also visible in Fig. 4.13 with the overlapping comparison intervals between their medians in the boxplots. Also, we can observe that HS had also very satisfactory performance. In fact, it had the smallest mean in the successful experiments but with a slightly worse success rate, which is the reason for not considering it as the best algorithm in the 12–dimensional case. The same question as previously for the DE variants rises also here, i.e., how (statistically) crucial is the observed difference.

In order to answer this question, we performed a statistical significance test for each pair of algorithms. For this purpose, the non–parametric Wilcoxon rank–sum test was used. Each pair was tested against the null hypothesis that the samples have the same

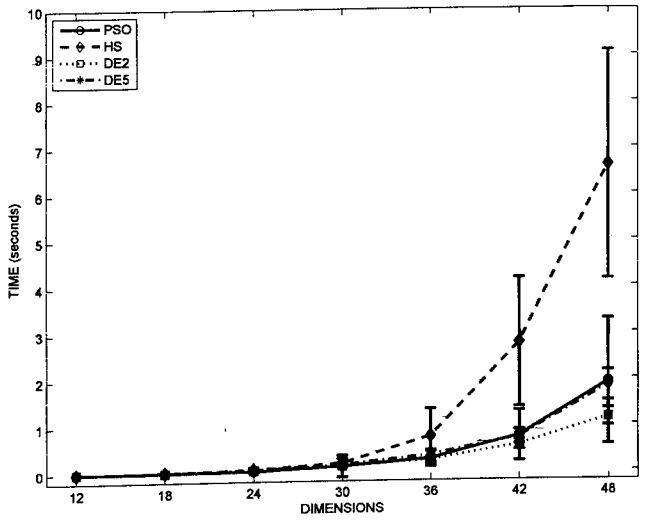

Figure 4.12: The required running time per algorithm and problem instance.

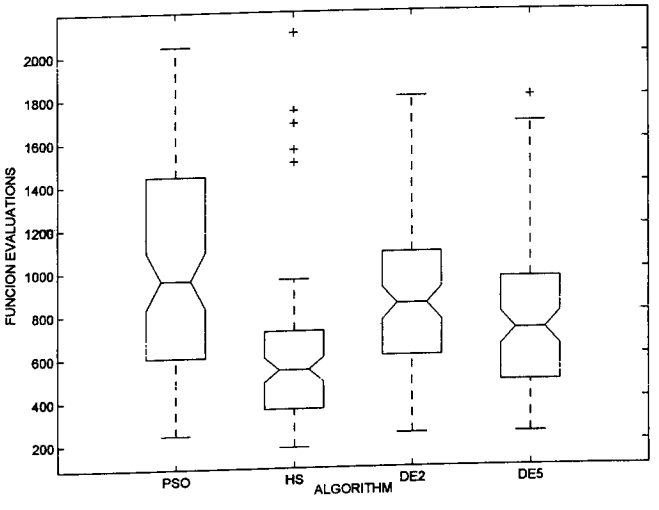

Figure 4.13: Boxplots of the mean number of function evaluations required for each algorithm in 100 independent experiments for the 12–dimensional problem instance.

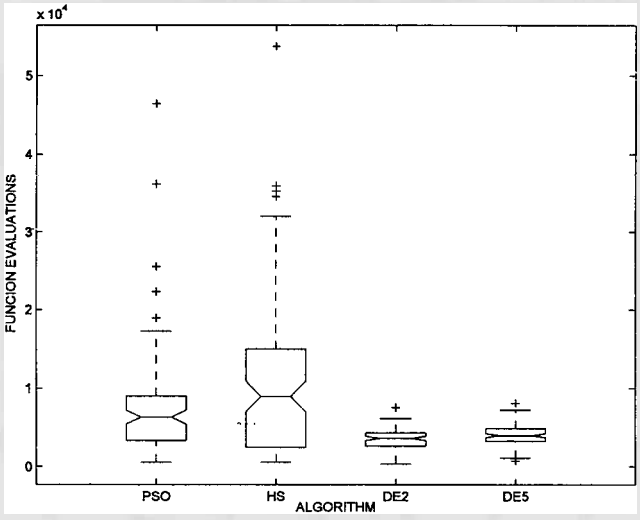

Figure 4.14: Boxplots of the mean number of function evaluations required for each algorithm in 100 independent experiments for the 18–dimensional problem instance.

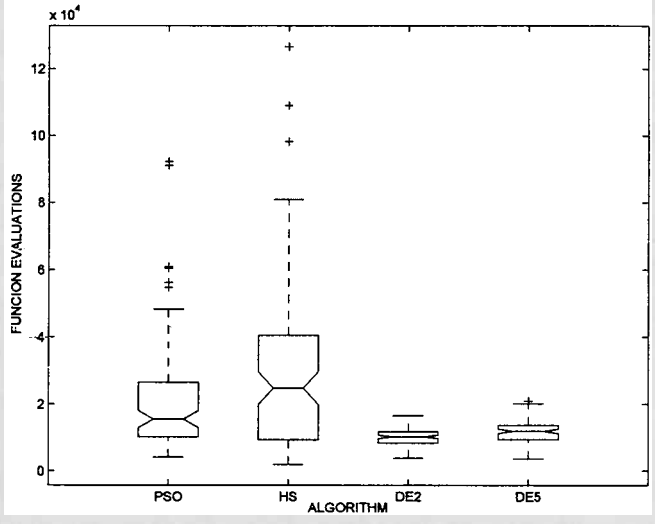

Figure 4.15: Boxplots of the mean number of function evaluations required for each algorithm in 100 independent experiments for the 24–dimensional problem instance.

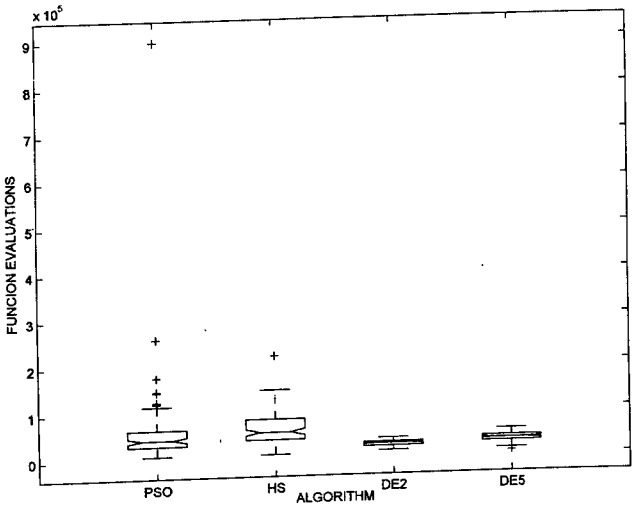

Figure 4.16: Boxplots of the mean number of function evaluations required for each algorithm in 100 independent experiments for the 30–dimensional problem instance.

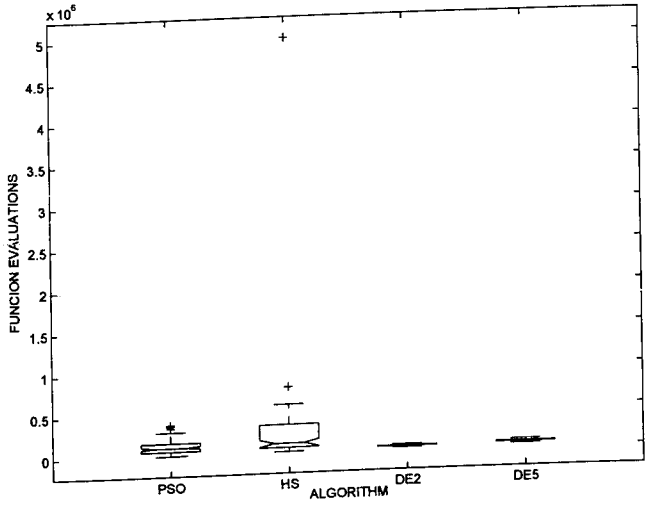

Figure 4.17: Boxplots of the mean number of function evaluations required for each algorithm in 100 independent experiments for the 36–dimensional problem instance.

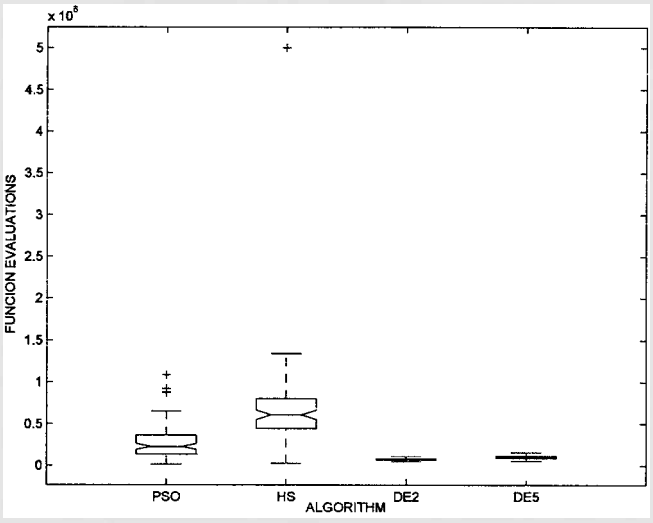

Figure 4.18: Boxplots of the mean number of function evaluations required for each algorithm in 100 independent experiments for the 42–dimensional problem instance.

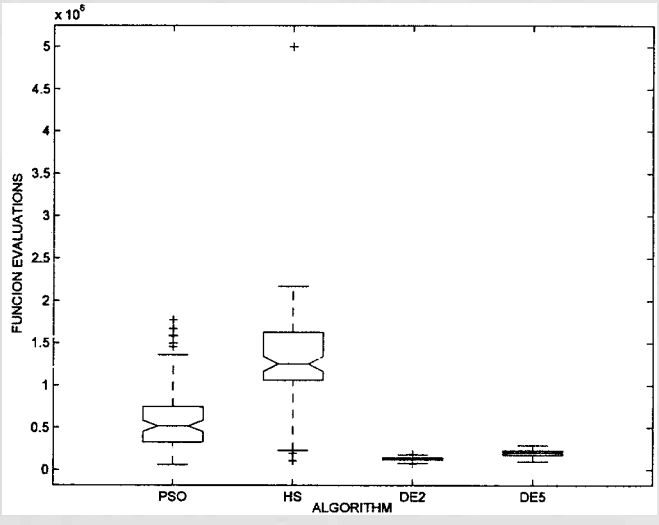

Figure 4.19: Boxplots of the mean number of function evaluations required for each algorithm in 100 independent experiments for the 48–dimensional problem instance.

median in a 95% level of significance. The outcome of the tests is reported in Table 4.15, where the existence of statistical significance is denoted with the symbol "*" and the lack is denoted with the symbol "-".

The statistical tests revealed that, as anticipated, the two DE algorithms had essentially the same performance in the 12–dimensional problem. Also, the visual evidence from Fig. 4.13, which suggests that HS has noteworthy performance in the specific problem instance, is verified by the statistical significance of HS against the rest of the algorithms. However, the performance of HS exhibits a rapid decline as dimension increases. The picture becomes clearer in higher–dimensional instances, where all algorithms are statistically different, with a single exception between PSO and HS in the 18–dimensional case, which can be attributed to their large standard deviations.

Regarding the running time illustrated in Fig. 4.12, we observe an anticipated superiority of the computationally cheapest approach, i.e., the DE2 variant, while PSO and DE5 had virtually the same run–time requirements. HS remained the more computationally demanding algorithm, evidently due to the excessive required number of function evaluations and its reduced success rate in the higher–dimensional cases.

As a closing remark, we must underline that notwithstanding their differences, the employed heuristic algorithms offered immense improvement against the exhaustive search, which is the trivial baseline for addressing such problems.

4.4 Synopsis

We have extensively presented the experimental settings for the specific problems and the algorithms used for both models, along with the results and the statistical analysis for the algorithms performance. They were part of the work published as [32] and [33]. In the next chapter, the concluding remarks of the current thesis are presented.

Table 4.14: Results for all algorithms and problem instances.

Dim.	Stat.	PSO	HS	DE2	DE5
12	Succ.	86	99	100	**100**
	Mean	911.16	556.36	823.20	**778.80**
	StD	454.33	291.81	363.19	**373.54**
	Min	240	180	240	**240**
	Max	2040	1740	1800	**1800**
	Time	0.00023	0.00000	0.00000	**0.00010**
18	Succ.	100	100	**100**	100
	Mean	7524.00	10693.80	**3556.80**	3997.80
	StD	6736.47	9927.87	**1277.40**	1417.39
	Min	540	540	**360**	720
	Max	46440	53730	**7560**	8100
	Time	0.007	0.014	**0.005**	0.007
24	Succ.	100	100	**100**	100
	Mean	20846.40	29565.60	**10022.40**	11714.40
	StD	16223.65	23897.46	**2440.96**	3581.99
	Min	4080	1920	**3840**	3600
	Max	92160	126600	**16560**	20880
	Time	0.026	0.063	**0.036**	0.053
30	Succ.	100	100	**100**	100
	Mean	61653.00	63703.50	**25302.00**	31272.00
	StD	92675.79	33648.22	**6189.35**	8393.15
	Min	8100	7200	**10800**	4800
	Max	899700	219600	**37800**	52500
	Time	0.119	0.191	**0.123**	0.152
36	Succ.	100	99	**100**	100
	Mean	122878.80	198676.36	**41648.40**	56170.80
	StD	79122.15	159591.75	**8280.73**	11729.42
	Min	11160	21780	**19440**	29880
	Max	383040	801360	**63000**	84960
	Time	0.274	0.756	**0.245**	0.354
42	Succ.	100	99	**100**	100
	Mean	277708.20	607986.06	**74991.00**	103286.40
	StD	198676.70	308601.75	**14390.06**	21287.47
	Min	16380	28560	**44520**	53760
	Max	1086120	1339380	**106680**	156660
	Time	0.756	2.772	**0.562**	0.735
48	Succ.	100	88	**100**	100
	Mean	607920.00	1199640.00	**130032.00**	203716.80
	StD	387796.66	452624.87	**21837.47**	36113.55
	Min	63360	105360	**72480**	98880
	Max	1770240	2167440	**189120**	291360
	Time	1.913	6.611	**1.149**	1.835

Table 4.15: Wilcoxon rank–sum tests between the algorithms.

Dim.		PSO	HS	DE2	DE5
12	PSO	–	*	*	*
	HS		–	*	*
	DE2			–	–
	DE5				–
18	PSO	–	–	*	*
	HS		–	*	*
	DE2			–	*
	DE5				–
24	PSO	–	*	*	*
	HS		–	*	*
	DE2			–	*
	DE5				–
30	PSO	–	*	*	*
	HS		–	*	*
	DE2			–	*
	DE5				–
36	PSO	–	*	*	*
	HS		–	*	*
	DE2			–	*
	DE5				–
42	PSO	–	*	*	*
	HS		–	*	*
	DE2			–	*
	DE5				–
48	PSO	–	*	*	*
	HS		–	*	*
	DE2			–	*
	DE5				–

Chapter 5

Conclusions

The first part of the thesis 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 alternatives solutions.

The Wagner–Whitin dynamic lot–size problem has been widely studied in literature. The original deterministic model recently has been extended by considering stochastic demand. This was the main problem tackled in the present thesis. Our approach was based on PSO, DE and HS, three established algorithms with an ongoing increasing popularity in research community. Proper modifications were introduced in the algorithms to address the most controversial part of problem, which consists of a binary optimization task. Special attention was paid to avoid radical modifications of the algorithms' dynamics.

Experimental results on a previously used test case with normally distributed demand manifest that the employed algorithms, especially DE and PSO, can be very efficient even in high–dimensional problems, with respect to both solution accuracy and time efficiency. The next step in our research will consider problems with different distributions of demand as well as different heuristic optimization approaches.

BIBLIOGRAPHY

[1] A. Aggarwal and J. K. Park. Improved algorithms for economic lot size problems. *Operations Research*, 41:549–571, May 1993.

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

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

[4] H.-G. Beyer and H.-P. Schwefel. Evolution strategies: A comprehensive introduction. *Natural Computing*, 1(1):3–52, 2002.

[5] J. Biethahn and V. Nissen. *Evolutionary Algorithms in Management Applications*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 1995.

[6] O. Braysy, W. Dullaert, and M. Gendreau. Evolutionary algorithms for the vehicle routing problem with time windows. *Journal of Heuristics*, 10:587–611, 2004.

[7] U. K. Chakraborty, K. Deb, and M. Chakraborty. Analysis of selection algorithms: A markov chain approach. *Evolutionary Computation*, 4:133–167, 1996.

[8] 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.

[9] M. G. Epitropakis, D. K. Tasoulis, N. G. Pavlidis, V. P. Plagianakos, and M. N. Vrahatis. Enhancing differential evolution utilizing proximity–based mutation operators. *IEEE Trans. Evol. Comput.*, 15(1):99–119, 2011.

[10] A. Federgruen and M. Tzur. A simple forward algorithm to solve general dynamic lot sizing models with n periods in o(nlogn) or o(n) time. *Management Science*, 37:909–925, 1991.

[11] 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.

[12] L. K. Gaafar and A. S. Aly. Applying particle swarm optimisation to dynamic lot sizing with batch ordering. *International Journal of Production Research*, 47(12):3345–3361, 2009.

[13] Z. Geem. Optimal scheduling of multiple dam system using harmony search algorithm. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 4507 LNCS:316–323, 2007.

[14] Z. Geem. Novel derivative of harmony search algorithm for discrete design variables. *Applied Mathematics and Computation*, 199(1):223–230, 2008.

[15] Z. W. Geem, J. H. Kim, and G. Loganathan. A new heuristic optimization algorithm: Harmony search. *Simulation*, 76(2):60–68, 2001.

[16] Y. Guan. Stochastic lot-sizing with backlogging: computational complexity analysis. *Journal of Global Optimization*, pages 1–28, 2010.

[17] Y. Guan and A. J. Miller. Polynomial-time algorithms for stochastic incapacitated lot-sizing problems. *Operations Research*, 56:1172–1183, 2008.

[18] 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.

[19] 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.

[20] R. Jans and Z. Degraeve. Modeling industrial lot sizing problems: a review. *International Journal of Production Research*, 46:1619–1643, 2008.

[21] B. Karimi, S. M. T. F. Ghomi, and J. M. Wilson. The capacitated lot sizing problem: a review of models and algorithms. *Omega, The International Journal of Management Science*, 31:365–378, 2003.

[22] J. Kennedy. Small worlds and mega–minds: Effects of neighborhood topology on particle swarm performance. In *Proc. IEEE Congr. Evol. Comput.*, pages 1931–1938, Washington, D.C., USA, 1999. IEEE Press.

[23] 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.

[24] E. C. Laskari, K. E. Parsopoulos, and M. N. Vrahatis. Particle swarm optimization for integer programming. In *Proceedings of the IEEE 2002 Congress on Evolutionary Computation*, pages 1582–1587, Hawaii (HI), USA, 2002. IEEE Press.

[25] K. S. Lee and Z. W. Geem. A new meta-heuristic algorithm for continuous engineering optimization: harmony search theory and practice. *Computer Methods in Applied Mechanics and Engineering*, 194:3902–3933, 2005.

[26] Q.-K. Pan, M. F. Tasgetiren, and Y.-C. Liang. A discrete particle swarm optimization algorithm for the no-wait flowshop scheduling problem. *Computers and Operations Research*, 35(9):2807 – 2839, 2008.

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

[28] 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.

[29] K. E. Parsopoulos and M. N. Vrahatis. Studying the performance of unified particle swarm optimization on the single machine total weighted tardiness problem. In A. Sattar and B. H. Kang, editors, *Lecture Notes in Artificial Intelligence (LNAI)*, volume 4304, pages 760–769. Springer, 2006.

[30] 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.

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

[32] G. Piperagkas, I. Konstantaras, K. Skouri, and K. Parsopoulos. Solving the stochastic dynamic lot-sizing problem through nature-inspired heuristics. *Computers and Operations Research*, 39(7):1555 – 1565, 2012.

[33] G. Piperagkas, C. Voglis, V. Tatsis, K. Parsopoulos, and K. Skouri. Applying pso and de on multi-item inventory problem with supplier selection. In *Proceedings of the 9th Metaheuristics International Conference*, Udine, Italy, 2011.

[34] 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.

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

[36] 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.

[37] 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.

[38] P. N. Suganthan. Particle swarm optimizer with neighborhood operator. In *Proc. IEEE Congr. Evol. Comput.*, pages 1958–1961, Washington, D.C., USA, 1999.

[39] M. Tasgetiren, Y.-C. Liang, and Q.-K. Pan. A discrete differential evolution algorithm for the permutation flowshop scheduling problem. *Computers and Industrial Engineering*, 55(4):795–816, 2008.

[40] I. C. Trelea. The particle swarm optimization algorithm: Convergence analysis and parameter selection. *Information Processing Letters*, 85:317–325, 2003.

[41] V. Vargas. An optimal solution for the stochastic version of the wagner-whitin dynamic lot-size model. *European Journal of Operational Research*, 198(2):447 – 451, 2009.

[42] A. Wagelmans, S. Van Hoesel, and A. Kolen. Economic lot sizing: An o(n log n) algorithm that runs in linear time in the wagner-whitin case. *Operations Research*, 40:145–156, 1992.

[43] H. M. Wagner and T. M. Whitin. Dynamic version of the economic lot size model. *Management Science*, 5:89–96, 1958.

[44] W. Zangwill. A backlogging model and a multi-echelon model of a dynamic economic lot size production system – a network approach. *Management Science*, 15:506–527, 1969.

PUBLICATIONS

Journal papers:

1. G.S. Piperagkas, I. Konstantaras, K. Skouri, K.E.Parsopoulos, "Solving the stochastic dynamic lot-sizing problem through nature-inspired heuristics", *Computers & Operations Research, Elsevier*, Volume 39, Issue 7, July 2012, Pages 1555-1565,

2. G.S. Piperagkas, A.G. Anastasiadis and N.D. Hatziargyriou, "Stochastic PSO-based heat and power dispatch under environmental constraints incorporating CHP and wind power units", *Electric Power Systems Research, Elsevier*, vol. 81, (2011) pp. 209-218.

Conference papers:

1. G.S. Piperagkas, G. Georgoulas, K.E. Parsopoulos, C.D. Stylios, A. Likas, "Integrating Particle Swarm Optimization and Reinforcement Learning in noisy problems", *Genetic and Evolutionary Computation Conference (GECCO '12), July 2012, Philadelphia, USA*

2. C. Voglis, G.S. Piperagkas, K.E. Parsopoulos, D.G. Papageorgiou, I.E. Lagaris, "MEMPSODE: Comparing Particle Swarm Optimization and Differential Evolution Within a Hybrid Memetic Global Optimization Framework", *BBOB workshop, Genetic and Evolutionary Computation Conference (GECCO '12), July 2012, Philadelphia, USA*

3. C. Voglis, G.S. Piperagkas, K.E. Parsopoulos, D.G. Papageorgiou, I.E. Lagaris, "MEMPSODE: An Empirical Assessment of Local Search Algorithm Impact on a Memetic Algorithm Using Noiseless Testbed", *BBOB workshop, Genetic and Evolutionary Computation Conference (GECCO '12), July 2012, Philadelphia, USA*

4. I.S. Kotsireas, K.E. Parsopoulos, G.S. Piperagkas, M.N. Vrahatis "Ant-based approaches for solving autocorrelation problems", *The 8th international conference in Swarm Intelligence (ANTS '12), LNCS, September 2012, Brussels, Belgium*

5. G.S. Piperagkas, C. Voglis, V.A. Tatsis, K.E. Parsopoulos, K. Skouri, "Applying PSO and DE on multi-item inventory problem with supplier selection", *The 9th Metaheuristics International Conference, July 2011, Udine, Italy*

Curriculum Vitae

Grigoris Piperagkas received his Diploma in Electrical and Computer Engineering from the National Technical University of Athens in October 2009, with specialization in Energy, Power Systems and Industrial Applications. He worked towards his diploma thesis in Electric Power Systems Laboratory of NTUA. He is now finishing his MSc studies in Computer Science at the Department of Computer Science, University of Ioannina, Greece. His work for the diploma and master thesis was publiched in international scientific journals and conference proceedings. His main research interests are computational optimization, simulation of stochastic systems, operations research and power systems with renewable power sources.