Metaheuristic approaches for scheduling the Trieste-Fernetti pickup and delivery service

George Georgoulas*. Grigoris Piperagkas*, Giorgio Iacobellis, Valentina Boschian***, Fabrizio Simeoni****, Srecko Maksimovic ****, Walter Geretto****, Chrysostomos D. Stylios***

**Technological Educational Institute of Epirus, 47100, Arta, Greece* (e-mail: georgoul@gmail.com; g.piperagkas@gmail.com; stylios@teiep.gr)
***Politecnico di Bari, Via E. Orabona4, 70125 Bari, Italy* (e-mail: *iacobellis@deemail.poliba.it*)
****University of Trieste, Via Valerio 10, 34127 Trieste, Italy (valentina.boschian@di3.units.it)*
*****Teorema Engineering Srl., Area Science Park Basovizza, Trieste, Italy* ***e-mail:*** *Fabrizio.Simeoni@teorema.net; Srecko.Maksimovic@teorema.net*; Walter.Geretto@teorema

Abstract: This work presents two metaheuristic optimization methods that are designed and developed to assist a shuttle service process for the case of the port of Trieste and the dry-port of Fernetti. The process is applied on simulated, yet realistic data, gathered in the context of the SAIL Marie Curie Project corresponding to probable operational every-day scenarios. The results suggest that a satisfactory solution can be achieved with more than one metaheuristic optimizers, providing a valuable tool to the personnel for handling container flow.

Keywords: Metaheuristics, Harmony Search, Particle Swarm Optimization, Vehicle Routing Problem, Dry-port.

1. INTRODUCTION

In recent years maritime traffic flow has been increasing in terms of volume of goods and interconnections with different means of transport, such as railway and road networks. Moreover, for many decades, management and decision making was considered an art acquired with experience combined with personal intuition, creativity and judgment. Although personal qualifications remain valuable, the increasing complexity of modern business environments and the vast volume of data needed to be taken into account, make the use of computerized methods a necessity (Turban et al., 2010).

In this context, the “ICT System Addressed to Integrated Logistic management and decision support for intermodal port and dry port facilities” project aims to develop an integrated ICT platform able to support logistic chain of goods flow, and all business operations provided in a port and a dry-port areas. A supportive technology is developed integrating data and models describing different levels of the system with various degrees of abstraction as well as expert knowledge in order to address various decisions for different time horizons.

More specifically, the test case environment concerns the Trieste-Fernetti complex, which plays a crucial role in the Friuli Venezia Giulia region, an Italian region acting as a gateway position towards East Europe and the Balkans. The logistic system in this region is particularly significant both for its geographical location, at the meeting point of the trans-European Corridor V and the Adriatic Corridor, and its concentration of ports and land, sea and railway transport networks. Therefore the impact of the proposed solutions will consider the benefits of new technological solutions on intermodal traffic and will have important effects on the overall logistic system.

One of the services that the Trieste-Fernetti complex (Figure 1) is going to establish, concerns the use of a “shuttle service” for the containers that are parked either at the port of Trieste or at Fernetti. This scenario involves the use of a fleet of trailers/trucks that move containers from Trieste to Fernetti or the other way around. Boschian *et al.* (2011) provided a good description of the case study through a UML metamodel.

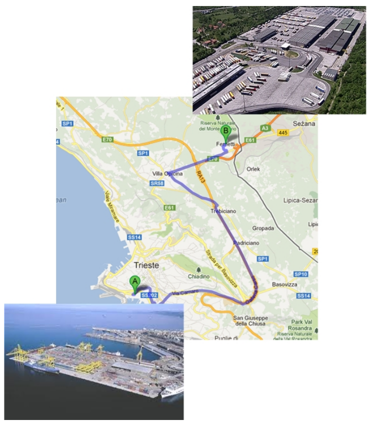

Fig. 1. The Trieste- Fernetti complex environment.

For this particular problem, a model based approach has been adopted employing a simplified mathematical model of the process under investigation. In this way the decision making process is cast as an optimization problem described by mathematical expression(s) (Giani *et al.,* 2004). Many attempts have been made to tackle various sub-problems encountered at intermodal terminals, especially those which are mainly handling containers (Stahlbock and Voss, 2008). Different assumptions, different optimization criteria as well as the different peculiarities of each case study lead to different formulations and different solution approaches. In other words there are no "off the shelf" solutions for such logistic problems. Nevertheless model-based decision support systems (DSSs) are among the most successful tools in nowadays complex logistic environments (Power and Sharda, 2007).

The main drawback of the model-based approach is that most of the processes taking place in an intermodal Port logistics environment (scheduling, routing etc.) are in fact NP-hard optimization problems. That can be solved consistently to optimality within a reasonable amount of time only if the instance size is sufficiently small. Most of them are combinatorial programming problems, integer programming (IP) and mixed-integer programming (MIP) problems which are difficult to solve within a reasonable time window (Giani *et al.,* 2004) using exact solution methods.

Therefore in order to have a working solution within a reasonable amount of time the designer has to resort to fast heuristic and meta-heuristic algorithms. (Meta)-heuristic algorithms search for "quite good" but not necessarily the best solution (Zapfel *et al.,* 2010). For our problem two popular metaheuristic algorithms are investigated, namely the Particle Swarm Optimization (PSO) (Kennedy and Eberhart, 1995) and the Harmony Search (HS) (Geem *et al.,* 2001).

The rest of the paper is structured as follows: in Section 2 the problem is described in detail. Section 3 describes in brief the two metaheuristic optimizers involved and Section 4 presents some indicative results. Finally section 5 concludes the paper offering also some hints for future improvements.

2. PROBLEM FORMULATION AND SOLUTION MECHANISMS

As was pointed out in the introduction, the problem at hand involves the management of a fleet of trailers/trucks that have to move containers from Trieste to Fernetti or the other way around. This is a problem that belongs to the broad family of Vehicle Routing Problems (VRPs). It could be narrowed down, to the VRP with time windows and/or the VRP with pickup and delivery. An overview of these problems the interested reader can refer to (Toth and Vigo 2002 and Cordeau *et al.* 2007a; 2007b). In fact this problem looks very similar to what is known as the dial-a-ride problem (Cordeau and Laporte 2007; Coslovich *et al.*, 2006) with the only difference that the transported quantities are not humans but containers.

However, as it was pointed by (Parragh *et al.,* 2010) every specific instance of a dial-a-ride like problem comes with its own peculiarities, making difficult the use of a universal model or formulation of the problem. In our case the special features refers to (i) the possibility of different starting times for the availability of the trailers, (ii) the use of two "depots" (a trailer could originally be either at the port of Trieste or at Fernetti) and (iii) the ability to park in any of the areas after serving the last "customer" (a flavour of the open VRP). There are also: (iv) only two (interchangeable) locations for pickup and delivery, (v) the more or less constant travelling time between the two points and (vi) the more or less constant service time. On top of all these, as in almost all real life applications, the objective is a synthesis of various criteria. The above features on one hand make the specific problem unique in one sense but on the other hand make the "solution" procedure a bit easier compared to the classic dial-a-ride problems.

A possible formulation of the situation described in the previous paragraphs is the following, where we are trying to minimize the accumulated time violation of "delivery" time for all the involved containers/trailers and the total number of trucks involved:

$$\min\left(w_1 |K| + w_2 \sum_{i=1}^{N} \sum_{k\in K} \max\left(0, \left(D_{ik} + s + t - l_i\right)\right)\right) \quad (1)$$

subject to:

$$\sum_{k\in K} \sum_{j=1}^{N} y_{ijk} = 1, \quad i = 1,\dots,N \quad (2)$$

$$\sum_{j=1}^{N} y_{iuk} - \sum_{j=1}^{N} y_{ujk} = 1, \quad k \in K \quad u = 1,\dots,N \quad (3)$$

$$\sum_{j=1}^{N} y_{0jk} = 1, \quad k \in K \quad (4)$$

$$D_{jk} \geq \left(R2G_k + f\left(LocC_i, \neg LocT_k\right)\cdot t\right)\cdot y_{0jk}, \quad j = 1,\dots,N, \quad k \in K \quad (5)$$

$$D_{jk} \geq \left(D_{ik} + t + s + f\left(LocC_i, LocC_j\right)\cdot t\right)\cdot y_{ijk} \quad j = 1,\dots,N, \quad k \in K, \quad i = 1,\dots,N \quad (6)$$

$$D_{jk} \geq e_j, \; j = 1,\dots,N, \quad k \in K \quad (7)$$

where

K is the set of trucks.

$y_{ijk} = 1$ if container *i* is served before container *j*, both by vehicle *k* ($y_{0jk} = 1$ means that container *j* is the first one to be served by trailer k while $y_{i0k} = 1$ means that container *i* is the last to be served by trailer *k* after which the trailer remains at the area where it disposed container *i*).

D_{jk} is the time that truck *k* start serving container *j*.

$LocT_k = \{0,1\}, \quad k \in K$ is the location of trailer k at the beginning of the scheduling.

$R2G_k, \quad k = \in K$ is the time instance that trailer k is ready to go for its first pickup.

$LocC_i = \{0,1\}, \quad i = 1,...,n$ is the location of container i at the beginning of the scheduling.

$e_i, \quad i = 1,...,n$ is the earliest point that the container i can be moved.

$l_i, \quad i = 1,...,n$ is the latest time instance that the container i has to reach its destination.

$f(a,b) = (a \wedge b) \vee (\neg a \wedge \neg b)$, where a, b are binary variables (this function returns 1 if a is the same as b, meaning that they are both located at the same site).

w_1, w_2 are the (normalised) relative weights of the two terms comprising the cost function, with $w_1 + w_2 = 1$.

Constraints (2)-(4) are routing constraints, constraints (5),(6) ensure that a truck cannot start serving a container before delivering the previous one or before the truck is ready to move and constraint (7) ensures that a truck cannot start serving a container before the earliest time that it can be moved. In this formulation the usual constraint for the delivery of the object (container) within a delivery window has been incorporated into the objective function.

For solving the above problem one way is to assign to each container a truck and define the order by which each truck will serve them. Since the early arrival of a truck is not penalised (a truck can wait till the container becomes available for transportation), for the calculation of the starting service time we used a quite common approach in the literature that "attempts to schedule each demand at the earliest possible time after the other demands already assigned to that resource" (Beniaminy et al. 2009). In other words the truck once it is ready it goes directly to the next assigned container.

The assignment of trucks to the n containers as well as the service order is performed by the two metaheuristic methods that are described in the following section using the following representation: each solution is represented as a $1 \times (2n)$ vector with the first n elements assigning trucks to containers and the rest n elements assigning priorities to containers using an implicit representation; real values in the range [0-1] are assigned to the containers. The container with the smallest real value in the respective field should be served first etc.

For example with 4 containers (n=4) and 2 trucks (K=2) a representation of a candidate solution could be: 1 2 2 1 0.12 0.01 0.91 0.8 or if we use a matrix (Table 1):

Table 1. Solution representation

	Trucks				Priorities			
	1	2	2	1	0.12	0.01	0.91	0.08
Containers	1	2	3	4	1	2	3	4

The above means that truck no 2 will serve containers 3 and 4 in that order because 0.01<0.91 while truck 1 will serve first container 4 and then container 1. The same representation has been used in frameworks involving PSO and variations of the vehicle routing problem (Ai, J., and Kachitvichyanukul, 2009 a-c).

Building the route for each truck is performed in a sequential manner: starting from its available ready-to-go time stamp, we proceed to the first assigned request; we added the required travelling and service times. Then we continue with the next truck until all the requests are served, calculating and accumulating possible penalty terms at the same time. This way neither time ordering nor routing constraints are violated.

3. METAHEURISTICS

As pointed out in the introduction, due to the difficulty in tackling hard optimization problems with traditional methods, dedicated heuristic solution approaches have been developed that aim at providing good solutions in reasonable time for a given problem.. However, such methods have two major drawbacks: first, they are tailored to a specific problem and their adaption to other problems is difficult or even impossible. Second, they are typically designed to "build" one single solution in the most effective way, whereas most decision problems have a vast number of feasible solutions. Hence usually the chances are high that there exist better ones. To overcome these limitations metaheurirstics have been proposed (Zapfel *et al.,* 2010).

These methods have come to be recognized as ones of the most practical approaches for solving many complex problems. Metaheuristics try to find good heuristic solutions to complex optimization problems with many local optima and limited inherent structure to guide the search balancing two conflicting mechanisms: intensification vs. diversification. Intensification means that we are trying to exploit some of the properties of already visited (good) solutions whereas diversification means that we are trying to explore unvisited regions by broadening the search.

3.1 The Particle Swarm Algorithm

Eberhart and Kennedy introduced the original PSO algorithm in 1995 (Eberhart and Kennedy, 1995) that has been applied since in many fields (Piperagkas *et al.* 2012, Parsopoulos *et al.* 2009). Its main concept includes a population, called a swarm, of potential solutions, called the particles, probing the search space. The particles iteratively move in the search space with an adaptable velocity, retaining in memory the best positions they have ever visited, i.e., the positions with the best function.

The exploration capability of PSO is promoted by information exchange among particles. More specifically, each particle is assigned a (usually index--dependent) neighborhood. In the global PSO variant, also known as *gbest*, the neighborhood of each particle is the whole swarm and the overall best position is the main information provider for all particles. On the other hand, in the local PSO variant, also known as *lbest*, the neighborhoods are strictly smaller, usually consisting of a few particles. In such cases, each particle may have its own leader that influences its velocity update. Perhaps the most common neighborhood topology is the ring, where each particle assumes as neighbors its mates with neighboring indices (Parsopoulos and Vrahatis, 2010). To put it formally, consider the minimization problem:

$$\min_{x \in V \subset \Re^n} f(x) \tag{8}$$

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

Assume that g_i is the index of the best position found so far in the neighborhood of x_i i.e. $g_i = \arg\min_{j \in B_i} f(p_i)$, and let t denote the iteration counter. Then, according to the constriction coefficient version of PSO (Clerc and Kennedy, 2002), the swarm is updated as follows:

$$v_{ij}^{(t+1)} = \chi\left[v_{ij}^{(t)} + \varphi_1\left(p_{ij}^{(t)} - x_{ij}^{(t)}\right) + \varphi_2\left(p_{gij}^{(t)} - x_{ij}^{(t)}\right)\right] \tag{9}$$

$$x_{ij}^{(t+1)} = x_{ij}^{(t)} + v_{ij}^{(t)} \tag{10}$$

where *i=1,2,...,N* and *j=1,2,...,n*. The parameter χ is the constriction coefficient and it is used as a means to control the magnitude of the velocities. The other two parameters are defined as $\varphi_1 = c_1 r_1$ and $\varphi_2 = c_2 r_2$, where c_1 and c_2 are positive constants, also called the cognitive and the social parameter, respectively, and r_1, r_2, are random variables uniformly distributed in [0,1], different for each *i, j* and *t*. Based on the stability analysis of (Clerc and Kennedy, 2002), the values, χ=0.729, $c_1 = c_2 = 2.05$, are considered as the default parameter set. If $x_i^{(t+1)}$ improves the best position $p_i^{(t)}$, it replaces it in $p_i^{(t+1)}$. Otherwise, the best position remains unchanged.

3.1 The Harmony Search Algorithm

Harmony search is a metaheuristic method inspired by the musing improvising process (Geem *et al.,* 2001). It was originally developed for integer variables but since then it has been modified in order to accommodate real variables (Lee and Geem, 2005) as well as binary variables (the extreme case of an integer variable) (Wang *et al.,* 2011). While it basically mimics musicians' behaviors such as memory consideration, pitch adjustment and random consideration, the HS model has problem-specific features in every different application. The way that HS creates new solutions makes it an ideal candidate for problems where the solution vector is comprised of variables of different nature. HS is performed in several steps that are described in the rest of this section both for integer as well as real variables.

3.2.1 Harmony memory initialization

Before the application of each one of these steps, multiple solution vectors are randomly generated (or alternatively some could be provided by the user based on expert knowledge or even intuition) and stored in harmony memory (HM) as follows:

$$\mathbf{HM} = \left[\begin{array}{cccc|c} D_1^1 & D_2^1 & \cdots & D_n^1 & f\left(\mathbf{D}^1\right) \\ D_1^2 & D_2^2 & \cdots & D_n^2 & f\left(\mathbf{D}^2\right) \\ \vdots & \vdots & \cdots & \vdots & \vdots \\ D_1^{HMS} & D_2^{HMS} & \cdots & D_n^{HMS} & f\left(\mathbf{D}^{HMS}\right) \end{array}\right] \tag{11}$$

where D_i^j is the *i*-th decision variable in the *j*-th solution vector, which has one discrete value out of a candidate set $\{D_i(1), D_i(2), \ldots, D_i(k), \ldots, D_i(K_i)\}$, and $f\left(\mathbf{D}^j\right)$ is the objective function value for the *j* th solution vector, and HMS is the harmony memory size (i.e. the number of multiple vectors stored in the HM). The number of random harmonies should be at least HMS or more, such as twice or three times as many as the HM size. Then, the top-HMS harmonies are selected as starting vectors (Degertekin, 2008).

3.2.2 Improvisation of a new harmony

The vectors stored in the Harmony Memory are used to produce a new vector (a new harmony) using three operations: A) Random selection, B) Memory consideration and C) Pitch adjustment, which are presented in detail in the following paragraphs.

In random selection a new value is chosen randomly out of a candidate set with a probability (1-HMCR) (see next paragraph for the definition of HMCR):

$$D_i^{New} \leftarrow D_i(k), D_i(k) \in \{D_i(1), D_i(2), \ldots, D_i(K_i)\} \tag{12}$$

In memory consideration, one value is chosen out of the HM set, with a probability equal to the harmony memory considering rate (HMCR):

$$D_i^{New} \leftarrow D_i(l), D_i(l) \in \{D_i^1, D_i^2, \ldots, D_i^{HMS}\} \tag{13}$$

In pitch adjustment a value that has been selected in the previous step of memory consideration is further changed into neighboring values with a probability equal to the pitch adjusting rate (PAR):

$$D_i^{New} \leftarrow D_i(l \pm 1), D_i(l) \in \{D_i^1, D_i^2, \ldots, D_i^{HMS}\} \tag{14}$$

If the newly improvised harmony $\mathbf{D}^{New}$ violates any constraint, HS either abandons it, or still keeps it by adding penalty to the objective function.

3.2.3. Update of harmony memory

If the newly generated vector $\left(D_1^{New}, D_2^{New}, \ldots, D_n^{New}\right)$ is better than the worst vector in HM with respect to the objective function, the former takes the place of the latter. However, for the diversity of harmonies in HM, other harmonies (in terms of least-similarity) can be considered. Also, maximum number of identical harmonies in HM can be considered in order to prevent premature HM. After the HM update and if the maximum number of iterations or a desired performance has been reached the algorithm continues to generate new harmonies.

3.2.4 Harmony search for real valued variables

In the case of real valued variables what changes is the random selection and the pitch adjustment mechanisms.

In the random selection we randomly select a value within the admissible range of values for the corresponding variable

$$D_i^{New} \leftarrow Range_i \cdot U(0,1) \tag{15}$$

where $Range_i$ is the range of values of variable i and $U(0,1)$ is a uniform random generator between 0 and 1.

In the pitch adjustment the new value is given by:

$$D_i^{New} \leftarrow D_i(l) + bw_i\left(2 \cdot U(0,1) - 1\right),$$
$$D_i(l) \in \left\{D_i^1, D_i^2, \ldots, D_i^{HMS}\right\} \tag{16}$$

where bw_i is an arbitrary distance bandwidth for the continuous design variable

3.2.5 Accidentaling

If the new harmony $\mathbf{D}^{New}$ is the best one when compared with every harmony in HM, the new harmony can consider an additional process named *accidentaling* during which we can further pitch-adjust every note of the new harmony if it is the ever-best harmony, which may find an even better solution:

$$D_i^{New} \leftarrow \begin{cases} D_i(k \pm m), & \text{for discrete variables} \\ D_i \pm \Delta, & \text{for continuous variables} \end{cases}, \quad i = 1, \ldots, n \tag{17}$$

In other words we have the application of a local search procedure for the newly discovered best harmony

4. RESULTS

In this preliminary examination we have tested 6 different configurations corresponding to 3 different levels of requests and 2 different fleet configurations. More specifically for the requests we used i) 80 (which is the average container traffic), ii) 120 and iii) 160. For the fleet size we employed i) 5 and ii) 10 trucks.

For the case of the PSO, we used a population of 10 particles and we let the algorithm run for 2000 iterations using the *gbest* variant. For handling the integer nature of the problem after the position update (eq.10) rounding to the nearest integer is taking place.

For the case of the Harmony search, we used a memory of size 20 and in order to have comparable results (even though it was not our intention at this phase to have a thorough comparison of the two algorithms) 20000 new harmonies were "improvised".

Due to the stochastic nature of the optimization algorithms for each setup we repeated the optimization procedure 30 times. The results (using the cost function described in eq. 1 with $w_1 = w_2 = 0.5$) are summarised in Table 2, while figure 2 depicts an instance of the evolution of the best solution for the two algorithms. The pattern was similar for all runs: PSO "slowly" moved towards better solutions whereas harmony search was converging quickly towards a good local minimum (due to the use of accidentaling mechanism).

Table 2. Solution representation

Requests/trucks	Harmony search (mean/std)	PSO (mean/std)
80/5	45.53/14.83	**42.35**/14.34
80/10	5.26/~0	**5.25**/0.06
120/5	**470.60**/44.88	477.77/50.78
120/10	**32.72**/13.85	33.92/14.71
160/5	**1447.2**/130.9	1436.2/65.7
160/10	**237.46**/62.37	252.01/34.41

From Table 2, we can see that for this configuration of both algorithms, there is no clear winner. However for larger problems the HS seems to perform slightly better.

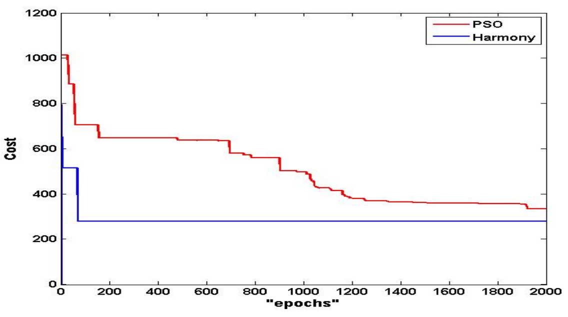

Fig. 2. The evolution of the "best" solution for PSO and Harmony search for an instance of the 160/10 problem. An "epoch" corresponds to 1 iteration of the PSO and 10 improvisations of the Harmony search.

5. CONCLUSIONS

This paper presents our preliminary results of the application of metaheuristic algorithms for solving a particular problem that arises at the Trieste-Fernetti port-dry port complex. In our approach we treated the problem in a static way. Even though this is not strictly true, due to the technologies that are being developed within the SAIL project (installation of on board computers, use of booking services etc), the information flow will allow us to provide satisfactory solutions even using the proposed approach by minimizing uncertainties. Moreover due to the fast execution of the algorithm even a complete rescheduling will be available in near-real time.

The use of metaheuristics allows for the easy modification of the cost function (other terms of the objective function could include the total travel distance of the trucks (forcing trucks to serve first units that are at the same location as they are and avoid travelling without a load), the balanced workload of trucks etc.). Moreover it allows for the quite easily substitution of the metaheuristic algorithm.

Our future work involves appropriate fine-tuning of the involved algorithms and testing of other metaheuristic search methods as well as the development of a (semi)automated method for setting the weights of the optimization function based for example on the analytic hierarchy process (Saaty, 1990).

Acknowledgement

This work was supported by the E.U. FP7–PEOPLE–IAPP–2009, Grant Agreement No. 251589, Acronym: SAIL.

REFERENCES

Ai, J., and Kachitvichyanukul, V. (2009.a). A particle swarm optimization for the vehicle routing problem with simultaneous pickup and delivery. *Computers & Operations Research,* 36, 1693-1702.

Ai, J., and Kachitvichyanukul, V. (2009.b). Particle swarm optimization and two solution representations for solving the capacitated vehicle routing problem. *Computers & Industrial Engineering,* 56 (1), 380-387.

Ai, J., and Kachitvichyanukul, V. (2009.c). A Particle Swarm Optimisation for Vehicle Routing Problem with Time Windows. *International Journal of OR*, 6 (4), 519-537.

Beniaminy, I., Yellin, D., Zahavi, U., and Žerdin, M. (2009). When the Rubber Meets the Road: Bio-inspired Field Service Scheduling in the Real World. In *Bio-inspired Algorithms for the Vehicle Routing Problem*, 191-213.

Boschian, V., Dotoli, M., Fanti, M. P., Iacobellis, G., and Ukovich, W. (2011). A Metamodeling Approach to the Management of Intermodal Transportation Networks. *IEEE Transactions on Automation Science and Engineering*, 8(3), pp. 457-469.

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

Cordeau, J. F., and Laporte, G. (2007). The dial-a-ride problem: models and algorithms. *Annals of OR*, 153(1), 29-46.

Cordeau, J.-F., Laporte, G., Potvin, J.-Y., and Savelsbergh, M. W. P. (2007a). Transportation on demand. In C. Barnhart and G. Laporte (Eds.), *Transportation.* Amsterdam: Elsevier.

Cordeau, J.-F., Laporte, G., Savelsbergh, M. W. P., and Vigo, D. (2007b). *Vehicle routing.* In C. Barnhart and G. Laporte (Eds.), *Transportation*. Amsterdam: Elsevier.

Coslovich, L., Pesenti, R., & Ukovich, W. (2006). A two-phase insertion technique of unexpected customers for a dynamic dial-a-ride problem. *European Journal of Operational Research*, 175(3), 1605-1615.

Degertekin, S. O. (2008). Optimum design of steel frames using harmony search algorithm. *Structural and Multidisciplinary Optimization*, 36(4), 393-401.

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

Giani, G. Laporte, G. and Musmanno, R. (2004*) Introduction to Logistics Systems Planning and Control*, Willey, 2004

Kennedy, J. and, Eberhart, R.C. (1995). Particle swarm optimization. *In IEEE International Conference on Neural Networks*, Perth, Australia, 1942–1948.

Lee, K. S., and Geem, Z. W. (2005). A new meta-heuristic algorithm for continuous engineering optimization: harmony search theory and practice. *Computer methods in applied mechanics and engineering*, 194(36), 3902-3933.

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

Parsopoulos, K. E., Kariotou, F., Dassios, G., and Vrahatis, M. N. (2009). Tackling Magnetoencephalography with Particle Swarm Optimization, *International Journal of Bio-Inspired Computation*, 1 (1/2), 32-49.

Parragh, S. N., Doerner, K. F., & Hartl, R. F. (2010). Variable neighborhood search for the dial-a-ride problem. *Computers & OR*, 37(6), 1129-1138.

Piperagkas, G. S., Georgoulas, G., Parsopoulos, K. E., Stylios, C. D., and Likas, A. C. (2012). Integrating particle swarm optimization with reinforcement learning in noisy problems. *In GECCO2012*. 65-72.

Power, D.J., and, Sharda. R. (2007) Model-driven decision support systems: Concepts and research directions. *Decision Support Systems,* 43(3): 1044-1061.

Saaty, T. L. (1990). How to make a decision: the analytic hierarchy process. *European journal of OR*, 48(1), 9-26.

Stahlbock, R., and Voss, S. (2008). Operations research at container terminals: A literature update. *OR Spectrum*, 30(1), 1–52.

Toth, P., and Vigo, D. (2002). *The vehicle routing problem, SIAM monographs on discrete mathematics and application*s. Society for Industrial and Applied Mathematics.

Turban, T., Sharda, R., and Delen, D. (2010). *Decision Support and Business Intelligence Systems*. Prentice Hall.

Wang, L., Mao, Y., Niu, Q., & Fei, M. (2011). A multi-objective binary harmony search algorithm. *Advances in Swarm Intelligence*, 74-81.

Zapfel, G., Braune, R. and. Bogl, M. Metaheuristic *Search Concepts to Production and Logistics. A Tutorial with Applications*, Springer, 2010