Available online at www.sciencedirect.com

ScienceDirect

Expert Systems with Applications 00 (2015) 1–16

Syst Appl.logo

Journal Logo

www.elsevier.com/locate/procedia

A BRILS METAHEURISTIC FOR NON-SMOOTH FLOW-SHOP PROBLEMS WITH FAILURE-RISK COSTS

A. Ferrer[a,1,*], D. Guimarans[b], H.Ramalhinho[c], A.A. Juan[d,**]

[a]*Department of Applied Mathematics, Technical University of Catalonia, Barcelona, Spain*
[b]*Optimisation Research Group National ICT Australia (NICTA), Sydney, Australia*
[c]*Department of Economics and Business, Universitat Pompeu Fabra, Barcelona, Spain*
[d]*Computer Science Department - Internet Interdisciplinary Institute, Open University of Catalonia, Barcelona, Spain*

Abstract

This paper analyzes a realistic variant of the permutation flow-shop problem (PFSP) by considering a non-smooth objective function that takes into account not only the traditional makespan cost but also failure-risk costs due to uninterrupted operation of machines. After completing a literature review on the issue, the paper formulates an original mathematical model to describe this new PFSP variant. Then, a Biased-Randomized Iterated Local Search (BRILS) algorithm is proposed as an efficient solving approach. An oriented (biased) random behavior is introduced in the well-known NEH heuristic to generate an initial solution. From this initial solution, the algorithm is able to generate a large number of alternative good solutions without requiring a complex setting of parameters. The relative simplicity of our approach is particularly useful in the presence of non-smooth objective functions, for which exact optimization methods may fail to reach their full potential. The gains of considering failure-risk costs during the exploration of the solution space are analyzed throughout a series of computational experiments. To promote reproducibility, these experiments are based on a set of traditional benchmark instances. Moreover, the performance of the proposed algorithm is compared against other state-of-the-art metaheuristic approaches, which have been conveniently adapted to consider failure-risk costs during the solving process. The proposed BRILS approach can be easily extended to other combinatorial optimization problems with similar non-smooth objective functions.

Keywords: Heuristic algorithms, biased randomization, iterated local search, scheduling, flow-shop, non-smooth objective functions.
2000 MSC: 65K05, 90C26, 90C27, 90C59

1. Introduction

Combinatorial Optimization Problems (COPs) have a large number of applications in many industries, from service to manufacturing, and from public to private sectors. For this reason, a large number of studies

*Principal corresponding author
**Corresponding author
Email addresses: alberto.ferrer@upc.edu (A. Ferrer), Daniel.Guimarans@nicta.com.au (D. Guimarans), helena.ramalhinho@upf.edu (H.Ramalhinho), ajuanp@uoc.edu (A.A. Juan)

[1]The research of this author was partially supported by Spanish Ministry of Economy and Competitiveness under grants MTM2011-29064-C03-02 & MTM2014-59179-C2-01

and scientific work have been dedicated to COPs. Solving a COP means finding an optimal or near-optimal solution among a finite (and usually extremely large) set of feasible solutions that represent combinations of several elements (Papadimitriou & Steiglitz, 1982). Examples of COPs are: scheduling problems (such as the Flow-Shop or Job-Shop Scheduling Problems); routing problems (such as the Traveling Salesman or the Vehicle Routing Problems); assignment problems (such as the Generalized Assignment or the Quadratic Assignment Problems); etc. Most of these problems can usually be formulated using Mixed Integer Linear Programming (MILP) models and are NP-complete (Freeman, 1979). Typically, however, the classical MILP methods can only solve small to medium instances in reasonable computing times, while heuristics and metaheheuristics are required to solve medium to large instances.

A good method to solve COPs should have the following characteristics (Cordeau et al., 2002): accuracy, simplicity, efficiency, robustness, and flexibility. These characteristics enable the method to solve complex and large problems. The accuracy is related to the quality of results; simplicity of design and implementation considers the avoidance of complex and time-consuming fine-tuning processes; efficiency is related to the computational time employed and quality of the generated results; robustness means lower dependency on inputs and constraint changes; finally, flexibility is the ability to deal with general combinatorial optimization problems in different scenarios.

In this work, we focus on COPs with non-smooth objective function (NSOF-COPs) so the model is formulated using continuous variables. The presence of non-smooth objective functions increases the difficulty of solving these combinatorial optimization problems. The lack of a well-defined structure, such as gradient information, leads to issues in applying traditional mathematical tools. Therefore, studying the development of new methods to solve NSOF-COPs is an important area to explore and, moreover, it has not been covered extensively by the literature so far. In particular, due to the complexity of these problems, heuristic and metaheuristic methods seem among the most appropriate techniques for solving NSOF-COPs. Local-search heuristics might be able to find several local optimal solutions in different parts of the feasible region. This characteristic enables these methods to find good solutions, close to the optimal one, even in the case of NSOF-COPs.

Most of the articles on the Permutation Flow-Shop Problem (PFSP) consider a makespan-based objective function. This classical version of the PFSP can be formulated as a MILP model (Pinedo, 2008). However, in real practice other functions are usually employed to measure the quality of the solutions. In particular, we consider a variant of the PFSP with a non-smooth objective function that takes into account not only the makespan cost but also failure-risk cost functions due to uninterrupted operation of machines. This last cost is associated with the running time of each machine. Notice that solutions with an extremely short makespan might imply long uninterrupted operation times of a machine, which can lead to higher failure-risk costs.

The present work contributes to the field of Expert and Intelligent Systems in several ways. The first one is the consideration of a more realistic scheduling problem that incorporates a failure-risk penalty cost functions. A second contribution is the mathematical formulation for this non-smooth version of the PFSP. As far as we know, there is a lack of publications considering realistic non-smooth cost functions in scheduling problems. Finally, another contribution is the development of a solving method for this non-smooth PFSP that combines Biased Randomization with an Iterated Local Search (BRILS) metaheuristic framework. The method pertains to the class of nondeterministic or stochastic algorithms and relies on the use of oriented (biased) random sampling, i.e., it makes use of a skewed (non-symmetric) probability distribution in order to introduce some bias in the generation of random values to better guide the random search. This approach can be a natural and efficient way to deal with other realistic PFSP under more complex scenarios dominated by non-smooth objective functions.

The structure of the paper is as follows: In Section 2 we present a review of non-smooth combinatorial optimization problems and probabilistic algorithms. Section 3 describes in detail the PFSP with non-smooth objective functions. The BRILS algorithm is presented in Section 4, followed by some computational experiments and comparative studies in sections 5 and 6, respectively. Finally, conclusions and future work are described in section 7.

2. Review on non-smooth combinatorial optimization problems and probabilistic algorithms

In this section, we review the work done in probabilistic algorithms for solving COPs. One of the most popular solution methods applied to COPs are metaheuristics, including probabilistic or randomized algorithms. These algorithms use pseudo-random numbers or random variates during the constructive and local search phases of the algorithm. An important characteristic of these algorithms is that, for the same set of inputs, the algorithm is likely to produce different outputs in different runs —unless, of course, the same simulation seed is used in these runs. Therefore, when properly designed, these algorithms can explore the solution space in a quite extensive way, thus finding many local optimal solutions. These algorithms include, among others: Genetic and Evolutionary Algorithms (Fleurent & Glover, 1999; Hemamalini & Simon, 2010; Davis, 1991; De Jong, 2006; Reeves, 2010), Simulated Annealing (Reeves, 1995; Suman & Kumar, 2006; Nikolaev & Jacobson, 2010), GRASP (Festa & Resende, 2009a,b; Feo & Resende, 1995; Resende & Ribeiro, 2010), Variable Neighborhood Search (Hansen et al., 2010), Iterated Local Search (Lourenço et al., 2003, 2010), Ant Colony Optimization (Dorigo & Stützle, 2010), Probabilistic Tabu Search (Lokketangen & Glover, 1998; Gendreau et al., 1994), and Particle Swarm Optimization (Kennedy & Eberhart, 1995). For a detailed review of randomized algorithms the reader is referred to (Collet & Rennard, 2006).

Probabilistic algorithms have been widely applied to solve several classical COPs: Sequencing and Scheduling Problems (Funke et al., 2005; Gendreau et al., 1994; Pinedo, 2008); Vehicle Routing Problems (Laporte, 2009), Quadratic and Assignment Problems (Loiola et al., 2007; Oncan, 2007; Fisher & Jaikumar, 1981); Location and Layout Problems (Drezner & Hamacher, 2002; Mester & Bräysy, 2007); Covering, Clustering, Packing and Partitions Problems (Chaves & Lorena, 2010; Muter et al., 2010), etc.

The interest in solving scheduling problems with more realistic assumptions has increased within the research community in the last years. The gap between theoretical and realistic flow-shop scheduling problems is discussed in Ruiz & Stützle (2008). Their work also presents some mathematical models and heuristics to solve realistic problems. Some recent examples of relevant works regarding realistic flow-shop scheduling problems are briefly reviewed next. In Ribas et al. (2015), a Discrete Artificial Bee Colony algorithm is proposed to solve the blocking PFSP with a flow-time criterion. Li & Pan (2015) describe a realistic large-scale hybrid PFSP problem with limited buffers and present a hybrid metaheuristic based on Tabu Search and Artificial Bee Colony Systems. Li et al. (2015) propose a discrete teaching-learning-based optimization method to rescheduling a PFSP taking into consideration realistic incidents as machine breakdown, new job arrival, cancellation of jobs, job processing variation, and job release variation. They consider a multi-objective environment, taking into account as objective function the makespan and an instability function. Sun et al. (2013) describe a general permutation PFSP where the processing time of a job is defined by a general non-increasing function of its scheduled position. They present a study with several objective functions as makespan, total completion time, total weighted completion time, etc. To solve these problems, they present an approximation algorithm based on solving a one-machine scheduling problem. All these articles consider classical objective functions as, for example, the makespan or other related completion times. As a novelty, the problem considered in this paper includes a non-smooth objective function representing not only the makespan but also the failure-risk costs.

With respect to the NSOF-COPs, only a few research publications regarding the application of probabilistic algorithms can be found. We performed an intensive search in the Thomson Reuters Web of Science (http://wokinfo.com/) and, after that, we reviewed the few papers found whenever they were related to the present work.

Within the Communication Systems knowledge area, the non-smooth optimization is quite relevant. Chiang (2009) presents an overview of some of the important non-smooth optimization problems in point-to-point and networked communication systems. For instance, Hamdan & El-Hawary (2002) and Oonsivilai et al. (2009) consider the Optimal Routing problem in Communication Networks. The objective is to find the best path for data transmission in a short amount of time. Hamdan & El-Hawary (2002) describe a Genetic Algorithm combined with Hopfield Networks to solve this problem, and Oonsivilai et al. (2009) present a Tabu Search Method. However, both methods lack the capacity to obtain good solutions in a reasonable time period.

Other applications can be found in the area of Clustering Problems. Bagirov & Yearwood (2006) de-

scribe a non-smooth Minimum Sum-of-Squares Clustering Problem and several solution approaches based on dynamic programming, K-Means algorithms, and branch-and-bound methods. The authors conclude that all methods have significant limitations and can only be applied in certain special settings. For example, a dynamic programming and branch-and-bound approach is successful only in small instances, while K-means algorithms can solve large instances, although their efficiency decreases rapidly as the size of the cluster increases. Al-Sultan (1995) and Selim & Al-Sultan (1991) present a Tabu Search and a Simulated Annealing algorithm, respectively, to solve the aforementioned problem. Both methods require a set of parameters and, therefore, a complex fine-tuning process. Most of these authors agree that the best approach to solve these problems is by using metaheuristic methods.

COPs with non-smooth functions have not been investigated in detail so far. Juan et al. (2010) propose a biased-randomized local search to solve the non-smooth Vehicle Routing Problem (VRP). Yang & Chou (2011) consider a multi-objective non-smooth manpower assignment problem and propose a Particle Swarm method to solve it. Vaisakh & Srinivas (2011) consider a non-smooth power flow problem and propose an Ant Colony and Evolutionary algorithm to solve it. Also, an extension of Ant Colony Optimization was proposed by Schlüter et al. (2009) to solve non-convex Mixed Integer Problems.

In the case of realistic PFSPs, it is natural to consider non-smooth objective functions that include failure-risk costs together with the classical makespan cost. In the following section we describe in detail this scheduling problem. The proposed method to solve this problem follows the structure of an Iterated Local Search (ILS) but including a Biased Randomization (BR) procedure in the initial solution generation. We designated this method as Biased-Randomized Iterated Local Search (BRILS). ILS is known as one of the most simple and efficient metaheuristic (Burke et al., 2010), and BR introduces a random element that favors the promising regions of the solution space. As mentioned before, BRILS has the four good characteristics of a good heuristic method (Cordeau et al., 2002). Lourenço et al. (2003, 2010) preset a detailed review on ILS. The initial solution generation method is the well-known heuristic from Nawaz, Enscore, and Ham (NEH) (Nawaz et al., 1983), but considering a biased (oriented) randomization process that uses a descending triangular probability distribution. This approach has been successfully applied to the PFSP with a makespan objective function (Juan et al., 2014). We also refer to this last work for an extensive description of the PFSP. Also, BRILS is inspired in a similar approach successfully applied to Routing Problems with a non-smooth objective function (Juan et al., 2010).

Comparing the present work with the previously reviewed literature, a new PFSP version, which includes a non-smooth objective function, is presented and solved. This new version is more realistic than only considering makespan since, in real practice, machines (equipment, resources etc.) can fail if working for a long time period without breaks.

3. The non-smooth version of PFSP with failure-risk penalty functions (NSPFSP)

The PFSP is a well-known scheduling problem and one of the most important problems in the area of production management. It can be briefly described as follows: There are a set of m machines and a set of n jobs. Each job comprises a set of m operations which must be done on different machines. All jobs have the same processing operation order when passing through the machines. There are no precedence constraints among operations of different jobs. Operations cannot be interrupted and each machine can process only one operation at a time. The most popular goal in this problem relates to find the job sequences on the machines which minimize the completion times of all operations, or makespan, denoted by $C_{\max}$. Figure 1 illustrates this problem for a simple case of 3 jobs and 4 machines. Nevertheless, there are some aspects in real applications that should be taken into account when applying these models and methods to a real problem. Most of the flow-shop problems arise in production industries, where a set of products or jobs must be produced. Then, a more realistic formulation of the PFSP must consider the cost of failure-risk penalties due to the lack of breaks. This new formulation of the flow-shop problem is described as the Non-Smooth Permutation Flow-Shop Problem (NSPFSP). In this section we discuss these aspects and propose a modification of the objective function of the classical PFSP by adding failure-risk penalties, that represent in a closer way the real objective function when scheduling the processing of jobs by the machines in a workshop. Both the PFSP and the NSPFSP are combinatorial and NP-complete (Rinnooy Kan, 1973).

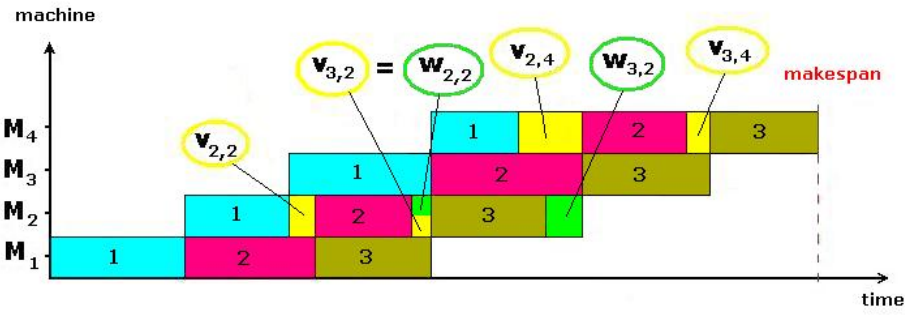

Fig. 1. Flowshop Sequencing Problem.

3.1. The Non-Smooth Permutation Flow-Shop Problem

To describe the classical version of the PFSP we use the sets:

a) a set of n jobs J_i, $i \in \mathcal{J} := \{1, 2, \ldots, n\}$,
b) a set of m machines M_k, $k \in \mathcal{M} := \{1, 2, \ldots, m\}$,
c) for all $(i, k) \in \mathcal{J} \times \mathcal{M}$, $p_{i,k}$ is the processing time of the job J_i on the machine M_k,

and the following sets of variables for all $(i, k) \in \mathcal{J} \times \mathcal{M}$:

d) $v_{i,k}$ is the waiting time on machine k before the start of job in position i,
e) $w_{i,k}$ is the waiting time of the job in position i while waiting for machine $k + 1$ to become free, after finishing processing on machine k.

Additionally, we define the following binary decision variables:

$$x_{i,j} = \begin{cases} 1, & \text{if the job } j \text{ is assigned to position } i, \\ 0, & \text{otherwise,} \end{cases}$$

for all $i,\ j \in \mathcal{J}$, defining the assignment matrix,

$$X := \begin{pmatrix} x_{1,1} & x_{1,2} & \ldots & x_{1,j} & \ldots & x_{1,n} \\ x_{2,1} & x_{2,2} & \ldots & x_{2,j} & \ldots & x_{2,n} \\ \ldots & \ldots & \ldots & \ldots & \ldots & \ldots \\ x_{i,1} & x_{i,2} & \ldots & x_{i,j} & \ldots & x_{i,n} \\ \ldots & \ldots & \ldots & \ldots & \ldots & \ldots \\ x_{n,1} & x_{n,2} & \ldots & x_{n,j} & \ldots & x_{n,n} \end{pmatrix}$$

Then, in the classical version of PSFP, we want to minimize the objective function,

$$C_{\max}(X) := \sum_{i=1}^{n} \left(v_{i,m} + \sum_{j=1}^{n} p_{j,m} x_{i,j} \right), \tag{1}$$

subject to the constraints,

$$\forall i \in \mathcal{J},\ \sum_{j=1}^{n} x_{i,j} = 1. \tag{2}$$

$$\forall j \in \mathcal{J},\ \sum_{i=1}^{n} x_{i,j} = 1. \tag{3}$$

$$\forall (i,j) \in \mathcal{J} \times \mathcal{J},\ x_{i,j} * (1 - x_{i,j}) = 0. \tag{4}$$

$$\forall k \in \mathcal{M} - \{m\},\ w_{1,k} = 0. \tag{5}$$

$$\forall j \in \mathcal{J} - \{1\},\ v_{j,1} = 0. \tag{6}$$

$$\forall k \in \mathcal{M} - \{1\},\ v_{1,k} = \sum_{r=1}^{k-1} \sum_{i=1}^{n} p_{i,r} x_{1,i}. \tag{7}$$

$$\begin{aligned} &\forall i \in \mathcal{J} - \{n\},\ \forall k \in \mathcal{M} - \{m\}, \\ &v_{i+1,k} + \textstyle\sum_{j=1}^{n} p_{j,k} x_{i+1,j} + w_{i+1,k} = w_{i,k} + \sum_{j=1}^{n} p_{j,k+1} x_{i,j} + v_{i+1,k+1}. \end{aligned} \tag{8}$$

Remark 1. Notice that the optimization function (1) is defined in the set of the square matrices of order n on the set of the real numbers, $\mathrm{M}_n(\mathbb{R})$, that can be seen as equivalent to the set $\mathbb{R}^{n^2}$. Thus, we have the objective function,

$$C\max : \mathbb{R}^{n^2} \mapsto \mathbb{R},$$

and the constraints (2), (3) and (4) (which determine the permutation, X, to be processed) are described by continuous variables. Actually, $X \in \mathrm{P}_n \subset \mathrm{M}_n(\mathbb{R})$, where P_n is the set of permutation matrices.

Remark 2. Constraints (5) describe that the job in position 1 never waits for the next machine. Constraints (6) describe that the waiting time of the first machine for all jobs is zero. Constraints (7) ensure that the waiting time on machine M_k for the job in the first position is the sum of the processing times for that job in all previous $k-1$ machines. Finally, constraints (8) can be seen as *flow conservation* constraints as it is explained in Example 1.

Example 1. For $i = 2$ and $k = 2$ in Figure 1 and from constraints (8), the corresponding flow conservation constraint is,

$$v_{3,2} + p_{3,2} + w_{3,2} = w_{2,2} + p_{2,3} + v_{3,3},$$

which graphically is displayed in Figure 2.

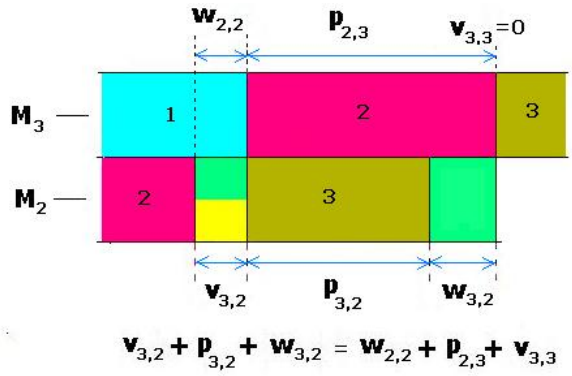

Fig. 2. Flow conservation constraint for job $i = 2$ and machine $k = 2$.

In the article we want to propose a more realistic formulation of the Flow-Shop Problem that takes into account the failure-risk cost function, $\Psi^k(X)$ on every machine $k \in \mathcal{M}$, i.e. the cost of the risk of failures due to the lack of breaks, which depends of the waiting time variables associated to the current permutation, $X \in \mathrm{P}_n$, and that will be described in the next subsection 3.2. The failure-risk total cost function is the sum of failure-risk cost functions for each machine, $\sum_{k\in\mathcal{M}} \Psi^k(X)$. Thus, the extended flow shop scheduling problem with failure-risk total cost function, named NSPFSP, will be described as a non-smooth program (NSOF-COPs) that minimizes the objective function,

$$\mathrm{TCost}(X) := C_{\max}(X) + \sum_{k\in\mathcal{M}} \Psi^k(X), \text{ where } X \in \mathrm{M}_n(\mathbb{R}), \tag{9}$$

subject to the constraints (2), (3), (4), (5), (6), (7) and (8), which define the feasibility set of the program.

3.2. *Evaluating the failure-risk cost functions,* Ψ^k.

Given a permutation, $X \in \mathrm{P}_n \subset \mathrm{M}_n(\mathbb{R}) \equiv \mathbb{R}^{n^2}$, the function, ϕ^k (time weighted), describes the risk of machines' breakdown. It penalizes the time that a machine, M_k, $k \in \mathcal{M}$, is operating continuously during a time T after a pause. The time T is named *period time* in this article. The function ϕ^k depends on parameter τ and the period time T. Without lack of generality, we will consider,

$$\tau := \max\{p_{j,i} : j \in \mathcal{J}, i \in \mathcal{M}\},$$

where τ is the maximum processing time over all jobs and all machines. For the numerical experiments carried out in this paper, the function $\phi^k(\tau, T)$, for every machine M_k, during the period of time T is defined as follows:

$$\phi^k(\tau, T) := \begin{cases} 0 & : & 0 & < & T & \leq & \tau, \\ 0.2T & : & \tau & < & T & \leq & 2\tau, \\ 0.3T & : & 2\tau & < & T & \leq & 3\tau, \\ 0.5T & : & 3\tau & < & T. & & \end{cases} \tag{10}$$

For obtaining the period times associated to each machine, M_k, $k \in \mathcal{M}$ (given a permutation, $X \in \mathrm{P}_n$) we define the set of indexes $\mathcal{M}^k$ in the form:

$$\mathcal{M}^k := \{1, n+1\} \cup \{i \in \mathcal{J} : v_{i,k} \neq 0\} = \{i_1 = 1 < i_2 < \cdots < i_s = n+1\}.$$

Then, for all $t = 2, 3, \ldots, s$ we consider the period time

$$T_t^k := p_{i_{(t-1)},k} + \cdots + p_{i_t-1,k}, \tag{11}$$

and calculate $\phi_t^k(\tau, T_t^k)$.

Example 2. Figure 3 illustrates how time periods are calculated for a set of 8 processed at machine k. From

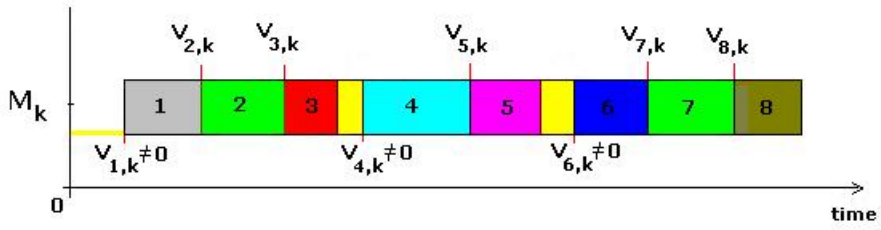

Fig. 3. Calculation of the time periods for a set of 8 jobs processed at machine k

Figure 3 the set of indices for machine k is

$$\mathcal{M}^k = \{i_1 = 1, i_2 = 4, i_3 = 6, i_4 = 9\},$$

with $t = 2, 3, 4$. Then, by using the expression (11), the time periods for machine k are calculated as follows:

$$\begin{aligned} T_2^k &= p_{1,k} + p_{2,k} + p_{3,k}, \\ T_3^k &= p_{4,k} + p_{5,k}, \\ T_4^k &= p_{6,k} + p_{7,k} + p_{8,k}. \end{aligned}$$

Nevertheless, we are working in the space $\mathrm{M}_n(\mathbb{R}) \equiv \mathbb{R}^{n^2}$ so the function ϕ_t^k must be extended to this space of continuous variables in the form,

$$\Phi_t^k(X) := \begin{cases} \phi_t^k(\tau, T_t^k) & : \quad X \in \mathrm{P}_n \subset \mathrm{M}_n(\mathbb{R}), \\ +\infty & : \quad X \notin \mathrm{P}_n \subset \mathrm{M}_n(\mathbb{R}), \end{cases} \tag{12}$$

which is a non-smooth and non-convex function. Then, given $X \in \mathrm{M}_n(\mathbb{R}) \equiv \mathbb{R}^{n^2}$, the failure-risk cost function (penalty function), Ψ^k, for each machine, M_k, $k \in \mathcal{M}$, is the sum of the corresponding extended failure-risk costs functions, normalized by the number of machines m to avoid an undesired dominance of failure-risk costs over makespan,

$$\Psi^k(X) := \sum_{t=1}^{s} \frac{\Phi_t^k(X)}{m}, \tag{13}$$

which are, non-smooth and non-continuous functions. This result implies that the objective function in (9) is also a non-smooth and non-continuous function.

4. Hybridizing an ILS metaheuristics with Biased Randomization of classical heuristics

The most common approaches to solve the PFSP are heuristics and metaheuristics, in part due to the difficulty of solving medium- and large-scale instances in reasonable computational times. One of the most popular and best performing heuristics was introduced by Nawaz et al. (1983), and it is known as the NEH heuristic. The main idea behind this heuristic is to schedule the jobs following the 'common sense' rule of prioritizing those with the highest total processing time. First of all, it calculates the total time each job requires to be processed by all machines; then, it creates an 'efficiency list' of jobs sorted in descending order according to this total processing time. At each step, the first job on the list is selected, removed from the list, and added to the partial solution. This selected job is inserted in a partial solution following a 'shift-to-left' movement with the objective of minimizing the makespan. The heuristic finishes when all jobs have been scheduled, i.e. the list is empty. Taillard (1993) introduced a data structure that reduces the NEH complexity and speeds up the computation. The NEH heuristic is the most used heuristic to obtain initial solutions for the PFSP within metaheuristics and exact algorithms (Companys & Mateo, 2007; Ladhari & Haouari, 2005).

Several authors have proposed metaheuristics to solve the PFSP. Osman & Potts (1989) used Simulated Annealing (SA). Widmer & Hertz (1989) proposed a Tabu Search (TS) algorithm known as SPIRIT. The NEH heuristic is also used by other Tabu Search algorithms (Moccellin, 1995; Reeves, 1993). Several Genetic Algorithms are also based on the NEH heuristic (Aldowaisan & Allahvedi, 2003; Chen et al., 1995; Reeves, 1995). Likewise, NEH has also been used in Ant Colony Optimization algorithms (Rajendran & Ziegler, 2004). In order to measure the efficiency of these algorithms, most authors use Taillard's benchmark instances (Taillard, 1993). Finally, some of the aforementioned algorithms can be adapted to other more realistic environments (Ruiz & Stützle, 2007). In this paper, we propose the Biased-Randomized Iterated Local Search (BRILS) algorithm, which extends the ILS-ESP algorithm proposed in Juan et al. (2014) by incorporating the failure-risk penalties introduced in our mathematical model. Figure 4 shows a flow chart describing the main steps of our BRILS algorithm. First, an initial solution is generated based on a biased-randomized version of the NEH heuristic. The biased-randomization process is explained in detail

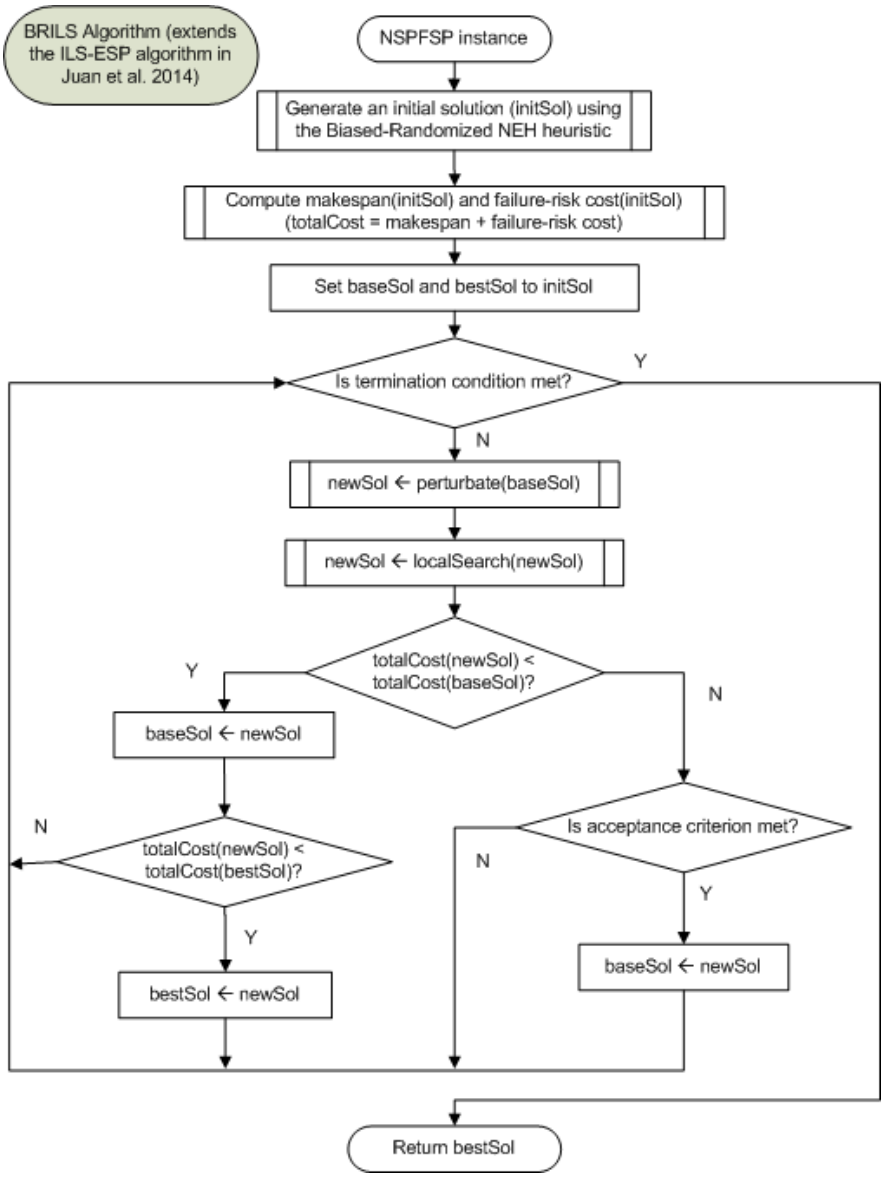

Fig. 4. BRILS flow chart

in Juan et al. (2014). Its main purpose is to generate a different starting solution –of similar quality to the one provided by the NEH heuristic itself–, each time the algorithm is run. This has a positive effect –in terms of the clock time necessary to obtain a near-optimal solution– when the algorithm uses basic parallelization strategies (Juan et al., 2014). Biased randomization of classical heuristics is a technique that has been employed to provide efficient methods for other COPs. For this initial solution, both makespan and total costs (including failure-risk penalties) are computed. Then, the base and best solutions are set to be the initial solution and an iterated improvement process is started. This process is usually controlled by a maximum time or by a maximum number of iterations. At each iteration, the following steps are carried out:

Step 1. A new solution is generated by using a perturbation process such as the one described in Juan et al.

(2014) or the one in Ruiz & Stützle (2007).

Step 2. The new solution is improved using a local search process such as the one described in the aforementioned papers. During this local search process, the makespan of the new solution is computed. This computation is extremely fast if the aforementioned Taillard's data structures are used. Moreover, the failure-risk cost associated to the new solution is also computed as described in section 3.2.

Step 3. At this point, if the total cost associated with the new solution (including not only the makespan but also the failure-risk cost) is better than the one associated with the base solution, the latter is updated to the former. Additionally, whenever the total cost of the new solution improves the one associated with the best-so-far solution, the latter is updated to the former.

Step 4. In the case the total cost of the new solution is higher than the cost of the base solution, an acceptation criterion is employed to decide whether to update the base solution or not. This acceptation criterion can be based on a Simulated Annealing approach as in Ruiz & Stützle (2007), or in a Demon approach as in Juan et al. (2014). The reason why, from time to time, it might be interesting to update the base solution to a worse solution is to reduce the probabilities of the algorithm getting trapped in a local minimum.

Finally, once the maximum allowed time or the maximum number of iterations has been met, the algorithm returns the best-found solution, i.e., the solution with the lowest total cost among the ones explored.

5. Numerical experiments

In this section we present the results of a series of numerical experiments designed to compare the two versions of the PFSP considered in this article. The first version, named MKSP1, is the classical version in which only the makespan cost is considered during the solving process. In other words, the solving approach does not consider failure-risk costs due to uninterrupted operation of machines. Only after the solving process has finished and a minimum-makespan solution has been provided, the failure-risk cost (in time units) is added to the makespan cost of this solution. The second version, named MKSP2, takes into account the failure-risk cost during the solving process itself, i.e., our algorithm searches the solution with the minimum total cost, including both the failure-risk cost (as described by the penalization function, $\sum_{k\in\mathcal{M}}\Psi^k$) and the classical makespan cost, $C_{\max}$.

The algorithm described in this paper was implemented as a Java application. The numerical experiments were carried out on a desktop PC, SO: Ubuntu 14.04, CPU: i7 3.4GHz, RAM: 8 Gb. The 120 Taillard's benchmark instances (Taillard, 1993) were used in our experiments. These instances are grouped in 12 sets of 10 instances each according to the number of jobs and the number of machines, i.e., set tai20-5, set tai20-10, set tai20-20, set tai50-5, set tai50-10, set tai50-20, set tai100-5, set tai100-10, set tai100-20, set tai200-10, set tai200-20, and set tai500-20. For each tested instance, 10 independent iterations (replicas) were run. Each replica was run for a maximum time $t_{max} = 0.01 second \cdot jobs \cdot machines$. Then, for each set of replicas, the best experimental solution found (BEST10), as well as the average value of the different replicas (AVG10) were registered.

Table 1 shows, for each set of benchmarks, the average of the best solutions found in MKSP1 and MKSP2. This table has four main column fields. The first field, named Set, refers to the name of the benchmarks set. Fields two (*MKSP1*) and three (*MKSP2*) refer to the average of the best results obtained for each of these two versions of the PFSP. The field *Gap (%)* contains the corresponding gaps of the results in the aforementioned fields two and three. Results in fields *MKSP1*, *MKSP2*, and *Gap (%)* are subdivided into three other subfields which refer, respectively, to the makespan cost ($C_{\max}$), failure-risk cost ($\sum_{k\in\mathcal{M}}\Psi^k$), and total cost (*TCost*).

From Table 1 it can be noticed that even for relatively moderated penalty functions as the one used in this paper (with makespan costs exceeding by far risk costs), whenever failure-risk costs are considered in the objective function it pays off to develop procedures that take these costs into account during the solving process instead of at the end of it. In our numerical example, the results show an average gap of -1.14%, i.e. on average, the MKSP2 approach outperforms in total cost the MKSP1 approach by that percentage.

	MKSP1			*MKSP2*			*Gap (%)*		
Set	$C_{\max}$	$\sum_{k\in\mathcal{M}}\Psi^k$	*TCost*	$C_{\max}$	$\sum_{k\in\mathcal{M}}\Psi^k$	*TCost*	$C_{\max}$	$\sum_{k\in\mathcal{M}}\Psi^k$	*TCost*
tai20-5	1,222.4	383.6	1,606.0	1,240.7	305.3	1,546.0	1.54%	-19.47%	-3.67%
tai20-10	1,514.3	299.2	1,813.5	1,539.0	231.0	1,770.0	1.64%	-22.28%	-2.35%
tai20-20	2,236.1	220.1	2,456.2	2,249.3	190.4	2,439.7	0.59%	-13.15%	-0.66%
tai50-5	2,736.4	1,084.0	3,820.4	2,746.0	1,016.9	3,762.9	0.35%	-6.24%	-1.52%
tai50-10	3,004.6	980.1	3,984.7	3,043.0	860.4	3,903.4	1.28%	-12.22%	-2.04%
tai50-20	3,738.3	814.9	4,553.2	3,788.7	718.3	4,507.0	1.35%	-11.73%	-1.01%
tai100-5	5,245.4	2,338.9	7,584.3	5,256.2	2,274.7	7,530.9	0.21%	-2.75%	-0.70%
tai100-10	5,636.8	2,156.4	7,793.2	5,674.2	2,070.6	7,744.8	0.66%	-3.98%	-0.62%
tai100-20	6,377.0	1,933.3	8,310.3	6,442.5	1,811.2	8,253.7	1.03%	-6.34%	-0.68%
tai200-10	10,690.0	4,589.9	15,279.9	10,722.6	4,521.8	15,244.4	0.31%	-1.47%	-0.23%
tai200-20	11,414.1	4,333.3	15,747.4	11,487.0	4,232.4	15,719.4	0.64%	-2.33%	-0.18%
tai500-20	26,528.8	11,729.0	38,257.8	26,621.9	11,625.4	38,247.3	0.35%	-0.88%	-0.03%
Averages	6,695.4	2,571.9	9,267.2	6,734.3	2,488.2	9,222.5	0.83%	-8.57%	-1.14%

Table 1. Averages of BEST10 solutions for each set of instances.

Obviously, this average gap will increase in scenarios where the magnitude of the failure-risk costs gets closer to the magnitude of the makespan costs.

As Figure 5 illustrates, positive gaps in makespan costs (where the MKSP1 approach outperforms the MKSP2 approach) are compensated by negative gaps in failure-risk costs (notice that the percentages are applied to different magnitudes, so the total gap is always negative but not as large as it appears to be in the graph).

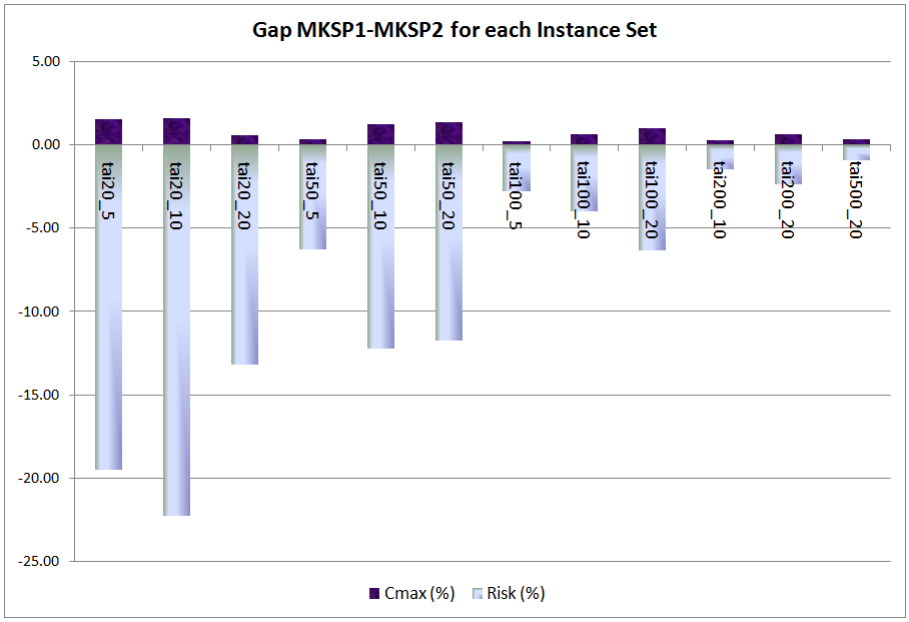

Fig. 5. Makespan and Risk gaps between MKSP1-MKSP2 for each set.

6. A comparison with other extended approaches

In order to compare our BRILS algorithm with other state-of-the-art ILS-based approaches, two well-known algorithms were also extended so they could consider the computation of the failure-risk costs during

the solving process (being ILS-based algorithms too, the extension process was similar to the one applied to develop our own algorithm):

C1. The ILS98-T04, which is the algorithm proposed in Stützle (1998) using $T = 0.4$. According to the author, this parameter value is the one offering the best performance.

C2. The IG-D4T04 and IG-D2T03, which represent two different parameterizations ($D = 4$, $T = 0.4$, and $D = 2$, $T = 0.3$, respectively) of the IG algorithm proposed in Ruiz & Stützle (2007), a highly cited reference in the PFSP literature and, to the best of our knowledge, one of the most efficient algorithms in this field. The first set of parameters was obtained by the algorithm's authors after completing a fine-tuning process. We selected the second set of parameters in order to show how IG's performance can be affected if parameter values other than the recommended ones are applied.

As Figure 6 shows, for the AVG10 metric the differences between our extended BRILS algorithm and each of the other extended ILS-based algorithms are far from being statistically significant. Nevertheless, the boxplot shows that our approach is only matched by the 'parameter-optimized' version of the IG algorithm (IG-D4T04), while both the ILS98T04 algorithm and the non-optimized version of IG seem to perform worse.

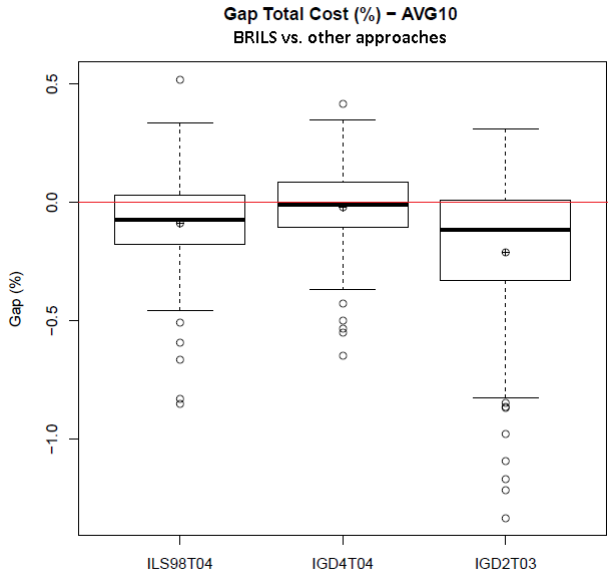

Fig. 6. Gap between our approach and others for AVG10.

A similar analysis can be made from Figure 7, which refers to the BEST10 metric. Here, the IG-D4T04 algorithm seems to slightly improve our results, while the other algorithms seem to perform worse. Again, the IG-D2T03 seems to provide the poorest results, showing a strong dependence of the IG algorithm on the right parameter choice. In all cases, however, differences are far from being statistically significant. These results, in summary, allow to increase our confidence in the quality of the results provided by the BRILS approach.

7. Conclusions

This article has analyzed the PFSP with a realistic non-smooth objective function, proposing both an original mathematical formulation of the problem and a solving probabilistic algorithm, BRILS, which combines Biased Randomization techniques with an Iterated Local Search framework.

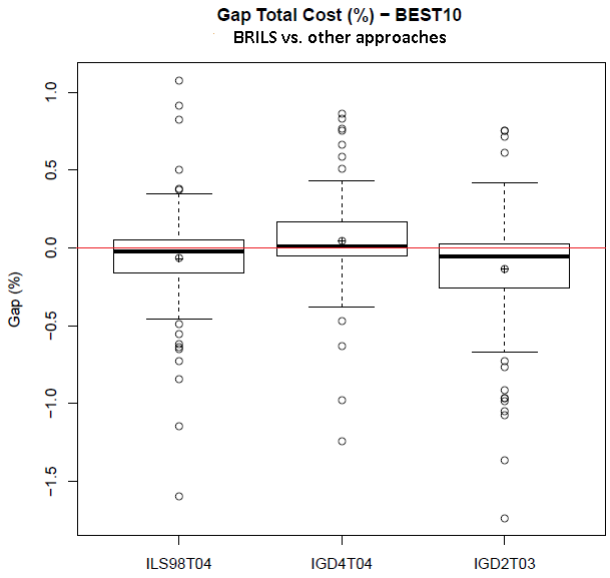

Fig. 7. Gap between our approach and others for BEST10.

The non-smooth objective function takes into account operation costs related to the makespan as well as failure-risk penalty costs, which might be important in real-life applications. These penalty costs introduce additional challenges to this scheduling problem, since gradient-based methods cannot be efficiently employed. Our BRILS algorithm extends a previous work on the PFSP with a (linear) makespan objective function, so that failure-risk costs can also be considered during the exploration of the solution space. The method allows the generation and assessment of a large number of local optimal solutions in a short computational time. A series of numerical experiments were performed to quantify the gains of considering failure-risk costs during the solving process instead of at the end of it. The results show that even for moderate levels of failure-risk costs, the savings that can be gained by using our approach are noticeable (average $gap = -1.14\%$). Moreover, the performance of our algorithm has been checked against other state-of-the-art metaheuristics, which have been previously adapted for the non-smooth version of the PFSP. The results of this comparison show that our algorithm matches or outperforms other competitive approaches, which require from additional fine-tuning processes.

The main limitations of this work are described next. First, we have studied the application of the BRILS approach to the PFSP with non-smooth objective functions; however, a considerable amount of additional research is needed to explore potential applications of BRILS to other COPs with non-smooth objective functions. Second, in our computational experiments we have assumed that both the makespan and the failure-risk costs can be measured in the same units; although this is a reasonable assumption in some real-life scenarios, there might be other cases in which these two costs cannot be measured in the same units, thus requiring a pure multi-objective approach.

Regarding future research lines, the most straightforward one is the application of a similar approach for solving of other combinatorial optimization problems with non-smooth objective functions, e.g. the Job-Shop Scheduling Problem with failure-risk penalties. Other relevant extension of the present work is to consider stochastic processing times in the actual non-smooth PFSP and solve this stochastic version throughout a simheuristic approach as the ones proposed in Juan et al. (2015) and Grasas et al. (2014). Finally, as discussed above, it can also be interesting to explore multi-objective approaches in scenarios where the makespan and the failure-risk costs cannot be measured in the same units.

Acknowledgments

This research has been partially supported by the Spanish Ministry of Economy and Competitiveness, projects MTM2011-29064-C03-02, MTM2014-59179-C2-01 & TRA2013-48180-C3-P, and FEDER. NICTA is funded by the Australian Government through the Department of Communications and the Australian Research Council through the ICT Centre of Excellence Program.

References

Al-Sultan, K. (1995). A tabu search approach to the clustering problem. *Pattern Recognition*, *28(9)*, 1443–1451.

Aldowaisan, T., & Allahvedi, A. (2003). New heuristics for no-wait flowshops to minimize makespan. *Computers and Operations Research*, *30(8)*, 1219–1231.

Bagirov, A., & Yearwood, J. (2006). A new nonsmooth optimization algorithm for minimum sum-of-squares clustering problems. *European Journal of Operational Research*, *170*, 578–596.

Burke, E., Curtois, T., Hyde, M., Kendall, G., G. Ochoa, G., & Petrovic, S. (2010). Iterated local search vs hyper-heuristics: towards general-purpose search algorithms. In *Proceedings of the IEEE Congress on Evolutionary Computation (CEC)* (pp. 1–8).

Chaves, A., & Lorena, L. (2010). A new nonsmooth optimization algorithm for minimum sum-of-squares clustering problems. *Clustering search algorithm for the capacitated centered clustering problem*, *37(3)*, 552–558.

Chen, C., Vempati, V., & Aljaber, N. (1995). An application of genetic algorithms for flow shop problems. *European Journal of Operational Research*, *80(2)*, 389–396.

Chiang, M. (2009). Nonconvex optimization for communication. *Networks Advances in Mechanics and Mathematics*, *17*, 137–196.

Collet, P., & Rennard, J. (2006). Stochastic optimization algorithms. *In Handbook of Research on Nature Inspired Computing for Economics and Management, Rennard, J.P. (ed.) Idea Group Reference, Hershey-PA, USA*, .

Companys, R., & Mateo, M. (2007). Different behaviour of a double branch-and-bound algorithm on fm—prmu—cmax and fm—block—cmax problems. *Computers and Operations Research*, *34*, 938–953.

Cordeau, J., Gendreau, M., Laporte, G., Potvin, J., & Semet, F. (2002). A guide to vehicle routing heuristics. *Journal of the Operational Research Society*, *53*, 512–522.

Davis, L. (1991). *Handbook of genetic algorithms*. Van Nostrand Reinhold, New York.

De Jong, K. (2006). Evolutionary computation: a unified approach. *The MIT Press. Cambridge, MA.*, .

Dorigo, M., & Stützle, T. (2010). Iterated local search: framework and applications. In M. Gendreau, & J. Potvin (Eds.), *Handbook of Metaheuristics (2nd. Edition)* (pp. 227–264). Springer US volume 146 of *International Series in Operations Research & Management Science*.

Drezner, Z., & Hamacher, H. (Eds.) (2002). *Facility location: applications and theory*. New York: New York, Springer.

Feo, T., & Resende, M. (1995). Greedy randomized adaptive search procedures. *Journal of Global Optimization*, *6*, 109–133.

Festa, P., & Resende, M. (2009a). An annotated bibliography of grasp- part i: Algorithms. *International Transactions in Operational Research*, *16*, 1–24.

Festa, P., & Resende, M. (2009b). An annotated bibliography of grasp- part ii: Algorithms. *International Transactions in Operational Research*, *16*, 131–172.

Fisher, M., & Jaikumar, R. (1981). A generalized assignment heuristic for vehicle routing. *International Transactions in Operational Research*, *11*, 109–124.

Fleurent, C., & Glover, F. (1999). Improved constructive multistart strategies for the quadratic assignment problem using adaptive memory. *INFORMS Journal on Computing*, *11(2)*, 198–204.

Freeman, W. (Ed.) (1979). *Computers and Intractability: A Guide to the Theory of NP-Completeness*. New York.

Funke, B., Grunert, T., & Irnich, S. (2005). Local search for vehicle routing and scheduling problems: review and conceptual integration. *Journal of Heuristics*, *11(4)*, 267–306.

Gendreau, M., Hertz, A., & Laporte, G. (1994). A tabu search heuristic for the vehicle routing problem. *Management Science*, *40*, 1276–1290.

Grasas, A., Juan, A., & Lourenço, H. (2014). Simils: a simulation-based extension of the iterated local search metaheuristic for stochastic combinatorial optimization. *Journal of Simulation*, . doi:10.1057/jos.2014.25.

Hamdan, M., & El-Hawary, M. (2002). Hopfield-genetic approach for solving the routing problem in computer networks. In *Proceedings of the IEEE Canadian Conference on Electrical and Computer Engineering* (pp. 823–827).

Hansen, P., Mladenovic, N., Brimberg, J., & Moreno-Pérez, J. (2010). Variable neighborhood search. *In Handbook of Metaheuristics, 2nd. Edition. M. Gendreau, and Potvin, J.Y. (eds.), Kluwer Academic Publishers, International Series in Operations Research & Management Science*, *146*, 61–86.

Hemamalini, S., & Simon, S. (2010). Artificial bee colony algorithm for economic load dispatch problem with non-smooth cost functions. *Electric Power Components and Systems*, *38(7)*, 786–803.

Juan, A., Faulin, J., Ferrer, A., Lourenço, H., & Barrios, B. (2010). Mirha: multi-start biased randomization of heuristics with adaptive local search for solving non-smooth routing problems. *TOP*, *21(1)*, 109–132.

Juan, A., Faulin, J., Grasman, S., Rabe, M., & Figueira, G. (2015). A review of simheuristics: extending metaheuristics to deal with stochastic optimization problems. *Operations Research Perspectives*, *2*, 62–72.

Juan, A., Lourenço, H., Mateo, M., Luo, R., & Castella, Q. (2014). Using iterated local search for solving the flow-shop problem: parallelization, parametrization, and randomization issues. *International Transactions in Operational Research*, *21(1)*, 103–126.

Kennedy, J., & Eberhart, R. (1995). Particle swarm optimization. *Proceedings of the 1995 IEEE International Conference on Neural Networks*, *1–6*, 1942–1948.

Ladhari, T., & Haouari, M. (2005). A computational study of the permutation flow shop problem based on a tight lower bound. *Computers and Operations Research*, *32*, 1831–1847.

Laporte, G. (2009). Fifty years of vehicle routing. *Transportation Science*, *43(4)*, 408–416.

Li, J., & Pan, Q. (2015). Solving the large-scale hybrid flow shop scheduling problem with limited buffers by a hybrid artificial bee colony algorithm. *Information Sciences*, *316*, 487–502.

Li, J., Pan, Q., & Mao, K. (2015). A discrete teaching-learning-based optimisation algorithm for realistic flow-shop rescheduling problems. *Engineering Applications of Artificial Intelligence*, *37*, 279–292.

Loiola, E., de Abreu, N., Boaventura-Netto, P., Hahn, P., & Querido, T. (2007). A survey for the quadratic assignment problem. *European Journal of Operational Research*, *176(2)*, 657–690.

Lokketangen, A., & Glover, F. (1998). Solving zero-one mixed integer programming problems using tabu search. *European Journal of Operational Research*, *106(2–3)*, 624–658.

Lourenço, H., Martin, O., & Stützle, T. (2003). Iterated local search. In F. Glover, & G. Kochenberger (Eds.), *Handbook of Metaheuristics* (pp. 320–353). Springer US volume 57 of *International Series in Operations Research & Management Science*.

Lourenço, H., Martin, O., & Stützle, T. (2010). Iterated local search: framework and applications. In M. Gendreau, & J. Potvin (Eds.), *Handbook of Metaheuristics (2nd. Edition)* (pp. 320–353). Springer US volume 146 of *International Series in Operations Research & Management Science*.

Mester, D., & Bräysy, O. (2007). Active-guided evolution strategies for the large-scale capacitated vehicle routing problems. *Computers and Operations Research*, *34*, 2964–2975.

Moccellin, J. (1995). A new heuristic method for the permutation flow-shop scheduling problem. *Journal of the Operational Research Society*, *46*, 883–886.

Muter, I., Birbil, S., & Sahin, G. (2010). Combination of metaheuristic and exact algorithms for solving set covering-type optimization problems. *Informs Journal on Computing*, *22(4)*, 603–619.

Nawaz, M., Enscore Jr, E., & Ham, I. (1983). A heuristic algorithm for the *m*-machine, *n*-job flowshop sequencing problem. *OMEGA*, *11(1)*, 91–95.

Nikolaev, A., & Jacobson, S. (2010). Simulated annealing. In M. Gendreau, & J. Potvin (Eds.), *Handbook of Metaheuristics (2nd. Edition)* (pp. 1–40). Springer US volume 146 of *International Series in Operations Research & Management Science*.

Oncan, T. (2007). A survey of the generalized assignment problem and its applications. *INFOR*, *45(3)*, 123–141.

Oonsivilai, A., Srisuruk, W., Marungsri, B., & Kulworawanichpong, T. (2009). Tabu search approach to solve routing issues in communication networks 1. *World Academy of Science, Engineering and Technology*, (pp. 1174–1177).

Osman, L., & Potts, C. (1989). Simulated annealing for permutation flowshop scheduling. *OMEGA*, *17(6)*, 551–557.

Papadimitriou, C., & Steiglitz, K. (1982). *Combinatorial optimization: algorithms and complexity*. Printice-Hall, New Jersey.

Pinedo, M. L. (2008). *Scheduling: theory, algorithms, and systems*. Springer Science & Business Media.

Rajendran, C., & Ziegler, H. (2004). Antcolony algorithms for permutation flowshop scheduling to minimize makespan/total flowtime of jobs. *European Journal of Operational Research*, *155(2)*, 426–438.

Reeves, C. (1993). Improving the efficiency of tabu search for machine scheduling problems. *Journal of the Operational Research Society*, *44(4)*, 375–382.

Reeves, C. (1995). A genetic algorithm for flowshop sequencing. *Computers and Operations Research*, *22(1)*, 5–13.

Reeves, C. (2010). Genetic algorithms. In M. Gendreau, & J. Potvin (Eds.), *Handbook of Metaheuristics (2nd. Edition)* (pp. 109–140). Springer US volume 146 of *International Series in Operations Research & Management Science*.

Resende, M., & Ribeiro, C. (2010). Greedy randomized adaptive search procedures: advances, hybridizations and applications. In M. Gendreau, & J. Potvin (Eds.), *Handbook of Metaheuristics (2nd. Edition)* (pp. 283–319). Springer US volume 146 of *International Series in Operations Research & Management Science*.

Ribas, I., Companys, R., & Tort-Martorell, X. (2015). An efficient discrete artificial bee colony algorithm for the blocking flowshop problem with total flowtime minimization. *Expert Systems with Applications*, *42 (15–16)*, 6155–6167.

Rinnooy Kan, A. (1973). The machine scheduling problem. *Stichting Mathematisch Centrum. Mathematische Besliskunde, Report No. BW 27/73*, (pp. 1–153).

Ruiz, R., & Stützle, T. (2007). A simple and effective iterated greedy algorithm for the permutation flow-shop scheduling problem. *European Journal of Operational Research*, *177(3)*, 2033–2049.

Ruiz, R., & Stutzle, T. (2008). An iterated greedy heuristic for the sequence dependent setup times flow-shop problem with makespan and weighted tardiness objectives. *European Journal of Operational Research*, *187(3)*, 1143–1159.

Schlüter, M., Egea, J., & Banga, J. (2009). Extended ant colony optimization for non-convex mixed integer nonlinear programming. *Computers & Operations Research*, *36(7)*, 2217–2229.

Selim, S., & Al-Sultan, K. (1991). A simulated annealing algorithm for the clustering problem. *Pattern Recognition*, *24(10)*, 1003–1008.

Stützle, T. (1998). Applying iterated local search to the permutation flow-shop problem. *FG Intellektik, TU Darmstadt, Darmstadt, Germany. Available at:* HTTP://IRIDIA.ULB.AC.BE/ STUETZLE/PUBLICATIONS/AIDA-98-04.PDF *(accessed June 10, 2013)*, .

Suman, B., & Kumar, P. (2006). A survey of simulated annealing as a tool for single and multiobjective optimization. *Journal of the Operational Research Society*, *57(10)*, 1143–1160.

Sun, L., Cui, K., Chen, J., Wang, J., & He, X. (2013). Some results of the worst-case analysis for flow-shop scheduling with a learning effect. *Engineering Annals of Operations Research*, *211(1)*, 481–490.

Taillard, E. (1993). Benchmarks for basic scheduling problems. *European Journal of Operations Research*, *64(2)*, 278–285.

Vaisakh, K., & Srinivas, L. (2011). Genetic evolving ant direction hde for opf with non-smooth cost functions and statistical analysis. *Expert Systems with Applications*, *38(3)*, 2046–2062.

Widmer, M., & Hertz, A. (1989). A new heuristic method for the flow shop sequencing problem. *European Journal of Operational Research*, *41(2)*, 186–193.

Yang, I., & Chou, J. (2011). Multiobjective optimization for manpower assignment in consulting engineering firms. *Applied Soft Computing*, *11(1)*, 1183–1190.