Applied Soft Computing 23 (2014) 298–307

Contents lists available at ScienceDirect

Applied Soft Computing

journal homepage: www.elsevier.com/locate/asoc

Hybrid algorithms for the twin–screw extrusion configuration problem

Cristina Teixeira[a], José Covas[a], Thomas Stützle[b], António Gaspar-Cunha[a,*]

[a] IPC/I3N – Institute for Polymer and Composites, University of Minho, Guimarães, Portugal
[b] IRIDIA, Université Libre de Bruxelles (ULB), CP 194/6, Av. F. Roosevelt 50, B-1050 Brussels, Belgium

ARTICLE INFO

Article history:
Received 20 June 2013
Received in revised form 25 March 2014
Accepted 18 June 2014
Available online 26 June 2014

Keywords:
Twin-screw extruder
Multi-objective optimization
Hybrid algorithms
Ant colony optimization
Evolutionary algorithms
Pareto local search

ABSTRACT

The twin-screw configuration problem (TSCP) arises in the context of polymer processing, where twin-screw extruders are used to prepare polymer blends, compounds or composites. The goal of the TSCP is to define the configuration of a screw from a given set of screw elements. The TSCP can be seen as a sequencing problem as the order of the screw elements on the screw axis has to be defined. It is also inherently a multi-objective problem since processing has to optimize various conflicting parameters related to the degree of mixing, shear rate, or mechanical energy input among others. In this article, we develop hybrid algorithms to tackle the bi-objective TSCP. The hybrid algorithms combine different local search procedures, including Pareto local search and two phase local search algorithms, with two different population-based algorithms, namely a multi-objective evolutionary algorithm and a multi-objective ant colony optimization algorithm. The experimental evaluation of these approaches shows that the best hybrid designs, combining Pareto local search with a multi-objective ant colony optimization approach, outperform the best algorithms that have been previously proposed for the TSCP.

Introduction

Twin-screw extruders are widely used by the plastic industry to prepare polymer blends, compounds and composites for increasingly more advanced applications. They contain two parallel intermeshing screws rotating inside a heated barrel. If both screws rotate in the same direction, these machines are known as co-rotating twin screw extruders. They often have a modular construction, i.e., a number of individual barrel and screw elements are available, enabling the assembly of geometrical configurations that are adapted to the characteristics of the material being processed [1].

It is well known that the performance of co-rotating twin-screw extruders is strongly dependent on the operating conditions, on the screw geometry/configuration used and on the properties of the polymeric system. Screws are built by connecting along a shaft modular screw elements with different geometric characteristics and, consequently, different impact on the compounding process. Thus, when a new compounding process is to be implemented, the best screw configuration needs to be defined. This problem, known as twin-screw configuration problem (TSCP), consists in defining the location of a pre-defined number of screw elements along the screw axis in order to optimize the process. The latter is evaluated according to different performance measures, which from an optimization perspective correspond to the objectives to be optimized. As a result, the TSCP is a multi-objective optimization problem, that is, it is characterized by the simultaneous optimization of several, conflicting objectives.

In recent years, an automatic process that links the optimization procedure to a modeling routine has been proposed and continuously improved. Initially, Gaspar et al. proposed a multi-objective evolutionary algorithm (MOEA) named reduced Pareto set genetic algorithm (RPSGA) [2] that was applied to solve the TSCP [3] in terms of the optimization of the processing conditions [4]. Teixeira et al. applied the same optimizer to the optimization of several screw configuration case studies dealing with reactive starch cationization [5]. This automatic procedure is only effective if the modeling routine is able to predict correctly the evolution of the different process performance measures (that is, the objectives to be optimized) along the extruder barrel (e.g., pressure, temperature, shear rate, residence time, viscous dissipation, average strain and dispersive and distributive mixing). This was achieved by a modeling routine [6], that takes into account the relevant physical phenomena and that is sensitive to changes in polymer properties, operating conditions and screw geometry/configuration. However,

* Corresponding author. Tel.: +351 919221333.
E-mail addresses: cteixeira@dep.uminho.pt (C. Teixeira), jcovas@dep.uminho.pt (J. Covas), stuetzle@ulb.ac.be (T. Stützle), agc@dep.uminho.pt (A. Gaspar-Cunha).

http://dx.doi.org/10.1016/j.asoc.2014.06.022
1568-4946/© 2014 Elsevier B.V. All rights reserved.

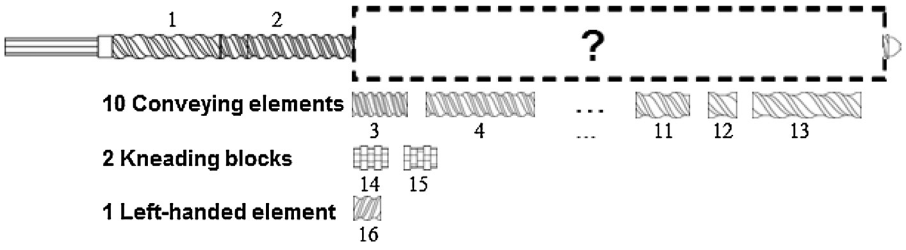

Fig. 1. Schematics of the twin-screw extrusion configuration problem.

since this requires 1–3 min to evaluate one screw configuration (measured on a single core of ing an AMD opteron TM 2116 dual-core processors running at 2.4 GHz with 2 MB L2 cache), it is important to develop efficient optimization algorithms in order to define adequate screw configurations within a reasonable computational time.

Along these lines, several tests were performed in order to obtain an effective single objective iterative improvement algorithm. This algorithm was extended to several bi-objective case studies of the TSCP, by embedding it into the two phase local search (TPLS) [7] and Pareto local search (PLS) frameworks [8]. TPLS is based on a search model using a scalarized acceptance criterion (SAC), whereas PLS is based on a search model using a component-wise acceptance criterion (CWAC) [9]. In both cases, good results were obtained when compared with RPSGA. In another line of research, multi-objective ant colony optimization (MOACO) algorithms were adapted to tackle the TSCP, resulting in a final MOACO algorithm that was often superior to the other algorithms had been considered previously [10].

All the algorithms developed so far for the TSCP rely on a single type of search method. Over the last years, the combination of various search methods into hybrid algorithms has received strong attention [11,12]. These combine components of different algorithmic ideas trying to obtain synergetic effects resulting in increased convergence speed and/or higher quality solutions. The probably most common approach to generate hybrid methods is to complement population-based metaheuristics with local search procedures [13–16] to increase convergence speed and to locally fine-tune solutions.

In this paper, we examine whether the heuristic algorithms we have developed previously can be improved by hybridization. We examine various possibilities for hybridization. A first possibility is to post-process solutions generated by RPSGA, TPLS and MOACO by PLS. This hybridization is motivated by the high performance combinations of TPLS and PLS reached in other studies [9,17]. A second possibility is to seed RPSGA and MOACO by good initial solutions obtained from TPLS, trying to speed-up the convergence of these population-based methods. A common feature of all the hybrid algorithms we study is that they consist of two distinct phases. In a first phase, a starting set S of non-dominated solutions is generated, which, in a second phase is then improved by another search method. Our experimental study of these hybrid methods on various bi-objective case studies of the TSCP shows the usefulness of hybridization: for the same number of maximum screw evaluations, the hybrid methods find better approximations to the Pareto front. Our experiments also establish a combination of MOACO with PLS as the best performing hybrid strategy.

This paper is structured as follows. Section “The twin-screw configuration problem” presents the TSCP. The metaheuristics studied in this work are explained in Section “Multi-objective algorithms”. In Section “Case studies” we identify the case studies and the results are discussed in Section “Results and discussion”. Finally, the main conclusions are presented in Section “Conclusions”.

The twin-screw configuration problem

The TSCP consists in the definition of the sequence of a fixed number of screw elements along the screw axis, in order to maximize the relevant performance measures of a given compounding process. The set of available screw elements is composed by the following three types of geometries:

1 *Conveying elements* have a positive drag conveying capability; they are characterized by different pitches and lengths. A screw element with higher pitch has a higher conveying capability.
2 *Left handed elements* create a backflow and thus impose a restriction to the axial progress of the polymer flow; they are generally used to induce melting and dispersive mixing, they are also characterized by pitch and length.
3 *Kneading blocks* are sets of kneading disks that are assembled to form positive, neutral or negative staggering angles. Positive angles create conveying capability, while negative angles impose a restriction to the polymer flow. The former promotes distributive mixing, while the latter are used for dispersive mixing purposes.

In Fig. 1, we give an example where a total of 14 screw elements must be located along the screw axis. They encompass 11 conveying elements, two kneading blocks and one left handed element.

As the individual modules have different geometric characteristics and, consequently, dictate different polymer flow characteristics, the process performance will be different depending on their relative positions [18Teixeira et al., 2007,6]. There is no closed analytical form to compute the performance measure of the extrusion process for a given screw configuration; thus, this computation relies on simulation, which essentially requires numerical solutions to the balance equations describing the process. Earlier we developed an appropriate modeling routine that is able to predict satisfactorily the evolution of the different process performance parameters along the extruder [6].

Next, we give a high-level process description. Table 1 represents a possible screw configuration composed by 14 conveying elements with different lengths (30, 60 and 120 mm) and pitches (20, 30, 45 and 60 mm), one left handed element (with a pitch of −20 mm) and two kneading blocks forming staggering angles of −45° and −60°. Fig. 2a depicts a co-rotating twin screw extruder with the screw configuration presented in Table 1. This machine comprises a screw feeder to feed the polymer at a pre-set rate, the screw modular barrel elements and a die to shape the final melt. Both barrel and die are heated to a set temperature by heater bands. Fig. 2a also shows the axial evolution of the computational pressure and temperature. The physical steps I–VI the material is typically subjected to as it progresses along the screw axis are identified in Fig. 2b [18Teixeira et al., 2007,6]. Initially the polymer will flow through conveying elements under starved conditions as a result of screw rotation and, consequently, without pressure (Fig. 2b, I). When the first restrictive screw element (that is, either a left handed element or a kneading block) is reached, the channel

Table 1
Screw profile used in the simulation presented in Fig. 2. Restrictive elements are indicated in boldface.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Length (mm)	97.5	120	30	**45**	**30**	12	30	30	120	60	**30**	60	60	37.5	30	60
Pitch angle (°)	45	30	30	**KB-45**	**KB-60**	60	30	60	30	20	**−20**	30	45	20	20	30

fills up (Fig. 2b, II) and heat transfer becomes more efficient. Eventually, a thin melt film surrounds the solid plug (Fig. 2b, III). Soon, the melt film near the active flanck of the screw will grow into a melt pool (Fig. 2b, IV). Eventually the solid bed will break, so that the surviving pellets become dispersed in polymer melt (Fig. 2b, V) and melting is quickly completed. Depending on the screw configuration, the melt can progress in a total or a partially filled channel, and consequently under or without pressure, respectively. Finally, the polymer is pushed through the die, and the extrudate is produced.

Case studies

In order to compare the performance of the hybrid and non-hybrid algorithms, we defined experiments that make use of the characteristics of a co-rotating twin screw extruder Leistritz LSM30.34 available at the University of Minho. The geometric characteristics (length and pitch) of the 16 individual screw elements available are described in Table 1. Taking into account the importance of restrictive elements in the process, four different numbers of screw elements where considered (Table 2): from one in instance TSCP1 to four in instance TSCP4. Note that KB-60 represents a set of kneading disks forming a backwards angle of 60°.

As illustrated in Fig. 1, to guarantee enough conveying capability upstream, the position and geometry of the first two conveying elements are fixed. Thus, the aim is to define the location of 14 screw elements that optimize the performance of the compounding process. Table 3 presents the objectives considered, the aims of the optimization (minimization or maximization) and the respective range of variation ($[x_{max}, x_{min}]$). Each of the combinations of two objectives defines one case study for the algorithms. Thus,

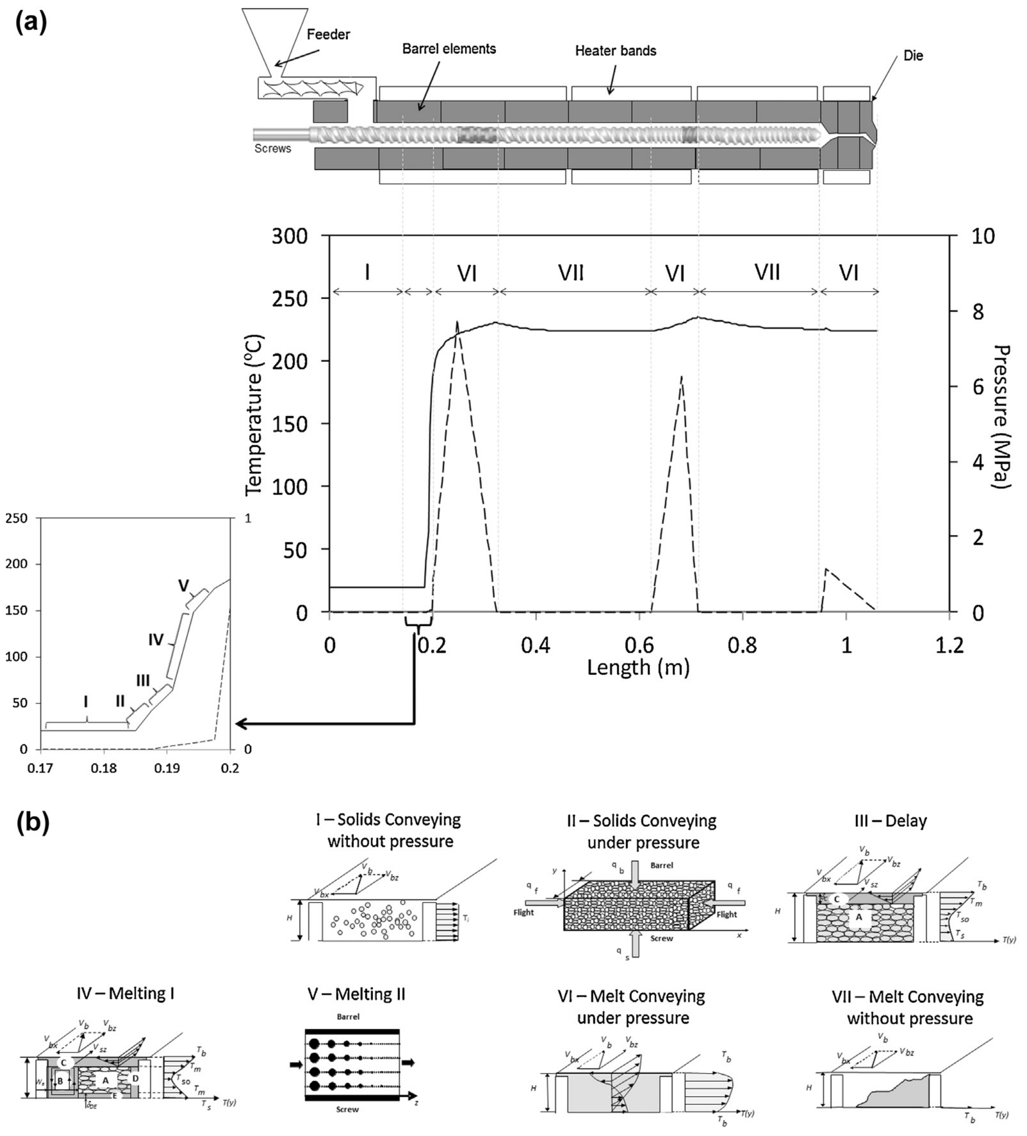

Fig. 2. Evolution of computational pressure and temperature along the extruder and compounding phenomenological steps.

Table 2
Screw elements used in each instance of the case studies. Restrictive elements are indicated in bold.

Instance	Screw	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
TSCP1	Length (mm)	97.5	120	45	60	30	30	30	60	**30**	120	30	120	37.5	60	60	30
	Pitch (mm)/angle (°)	45	30	45	30	20	60	30	20	**KB-60**	30	30	60	20	45	30	20
TSCP2	Length (mm)	97.5	120	45	60	**30**	30	30	60	**30**	120	30	120	37.5	60	60	30
	Pitch (mm)/angle (°)	45	30	45	30	**-20**	60	30	20	**KB-60**	30	30	60	20	45	30	20
TSCP3	Length (mm)	97.5	120	**45**	60	**30**	30	30	60	**30**	120	30	120	37.5	60	60	30
	Pitch angle (°)	45	30	**KB-45**	30	**-20**	60	30	20	**KB-60**	30	30	60	20	45	30	20
TSCP4	Length (mm)	97.5	120	**45**	60	**30**	30	30	60	**30**	120	30	120	**37.5**	60	60	30
	Pitch angle (°)	45	30	**KB-45**	30	**-20**	60	30	20	**KB-60**	30	30	60	**KB-30**	45	30	20

we have a total of 12 TSCP instances to apply the algorithms and each of these instances corresponds to the combination of a specific number of screw elements and two objectives.

As mentioned in Introduction Section, the modeling routine requires considerable computation time, in the range of one or more minutes on a modern CPU. Thus, even for small problems such as the ones with 14 screw elements that we consider in this paper, an exhaustive enumeration of the search space is not possible. In fact, assuming one minute per evaluation, an exhaustive enumeration would result in computation times of more than 150 000 CPU years. Hence, heuristic algorithms are the main feasible way to tackle the TSCP. To limit the computation time of the heuristics, we use as a stop criterion for each algorithm 3000 evaluations of the modeling routine. Thus, the comparison of the algorithms is based on the same computational effort allocated to each algorithm.

Multi-objective algorithms

In this section, we describe the local search and the population-based heuristic algorithms that we explored in previous research efforts and that we used as our basis for the hybrid algorithms we study in this paper.

Two-phase local search

The main idea underlying TPLS is to aggregate the objectives of the multi-objective problem into a sequence of single objective problems that are solved with effective local search methods [19,9]. The single objective problems are typically generated by using weighted sum aggregations and the sequence of single objective problems is obtained through modifications of the weights during the search process, which allows TPLS to search different areas of the Pareto front. Details about the construction of the weights adopted for this work can be found in [7]. The TPLS algorithm we apply is presented in Algorithm 1 in pseudo-code. It starts with an initial solution *s*, which is generated randomly (line 1). Then, *s* is improved by a stochastic local search [20] algorithm SLS_1 (line 2) to find a solution that is as good as possible for the set of weights considered. The final solution *s* is added to the archive (line 3). Next, TPLS explores a sequence of aggregations applying an SLS algorithm SLS_2, considering as initial solutions for each aggregation the best solution of the previous aggregation. All non-dominated solutions found in this process are added to the archive (lines 4–7). The algorithm stops when the maximum number of evaluations is reached and returns the filtered archive (lines 8–9) as a final approximation to the Pareto front.

Table 3
Optimization objectives, direction of optimization and prescribed range of variation.

Objectives	Direction	X_{min}	X_{max}
Specific mechanical energy	Minimization	0.1	2
Viscous dissipation	Minimization	0.9	1.5
Average strain	Maximization	1000	15 000

The TPLS algorithm we apply here is an iterative first-improvement algorithm that makes use of a two-exchange neighborhood that exchanges the position of two screw elements in the sequence and that uses the neighborhood restriction techniques as described in [7]. When running TPLS, both the final best solution found by the iterative improvement process and all non-dominated solutions that have been examined during the local search process are added to the archive.

Algorithm 1. TPLS algorithm

```
1:   s is a randomly generated solution
2:   s′ = SLS_1(s)    /* First phase */
3:   Adds′ to Archive
4:   for all weight vectors λ do
5:     s = s′
6:     s′ = SLS_2(s, λ)    /* Second phase */
7:     Add s′ to Archive
8:   end for
9:   Filter Archive
10:  return Archive
```

Pareto local search

PLS uses dominance criteria for accepting neighboring candidate solutions into an archive that collects the current set of non-dominated solutions found so far [21,9]. PLS can be seen as a straightforward extension of iterative improvement algorithms from single objective problems to the multi-objective case. An outline of PLS is presented in Algorithm 2. The archive is initialized with a set of solutions *S* and for each solution *s* in the Archive an associated visited flag is set to *false* (line 1). At each step of PLS, a solution is first chosen uniformly at random among all still unvisited solutions in the archive (that is, among all solutions whose visited flag is *false*; line 3). Next, the neighborhood of the chosen solution *s* is examined and all neighboring solutions that are non-dominated with respect to *s* (line 4) are added to the archive with their visited flag set to *false* (line 5). At the end of the neighborhood examination, the visited flag of *s* is set to *true* (line 6). Before the next iteration, the archive is filtered by removing all dominated solutions (line 7). This process continues until either all solutions in the archive are visited or the limit of the number of screw evaluations is reached. When PLS is used as a stand-alone algorithm, it is typically initialized by one single solution; however, when hybridized with other algorithms, the initial set of PLS is the candidate set of solutions output by the other algorithm.

Given the possibility of the exponential increase of the number of non-dominated solutions, a bounding technique has been used [22]. The main idea is to allow only one solution in a specific region of the objective space to be added to the archive. For this purpose, the objective space is divided into a grid according to a geometric sequence and only one solution is allowed to occupy each hypercube. This method limits the increase of the non-dominated solutions in the archive and prevents cycling, given that one solution will only enter into a hypercube if it dominates the solution already there in.

PLS uses the same two-exchange neighborhood as TPLS; more details on the PLS algorithm can be found in [8].

Algorithm 2. PLS Algorithm

```
1: Initialize Archive $A$, mark all $s \in A$ as unvisited
2: while termination condition not met do
3:    $s \leftarrow RandomChoice(A, \text{unvisited})$
4:    $S' \leftarrow$ neighbors $s'$ not dominated w.r.t. $s$, mark all $s'$ as unvisited
5:    $A \leftarrow A \cup S'$
6:    mark $s$ as visited
7:    Filter($A$)
8: end while
9: return $A$
```

Reduced Pareto set genetic algorithm

Evolutionary algorithms are based on the concept of natural evolution and use a set of solutions that, in each generation, will undergo selection, mutation and crossover operations, trying to improve the quality of the following generations of solutions [23]. In this study, we apply the reduced Pareto set genetic algorithm (RPSGA) proposed by Gaspar et al. [3] as a representative MOEA algorithm. RPSGA uses two populations: the main and an elitist population. The first population with n individuals stores the best individuals found in each iteration; the elitist population with $2n$ individuals applies the inver-over operator [24]. Additionally, an archive of non-dominated solutions is kept in order to avoid good solutions getting lost. The main steps of the RPSGA algorithm are illustrated in Algorithm 3.

Algorithm 3. RPSGA

```
1:  Initialize $p_e$ (external population) and Archive to empty set
2:  $p_i$ is a randomly generated, initial population (internal)
3:  while termination condition not satisfied do
4:     Evaluate $p_i$
5:     Evaluate individuals' fitness considering clustering
6:     Copy best individuals to $p_e$
7:     if external population full then
8:        $p_e \leftarrow Clustering(p_e)$
9:        Copy best individuals of $p_e$ to $p_i$
10:    end if
11:    Select individuals for reproduction
12:    Apply Inver-over operator to selected pairs of individuals
13:    Add non-dominated solutions to Archive
14:    Filter Archive
15: end while
16: return Archive
```

First, the algorithm generates an empty external population and archive (line 1) and randomly creates the internal population (line 2). At each iteration, the following steps are carried out. The solutions of the internal population are evaluated by the modeling routine (line 4); then, the fitness of each solution is calculated making using of the clustering technique (line 5) and a fixed number of best solutions are copied to the external population (line 6). If the latter is not full, the selection and inver-over operators are applied to individuals of the internal population to produce a new population (lines 11 and 12). The non-dominated solutions found are copied to the archive (line 13) and this is filtered to remove dominated solutions (line 14). If the external population becomes full, a clustering technique is applied to sort the individuals of the external population, and a pre-defined number of the best individuals are incorporated in the internal population by replacing the lowest fitness individuals (lines 8 and 9). More details on the algorithm can be found in [3].

Multi-objective ant colony optimization (MOACO)

Ant colony optimization is a population-based algorithm that takes inspiration on real ants' foraging behavior [25,26]. The main idea of ACO is to probabilistically construct solutions as a function of the concentration of pheromone on the ants trail and to iteratively reinforce the components of solutions with better performance. MOACO algorithms apply the mechanisms of ACO algorithms to tackle multi-objective problems. Algorithm 4 presents the outline of our MOACO algorithm. It starts by defining the initial pheromone value of each entry of the pheromone matrix and the initial values of pheromone limits. Then the main loop is repeated (lines 2–7) until some termination condition is met. First, the ants probabilistically construct solutions to the TSCP (line 3). Then, each solution is evaluated by the modeling routine (line 4) and all non-dominated solutions found in the current algorithm iteration are added to the archive; the archive is filtered by removing dominated solutions (line 5). Finally, the pheromone matrices will be updated (line 6) first by evaporating a part of the pheromone and then depositing pheromone on the solution components of elite solutions. This pheromone updating process will induce a search around the best solutions found so far.

Algorithm 4. MOACO for TSCP

```
1: Initialize Initialize pheromone values matrices and Archive
2: while termination condition not satisfied do
3:    Construct solutions
4:    Evaluate the solutions' fitness
5:    Add nondominated solutions to Archive and Filter Archive
6:    Update pheromone trails
7: end while
8: Filter Archive
9: return Archive
```

In our previous work, we tested various choices to be taken in a MOACO algorithm such as the number of ant colonies to be used, or specific settings of parameters [10]. Here, we use the best ACO configuration found previously: a set of 60 ants divided into three colonies, one pheromone matrix for each objective and the update of the matrices done by region. Details about this MOACO algorithm for the TSCP can be found in [10].

Hybrid algorithms

Earlier, we studied separately the above four algorithms [3,8,7,10]. Here, we combine them by concatenating two of them. To do so, we follow two main approaches. The first is to improve the solutions generated by TPLS, MOACO, or MOEA by the PLS algorithm. The second is to first execute TPLS with very few scalarizations and then to use the non-dominated solutions that have been found as an initial population for the other algorithms.

Improving solutions with PLS

One natural choice is to apply PLS to post-process the solutions generated by other algorithms and this idea has been applied also in other researches. For example, the post-processing of solutions generated by TPLS is also known as the TPLS + PLS framework [17]. If MOEA or MOACO are used in the first phase, in what follows we refer to the resulting algorithms as MOEA+PLS and MOACO+PLS. When combined with PLS, the algorithms TPLS, MOACO and MOEA are stopped early to compare the solution quality reached by the hybrid algorithms with the pure strategies using the same computational effort. (TPLS is stopped after four scalarizations have been executed as described in more detail in the next subsection, while the MOACO and the MOEA algorithm are stopped after 1500 evaluations, allowing PLS a further 1500 evaluations until the termination criterion is reached.) Note that for TPLS, MOEA and MOACO an archive with all non-dominated solutions found during the search process is saved. Once these methods reach a certain number of evaluations, PLS is started, using as initial set of solutions, all non-dominated solutions existing in the archive. Note that the archive bounding used in PLS is not applied to filter the archive of non-dominated solutions given to it as input. However, when choosing a solution for neighborhood exploration, the rules of the archive bounding are used for selecting the right solution in each hypercube.

Seeding population-based algorithms with TPLS

The quality of the initial population can have a significant influence on the search performance of population-based algorithms. Therefore, a common goal in memetic algorithms [13] is to generate a high quality initial population, which is advantageous in the early iterations of MOEAs and MOACO algorithms. Here, we generate such a set of high-quality initial solutions by TPLS. In particular, TPLS is run with four pre-defined weights $\lambda \in \{0, 1/3, 2/3, 1\}$, which results in weight vectors $(0, 1)$; $(1/3, 2/3)$; $(2/3, 1/3)$; $(1, 0)$. TPLS stops once the best solution is found for each weight. It is straightforward to seed RPSGA with some initial solutions. This is done by including in the initial population all non-dominated solutions generated during the execution of TPLS; if the population cannot be filled with the non-dominated solutions generated by TPLS, it will be completed by solutions that are generated uniformly at random. Seeding a MOACO algorithm can be done by initializing the pheromone trails in a specific way. Our choice in the hybrid TPLS+MOACO algorithm is to update the respective pheromone matrices with an amount of pheromone corresponding to five times the solution's final quality before the actual start of the construction of solutions in the MOACO algorithm. In this way, the first solutions generated by MOACO already take into account information on the solutions found by TPLS. In other words, the TPLS solutions are used to bias the solution construction during the MOACO algorithm.

Finally, we need to mention that there are actually two interpretations for the TPLS+PLS hybrid. The first is that PLS is used to post-process the non-dominated solutions generated by TPLS, while the same hybrid can also be seen as using TPLS to seed the PLS algorithm with high-quality initial solutions instead of using a single random solution as seed for PLS. Anyway, both interpretations lead to the same algorithm.

Results and discussion

We now evaluate the performance of the heuristic algorithms that we consider in this study. As a first step in the evaluation, we use attainment functions [27]. This methodology gives for each objective vector z in the objective space the probability that z is attained in one single run. As it is not possible to compute the exact attainment function for the type of algorithms used here, it is estimated based on several runs of the optimizer. This estimation is called the empirical attainment function (EAF) [28]. To allow the visualization of the EAF differences, we plot the points in the objective space that correspond to the jump points of the EAF differences between the pair of algorithms compared; the level of gray indicates the value of the differences [29]. The estimation of the EAF uses 10 runs of each algorithm with a different random number seed. Each algorithm was run for a maximum of 3000 evaluations of the modeling routine on each of the 12 instances. Given the large number of resulting plots, the paper uses only a few examples, mainly taken from instance TSCP4 with four restrictive screw elements. The EAF plots on all instances can be consulted in the Supplementary results at http://www.dep.uminho.pt/agc/agc/Supplementary_Information_Page.html.

In a second step, we summarize the performance of the heuristic algorithms using performance indicators for multi-objective optimization. In particular, we use the hypervolume indicator, which is the unary performance indicator that complies best with the principle of Pareto optimality [30]. In bi-objective problems and in the minimization case, the hypervolume measures the surface dominated by the non-dominated solutions obtained by an algorithm and limited by a point that in each objective is larger than any solution in the non-dominated front. For maximization problems, an analogous description applies. Here, we first normalize the results of all algorithms in all runs to the interval [1,2]. We do so by first removing dominated points, then converting any maximization objective into a minimization one, and finally considering for each objective and instance the smallest and largest objective values found by any of the algorithms. The smallest value is then mapped to 1.0 and the largest to 2.0. The hypervolume is computed taking point (2.0,2.0) as a reference. Where statistical tests are used, we assume as default a threshold for the error of first type of $\alpha = 0.05$.

Improving solutions by PLS

As a first step in our experimental analysis, we examine the EAF differences between the single, non-hybrid search methods and the hybrids that use PLS for post-processing. Examples of results are given in Fig. 3 for instance TSCP4 using as objectives SME and average strain. In general, MOACO+PLS shows clear advantages over MOACO in; the differences are representative of the advantage of MOACO+PLS over MOACO on all combinations of objectives studied for instance TSCP4. When comparing MOACO+PLS to MOACO on the other instances, it can be observed that the advantage of MOACO+PLS over MOACO becomes stronger as the number of restrictive screw elements increases from one in TSCP1 to four in TSCP4 (see Figs. 1–4 in the Supplementary results).

The conclusions when comparing MOEA+PLS with the original MOEA are similar to those for MOACO; in Fig. 3 strong advantages of MOEA+PLS over MOEA can be observed and these are representative of the other objective combinations on TSCP4. Similar to the MOACO+PLS hybrid algorithm, the relative improvement obtained by combining PLS with MOEA over using MOEA as a standalone algorithm increases with the number of restrictive screw elements (see Figs. 13–16 in the Supplementary results).

The same trend as for MOACO+PLS and MOEA+PLS, although less strong, is observable for the TPLS+PLS hybrid. An example of the EAF differences is shown in the bottom row of Fig. 3; other comparisons can be found in the Supplementary material.

In summary, the examination of the EAF difference plots indicates that using PLS for post-processing the solutions generated by MOACO, MOEA, and TPLS is beneficial and generally improves performance.

Seeding algorithms by TPLS

Next, we consider the hybrid algorithms that use TPLS to generate an initial set of non-dominated solutions. The computational results can be observed in Figs. 6–8 and 17–20 in the Supplementary material. Here, we summarize the main conclusions. When comparing TPLS+MOACO and TPLS+MOEA to MOACO and MOEA, respectively, the conclusions are rather mixed. For the instances with three or four restrictive elements, the hybrid algorithms show some advantages with respect to the EAF differences when using MOACO and MOEA alone, while for one or two restrictive screw elements, the non-hybrid algorithms reach similar or slightly better Pareto front approximations. As mentioned in Section "Hybrid algorithms", TPLS+PLS can be seen as seeding PLS by TPLS or as post-processing TPLS by a PLS algorithm. If we take this second perspective, we have to compare TPLS+PLS to PLS, where PLS is seeded by one randomly generated configuration. When comparing the results of these two algorithms, there seems to be a slight edge of the hybrid algorithm over PLS, although the differences

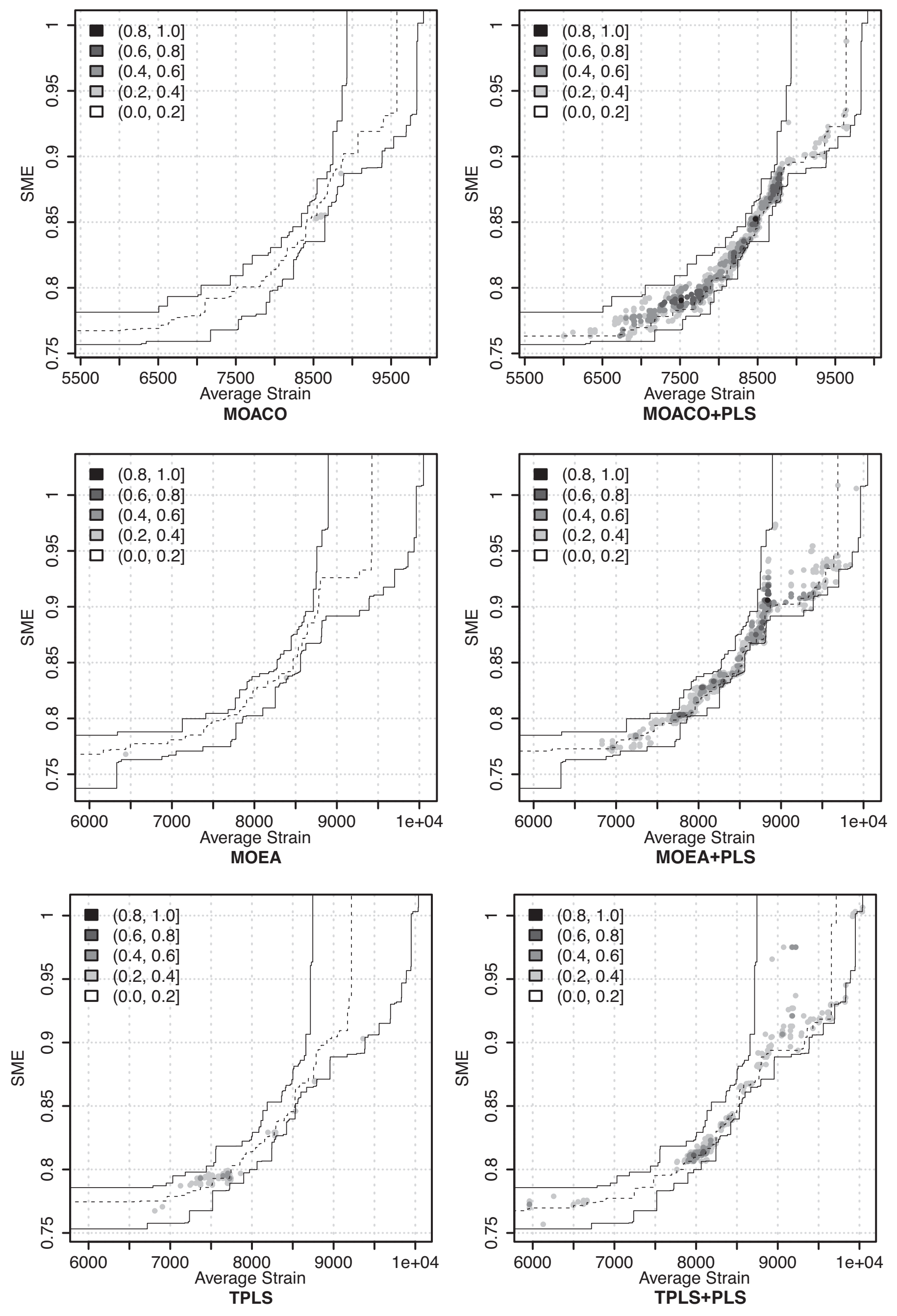

Fig. 3. EAF difference plots for instance TSCP4 and the objectives SME and average strain comparing MOACO to MOACO+PLS (top row), MOEA to MOEA+PLS (middle row) and TPLS to TPLS+PLS (bottom row) after 3000 evaluations. Advantages in favor of the pure algorithm are indicated on the left side, those in favor of the hybrid algorithm on the right side.

between the two are small (see Figs. 29–32 in the Supplementary material).

Other comparisons

In a next step, we compare the performance of the various hybrid algorithms. When comparing MOACO+PLS with MOEA+PLS, the advantages are mainly on the side of the MOACO+PLS algorithm (see Figs. 33–36 of the Supplementary pages). For TSCP1 this advantage is limited to small areas of the objective space. However, for more restrictive elements the differences become more marked. Some illustrative results for TSCP2 and TSCP4 are given in Fig. 4. Similarly, when comparing TPLS+MOACO to TPLS+MOEA, the former typically reaches better Pareto front approximations (see Figs. 37–40 of the Supplementary material).

When we compare the two types of hybrid, that is, MOACO+PLS and TPLS+MOACO or MOEA+PLS and TPLS+MOEA, most often the advantages are in favor of the hybrids that use a post-processing by PLS. In fact, in the case of the MOACO hybrids, the advantage of MOACO+PLS over TPLS+MOACO is rather large (see Figs. 9–12 of the Supplementary material), while in the case of MOEA+PLS vs. TPLS+MOEA, the advantage of MOEA+PLS over TPLS+MOEA is restricted to some areas of the objective space (see Figs. 21–24 of the Supplementary material).

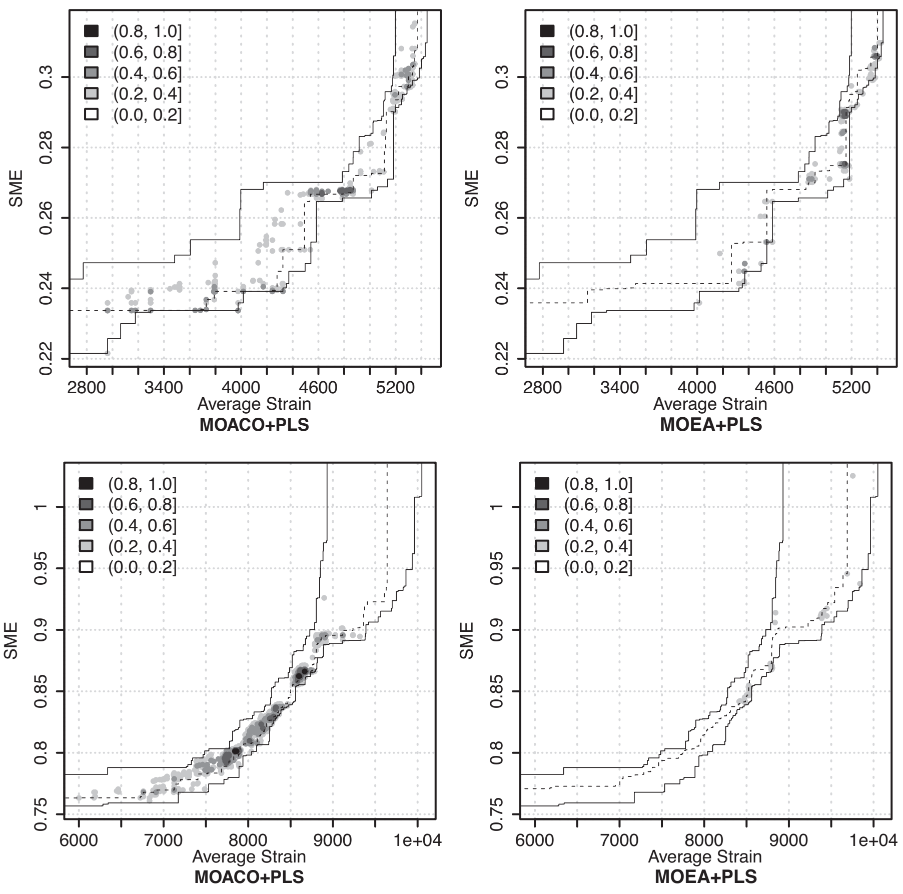

Fig. 4. EAF difference plots for instance TSCP2 and TSCP4 using as objectives SME and average strain. We compare MOACO+PLS to MOEA+PLS after 3000 evaluations. Advantages in favor of MOACO+PLS are indicated on the left side, those in favor of MOEA+PLS are indicated on the right side.

Overall results and statistical analysis

As the comparison of the hybrid and non-hybrid algorithms through EAF differences is not always fully conclusive, we computed the hypervolume indicator for each algorithm. Thus, for each algorithm we obtain 10 independent values for the hypervolume measured on each instance.

In a first step, we perform a statistical analysis of the hypervolume on each instance. Each algorithm is run using at each independent trial the same random number seed as a means to reduce the variance of the results; thus, the common random number seed can serve as a blocking factor in a statistical analysis. We used the Friedmann test and computed the minimum difference of the sum of ranks for which a pair of results is considered significantly different. The results of this statistical analysis are given in Table 4. From these results we can observe that MOACO+PLS performs best: for eight instances it ranks best and it is never statistically significantly worse than the best ranking algorithm. Another high performing algorithm appears to be MOEA+PLS, which for nine instances is among the top three ranking algorithms and never among the worst three of the remaining.

To further summarize the results, we give in Table 5 the ranking of each algorithm across the 12 instances, together with their average ranking. These results confirm the insights obtained by the

Table 4
Results after applying Friedman test and post hoc Friedman tests. In the first column is indicated the instance (first number refers to the number of restrictive elements; the second to the combination of objectives – 1 is SME and average strain, 2 is viscous dissipation and average strain and 3 is viscous dissipation and SME). The second column, Rdiff, gives the minimum significant difference of the sum of ranks for which a result is significantly different from the best ranking algorithm. The other columns give the algorithms ordered according to the rank; an entry $x(y)$ indicates for x the algorithm and for y the difference of the sum of ranks to the best ranked one. The matching between the numbers in the above ranking and the algorithms is as follows: 1 MOACO, 2 MOACO+PLS, 3 MOEA, 4 MOEA+PLS, 5 PLS, 6 TPLS, 7 TPLS+MOACO, 8 TPLS+MOEA, 9 TPLS+PLS.

Instance	Rdiff	Ranking								
1 – 1	20.02	**4 (0)**	**2 (6)**	**1 (19)**	9 (30)	7 (32)	3 (34)	6 (37)	5 (38)	8 (56)
1 – 2	19.98	**2 (0)**	**4 (8)**	**9 (16)**	**1 (17)**	3 (21)	6 (21)	7 (27)	5 (38)	8 (59)
1 – 3	12.84	**7 (0)**	**2 (0.5)**	**1 (7)**	8 (22.5)	9 (41.5)	4 (42)	5 (46.5)	3 (50)	6 (64.5)
2 – 1	17.71	**2 (0)**	**4 (6)**	**1 (17)**	7 (18)	3 (38)	5 (44)	8 (45)	6 (50)	9 (52)
2 – 2	19.75	**2 (0)**	**4 (4)**	7 (23)	1 (26)	9 (31)	8 (42)	6 (43)	3 (44)	5 (48)
2 – 3	17.81	**2 (0)**	**7 (4)**	**1 (9)**	**3 (12)**	8 (19)	4 (24)	9 (34)	5 (48)	6 (57)
3 – 1	21.18	**2 (0)**	**7 (20)**	4 (22)	9 (23)	5 (24)	1 (25)	8 (42)	3 (46)	6 (50)
3 – 2	21.44	**2 (0)**	**1 (5)**	**4 (7)**	**9 (16)**	**5 (17)**	7 (22)	6 (31)	3 (39)	8 (43)
3 – 3	21.77	**2 (0)**	**9 (15)**	**5 (20)**	4 (23)	1 (28)	7 (30)	8 (38)	6 (41)	3 (48)
4 – 1	Inf	**4 (0)**	**5 (1)**	**9 (1)**	**7 (6)**	**2 (7)**	**8 (9)**	**1 (19)**	**6 (24)**	**3 (32)**
4 – 2	21.65	**7 (0)**	**2 (4)**	**4 (6)**	**8 (7)**	**9 (13)**	**1 (19)**	5 (28)	6 (33)	3 (43)
4 – 3	19.95	**2 (0)**	**4 (19)**	5 (21)	7 (23)	1 (25)	9 (38)	8 (39)	3 (47)	6 (58)

Table 5
The rank of each algorithm on the 12 instances. Rank 1 refers to the best algorithm, rank 9 to the worst ranking algorithm. At the bottom of the table, the average rank for each algorithm is shown.

Instance	MOACO+PLS	MOACO	MOEA+PLS	MOEA	PLS	TPLS	TPLS+MOACO	TPLS+MOEA	TPLS+PLS
1 – 1	2	3	1	6	8	7	5	9	4
1 – 2	1	4	2	5	8	6	7	9	3
1 – 3	2	3	6	8	7	9	1	4	5
2 – 1	1	3	2	5	6	8	4	7	9
2 – 2	1	4	2	8	9	7	3	6	5
2 – 3	1	3	6	4	8	9	2	5	7
3 – 1	1	6	3	8	5	9	2	7	4
3 – 3	1	2	3	8	5	7	6	9	4
3 – 3	1	5	4	9	3	8	6	7	2
4 – 1	5	7	1	9	2	8	4	6	3
4 – 3	2	6	3	9	7	8	1	4	5
4 – 3	1	5	2	8	3	9	4	7	6
Avg.	1.67	4.25	2.92	7.25	5.92	7.92	3.75	6.67	4.75

Table 6
Results after applying Friedman test and post hoc Friedman tests to the data from Table 5. Rdiff gives the minimum significant difference of the sum of ranks for which a result is significantly different from the best ranking algorithm. After each algorithm identifier is given in parenthesis the observed difference in the sum of ranks.

Rdiff	Ranking			
17.83	MOACO+PLS (0)	MOEA+PLS (16)	TPLS+MOACO (26)	MOACO (32)
	TPLS+PLS (38)	PLS(52)	TPLS+MOEA (61)	MOEA (68)
	TPLS (76)			

discussion of the EAF differences plots and the ranking results in Table 4. We may also apply a Friedman test on these additional data, using now the instances as blocking factor. In Table 6 we give the results of this test. It confirms MOACO+PLS as the best performing algorithm for the TSCP. It is statistically significantly better than all other algorithms, the only exception being MOEA+PLS. In addition, these results also indicate that overall the hybrid algorithms typically rank better than the non-hybrid search algorithms on which they are based, thus confirming the usefulness of hybridization in the algorithmic approaches to the TSCP.

Conclusions

A main conclusion from our computational results is that for the TSCP hybridization of algorithms improves performance when compared to the underlying, non-hybrid search methods. The hybrid algorithms can be classified into two approaches. The first is to use PLS as a post-processing of the non-dominated set of solutions generated by TPLS, MOACO, or MOEA. In this case, we obtained rather large improvements over the non-hybrid search methods used in the first phase. The second approach to hybridization is to seed the initial set of solutions to PLS, MOACO or MOEA by the non-dominated solutions returned by TPLS. In this case, TPLS used only very few weight vectors to limit its computation time. Although seeding PLS, MOACO, and MOEA by TPLS solutions resulted in a better ranking of the hybrid algorithms than the individual search techniques, the relative improvements according to the EAF differences were rather small. Hence, the conclusion from our experimental analysis is that the post-processing of non-dominated sets by PLS is the most promising hybrid approach. Among these, MOACO+PLS performs best: it reaches the overall best ranking among the nine algorithms we compared and obtains the highest hypervolume on eight of the 12 instances. It also ranks statistically significantly better than all other competing algorithms, except MOEA+PLS.

From the point of view of the twin-screw extrusion problem, we can conclude that the hybrid algorithms allow to potentially identify better screw configurations. Given that the hybrid algorithms obtained, for a same number of screw evaluations, better approximations to the Pareto front, we have also evidence that they are faster to reach high quality Pareto fronts than the non-hybrid ones. This is important as the computation time required to evaluate solutions is high.

There are a number of possible directions for future research on the TSCP. First, we would like to improve the performance of the algorithms by trying to introduce recent improvements on the anytime behavior of PLS [31]. Such extensions will be crucial for adapting our algorithms to three and more objectives. Finally, we would also like to explore more advanced versions of the problem, where in a first step appropriate screw elements have to be chosen from a large set of available ones and then the chosen screw elements be sequenced in the second step. This more complex problem results in much larger search spaces and therefore it provides a significant challenge for future research.

Acknowledgments

This work has been supported by the Portuguese Fundação para a Ciência e Tecnologia under PhD grant SFRH/BD/21921/2005. This work was supported by the Meta-X project funded by the Scientific Research Directorate of the French Community of Belgium. Thomas Stützle acknowledges support from the Belgian F.R.S.-FNRS, of which he is a senior research associate. José Covas and Gaspar-Cunha acknowledge the Portuguese Fundação para a Ciência e Tecnologia under grant PEst-C/CTM/LA0025/2011 (Projecto Estratégico - LA 25 - 2011-2012 - Strategic Project - LA 25 - 2011-2012). The research leading to the results presented in this paper has received funding from the European Research Council under the European Union's Seventh Framework Programme (FP7/2007-2013)/ERC grant agreement n 246939.

References

[1] K. Kohlgrüber, Co-Rotating Twin-Screw Extruders: Fundamentals, Technology, and Applications, Hanser Publishers, Munich, Germany, 2008.
[2] A. Gaspar-Cunha, J.A. Covas, RPSGAe – a multiobjective genetic algorithm with elitism: application to polymer extrusion, in: X. Gandibleux, M. Sevaux, K. Sörensen, V. T'kindt (Eds.), Metaheuristics for Multiobjective Optimisation, Lecture Notes in Economics and Mathematical Systems, Springer, Heidelberg, Germany, 2004, pp. 221–249.
[3] A. Gaspar-Cunha, J.A. Covas, B. Vergnes, Defining the configuration of co-rotating twin-screw extruders with multiobjective evolutionary algorithms, Polym. Eng. Sci. 45 (2005) 1159–1173.
[4] A. Gaspar-Cunha, A. Poulesquen, J.A. Covas, B. Vergnes, Optimization of processing conditions for polymer twin-screw extrusion, Int. Polym. Process. 17 (2) (2002) 201–213.
[5] C. Teixeira, J. Covas, F. Berzin, B. Vergnes, A. Gaspar-Cunha, Application of evolutionary algorithms to the definition of the optimal twin-screw extruder configuration for starch canonization, Polym. Eng. Sci. 51 (2) (2010) 330–340.
[6] C. Teixeira, A. Gaspar-Cunha, J. Covas, Flow and heat transfer along the length of a co-rotating twin screw extruder, Polym. Plast. Technol. Eng. 51 (15) (2012) 1567–1577.

[7] C. Teixeira, J. Covas, T. Stützle, A. Gaspar-Cunha, Engineering an efficient two-phase local search for the co-rotating twin-screw configuration problem, Int. Trans. Oper. Res. 18 (2) (2011) 271–291, http://dx.doi.org/10.1111/j.1475-3995.2010.00798.x.

[8] C. Teixeira, J. Covas, T. Stützle, A. Gaspar-Cunha, Optimization of co-rotating twin-screw extruders using Pareto local search, in: X.-Z. Gao, A. Gaspar-Cunha, M. Köppen, M. Schaefer (Eds.), Soft Computing in Industrial Applications, Vol. 75 of Advances in Intelligent and Soft Computing, Springer, Berlin, Germany, 2010, pp. 3–10.

[9] L. Paquete, T. Stützle, Stochastic local search algorithms for multiobjective combinatorial optimization: a review, in: T.F. Gonzalez (Ed.), Handbook of Approximation Algorithms and Metaheuristics, Chapman & Hall/CRC, Boca Raton, FL, 2007, 29-1–29-15.

[10] C. Teixeira, J. Covas, T. Stützle, A. Gaspar-Cunha, Multi-objective ant colony optimization for solving the twin-screw extrusion configuration problem, Eng. Optim. 44 (3) (2012) 351–371.

[11] E.-G. Talbi, A taxonomy of hybrid metaheuristics, J. Heuristics 8 (5) (2002) 541–564.

[12] C. Blum, J. Puchinger, G.R. Raidl, A. Roli, Hybrid metaheuristics in combinatorial optimization: a survey, Appl. Soft Comput. 11 (6) (2011) 4135–4151.

[13] P. Moscato, Memetic algorithms: a short introduction, in: D. Corne, M. Dorigo, F. Glover (Eds.), New Ideas in Optimization, McGraw Hill, London, UK, 1999, pp. 219–234.

[14] H. Mühlenbein, M. Gorges-Schleuter, O. Krämer, Evolution algorithms in combinatorial optimization, Parallel Comput. 7 (1988) 65–85.

[15] N.L.J. Ulder, E.H.L. Aarts, H.-J. Bandelt, P.J.M. van Laarhoven, E. Pesch, Genetic local search algorithms for the travelling salesman problem, in: H.-P. Schwefel, R. Männer (Eds.), Proceedings of PPSN-I, First International Conference on Parallel Problem Solving from Nature, Vol. 496 of Lecture Notes in Computer Science, Springer, Heidelberg, Germany, 1991, pp. 109–116.

[16] M. Dorigo, T. Stützle, Ant Colony Optimization, MIT Press, Cambridge, MA, 2004.

[17] J. Dubois-Lacoste, M. López-Ibá nez, T. Stützle, Combining two search paradigms for multi-objective optimization: two-phase and Pareto local search, in: E.-G. Talbi (Ed.), Hybrid Metaheuristics, Vol. 434 of Studies in Computational Intelligence, Springer-Verlag, 2013, pp. 97–117, http://dx.doi.org/10.1007/978-3-642-30671-6_3.

[18] C. Teixeira, R. Faria, J.A. Covas, A. Gaspar-Cunha, Modelling flow and heat transfer in co-rotating twin-screw extruders, in: E. Cueto, F. Chinesta (Eds.), 10th ESAFORM Conference on Material Forming, Vol. 907 of AIP Conference Proceedings, Springer, Berlin, Germany, 2007, pp. 980–985.

[19] L. Paquete, T. Stützle, A two-phase local search for the biobjective traveling salesman problem, in: C.M. Fonseca, P.J. Fleming, E. Zitzler, K. Deb, L. Thiele (Eds.), Evolutionary Multi-criterion Optimization (EMO 2003), Vol. 2632 of Lecture Notes in Computer Science, Springer, Heidelberg, Germany, 2003, pp. 479–493.

[20] H.H. Hoos, T. Stützle, Stochastic Local Search – Foundations and Applications, Morgan Kaufmann Publishers, San Francisco, CA, 2005.

[21] L. Paquete, M. Chiarandini, T. Stützle, Pareto local optimum sets in the biobjective traveling salesman problem: an experimental study, in: X. Gandibleux, M. Sevaux, K. Sörensen, V. T'kindt (Eds.), Metaheuristics for Multiobjective Optimisation, Lecture Notes in Economics and Mathematical Systems, Springer, Berlin, Germany, 2004, pp. 177–200.

[22] E. Angel, E. Bampis, L. Gourvés, Approximating the Pareto curve with local search for the bicriteria TSP(1,2) problem, Theor. Comput. Sci. 310 (1–3) (2004) 135–146, http://dx.doi.org/10.1016/S0304-3975(03)00376-1.

[23] K. Deb, Multi-Objective Optimization Using Evolutionary Algorithms, Wiley, Chichester, UK, 2001.

[24] G. Tao, Z. Michalewicz, Inver-over operator for the TSP, in: Parallel Problem Solving from Nature-PPSN V, Vol. 1498 of Lecture Notes in Computer Science, Springer-Verlag, Heidelberg, Germany, 1998, pp. 803–812.

[25] M. Dorigo, G. Di Caro, The ant colony optimization meta-heuristic, in: D. Corne, M. Dorigo, F. Glover (Eds.), New Ideas in Optimization, McGraw Hill, London, UK, 1999, pp. 11–32.

[26] M. Dorigo, G. Di Caro, L.M. Gambardella, Ant algorithms for discrete optimization, Artif. Life 5 (2) (1999) 137–172.

[27] C.M. Fonseca, P.J. Fleming, On the performance assessment and comparison of stochastic multiobjective optimizers, in: H.-M. Voigt, et al. (Eds.), Parallel Problem Solving from Nature, PPSN IV, Vol. 1141 of Lecture Notes in Computer Science, Springer, Heidelberg, Germany, 1996, pp. 584–593.

[28] V. Grunert da Fonseca, C.M. Fonseca, A. Hall, Inferential performance assessment of stochastic optimizers and the attainment function, in: E. Zitzler, K. Deb, L. Thiele, C.C. Coello, D. Corne (Eds.), Evolutionary Multi-criterion Optimization (EMO 2001), Vol. 1993 of Lecture Notes in Computer Science, Springer, Heidelberg, Germany, 2001, pp. 213–225.

[29] M. López-Ibáñez, L. Paquete, T. Stützle, Exploratory analysis of stochastic local search algorithms in biobjective optimization, in: T. Bartz-Beielstein, M. Chiarandini, L. Paquete, M. Preuss (Eds.), Experimental Methods for the Analysis of Optimization Algorithms, Springer, Berlin, Germany, 2010, pp. 209–222, http://dx.doi.org/10.1007/978-3-642-02538-9_9.

[30] E. Zitzler, L. Thiele, M. Laumanns, C.M. Fonseca, V. Grunert da Fonseca, Performance assessment of multiobjective optimizers: an analysis and review, IEEE Trans. Evol. Comput. 7 (2) (2003) 117–132.

[31] J. Dubois-Lacoste, M. López-Ibá nez, T. Stützle, Pareto local search algorithms for anytime bi-objective optimization, in: J.-K. Hao, M. Middendorf (Eds.), Proceedings of EvoCOP 2012 – 12th European Conference on Evolutionary Computation in Combinatorial Optimization, Vol. 7245 of Lecture Notes in Computer Science, Springer, Heidelberg, Germany, 2012, pp. 206–217, http://dx.doi.org/10.1007/978-3-642-29124-1_18.