Scheduling with controllable processing times and compression costs using population-based heuristics

Andreas C. Nearchou

Scheduling with controllable processing times and compression costs using population-based heuristics

Journal:	*International Journal of Production Research*
Manuscript ID:	TPRS-2009-IJPR-0560.R1
Manuscript Type:	Original Manuscript
Date Submitted by the Author:	25-Sep-2009
Complete List of Authors:	Nearchou, Andreas; University of Patras, Department of Business Administration
Keywords:	SCHEDULING, EVOLUTIONARY ALGORITHMS
Keywords (user):	controllable processing times, crash costs

Scheduling with controllable processing times and compression costs using population-based heuristics

Andreas C. Nearchou

Department of Business Administration

University of Patras

26 500 Rio, Patras, Greece

E-mail: nearchou@upatras.gr

Tel.: +30 2610-969980, Fax: +30 2610-969990

Abstract: This paper considers the single machine scheduling problem of jobs with controllable processing times and compression costs and the objective to minimize the total weighted job completion time plus the cost of compression. The problem is known to be intractable, and therefore it was decided to be tackled by population-based heuristics such as differential evolution (DE), particle swarm optimization (PSO), genetic algorithms (GAs), and evolution strategies (ES). Population-based heuristics have found wide application in most areas of production research including scheduling theory. It is therefore surprising that this problem has not yet received any attention from the corresponding heuristic algorithms community. This work aims at contributing to fill this gap. An appropriate problem representation scheme is developed together with a multi-objective procedure to quantify the trade-off between the total weighted job completion time and the cost of compression. The four heuristics are evaluated and compared over a large set of test instances ranging from 5 to 200 jobs. The experiments showed that a differential evolution algorithm is superior (with regard to the quality of the solutions obtained) and faster (with regard to the speed of convergence) to the other approaches.

Key words: scheduling, controllable processing times, crash costs, meta-heuristics, differential evolution, swarm intelligence, genetic and evolutionary algorithms, evolution strategies.

1. Introduction

Scheduling involving controllable job processing times has received increasing research attention in the last two decades due to its compliance with the real needs of modern production systems. In such systems, production managers usually face the problem of scheduling the jobs at faster processing times with higher costs. Jobs' processing times are usually controllable (compressible) by allocating additional resources, such as working overtime, performing subcontracting, running machines at higher speeds, consuming more energy, fuels, etc. The efficient coordination of job scheduling and resource allocation decisions is a critical factor for a modern production system to achieve competitive advantage.

This paper considers the problem of scheduling multiple jobs with controllable processing times on a single machine and the objective to minimize the total weighted job completion time plus the cost of compression. We will refer to this problem as TWJCTP for short. TWJCTP is NP-hard (Wan *et al.* 2001, Hoogeveen and Woeginger 2002) and consequently the right way to proceed is through the use of heuristic techniques. In all our knowledge no other heuristic exists in the literature for directly solving large size instances of TWJCTP. This paper aims at contributing to fill this gap facing TWJCTP by means of modern population-based heuristics namely differential evolution (DE), particle swarm optimization (PSO), genetic algorithm (GA), and evolution strategies (ES). The performance of each one of them is examined under the influence of two different encoding schemes necessary for mapping the genotypes (evolving vectors) to phenotypes (actual job schedules). A new, simple control scheme for estimating the control parameters settings for the case of DE, PSO and GA is also presented. The proposed control scheme is adaptive and found superior to traditional deterministic control schemes with regard to the quality of solutions obtained.

The scheduling problem with controllable processing times and costs has been studied by researchers such as (Vickson 1980a, 1980b, Van Wassenhove and Baker 1982, Daniels and Sarin 1989, Zdrzalka 1991, Panwalkar and Rajagopalan 1992, Alidace and Ahmadian 1993, Guochun and Foulds 1998, Biskup and Cheng 1999, Foulds and Guochun 1999, Wan *et al.* 2001, Hoogeveen and Woeginger 2002). Vickson (1980a, 1980b) initiates the topic, first, by considering the problem with the objective of minimizing the total flow time and the total processing cost incurred due to the job processing time compression (Vickson 1980a). Then by considering the problem of minimizing the total flow and resource costs under the assumption that the job flow costs are identical (Vickson 1980b). Van Wassenhove and Baker (1982) proposed an algorithm to determine the trade-off curve between maximum tardiness and total amount of compression on a single machine. Later, Daniels and Sarin (1989) extended the work of Van Wassenhove and Baker (1982) by considering the additional constraint of allowed maximum job tardiness. Zdrzalka (1991) considered a single machine scheduling problem in which each job has a release date, a delivery time and a controllable processing time; and gave an approximation algorithm for minimizing the overall schedule cost. Panwalkar and Rajagopalan (1992) considered the common due date assignment and single machine scheduling problem in which the objective is the sum of penalties based on earliness, tardiness and processing time compressions. The authors showed that the problem can be reduced to a linear assignment problem. Their results extended later by Alidace and Ahmadian (1993) to the parallel machine scheduling case. Biskup and Cheng (1999) also extended the work of Panwalkar and Rajagopalan (1992) by adding the total completion time in the objective function, and showed that the extended problem can be solved as an assignment problem. An early survey with results on the specific research field can be found in (Nowicki and Zdrzalka 1990). An up-to-date extended survey in the field together with a unified framework for the related scheduling problems can be found in the recent paper of Shabtay and Steiner (2007).

The rest of this paper is organized as follows: Section 2 formulates TWJCTP. Section 3 describes very briefly DE, PSO, GA and ES. Section 4 introduces the way the four population-based heuristics can be applied on TWJCTP, while Section 5 presents and discusses the results of the experimental evaluations of the algorithms. Finally, Section 6 summarizes the contribution of the paper and states some directions for future work.

2. Problem formulation

To facilitate the presentation, the following notations are used throughout the paper:

n	number of jobs
J_i	job i
π	a job sequence (a schedule) of the n jobs
$\pi[i]$	job in the i-th position of sequence π
np_i	normal (initial) processing time of job i
y_i	amount of compression of job i
u_i	maximum permitted amount of compression for job i
ap_i	actual processing time of job i
ϕ_i	unit cost of compressing job i
C_i	completion time of job i
w_i	weight factor of job i
$TWCT$	*total weighted completion time* of a given π schedule
CoC	*cost of compression* of a given π schedule

TWJCTP formally written as $1/contr/\sum w_i C_i$ (Hoogeveen and Woeginger 2002) can be defined as follows: consider a set of n independent jobs $\{J_1, J_2, \ldots, J_n\}$ to be processed without interruption on a single machine that can handle only one job at a time. Each job J_i (i=1,…,n) is available at time zero, and its initial (normal) processing time np_i can be compressed by an amount y_i ($0 \le y_i \le u_i$) with u_i being an upper bound on the compression ability of J_i. Hence, the actual processing time of J_i is estimated by $ap_i = np_i - y_i$. Performing this compression incurs a cost $\phi_i y_i$, with ϕ_i being the unit cost of compressing J_i. Let C_i the completion time of job J_i (i=1,…,n) in some schedule, and w_i a weighted factor corresponding to J_i. Then, the objective of TWJCTP is to determine a job sequence π for the jobs, and a corresponding compression vector $\mathbf{y} = (y_1, \ldots, y_n)$ (with the compressions of all the jobs in π) that minimizes

$$f(\pi, \mathbf{y}) = f_{TWCT} + f_{CoC} = \sum_{i=1}^{n} w_{\pi[i]} C_{\pi[i]} + \sum_{i=1}^{n} \phi_{\pi[i]} y_{\pi[i]} \tag{1}$$

The completion time of $\pi[i]$ is estimated by

$$C_{\pi[i]} = \sum_{j=1}^{i} \left(np_{\pi[j]} - y_{\pi[j]} \right) \tag{2}$$

Hence, f is a bi-criteria objective function composed by total weighted completion time of the n jobs in π, and the corresponding total cost of compressing these jobs. According to the literature (Hoogeveen and Woeginger 2002, Hoogeveen 2005), four variants of the basic TWJCTP arise:

P1: to minimize the total cost given by Eq.(1),

P2: to minimize $f_{TWCT} = \sum_{i=1}^{n} w_{\pi[i]} C_{\pi[i]}$ under the constraint $f_{CoC} = \sum_{i=1}^{n} \phi_{\pi[i]} y_{\pi[i]} \le A$,

P3: to minimize f_{CcC} under the constraint $f_{TWCT} \le B$,

P4: to identify the trade-off curve for (f_{TWCT}, f_{CoC}).

This paper studies problem P4. The trade-off curve connects all points (f^*_{TWCT}, f^*_{CoC}) where f^*_{TWCT} is the best possible value of f_{TWCT} given that $f_{CoC} \le f^*_{CoC}$ and vice versa. In other words, P4 consists in identifying the set of Pareto-optimal solutions for (f_{TWCT}, f_{CoC}), i.e., the set of solutions that are not dominated by any other solution in the search space when all the objectives are considered, and they do not dominate each other in the set.

Lemma 1: There is always an optimal schedule to TWJCTP in which every job J_i (i=1,…,n) is either totally uncompressed (y_i=0), or totally compressed ($y_i = u_i$). The proof can be found in Hoogeveen and Woeginger (2002).

3. Population-based heuristics

DE (Storn and Price 1997), PSO (Eberhart and Kennedy 1995), GAs (Holland 1975) and ES (introduced in the early of 1960s by Rechemberg and Schwefel; see Fogel (1995) for a detailed description) belong to a modern class of heuristics known as *evolutionary algorithms*. Independently of the form of the optimization problem, any evolutionary algorithm undergoes the following general operation mechanism (Michalewicz and Fogel 2000):

(a) Create (usually in a random way) a population S of individuals that represent potential solutions to the physical problem.

(b) Evaluate the quality of each individual in S.

(c) Reward individuals of higher quality (so that to survive and reproduce their structure in the next generation) by introducing selective pressure on S.

(d) Generate new individuals by applying variation operators on S.

(e) Repeat steps (b)-(d) several times until the satisfaction of a suitable termination criterion.

The main differences between DE, PSO, GAs, and ES rely on the way they perform steps (c) and (d). In particular, DE attempts to replace in each generation all the individuals in S by new, better solutions. Each individual solution becomes a target for replacement by a competitor called trial solution. Mutation and crossover operators are used to create a trial for each target solution. A one-to-one comparison between targets and trials determines the new members in S. Similarly, PSO attempts to replace all the solutions in S with better solutions, by exploiting information such as the current quality of an individual, its own best quality in history, and the quality of its neighbours. A GA from the other site replaces only a subset of S using a suitable parent selection strategy. Offspring are created by applying crossover and mutation operators on the selected population subset. In ES the whole population is seen as the parent. From this population an offspring population is generated and evaluated. The new S is created deterministically, either from the best members of the offspring population ((μ, λ)-selection), or from the union of parents and offspring ((μ+λ)-selection). Offspring are created using simple recombination between the alleles of the parent strings, followed by a self-adaptive mutation scheme based on Gaussian perturbations.

DE, PSO, and ES are stochastic optimizers over continuous search spaces, meaning that they utilize real-valued vectors to represent solutions of the physical problem. While a GA can be found in various forms such as binary-coded, permutation-coded, or real-valued, depending on the way it codes the solution space of the physical optimization problem. To achieve a fair comparison between the heuristics, it was decided to develop a real-valued GA (rGA) in this work. That is, genotypes (individual solutions in S) are floating-point vectors as in the case of DE, PSO, and ES.

4. Problem representation: mapping real-valued vectors to TWJCTP solutions

For the application of these heuristics on TWJCTP one must decide how to decode the real-valued vectors maintained and evolved (the genotypes) to actual TWJCTP solutions (phenotypes). For a *n*-job TWJCTP a candidate solution consists of a job sequence π, i.e., a permutation of the integers 1,2,…,*n*, together with a compression vector $\mathbf{y}$, i.e., a string of integers $y_{\pi[1]},\ldots,y_{\pi[n]}$ (with $y_{\pi[i]} \in [0,\ u_{\pi[i]}]\ \forall\ i \in [1,n]$). To that purpose, a real-valued vector containing 2*n* real numbers has been made was selected for use (see Fig. 1). The *n* most left components of the vector corresponds to π and its *n* most right components corresponds to $\mathbf{y}$. In the following we will refer to the two parts of the vector with *π-part* and *y-part* for short, respectively.

< Insert Figure 1 about here >

4.1. Creating a job sequence from a real-valued vector

After the decision about the structure of the implemented genotype, a way to map this structure to an actual TWJCTP solution must be determined. In the literature there are at least two different encoding schemes for representing permutations through real-valued vectors namely, *random keys* (Bean 1994) and *sub-range keys* (Nearchou 2006). Both of them were adopted and their influence on the performance of the examined heuristics was investigated. The application of the two encoding schemes on TWJCTP is explained below through a simple example.

< Insert Table 1 about here >

Let us assume a 5-job TWJCTP with the characteristics given in Table 1. Furthermore, let us also assume that in some point in time the following real-valued vector was generated by a heuristic:

$$x = (\overbrace{0.75, 0.42, 0.10, 0.27, 0.62}^{\pi-part}, \underbrace{0.31, 0.04, 0.60, 0.15, 0.91}_{y-part}) \tag{3}$$

Random keys work as follows: the numbers in *π-part* of x are sorted and their order in x determines π. That is, (0.75, 0.42, 0.10, 0.27, 0.62) corresponds to the job sequence (3–4–2–5–1). Since the 3rd number in the vector has the lowest value (=0.10), followed by the 4th number in the vector, which is the second smallest number (=0.27), etc.

Sub-range keys work as in the following: the range [1…*n*] is divided into *n* equal sub-ranges and the upper bound of each sub-range is saved in an array $SR=[1/n, 2/n, \ldots, n/n]^T$ (*SR* stands for Sub-Ranges). Then, we take each number from *π-part* and determine the sub-range in which this number belongs. The order of these sub-ranges in *SR* constitutes the final solution. For the above example, *n*=5 and thus, $SR = [0.2, 0.4, 0.6, 0.8, 1.0]^T$. The first number from *π-part* (=0.75) lies in the fourth sub-range (0.6<0.75≤0.8), therefore the resulting schedule is (4 _ _ _ _). The second number (=0.42) lies in the third sub-range (0.4<0.42≤0.6), and so on. Finally, the generated (by *sub-range keys*) schedule is (4 3 1 2 4).

As it is clear, this schedule is illegal since it contains duplicated jobs. To produce a valid version of the schedule the following very simple two-steps repairing procedure is applied on the proto-schedule:

(a) Delete all the duplicate jobs: (4 3 1 2 _)

(b) Fill the empty locations in the schedule with the remaining (unused) jobs following an ascending order of their values: (4 3 1 2 **5**)

Therefore, (0.75, 0.42, 0.10, 0.27, 0.62) corresponds to the job sequence (4–3–1–2–5), which is different from that obtained by *random-keys*.

4.2. Creating a compression vector from a real-valued vector

It is now the time to obtain the compression vector $\mathbf{y}$ corresponding to x. To achieve this mapping, the following encoding mechanism is proposed

$$y_j = \begin{cases} 0, & \text{if } x^j \leq 0.5 \\ u^{\pi[j-n]}, & \text{otherwise} \end{cases} \quad \text{for all } j=n+1, n+2, \ldots, 2n \tag{4}$$

where y_j (j=n+1, n+2,…,2n) denotes the amount of compression for the job lying in the (j-n) position of π. $u^{\pi[j-n]}$ denotes the maximum permitted amount of compression for this job, and x^j is the corresponding real number in *y-part* of x. If x^j is less than or equal to 0.5 then job $\pi[j-n]$ is not compressed at all, or is totally compressed. Note that, Eq. (4) implements lemma 1 for optimal TWJCTP solution.

Therefore, applying Eq. (4) on the above example (Eq. (3)), results in a mapping of *y-part* into vector $\mathbf{y}$=(0,0,8,0,4) if *random-keys* method is adopted; and $\mathbf{y}$=(0,0,4,0,7) if *sub-range keys* method is adopted. The total cost for the two TWJCTP solutions $(\pi;\mathbf{y})_{random\text{-}keys}$=(3,4,2,5,1;0,0,8,0,4) and $(\pi;\mathbf{y})_{sub\text{-}range\text{-}keys}$=(4,3,1,2,5;0,0,4,0,7), are thereby 1415 and 1664 units, respectively.

5. A multi-objective procedure for TWJCTP

The attempt with the multi-objective problem (MOP) studied in this paper is to find a Pareto set of optimal solutions for (f_{TWCT}, f_{CoC}) (see Eq.(1)). A Pareto set contains all those not dominated solutions to TWJCTP, such that no other solutions are superior to them in respect to both the objectives shown in Eq.(1). In order to determine a Pareto set of TWJCTP solutions, the operation mechanism of each one of the four heuristics under consideration was enhanced with the following two main features:

a) A separate secondary population of diverse Pareto-optimal TWJCTP solutions is maintained and iteratively updated from generation to generation. This population will be composed of all Pareto solutions found during the search.

b) The main population is iteratively updated using an elitist preserving strategy. Based on this strategy, a portion of the main population is randomly replaced by a number of elite Pareto solutions.

In MOP a solution with the best values for each objective can be regarded as an elite solution. Hence, for TWJCTP there are two elite (extreme) solutions in the evolving population each of which optimizes one objective. These solutions are candidates to be copied into Pareto population. Pareto set is further completed by additional elite solutions using the procedure given below. A Pareto population of the final generation contains the near-optimal solutions to TWJCTP. The decision maker can then select that solution accomplishing more her or his preferences.

Procedure Pareto_Population

Input: (a) The main population of solutions S and its size Ns;

(b) The Pareto population Γ and its size Γ_size.

Output: The updated versions of S and Γ.

Begin

// Check each one of the individual solutions in S whether constitutes a Pareto solution //

cS=cΓ=0 ; // initialize counters for the members in S and Γ, respectively //

While ($c\Gamma \leq \Gamma$_size) **and** ($cS \leq Ns$) **do**

cS=cS+1;

Compare S(cS) with all Pareto solutions in Γ; // S(cS) is the cS^{th} member of S //

If S(cS) is not contained in Γ **then**

If S(cS) dominates some Pareto solutions **then**

Add S(cS) into Γ and delete the solutions dominated by it;

Increment accordingly counter cΓ;

Else if there is empty space in Γ **then**

Add S(cS) into Γ.

Increment accordingly counter cΓ;

Endif

Endif

Endwhile

// apply elitist preserving strategy //

Determine the two elite Pareto solutions in Γ;

Replace two randomly selected members in S with the two elite Pareto;

Return (S, Γ);

End;

5.1 Fitness assignment mechanism

A critical question arising when facing a MOP by the means of evolutionary algorithms is how to estimate the fitness function of individual solutions with regard to the multiple objectives. A simple method to combine multiple objective functions into a composite fitness solution is the well-known weighted-sum method. According to this method, the MOP under consideration is written as in the following:

$$\min f(\pi, \mathbf{y}) = \omega_1 \cdot f_{TWCT} + \omega_2 \cdot f_{CoC} = \omega_1 \cdot \sum_{i=1}^{n} w_{\pi[i]} C_{\pi[i]} + \omega_2 \cdot \sum_{i=1}^{n} \phi_{\pi[i]} y_{\pi[i]} \quad (5)$$

The weights ω_1 and ω_2 specify the relative importance of the corresponding objectives. The determination of the suitable values for these weights is in general a difficult task and constitutes another critical research issue in multi-objective optimization. In the literature, there are three general methods to compute the weights ω_i (i=1,…,Q) for a weighted-sum objective function with Q objectives: the fixed-weight method, the random-weight method and the adaptive-weight method. The former uses constant weights satisfying the relation,

$$\sum_{i=1}^{Q} \omega_i = 1, \quad \omega_i > 0 \;\; for \text{ all } \; i = 1,\ldots,Q \tag{6}$$

However, as Murata *et al.* (1996) showed, using constant weights within an evolutionary algorithm the search direction is fixed, and for this reason it is difficult for the search process to obtain a variety of not dominated solutions. To overcome this drawback, Murata *et al.* (1996), proposed the use of random weights according to the following formula,

$$\omega_i = \frac{random_i}{\text{random}_1 + random_2 + \cdots + random_Q} \qquad i = 1,2,\ldots,Q \tag{7}$$

where $random_i$ (i=1,…,Q) are non-negative random numbers.

Furthermore, Gen and Cheng (2000) proposed an adaptive-weight method which readjusts the weights by utilizing some useful information from the current population. This method computes the weights by,

$$\omega_i = \frac{1}{z_i^{\max} - z_i^{\min}}, \;\; for \text{ all } \; i = 1,\ldots,Q \tag{8}$$

where $z_i^{\max}$ and $z_i^{\min}$ are the maximum and minimum values of the i^{th} objective in the population, respectively.

After much experimentation with the above methods we found the random-weight method superior to the others with regard to the quality of the solutions obtained, and hence it was decided to adopt this method in our study.

6. Computational experiments

6.1 Data generation

To examine the performance of the heuristics on TWJCTP, multiple experiments were performed over a set of test problems with n=5, 10, 20, 50, 100 and 200 jobs. No standard TWJCTP data are reported in the literature and for this reason it was decided to proceed with the random generation of this data set. For each category of problems 10 test instance were generated as in the following: for every job J_i (i=1,2,…,n) four integer quantities were randomly drawn from discrete uniform distributions: the job's initial normal processing time $np_i \in [1,100]$, an upper bound of the compression permitted for this job $u_i \in [0.6\times np_i, 0.9\times np_i]$, the unit cost of compressing this job $\phi_i \in [2,9]$, and a weight factor related to the time completion of the job $w_i \in [1,15]$. The author will be glad to distribute this data set to any reader who is interested hoping that this will become a common test bed for TWJCTP.

The performance of the heuristics was quantified through the use of the following indices:

(a) ***Index P***: denoting the number of the different Pareto solutions generated by a heuristic over a specific test instance.

(b) ***Index P****: denoting the number of not dominated solutions among all Pareto solutions obtained by all the heuristics. In particular, since some Pareto solutions obtained by one heuristic may be dominated by other heuristics, all the obtained solutions are compared to each other and the not dominated among them are selected.

(c) The **quality ratio P*/P** in percentage. The larger the value for this ratio for a given heuristic, the higher the performance of the heuristic.

(d) The actual processing time consumed in seconds.

To get the average performance of the heuristics, each one of them was run 20 times over every test instance (starting each time from a different random number seed) and the solution quality was averaged. Hence, as there are 10 instances in each one of the 6 categories of problems, this means that each heuristic was run 6×10×20 times=1200 times in total. All the heuristics were coded in Borland Pascal and run on an IBM-compatible PC with the following hardware and software specifications: an AMD Dual Core 2.11 GHz processor, 2.0 GB of RAM, and Microsoft Windows XP Professional operating system.

6.2. Choice of the control parameters

When designing a population-based heuristic among others, one has to decide about the size *Ns* of the population used. This is a common parameter for all the examined heuristics. To make the heuristics comparable it was decided to use the same $Ns=n$ population size for all, and limit the running process of each one of them to a maximum of $3n$ CPU seconds. This means a maximum running time for them equal to 15 sec for 5-job problems, 30 sec for 10-job problems, etc. The size of the Pareto population was defined to be equal to 50 for the small size instances ($n \leq 20$) and equal to 100 for the large size instances ($n \geq 50$). It is worth pointing out that the basic data structures required are identical for all compared algorithms.

The settings for the additional control parameters involved in DE, PSO and rGA were determined after experimenting with two different control schemes: a static scheme consistent to the general indications of the literature, and a proposed dynamic control scheme with which some parameters are fixed during the search process while some other parameters are altered according to the diversity of the entire population. These control schemes are described below in detail, while a synopsis of them is given in Table 2.

< *Insert Table 2 about here* >

a) *DE - static control scheme*: Crossover rate (C_R) was defined to take values within the discrete range {0.01, 0.1, 0.5, 0.7, 0.9} while varying F (a scaling positive parameter used in the creation of the mutant vectors) to take a value within the range {0.5, 0.75, 0.95}. That is, this scheme results in 15 different combinations of (C_R, F).

DE - dynamic control scheme: C_R was set equal to 0.01 and F being adapted within the range [0.4, 1.0] as in the following. At the beginning of the search process, F is high ($F=F_0=1$) and decreases slowly by a factor $\vartheta=0.9$ using the relation $F=\vartheta\times F$. When the population's diversity becomes too 'small', or F becomes lower than 0.4, then it takes again its initial high value ($F=F_0$). A small diversity of the population is encountered when the fitness of the worst member of the population ($fitness_{worst}$) and the average population fitness ($fitness_{avg}$) are almost the same. That is, F is being estimated by the following rule:

$$if\ (fitness_{worst} \geq 0.95 \times fitness_{avg})\ or\ (F < 0.4)\ then\ \ F = F_0\ \ else\ \ F = \vartheta \times F \qquad (9)$$

It is also highlighted that mutant vectors in DE were implemented using the standard DE1 scheme (Storn and Price, 1997).

b) *PSO - static control scheme*: We followed the indications of Kennedy (1998), Parsopoulos and Vrahatis (2002). Firstly, c_1 and c_2 (*cognitive* and *social* parameters) were both set to a fixed and equal value within the range {0.5, 1.0, 2.0}. Then, since some recent works (Parsopoulos and Vrahatis 2002) report that it might be better to choose $c_1>c_2$, with $c_1+c_2\leq 4$, experiments were also performed using the combination $c_1=2$, $c_2=1.5$. The experimental investigations led us to adopt the latter settings since higher quality solutions were encountered. Furthermore, inertia weight factor Iw was defined to gradually decreased from $Iw_0=1.2$ towards 0.4 using the relation $Iw=\Theta\times Iw$ (with $\Theta=0.95$). If Iw becomes lower than 0.4 then it is reset to Iw_0.

PSO – dynamic control scheme: c_1 and c_2 are estimated as in the static scheme. Iw by the relation $Iw=\Theta\times Iw$, starting from Iw_0 and being reset to this value if the following condition exists: ($Iw<0.4$) or ($fitness_{worst}\geq 0.95\times fitness_{avg}$).

c) *rGA, - static control scheme*: We experimented with various recommended crossover and mutation rates (Goldberg 1989) such as $C_R\in\{0.6, 0.8\}$ and $M_R\in\{0.01, 0.0333, 1/n, 0.1\}$.

rGA, - dynamic control scheme: A fixed crossover rate equal to 1.0 was defined, and an adapted mutation rate $M_R\in[0.1, 0.8]$ which is high at the beginning and decreases slowly by the population's diversity. When the population's diversity becomes too 'small', then M_R takes again its original high value. More specifically, M_R is initially defined equal to 0.8, and decreased in each new generation by a factor $\vartheta=0.9$ using the relation $M_R=\vartheta\times M_R$. Similarly, to DE and PSO, a small population's diversity is encountered when the minimum population fitness and the average population fitness are almost the same. M_R is reset to 0.8 if the following condition is satisfied: ($M_R<0.0333$) or ($fitness_{\text{worst}}\geq 0.95\times fitness_{\text{avg}}$).

The three genetic operators, i.e., selection, crossover, and mutation were implemented through binary tournament selection, one-point crossover and uniform mutation, respectively. These operators were found to be the best among a set of known operators with regard to the quality of the solutions obtained.

After much experimentation with the above schemes the following best settings for the control parameters were determined: for DE, the dynamic scheme i.e., (C_R, F)=(0.01, adapted within the range [0.4, 1.0] using Eq.(9)). For PSO, the static scheme (c_1, c_2, Iw_k)=(2, 1.5, adapted within the range [0.4, 1.2]). For rGA, the static scheme with (C_R,M_R)=(0.8, 0.01). All the results presented below are conducted by these settings.

d) For the case of ES, we followed the recommendations of the literature (Eiben and Smith 2003). After experimentation for choosing the correct survivor selection scheme, we found that using (μ,λ)-scheme within ES results in a much superior optimizer to that of using (μ+λ)-scheme. For this reason (μ,λ)-scheme was adopted. μ was estimated using the heuristic rule λ/μ=7. Mutation was performed through Gaussian perturbation. The mutation step sizes σ were estimated using self-adaptation through a suitable formation of the genotype structure. In particular, for n-job TWJCTP, the genotype in ES has the form $\{x^1,\ldots,x^n;\sigma^1,\ldots,\sigma^n\}$. x part of the genotype denotes the real-valued vector solution and σ part of the genotype the mutation step size corresponding to each component of the real-valued vector. Offspring were generated using discrete recombination for x values and intermediate recombination for σ values.

6.3. Comparative results

For each one of the examined heuristics two versions were implemented corresponding to a distinct encoding scheme, either to *random-keys*, or to *sub-range keys*. In the following we will refer to them by the abbreviations xx1 (meaning heuristic xx with *random-keys*) and xx2 (heuristic xx with *sub-range keys*) for short. Depending on the problem size the following (see Table 3) average processing times were needed. These times correspond to the mean CPU time spent by each heuristic till the creation of the best individual solution within the permitted running duration. As can be seen from Table 3, DE heuristics seem to be the fastest optimizers especially for large size problems. Almost identical average convergence times are reported for all the heuristics on the small size problems. While rGA and ES appear to have the slowest rate of convergence for problems with size greater than 50.

< Insert Table 3 about here >

Table 4 displays the number of unique Pareto solutions obtained by each one of the heuristics after a single trial over the 60 test beds. For example, for 5-job problems (n=5) in the case of the first test instance, three of the heuristics (de1, rga1, es2) determined 13 different Pareto solutions, while each one of the rest five heuristics determined 14 Pareto solutions. For 10- and 20-job problems, PSO heuristics (pso1, pso2) managed to obtain a larger set of Pareto solutions than that obtained by the other heuristics. For 50-job problems de1 and pso2 outperformed all the others with the latter being slightly better. While, in the case of the most difficult classes of problems (n=100, 200) DE heuristics (de1, de2) showed the highest performance, generating a larger set of Pareto solutions than that obtained by the other approaches.

< Insert Table 4 about here >

Fig. 2 illustrates the not dominated solutions obtained by the heuristics after a single run over the first test instance of each class of problems with $n \geq 20$. As one can see from this figure, in the case of $n=20$ test instance (Fig. 2(a)), the solutions obtained by PSO heuristics (▲ and ▿) and those obtained by de2 (○) are of higher quality (lower curves) than those obtained by the other approaches. Similar high performance for pso1, pso2 and de2 can be also observed for the large size instances. In the most difficult class of problems ($n=200$) de2 and pso2 clearly outperforms all the other approaches. Poor performance was encountered in the case of de1 and rga1 heuristics (higher curves in Fig.2). Some solutions obtained by them have small values of TWCT and others have small values of CoC, while very few solutions have small values of both objectives if compared with the not dominated solutions obtained by the other heuristics. The performance of ES approaches is higher than that of de1 and rga1, but inferior than that of pso1, pso2 and de2.

< Insert Figure 2 about here >

The above discussion concerns the results obtained by the heuristics after a single trial over each one of the 60 test instances. Due to the stochastic behavior of these heuristics, one must evaluate their average performance after multiple trials on the test beds. Hence, each heuristic was applied 20 times on each one of the 60 test instances. After each trial, we first reported the unique Pareto solutions (index P) obtained by each heuristic. Then, all the obtained solutions were compared to each other, and the not dominated among them were selected (index P*). The values of P and P* were averaged over the 20 trials of each heuristic on every test instance. Recall that, every new trial was starting from a different random number seed (same for all the examined heuristics). Tables 5 and 6 display the final averaged values of P and P* respectively, on the examined test instances. To make things more clearly, we describe some lines of these tables. For example, let us take the case of the 1st instance of 100-job problems ($n=100$) and measure P (Table 5). As one can see, best results for this instance were obtained by de1 (P=25.4) and worst results by es2 (P=14.6). That is, de1 was found able to generate a Pareto set of 25.4 unique solutions in average, while the corresponding ability of es2 was a Pareto set of 14.6 solutions in average. But how good were the obtained solutions? The answer to this question can be found in Table 6. In particular, for the specific benchmark ($n=100$, 1st instance), only 0.2 solutions in average from the Pareto set obtained by de1 (P*=0.2) were not dominated by any other solution. This is the smallest P* value among the examined heuristics for the specific test instance. Meaning that, although the variety of the solutions obtained by de1 was in average greater than that of the other approaches (P=25.4, from Table 5); almost all of them were of very poor quality and being dominated by the other Pareto solutions. This information is illustrated more clearly in

Fig 2(c) mentioned above. Furthermore, some other observations for this benchmark (see Table 6) are the following: (a) the highest P* value was encountered by de2 (P*=18.4). Which means that, 18.4 solutions (in average) out the 23.6 (P index) in total, were not dominated by any other Pareto solution. (b) de2 is by far the most effective heuristic. (c) The second best performance is due to pso1 (P*=4.4).

< Insert Table 5 about here >

< Insert Table 6 about here >

Table 7, gives a synopsis of the results shown in Tables 5 and 6. Particularly, Table 7 displays the mean values of P and P* (Table 7(a)) and quality ratio (P*/P) (Table 7(b)) averaged over the 10 instances of each different benchmark class. As can be seen from Table 7(b), best results are due to de2 which achieved a % quality ratio equal to 98.5, 85.5 and 43.1 for small size problems (n=5, 10 and 20), respectively; and a quality ratio equal to 65.7% for 50-job problems, 71% for 100-job, and approximately 75.5% for 200-job problems. The second best performance was achieved by PSO heuristics with pso2 being slightly better than pso1 for large size instances (n=100, 200). The worst performance was due to de1 and es1. Furthermore, examining the influence of the two encoding schemes (*random-keys* and *sub-range keys*) on the performance of the examined heuristics, one can safely conclude that *sub-range keys* are more suitable to be used within DE. For the other three approaches both coding schemes seem to perform almost the same, with *random-keys* being slightly more suitable within PSO and *sub-range keys* being more suitable within rGA and ES.

< Insert Table 7 about here >

Finally, to verify the correctness of the reported results it was decided to test the performance of the heuristics over a similar scheduling problem for which polynomial time exact algorithms exists. The additional experiments were performed on problem $1/contr/\sum C_i$. The objective of this problem is to minimize the total job completion time plus the total cost incurred due to job processing time compression. As Vickson (1980a) showed, assuming equal weight factors ($w = n$) for all the n jobs, this scheduling problem can be formulated as an assignment problem and solved to optimality by an assignment algorithm such as the famous Kuhn's Hungarian algorithm (Bazaraa *et al.* 1990). As test beds we select the test instances with n=5 and n=10 included in the benchmarks data set described above. Table 8 reports the optimum solutions obtained by Vickson's method for these instances. All of the examined heuristics found rather easily the particular optimum solutions so it was decided to withhold the associated results. Fig. 3 depicts the optimum schedule corresponding to the first instance of the examined 10-job scheduling problem. The jobs' characteristics for this instance are given in Table 9. Note that, the first four jobs in the generated optimum schedule (i.e. jobs 3, 6, 10, 1) are crashed while the other jobs are not crashed.

< Insert Table 8 about here >

< Insert Figure 3 about here >

< Insert Table 9 about here >

7. Conclusions

Scheduling jobs with controllable processing times on a single machine and objective to minimize the total weighted job completion time plus the cost of compression is an *NP*-hard bi-criteria combinatorial optimization problem. Therefore, large size instances of the problem must be tackled through the use of heuristics. This paper examined the performance of four known population-based heuristics, namely, differential evolution (DE), particle-swarm optimization (PSO), genetic algorithm (GA), and evolution strategies (ES), for the solution of this problem. An appropriate problem representation was developed and two different encoding schemes for mapping the genotypes (evolving vectors) to phenotypes (actual job schedules) were investigated, namely *random-keys* and *sub-range keys*, respectively. Furthermore, a new technique for dynamically estimating the correct settings of the heuristics' control parameters was presented and examined to improve their efficiency. Extensive experiments were performed over a set of randomly generated test problems with up to 200 jobs. The results obtained showed that DE with *sub-range keys* is by far superior to the other approaches with regard to the quality of the solutions obtained; and rather faster with regard to the speed of convergence to near-optimal solutions.

On going research considers the common due date single machine scheduling problem of a number of jobs with controllable processing times. This type of scheduling sets costs depending on whether a job finished before (earliness), or after (tardiness) the specified due date. The objective is thereby, to find a sequence of jobs that minimizes a cost function that includes the cost of earliness and tardiness, due date assignment, makespan, and resource consumption. Moreover, future work will be directed to apply similar heuristic algorithms to multi-machine scheduling problems with controllable parameters including the job processing times, release dates, and delivery times.

References

Alidace B. and Ahmadian A. (1993), Two parallel machine sequencing problems involving controllable job processing times, European Journal of Operational Research, 70, 335-341.

Bazaraa M.S., Jarvis J.J. and Sherali H.D., (1990), Linear Programming and Network Flows, Wiley, N.Y. pp. 49-508.

Bean J. (1994), Genetics and random keys for sequencing and optimization, ORSA Journal on Computing, 6(2), 154-160,

Biskup D. and Cheng T.C.E. (1999), Single machine scheduling with controllable processing times and earliness, tardiness and completion time penalties, Engineering Optimization 31, 329–336.

Daniels R. L. and Sarin R.K. (1989), Single machine scheduling with controllable processing times and number of jobs tardy, Operations Research, 37 (6), 981-984.

Eberhart R.C. and Kennedy J. (1995), A new optimizer using particle swarm theory, in Proc. of the 6th Int. Symposium on Micro Machine and Human Science, 39-43.

Eiben A.E. and Smith J.E. (2003), Introduction to evolutionary computing, Springer, Berlin.

Fogel D.B. (1995), Evolutionary Computation, IEEE Press.

Foulds L.R. and Guochun T. (1999), Single machine scheduling with controllable processing times and compression costs (Part II: Heuristics for the general case), Applied Mathematics, 14B, 75-84.

Gen, M. and Cheng, R. (2000), Genetic Algorithms and Engineering Optimisation, Wiley-Interscience: New York.

Goldberg D.E. (1989), Genetic algorithms in search, optimization, and machine learning, Addison-Wesley, Reading, MA.

Guochun T. and Foulds L.R. (1998), Single machine scheduling with controllable processing times and compression costs (Part I: equal times and costs), Applied Mathematics, 13B, 417-426.

Holland J.H. (1975), Adaptation in natural and artificial systems: an introductory analysis with application to biology, control and artificial intelligence, University of Michigan Press, Ann Arbor, MI.

Hoogeveen H. (2005), Multicriteria Scheduling, European Journal of Operational Research, 167, 592-623.

Hoogeveen H. and Woeginger G.J. (2002), Some comments on sequencing with controllable processing times, Computing, 86, 181-192.

Kennedy J (1998), The behavior of particles, In: Porto V.W., Saravanan N, Waagen D. and Eiben A.E. (eds) Evolutionary Programming VII, Springer, 581–590.

Michalewicz Z. and Fogel D.B. (2000), How to solve it: Modern heuristics, Springer-Verlag, Berlin.

Murata, T., Ishibuchi, H. and Tanaka, H. (1996), Multi-objective genetic algorithms and its application to flowshop scheduling. Computers and Industrial Engineering, 30(4), 957–968.

Nearchou A.C. (2006), Meta-heuristics from nature for the loop layout design problem, Int. Journal of Production Economics, 101/2, 312-328.

Nowicki E. and Zdrzalka S. (1990), A survey of results for sequencing problems with controllable processing times, Discrete Applied Mathematics, 26, 271-287.

Panwalkar S. and Rajagopalan R. (1992), Single-machine sequencing with controllable processing times, European Journal of Operational Research, 59, 298-302.

Parsopoulos K.E. and Vrahatis M.N. (2002), Recent approaches to global optimization problems through particle swarm optimization, Natural Computing 1: 235-306.

Salman A., Ahmad I., Al-Madani S. (2003), Particle swarm optimization for task assignment problem, Microprocessors and Microsystems, 26, 363-371.

Shabtay D. and Steiner G. (2007), A survey of scheduling with controllable processing times, Discrete Applied Mathematics, 155, 1643-1666.

Storn R. and Price K. (1997), "Differential Evolution – A simple and efficient heuristic for global optimization over continues spaces", Journal Global Optimization, 11, 241-354.

Van Wassenhove L.N. and Baker K.R. (1982), A bicriterion approach to time/cost trade-offs in sequence, European Journal of Operational Research, 11, 48-54.

Vickson R.G. (1980a), Choosing the job sequence and processing times to minimize total processing plus flow cost on a single machine, Operations Research, 28 (5), 1155-1167.

Vickson R.G. (1980b), Two single machine sequencing problems involving controllable job processing times, AIIE Transactions, 12 (3), 258-262.

Wan G. Yen B.P.C., and Li C.L. (2001), Single machine scheduling to minimize total compression plus weighted flow cost is NP-hard, Information Processing Letters, 79, 273-280.

Zdrzalka S. (1991), Scheduling jobs on a single machine with release dates, delivery times and controllable processing times: worst-case analysis, Operations Research Letters, 10, 519–524.

List of Figures

List of Tables

Figure 1: The structure of the *i*-th (*i*=1, 2,…, Ns) individual of the *k*-th generation.

$$x_{ik} = \overbrace{x_{ik}^{1}, x_{ik}^{2}, \ldots, x_{ik}^{n}}^{\pi-part}, \underbrace{x_{ik}^{n+1}, x_{ik}^{n+2}, \ldots, x_{ik}^{2n}}_{\mathbf{y}-part}$$

Figure 2: Unique Pareto solutions obtained by the heuristics over the 1st test instance of TWJCTP test beds with, (a) 20-jobs, (b) 50-jobs, (c) 100-jobs, and (d) 200-jobs.

Figure 2(a): 20-jobs TWJCTP

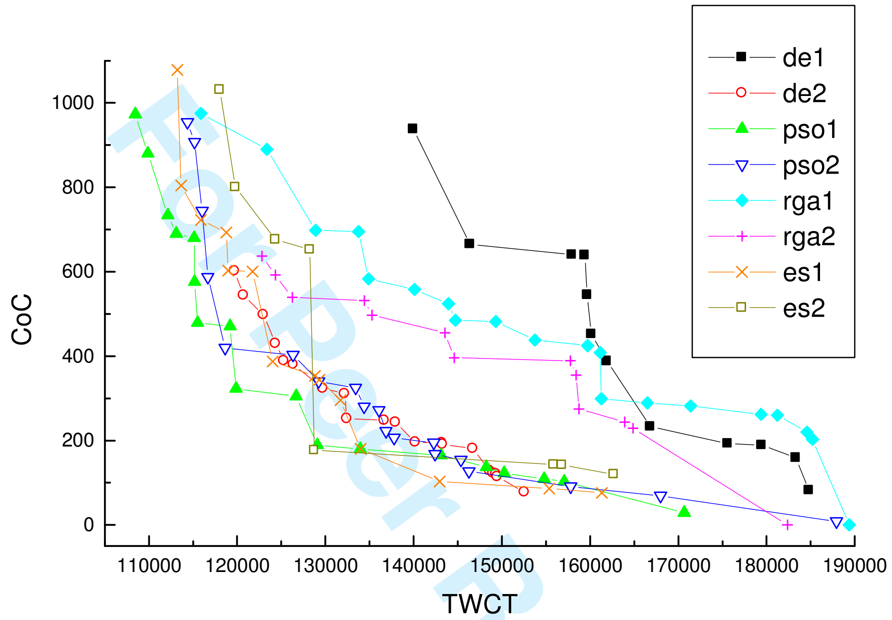
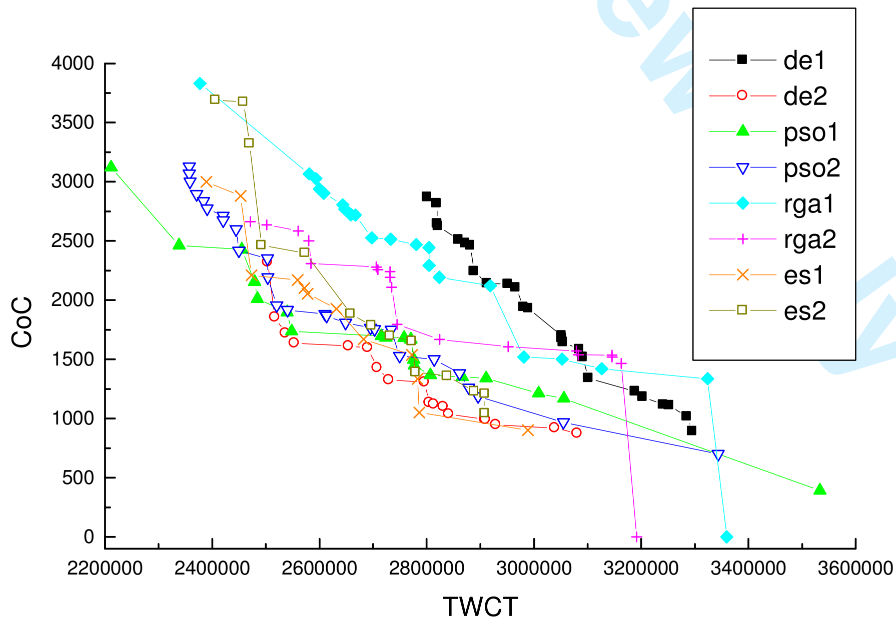


Figure 2(b): 50-jobs TWJCTP



A.C. Nearchou

Figure 2(c): 100-jobs TWJCTP

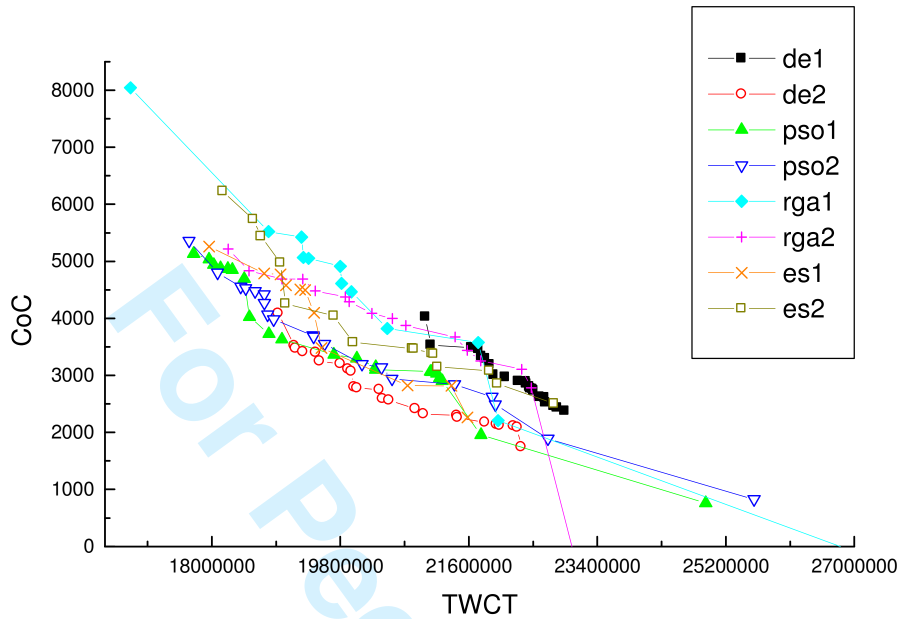


Figure 2(d): 200-jobs TWJCTP

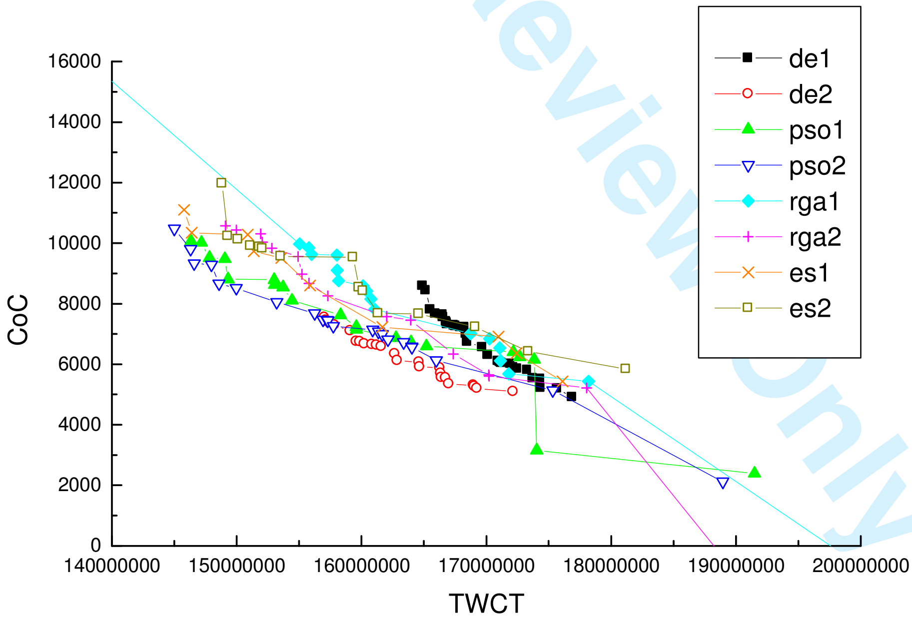


Figure 3: The optimum schedule for the 10-job 1/contr/$\sum C_i$ with jobs' characteristics shown in Table 9.

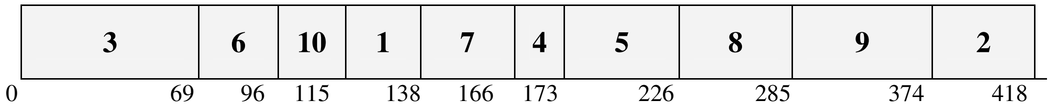


Table 1: Jobs' characteristics for a 5-job TWJCTP.

	J_i	J_2	J_3	J_4	J_5
np_i	20	6	15	10	12
u_i	4	8	5	7	7
w_i	5	15	13	13	6
ϕ_i	1	2.5	1.5	1	2

Table 2: A synopsis of the control schemes used to determine the correct settings for the heuristics' control parameters.

	Heuristic		
	DE	**PSO**	**rGA**
	Control parameters		
Control scheme	Crossover rate: C_R Scaling factor: F	Cognitive parameter: c_1 Social parameter: c_2 inertia weight factor: Iw	Crossover rate: C_R Mutation rate: M_R
Static	$C_R \in \{0.01, 0.1, 0.5, 0.7, 0.9\}$ $F \in \{0.5, 0.75, 0.95\}$	• $c_1 = c_2 \in \{0.5, 1.0, 2.0\}$ • $c_1=2$, $c_2=1.2$ $Iw = \Theta \times Iw \in [0.4, 1.2]$	$C_R \in \{0.6, 0.8\}$ $M_R \in \{0.01, 0.0333, 1/n, 0.1\}$
Dynamic	$C_R=0.01$ $F = \vartheta \times F \in [0.4, 1.0]$ Adapted using Eq. (9)	$c_1=2$, $c_2=1.2$ $Iw = \Theta \times Iw \in [0.4, 1.2]$ adapted by population diversity	$C_R=1.0$ $M_R = \vartheta \times M_R \in [0.1, 0.8]$ and adapted by population diversity

Table 3: Average running times in CPU seconds on a Dual Core 2.11 GHz PC

n	de1	de2	pso1	pso2	rga1	rga2	es1	es2
5	3.4	4.0	5.1	5.4	5.7	4.6	5.5	5.1
10	11.0	10.5	11.9	13.4	11.1	11.2	13.2	11.8
20	26.7	24.3	29.2	26.6	27.4	28.6	28.5	28.1
50	69.9	52.9	74.5	77.3	78.3	72.4	78.8	76.3
100	118.2	111.6	144.6	138.8	152.6	179.4	229.7	217.5
200	215.1	213.2	271.5	258.3	323.6	335.3	386.1	389.0

Table 4: Index P: Number of unique Pareto solutions generated by the heuristics after a single run over the examined test instances.

		DE		PSO		rGA		ES				DE		PSO		rGA		ES	
n		**de1**	**de2**	**pso1**	**pso2**	**rga1**	**rga2**	**es1**	**es2**	***n***		**de1**	**de2**	**pso1**	**pso2**	**rga1**	**rga2**	**es1**	**es2**
5	1	13	14	14	14	13	14	14	13	50	1	25	18	19	27	22	19	12	14
	2	13	15	15	16	17	17	16	15		2	28	25	19	24	20	26	18	15
	3	12	15	17	16	15	13	16	17		3	18	11	20	27	21	20	20	17
	4	14	17	18	17	18	19	19	16		4	23	20	21	21	22	15	14	14
	5	17	16	16	18	16	17	18	17		5	18	28	25	33	14	23	19	16
	6	13	11	13	13	12	13	13	13		6	21	20	23	20	18	19	13	11
	7	12	12	13	13	13	13	11	13		7	20	23	22	20	20	18	18	14
	8	7	7	9	9	8	7	19	9		8	18	20	17	25	25	21	14	21
	9	13	14	15	15	13	15	15	13		9	32	23	21	19	16	17	19	20
	10	12	13	13	11	13	12	13	10		10	24	24	30	24	28	23	14	25
		126	**134**	**143**	**142**	**138**	**140**	**154**	**136**			**227**	**213**	**217**	**238**	**206**	**202**	**161**	**168**
10	1	6	23	17	24	15	16	18	16	100	1	22	24	21	20	12	16	11	15
	2	11	20	18	22	22	11	17	26		2	30	30	27	16	8	18	15	13
	3	21	22	22	25	21	21	21	21		3	24	24	25	19	17	13	15	19
	4	12	13	12	14	13	13	13	14		4	33	26	23	17	12	23	15	8
	5	27	25	27	28	23	20	21	23		5	30	27	15	21	19	22	14	13
	6	16	26	24	24	22	17	18	25		6	35	14	23	23	30	26	10	10
	7	18	21	25	23	20	18	24	25		7	25	18	25	17	19	18	12	15
	8	15	20	27	23	19	14	27	24		8	35	18	18	26	23	22	14	8
	9	12	22	24	29	22	21	25	22		9	42	27	21	29	30	29	12	15
	10	18	21	27	22	14	16	17	18		10	40	22	18	19	19	15	25	13
		156	**213**	**223**	**234**	**191**	**167**	**201**	**214**			**316**	**230**	**214**	**207**	**189**	**202**	**143**	**129**
20	1	12	20	18	19	20	13	14	8	200	1	30	24	20	22	18	17	10	15
	2	17	20	10	18	17	15	19	19		2	27	46	20	19	21	14	12	12
	3	18	15	24	16	17	17	13	16		3	16	24	23	22	12	14	17	18
	4	17	24	22	25	20	19	15	19		4	34	32	17	15	13	13	20	14
	5	13	17	21	19	13	15	13	15		5	36	22	20	29	20	24	13	13
	6	20	18	20	16	19	19	21	14		6	33	17	16	20	21	18	18	14
	7	19	19	16	18	16	15	18	17		7	28	21	15	25	20	21	12	14
	8	18	23	24	20	19	20	22	12		8	26	20	24	24	18	16	20	17
	9	23	16	25	23	15	18	12	19		9	42	32	21	32	11	12	17	11
	10	19	24	28	16	21	9	13	16		10	38	16	23	29	15	18	12	11
		176	**196**	**208**	**190**	**177**	**159**	**160**	**156**			**310**	**253**	**197**	**238**	**169**	**167**	**151**	**139**

Table 5: Index P: Average number of unique Pareto solutions obtained after 20 runs of each heuristic over the examined test instances.

		DE		PSO		rGA		ES				DE		PSO		rGA		ES	
n		de1	de2	pso1	pso2	rga1	rga2	es1	es2	*n*		de1	de2	pso1	pso2	rga1	rga2	es1	es2
5	1	12.6	12.8	13.6	13.8	13.2	13.2	13.8	13.6	50	1	23.0	21.4	21.6	21.4	21.0	22.8	17.4	16.0
	2	14.6	15.2	16.4	17.0	16.6	16.4	17.4	17.0		2	31.0	23.2	21.8	21.8	19.2	22.4	16.0	16.0
	3	12.6	14.6	15.4	14.4	14.6	15.4	15.0	15.6		3	21.4	20.4	19.6	25.6	19.6	16.6	16.6	16.4
	4	14.2	17.4	17.4	16.4	18.6	18.2	17.6	14.8		4	22.4	20.6	20.0	22.0	20.0	20.2	13.6	15.4
	5	15.6	14.4	17.0	17.8	17.0	16.8	17.6	17.0		5	21.4	26.0	20.6	26.2	19.6	27.4	16.8	17.2
	6	11.4	12.2	12.4	12.8	12.2	13.0	12.8	12.8		6	21.8	22.4	20.6	21.6	16.4	22.0	15.0	14.4
	7	11.6	12.4	12.8	13.0	12.4	12.8	12.6	13.0		7	25.0	22.8	22.6	26.2	21.8	20.6	15.2	14.4
	8	7.2	6.6	9.0	9.0	8.8	8.4	10.8	8.6		8	24.2	19.8	18.0	21.8	19.6	19.4	15.6	17.2
	9	14.2	13.6	14.0	13.8	14.4	14.0	14.8	13.2		9	24.0	24.4	18.6	22.6	16.6	21.0	16.4	19.6
	10	11.8	11.8	12.0	11.8	12.2	11.2	12.6	10.8		10	25.4	26.4	24.2	22.8	24.6	23.2	16.2	17.6
10	1	11.8	20.2	20.6	22.2	19.4	18.0	20.2	18.8	100	1	25.4	23.6	19.4	23.0	16.0	17.8	15.4	14.6
	2	13.4	19.2	23.6	23.4	19.0	14.6	19.2	23.4		2	32.8	27.2	23.4	21.0	18.0	20.0	11.4	11.8
	3	17.8	23.4	24.4	24.4	18.2	18.8	22.4	24.2		3	29.2	22.0	22.2	23.0	17.6	14.4	14.4	13.6
	4	10.0	13.4	15.8	15.2	15.2	13.2	11.8	14.6		4	34.0	27.2	19.6	22.4	16.2	21.6	12.4	11.6
	5	25.6	26.4	26.6	25.2	20.8	22.0	25.6	23.8		5	31.4	27.2	17.4	20.4	19.6	19.0	11.4	14.2
	6	19.0	22.4	27.2	24.4	18.0	18.8	20.0	26.0		6	34.8	22.0	20.0	21.4	25.2	23.2	10.2	14.4
	7	18.0	20.0	25.4	24.8	18.0	17.6	20.6	21.2		7	29.2	24.4	23.4	23.8	19.0	19.6	12.2	12.0
	8	14.0	21.2	24.2	27.2	17.4	20.8	21.2	26.8		8	33.6	25.0	21.0	24.6	23.4	17.8	12.6	11.6
	9	21.0	24.0	24.4	27.2	21.4	20.8	24.0	24.2		9	40.4	29.2	21.6	23.8	23.4	22.4	13.4	13.0
	10	15.6	20.0	24.2	23.0	14.4	18.6	18.0	23.4		10	34.8	23.8	19.6	20.0	20.6	18.4	14.8	9.0
20	1	15.2	18.6	17.6	20.0	18.4	12.8	16.8	13.4	200	1	32.3	20.3	17.7	23.3	17.7	18.7	10.0	10.7
	2	12.0	17.0	17.8	22.2	17.4	16.8	13.8	16.8		2	32.0	33.3	19.7	19.3	17.7	18.3	11.3	12.0
	3	18.4	16.4	22.6	17.0	17.8	17.2	14.2	15.4		3	20.7	28.7	21.0	20.3	12.3	16.7	15.3	13.7
	4	19.6	23.2	22.4	25.8	23.0	16.8	18.8	18.8		4	32.3	28.7	19.3	18.7	13.7	18.0	16.7	13.3
	5	14.6	17.2	18.0	18.4	15.4	13.4	17.0	15.8		5	35.3	23.3	21.7	23.7	20.0	20.0	13.3	13.0
	6	21.2	19.8	19.8	18.4	18.0	12.2	17.4	15.4		6	35.3	21.0	19.3	23.0	20.3	16.3	15.7	11.7
	7	18.4	18.2	16.4	20.8	18.4	18.0	15.4	17.2		7	27.3	23.7	19.0	21.0	22.3	21.7	14.0	12.3
	8	15.2	20.2	20.6	22.2	23.2	18.6	19.0	19.2		8	27.0	30.0	25.0	22.0	15.7	18.3	12.7	12.7
	9	18.0	19.6	23.2	21.6	15.0	17.8	16.6	16.6		9	30.7	29.3	21.3	22.7	15.7	15.7	14.0	11.3
	10	16.4	18.6	22.6	18.2	21.4	13.2	16.2	15.8		10	34.0	20.7	20.0	23.0	16.7	19.0	13.3	12.0

Table 6: Index P*: Number of not dominated solutions between all Pareto solutions obtained by the algorithms. Average results over 20 runs of each algorithm for the examined test problems.

		DE		PSO		rGA		ES				DE		PSO		rGA		ES	
n		**de1**	**de2**	**pso1**	**pso2**	**rga1**	**rga2**	**es1**	**es2**	***n***		**de1**	**de2**	**pso1**	**pso2**	**rga1**	**rga2**	**es1**	**es2**
5	1	12.0	12.3	13.0	13.7	11.7	12.3	13.3	13.0	50	1	0.0	16.2	6.0	2.4	1.0	3.0	2.0	0.6
	2	13.0	14.0	14.0	15.3	14.7	15.7	15.7	15.0		2	0.0	18.2	3.6	2.4	1.0	2.8	0.6	0.2
	3	6.0	14.0	15.1	14.3	13.7	14.0	13.7	13.7		3	0.0	6.4	11.0	4.4	1.0	2.0	1.6	0.2
	4	14.3	17.0	17.0	16.0	17.7	18.0	17.0	13.3		4	0.2	15.8	1.0	6.8	1.2	3.2	0.6	1.4
	5	15.3	13.3	17.0	17.0	16.7	15.3	17.3	16.8		5	0.0	19.8	9.6	4.4	1.0	2.4	0.8	0.8
	6	12.0	11.7	12.0	12.0	11.7	12.7	12.0	12.0		6	0.0	19.4	8.0	3.6	1.0	2.2	0.6	1.0
	7	11.7	12.0	12.7	13.0	11.7	12.7	12.1	13.0		7	0.0	11.2	3.6	13.8	1.2	2.4	1.2	0.8
	8	7.0	6.0	8.8	9.0	7.7	7.7	9.0	8.0		8	0.0	13.6	5.2	3.8	1.0	2.8	0.6	0.8
	9	14.0	13.0	14.0	13.3	14.3	14.0	14.7	13.1		9	0.0	9.6	10.6	2.2	1.4	2.0	1.8	1.0
	10	11.0	11.7	11.7	11.3	11.7	10.7	12.3	10.6		10	0.0	19.2	6.0	5.2	1.0	2.0	0.6	0.6
10	1	2.0	18.4	11.0	14.8	1.4	5.8	3.0	6.6	100	1	0.2	18.4	4.4	2.8	1.4	1.8	0.4	0.8
	2	2.4	17.0	12.4	20.2	1.4	4.2	3.6	5.2		2	0.4	18.4	5.6	4.2	1.8	2.0	0.4	0.8
	3	1.6	17.2	14.2	18.6	1.4	5.6	2.8	4.2		3	0.0	13.0	3.2	7.0	1.6	2.0	0.2	0.8
	4	5.2	13.1	14.8	6.6	1.4	5.4	6.2	4.8		4	0.0	14.2	1.4	7.4	1.6	2.0	0.2	2.0
	5	2.4	21.8	11.0	15.2	11.8	2.4	5.4	7.6		5	0.0	20.4	3.6	5.6	1.6	2.0	0.6	0.8
	6	1.4	19.6	17.0	15.2	4.2	3.2	4.0	4.8		6	0.0	14.8	1.8	3.8	1.6	2.4	1.0	0.2
	7	0.8	17.4	14.4	15.6	3.4	3.8	2.2	7.8		7	0.2	20.2	4.0	3.0	1.4	2.0	0.6	2.2
	8	2.8	15.8	10.0	20.6	1.4	8.8	3.2	12.4		8	0.0	20.0	3.0	5.2	1.4	2.2	1.2	0.8
	9	0.8	19.2	22.8	21.2	2.0	3.8	4.0	10.0		9	0.0	24.2	3.4	3.4	1.2	2.4	0.2	0.6
	10	2.4	19.6	16.4	20.6	1.2	4.0	4.4	5.4		10	0.2	15.0	2.4	2.8	1.2	2.0	0.0	0.6
20	1	0.0	4.0	11.4	3.6	1.0	1.6	1.8	0.2	200	1	1.3	17.7	2.3	6.3	0.7	2.3	0.7	0.3
	2	0.0	5.0	9.4	4.8	1.0	3.4	1.2	1.0		2	2.3	23.3	9.7	5.3	0.7	2.0	0.7	0.0
	3	0.0	9.4	5.8	7.0	1.0	2.2	0.6	1.0		3	6.0	18.7	7.7	4.0	1.0	1.3	0.3	0.3
	4	0.0	12.4	10.4	6.2	1.2	3.0	0.6	0.4		4	6.3	20.7	8.0	3.0	1.0	1.3	0.0	0.0
	5	0.0	12.6	5.8	3.2	1.2	2.4	0.8	0.0		5	1.7	21.7	1.7	10.3	1.0	1.3	0.3	0.3
	6	0.0	10.2	6.6	6.8	1.2	2.0	0.4	0.0		6	6.3	15.7	4.0	8.7	1.0	1.3	0.0	0.3
	7	0.0	8.8	5.0	3.6	1.2	6.2	0.2	2.2		7	5.0	21.7	1.3	7.7	0.7	1.3	0.0	0.3
	8	0.0	4.8	9.8	10.4	1.0	3.0	0.4	1.0		8	5.7	21.7	6.3	7.0	1.0	1.3	0.3	0.0
	9	0.0	5.6	15.4	3.2	1.0	4.8	1.2	1.8		9	4.7	19.7	7.7	4.0	1.0	1.3	0.0	0.0
	10	0.0	8.6	9.4	3.2	1.0	4.2	1.4	0.6		10	4.0	15.0	4.3	10.3	1.3	1.3	0.3	0.0

Table 7: (a) A synopsis of the results shown in Tables 5 and 6. The values are averaged over the 10 instances of each problem's category. Numbers in brackets correspond to index P*, while those outside the brackets to index P. (b) Quality ratio P*/P.

n	de1		de2		pso1		pso2		rga1		rga2		es1		es2	
5	12.6	(11.6)	13.1	(12.5)	14.0	(13.5)	14.0	(13.4)	14.0	(13.2)	13.9	(13.3)	14.5	(13.7)	13.6	(12.9)
10	16.6	(2.2)	21.0	(17.9)	23.6	(14.4)	23.7	(16.9)	18.2	(3.0)	18.3	(4.7)	20.3	(3.9)	22.6	(6.9)
20	16.9	(0.0)	18.9	(8.1)	20.1	(8.9)	20.5	(5.2)	18.8	(1.1)	15.7	(3.3)	16.5	(0.9)	16.4	(0.8)
50	24.0	(0.0)	22.7	(14.9)	20.8	(6.5)	23.2	(4.9)	19.8	(1.1)	21.6	(2.5)	15.9	(1.0)	16.4	(0.7)
100	32.6	(0.1)	25.2	(17.9)	20.8	(3.3)	22.3	(4.5)	19.9	(1.5)	19.4	(2.1)	12.8	(0.5)	12.6	(1.0)
200	30.7	(4.3)	25.9	(19.6)	20.4	(5.3)	21.7	(6.7)	17.2	(0.9)	18.3	(1.5)	13.6	(0.3)	12.3	(0.2)

Table 7(a)

n	DE		PSO		rGA		ES	
	de1	de2	pso1	pso2	rga1	rga2	es1	es2
5	92.4	95.4	96.6	95.8	94.0	95.5	94.6	94.4
10	13.1	85.2	60.9	71.1	16.3	25.7	19.1	30.4
20	0.0	43.1	44.3	25.4	5.7	20.9	5.2	5.0
50	0.1	65.7	31.1	21.1	5.4	11.5	6.5	4.5
100	0.3	71.0	15.8	20.2	7.4	10.7	3.7	7.6
200	14.1	75.6	26.0	30.7	5.5	8.0	1.9	1.2
average	**20.0**	**72.7**	**45.8**	**44.1**	**22.4**	**28.7**	**21.8**	**23.9**

Table 7(b)

Table 8: Optimum solutions for the test instances included in the 5-job and 10-job ***1/contr/*** **$\sum C_i$. problems.**

Problem size	Test instances									
	1	2	3	4	5	6	7	8	9	10
5	606	381	578	511	888	477	619	821	361	511
10	1811	1794	1994	1430	1831	2135	2153	1473	2217	2038

Table 9: Jobs' characteristics for a 10-job 1/contr/$\sum C_i$ problem.

Jobs	np_i	u_i	ϕ_i
1	32	23	4
2	44	39	5
3	96	69	10
4	7	5	8
5	53	40	10
6	39	27	3
7	28	25	7
8	59	40	6
9	89	78	4
10	32	19	6