European Journal of Operational Research 215 (2011) 512–523

Contents lists available at ScienceDirect

European Journal of Operational Research

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

Discrete Optimization

A hybrid single and dual population search procedure for the job shop scheduling problem

Veronique Sels [a], Kjeld Craeymeersch [a], Mario Vanhoucke [a,b,c,*]

[a] Faculty of Economics and Business Administration, Ghent University, Tweekerkenstraat 2, 9000 Gent, Belgium
[b] Operations and Technology Management Centre, Vlerick Leuven Gent Management School, Reep 1, 9000 Gent, Belgium
[c] Department of Management Science and Innovation, University College London, Gower Street, London WC1E 6BT, United Kingdom

ARTICLE INFO

Article history:
Received 7 July 2010
Accepted 26 June 2011
Available online 3 July 2011

Keywords:
Job shop scheduling
Genetic algorithms
Scatter search

ABSTRACT

This paper presents a genetic algorithm and a scatter search procedure to solve the well-known job shop scheduling problem. In contrast to the single population search performed by the genetic algorithm, the scatter search algorithm splits the population of solutions in a diverse and high-quality set to exchange information between individuals in a controlled way. The extension from a single to a dual population, by taking problem specific characteristics into account, can be seen as a stimulator to add diversity in the search process. This has a positive influence on the important balance between intensification and diversification. Computational experiments verify the benefit of this diversity on the effectiveness of the meta-heuristic search process. Various algorithmic parameters from literature are embedded in both procedures and a detailed comparison is made. A set of standard instances is used to compare the different approaches and the best obtained results are benchmarked against heuristic solutions found in literature.

1. Introduction

The job shop scheduling problem (JSSP) is a well-known optimization problem that can be formulated as follows. A set of n jobs (index $j = 1,2,\ldots,n$) needs to be processed on a set of m different machines (index $i = 1,2,\ldots,m$). Each job consists of a number of operations that are associated with a particular machine. Therefore, the dimension of the problem is often denoted as $n \times m$. The job operations must be carried out in a predefined sequence and once an operation is started, no preemption is permitted. Every machine can process only one operation at a time and a job cannot be processed by two machines at the same time. An operation O_{ij}, associated with a particular job j and machine i, will have an estimated duration p_{ij}. The objective is to schedule each operation on the machines, taking the precedence constraints into account such that the total makespan (C_{max}) is minimized. The problem can be represented by $J||C_{max}$ using the classification scheme of Graham et al. (1979) and is known to be NP-hard (Garey et al., 1976).

A job shop problem instance can be visualized by a directed graph $G = (N,A,B)$, where N represents the set of nodes, A the set of conjunctive arcs and B the set of disjunctive arcs. The set of nodes contains one element for each job operation O_{ij}, a source node S connected to the first operation of each job and a sink node T linked with the last operation of each job. Conjunctive arcs are used to represent the routings of the different operations of the jobs and connect each pair of consecutive operations of the same job. Pairs of disjunctive arcs connect two operations, belonging to different jobs, which are to be processed on the same machine. The disjunctive arcs form a clique for each machine. A feasible solution corresponds to an acyclic subgraph that contains all conjunctive arcs and that contains exactly one disjunctive arc for each pair of disjunctive arcs between two nodes. An optimal solution corresponds to the feasible subgraph with the minimal makespan C_{max}. An example of a disjunctive graph for a JSSP with three machines and three jobs is given in Fig. 1.

The contribution of our paper is threefold. First, the paper builds on concepts borrowed from various sources in literature taking more than 60 papers on meta-heuristic optimization for the job shop scheduling problem into account, and compares the performance of these building blocks on a computational experiment. Second, a comparison between a single and dual population meta-heuristic is made, highlighting the similarities as well as the differences between the two approaches. We will show that each meta-heuristic has a different way of stimulating diversity in the search process, which seems to be a crucial element for finding high-quality solutions. Finally, we believe that our computational experiments have two main advantages. On the one hand, we show that our algorithm(s) can sometimes compete with many or most of the other procedures in literature, although we do not

* Corresponding author at: Faculty of Economics and Business Administration, Ghent University, Tweekerkenstraat 2, 9000 Gent, Belgium. Tel.: +32 93643569.
E-mail addresses: veronique.sels@ugent.be (V. Sels), mario.vanhoucke@ugent.be, mario.vanhoucke@vlerick.be (M. Vanhoucke).

0377-2217/$ - see front matter © 2011 Elsevier B.V. All rights reserved.
doi:10.1016/j.ejor.2011.06.031

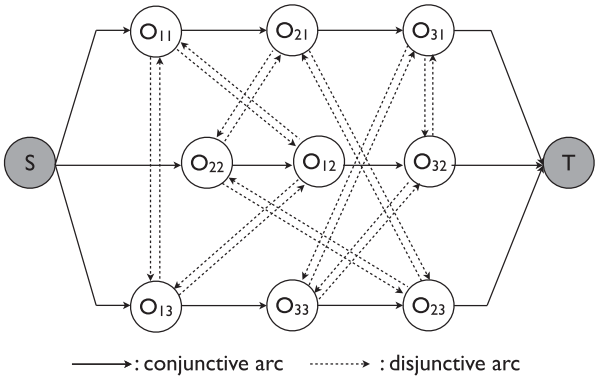

Fig. 1. Disjunctive graph representation of a 3 × 3 job shop.

outperform the best performing algorithms. On the other hand we provide detailed comparative results with 60 other algorithms by means of an extensive comparative table (online) for which the main results are summarized in an overview table published in this paper.

The outline of this paper can be summarized along the following lines. In Section 2, a brief literature overview of the JSSP is given, with special attention given to the papers related to the topic of the current research. In the same section, a detailed overview of the various elements to construct a population based meta-heuristic search procedure is given. Both a single and dual population search procedure is presented. Most elements of the procedures have been borrowed from meta-heuristic procedures available in literature, but will be extensively tested in the light of a single or dual population information exchange. Section 3 reports the results of a detailed computational experiment with respect to these different elements and compares them to state-of-the art results from literature. It will be shown that the balance between diversification and intensification of the search space plays a significant role in the performance of the algorithms. Computational tests will confirm that, thanks to the introduced notion of diversity, the dual population scatter search (SS) algorithm outperforms the single population genetic algorithm (GA). In Section 4, the main overall conclusions are drawn.

2. Solution procedure

2.1. Literature overview

The JSSP with a minimum makespan objective is widely acknowledged as one of the most difficult combinatorial optimization problems. Thanks to its practical importance, numerous exact as well as heuristic solution procedures have been developed in the past four decades. Enumerative approaches, such as the branch-and-bound algorithms of Carlier and Pinson (1989) and Brucker et al. (1994), have been successfully applied in solving small problem instances. However, due to the complexity of the JSSP, these exact procedures fail to obtain optimal schedules within acceptable time limits for instances with a dimension larger than 20 × 10. Therefore, the research focus was shifted towards approximation algorithms. These algorithms search for good rather than optimal schedules in reasonable computation times. A comprehensive survey of such scheduling techniques can be found in, for example, Blazewicz et al. (1996) and Jain and Meeran (1999). The most common heuristic approaches for makespan minimization include dispatching rules, shifting bottleneck procedures (Balas and Vazacopoulos, 1998; Pezzella and Merelli, 2000) and meta-heuristics such as genetic algorithms (Cheng et al., 1996b, 1999), tabu search (Nowicki and Smutnicki, 2005; Zhang et al., 2007, 2008a), simulated annealing (Kolonko, 1999), ant colony optimization (Huang and Liao, 2008), particle swarm optimization (Sha and Hsu, 2006) and artificial immune system methods (Coello et al., 2003; Chandrasekaran et al., 2006).

In the past decade, the genetic algorithms clearly received the most attention. This is justified by the numerous genetic algorithms developed for the JSSP. Most of these GAs focus on the performance of the different genetic operators and on the search for ideal parameter values. Sadegheih (2006), for example, examined the effect of the different parameter values on the performance of the GA. In Goh et al. (2003), an overview of selection schemes is given together with a new selection procedure based on sexual selection. Another example can be found in Watanabe et al. (2005), who proposed a genetic algorithm with search space adaptation and a modified crossover operator.

Additionally, a fair number of authors tried to increase the performance of the GA by incorporating other traditional heuristics in the algorithm. This hybridization can happen with simple techniques, such as simple scheduling rules (Zhou et al., 2001; Cheung and Zhou, 2001) or straightforward local search algorithms (Gonçalves et al., 2005; Zhang et al., 2008b; Gao et al., in press), but also with more advanced heuristics. Such advanced hybridizations are studied by Zhang and Gen (2005) (with a simulated annealing approach), Ombuki and Ventresca (2004) and Vilcaut and Billaut (2008) (with a tabu search technique) and Liu et al. (2006) or (Tsai et al., 2008) (with a Taguchi method).

Rather new approaches were discussed in Al-Hakim (2001), who suggested an analogue GA with a new encoding scheme, in Pérez et al. (2003), who presented niching GA methods to get multiple solutions, in Park et al. (2003), who developed the island-parallel GA to exploit parallel populations, and in Xu and Li (2007), who implemented an immune GA. We will not elaborate on these new techniques, but the algorithm of Park et al. (2003) deserves some special attention, as the authors made an attempt to divide the population in distinct subpopulations, among which solutions can migrate from one population to another. This idea of splitting the population is the essence of the scatter search technique, which will be discussed in this paper.

More recent research often focused on extensions of the JSSP. Examples are the inclusions of setup times (Cheung and Zhou, 2001; Manikas and Chang, 2009; Sun, 2009), the adaptation of the JSSP to the no-wait job shop (Pan and Huang, 2009) the incorporation of alternative objective functions (Essafi et al., 2008; Chiang and Fu, 2008) or the extension to the multi-objective JSSP (Esquivel et al., 2002; Qian et al., 2008). The extension with alternative job routings or the existence of parallel machines per work center is the subject of the flexible job shop scheduling problem (FJSP). Some noteworthy GAs developed for the FJSP are the algorithms of Zhao and Wu (2001), Gao et al. (2006, 2007, 2008), Moon et al. (2008), Pezzella et al. (2008), Rossi and Boschi (2009), Girish and Jawahar (2009)and De Giovanni and Pezzella (2010).

Finally, more real-life applications of the JSSP were discussed in Wu and Zhao (2000), Yu and Liang (2001), Vázquez and Whitley (2000), Chryssolouris and Subramaniam (2001), Caumond et al. (2008), Chan et al. (2009), Varela et al. (2003), Sels et al. (2010). Vázquez and Whitley (2000) and Chryssolouris and Subramaniam (2001) considered a dynamic JSSP and Caumond et al. (2008) studied the JSSP with time-lags. In Chan et al. (2009), a JSSP with lot-streaming was examined, while Varela et al. (2003) considered the JSSP with bottlenecks.

In all these GAs, the balance between diversification and intensification seems to be indispensable. An important element in this balance is the concept of diversity. A standard GA does not make use of a metric to measure diversity. In this paper, the importance of a distance measure will be shown, by comparing a genetic

algorithm with a scatter search technique. To our knowledge, the use of a scatter search technique in solving a JSSP problem is limited. The paper of Manikas and Chang (2008) discusses a somewhat similar scatter search technique for the JSSP with sequence-dependent setup times, while (Nasiri and Kianfar, 2010) propose a hybrid scatter search approach for the partial job shop problem. However, the focus of the authors slightly differs of the one used in this paper, as we put our main focus on the comparison of a single population and a dual population algorithm with the explicit use of a distance measure.

2.2. Solution approach

The well-known genetic algorithm is a search technique for solving optimization problems based on the principles of genetics and natural selection. The method, inspired on Charles Darwin's theory of evolution, was initially developed by John Holland in the 1970s (Holland, 1975). The more recent scatter search approach has been proposed by Glover (1998) as a population-based meta-heuristic in which solutions are intelligently combined to yield better solutions. The genetic algorithm and the scatter search procedure have roughly the same search structure and the main difference lies in the use and construction of the population set. While the genetic algorithm makes use of a single population, the scatter search procedure splits the population in a dual population, which has an effect on the selection strategy of solutions, the combinations of children and other solution elements. This dual population contains a high quality and a diverse subset, and is a very flexible approach that can be implemented in a variety of ways and degrees of sophistication. For an overview of the features of the scatter search technique, the reader is referred to Glover et al. (2000) and Marti et al. (2006).

The generic framework of the single population genetic algorithm and the dual population scatter search methodology can be described as follows:

Genetic Algorithm	**Scatter Search**
`Initial population`	`Diversification Generation Method`
`  Generation`	`While Stop Criterion not met`
`While Stop Criterion not met`	`  Subset Generation Method`
`  Parent Selection`	`  Solution Combination Method`
`  Crossover Operations`	`  Improvement Method`
`  Local Search`	`  Reference Set Update Method`
`  Update Population`	`Endwhile`
`Endwhile`	

The frameworks indicate that every component of the GA has it counterpart in the SS. They are often only labeled differently even though they perform a similar operation. In the next section, the scatter search terminology will be used for subtitling the different chapters that explain the way the various elements of both procedures have been modeled. Fig. 2 gives a simplified graphical overview of the various elements of the search procedures that acts as a guidance for the future sections.

2.3. Data representation

In literature, various representation schemes have been proposed for the job shop problem. An overview of the most commonly used representations is given in a tutorial survey written by Cheng et al. (1996a). An often used representation scheme is

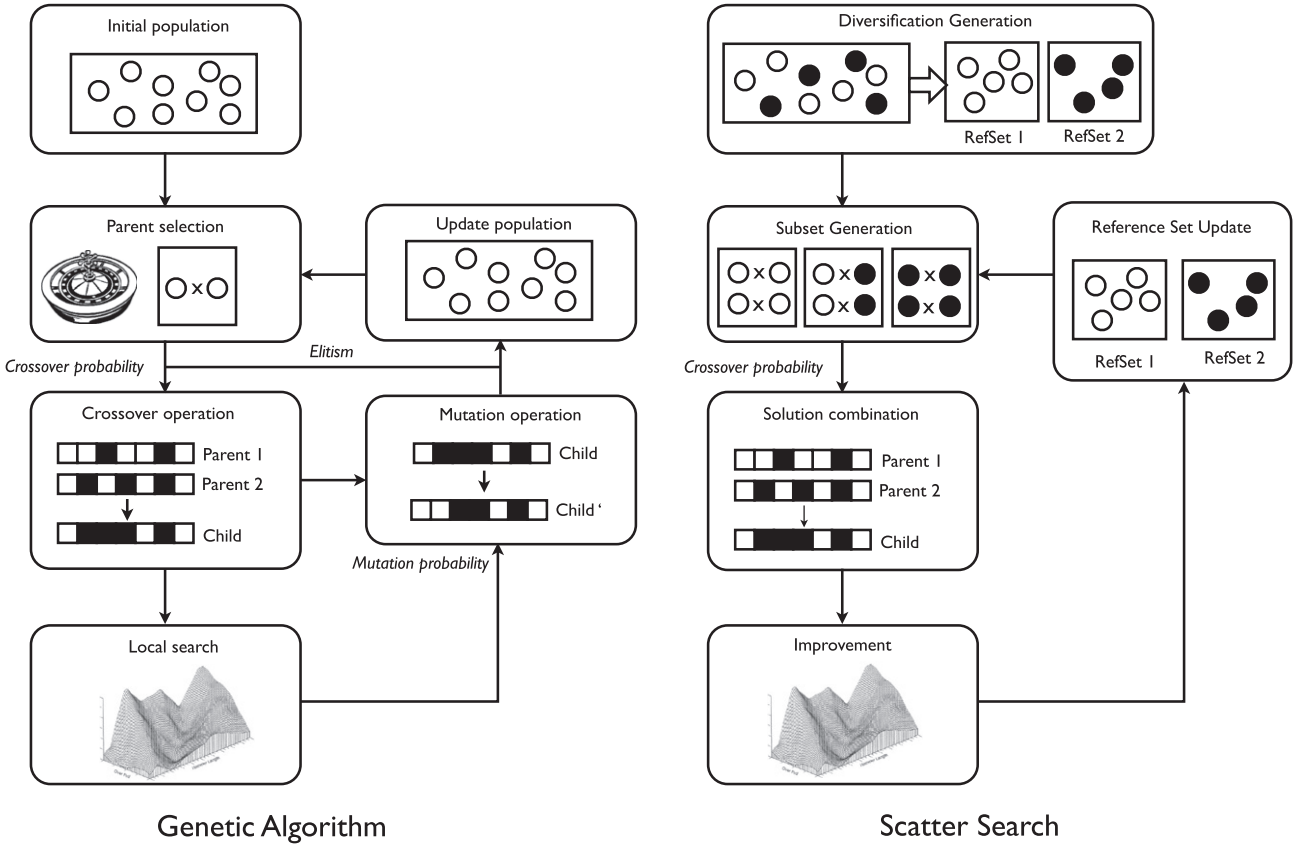

Fig. 2. A graphical overview of the elements for the single and dual population meta-heuristic search procedure.

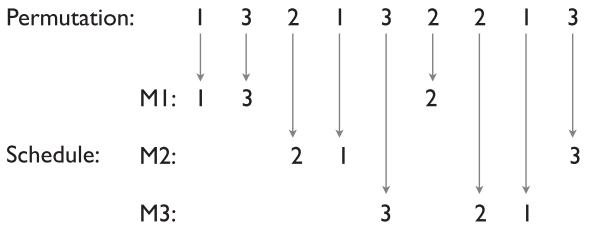

Fig. 3. Decoding of the permutation with repetition representation.

the operation-based representation. This representation encodes a schedule as a permutation of the $n \times m$ operations and each gene stands for exactly one operation. However, because of the precedence constraints between operations, not all possible permutations will result in feasible schedules. To guarantee the feasibility of a schedule, several strategies were described in literature, i.e. penalizing infeasible schedules (Zhou et al., 2001), using repair procedures (Nakano et al., 1991) or using adjusted operation-based representation schemes (Bierwirth, 1995). Such an adjusted representation scheme is the *permutation with repetition*. In this representation, all operations of the same job will be encoded using the corresponding job symbol. As such, each job symbol occurs exactly m times in the permutation. By scanning the permutation from left to right, the kth occurrence of a job number refers to the kth operation in the technological sequence of this job. Consequently, the permutation with repetition only contains information about the order in which the operations of jobs are scheduled (Park et al., 2003; Zhang et al., 2008b). In our algorithms, we also opted to use this representation scheme. The major advantage of the representation is that any permutation will result in a feasible schedule. Moreover, this type of representation will also allow us to use fairly standard genetic operators and local search techniques. A possible drawback of the method is that different permutations could result in the same schedule, having an adverse effect on the size of the search space. However, our computational experiments showed that by adjusting the population size accordingly, this drawback could be limited. An example of the permutation with repetition representation for the JSSP of Fig. 1 is given in Fig. 3. The decoding of the permutation results in the corresponding schedule below.

2.4. Diversification generation method

An important element in population-based meta-heuristics is the generation of an initial population. In order to construct the initial population, a large set of solution elements needs to be generated to ensure that a diverse region of the solution space will be covered. To generate a suitably diverse collection of solutions, the population set is often generated at random. However, it has been observed by many authors that introducing high-quality solutions in the population can help a meta-heuristic to find good solutions more quickly. These high-quality solutions can be obtained from a heuristic technique that introduces problem-specific information in the generation method. Nevertheless, it has been pointed out that this introduction will only increase the performance of the meta-heuristic on the condition that sufficient diversity is preserved.

In order to test the importance of a good and diverse initial solution, we have implemented three different methodologies to seed the initial population. The first strategy generates all solutions elements *randomly*. The second strategy generates all population elements with a constructive heuristic. We opted for the *Giffler-and-Thompson algorithm* that generates active schedules. As Giffler and Thompson (1960) observed that the optimal solution of the JSSP will always be an active schedule, this algorithm is assumed to provide high quality initial solutions. The third strategy tries to balance the strategies above and generates the initial population as a *combination* of the constructive heuristic of Giffler and Thompson (1960) and random generated solutions.

These strategies will be used to generate the single population of the GA and to seed the dual population of the SS. As already mentioned before, in the scatter search algorithm, the initial population with size equal to b will be divided in two sub-populations. The idea of multiple populations is not new and hence not restricted to the scatter search approach alone. As an example, (Park et al., 2003) embedded parallel populations and introduced a periodic exchange between these populations in their hybrid genetic algorithm in order to overcome premature convergence. However, the dual population of the scatter search procedure embeds the trade-off between solution quality and diversity even more cleverly as reflected in the reference sets $RefSet_1$ and $RefSet_2$. More precisely, only the b_1 best elements (in terms of solution quality) of the initialized population will enter the high-quality $RefSet_1$. The diverse $RefSet_2$ will contain the b_2 most diverse solutions relative to the solutions incorporated in $RefSet_1$. This means that solutions will only enter the second reference set if a diversity threshold with the solutions in $RefSet_1$ is exceeded. In order to determine this diversity threshold, a distance measure is calculated between every pair of solutions. We have implemented and compared two distance measures, the Hamming distance and a distance measure based on the starting times of the operations. The first one, the *Hamming distance*, is calculated by transforming the permutation with repetition representation to a binary string that expresses the absolute order of any two operations on the same machine. In this binary string, each gene determines if an operation O_{ij} precedes an operation O_{ik} ($O_{ij} < O_{ik}$) on a certain machine i (=1) or not (=0). Because of the dimension of the problem ($n \times m$), this results in a binary string with size equal to $m\frac{n^2-n}{2}$. This string allows us to calculate the normalized Hamming distance d between any two strings x and y as follows (Bierwirth et al., 1996):

$$d_{(x,y)} = \frac{1}{l\sum_i \mathbf{xor}(x_i, y_i)} \tag{1}$$

with l the length of the binary string, x_i (y_i) the ith element of the binary string x (y) and **xor** the exclusive or-function.

In order to illustrate this, an example is given in Fig. 4, in which two permutations of the JSSP of Fig. 1 are translated to a binary string with length l equal to 9. The normalized Hamming distance between the two strings is then equal to $d = 1/(9 \times 3) = 0.037$. By setting a certain minimal d-value as threshold value, it can be determined if a certain string, and thus the corresponding solution element, is diverse enough to enter the second reference set.

The second distance measure is based on the starting times of the operations. For this, no conversion to a binary string is needed

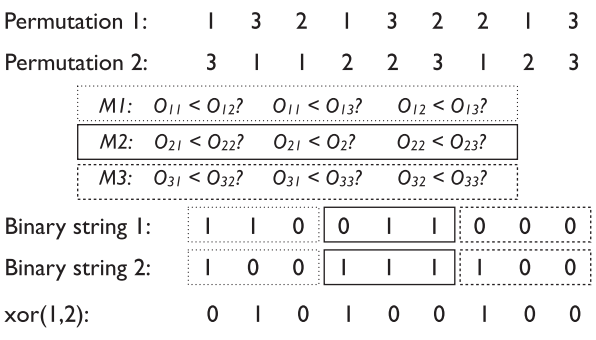

Fig. 4. Calculation of the normalized Hamming distance.

but only a translation of our permutations with repetition to (feasible) schedules. The distance between the operations' starting times in both schedules is then calculated according to formula (2).

$$d_{(x,y)} = \sum_i \left| S(x)_{Oij} - S(y)_{Oij} \right| \quad (2)$$

with $S(x)_{Oij}$ ($S(y)_{Oij}$) the starting time of operation O_{ij} according to permutation x (y).

In the computational experiment of Section 3.2, the various diversification methods will be tested on a set of benchmark test instances in order to select the best performing strategy. The three strategies discussed above will be referred to as RAND, HEUR and COMB, respectively, and will be tested on the single population (GA) and the dual high-quality/high-diversity population set (SS). The distance measures used in the scatter search technique are denoted as HAMD and STD. In the remainder of the paper, the abbreviations used in the computational experiments will be indicated in parentheses.

2.5. Subset generation method

In both solution approaches, the subset generation or selection procedure is a very important step. The main purpose is to select better individuals, while respecting the balance between intensification and diversification.

In a single population search meta-heuristic, such as the genetic algorithm, the selection is performed on population elements from the single population. These population elements are selected based on their fitness or objective value. One such selection procedure is the *roulette wheel selection* (RWS), in which each population member is assigned a chance of being selected proportional to its objective value. As the RWS has been extensively used in literature in the past decade, we also opted to include this selection technique in the genetic algorithm. Of course, as the risk exists that good, but suboptimal solutions will dominate the population too much, resulting in a converging population towards these solutions and a low diversity level, a scaling mechanism, based on the sigma scaling approach described in Goh et al. (2003), was tested as well (RWS_SC). This mechanism makes use of a function that rescales the fitness values of the population members to ensure a fair selection process. The rescaled fitness value of an individual solution of the population is calculated according to the following formula:

$$f_{scaled} = SC + (f_{org} - f_{avg})/2\sigma \quad (3)$$

with SC the scaling factor ($SC \in [2,5]$), f_{org} the original fitness value of the solution, f_{avg} the average fitness value of the population and σ the standard deviation of the fitness values in the population. The approach is assumed to have a positive effect on the genetic diversity and on the rate of convergence.

In order to further examine the balance between diversification and intensification in the single population, some alternative selection approaches were implemented as well. A *sexual selection* strategy (SEX) as proposed by Goh et al. (2003), separates the intensification and diversification process, by randomly dividing the population into two subsets of 'male' and 'female' solutions. The female population will explore the search space, while the male population members will be responsible for intensifying the search process. Implementation details are described in Goh et al. (2003). Moreover, this strategy can be easily extended with an *incest prevention* strategy (INC) proposed by Eshelman and Schaffer (1991) and discussed in Esquivel et al. (2002). The incest prevention will avoid that solutions that have the same ancestors will be selected at the same time to produce children. This should prevent premature convergences and should keep the diversity at a rather high level. Another strategy that can be used is the *elitist recombination* strategy (ELI) (Chryssolouris and Subramaniam, 2001). This strategy evaluates the solution quality of the children and their parents, and the best "family" member is directly injected in the population (in case the parents are better than their children, no replacement is done).

The dual population scatter search approach, however, operates on the reference sets instead of on the entire population, by combining pairs of reference solutions in a controlled, structured way. This structured way of subset generation makes the approaches used in the GA to prevent premature convergence redundant. In the subset generation method two (or more) elements of the reference set are chosen in a systematic way to produce points both inside and outside the convex regions spanned by the reference solutions (Glover et al., 2000). Choosing the two reference solutions from the same cluster (i.e., $RefSet_1 \times RefSet_1$ or $RefSet_2 \times RefSet_2$) stimulates intensification, while choosing them from different clusters ($RefSet_1 \times RefSet_2$) stimulates diversification. Because of the relative small sizes of the reference sets, it is possible to consider *all pairs* within and between the two sets (referred to as ALL) in reasonable computation times. However, as we use the number of evaluated schedules as our stopping criteria and we want to allow the SS to run over more generations, we have tested a restricted approach (RESTR) as well, where only a subset of all the possible pairs was considered. These pairs (within and between the reference sets) were selected based on the RWS procedure used in the GA.

2.6. Solution combination method

In general, the solution combination method or crossover method is considered to be the most important operator of both procedures. In a solution combination operation, the characteristics of two parent solutions are combined to form a new offspring solution. At first sight, the recombination of solutions seems to be substantially different in a genetic algorithm and a scatter search approach. The classical SS makes use of linear combinations of parents to generate offspring solutions, while the GA use genetic crossover operators. However, Glover (1994) showed that the traditional genetic crossovers are comprehended in these linear combinations. As a result, we opted to use these traditional genetic crossover operators in both the GA (within the population) and the SS (within and between the reference sets).

The crossover operation is executed with a certain probability, the *crossover rate*. With respect to the permutation with repetition representation scheme used, different crossover operators can be used. All these crossover operators either preserve the relative or the absolute order of the jobs or respect the absolute position of the jobs. The first one is the *generalized order crossover* (GOX) developed by Bierwirth (1995) to pass on the relative order of the solution elements. This relative order is, to some extent, also respected by the *generalized partially mapped crossover* (GPMX), which tends to preserve the absolute position of the solution elements as well. In both crossovers a substring is selected from one of the parent solutions and the elements of this substring, i.e. the job symbols with matching occurrences, are removed from the second parent solution. The offspring solution is then formed by placing the substring at the position were the first element of the substring occurred in the second parent solution (GOX) or in the first parent solution (GPMX) and by inserting the remaining job symbols of the second parent from the first position on. The absolute order of the jobs, expressing information about the precedence constraints between job operations, is respected by the *precedence preservative crossover* (PPX) of Bierwirth et al. (1996) and, to a certain extent, by the *cycle crossover* (CX) of Oliver et al. (1987). The PPX relies on a binary string that defines the order in

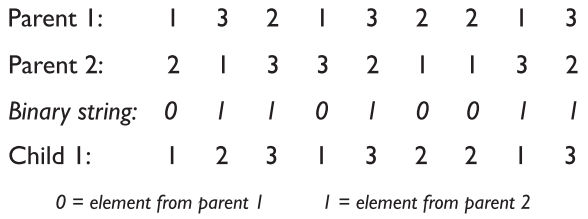

Fig. 5. Example of the PPX operator.

which the solution elements will be copied from the first (0) or second (1) parent solution. After a solution element (i.e. the *k*th occurring job symbol) is copied from one of the parents to the offspring solution, it is deleted from the other parent. This process is repeated until all elements of the parent solutions are removed or are copied to the offspring solution. An example of the PPX operator is shown in Fig. 5. The CX operator passes the solution elements through by means of cycles. These cycles are created by starting from a position in one of the parent solutions, looking for the corresponding job symbol at the same position in the second parent and finding the position of its counterpart (i.e. the *k*th occurring job symbol) in the first parent again. The cycle closes when the job symbol of the starting position is found. The process is repeated until no more cycles can be formed. In addition to these (fairly standard) crossovers, a crossover based on the Giffler-and-Thompson algorithm was tested as well. The *Giffler-and-Thompson crossover* (GT) was presented by Yamada and Nakano (1992) in order to ensure that only active schedules were build during the crossover operation. When generating an offspring solution, one operation (i.e. job symbol) is selected to be copied into the partial schedule and possible conflicts among operations/job symbols are resolved by specifying the priority of the operations according to the schedules of the parent solutions. For more information on this crossover we refer to Cheng et al. (1999).

2.7. Improvement method

In order to combine the global exploration capacities of the GA and the SS with a more local exploitation, both methods were hybridized with a local search technique. This local search algorithm tries to locally optimize the solutions during every iteration. An important feature in the local search algorithm is the neighborhood (NH) function. This function specifies the possible neighbors of a solution. As a solution is represented by a permutation of operations, the most common neighborhood functions used are the *insertion* and the *swap* NH (Michiels et al., 2007). An insertion NH function generates a neighbor by deleting an operation from the sequence and inserting it at another position in that sequence. In a *random insertion* (RI), the operation as well as its new position is chosen arbitrarily. In a swap NH, a neighboring solution is generated by interchanging two (or more) arbitrary or adjacent operations of the sequence. Interchanging two arbitrary jobs is called *randomized pairwise swap* (RPS), while swapping two adjacent jobs is referred to as *adjacent pairwise swap* (APS). If more than two operations are selected and the best permutation of these operations is chosen, this is referred to as a *scramble* (SCRAM) of operations. Combining both neighborhood functions is done in a *variable neighborhood search* (VNS). In a VNS approach, the local search systematically changes the neighborhood of the solutions. In our algorithm, we have implemented an adjusted version of the VNS of Sevkli and Aydin (2006), in which a randomized swap NH is alternated with a random insertion NH. The two neighborhood functions are illustrated by means of an example in Fig. 6. The different alternatives were implemented and tested in both

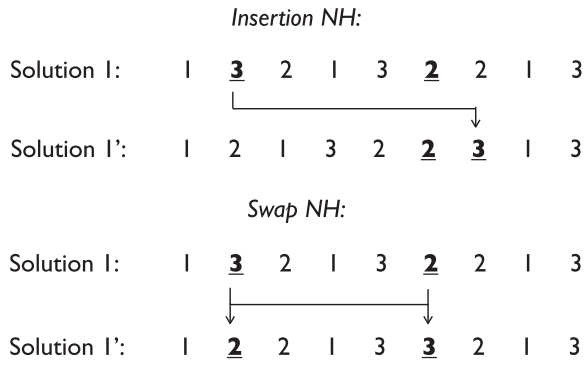

Fig. 6. Illustration of the insertion and swap NH functions.

solution procedures. The computational results will be discussed in Section 3.2.

2.8. Reference set update method

In both solution procedures, the (single or dual) population evolves over time with the entrance of new solutions replacing the older solutions, searching to improve the quality of the best known solution.

In the genetic algorithm, the children replace their parents if they have a better objective value. Otherwise, the best member of the "family" is admitted in the population by the elitist recombination strategy as already discussed in Section 2.5. Moreover, in order to ensure that the best population members will not get lost during the search process, an *elitism strategy* can be implemented as well. This principle states that the best solutions will have to be preserved in the next generation. It has been shown that this strategy has a positive influence on the overall solution quality of the GA Chryssolouris and Subramaniam, 2001. Of course, in order to respect the important balance between diversification and intensification in the single population of the GA, some diversity has to be ensured as well. Therefore, a (limited) mutation of some solutions in the population was performed by randomly inverting a portion of the permutation encoding. This inversion mutation (cf. Fig. 7) is known to have a good disruptive effect and was implemented according to two strategies based on the probability of occurrence, i.e. the *mutation rate*. If this probability of mutation is kept constant over the different generations, a *static* mutation strategy (SM) is used. On the contrary, if the mutation probability varies according to the convergence of the population, we make use of a *variable* mutation rate. A mutation with varying mutation probabilities (VM) dynamically updates the mutation probability when the convergence of population elements exceeds a predefined threshold. This threshold is based on a rather straightforward measure of diversity as defined by Cheung and Zhou (2001):

$$\text{pop\_diversity} = \frac{|f_{max} - f_{min}|}{f_{avg}} \tag{4}$$

with f_{max} and f_{min} the largest and the smallest fitness value in the population and f_{avg} the average fitness value of the population. Both strategies were tested and the results are presented in Section 3.2.

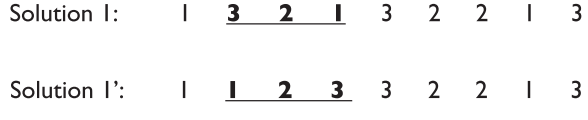

Fig. 7. Example of the inversion mutation operator.

The reference set update method in the scatter search approach happens in a slightly different manner. The method uses a dynamic update strategy that evaluates each possible reference set entrance instantly upon generation of a new solution element. A new solution may become a member of one of the reference sets either if the solution point has a better objective function value than the solution with the worst objective function value in $RefSet_1$ or if the solution is more diverse with respect to $RefSet_1$ than the least diverse solution point in $RefSet_2$. As such diversity as well as quality is implicitly preserved in the dual population, making the strategies used in the GA superfluous. If the reference set update method results in no new solutions entering the reference sets, both sets are re-initialized.

3. Computational results

In order to validate the performance of the genetic algorithm and the scatter search approach, the algorithms have been coded in C++ and all test runs have been carried out on a 2.0 GHz Intel Core 2 Duo Processor with 3 GB memory. Test instances (162 in total) are taken from literature and are available on http://people.brunel.ac.uk/mastjjb/jeb/orlib/ and http://mistic.heig-vd.ch/taillard/default.htm. These are the instances of Fisher and Thompson (1963) (ft06, ft10, ft20), Lawrence (1984) (la01-40), Adams et al. (1988) (abz5-9), Applegate and Cook (1991) (orb01-10), Yamada and Nakano (1992) (yn1-4), Storer et al. (1992) (swv01-20) and Taillard, 1993 (Ta01-80). Section 3.1 discusses the preliminary parameter fine-tuning of both solution procedures. In Section 3.2, a detailed overview of the computational experiments with respect to the various meta-heuristic elements described in the previous sections is given. This section is used to select the best performing elements of the single and dual solution search procedure in order to create two final hybrid search procedures. These procedures are then used to obtain our best solutions, which can be compared with the results found by competitive meta-heuristic search procedures from literature. Results of this test experiment will be discussed in Section 3.3.

3.1. Parameters fine-tuning

In order to refine our algorithms, decisions with respect to the parameter values have to be made. Examples of important parameters are the population size, the size of the reference sets, the crossover rate, the mutation rate and the local search stopping criterium. We executed some preliminary tests to fine-tune these parameters. Those tests were performed in cycles. In each cycle, a single parameter was chosen to be fine-tuned, while the other parameters were set to a certain value. All possible values for that parameter were tested and its best value was fixed before going to the next parameter. This was done for all parameters and the whole process was repeated until no more improvement was found. In Table 1, the different parameters and their final value are given for both solution procedures.

The results in the table show that relatively small population and reference sets sizes are sufficient for ensuring both quality as well as diversity in the single and/or dual population. Moreover, it will be shown that the usage of a dual population with 40 population elements in total will be equivalent with (or even better than) a single population with size 100. The crossover rates in both procedures are set at a high level (90%) to fully exploit the exploration abilities of the combination methods, whereas the mutation rate in the genetic algorithm is kept limited. The first mutation rate of 5% relates to the static mutation strategy. The variable mutation rates are set equal to 1%, 5% or 10% for the different classes of diversity. These classes were determined through preliminary testing. A hundred initial populations with random as well as heuristic solutions were generated and evaluated according to formula (4) in order to define three classes of diversity. The boundaries of each class were determined by the 15th and the 85th percentile of the population's diversity scores. At each generation, the diversity level of the current population is evaluated and classified in one of the three classes to determine the proper mutation rate. If the population has a *low* level of diversity, a mutation rate of 10% is used. If the population has a *normal* level of diversity, we apply the static 5% mutation rate. Finally, if the diversity is relatively *high*, a mutation rate of 1% is utilized. The local search algorithm embedded in both procedures is allowed to explore about 500 schedules during each generation of the genetic algorithm or the scatter search.

Table 1
Results of the parameter fine-tuning.

	Genetic algorithm	Scatter search
Population size (b)	100	–
RefSet size (b_1,b_2)	–	2×20
Crossover rate	90%	90%
Mutation rate (SM)	5%	–
Mutation rate (VM)	1–5–10%	–
LS stopping criterium	500	500

3.2. Procedure parameters

This section validates the performance of the various solution elements used in the genetic algorithm (one population) and the scatter search (two populations) as discussed in Section 2.2. Table 2 shows the results of the computational experiments. The first column refers to the solution elements described and abbreviated in Sections 2.4–2.7. For each of the implemented elements, the average makespan over a restricted set of data instances was calculated. This restricted set was based on the findings of Jain

Table 2
Computational experience on the various elements of the solution procedures (in %).

	Genetic algorithm	Scatter search
Diversification generation:		
RAND	3.65	1.10
HEUR	2.40	0.89
COMB	1.67	0.00
HAMD	–	1.14
STD	–	0.00
Subset generation:		
RWS	1.67	–
RWS_SC	7.69	–
SEX	3.24	–
INC	5.14	–
ELI	1.67	–
ALL	–	0.99
RESTR	–	0.00
Solution combination:		
GTX	9.07	2.40
GOX	3.41	1.49
GPMX	1.75	1.67
CX	2.09	0.51
PPX	1.67	0.00
Improvement:		
RPS	4.47	0.95
APS	15.45	14.76
SCRAM	4.77	3.07
RI	1.67	0.00
VNS	3.72	1.67
Reference set update:		
SM	1.76	–
VM	1.67	–

and Meeran (1999), who defined a number of computational challenging instances (i.e. ft10, la19, la21, la24-25, la27, la29, la36-40, abz7-9), which were also used in the computational study of Zhang et al. (2008b). These instances were expanded by some other difficult instances of the orb-, yn- and swv-sets (i.e. orb04, swv04, swv15, yn4). Computational experiments on these more challenging instances demonstrate the impact of the different elements on the solution quality more clearly, as experiments on the whole set of instances were biased by the good performance on the "easy" instances. The body of Table 2 shows the percentage deviation in average makespan from the best average makespan found, represented by the cells with a 0.00% deviation. This 0.00% deviation is equivalent with an average deviation of 3.01% from the best known solutions for these instances. In the next paragraphs, the results on the various solution procedure elements are discussed in more detail.

Diversification generation: The first part of Table 2 allows us to examine the effect of incorporating heuristic information in the initial population of the genetic algorithm. The results indicate that this incorporation is indeed advantageous. Nevertheless, the combination of pure random solutions with heuristic solutions obtained by the Giffler-and-Thompson algorithm results in a better performance than using the heuristic solutions alone. As already mentioned in Section 2.4, incorporating heuristic information is only beneficial if sufficient diversity is kept within the population. This diversity is necessary to guarantee a uniform sample of the search space. Therefore it is not surprising that the combination of random and heuristic solutions will be the best strategy to generate good as well as diverse solutions. It is important to notice that the magnitude of the effect is remarkably smaller for the SS procedure. Due to the dual population strategy, where a distance measure controls the quality and diversity level of the solutions in the two subsets in advance, a natural balance between diversification and intensification is already obtained. With respect to the distance measure used in the dual population approach, the computational experiments reveal that the rather simple distance measure based on the operations' starting times is capable of obtaining an adequate level of diversity. The method can compete with and even outperforms the computationally more expensive Hamming distance measure.

Subset generation: As discussed in the previous section, we have implemented the roulette wheel selection procedure in the genetic algorithm. A scaling mechanism was tested to prevent that some good, but suboptimal, solutions would dominate the search process too much. The test results show that this scaling mechanism has a downside effect on the performance of the GA, for every possible value of S. It seems to be advantageous to give the better solutions a higher chance of being selected. The implementation of the sexual selection strategy did not increase the performance of the genetic algorithm. A possible explanation can be found in the random division of the population in a male and female subpopulation. This randomness excludes some combinations between chromosomes, decreasing the likelihood of examining the whole search space. This drawback could be solved by dividing the population in a more structured way, what is exactly done in the SS algorithm. In addition, the incest prevention strategy failed to contribute to a better performance as well. The increase in diversity does not outweigh the loss in intensification ability. On the contrary, using the elitist recombination strategy clearly increases the performance of the genetic algorithm. This is not surprising, as it increases the quality of the population by including the best member of a family in the population. Of course, to keep the strict balance between quality and diversity and to prevent premature convergence, attention has to be paid to the diversity level in the population as well. We have found that this can be obtained by setting the crossover rate at a rather high level (cf. Section 3.1). In addition, the variable mutation rate that will be discussed in a subsequent paragraph, will play an important role as well. As already mentioned in Section 2.5, the above discussed strategies are not applicable in the scatter search procedure. As the diversification and the reference set update methods ensure that the highest quality and the most divers solutions will be members of the population, the balance between diversification and intensification is obtained in a controlled and structured manner. With respect to the subset generation method in the dual population approach, the computational results reveal that the restricted method outperforms the standard subset generation technique. Examining all possible pairs of solutions is computationally expensive compared to the limited number of solutions that will become member of the reference set. By restricting the examination of pairs, the SS will be able to run for more generations, which will increase the exploration capacity of the algorithm. Nevertheless, our computational experiments showed that considering subsets of three high-quality solutions, next to the pairs formed in RefSet1, increased the performance of the SS algorithm drastically. These extended subsets help intensifying the search process even further, especially for the larger size instances (cf. Section 3.3).

Solution combination: As already observed by Bierwirth et al. (1996), the preservation of precedence constraints is one of the most important aspects in combining solutions for the JSSP. Therefore, a crossover operator that respects the absolute order of the operations in the chromosome is desirable. Not surprisingly, the precedence preservative crossover (PPX) of Bierwirth et al. (1996) outperforms all other crossover operators in both solution procedures. As the cycle crossover (CX) operator of Oliver et al. (1987) respects the absolute order to a certain extent as well, it approaches the performance of the PPX the most. The performance of the Giffler-and-Thompson crossover is somewhat noteworthy. Although it reduces the search space by examining active schedules only, it is not able to increase the overall solution quality. The translation to active schedules often leads to a larger number of permutations resulting in the same schedule and hence, resulting in a lower exploration ability.

Improvement: The importance of a good neighborhood is demonstrated in the computational results of the improvement methods used. As can be seen in Table 2, the adjacent pairwise swap has the worst performance in both procedures, as its neighborhood function is rather restricted. In this local search algorithm, only pairs of jobs that are adjacent will be considered for swapping. Due to the representation scheme used, the effect of multiple chromosomes resulting in the same schedule will be enlarged by the usage of such a restricted NH function. Therefore, a more disruptive neighborhood is necessary. According to the computational results for both the genetic and the scatter search algorithm, the best performance is obtained by using the random insertion NH function, which even outperforms the variable neighborhood strategy that relies partly on the swap NH function.

Reference set update: As already shown in the previous paragraph concerning the subset generation method, the implementation of an elitism recombination strategy has a favorable effect on the solution quality of the genetic algorithm. Next to quality however, the diversity in the single population is maintained by means of an inversion mutation with a variable mutation rate. The results in Table 2 reveal that this dynamic strategy slightly increases the performance of the GA. This strategy increases the mutation rate as the diversification within the population decreases too much. It prevents early convergence of the single population and consequently, it has an important influence on the balance between diversification and intensification.

In Table 3, we summarize the implemented components of the genetic algorithm and the scatter search. By doing so, the parallelism and the differences between both methods is highlighted. In

Table 3
Overview of the implemented components of the GA and the SS.

Genetic algorithm		Scatter search	
Initial population	Single population with heuristic and random solutions	Dual population structure with starting time-based distance measure	Diversification generation
Selection	Roulette wheel selection of 2 parent solutions with elitism within population	Restricted selection of 2 or 3 parent solutions within and between the two subsets	Subset generation
Crossover	Precedence preservative crossover		Solution combination
Local search	Random insertion		Improvement
Update population	Preserving best solutions and variable mutation	Preserving best and most diverse solutions and re-initialization (if necessary)	Reference set update

Table 4
Comparison of solutions with a set of meta-heuristic search procedures from literature

Test set (# inst.)	Avg. Nr.	Hybrid genetic algorithm			Hybrid scatter search		
		Dev.BKS	Dev.Avg.	Dev.Best	Dev.BKS	Dev.Avg.	Dev.Best
abz (5)	26	6.76	3.12	6.66	2.60	−0.85	2.51
ft (3)	42	1.60	0.01	1.60	0.29	−1.28	0.29
la (40)	29	1.27	−0.54	1.27	0.45	−1.33	0.45
orb (10)	21	1.54	−1.21	1.54	0.11	−2.59	0.11
swv (20)	3	12.72	9.64	12.65	6.43	3.49	6.28
yn (4)	9	11.00	7.12	10.97	4.48	0.83	4.45
ta (80)	9	12.58	11.96	12.55	7.31	6.72	7.28

both procedures, the initial population is generated partly randomly and partly based on the Giffler-and-Thompson algorithm, the precedence preservative crossover is used as solution combination method and the random insertion NH function is implemented as improvement method. Moreover, the HGA is extended with the elitist recombination strategy and with a variable mutation rate. The HSS on its turn uses the operations' starting time distance measure and makes use of a restricted subset generation method with extended subsets (i.e. with $RefSet1 \times RefSet1 \times RefSet1$).

3.3. Best found solutions

In this section, we compare the performance of the best performing procedures of the previous section (further referred to as the hybrid genetic algorithm (HGA) and the hybrid scatter search (HSS)) with current state-of-the-art meta-heuristic procedures from literature. An extensive search in literature revealed that many (meta-) heuristic search procedures have been tested on only a small subset of the test instances available in literature, which makes it difficult to objectively validate the procedures in comparison with others from literature. In order to overcome this difficulty, we have created a detailed table where we summarized all available test results of meta-heuristic procedures found in literature from year 2000 on, as an overview of earlier procedures is given in Jain and Meeran (1999). We have calculated the average solution quality for each problem instance set as well as the best solution found by all procedures of which test results were available in literature. In doing so, we have a created a database of average and best found heuristic solutions on well-known benchmark sets that can be used for comparison purposes. This database with all computational details can be downloaded as an excel file from the website www.projectmanagement.ugent.be/machinescheduling.html. Next to all available heuristic solutions for the 162 individual test instances, the file contains the solutions obtained by our hybrid genetic algorithm (HGA) and scatter search procedure (HSS) as well. However, to summarize and present our results, the comparison of the solutions of the HGA and the HSS with the solutions obtained by the various (meta-) heuristic search procedures from literature is given in Table 4. The results are grouped per benchmark set. The following abbreviations are used to put the results into the right perspective:

- *Avg.Nr* average number of papers (after 2000) that published computational results on the corresponding test set
- *Dev.BKS* average deviation between the solution found by the HGA and HSS and the optimal or best known solution of the test instances in literature
- *Dev.Avg.* average deviation between the solution found by the HGA and HSS and the average solution quality from papers included in *Avg.Nr.*
- *Dev.Best* average deviation between the solution found by the HGA and HSS and the best found solution from papers included in *Avg.Nr.*

Notwithstanding the similar general framework of both procedures, the HSS clearly outperforms the HGA. An important reason can be found in the notion of "diversity". Although the GA was extended with several strategies to prevent premature convergence, it fails at preserving the required diversity level. In order to fully guarantee a certain diversity level, a distance measure should be used. As already mentioned in the previous sections, this distance measure is the core of the scatter search algorithm. Of course, next to diversity, attention should be paid to the quality of the solutions as well. This important balance is explicitly obtained by the dual population of the scatter search algorithm. Through the use of a good ($RefSet1$) as well as a diverse ($RefSet2$) reference set and the use of intelligent subset generation and combination methods, the balance between diversification and intensification is obtained in a controlled way. In a genetic algorithm, this balance is aimed as well, but its implicit higher randomness sometimes has a negative effect on the exploration and exploitation abilities.

If we look at the comparison with other (meta-)heuristic approaches, we can summarize the computational results of Table 4 along the following three remarks. Firstly, the table shows that the results of the GA procedure, and certainly for the SS procedure

outperform, on average, the results of similar procedures from literature for the ft-, la- and orb-data instances (indicated by the negative values in the *Dev.Avg.*-column). Moreover, it is shown that for these instances, the deviation for the best solutions found by the procedures (*Dev.Best*) and the best known solutions from literature (*Dev.BKS*) are relatively small, ranging from 0.11% to 0.45% for the SS procedure. It should be noted that the *Dev.Best* is measured as the deviation from the best found solutions found by all procedures mentioned in the *Avg.Nr.* column (e.g. 42 procedures for the ft-set), and hence, it does not report the solution quality relative to the quality found by one single solution procedure. The *Dev.BKS* solutions are obtained by various solution procedures, including various exact methods, often found after a much higher computation time. Consequently, the comparison with the average quality of many genetic algorithms and with the best known solutions available in literature show that the SS procedure performs relatively well and outperforms most other heuristic procedures on these data instances. However, the average deviations for the computational challenging instances discussed in Section 3.2 (i.e. ft10, la19, la21, la24-25, la27, la29 and la36-40) is equal to 1.30%. Secondly, the results show that the GA is not able to compete, on average, with the current GA procedures from literature for the abz- instances. The SS performs much better, with an average deviation of −0.85% compared to the average quality of other meta-heuristic procedures. Nevertheless, the gap with the best known solutions remains 2.60%. Thirdly, it can be seen that the worst results are obtained on the swv-, yn- and the Ta-benchmarks. This is in line with the findings of Jain and Meeran (1999), who stated that the difficulty level of instances is influenced by the ratio between the number of jobs and the number of machines and the 2sets-principle. More precisely, they observed that when the ratio of the numbers of jobs over the number of machines decline, the instances become harder. Special cases are the instances with an equal number of jobs and machines. In addition, the complexity increases if $m \geqslant 15$. This explains the difficulty level of the yn- and Ta-benchmark instances. The swv-instances are characterized by the 2sets-principle. According to this principle, the machines are divided in two sets and the routing consists of a random permutation of the machines in one set, followed by a random permutation of the machines in the second (Demirkol et al., 1997). However, the inferior results need to be nuanced. As said in the previous paragraphs, we have considered all heuristic solution procedures that were published after 2000. About 60 of these papers have published transparent computational results on the commonly used benchmark sets. However, only a small percentage of these papers have reported on the performance of their algorithm on the swv-, yn-, and Ta-set (respectively 4, 9 and 9 papers). Moreover, these papers used rather advanced heuristic techniques requiring high computational times. This makes an objective and fair comparison rather difficult for these data instances.

4. Conclusions

In this paper, a hybrid single and a dual population based procedure to solve the well-known job shop scheduling problem have been presented. The search procedures are based on the principles of genetic algorithms and scatter search procedures and are flavored with various elements taken from meta-heuristics available in literature.

A detailed computational experiment has been set up to determine the ideal combination of the different elements of both procedures. It has been shown that various elements of the meta-heuristics have similar characteristics, but often show a different behavior when a single versus a dual population is used. It has been shown in the paper that diversity is probably the most crucial component of both procedures. While the genetic algorithm often requires extra components to add search diversity (such as mutation), the extension from a single to a dual population by taking problem specific characteristics into account can be seen as a stimulator to add diversity in the search process. Consequently, the creation of a dual population set based on solution quality and diversity and the corresponding use of a distance measure to quantify this diversity has led to the development of a scatter search procedure producing the best results.

Both the genetic algorithm and the scatter search procedure have been used in a computational experiment where their solution quality has been compared with various other recent meta-heuristic search procedures from literature. Their average and best possible solution quality have been used as a criterion to validate the potential of the two newly created procedures. In addition, a detailed table is created with all available test results of the meta-heuristic procedures found in literature from year 2000 on. This table will be available online and can serve as a database tool for further benchmarking.

Moreover, the solutions found by the two new procedures have also been benchmarked with best known solutions from literature. The results show that the procedures can compete with and often outperform many current meta-heuristic procedures from literature for the ft-, la- and orb-data instances. For the abz-data, the GA performs on average worse, while the SS is able to contend with the existing procedures. The results found for swv-, yn- and Ta-benchmark instances could not compete with the average quality of current meta-heuristic procedures, but it should be noted that the number of procedures that have used these instances in their computational tests is much smaller compared to the other benchmark datasets.

Acknowledgements

We acknowledge the support given by the 'Bijzonder Onderzoeks Fonds (BOF), Vlaanderen, Belgium' under contract number BOF07/24j/034.

References

Adams, J., Balas, E., Zawack, D., 1988. The shifting bottleneck procedure for job shop scheduling. Management Science 34, 391–401.
Al-Hakim, L., 2001. An analogue genetic algorithm for solving job shop scheduling problems. International Journal of Production Research 39, 1537–1548.
Applegate, D., Cook, W., 1991. A computational study of the job-shop scheduling problem. ORSA Journal on Computing 3, 149–156.
Balas, E., Vazacopoulos, A., 1998. Guided local search with shifting bottleneck for job shop scheduling. Management Science 44, 262–274.
Bierwirth, C., 1995. A generalized permutation approach to job shop scheduling with genetic algorithms. OR Spektrum 17, 87–92.
Bierwirth, C., Mattfeld, D., Kopfer, H., 1996. On permutation representation for scheduling problems. Lecture Notes in Computer Science 1141, 310–318.
Blazewicz, J., Domschke, W., Pesch, E., 1996. The job shop scheduling problem: Conventional and new solution techniques. European Journal of Operational Research 93, 1–33.
Brucker, P., Jurisch, B., Sievers, B., 1994. A branch and bound algorithm for the job-shop scheduling problem. Discrete Applied Mathematics 49, 107–127.
Carlier, J., Pinson, E., 1989. An algorithm for solving the job-shop problem. Management Science 35, 164–176.
Caumond, A., Lacomme, P., Tchernev, N., 2008. A memetic algorithm for the job-shop with time-lags. Computers and Operations Research 35, 2331–2356.
Chandrasekaran, M., Asokan, P., Kumanan, S., Balamurugan, T., Nickolas, S., 2006. Solving job shop scheduling problems using artificial immune system. International Journal of Advanced Manufacturing Technology 31, 580–593.
Chan, F., Wong, T., Chan, L., 2009. The application of genetic algorithms to lot streaming in a job-shop scheduling problem. International Journal of Production Research 47, 3387–3412.
Cheng, B., Lee, J., Wu, J., 1996a. A constraint-based nurse rostering system using a redundant modeling approach. In: Proceedings of 8th International Conference on Tools with Artificial Intelligence, pp. 140–148.

Cheng, R., Gen, M., Tsujimura, Y., 1996b. A tutorial survey of job-shop scheduling problems using genetic algorithms. Part I: Representation. Computers and Industrial Engineering 30, 983–997.
Cheng, R., Gen, M., Tsujimura, Y., 1999. A tutorial survey of job-shop scheduling problems using genetic algorithms. Part II: Hybrid genetic search strategies. Computers and Industrial Engineering 36, 343–364.
Cheung, W., Zhou, H., 2001. Using genetic algorithms and heuristics for job shop scheduling with sequence-dependent setup times. Annals of Operations Research 107, 65–81.
Chiang, T., Fu, L., 2008. A rule-centric memetic algorithm to minimize the number of tardy jobs in the job shop. International Journal of Production Research 46, 6913–6931.
Chryssolouris, G., Subramaniam, V., 2001. Dynamic scheduling of manufacturing job shops using genetic algorithms. Journal of Intelligent Manufacturing 12, 281–293.
Coello, C., Rivera, D., Cortes, N., 2003. Use of an artificial immune system for job shop scheduling. Lecture Notes in Computer Science 2787, 1–10.
De Giovanni, L., Pezzella, F., 2010. An improved genetic algorithm for the distributed and flexible job-shop scheduling problem. European Journal of Operational Research 200, 395–408.
Demirkol, E., Mehta, S., Uzsoy, R., 1997. A computational study of shifting bottleneck procedures for shop scheduling problems. Journal of Heuristics 3, 111–137.
Eshelman, L., Schaffer, J., 1991. Preventing premature convergence in genetic algorithms by preventing incest. In: Fourth International Conference on Genetic Algorithms, pp. 115–122.
Esquivel, S., Ferrero, S., Gallard, R., Salto, C., Alfonso, H., Schütz, M., 2002. Enhanced evolutionary algorithms for single and multiobjective optimization in the job shop scheduling problem. Knowledge-Based Systems 15, 13–25.
Essafi, I., Mati, Y., Dauzère-Pérès, S., 2008. A genetic local search algorithm for minimizing total weighted tardiness in the job-shop scheduling problem. Computers and Operations Research 35, 2599–2616.
Fisher, H., Thompson, G., 1963. Industrial Scheduling, Chapter Probabilistic Learning Combinations of Local Job-Shop Scheduling Rules. Prentice-Hall, Inc., pp. 225–251.
Gao, J., Gen, M., Sun, L., 2006. Scheduling jobs and maintenances in flexible job shop with a hybrid genetic algorithm. Journal of Intelligent Manufacturing 17, 493–507.
Gao, J., Gen, M., Sun, L., Zhao, X., 2007. A hybrid of genetic algorithm and bottleneck shifting for multiobjective flexible job shop scheduling problems. Computers and Industrial Engineering 53, 149–162.
Gao, J., Sun, L., Gen, M., 2008. A hybrid genetic and variable neighborhood descent algorithm for flexible job shop scheduling problems. Computers and Operations Research 35, 2892–2907.
Gao, L., Zhang, G., Zhang, L., Li, X., in press. An efficient memetic algorithm for solving the job shop scheduling problem. Computers and Industrial Engineering.
Garey, M.R., Johnson, D.S., Seth, R., 1976. The complexity of flowshop and jobshop scheduling. Mathematics of Operations Research 1, 117–129.
Giffler, B., Thompson, G., 1960. Algorithms for solving production-scheduling problems. Operations Research 8, 487–503.
Girish, B., Jawahar, N., 2009. Scheduling job shop associated with multiple routings with genetic and ant colony heuristics. International Journal of Production Research 47, 3891–3917.
Glover, F., 1994. Genetic algorithms and scatter search: Unsuspected potentials. Statistics and Computing 4, 131–140.
Glover, F., 1998. A template for scatter search and path relinking. Lecture Notes in Computer Science 1363, 13–54.
Glover, F., Laguna, M., Marti, R., 2000. Fundamentals of scatter search and path relinking. Control and Cybernetics 29, 653–684.
Goh, K.S., Lim, A., Rodrigues, B., 2003. Sexual selection for genetic algorithms. Artificial Intelligence Review 19, 123–152.
Gonçalves, J., Mendes, J., Resende, M., 2005. A hybrid genetic algorithm for the job shop scheduling problem. European Journal of Operational Research 167, 77–95.
Graham, R., Lawler, E., Lenstra, J., Rinnooy Kan, A., 1979. Optimization and approximation in deterministic sequencing and scheduling: A survey. Annals of Discrete Mathematics 5, 287–326.
Holland, J., 1975. Adaptation in Natural and Artificial Systems. University of Michigan Press, Ann Arbor.
Huang, K.-L., Liao, C.-J., 2008. Ant colony optimization combined with taboo search for the job shop scheduling problem. Computers and Operations Research 35, 1030–1046.
Jain, A., Meeran, S., 1999. Deterministic job-shop scheduling: Past, present and future. European Journal of Operational Research 113, 390–434.
Kolonko, M., 1999. Some new results on simulated annealing applied to the job shop scheduling problem. European Journal of Operational Research 113, 123–136.
Lawrence, S., 1984. Resource constrained project scheduling: An experimental investigation of heuristic scheduling techniques. Graduate School of Industrial Administration, Carnegie-Mellon University, Pittsburgh, PA.
Liu, T.-K., Tsai, J.-T., Chou, J.-H., 2006. Improved genetic algorithm for the job-shop scheduling problems. International Journal of Advanced Manufacturing Technology 27, 1021–1029.
Manikas, A., Chang, Y.-L., 2008. A scatter search approach to sequence-dependent setup times job shop scheduling. International Journal of Production Research 47, 5217–5236.
Manikas, A., Chang, Y.-L., 2009. Multi-criteria sequence-dependent job shop scheduling using genetic algorithms. Computers and Industrial Engineering 56, 179–185.
Marti, R., Laguna, M., Glover, F., 2006. Principles of scatter search. European Journal of Operational Research 169, 359–372.
Michiels, W., Aarts, E., Korst, J. 2007. Theoretical Aspects of Local Search Series: Monographs in Theoretical Computer Science. An EATCS Series.
Moon, I., Lee, S., Bae, H., 2008. Genetic algorithms for job shop scheduling problems with alternative routings. International Journal of Production Research 46, 2695–2705.
Nakano, R., Yamada, T., 1991. Conventional genetic algorithm for job shop problems. In: Fourth International Conference on Genetic Algorithms, pp. 474–479.
Nasiri, M., Kianfar, F., 2010. A hybrid scatter search for the partial job shop scheduling problem. International Journal of Advanced Manufacturing Technology, 1–8.
Nowicki, E., Smutnicki, C., 2005. An advanced tabu search algorithm for the job shop problem. Journal of Scheduling 8, 145–159.
Oliver, I., Smith, D., Holland, J., 1987. A study of permutation crossover operators on the traveling salesman problem. In: Second International Conference on Genetic Algorithms, pp. 224–230.
Ombuki, B., Ventresca, M., 2004. Local search genetic algorithms for the job shop scheduling problem. Applied Intelligence 21, 99–109.
Pan, J.-H., Huang, H.-C., 2009. A hybrid genetic algorithm for no-wait job shop scheduling problems. Expert Systems with Applications 36, 5800–5806.
Park, B., Choi, H., Kim, H., 2003. A hybrid genetic algorithm for the job shop scheduling problems. Computers and Industrial Engineering 45, 597–613.
Pérez, E., Herrera, F., Hernández, C., 2003. Finding multiple solutions in job shop scheduling by niching genetic algorithms. Journal of Intelligent Manufacturing 14, 323–339.
Pezzella, F., Merelli, E., 2000. A tabu search method guided by shifting bottleneck for the job shop scheduling problem. European Journal of Operational Research 120, 297–310.
Pezzella, F., Morganti, G., Ciaschetti, G., 2008. A genetic algorithm for the flexible job-shop scheduling problem. Computers and Operations Research 35, 3202–3212.
Qian, B., Wang, L., Huang, D., Wang, X., 2008. Scheduling multi-objective job shops using a memetic algorithm based on differential evolution. International Journal of Advanced Manufacturing Technology 35, 1014–1027.
Rossi, A., Boschi, E., 2009. A hybrid heuristic to solve the parallel machines job-shop scheduling problem. Advances in Engineering Software 40, 118–127.
Sadegheih, A., 2006. Scheduling problem using genetic algorithm, simulated annealing and the effects of parameter values on GA performance. Applied Mathematical Modelling 30, 147–154.
Sels, V., Steen, F., Vanhoucke, M., 2010. A hybrid job shop procedure for a Belgian manufacturing company producing industrial wheels and castors in rubber. Technical Report, Ghent University.
Sevkli, M., Aydin, M., 2006. A variable neighbourhood search algorithm for job shop scheduling problems. Lecture Notes in Computer Science 3906, 261–271.
Sha, D., Hsu, C.-Y., 2006. A hybrid particle swarm optimization for job shop scheduling problem. Computers and Industrial Engineering 51, 791–808.
Storer, R., Wu, S., Vaccari, R., 1992. New search spaces for sequencing problems with application to job shop scheduling. Management Science 38, 1495–1509.
Sun, J., 2009. A genetic algorithm for a re-entrant job-shop scheduling problem with sequence-dependent setup times. Engineering Optimization 41, 505–520.
Taillard, E., 1993. Benchmarks for basic scheduling problems. European Journal of Operational Research 64, 278–285.
Tsai, J., Liu, T., Ho, W., Chou, J., 2008. An improved genetic algorithm for job-shop scheduling problems using Taguchi-based crossover. International Journal of Advanced Manufacturing Technology 38, 987–994.
Varela, R., Vela, C., Puente, J., Gomez, A., 2003. A knowledge-based evolutionary strategy for scheduling with bottlenecks. European Journal of Operational Research 145, 57–71.
Vázquez, M., Whitley, L., 2000. A comparison of genetic algorithms for the dynamic job shop scheduling problem. Lecture Notes in Computer Science 1917, 303–314.
Vilcaut, G., Billaut, J.-C., 2008. A tabu search and a genetic algorithm for solving a bicriteria general job shop scheduling problems. European Journal of Operational Research 190, 398–411.
Watanabe, M., Ida, K., Gen, M., 2005. A genetic algorithm with modified crossover operator and search area adaptation for the job-shop scheduling problem. Computers and Industrial Engineering 48, 743–752.
Wu, Z., Zhao, C., 2000. Genetic algorithm approach to job shop scheduling and its use in real-time cases. International Journal of Computer Integrated Manufacturing 13, 422–429.
Xu, X.-D., Li, C.-X., 2007. Research on immune genetic algorithm for solving the job-shop scheduling problem. International Journal of Advanced Manufacturing Technology 34, 783–789.
Yamada, T., Nakano, R., 1992. A genetic algorithm applicable to large-scale job-shop problems. In: Proceedings of the Second International Conference on Parallel Problem Solving from Nature, pp. 281–290.

Yu, H., Liang, W., 2001. Neural network and genetic algorithm-based hybrid approach to expanded job-shop scheduling. Computers and Industrial Engineering 39, 337–356.

Zhang, H., Gen, M., 2005. Multistage-based genetic algorithm for flexible job-shop scheduling problem. Complexity International 11, 223–232.

Zhang, C., Li, P., Guan, Z., Rao, Y., 2007. A tabu search algorithm with a new neighborhood structure for the job shop scheduling problem. Computers and Operations Research 34, 3229–3242.

Zhang, C., Li, P., Rao, Y., Guan, Z., 2008a. A very fast ts/sa algorithm for the job shop scheduling problem. Computers and Operations Research 35, 282–294.

Zhang, C., Rao, Y., Li, P., 2008b. An effective hybrid genetic algorithm for the job shop scheduling problem. International Journal of Advanced Manufacturing Technology 39, 965–974.

Zhao, C., Wu, Z., 2001. A genetic algorithm approach to the scheduling of fmss with multiple routes. The International Journal of Flexible Manufacturing Systems 13, 71–88.

Zhou, H., Feng, Y., Han, L., 2001. The hybrid heuristic genetic algorithm for job shop scheduling. Computers and Industrial Engineering 40, 191–200.