On Hybrid Evolutionary Algorithms for Scheduling Problem with Tardiness Criterion

Yu V Kovalenko
Institute of Scientific Information for Social Sciences RAS, 51/21, Nakhimov Avenue, Moscow, 117997, Russia
Dostoevsky Omsk State University, 55a, Mira Prospect, Omsk, 644077, Russia
E-mail: julia.kovalenko.ya@yandex.ru

Abstract. We present hybrid algorithms based on various metaheuristics for the scheduling problem with one machine and total weighted tardiness criterion. We research operators and algorithmic mechanisms provided in the literature. Advantages and disadvantages of the considered approaches are identified and analyzed. Some of the stated properties are common for scheduling problems on permutations.

1. Introduction
The classical scheduling problem $1|\text{r}_j, \text{d}_j, \text{w}_j|\sum_j \text{w}_j \text{T}_j$ is considered. Jobs $j\in\mathcal{J} = \{1,\ldots,n\}$ must be scheduled on one machine. Each job $j$ has duration $p_j$ (processing time), release date $r_j$, due date $d_j$ and weight $w_j$ (tardiness penalty). Jobs can not be started before their release dates. Each machine can process no more than one job at each time moment. Jobs preemptions are disallowed.

We denote by $C_j$ the completion time of job $j$. The tardiness of job $j$ is defined as $\max\{0; C_j - d_j\}$. The aim is to schedule all jobs from $\mathcal{J}$ minimizing the overall weighted tardiness $\sum_{j\in\mathcal{J}} w_j \text{T}_j$. In the scheduling theory this problem is called the one-machine total weighted tardiness problem (TWTP) [1], and arises in several practical settings, in particular in the chemical applications and production management systems. Moreover, modern algorithms for multiple machine scheduling problems often use as subroutine effective methods developed for the case of one machine. For example, jobs are assigned to machines by some rule, and an optimal or near optimal solution is calculated for each machine independently.

The natural encoding for feasible solutions is the permutation. Let $\pi = (\pi_1,\ldots,\pi_n)$ define a sequence of jobs. Then the completion times $C(\pi_i) := C(\pi_{i-1}) + p_{\pi_i}$, and tardiness $T(\pi_i) := \max\{0; C(\pi_i) - d_{\pi_i}\}$ for jobs in positions $i = 1,\ldots,n$ (suppose $C(\pi_0) := 0$). We denote the total weighted tardiness for permutation $\pi$ by $T(\pi) = \sum_{i=1}^n w_{\pi_i} T(\pi_i)$.

The considered problem is strongly NP-hard [2], so exact techniques (e. g. branch and bound algorithm, dynamic programming) are only applicable to small-sized problems in practice. NP-hard cases and approximation algorithms were proposed in [3, 4] for various dependencies between input data. These algorithms have polynomial, pseudopolynomial, and quasipolynomial nature. In [5, 6], the authors have developed fully polynomial time approximation schemes. Metaheuristics are appropriate for solving the problem with real input data. The most popular metaheuristics for the considered problem are local search, hybrid evolutionary algorithms,
particle swarm optimization, differential evolution, ant colony optimization and bee colony algorithms. Now we consider these metaheuristic approaches in details.

2. Hybrid evolutionary algorithms

The evolutionary algorithm (EA) is a search method based on ideas from genetics and natural selection [7]. An individual represents a tentative solution using some encoding scheme. The components of the individuals are called genes. Such reproduction operators as crossover and mutation are used for constructing new individuals. The crossover operator produces the offspring from parents by combining and exchanging their genes. The mutation adds random changes to an individual. Theoretical and experimental results show that in practice genetic operators should be developed considering their effect on each other.

The general structure of the hybrid evolutionary algorithm has the following form.

Algorithm 1

1: Construct the initial population of candidate solutions (as variant with local improvements).
2: Evaluate all candidate solutions.
3: Repeat until termination condition is satisfied.
3.1 Select a subset of solutions from the current population.
3.2 Construct offspring solutions using randomized and optimized reproduction operators.
3.3 If the perturbation/local improving criterion is satisfied then perform perturbation/improvement.
3.4 Select individuals for the next population.
4: Return the best found solution.

The well-known population management strategies are generational model, steady state replacement, tree-structured population, hypercube-based strategy, island model (multiple population), cellular model (spatial distribution within one population) [7].

Mutation is used in the following variants: insert (value of a gene is inserted in new position), swap (values of two genes are swapped), shift (value of a gene is moved to a number of positions) and inversion (order of a subsequence is reversed).

Performance of EAs depends on the recombination, where the offspring is built through combination of the components of parent solutions. Below we present the popular crossover operators for problems on permutations similar to the considered one.

Order based and position based crossovers: a part of the solution is copied from one parent, the rest positions are filled in the order corresponding to the second parent. Positions for coping are selected in different ways.

Partially mapped crossover: a segment between two crossover points is copied from one parent; the values for other positions are copied from the second parent, when it is possible. The rest empty positions of the offspring are filled by the pairwise exchanges between parents.

Cycle Crossover: cycle is the minimal subset of elements, for which the total set of assigned positions is the same in both parents; we construct offspring by selecting placement of elements in each cycle in accordance of their positions in the first or in the second parent, so value of each position is copied from one of the parents and solution feasibility is guaranteed.

Optimal recombination (ORP) consists in searching the best possible offspring as a result of a binary crossover operator, which satisfies respectful and gene transmitting properties [8].

The optimal recombination problem for \(1|r_j, d_j, w_j| \sum_j w_j T_j\) with position-based representation of solutions is stated as follows. The input consists of a problem instance and two parent permutations \(\pi_1\) and \(\pi_2\). The aim is to find a feasible solution \(\pi'\), for which:
(I) $\pi'_i = \pi_1^i$ or $\pi'_i = \pi_2^i$ for $i = 1, \ldots, n$;
(II) $\pi'$ has the minimum total weighted tardiness $T(\pi')$ among all feasible solutions satisfying property (I).

Previous research indicated that the optimal recombination may be effectively applied in the EAs for permutation problems [9–12]. Local improvements are often used on different stages of the evolutionary algorithms. An optimal recombination may be stated as a best local move in a neighborhood defined by condition (I) for two given parents.

One of the first genetic algorithm for the TWTP was presented in [13]. The algorithm uses various greedy heuristics in operators assigning jobs to positions. The authors applied uniform and single-point crossovers, and perturbation techniques. Different crossover operators were compared in [14]. The best results in the canonical genetic algorithm (basic scheme) for the TWTP were demonstrated by position and order based crossovers.

As far as we know no one provided and tested an optimized crossover in EAs for the considered problem. So, it is a good idea for the further research. Moreover, optimal recombination may be successfully used not only in EAs, but for example, for recombination of solutions obtained in local search algorithms during the search and at post-processing stage. Such algorithms belong to the class of methods from memetic computing [15].

2.1. Local Search
A local search algorithm (LS) starts from an initial solution. It moves iteratively from one solution to a better neighboring solution and terminates at a local optimum. Important role plays initial solutions, which are usually generated by some greed constructive heuristics. Classic neighborhoods for the permutation problems are insert, swap, shift, inversion, Kernigan-Lin neighborhood.

Popular variations of the local search are the following.

*Tabu search:* moves or solution elements that lead to previously obtained local optimums are marked as tabu and not used by a certain number of iterations, the length of the tabu list and tabu tenure for the tabu elements are the controlled parameters in the algorithm.

*Variable neighborhood search and variable neighborhood descent:* several neighborhoods are used and systematically changed in the searching process, shaking and local search stages are usually involved; various schemes of the neighborhoods changing and variable depth searches are used.

*Path-relinking:* the algorithm explores a path from the current solution and the best solution found so far, where at each iteration a move from the current to the best solution is performed, and we improve the obtained tentative solution by some greedy algorithm.

*Dynasearch algorithm:* the local search method, where neighborhoods of exponential size are explored by the dynamic programming in polynomial time, the feature consists in the fact that several independent moves are applied at each iteration; for example, the swap neighborhood is used in [16] for the considered problem.

*Simulated annealing:* the iterated algorithm, where a new tentative solution is built by perturbing the previous one, and the probability of accepting new solution depends in exponential form on such tuning parameter as temperature, and the difference in objective value between the new solution and the previous one.

*Iterated local search:* the algorithm produces the starting solution for the next iteration by perturbing the current local optimum; here we use the idea that a solution located near the local optimum is better than the incumbent solution, but close in distance.

*Population-based local search:* approach is similar to the previous one, but uses a population of solutions in local improvement strategies of the search process.
To prevent a local search algorithm to get stuck in the regions of poor local optima, the restart rules (disturbing, multi-start), diversification and intensification (search in width and depth) are applied.

Good experimental results for $1|r_j, d_j, w_j|\sum_j w_j T_j$ are demonstrated by single-solution and population-based local search methods [17–23]. The reason is the “good” properties of the objective function, which significantly reduce the search area of solutions in the neighborhoods (see, e.g., [17]). L. Chaabane [24] has proposed for TWTP a genetic algorithm applying simulated annealing at local improving stage, one point crossover and insert mutation. The hybrid method [16] combining evolutionary algorithm with position-based crossover and dynasearch procedure demonstrates the most effective results at this time in terms of both objective value and running time.

2.2. Other evolutionary algorithm variants

Particle swarm optimization and differential evolution were initially proposed for the continuous optimization [7].

**Particle swarm optimization (PSO)** [25, 26] is a population-based algorithm, where analog of individual is particle with location $x$ and velocity $v$, and evolution is realized by means of mutation:

$$x' := x + v'$$

$$v' = C_v v + C_{yv}(y - x) + C_{xz}(z - x),$$

where $C_v$, $C_{yv}$, $C_{xz}$ are weights (interia, personal influence, social influence), $y$ in the personal best value, $z$ is the global best value. During the search process each particle $(x, v)$ is replaced by new particle $(x', v')$.

**Differential evolution (DE)** [27] uses differential mutation and crossover. For each individual $u$ we select an individual $x$, and mutate the last one as

$$x' = x + S(y - z),$$

where $S$ is the scaling value, $y$ and $z$ are some individuals (may be randomly chosen from the population). Then a crossover operator combines individuals $u$ and $x'$ and obtains a new vector $u'$ (for example, uniform crossover). For constructed pair $u$ and $u'$ we select the best one for the next generation.

In DEs [25–27], for the considered problem a solution is encoded by a real vector of length $n$, and the corresponding permutation is constructed by means of the smallest position value (SPV) rule, where jobs are sorted in increasing order of vector-component values. Various local search methods (for example, insert and swap) and their combinations are used at iterations to improve results.

In DEs [25–27], for the considered problem a solution is encoded by a real vector of length $n$, and the corresponding permutation is constructed by means of the smallest position value (SPV) rule, where jobs are sorted in increasing order of vector-component values. Various local search methods (for example, insert and swap) and their combinations are used at iterations to improve results.

The simple encoding scheme for solutions based on SPV rule is integrated in PSO and DE from papers [25–27]. As variant, more complex structure of solution representation and more effective local search technics may be investigated to improve the results of the mentioned algorithms.

3. Other population-based heuristics

**Ant colony optimization (ACO)** [28] is inspired by the behavior of real ant colonies. In ACO algorithms a colony of ants iteratively constructs solutions be means of pheromone trails and heuristic information. The ants communicate by modifying the pheromone trails during the execution of the algorithm. Note that the ants improve their solutions by local search in many state-of-the-art ACO-based methods.
The main steps are constructing solutions, local improving, updating trails. In [29–31], competitive ACO algorithms have been developed for the TWTP. The pheromone trail $\tau_{ij}$ associated to the assignment of job $j$ to position $i$ is globally updated as $\tau_{ij} = (1 - \rho)\tau_{ij} + \rho \Delta \tau_{ij}$, when the current best solution is improved and contains job $j$ at position $i$. Initially all $\tau_{ij}$ are identical. Moreover, local updating rules are used (see [30, 31]).

The following heuristic values are used for assigning job $j$ to position $i$:

- $\eta_{ij} = \frac{1}{d_j}$,
- $\eta_{ij} = \frac{1}{\max(d_j, C + p_j)}$, where $C$ is the total completion time of the already scheduled jobs,
- $\eta_{ij} = \frac{p_{wj}}{a_j} \exp\left(\max\{d_j - C - p_j, 0\}/n p_{\text{aver}}\right)$, where $p_{\text{aver}}$ is the average processing time of jobs.

The permutation of jobs is constructed as follows: with probability $q$ the job $j$ that gives the maximum value of $\tau_{ij} \cdot \eta_{ij}^\beta$ is placed at the current position $i$ ($\beta$ is a tunable parameter), and with probability $(1 - q)$, the job $j$ is selected randomly with probability

$$p_{ij} = \frac{\tau_{ij} \cdot \eta_{ij}^\beta}{\sum_k \tau_{ik} \cdot \eta_{ik}}.$$

**Bee colony optimization (BCO)** [32] is also a population-based nature-inspired method based on the nectar searching behaviors of honeybees and their activities in the hive. The following idea is used. Scout and onlooker bees are considered. Scout bees search for nectar sources and carry nectar from the founded sources. Onlooker bees watch the behavior of the scout bees returning to the hive, and head towards a high-quality source and become employed bees.

In the artificial bee colony algorithms, nectar sources correspond to tentative solutions (the initial population is usually constructed at random or by some constructive heuristics). Each scout bee search for new solutions (sources) by some local greedy procedures. Onlooker bees chose sources in accordance with the probabilities proportional to objective values of solutions obtained by scout bees. The process is repeated until a termination condition is met. Moreover, we iteratively try to improve the best solution, and update the worst solutions using new random or greedy solutions.

In the BCO algorithm from [33] for the considered problem the solutions are represented by real-valued vectors, and such vectors are decoded into permutations using the (SPV) rule. The operators from continuous optimization and local search improving methods are involved in the search mechanism.

It is interesting to investigate BCO, where solutions are encoded by permutations in explicit form, and permutation-based operators and various selection procedures are incorporated.

4. Conclusion

In this article we have provided a survey on evolutionary algorithms and other population-based methods for the total weighted tardiness problem on one machine. All considered approaches have hybrid nature and incorporate various local search technics. A literature review has been presented, and new directions are indicated for the further research. Note that now one of the intensively studied direction is hyper-heuristics with adaptive turning of parameters, and it is interesting to investigate such approach for the considered problem using the presented here algorithms as low-level heuristics.

**Acknowledgments.** The research was supported by the Russian Science Foundation, project number 17-18-01536.
References

[1] Blazewicz J, Ecker K H, Pesch E, Schmidt G and Weglarz J 2007 Handbook on Scheduling: From Theory to Applications (Springer-Verlag Berlin Heidelberg)

[2] Lenstra J, Rinnooy Kan A and Brucker P 1977 Annals of Discrete Mathematics 1 343–362

[3] Cheng T, Ng C, Yuan J and Liu Z 2005 European Journal of Operational Research 165 423–443

[4] Kolliopoulos S and Steiner G 2006 Theoretical Computer Science 355 261–273

[5] Kacem I 2010 Discrete Applied Mathematics 158 1035–1040

[6] Kellerer H and Strusevich V 2006 Theoretical Computer Science 369 230–238

[7] Eiben A and Smith J 2015 Introduction to evolutionary computing (Springer-Verlag Berlin Heidelberg)

[8] Radcliffe N 1994 Ann. Math. Artif. Intell. 10(4) 339–384

[9] Tinos R, Whitley D and Ochoa G 2014 The 2014 annual conference on genetic and evolutionary computation (ACM, New York) pp 501–508

[10] Cook W and Seymour P 2003 INFORMS J. Comput. 15(2) 233–248

[11] Yagiura M and Ibaraki T 1996 Eur. J. Oper. Res. 92 387–401

[12] Eremin A and Kovalenko Y 2020 Memetic Comp. 12 23–36

[13] Kellegoz T, Toklu B and Wilson J 2008 Applied Mathematics and Computation 199 590–598

[14] Nerl F and Cotta C 2012 Swarm. Evol. Comput. 2 1–14

[15] Ding J, Lu Z, Cheng T and Xu L 2017 Computers and Industrial Engineering 108 70–80

[16] Geiger M J 2010 European Journal of Operational Research 207 1235–1243

[17] Geiger M J 2010 European Journal of Operational Research 207 1235–1243

[18] Yahyaoui H, Krichen S, Derbel B and Talbi E G 2013 2013 5th International Conference on Modeling, Simulation and Applied Optimization (ICMSAO), IEEE vol 13641151

[19] Wang X and Tang L 2009 Computers and Operations Research 36 2105–2110

[20] Fu Q and Chung T P 2016 2016 IEEE International Conference on Industrial Engineering and Engineering Management (IEEM), IEEE vol 16560230

[21] Congram R, Potts C and van de Velde S 2002 INFORMS Journal on Computing 14(1) 52–67

[22] Chunga T P, Pua Q, Liaob C J and Liub Y T 2017 Engineering optimization 49(7) 1133–1147

[23] Bilge U, Kurtulan M and Kirac F 2007 European Journal of Operational Research 176 1423–1435

[24] Chaabane L 2017 Journal of Information and Computing Science 12(4) 270–279

[25] Tasgetiren M, Liang Y C, Sevkli M and Gencyilmaz G 2006 International Journal of Production Research 44(22) 4737–4754

[26] Tasgetiren M, Sevkli M, Liang Y C and Gencyilmaz G 2004 Proceedings of the 2004 Congress on Evolutionary Computation, IEEE vol 8229093

[27] Nearchoiu A 2012 Cybernetics and Systems 43 651–668

[28] Dorigo M and Blum C 2005 Theor. Comput. Sci. 344 243–278

[29] den Besten M, Stutzle T and Dorigo M 2000 Parallel Problem Solving from Nature (PPSN-2000), LNCS vol 1917

[30] Anghinolfi D and Paolucci M 2008 International Journal of Operations Research 5(1) 44–60

[31] Abitz D, Hartmann T and Middendorf M 2020 preprint arXiv:2004.08433

[32] Davidovic T, Teodorovic D and Selmic M 2015 Yugoslav Journal of Operations Research 25(1) 33–56

[33] Sharma N, Sharma H and Sharma A 2019 IEEE/ACM Transactions on Computational Biology and Bioinformatics, IEEE