Metropolitan Delivery with Time Windows as a Scheduling Problem

Wojciech Bożejko
and Czesław Smutnicki
Wroclaw University of Science and Technology
Wroclaw, Poland
Email: wojciech.bozejko,
czeslaw.smutnicki@pwr.edu.pl

Mieczysław Wodecki
Wroclaw University
Wroclaw, Poland
Email: mwd@ii.uni.wroc.pl

***Abstract*—The paper deals with a complex transportation problem with an immediate practical application. A deterministic model is considered. The vehicle has to perform a route through the city to fulfil set of delivery demands having defined time windows to make service. We would like to find the route with minimal cost measured by weighted sum of earliness/tardiness penalties caused by violation of determined time windows. This problem is equivalent to the scheduling problem with earliness and tardiness penalties known already in the literature. As the added value of this paper, we provide some new theoretical properties based on so called block approach. Using these properties we devise a efficient metaheuristic algorithm to solve the problem. Paper provides components of the algorithm as well as results of computer test. Numerical tests confirm better algorithm quality evaluation comparing to other known so far approaches. The problem can be extended on the fleet of vehicles with limited capacities and pickup and/or delivery demands.**

I. Introduction

Depending on the character of the input data and crucial features of real transportation system, the systems can be modelled in different ways, with the fundamental goals expressed as: "to model", "to simulate behaviour", or "to optimize". Skipping consciously the long list of particular transportation problems formulated till now, modelling technologies, theoretical tools dedicated for modelling and solving, we refer in this paper chiefly to the class of transportation problems built on the ground of discrete/combinatorial optimization (CO).

CO problems derived from practice, because of characteristic features of the whole domain, require relatively high calculation power in order to find a solution satisfactory for the user, [18]. In case of batch planning, scheduling and load balancing this power has to be supplied in a relatively short slot of time, to hammer out the properly good solution, applied next in the long time period. The amount of calculations has tendency contrary to the quality of provided results.

Researchers' and practitioners' view on optimization tasks generated by CO problems to realize an on-line decision, resource usage balancing, production and/or transport planning, scheduling, timetabling, etc. significantly has changed in recent years. Huge effort has been done by scientist in order to reinforce power of solution methods and to fulfil expectations of practitioners, see also the comprehensive review in [19] and excellent monograph [24]. Metaheuristics, perceived as universal "medicine" for basic troubles of CO algorithms, have reached the limit in very recent years and are replaced by a new ultramodern approaches being as yet in the development phase. Successive progress is expected now in the application of parallel methods, [1], [3], [19].

CO problems with unimodal, convex, differentiable scalar goal functions disappeared from research labs, because a lot of satisfactory efficient methods were already designed. There are still remained very hard cases: multimodal, multi-criteria, non-differentiable, NP-hard, discrete, with huge dimensionality, with exponential increase of the number of local extremes, without a priori information about data, etc. These practical tasks, generated by computer systems and networks, industry and market, evoke serious troubles in the process of seeking global optimum. Any success in algorithms development strike practitioners fancy, so permanent research in this area are still welcome.

II. Delivery Problem

The Delivery Problem with Time Windows (DPTW) is defined formally as follows. Transportation network is described by digraph $G = (N, A)$, where $N = \{1, 2, \ldots, n\}$ is the set of nodes and $A \subset N \times N$ is the set of arcs. The distance between nodes is expressed by matrix $[s]_{n\times n}$, where s_{ij} is the distance (in time units) necessary to pass from i to its immediate successor j. Each node $i \in P$ has service time p_i, and time window (e_i, d_i) in which delivery should be performed in the ideal case. Without losing generality one can assume that node 1 is the depot and contains single vehicle. We assume that time windows are soft and can be violated under some additional cost. The overall aim is to find optimal tour of the vehicle passing through all nodes from N as well as the schedule $S = [S_1, S_2, \ldots, S_n]$ of visit nodes on the tour, which minimizes cost function

$$K(S) = \sum_{i=1}^{n} (v_i E_i + w_i T_i) \tag{1}$$

where $E_i = \max\{0, e_i - S_i\}$ denotes too early visit, $T_i = \max\{0, S_i - d_i\}$ denotes too late visit, v_i, w_i are arbitrary

978-1-4673-9486-4/16/$31.00

weights. The problem is also known in the literature as a variant of Traveling Salesman Problem with Time Windows (TSPTW), being a generalisation of classical TSP, [11], [17]. DPTW can be considered a source and fundamental problem for wide family of pickup-and-delivery problems considered in the literature, [2], [9], [12], [16], together with a full spectrum of solution methods from exact to metaheuristics. Nevertheless, in order to present crucial features of our original approach based on blocks' notions, we focus the description on a problem with medium generality of the mathematical model.

III. Scheduling Problem

The delivery problem presented above is equivalent to the scheduling problem shown in this Section. Further analogies and extensions will be discussed at the end of this paper.

Set of jobs $N = \{1, 2, \ldots\}$ has to be processed sequentially (at most one job at each time moment) on a single machine. Each job has defined: p_i processing time, earliest ready time e_i, due date d_i, $e_i \leq d_i$, and weights $v_i, w_i \geq 0$, $i = 1, 2, \ldots, n$. There is also given sequence-dependent setup times s_{ij}, $i, j = 1, 2, \ldots, n$, $i \neq j$. Job earliness we define as $E_i = [e_i - C_i]^+$, job tardiness as $T_i = [C_i - d_i]^+$, where $[x]^+ = \max\{0, x\}$ and C_i is the job completion time in the schedule $C = [C_1, C_2, \ldots, C_n]$. Note that job starting times $S_i = C_i - p_i$ are defined in a unique way, so schedule can be characterised equivalently either by vector C or by vector $S = [S_1, S_2, \ldots, S_n]$. Our aim is to find schedule C which minimizes total cost defined as follows

$$K(C) = \sum_{i=1}^{n} (v_i E_i + w_i T_i) \tag{2}$$

For the single machine case, the schedule C can be found on the base of job service sequence, which commonly is represented by a permutation $\pi = (\pi(1), \pi(2), \ldots, \pi(n))$ on N. Using permutation, one can formulate two mutually different schedules: (1) as soon as possible (ASAP), without machine idle time, (2) allowing machine idle time (FREE). For ASAP schedule, job completion times can be found from quite intuitive equation calculated for $i = 1, 2, \ldots, n$, which can be converted next to a recurrence form run in $O(n)$ time,

$$C_{\pi(i)} = s_{0,\pi(1)} + \sum_{j=1}^{i-1} (p_{\pi(j)} + s_{\pi(j),\pi(j+1)}) + p_{\pi(i)}. \tag{3}$$

For ASAP, substituting (3) to expressions on E_i and T_i, cost function (2) can be re-defined for π as the new argument

$$K(\pi) = \sum_{i=1}^{n} (v_{\pi(i)} E_{\pi(i)} + w_{\pi(i)} T_{\pi(i)}). \tag{4}$$

ASAP criterion value (4) can be found in a time $O(n)$ for the given π. FREE schedule C for the given π as well as $K(\pi)$ can be found by solving the following optimization problem

$$K^*(\pi) = \min_{E,T,C} \sum_{i=1}^{n} (v_i E_i + w_i T_i) \tag{5}$$

$$E_i = [e_i - C_i]^+, \;\; T_i = [C_i - d_i]^+, \;\; i = 1, 2, \ldots, n, \tag{6}$$

$$C_{\pi(i)} \geq C_{\pi(i-1)} + s_{\pi(i-1),\pi(i)} + p_{\pi(i)}, \;\; i = 1, 2, \ldots, n, \tag{7}$$

$$s_{0,\pi(1)} + p_{\pi(1)} \leq C_{\pi(1)}. \tag{8}$$

For both cases, ASAP and FREE, our aim is to find π^* which minimizes $K(\pi^*) = \min_\pi K(\pi)$, where $K(\pi)$ is given either by (4) or by (5). This problem denoted as $1|setup|\sum(v_i E_i + w_i T_i)$, using Graham notation, is strongly NP-hard, because already the problem $1||\sum w_i T_i$ is strongly NP-hard. Thus only approximate approaches will be considered further as a potential solution method.

It is clear that $K^*(\pi) \leq K(\pi)$, thus (5)-(8) is not only more attractive but also better fits to reality. Problem (5)–(8) can be converted to Linear Programming (LP) task, although exist in the literature simplest methods with the polynomial time computational complexity however dedicated for certain very special cases. Seeing in broader context, because of problem NP-hardness, approximate algorithms seems to be only reasonable alternative. Using commonly used local search approach (like tabu search) with swap moves, cost function has to be calculated $O(n^2)$ times. For ASAP schedules, time necessary to check all neighbouring solutions increases to $O(n^3)$, for FREE schedules – significantly more. Since cost of solving LP is rather high, several alternative approaches to find $K(\pi^*)$ are possible: (A) approximate $K^*(\pi)$ by $K(\pi)$ of ASAP and search the best permutation π in terms of $K(\pi)$, whereas at the end solve once $K^*(\pi)$, (B) search the best permutation π in terms of $K^*(\pi)$ using $K\pi)$ as the fast method of preliminary evaluation of examined π. In both cases, any special properties of the problem are especially welcome.

IV. Blocks

This section aim is to introduce concept of *blocks* which allow us to significantly reduce the size of the neighbourhood used in the local search method (tabu search in our case), thus speeds up the search as well as reduces the amount of calculations performed. Using *swap* moves we skip those made inside the block. The idea of blocks is based on approaches proposed in [4], [5], [22] for the single machine total tardiness problem, i.e. without earliness.

Let us consider a permutation π in model ASAP of the schedule. Each job can be processed *early* $C_i < r_i$, *on time* $r_i \leq C_i \leq d_i$, or *tardy* $d_i < C_i$. A sequence of successive jobs $\pi_{ab} = (\pi(a), \pi(a+1), \ldots, \pi(b))$, $1 \leq a \leq b \leq n$, with specific early/on-time/tardy properties in the permutation π will be called the *block*. Set of jobs from the block will be denoted hereafter by $B_{ab} = \{\pi(a), \pi(a+1), \ldots, \pi(b)\}$. Length $L(\pi_{ab})$ of the block π_{ab} we define as

$$L(\pi_{ab}) = s_{\pi(a-1),\pi(a)} + \sum_{j=a}^{b} (p_{\pi(j)} + s_{\pi(j),\pi(j+1)}). \tag{9}$$

To complete the definition we set $\pi(0) = 0 = \pi(n+1)$ and extend setups by initial and final values $s_{0,j}$ and $s_{j,0}$. Notice,

finding the minimal measure of (9) is equivalent to TSP, which is strongly NP-hard. Cost of the block represented by π_{ab} is

$$K(\pi_{ab}) = \sum_{i=a}^{b} (v_{\pi(i)} E_{\pi(i)} + w_{\pi(i)} T_{\pi(i)}). \tag{10}$$

Our overall aim is to re-arrange the block in order to optimize its structure. Optimization may refer to the cost (10) as well as to the length (9). Instead of seeking optimal sequence of jobs from the block, we refer to suboptimal arrangement of B called θ-optimal (see definition in the sequel of this paper), which we hope will be close to optimal one. We distinguish fundamentally three types of blocks:

- early jobs, E-block (see Fig. 1),
- on-time jobs, O-block (Fig. 2),
- tardy jobs, T-block (Fig. 3).

B is the block of early jobs (E-block) if:

- every job from B scheduled as the last one in B is early,
- local optimization of jobs from B with respect to cost function $K(\pi)$ provides θ-optimal solution π',
- after local optimization made in the previous step, we have $L(\pi') \leq L(\pi_{ab})$.

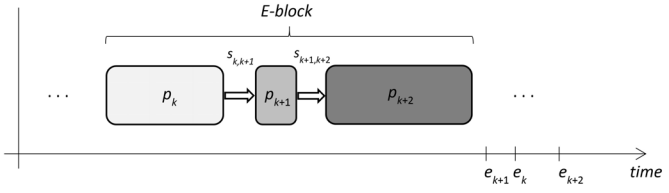

Fig. 1. E-block.

B is the block of tardy jobs (T-block) if:

- every job from B scheduled as the first in B is tardy,
- local optimization of jobs from B with respect to cost function $K(\pi)$ provides θ-optimal solution π',
- after local optimization made in the previous step, we have $L(\pi') \leq L(\pi_{ab})$.

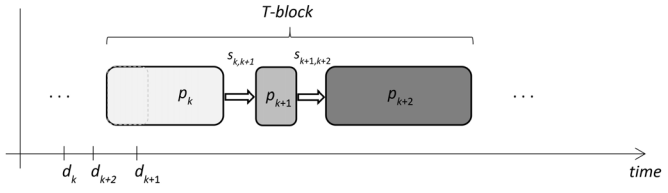

Fig. 2. T-block.

B is the block of on-time jobs (O-block) if:

- every job from B is on-time,
- local optimization of jobs from B provides θ-optimal solution π' with respect to block length.

Alternation of jobs form tardy block can provide better solution than π in terms of the cost $K(\pi)$ by if: (1) we decrease the component of the cost derived from (9), (2) we reduce

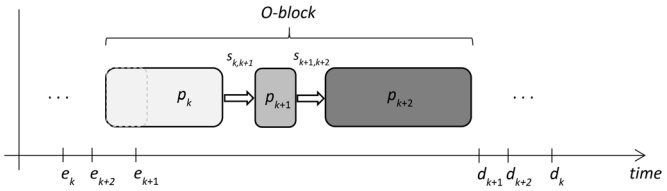

Fig. 3. O-block.

the starting time of the task appeared after block B, i.e task $\pi(b+1)$.

We can break each permutation into blocks in such a way, that any element of the permutation belongs to a block. In a pessimistic case it can be one elementary block.

Remark 1: For any permutation π there are partitions into blocks such, that every of them is: E-block, or O=block, or T-block.

V. Properties

Block notions are attractive because allow us to eliminate a priori a large number of neighbouring solutions generated from π, [21]. To be appropriately strong, we expect at least three features: (a) block contains large number of elements, (b) number of blocks is as large as possible, (c) block is arranged in sub-optimal order accordingly to its definition. In the sequel we discuss the possibility of re-organizing block structure to optimize its structure. To this aim we refer to TSP and following Assignment Problem (AP), find

$$\min_{\pi} \sum_{i=1}^{m} c_{i,\pi(j)} \tag{11}$$

where $[c]_{m\times m}$ is the cost matrix. Problem is classical in the discrete optimization and owns polynomial time algorithm with computational complexity $O(m^3)$. It is well known that optimal solution of AP is a lower bound of optimal solution of TSP with distance matrix c. Thinking about solution of TSP (as an auxiliary problem for TSPTW) we refer to the following approaches: (1) constructive heuristics (e.g. NN), (2) improvement heuristic (e.g. 2-opt, 3-opt), (3) local improvement (various metaheuristics), (4) hybrid, (5) exact (e.g. B&B scheme).

Starting from the block B_{ab} notion we define set of all permutations $\Pi_{ab} \subseteq \Pi$ obtained by alternating only jobs from B_{ab}, cost of the block for π_{ab} given in (10) and length of the block for π_{ab} re-defined in more convenient (equivalent) way

$$L(\pi_{ab}) = C_{\pi(b)} + s_{\pi(b)\pi(b+1)} - C_{\pi(a-1)}, \tag{12}$$

where $C_{\pi(j)}$ are given by recursive version of (3), namely

$$C_{\pi(i)} = C_{\pi(i-1)} + s_{\pi(i-1),\pi(i)} + p_{\pi(i)}, \;\; i = 2, \ldots, n, \tag{13}$$

with the initial condition $C_{\pi(1)} = s_{0,\pi(1)} + p_{\pi(1)}$.

Let B_{ab} be a block. We refer to limitary values of the cost/length over permutations from Π_{ab}

$$\underline{F}_{ab} = \min\{F(\sigma) : \;\; \sigma \in \Pi_{ab}\}, \tag{14}$$

$$\overline{F}_{ab} = \max\{F(\sigma) : \ \sigma \in \Pi_{ab}\} \quad (15)$$

where F means K or L. From the definition, for any $\gamma \in \Pi_{ab}$ we have $\underline{F}_{ab} \leq F(\gamma) \leq \overline{F}_{ab}$. Unfortunately, generally different permutations ensure minimization of particular measures K and L. Therefore, in the sequel, we use notion θ-optimal permutation $\pi^\theta \in \Pi_{ab}$, defined by

$$LB \leq \underline{F}_{ab} \leq F(\pi^\theta) \leq \overline{F}_{ab} \leq LB + \theta(UB - LB) \quad (16)$$

Lower Bound (LB) and Upper Bound (UB) depend on type of blocks and the block location, i.e. B_{ab}, nevertheless we skip this fact in the denotation. θ is a parameter chosen experimentally. In the sequel we discuss methods of finding LB and UB in each particular case.

A. LB for length of O-block

Cost of O-block is zero, so only the length of the block is considered in the context of the evaluation. The problem of length reduction is equivalent to TSP. Indeed, sequence of processing times p_i, $i \in B_{ab}$ does not influence on the block length, but sequence-dependent setup times. The problem of optimal arrangement of jobs inside the block with the criterion of the block length can be evaluated and found by AP (this is lower bound for TSP) with respect to $b-1+1$ setup times of jobs from this block plus "initial" and "final" setup time $s_{a-1,a}, \ \ s_{b,b+1}$. Thus, cost coefficients c_{ij} for AP are defined as setup times s_{ij} between jobs $i, j \in B_{ab} \cup \{a-1, b+1\}$ (distance between towns in TSP). Transformation of indices between B_{ab} and c matrix is obvious.

B. UB for length of O-block

Similarly as above, due to zero cost, only the length of the block is considered in the context of the evaluation. UB for the block length evaluation is provided by using 2-opt method of TSP with distances equal setup times for jobs from $i, j \in B_{ab} \cup \{a-1, b+1\}$.

C. LB for the cost of T-block

The problem of optimal arrangement of jobs inside T-block with respect to the cost can be evaluated by specific task of AP. Cost matrix for AP is defined as the approximation c_{ik} of weighted tardiness of job $i \in B_{ab}$ processed as k-th in order in this block. To simplify description, one can assume $B_{ab} = \{1, 2, \ldots, m-1\}$, $a = 1$, $m = b+1$. To find values c_{ik} we use an auxiliary graph with the set of weighted nodes $M = B_{ab} \cup \{a-1, b+1\}$ and set of weighted arcs $A \subset M \times M$. Node $i \in M$ has weight p_i, arc $(i, j) \in A$ has weight s_{ij}; nodes $a-1$ and $b+1$ have weight zero. Approximation of starting time of job i scheduled on position k is represented by shortest path t_{ik} with exactly $k+1$ arcs. Coefficient c_{ik} represents cost of scheduling job i on position k in the block, on the base of which we can calculate weighted tardiness component.

D. UB for the cost of T-block

At the begin we solve AP with complementary function "maximization" instead of minimization. Cost coefficients have been defined in the same way. Then we use 2-opt procedure to improve obtained sequence of jobs in block.

E. Length of T-block, E-block

LB and UB for the length of T-block and E-block are calculated in the same way as LB and UB, respectively, for O-block.

F. LB for the cost of E-block

We use procedure analogous to this described in Section "LB for cost of T-block". Coefficients c for AP determine earliness cost derived from processing job i as k-th in the block.

G. UB for E-block

We use approach analogous to this described in Section "UB for cost of T-block"

VI. Solution method

The proposed solution method we built fundamentally on tabu search (TS), [10], approach with so called swap local neighbourhood. Built-in block properties differ essentially our TS approach from other TS approaches used to solve this problem, [6]. Block properties are implemented in order to: (1) reduce the neighbourhood size, (2) reduce the cost of neighbourhood search, (3) direct the search into most promising regions of the solution space. Since TS realizes well process of the search intensification, to ensure proper balance between search intesification and diversification we embeded TS into more general Scatter Search scheme, [10]. Thus, finally proposed our Scatter Search (SS) algorithm was based on the idea of executing multiple path-relinking procedures between elements from a solutions set and improving them by using tabu search metaheuristics used as an effective local search procedure. We consciously skip here many technical details necessary to implement the algorithm, focus rather on theoretical properties crucial for algorithm design.

VII. Computational experiments

Parallel version of scatter search metaheuristics, [3], was implemented in C++ and run on HP Z1 workstation equipped with Intel Xeon E31280 CPU 3.50 GHz and 8 GB RAM working under Windows 7 operating system. Computational experiments of the proposed Scatter Search (SS) algorithm were done and compare with the common benchmarks from literature [7] and results of other metaheuristics for the considered problem [8], [14], [13], [20].

Each instance was run 10 times. For each run, each instance and for each algorithm A we define Percentage Relative Deviation

$$PRD = 100 \cdot \frac{K^A - K^*}{K^*} \quad (17)$$

of the cost K^A provided by algorithm A with respect to the reference cost K^* from [7]. Average value $APRD$ was calculated among 10 runs for each instance and each algorithm.

Table I shows the average relative percentage deviation (APRD) to reference solutions published in [7] and SA, GA and TS approaches from the paper [14], ACO from [13] and RBS from [20] compared to the proposed SS algorithm. Standard deviations of the SS algorithm denoted as σ^{SS} are determined over 10 runs for each instance.

TABLE I
APRD (%) FOR TESTED ALGORITHMS AGAINST SS.

n^o	SA	GA	TS	ACO	RBS	SS	σ^{SS}
1-10	-20.00	-22.83	-19.12	-13.77	39.06	-24.79	5.19
11-20	-20.89	-27.60	-18.46	-4.73	53.33	-42.22	25.92
21-30	-30.39	-30.93	-29.18	-14.13	345.64	-26.81	34.08
31-40	-6.86	-6.42	-5.81	-1.52	-8.10	-9.10	3.63
41-50	-5.21	-5.65	-5.33	-0.85	-4.55	-5.95	2.57
51-60	-5.29	-5.65	-4.44	-22.29	-22.84	-27.05	6.64
61-70	-7.25	-6.56	-7.25	-3.30	13.33	-7.80	4.68
71-80	-15.39	-15.02	-16.32	-5.63	-29.59	-17.82	7.42
81-90	-0.66	-0.56	-0.56	0.17	-0.07	-0.94	0.34
91-100	0.47	0.50	0.11	0.27	0.29	-1.09	1.31
101-110	-0.60	-0.24	-0.64	0.26	0.21	-3.36	0.53
111-120	-0.23	-0.44	-0.23	-0.86	-0.44	-1.02	1.28
AV	-9.32	-9.97	-8.90	-5.58	36.76	-14.47	7.80

VIII. CONCLUSION

As it is shown in the Table I proposed Scatter Search (SS) approach has the best APRD among considered algorithms and it is 31.0% better than Genetic Algorithm, 38.6% better than Tabu Search, 61,3% better than Ant Colony Optimization and 353.8% better than RBS Beam Search procedure.

IX. EXTENSIONS

Multiple vehicles in the depot can be modelled in terms of scheduling problem with parallel machines instead of the single machine. There is still certain freedom with interpretation of *identical*, *uniform*, *unrelated* machines in the scheduling model in the relation to transportation system. Clearly, workload of vehicles plays here significant role. Pickup points can be modelled as usual as delivery points with negative demand. Such modelling technology allow us to control current vehicle load and treat them as an additional constraint. To prevent frequent elimination of unfeasible solutions, we prefer penalty for the overloading trucks. Metropolitan map actually allows to go several times through the same locations, so crossroad points (with no delivery and no time window) are successive extensions. Blocks properties can be introduced in most of the extended in this way problems, making them useful for constructing more efficient algorithms.

ACKNOWLEDGMENT

The paper is supported by the grant S50204 funded by Ministry of Science and Higher Education performed in Wroclaw University of Science nad Technology, Faculty of Electronics.

REFERENCES

[1] E. Alba, Parallel metaheuristics: a new class of algorithms, John Wiley & Sons, 2005.
[2] G. Berbeglia, J.F. Cordeau, I. Gribkovskaia, G. Laporte, Static Pickup and Delivery Problems: A Classification Scheme and Survey, *Top*, 15(1), 1–31, 2007.
[3] W. Bożejko, A new class of parallel scheduling algorithms, Publishing House, Wroclaw University of Science and Technology, Wroclaw, 2010.
[4] W. Bożejko, J. Grabowski, M. Wodecki, Block approach-tabu search algorithm for single machine total weighted tardiness problem, *Computers & Industrial Engineering*50(1/2), 1–14, 2006
[5] W. Bożejko, Parallel path relinking method for the single machine total weighted tardiness problem with sequence-dependent setups, *Journal of Intelligent Manufacturing* 21(6), 777–785, 2010.
[6] W.C. Chiang, R. Russel, A reactive tabu search metaheuristic for the vehicle routing problem with time windows, INFORMS Journal on Computing 9, 417-430, 1997.
[7] V.A. Cicirello, S.F. Smith, Enhancing stochastic search performance by value-based randomization of heuristics. Journal of Heuristics 11, 5–34, 2005.
[8] V.A. Cicirello, Non-Wrapping Order Crossover: An Order Preserving Crossover Operator that Respect Absolute Position, 8th Annual Genetic and Evolutionary Computation Conference GECCO 2006, ACM Press 2006; 1125–1131.
[9] Y. Dumas, J. Desrosiers, F. Soumis, A dynamic programming solution of the large-scale single vehicle dial-a-ride problem with time windows, *American Journal of Mathematical and Management Science* 16, 301-325, 1986.
[10] F. Glover, M. Laguna, Tabu search. Kluwer Academic Publishers, Boston, MA, 1997.
[11] G. Gutin, A.P. Punnen (Eds), The Traveling Salesman Problem and its Variation, *Combinatorial Optimization* 12, Springer, 2007.
[12] H. Li, A. Lim, A metaheuristic for the pickup and delivery problem with time windows, *International Journal on Artificial Intelligence Tools*, 12(02), 173–186, 2003.
[13] C.J. Liao, H.C. Juan, An ant optimization for single-machine tardiness scheduling with sequence-dependent setups, Computers & Operations Research 34, 1899–1909, 2007.
[14] S.W. Lin, K.C. Ying, Solving single-machine total weighted tardiness problems with sequence-dependent setup times by meta-heuristics, International Journal of Advanced Manufacturing Technology 34(11), 1183–1190, 2007.
[15] H. Psarafis, A dynamic programming solution to the single vehicle many-to-many immediate request dial-a- ride problem, *Transportation Science* 14, 130-154, 1980.
[16] H. Psarafis, An exact algorithm for the single vehicle many-to-many immediate request dial-a-ride problem, Transportation Science 17(4), 351-361, 1983.
[17] G. Reinelt, The Traveling Salesman: Computational Solutions for TSP Applications. Springer-Verlag, 1994.
[18] C. Smutnicki, Optimization technologies for hard problems., in: Fodor J., Klempous R., Araujo C. P. S. (eds.), Recent advances in intelligent engineering systems. Berlin, Heidelberg, Springer, 79–104, 2011.
[19] C. Smutnicki, W. Bożejko, Parallel and Distributed Metaheuristics, Lecture Notes in Computer Science 9520, 72–79, Springer 2015.
[20] J.M.S. Valente, R.A.F.S. Alves, Beam search algorithms for the single machine total weighted tardiness scheduling problem with sequence-dependent setups, Computers & Operations Research 35, 2388–2405, 2008.
[21] M. Wodecki, Methods of Aggregation in Discrete Optimization Problems (Polish), Publishing House, Wroclaw University of Science and Technology, Wroclaw, 2009.
[22] M. Wodecki, A block approach to earliness-tardiness scheduling problems, *International Journal on Advanced Manufacturing Technology* 40, 797–807, 2009.
[23] J. Yang, P. Jaillet, H. Mahmassani, Real-Time Multivehicle Truckload Pickup and Delivery Problems, *Transportation Science* 38(2), 135-148, 2004.
[24] X.S. Yang, Nature-Inspired Optimization Algorithms, Elsevier, London, 2014.