Computer Science • 18 (1) 2017 http://dx.doi.org/10.7494/csci.2017.18.1.41

Ignacy Kaliszewski
Janusz Miroforidis
Jarosław Stańczak

THE AIRPORT GATE ASSIGNMENT PROBLEM – MULTI-OBJECTIVE OPTIMIZATION VERSUS EVOLUTIONARY MULTI-OBJECTIVE OPTIMIZATION

Abstract *In this paper, we approach the Airport Gate Assignment Problem by Multi-objective Optimization as well as Evolutionary Multi-objective Optimization. We solve a bi-criteria formulation of this problem by the commercial mixed-integer programming solver CPLEX and a dedicated Evolutionary Multi-objective Optimization algorithm. To deal with multiple objectives, we apply a methodology that we developed earlier to capture decision-maker preferences in multi-objective environments. We present the results of numerical tests for these two approaches.*

Keywords airport gate assignment problem, Evolutionary Multi-objective Optimization, mixed-integer programming

Citation Computer Science 18 (1) 2017: 41–52

1. Introduction

This work is a continuation of earlier research reported in [7, 8], where an approach for solving a bi-criteria model of the Airport Gate Assignment Problem (AGAP) was presented. The idea was to assist airport ground services in a small- or medium-sized airport in assigning landing flights to gates under conflicting criteria.

With the exception of small-sized problems, the exact methods are rarely used to solve such difficult problems ([3, 9]); however, we harness the commercial optimization package CPLEX for this task in our work. We complement this work by mirroring CPLEX computations by a dedicated Evolutionary Multi-objective Optimization (EMO) algorithm. Our intention is to compare the two distinct optimization paradigms – exact versus approximate – with respect to tractable sizes, computing time, and solution quality (suboptimality gap).

To cope with multiple objectives, we assume human interactions (the decision maker, DM) with a multi-objective model. To operationalize, we make use of a methodology to capture the DM's preferences in the course of interactive decision-making processes ([6]).

In the next section, we present two models of the AGAP problem: one for the exact optimization with mixed-integer programming (MIP) optimization methods and the other suitable for Evolutionary Multi-objective Optimization (EMO) methods. The latter is much more compact than the MIP model (where all logical conditions are to be modeled with the use of binary variables). In Section 3, we describe how efficient solutions to the AGAP problem are derived (in the sense of Pareto) with the use of the preference capture methodology we apply. Section 4 contains the description of the dedicated EMO algorithm. In Section 5, the results of numerical experiments are presented, whereas Section 6 concludes.

2. The AGAP problem formulation

The main task in the AGAP problem is to assign landing flights to handling gates. If there is no gate available, the flight has to wait for an idle gate or it is handled on the apron. The ideal situation is when the total waiting time for all flights (the less, the better) is zero. In this case, the number of flights handled on the apron (the fewer, the better) is also zero. However, in case of congestion, both values become positive. There is a clear trade-off between them – an improvement in one degrades the other.

Bellow, we shall assume that:

1. The airport is small, so passenger walking distances to gates do not affect the times of passenger transit and gate handling.
2. Each gate can handle any flight.
3. Gate handling does not influence any other gate handling.
4. Each flight can wait for up to T^{max} for being handled at a gate, and after that, the flight is handled (discharged) on the apron; the capacity of the apron is not a limiting factor.

Although both of the approaches described below (EMO and CPLEX) solve the same problem, each approach requires a specific model. The use of two different models of the AGAP problem is the result of different requirements for each approach we apply (see the detailed discussion of these issues in [5]); however, both models are equivalent and built on the same set of assumptions. A model of the AGAP problem (suitable for CPLEX) necessitates the use of binary variables to represent logical conditions of the AGAP problem in the form of linear constraints. This is the main cause of combinatorial-type growth of such models with increasing number of flights and gates.

In principle, an AGAP model built for CPLEX can be solved with an EMO algorithm, but this would be dramatically ineffective due to the unnecessary increase in the search space. Instead, a model suitable for evolutionary computations is proposed; this is presented in the next section.

2.1. The AGAP model for EMO

Here, we model the AGAP problem as a queuing system with the gates and apron as processors and landing flights as processes to be served (handled). Processes are deterministic, indivisible, and all need to be processed. Time is discrete with interval δ.

We propose the following model for solving the AGAP problem with EMO:

$$\min \sum_{l=1}^{m-1} \sum_{s=1}^{|k_l|} \left(t^p_{k_l(s)} - t^q_{k_l(s)} \right) \tag{1}$$

$$\min |k_m| \tag{2}$$

$$k_l(s) \in \{F_1, \ldots, F_n\}, \quad l = 1, \ldots, m \tag{3}$$

$$t^p_{k_l(s)} - t^q_{k_l(s)} \leqslant T^{max} \tag{4}$$

$$\left.\begin{array}{ll} t^p_{k_l(s)} & = \max\left(t^p_{k_l(s-1)} + t^o + \delta\,, t^q_{k_l(s)} \right) \\ t^p_{k_l(1)} & = t^q_{k_l(1)} \end{array}\right\} \quad \text{for } l \neq m \tag{5}$$

$$t^p_{k_l(s)} = t^q_{k_l(s)} \text{ for } l = m \tag{6}$$

where k_l – queue of flights assigned for handling at gate l, $l \leqslant m$, k_m – queue of flights assigned for handling at the apron (in the widely-used Erlang notation of queuing processes, they are $D/D/1$ queues with deterministic flight landing times, deterministic flight handling times, and single processors); $|k_l|$ – length of queue k_l; $F_1, \ldots, F_n$ – landing flights to be handled, indices correspond to the order of landing; n – number of flights; $k_l(s)$ – a flight in position s in queue k_l ; t^p – handling start time of some flight; t^q – landing time of some flight; T^{max} – maximal waiting time for flight handling at a gate; t^o – the time of flight handling.

Objective functions (1) and (2) are equivalent (respectively) to objective functions (12) and (14) of the MIP model presented in the next section. As shown in the next section, all conditions of the AGAP problem, which can be automatically managed in the EMO approach via queuing mechanisms, problem encoding, and genetic

operators (see Section 4), have to be explicitly specified in the case of the MIP model. In contrast, the only explicit conditions in the EMO model are (4) (limiting waiting time) and (5), (6) (queuing rules at the gates and at the apron).

2.2. The AGAP model for exact optimization – an MIP formulation

Let us recall that there are n flights and m gates.

Flight $j, j = 1, \ldots, n$, is described by arrival time a_j and handling time g_j. Time in the problem is discretized with interval δ. The maximal waiting time for flight handling at a gate is $T^{max} = \alpha\delta$ for some positive integer α.

Let $x^t_{j,i}$ be a binary variable equal to 1 if flight j is assigned to gate i in time t, and 0 otherwise. No flight can be assigned to any gate before its landing time; thus, for $t < a_j$, variables $x^t_{j,i}$ are undefined. Similarly, it is not possible to assign a flight to a gate in $t > a_j + \alpha\delta$ (if this condition holds, the flight is handled on the apron, since $\alpha\delta = T^{max}$) and variables $x^t_{j,i}$ for $t > a_j + \alpha\delta$ are undefined too. There are $m\sum_{j=1}^{n}(\alpha+1) = mn(\alpha+1)$ variables $x^t_{j,i}$.

Flight j can only be assigned to one gate, and this is modeled by n constraints (7)

$$\sum_{i=1}^{m}\sum_{t=a_j}^{a_j+\alpha} x^t_{j,i} \leqslant 1 \text{ for } j = 1, \ldots, n. \tag{7}$$

Let $y^t_{j,i}$ be a binary variable equal to 1 if gate i handles flight j in time t, and 0 otherwise. No flight can be assigned to any gate before its landing time; thus, for $t < a_j$, variables $y^t_{j,i}$ are undefined. Similarly, gate i cannot handle flight j in $t > a_j + g_j$ and variables $y^t_{j,i}$ are undefined too. There are $m\sum_{j=1}^{n}(\alpha + g_j + 1)$ variables $y^t_{j,i}$.

If flight j is assigned to gate i, $i = 1, \ldots, m$, in time t, then this gate is unavailable for other flights in time $t+1, \ldots, t+g_j$; hence, $m\sum_{j=1}^{n}\alpha(g_j+1)$ constraints

$$\begin{aligned} x^t_{j,i} &\leqslant y^t_{j,i}, \\ x^t_{j,i} &\leqslant y^{t+1}_{j,i}, \\ &\vdots \\ x^t_{j,i} &\leqslant y^{t+g_j}_{j,i}, \end{aligned} \tag{8}$$

must hold, where $a_j < t < a_j + \alpha$.

Constraints of type (8) can also be written in a shorter form as $m\sum_{j=1}^{n}(\alpha+1) = mn(\alpha+1)$ constraints:

$$g_j x^t_{j,i} \leqslant y^t_{i,j} + y^{t+1}_{j,i} + \ldots + y^{t+g_j}_{j,i}. \tag{9}$$

Only one flight can be handled by one gate in time, so:

$$\sum_{j=1}^{n} x^t_{j,i} \leqslant 1, \quad t = a_*, \ldots, \beta, \quad i = 1, \ldots, m, \tag{10}$$

where a_* denotes the index of the flight landing as the second. There are, at most, $m(\beta-\alpha_*+1)$ constraints (10); this number depends on the schedule of landing flights.

Gate i in moment t can handle, at most, one flight in time; so:

$$\sum_{j=1}^{n} y_{j,i}^{t} \leqslant 1, \quad t = a_*, \ldots, \beta, \quad i = 1, \ldots, m. \tag{11}$$

There are, at most, $m(\beta - \alpha_* + 1)$ constraints (10); this number depends on the schedule of landing flights.

The sum of all waiting times for flights to be handled at a gate is described by formula (12) and is the first objective function of the model (to be minimized):

$$f_1(x) = \delta \sum_{j=1}^{n} \left(\sum_{i=1}^{m} x_{j,i}^{a_j+1} + 2 \sum_{i=1}^{m} x_{j,i}^{a_j+2} + \ldots + \alpha \sum_{i=1}^{m} x_{j,i}^{a_j+\alpha} \right). \tag{12}$$

If flight j is not handled at any gate, it is handled on the apron; this is modeled by n constraints (13):

$$\sum_{i=1}^{m} \sum_{t=a_j}^{t=a_j+\alpha} x_{j,i}^{t} = 1 - y_j, \text{ for } j = 1, \ldots, n, \tag{13}$$

where y_j denotes a binary variable equal to 0, when flight j is handled at the gate and 1 otherwise (is handled on the apron). Variables y_j are used to formulate the second objective function (to be minimized):

$$f_2(x) = \sum_{j=1}^{n} y_j. \tag{14}$$

Objective functions (12) and (14) together with constraints (7), (9), (10), and (11) constitute the binary bi-criteria model of AGAP.

3. Derivation of efficient solutions

Let x denote a solution, X a space of solutions, X_0 a set of feasible solutions, and $X_0 \subseteq X$. Then, the multi-objective optimization problem is:

$$\begin{gathered} vmax f(x) \\ x \in X_0 \end{gathered} \tag{15}$$

where $f\colon X \to R^k$, $f = (f_1, \ldots, f_k)$, $f_l\colon X \to R$, $l = 1, \ldots, k$, $k \geqslant 2$, are objective functions, and $vmax$ denotes the operator of deriving all efficient solutions in X_0.

Solution $\bar{x}$ is efficient if $f_l(x) \geqslant f_l(\bar{x})$, $l = 1, \ldots, k$, implies $f(x) = f(\bar{x})$.

It is a well-established result (cf. [4, 6, 10]) that solution x is efficient[1] if and only if it solves the optimization problem.

$$\min_{x \in X_0} \max_{l} [w_l(y_l^* - f_l(x)) + \rho e^k (y^* - f(x))] \tag{16}$$

where $w_l > 0$, $l = 1, \ldots, k$, $e^k = (1, 1 \ldots, 1)$, y^* is such that $y_l^* > f_l(x)$, $l = 1, \ldots, k$, $x \in X_0$, and ρ is a positive "sufficiently small" number[2] [2].

By the "only if" part of this result, no efficient solution is a priori excluded from being derived by solving an instance of an optimization problem (16). In contrast to this, maximization of a weighted sum of objective functions over X_0 does not possess this property (in general, and especially in the case of problems with discrete variables).

At first glance, the objective function in (16) seems to be difficult to handle; however, observe that optimization problem (16) is equivalent to

$$\begin{aligned} &\min s, \\ &s \geqslant w_l \, (y_l^* - f_l(x)) + \rho e^k (y^* - f(x)), \quad l = 1, \ldots, k, \\ &x \in X_0. \end{aligned} \tag{17}$$

4. The Evolutionary Multi-objective Optimization algorithm for solving the AGAP problem

The Evolutionary Multi-objective Optimization algorithm we propose here uses the notion of domination (in the sense of Pareto), as the base for solution selection [2], and standard genetic operators tailored to the specific needs of the AGAP problem. It also uses the standard $(\mu+\lambda)$ strategy (μ – the cardinality of the parent population, λ – the cardinality of the offspring population) of population development with a fixed number of epochs.

Feasible solutions (in terms of evolutionary optimization algorithms – individuals) are encoded as queues of landing flights assigned to gates in a specific order. There is one queue for each gate and one for the apron, as illustrated in Figure 1. In the initialization step, landing flights are randomly assigned to a queue.

The positions of flights in queues and their handling times determine the individual flight waiting time for the gate assignment and, as a consequence, total waiting time.

If the waiting time for a flight reaches T^{max}, that flight has to be redirected to the apron. The functionality of the algorithm, called *scheduler*, takes care of this.

[1] Actually, variant is properly efficient for a formal treatment of this issue.

[2] In the literature, symbol w competes with λ, the former standing for *weights*. Here, we opt for symbol w to avoid clashing with symbol λ used in the EMO (as in Section 4). By changing w, different efficient solutions can be obtained.

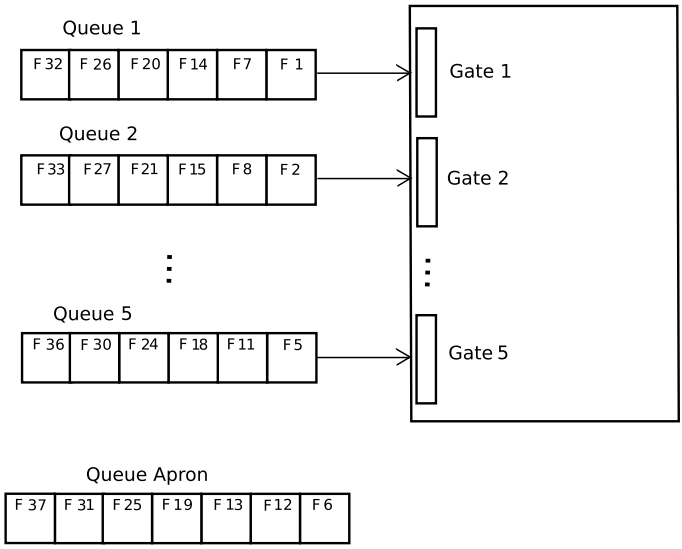

Figure 1. Encoding of feasible solutions in the proposed algorithm.

Offspring subpopulations are modified by five genetic operators:

- exchange of two randomly selected flights from two different randomly selected queues;
- relocation of a randomly selected flight from one queue to another randomly selected queue;
- exchange of all flights from a randomly selected gate queue with all flights from the apron gate (all gates are identical; hence, exchanging all flights between two gates is equivalent to gate renumbering – an action producing no added value);
- relocation of a flight waiting to be handled from a queue to another queue where no flight is handled at the moment; relocation takes place only when the time difference between the start of waiting and the start of being idle is not greater than 2δ;
- relocation of a flight from the apron queue to a gate queue where no flight is handled at the moment; relocation takes place only when the time difference between the start of waiting and the start of being idle is not greater than 2δ.

All of these operators produce feasible solutions to the problem. To avoid the burden of repairing infeasible solutions, the crossover operator is not used in the algorithm.

The algorithm uses a specific technique to trigger genetic operators [11]. In this technique, operators that have greater success in improving solutions are triggered more often. However, one operator can work better for one feasible solution and worse for another. Therefore, the performance of operators is controlled separately for each

feasible solution in the algorithm; for each feasible solution, operators are ranked with respect to their decreasing performance. The performance of operators is measured by the probabilities of operator success, recalculated each time an operator is used. Operators are selected with the respective probabilities. The method to calculate probabilities comes from *reinforcement learning*; for details, the reader is referred to e.g. [1, 12].

As any other EMO algorithm, our algorithm attempts to derive a great variety of efficient solutions (possibly all). To ensure this, we proposed a specialized selection method. The selection method works as follows:

- the current population is divided into three groups:
  1) non-dominated solutions, different in values of the fitness function, no clones,
  2) the remaining non-dominated solutions that have the same values of fitness function as the non-dominated solutions in the first group, no clones,
  3) dominated solutions and also clones of non-dominated solutions from the first two groups;
- all solutions from the first group are selected to the next offspring subpopulation;
- if, in the next offspring subpopulation, there are more solutions than the subpopulation limit, the excess is randomly chosen and deleted;
- if, in the next offspring subpopulation, there are fewer solutions than the subpopulation limit, the necessary number of individuals is randomly selected from the second group;
- if, in the next offspring subpopulation, there are still fewer solutions than the subpopulation limit, the solutions from the third group are added up to the subpopulation limit.

The possibility of selection for the new population non-dominated solutions with the same value of the fitness functions is a novel element of the proposed algorithm. To the best knowledge of the authors, this option has not yet been exploited in other researches reported in the literature. The advantage of this is twofold: first, it allows us to keep the subpopulation diverse to the limit; and second, it enriches the search space ("the landscape"), which has a plausible effect on the algorithm's behavior.

5. Numerical experiments

5.1. Data

Both methods (CPLEX and EMO) were tested on problems:

1) 2 gates, apron, and 4 flights, landing in time intervals: 0, 10, 35, 40;
2) 2 gates, apron, and 5 flights, landing in time intervals: 0, 10, 25, 35, 40;
3) 2 gates, apron, and 10 flights, landing in time intervals: 0, 10, 35, 40, 45, 50, 55, 60, 70, 85;
4) 2 gates, apron, and 15 flights, landing in time intervals: 5, 15, 30, 40, 45, 50, 55, 60, 70, 85, 95, 105, 130, 140, 145;

5) 3 gates, apron, and 30 flights, landing in time intervals: 5, 15, 30, 40, 45, 50, 55, 60, 70, 85, 95, 105, 130, 140, 145, 150, 155, 160, 165, 170, 180, 185, 190, 195, 200, 205, 215, 230, 240, 245;
6) 4 gates, apron, and 100 flights, landing in time intervals: 5, 15, 30, 40, 45, 50, 55, 60, 70, 85, 95, 105, 130, 140, 145, 150, 155, 160, 165, 170, 180, 185, 190, 195, 200, 205, 215, 230, 240, 245, 250, 265, 280, 290, 295, 300, 315, 330, 350, 385, 395, 405, 520, 540, 545, 550, 555, 560, 565, 570, 580, 585, 590, 595, 600, 605, 615, 630, 640, 645, 655, 670, 680, 685, 690, 695, 700, 705, 715, 725, 730, 735, 740, 745, 750, 755, 760, 765, 770, 780, 785, 790, 795, 800, 805, 815, 830, 840, 845, 850, 855, 860, 865, 870, 875, 880, 890, 895, 900.

In the experiments, we used four preselected sets of weights w, as in Table 1.

The data does not represent the operations of a real airport. They were generated "manually", but in a manner representative for situations met in practice.

Our EMO algorithm and data files prepared for CPLEX can be made available on request.

5.2. Discussion of results

Table 1 presents the results obtained by CPLEX and our dedicated EMO. For each instance of the AGAP problem, CPLEX derived an optimal solution, and each such solution (in virtue of Section 3) was an efficient solution to the problem (in the sense of Pareto). In the case of the EMO algorithm, the solutions derived were not always efficient, and this happened more frequently for larger problems. For instance, the EMO algorithm found only one efficient solution for problems with 100 flights; namely, the one with criteria values (0, 45).

It must be noticed that CPLEX does provide one efficient solution to the AGAP problem for one instance of weights in optimization problems (16) and (17). Thus, results of a single run of CPLEX and the EMO algorithm are different. One run of the EMO algorithm is expected to derive many efficient solutions (possibly all). To this aim, CPLEX has to be run multiple times.

Computation times for separate runs of CPLEX and times for the EMO algorithm are given in Table 1.

The computations were performed on different systems. In the case of CPLEX, it was the NEOS open platform run by the University of Wisconsin. The EMO algorithm, coded in C++, was run on a PC computer with a 4x3200MHz Intel processor and Linux system.

Running times for the EMO algorithm are averages of ten runs. All computations for the EMO algorithm lasted 10,000 epochs with parameters[3] $\mu = 500$ and $\lambda = 3500$. The number of epochs is too big for small problems and is proper for bigger ones, but this value was kept constant to allow for comparisons. For rather simple data sets 1–4

[3] Similarly as earlier in this text describing EMO, parameters "μ" and "λ" denote the cardinality of the population of EMO.

(described in the previous section) up to 20 epochs and for more complicated data set 5 up to 200 epochs of evolutionary computations were enough to derive all efficient solutions. In the case of the most difficult data set 6, 1000 epochs were enough to derive some solutions (but not all, which we know because some efficient solutions derived by CPLEX were not derived by the EMO algorithm).

Table 1

Results of numerical experiments – values of objective functions and computation times (#APRON – the number of flights handled at the apron, w – the parameter of the scalarizing function representing decision-maker preferences).

		CPLEX			EMO		
	w	WAITING TIME [min]	#APRON	SOLUTION TIME [s]	WAITING TIME [min]	#APRON	SOLUTION TIME [s]
1	(1.00, 0.00)	0	2	0.34	0	2	38.362
	(0.75, 0.25)	0	2	0.31	0	2	
	(0.50, 0.50)	0	2	0.53	0	2	
	(0.25, 0.75)	0	2	0.98	0	2	
	(0.00, 1.00)	45	0	0.72	45	0	
2	(1.00, 0.00)	0	3	0.28	0	3	70.323
	(0.75, 0.25)	0	3	0.41	0	3	
	(0.50, 0.50)	0	3	0.46	0	3	
	(0.25, 0.75)	0	3	0.17	0	3	
	(0.00, 1.00)	45	1	0.51	45	1	
3	(1.00, 0.00)	0	6	0.29	0	6	157.137
	(0.75, 0.25)	0	6	0.63	0	6	
	(0.50, 0.50)	0	6	0.79	0	6	
	(0.25, 0.75)	0	6	0.45	0	6	
	(0.00, 1.00)	30	5	0.41	30	5	
4	(1.00, 0.00)	0	9	0.14	0	9	249.62
	(0.75, 0.25)	0	9	0.27	0	9	
	(0.50, 0.50)	0	9	0.55	0	9	
	(0.25, 0.75)	0	9	1.17	0	9	
	(0.00, 1.00)	35	8	1.11	35	8	
5	(1.00, 0.00)	0	17	11.48	0	17	447.655
	(0.75, 0.25)	5	16	2.63	0	17	
	(0.50, 0.50)	15	15	2.17	0	17	
	(0.25, 0.75)	15	15	6.44	5	16	
	(0.00, 1.00)	15	15	82.80	15	15	
6	(1.00, 0.00)	0	45	36.29	0	45	857.018
	(0.75, 0.25)	10	44	466.67	0	45	
	(0.50, 0.50)	25	43	1230.33	0	45	
	(0.25, 0.75)	90	41	799.86	0	45	
	(0.00, 1.00)	230	39	905.57	30	43	

6. Conclusions

In this work, we have shown that EMO-based algorithms can be time-efficient and, if specialized, they are able to solve problems of impressive sizes (as in the case of the AGAP problem). For small problems, CPLEX can derive efficient solutions even faster than our specialized EMO algorithm, but the need to prepare data in a specified format, the need to have license for its use (for commercial applications), and its inability to cope with any nonlinearities so often met in real applications do not favor its use in practice.

Our future work will concentrate on enriching our algorithm with more-advanced mechanisms for selecting non-dominated solutions to successive offspring subpopulations and with more-advanced genetic operators.

References

[1] Cichosz P.: *Systemy uczące się*. WNT, Warszawa, 2000.

[2] Deb K., Pratap A., Agarwal S., Meyarivan T.: A Fast and Elitist Multiobjective Genetic Algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, vol. 6(2), pp. 182–197, 2002.

[3] Ding H., Lim A., Rodrigues B., Zhu Y.: New heuristics for the over constrained airport gate assignment problem. *Journal of the Operational Research Society*, vol. 55, pp. 760–768, 2004.

[4] Ehrgott M.: *Multicriteria Optimization*. Springer, 2005.

[5] Hu X., Di Paolo E.: An efficient genetic algorithm with uniform crossover for the multi-objective airport gate assignment problem. In: C.K. Goh, Y.S. Ong, K.C. Tan, eds., *Studies in Computational Intelligence*, vol. 171, pp. 71–89, 2009.

[6] Kaliszewski I.: *Soft Computing for Complex Multiple Criteria Decision Making*. Springer, 2006.

[7] Kaliszewski I., Miroforidis J.: On interfacing multiobjective optimization models – the case of the Airport Gate Assignment problem. In: *Proceedings of the 2nd International Conference on Application and Theory of Automation in Command and Control Systems (ATACCS'2012)*, pp. 93–97, IRIT Press, 2012.

[8] Kaliszewski I., Miroforidis J., Stańczak J.: Decision maker's preferences, airport gate assignment problem and multiobjective optimization. In: *Multiple Criteria Decision Making*, pp. 84–100, 2013.

[9] Marinelli M., Dell'Orco M., Sassanelli D.: A Metaheuristic Approach to Solve the Flight Gate Assignment Problem. *Transportation Research Procedia*, vol. 5, pp. 211–220, 2015.

[10] Miettinen K.: *Nonlinear Multiobjective Optimization*. Kluwer Academic Publishers, 1999.

[11] Stańczak J.: Biologically inspired methods for control of evolutionary algorithms. *Control and Cybernetics*, vol. 32, pp. 411–433, 2003.

[12] Sutton R., Barto A.: *Reinforcement Learning: An Introduction*. MIT Press, 1998.

Affiliations

Ignacy Kaliszewski
Systems Research Institute, Polish Academy of Sciences, ul. Newelska 6, 01-447 Warsaw, Poland, Ignacy.Kaliszewski@ibspan.waw.pl

Janusz Miroforidis
Systems Research Institute, Polish Academy of Sciences, ul. Newelska 6, 01-447 Warsaw, Poland, Janusz.Miroforidis@ibspan.waw.pl

Jarosław Stańczak
Systems Research Institute, Polish Academy of Sciences, ul. Newelska 6, 01-447 Warsaw, Poland, Jaroslaw.Stanczak@ibspan.waw.pl

Received: 21.12.2015
Revised: 4.10.2016
Accepted: 4.10.2016