ISSN 0005-1179, Automation and Remote Control, 2017, Vol. 78, No. 7, pp. 1276–1289. © Pleiades Publishing, Ltd., 2017.
Original Russian Text © A.B. Dolgui, A.V. Eremeev, V.S. Sigaev, 2017, published in Avtomatika i Telemekhanika, 2017, No. 7, pp. 125–140.

STOCHASTIC SYSTEMS

Analysis of a Multicriterial Buffer Capacity Optimization Problem for a Production Line

A. B. Dolgui[*,a], **A. V. Eremeev**[**,b], **and V. S. Sigaev**[***,c]

[*] *IMT Atlantique, LS2N, Nantes, France*
[**] *Sobolev Institute of Mathematics, Siberian Branch, Russian Academy of Sciences, Novosibirsk, Russia*
[***] *SJC "Avtomatika-servis," Omsk, Russia*
e-mail: [a] *alexandre.dolgui@mines-nantes.fr,* [b] *eremeev@ofim.oscsbras.ru,* [c] *sigvs@mail.ru*

Received December 19, 2015

Abstract—We consider a multicriterial optimization problem for volumes of buffers in a production line. We assume that the line has a series-parallel structure, and during its operation equipment stops occur due to failures, stops that are random in the moments when they arise and in their durations. The volumes of buffers are integer-valued and bounded from above. As criteria we consider the average production rate of the line, capital costs for installing buffers, and the inventory cost for intermediate products. To approximate the Pareto optimal set we use evolutionary algorithms SIBEA and SEMO. Problems with larger dimension experimentally support the advantage of the modified SEMO algorithm with respect to the hypervolume of the resulting set of points.

Keywords: production system, evolutionary algorithms, productivity, capital costs, inventory costs.

DOI: 10.1134/S0005117917070098

1. INTRODUCTION

In the design of automated control systems for production and a number of other decision making support systems, it is important to develop approximation algorithms for the set of Pareto optimal solutions. In control over such industrial systems as machining lines, flexible manufacturing systems, or automated assembly lines where parts move from one machine to the next with some transport mechanism, the following optimization problem arises for the volumes of buffers.

Due to equipment failures, during a line's operation, failures of units of equipment (EU) arise, with random starting times and durations. The consequences of these failures extend to adjacent operations because it either becomes impossible to pass a part to the next operation or there are no parts at the EU's input. The presence of buffers (reservoirs) for storing the parts between EUs lets one decrease the influence of failures on adjacent operations and improve the average production rate of a line. However, to install buffers one has to invest into additional capital costs and increase the number of parts being stored. The problem is to choose volumes of buffers while taking into account the line's average productivity, capital costs for installing the buffers, and storage costs for the parts.

The importance of such optimization problems for production lines has been demonstrated in [1]. A significant economic effect of applying the methods for solving such problems on automobile plants of PSA Peugeot Citroën has been shown in [2].

Analysis of industrial lines that accounts for equipment failures is usually done via constructing Markov models with discrete or continuous time under the assumption of geometric or exponential

distributions of EU's time to failure and their repair times (see, e.g., [3]). The duration of processing a part can be assumed to be either deterministic or random (usually with geometric, exponential, or Erlang probability distribution). In case of continuous time and deterministic durations of processing the parts individual non-Markov transitions are approximated by Markov ones under the assumption of exponential distributions of the corresponding random values [4–6]. Under sufficiently natural assumptions, the resulting Markov models have a stationary distribution (see [7], Chapter 2), and the average throughput and average number of parts in each buffer in steady state mode are well defined for them.

Most works in the literature on the problem of optimizing volumes of buffers have been related to single-criterial settings (see, e.g., [8–10]). Other publications have considered more than one criterion but used their weighted sum [11, 12]. In [13], the ant colony algorithm and the evolutionary algorithm from [14] have been adapted for the multicriterial optimization problem of buffer sizes. As the criteria they considered maximization of average production rate of a line, computed with a simulation algorithm, and minimization of the total size of buffers. A well-known version of the multicriterial genetic algorithm [15] has been adapted in [16] to a bicriterial problem of buffer allocation, where the criteria are average production rate of the line in the steady state and capital costs for installing the buffers.

One characteristic feature of this work is that in the problem setting we use three criteria: maximization of average line throughput in the steady state, minimization of capital costs for installing buffers, and minimization of average inventory costs for the parts in intermediate buffers. Computations of the average productivity of a line and average number of parts in the buffers use an efficient approximate method for analyzing it [4] based on replacing line segments with "equivalent" EU.

Unlike single-criterial variations of the problem, which use a convolution of criteria or place some criteria into constraints, the tri-criterial setting has not one optimal value of a criterion but rather an entire set of unimprovable values for the vector of three criteria (Pareto front). The considered tri-criterial problem setting appears to be the most relevant on the stage of finding a way to develop production, when we have not yet chosen the planned production rate and the structure of production line. On this stage, analysis of the Pareto front lets a decision maker make his or her choice among the set of unimprovable options of buffer distributions. On latter stages of decision making, when we have defined the required production rate or bounded the total volume of buffers, single-criterial problem settings may prove to be more relevant.

Exact methods for computing the average line throughput are known for the case of two and, in some special cases, three sequential EUs divided by buffers (see, e.g., the survey [3]). For the general case one uses approximate approaches such as decomposition, aggregation, or simulation modeling [3, 4].

Since there are no exact methods to compute the average line productivity and average number of stored parts, in this work to approximate the Pareto set we use evolutionary algorithms, whose application to this problem does not require us to compute the criteria exactly.

2. PROBLEM SETTING

Let n be the number of buffers and m be the number of EUs in the line. We denote by $\mathbb{Z}_+$ the set of nonnegative integers. We introduce the vector of variables $\mathbf{h} = (h_1, \ldots, h_n) \in \mathbb{Z}_+^n$, where h_i is the volume of the ith buffer, $i = 1, \ldots, n$. The structure of the line is represented by a directed series parallel graph G, where vertices $b_0, \ldots, b_{n+1}$ correspond to buffers and arcs $a_1, \ldots, a_m$ correspond to units of equipment. Each arc is directed from the vertex of the input buffer of the corresponding EU to the vertex of its output buffer. Multiple arcs are possible, they correspond to EUs with common buffers working in parallel. The only vertex b_0 that has no incoming arcs corresponds

to the input buffer of the line, and the only vertex b_{n+1} without outgoing arcs corresponds to the output buffer. Each vertex b_j is assigned with the maximal number of parts h_j that can be stored in buffer j, and $h_0 = h_{n+1} = \infty$. We assume that the line input always has a sufficient number of parts, and processed parts can be stored in the output buffer since it has unbounded size.

Any EU can be in states "operational," "failure" (EU is being repaired after it has broken down), "blocked" (impossible to transfer a processed part to the next operation), and "idle" (there are no parts at the EU's input). According to the above assumptions, EUs related to the buffer on system input cannot be in the "idle" state, while EUs related to the buffer on the system output cannot be in the "blocked" state.

In case when the buffer is full at the output of some EU, the part processed by it cannot be moved out of the EU and remains in the EU until some space frees up in the buffer. In case when several EUs are waiting for free space in the same buffer, when the space is freed for one part, one of these EUs chosen with equal probabilities. In a situation when several EUs are waiting for parts to come from some buffer which is empty at the present time, at the moment when part appears in the buffer one of these EUs chosen with equal probabilities and stops being idle (the arrived part is loaded to this EU for processing).

Each arc a_i in graph G is characterized by a triple of parameters $(T_i^{\mathrm{F}}, T_i^{\mathrm{R}}, U_i) \in [1, \infty)^2 \times \mathbb{N}$, where $\mathbb{N}$ is the set of natural numbers. While in the "operational" state, EU number i has a constant time for processing a part, denoted by $U_i \in \mathbb{N}$, $i = 1, \ldots, m$. Failures and repairs for different EUs occur independently, and time to failure, similar to the repair time for an EU, has a geometric distribution. For each EU, $T_i^{\mathrm{F}} \in [1, \infty)$ is the average time to failure, and $T_i^{\mathrm{R}} \in [1, \infty)$ is the average repair time. If an EU number i is operational, and its input buffer j is nonempty and buffer j' at its output is not full (i.e., it is not in the "failure," "blocked," or "idle" state), then every U_i units of time this EU adds to buffer j' one part and takes one part from buffer j. In the "blocked" and "idle" states EU does not fail. In the "failure," "blocked," and "idle" states EU does not change the numbers of parts in its input and output buffers. At the moment of failure, processing of the current part is stopped, and after repair it is continued until the completion. The processing time for a part after repair equals the processing time remaining at the moment of failure.

The possibility to model such a production line with a discrete time Markov chain is discussed in Appendix A. There we also give a formal definition of average line production rate $V(\mathbf{h})$ and average number of parts $q_j(\mathbf{h})$ in buffer j, $j = 1, \ldots, n$, in stationary mode.

We define the set of admissible solutions for the considered problem of choosing buffer sizes as $D = \{\mathbf{h} | 0 \leqslant h_i \leqslant d_i, i = 1, \ldots, n\}$, where d_i is the maximal admissible size of buffer i, $i = 1, \ldots, n$. In this work we consider the multicriterial optimization problem for volumes of buffers for a series parallel structure of the line with the following criteria:

—average throughput of the line in stationary mode $V(\mathbf{h}) \to \max$;

—capital costs for installing buffers $J(\mathbf{h}) \to \min$, where

$$J(\mathbf{h}) = \sum_{j=1}^{n} J_j(h_j),$$

$J_j(h_j)$ is the cost of installing buffer j with volume h_j;

—the inventory cost for parts in intermediate buffers $Q(\mathbf{h}) \to \min$, where

$$Q(\mathbf{h}) = \sum_{j=1}^{n} w_j q_j(\mathbf{h}),$$

w_j is the storage cost for parts in buffer j.

2.1. Aggregation Schemes for Line Segments

Similar to [4–6, 17, 18] and some other papers we assume in what follows that the number of parts x_j in buffer j is measured not by an integer but by a continuous value from $[0, h_j]$. This simplification is due to the need to reduce computational costs and is based on the observation that only boundary states when the buffer's contents are either 0 or h_j can lead to stopping the equipment. For all other states, the line's throughput does not change regardless of the buffer state in $(0, h_j)$. By passing from discrete states to a continuous ones, we can replace by a single state with real parameter x_j the entire set of internal states with buffer occupancy in the interval $(0, h_j)$.

In this work, as an algorithm for approximate computation of the line's parameters in the steady state we use an algorithm from [4] based on replacing line segments with "equivalent" EUs. Next, approximate values for $V(\mathbf{h})$, $q_1(\mathbf{h}), q_2(\mathbf{h}), \ldots, q_n(\mathbf{h})$ and $Q(\mathbf{h})$ found with this aggregation method are denoted respectively by $V'(\mathbf{h})$, $q'_1(\mathbf{h}), q'_2(\mathbf{h}), \ldots, q'_n(\mathbf{h})$, and $Q'(\mathbf{h})$.

Aggregation procedures have been developed for two kinds of line segments: with two sequential EUs (rule R_1) and with two parallel EUs (rule R_2). During operation, the algorithm sequentially chooses two-machine segments and replaces them with single EUs with "equivalent" parameters until it gets a line consisting of a single EU. Average productivity of this EU yields the value $V'(\mathbf{h})$, and average values of the number of parts in buffers excluded in aggregation by rule R_2 yield the corresponding values $q'_j(\mathbf{h})$.

The accuracy of the above-described heuristic depends on the order in which we aggregate line segments. The efficiency of various rules for choosing an EU pair to apply the aggregation procedure has been studied in [19] in detail. In this case rule R_1 always, whenever possible, has higher priority than rule R_2. In case when we have several alternatives when choosing a pair with sequential EUs, the higher priority goes to a pair with smaller buffer capacity.

Computational experiments [4] with instances from [18] have shown that this aggregation method has deviation inside 5 % of the average values obtained by simulation modeling. By computing confidence intervals for $V(\mathbf{h})$ we have established for some of these examples that the value $V'(\mathbf{h})$ found by aggregation lies in the interval with confidence level 0.99.

2.2. Pareto Front Approximation Algorithms

Suppose that on the set of admissible solutions D we know a criterial vector function $\mathbf{f} = (f_1, \ldots, f_k)$ that takes values in the space of criteria $\mathbf{f}(x) = (f_1(x), \ldots, f_k(x)) \in \mathbb{R}^k$, $x \in D$, where k is the number of criteria. We introduce the Pareto domination relation in the space $\mathbb{R}^k$ for the case when all criteria are supposed to be maximized: vector $\mathbf{f} = \mathbf{f}(x)$, $x \in D$ Pareto dominates vector $\bar{\mathbf{f}} = \mathbf{f}(\bar{x})$, $\bar{x} \in D$, if there is at least one strict inequality among $f_i(x) \geqslant f_i(\bar{x})$, $i = 1, \ldots, k$. In case of criteria that have to be minimized the domination relation in the space of criteria is introduced in a similar way. A solution $x \in D$ dominates solution $\bar{x} \in D$ if the vector $\mathbf{f}(x)$ Pareto dominates vector $\mathbf{f}(\bar{x})$. The set $\tilde{D}$ of all non-dominated admissible solutions is called the set of Pareto optimal solutions (Pareto set). The Pareto front is the set $F := \mathbf{f}(\tilde{D})$.

Since in the general case it appears to be impossible to exactly compute the criteria $V(\mathbf{h})$ and $Q(\mathbf{h})$, as the first step of approximation for the Pareto front we replace the domination relation defined by criteria V, J, and Q, with the domination relation defined by criteria V', J, and Q'. We denote the latter relation by $\succ'$, and the corresponding Pareto front by F'. Further, it follows from the results of [20] that exactly computing F' is an NP-hard problem and may require excessive computational resources. Therefore, we make the second approximation step, which is to find an approximate solution for the problem of finding the Pareto front F' via evolutionary algorithms SIBEA [21] and SEMO [22].

In the operation of the SEMO algorithm (Simple Evolutionary Multiobjective Optimizer), on each iteration we choose a parent individual at random from the population, which contains pairwise non-dominated solutions (individuals). Next, from the parent individual with the mutation operation we obtain an offspring, which we add to the population if it does not contain individuals that dominate the offspring or individuals with the same value of the vector criterion. All individuals in the population that are dominated by this offspring are removed. As a result of this algorithm, we get the population Π computed by the end of its operation.

Algorithm 1 (SEMO).

1. Generate at random solution x and let $\Pi := \{x\}$.

2. While the stopping criterion is not satisfied:

2.1. Choose at random an individual x from population Π.

2.2. Construct an offspring $x' := Mut(x)$ (operator $Mut(x)$ will be defined below).

2.3. Remove dominated individuals from the population, $\Pi := \Pi \backslash \{z \in \Pi | x' \succ' z\}$.

2.4. If $\nexists z \in \Pi$ such that $z \succ' x'$ or $\mathbf{f}(z) = \mathbf{f}(x')$ then $\Pi := \Pi \cup \{x'\}$.

The algorithm's stopping criterion is defined in one of the following ways:

—stop if the maximal number of iterations has been reached;

—stop if the maximal time allotted for this run has been reached.

An individual taken at random from the population is subject to the mutation operator for which the chosen solution is changed at random. In application to the considered problem, the action of a mutation operator is defined by the function $Mut(\mathbf{h}) = (h_1 + \xi_1, \ldots, h_N + \xi_N)$, where ξ_i are integer-valued random values uniformly distributed from $\max\{-h_j, -\Delta\}$ to $\min\{d_j - h_j, \Delta\}$, $j = 1, \ldots, N$. Here Δ is an algorithm parameter that defines the intensity of the mutations.

To define the next algorithm, we will need the definition of *hypervolume* [23, 24]. Let us choose in the space of criteria a *reference point* $\mathbf{r} = (r_1, \ldots, r_k)$ (one usually selects the point $\mathbf{r}$ which is definitely dominated by all points $\mathbf{f}(x)$, $x \in D$). For the set $A \subseteq D$, the hypervolume $I_H(A)$ is defined as follows:

$$I_H(A) = VOL\left(\bigcup_{a \in A} [r_1, f_1(a)] \times \cdots \times [r_k, f_k(a)]\right),$$

where $VOL(\cdot)$ denotes the Lebesgue measure. In case when criteria are supposed to be minimized, the definition of hypervolume is modified in an obvious way.

The $(\mu + 1)$-SIBEA algorithm is based on the Simple Indicator-Based Evolutionary Algorithm (SIBEA) proposed in [24]. During the execution of this algorithm, on each iteration we add to population Π of size μ an offspring x created from the parent individual chosen at random. Then, we remove from the population an individual whose contribution to the value of the population's hypervolume according to criteria V', J, and Q' is minimal. As the reference point we have chosen the point $\mathbf{r} = (0, J(d_1, \ldots, d_n), \sum_{j=1}^{n} w_j d_j)$. The algorithm's stopping criterion is defined in the same way as in the SEMO algorithm.

Algorithm 2 (the scheme of $(\mu + 1)$-SIBEA).

1. Generate at random a population Π of size μ.

2. While the stopping criterion is not satisfied:

2.1. Choose at random an individual x from population Π.

2.2 Construct an offspring $x' := Mut(x)$ and add it to the population $\Pi' := \Pi \cup \{x'\}$ (operator $Mut(x)$ has been considered above).

2.3. Find $d(x) := I_H(\Pi') - I_H(\Pi' \backslash \{x\})$ for each individual $x \in \Pi'$.

2.4. Choose an individual $z \in \Pi'$ such that $d(z) = \min_{x \in \Pi'} d(x)$. If there are several such individuals, choose one of them at random.

2.5. Remove the selected individual from the population, $\Pi := \Pi' \backslash \{z\}$.

Similar to the SEMO algorithm, the value of the entire population's hypervolume does not decrease as the number of iterations grows, but unlike SEMO the population size in the SIBEA algorithm remains constant.

The SIBEA algorithm aims to approximate the Pareto set with a bounded number of points. It is important to bound the population size if we want to approximate the Pareto optimal set with a comparatively small number of solutions.

3. COMPUTATIONAL EXPERIMENT

SEMO and SIBEA have been implemented in the Visual Studio 2010 integrated development environment. In order to further diversify the population, we also consider a modification of the SEMO algorithm where the initial population consists of 1000 individuals generated at random. In what follows we denote this algorithm by SEMO+. All experiments have been done on an Intel Core i5 desktop computer (2,4 GHz CPU, 4 Gb of RAM). The mutation intensity was set to $\Delta = 2$.

In the experiments, we used a series of problems AS from [18] and a problem P_0 of small dimension. The P_0 problem is constructed based on a line with a sequential structure with three intermediate buffers and four EUs, where $d_1 = d_2 = d_3 = 20$, $T_1^{\mathrm{F}} = T_4^{\mathrm{F}} = 30$, $T_2^{\mathrm{F}} = T_3^{\mathrm{F}} = 15$, $T_1^{\mathrm{R}} = T_4^{\mathrm{R}} = 200$, $T_2^{\mathrm{R}} = T_3^{\mathrm{R}} = 100$, $U_1 = U_2 = U_3 = 3000$. The AS series consists of problems constructed based on flow lines no. 1, 2, 6, 7, 8 from [18] with real data from the Renault industrial facilities. Test example no. 3 from [18] was not used since it demonstrates a special case when approximate computation of the line's throughput V' significantly differs from the results of simulation modeling. Besides, it was noted in [19] that examples of this kind are not realistic.

Lines nos. 4 and 5 from [18] are not used due to their low dimension. Parameters of the other lines are given in Appendix C. The storage costs for parts w_j, $j = 1, \ldots, n$, were assumed to be unit in all problems, $J_j(h_j) \equiv h_j$, $j = 1, \ldots, n$.

The small size of the space of solutions for problem P_0 has let us fully enumerate it with a simple lexicographic enumeration algorithm. Based on these results, we have evaluated the speed of finding the solutions from the Pareto front F' depending on the number of generated sample points in the SEMO and SIBEA algorithms (by a sample point we mean a solution for which we compute the values of the criteria). In the experiments, the operation time of the algorithms was limited to one hour.

Figure 1 demonstrates the process of the algorithms' operation, where the horizontal axis shows the number of elements found in the Pareto front, and the vertical axis shows the number of generated sample points. Figure 1 indicates that algorithm SEMO wins over SIBEA in terms of the number of found elements from the Pareto front F'. Similar results also follow for the AS1 problem, where the space of solutions could also be exhaustively studied with lexicographic enumeration. Besides, Fig. 1 illustrates a situation when on a low dimensional problem it is better to use lexicographic enumeration than evolutionary algorithms. For the problems with larger dimension, such as experimentally support solutions such as AS2, AS6, and AS7, the complete enumeration of all solutions in reasonable time does not appear to be possible.

As we have already mentioned above, to compare the algorithms on a series of problems AS, each algorithm got one hour of operation, and after this hour we computed the hypervolume of the resulting population. The size of the initial population for the SIBEA algorithm was equal to the number of non-dominated solutions found by the lexicographic enumeration algorithm (pos-

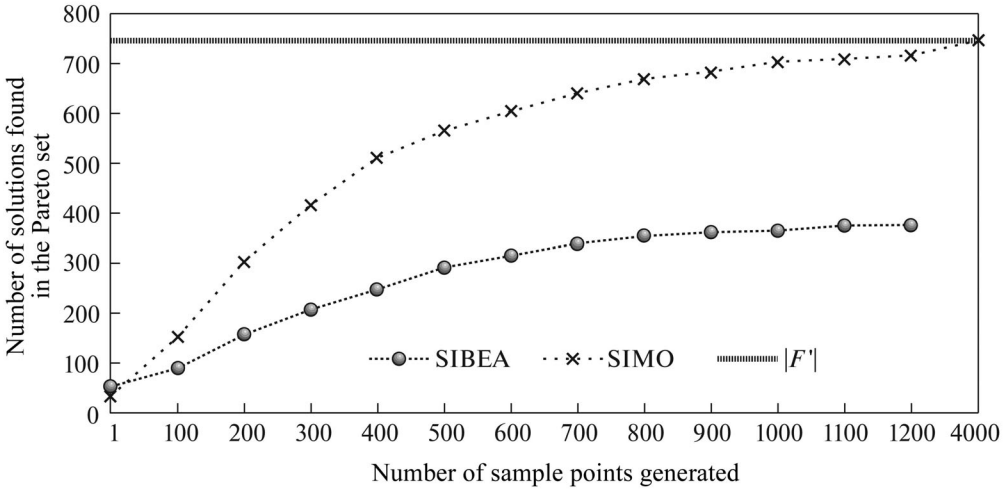

Fig. 1. Number of found points from F' in the algorithms' operation. Problem P_0.

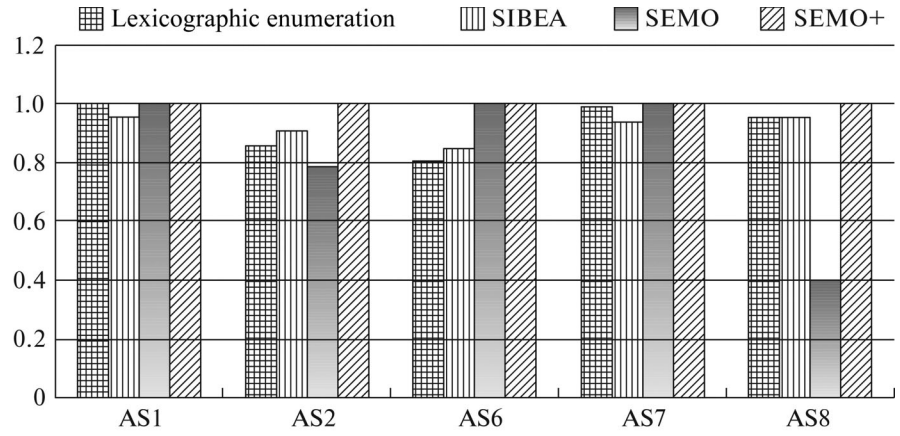

Fig. 2. Relative value of the hypervolume.

sibly incomplete) over the same time period of one hour. In one hour of computation, complete enumeration of the solution space stopped and resulted in the full set F' only in the problem AS1. Enumeration was done in increasing lexicographic order $\succ_L$, where $\mathbf{h} \succ_L \mathbf{h}'$ if for some k it holds that $h_k > h'_k$, $h_{k+1} = h'_{k+1}, \ldots, h_n = h'_n$. Figure 2 demonstrates the results of the algorithms' operation on the entire series of problems AS. The number of solutions enumerated lexicographically, size of the set D, and the number of sample solutions computed in algorithms SIBEA, SEMO, and SEMO+, are shown in Table 1. The horizontal axis in Fig. 2 shows the ratio of the hypervolume of the population obtained by each algorithm to the best value of the hypervolume found in our

Table 1. Problem dimensions and number of sample solutions computed in one hour

	n	$\lvert D\rvert$	Lexicographic enumeration	SIBEA	SEMO	SEMO+
AS1	4	4×10^5	1.3×10^5	1.4×10^3	8.6×10^6	8.7×10^6
AS2	9	9×10^{11}	1.3×10^5	2.1×10^3	1.4×10^6	9.6×10^5
AS6	13	2×10^{22}	1.3×10^5	1.2×10^4	8.8×10^6	7.8×10^6
AS7	7	9×10^7	1.3×10^5	2.2×10^4	1.7×10^7	1.6×10^7
AS8	7	5×10^8	1.3×10^5	7.7×10^3	1.2×10^7	1.5×10^7

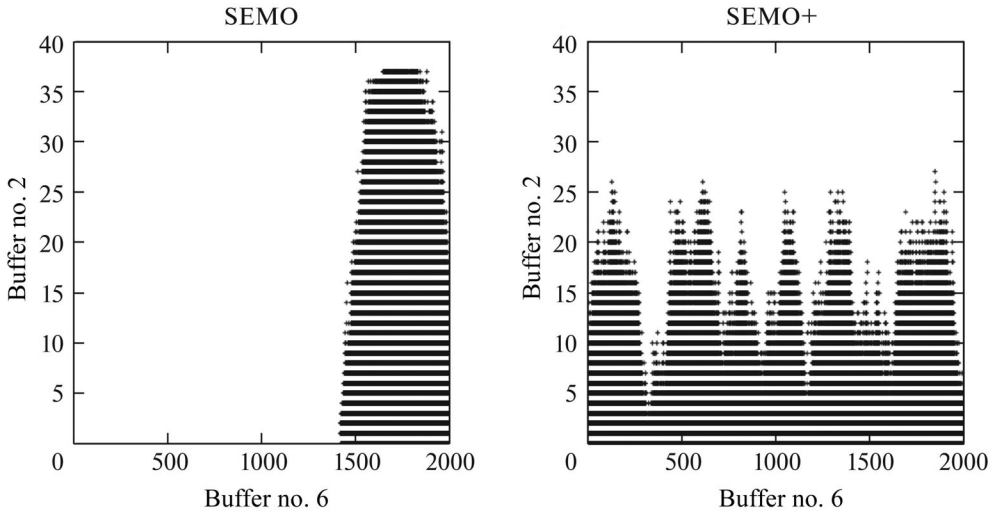

Fig. 3. Populations obtained by algorithms SEMO and SEMO+ for problem AS6.

experiments for this problem in one hour. The graph indicates that problem AS1 proved to be easy for all algorithms; on problems AS2 and AS8 the value of the population hypervolume obtained by SIBEA is larger than for SEMO; on problems AS6 and AS7, the SEMO algorithm has the upper hand.

We have established that on problems AS2 and AS8 the population in algorithm SEMO is localized in a comparatively small subset of the space of solutions. To find the reason for the localization, we have obtained the entire set F' for problem AS8 with complete lexicographic enumeration (which took more than six hours). We have established that for the precision of comparing the values $V'(\mathbf{h})$ and $Q'(\mathbf{h})$ around 10^{-15}, the preimage of the Pareto front F' is the set of solutions of the form $(0,0,0,0,0,h_6,0)$, where $h_6 = 0,\ldots,2000$.

Figure 3 shows on the left the values of buffers 2 and 6 for individuals from the population constructed by the SEMO algorithm. The figure indicates that the population of the SEMO algorithm also does not have enough diversity to get the entire set F'. As the experiment with SEMO+ showed, the population of this algorithm was not localized (see Fig. 3 on the right), as it did for SEMO, which led to a better approximation of the Pareto front F'.

To perform statistical comparison of the operation of algorithms SEMO and SEMO+ on the problem AS8, we have run each of them 30 times until we obtained the entire Pareto front F'. The average number of iterations that the SEMO+ algorithm needs to find the entire set F' was 3×10^5, and for the SEMO algorithm this value was equal to 3×10^6. By the results of these experiments we constructed two 30-element samples where we observed the number of sample solutions until the entire set F' was constructed on each of the 30 runs of the algorithm. The nonparametric Mann–Whitney criterion applied to compare these samples showed that the differences were statistically significant at level $p < 0.01$.

Thus, for the AS8 problem the SEMO+ algorithm finds a Pareto front F' faster than the SEMO algorithm. In general, for the entire series of problems AS the values of population hypervolumes obtained by the SEMO+ algorithm have also proven to be maximal, as shown in Fig. 2.

4. CONCLUSION

Our experiments have shown that if the number of points in our approximation of a Pareto front is not bounded from above by some predefined value, then the SEMO+ algorithm has an advantage over the SEMO and SIBEA algorithms with regard to the hypervolume of the resulting

set of solutions. However, if the approximating set must consist of a relatively small number of solutions then the SEMO and SEMO+ algorithms lose to the SIBEA algorithm.

In practice, the decision maker can analyze only a comparatively small number of "promising" variations of the buffer distribution, so the best approach for a decision maker appears to be to use the set of solutions found by the SIBEA algorithm. The accuracy of our approximation for the Pareto front in this case can be approximately estimated by comparing the hypervolumes of populations constructed by SIBEA and SEMO+.

ACKNOWLEDGMENTS

This work was supported by the Russian Science Foundation, project no. 15-11-10009.

The authors are grateful to V.A. Topchii and anonymous referees for a number of valuable remarks in the preparation of this work.

APPENDIX A

This Appendix describes the Markov model for a production line with series-parallel structure as shown in Section 2.

The states of the line in question are defined by a set of parameters $(\alpha_1, \ldots, \alpha_m, x_1, \ldots, x_n, y_1, \ldots, y_m)$. Here $\alpha_i = 0$ if EU number i is in the "failure" state, and $\alpha_i = 1$ otherwise. The value $x_j \in \{0, 1, \ldots, h_j\}$ determines the number of parts in buffer j. The value $y_i \in \{0, 1, \ldots, U_i\}$ is the number of units of time already spent to process an existing part on EU number i. The value y_i is zero if the EU is in the "idle" state. When an EU is in the "blocked" state, the value of y_i remains unchanged and equal to U_i. The set of all states is

$$S = \{0,1\}^m \times \Pi_{j=1}^n \{0, 1, \ldots, h_j\} \times \Pi_{i=1}^m \{0, 1, \ldots, U_i\}.$$

The system in question is described with a Markov chain M with discrete time and set of states S. With nonzero probability, in $\sum_{i=1}^m U_i(m + \sum_{j=1}^n h_j)$ transitions we can, starting from any state, reach the state $(0, \ldots, 0)$ where all buffers are empty, there are no parts under processing, and all machines are in the "failure" state. Properties of Markov chains (see, e.g., [25], Chapter V, Section 2) imply the following proposition.

Proposition 1. *A Markov chain M has stationary distribution π to which the probability distribution of the states of the line tend for $t \to \infty$ from any initial state.*

The average line throughput in the steady state is $\sum_{i:a_i=(b_j,b_{n+1})} \pi(S_i)/U_i$, where summation is done over the set of arcs incoming into the vertex b_{n+1}, and S_i is the set of states of the system where $\alpha_i = 1$ and $y_i = U_i$.

APPENDIX B

Let us describe a simplified Markov model for a production line in the case of two sequential EUs proposed in [6] (similar models were considered in [5, 26]) together with aggregation rules for pairs of sequential and parallel EUs that let one use this model for lines with series-parallel structure and arbitrary complexity.

The approximation to the model described in Section 2 and Appendix A is that the occupied volume of the buffer and time parameter are assumed to be real, and the geometric distributions of transition probabilities are replaced by exponential distributions with the same expectations. The failure intensity for the ith EU equals $\lambda_i = 1/T_i^{\mathrm{F}}$, the repair intensity is $\mu_i = 1/T_i^{\mathrm{R}}$. We denote the rate of processing for the parts on the ith EU by c_i, $c_i = 1/U_i$.

Let h be the volume of the buffer between EUs. We consider two kinds of states: the set of internal states $S_{\text{int}} = \{(\alpha_1, \alpha_2, x) : \alpha_1, \alpha_2 \in \{0,1\}, x \in (0,h)\}$ and the set of boundary states $S_{\text{fr}} = \{(\alpha_1, \alpha_2, x) : \alpha_1, \alpha_2 \in \{0,1\}, x \in \{0,h\}\}$, where α_i have the same meaning as in Appendix A. Let Boolean random values A_i correspond to the values α_i, and let a real random variable X correspond to the number of parts in the buffer. We denote the probabilities of states from S_{fr} at time moment t by

$$P_{\alpha_1,\alpha_2}(0,t) := \mathbf{P}\{(A_1, A_2, X) = (\alpha_1, \alpha_2, 0) \text{ at time moment } t\}$$

and

$$P_{\alpha_1,\alpha_2}(h,t) := \mathbf{P}\{(A_1, A_2, X) = (\alpha_1, \alpha_2, h) \text{ at time moment } t\}.$$

We denote by $F_{\alpha_1',\alpha_2'}(x,t)$ the probability of the event that at time moment t random variable (A_1, A_2, X) equals (α_1, α_2, x'), where $x' < x$, and assume that there exist derivatives $f_{\alpha_1,\alpha_2}(x,t) = \partial F_{\alpha_1,\alpha_2}(x,t)/\partial x$.

The stationary distribution for the Markov process that models the system in question, as found in [6], is defined by limit values

$$P_{\alpha_1,\alpha_2}(0) = \lim_{t\to\infty} P_{\alpha_1,\alpha_2}(0,t),$$
$$P_{\alpha_1,\alpha_2}(h) = \lim_{t\to\infty} P_{\alpha_1,\alpha_2}(h,t),$$
$$f_{\alpha_1,\alpha_2}(x) = \lim_{t\to\infty} f_{\alpha_1,\alpha_2}(x,t).$$

Here internal states are mutually related by the system of equations

$$0 = \lambda_1 f_{10}(x) + \lambda_2 f_{01}(x) - (\mu_1 + \mu_2) f_{00}(x),$$
$$-c_2 \frac{\partial f_{01}(x)}{\partial x} = \lambda_1 f_{11}(x) + \mu_2 f_{00}(x) - (\mu_1 + \lambda_2) f_{01}(x),$$
$$c_1 \frac{\partial f_{10}(x)}{\partial x} = \lambda_2 f_{11}(x) + \mu_1 f_{00}(x) - (\lambda_1 + \mu_2) f_{10}(x),$$
$$(c_1 - c_2) \frac{\partial f_{11}(x)}{\partial x} = \mu_1 f_{01}(x) + \mu_2 f_{10}(x) - (\lambda_1 + \lambda_2) f_{11}(x),$$

and the systems of equations for boundary states depend on the relation between c_1 and c_2: $c_1 < c_2$, $c_1 > c_2$, or $c_1 = c_2 = c$. As an example we show the case when $c_1 = c_2 = c$:

$$P_{10}(0) = P_{00}(0) = P_{00}(h) = P_{01}(h) = 0,$$
$$\mu_2 P_{10}(h) = \lambda_2 P_{11}(h) + c f_{10}(h) = (\lambda_1 + \lambda_2) P_{11}(h),$$
$$\mu_1 P_{01}(h) = \lambda_1 P_{11}(0) + c f_{01}(0) = (\lambda_1 + \lambda_2) P_{11}(0),$$
$$c f_{01}(h) = \lambda_1 P_{11}(h), \quad c f_{10}(0) = \lambda_2 P_{11}(0).$$

A solution of this system of equations yields expressions for $f_{\alpha_1,\alpha_2}(x)$, $P_{\alpha_1,\alpha_2}(0)$, and $P_{\alpha_1,\alpha_2}(h)$.

Rule R_1. Based on the above model and the resulting probabilities of its states in the steady state, we can find an "equivalent" EU that has similar characteristics in the steady state and can replace two EUs and the buffer between them. For this purpose, we introduce the failure state for a system of two EUs and a buffer between them. We define this state from the side of the less productive EU since it is idle less often and the computational errors will be smaller. By a failure we will mean a state when the less productive EU is not in the "operational" state.

A EU that replaces two consecutive EUs has parameters λ', μ', and c'. Intensities λ' and μ' are computed based on the transition intensities between "failure" and "operational" states for the considered two-machine subsystem that accounts for the stationary probability distribution for two sequential machines and the buffer between them [4]. For example, for $c_2 > c_1$ we have

$$c' = c_1, \quad \lambda' := \lambda_1 + \frac{P_{10}(h)\mu_2}{F_{11}(h) + F_{10}(h) + P_{11}(0)},$$
$$\mu' := \mu_1 + \frac{P_{10}(h)(\mu_2 - \mu_1)}{P_{10}(h) + P_{01}(0) + F_{01}(h) + F_{00}(h)}.$$

The average number of parts in a buffer in stationary mode has the form [6]

$$q = \sum_{\alpha_1,\alpha_2 \in \{0,1\}} \left(\int_0^h x f_{\alpha_1\alpha_2}(x)dx + hP_{\alpha_1\alpha_2}(h) \right).$$

Rule R_2. Parameters of an EU that replaces two parallel EUs [18] are

$$\lambda' := \lambda_1 \frac{\mu_2}{\mu_2 + \lambda_2} + \lambda_2 \frac{\mu_1}{\mu_1 + \lambda_1}, \quad c' := c_1 + c_2,$$
$$\mu' := \frac{\lambda^*}{(c_1 + c_2)/v' - 1}, \quad \text{where} \quad v' = \frac{c_1}{1 + \lambda_1/\mu_1} + \frac{c_2}{1 + \lambda_2/\mu_2}.$$

APPENDIX C

Let us show the description of lines constructed based on examples from [18]. In problems AS1, AS2 and AS6 the lines have a serial structure. Their parameters are shown in Tables 2 and 3.

In problems AS7, AS8 the lines contain 10 EUs and have a series-parallel structure. Their schemata are shown in Fig. 4. Parameters for the AS7 and AS8 problems are shown in Table 4.

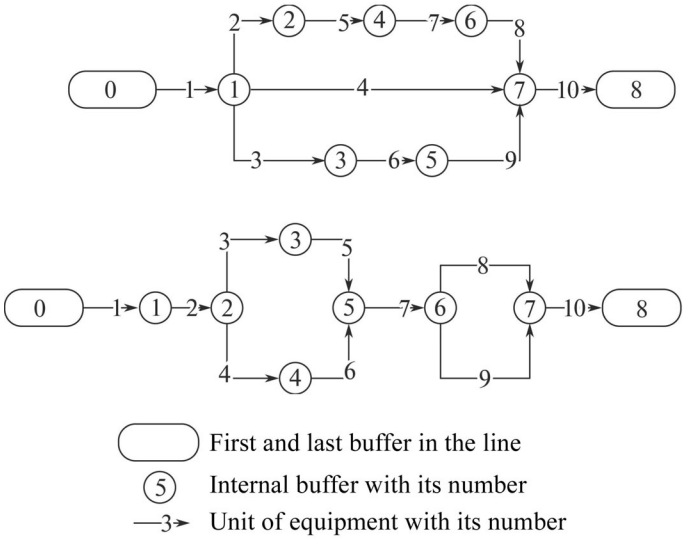

Fig. 4. Graphs of the lines in examples AS7 (top) and AS8 (bottom).

Table 2. Parameters of problem AS1

Buffers		EU			
i	d_i	j	T_j^{F}	T_j^{R}	U_i
1	20	1	244.2	150	10
2	17	2	255.3	300	10
3	38	3	176	75	10
4	48	4	184	600	10
		5	192	450	10

Table 3. Parameters of problems AS2 and AS6

AS2						AS6					
Buffers		EU				Buffers		EU			
i	d_i	j	T_j^{F}	T_j^{R}	U_i	i	d_i	j	T_j^{F}	T_j^{R}	U_i
1	0	1	10 000	440	22	1	60	1	29 880	22 000	385
2	50	2	20 000	440	23	2	60	2	29 880	22 000	426
3	20	3	5000	430	22	3	50	3	876 000	22 300	330
4	50	4	40 000	520	23	4	70	4	29 880	22 000	372
5	0	5	30 000	430	24	5	60	5	33 250	27 500	316
6	80	6	2442	440	22	6	80	6	144 000	8500	340
7	20	7	1840	520	23	7	45	7	102 300	74 000	340
8	100	8	1680	430	21	8	25	8	113 300	7200	340
9	100	9	2208	920	24	9	35	9	540 000	60 000	380
		10	2464	780	22	10	80	10	538 800	349 000	350
						11	40	11	5 064 000	73 700	400
						12	45	12	468 000	306 000	400
						13	65	13	1 032 000	54 000	319
								14	45 600	31 120	319

Table 4. Parameters of problems AS7 and AS8

AS7						AS8					
Buffers		EU				Buffers		EU			
i	d_i	j	T_j^{F}	T_j^{R}	U_i	i	d_i	j	T_j^{F}	T_j^{R}	U_i
1	15	1	50 000	12 000	1000	1	1300	1	87 000	27 000	23
2	10	2	48 000	2000	3450	2	200	2	77 000	22 000	27
3	15	3	55 000	9000	2780	3	0	3	580 000	18 000	38
4	10	4	39 000	6000	3030	4	0	4	410 000	12 500	30
5	25	5	75 000	10 000	3333	5	0	5	580 000	18 000	38
6	10	6	59 000	11 000	2560	6	2000	6	410 000	12 500	30
7	10	7	28 000	8000	3030	7	0	7	725 000	21 000	20
		8	35 000	8000	3125			8	550 000	14 000	40
		9	65 000	35 000	2174			9	430 000	24 000	43
		10	20 000	4000	800			10	270 000	22 000	33

REFERENCES

1. Tempelmeier, H., Practical Considerations in the Optimization of Flow Production Systems, *Int. J. Product. Res.*, 2003, vol. 41, no. 1, pp. 149–170.

2. Patchong, A., Lemoine, T., and Kern, G., Improving Car Body Production at PSA Peugeot Citroen, *Interfaces*, 2003, vol. 33, no. 1, pp. 36–49.

3. Dallery, Y. and Gershwin, S.B., Manufacturing Flow Line Systems: A Review of Models and Analytical Results, *Queueing Syst.*, 1992, vol. 12, nos. 1–2, pp. 3–94.

4. Dolgui, A.B. and Svirin, Yu.P., Models for Estimating the Probabilistic Productivity of Automated Industrial Complexes, *Vestn. Akad. Nauk Belarusi*, 1995, no. 1, pp. 59–67.

5. Levin, A.A. and Pas'ko, N.I., Computing the Productivity of Automated Lines, *Stanki Instrum.*, 1969, no. 8, pp. 8–10.

6. Dubois, D. and Forestier, J.-P., Productivité et en cours moyen d'un ensemble de deux machines séparées par une zone de stockage, *RAIRO Automat.*, 1982, vol. 16, no. 2, pp. 105–132.

7. Li, J. and Meerkov, S.M., *Production Systems Engineering*, New York: Springer, 2009.

8. Altiparmak, A., Bugak, A., and Dengiz, B., Optimization of Buffer Sizes in Assembly Systems Using Intelligent Techniques, *Proc. 2002 Winter Simulat. Conf.*, 2002, pp. 1157–1162.

9. D'Souza, K. and Khator, S., System Reconfiguration to Avoid Deadlocks in Automated Manufacturing Systems, *Comput. Indust. Eng.*, 1997, vol. 32, pp. 445–465.

10. Hamada, M., Martz, H., Berg, E., and Koehler, A., Optimizing the Product-Based Avaibility of a Buffered Industrial Process, *Reliab. Eng. Syst. Safety*, 2006, vol. 91, pp. 1039–1048.

11. Abdul-Kader, W., Capacity Improvement of an Unreliable Production Line—An Analytical Approach, *Comput. Oper. Res.*, 2006, vol. 33, pp. 1695–1712.

12. Dolgui, A., Eremeev, A., Kolokolov, A., and Sigaev, V., A Genetic Algorithm for the Allocation of Buffer Storage Capacities in a Production Line with Unreliable Machines, *J. Math. Modeling Algorithms*, 2002, vol. 1, pp. 89–104.

13. Chehade, H., Yalaoui, F., Amodeo, L., and De Guglielmo, P., Optimisation multiobjectif pour le problème de dimensionnement de buffers, *J. Decision Syst.*, 2009, vol. 18, pp. 257–287.

14. Zitzler, E., Laumanns, M., and Thiele, L., SPEA2: Improving the Strength Pareto Evolutionary Algorithm, *Technic. Report 103*, Comput. Eng. Commun. Networks Lab, Swiss Federal Institute Technol., Zurich, 2001.

15. Deb, K., Pratap, A., Agarwal, S., and Meyarivan, T., A Fast and Elitist Multiobjective Genetic Algorithm: NSGA-II, *IEEE Trans. Evolut. Comput.*, 2002, vol. 6, no. 2, pp. 182–197.

16. Cruz, F.R.B., Van Woensel, T., and Smith, J.M., Buffer and Throughput Trade-Offs in M/G/1/K queuing networks: A Bicriteria Approach, *Int. J. Product. Econom.*, 2010, vol. 125, pp. 224–234.

17. Sevast'yanov, B.A., The Problem of How Bunker Capacity Influences Averages Idle Time for an Automated Line of Machines, *Teor. Veroyat. Primen.*, 1962, vol. 7, no. 4, pp. 438–447.

18. Ancelin, B. and Semery, A., Calcul de la productivité d'une ligne integrée de fabrication, *RAIRO Autom., Productiq. Inform. Industrielle*, 1987, vol. 21, pp. 209–238.

19. Terracol, C. and David, R., Performance d'une ligne composée de machines et de stocks intermédiaires, *RAIRO Automatiq., Productiq. Informatiq. Industrielle*, 1987, vol. 21, pp. 239–262.

20. Dolgui, A., Eremeev, A., Kovalyov, M.Y., and Sigaev, V., Complexity of Buffer Capacity Allocation Problems for Production Lines with Unreliable Machines, *J. Math. Modell. Algorithms*, 2013, vol. 12, pp. 155–165.

21. Brockhoff, D., Friedrich, T., and Neumann, F., Analyzing Hypervolume Indicator Based Algorithms, *Proc. Parallel Probl. Solving from Nature—PPSN X: 10th Int. Conf. 2008*, Berlin: Springer, 2008, vol. 5199, pp. 651–660.

22. Laumanns, M., Thiele, L., Zitzler, E., Welzl, E., and Deb, K., Running Time Analysis of a Multi-Objective Evolutionary Algorithm on a Simple Discrete Optimization Problem, *Parallel Probl. Solving from Nature, 2002*, Berlin: Springer, 2002, vol. 2439, pp. 44–53.

23. Belous, V.V., Groshev, S.V., Karpenko, A.P., and Shibitov, I.A., Software Systems for Evaluating the Quality of Pareto Approximations in Multicriterial Optimization Problems. A Survey, *Nauka i Obrazovanie: FGBOU VPO "MGTU im. N.E. Baumana,"* 2014, El. no. FS 77–48211, pp. 300–320.

24. Zitzler, E., Brockhoff, D., and Thiele, L., The Hypervolume Indicator Revisited: On the Design of Pareto-Compliant Indicators via Weighted Integration, *Proc. Conf. Evolut. Multi-Criter. Optim. (EMO 2007)*, LNCS, Berlin: Springer, 2007, vol. 4403, pp. 862–876.

25. Doob, J.L., *Stochastic Processes*, New York: Wiley, 1953. Translated under the title *Veroyatnostnye protsessy*, Moscow: Inostrannaya Literatura, 1956.

26. Gershwin, S.B. and Schick, I.C., *Continuous Model of an Unreliable Two-Stage Material Flow System with a Finite Interstage Buffer*, Report LIDS-R-1039, Massachusetts Inst. of Technology, Cambridge, 1980.

This paper was recommended for publication by O.N. Granichin, a member of the Editorial Board