GENETIC ALGORITHMS IN COMPUTER-AIDED DESIGN OF ROBOTIC MANUFACTURING CELLS

Anatoly Pashkevich

Robotic Laboratory,
Department of Automatic Control
Belarusian State University of Informatics and Radioelectronics
6 P.Brovka St., Minsk, 220600,Belarus
e-mail: pap@gw.bsuir.unibel.by

Abstract: Computer-aided design of modern robotic manufacturing cells involves solution of a number of complicated optimization problems that are related to equipment selection, scheduling, routing and path planning. The paper deals with three of them: optimal selection of robotic tools, optimal clustering of welding dots among robots, and optimal layout design. A number of efficient genetic algorithms are proposed for these specific integer optimization problems. They have been incorporated in commercial software package ROBOMAX that is widely used in Russian automotive industry. The efficiency of the proposed algorithms is carefully investigated via computer simulation; computational results indicate that the algorithms are effective in solving problems of practical size.

Keywords: robotics, integer programming, computer-aided design.

1. INTRODUCTION

Industrial application of robots requires development of special algorithms and software tools that are aimed at shortening of the design-to-manufacturing cycle. In automotive industry, for instance, changing a car model is a very expensive and time-consuming process because robots have to be reprogrammed to different welding patterns. The traditional approach, based on manual teaching, is unsuitable for modern applications since the whole manufacturing line has to be stopped. An alternative way is using off-line programming incorporated in robotic CAD systems that enables user to generate robot-level programs while the production line is still in action (McKerrow, 1991).

Commercially available packages are mainly based on interactive design that involves an experienced user for decision making, while the software provides sophisticated 3D geometrical modelling only and allows to estimate an acceptability of a user-proposed solution. Therefore, automatic generation of acceptable solutions is a challenging problem for CAD of industrial robotic systems.

This paper focuses on combinatorial optimization problems that arise in computer-aided design of robotic cells for spot welding applications. In contrast to other works devoted to general FMS design (Heragu, 1992; Stecke, 1983), it takes into account the essential features of the robotic technology. Selected aspects of these problems have been previously discussed by a number of authors. In particular, Cook and Han (1994) have used combinatorial optimization algorithms for robot selection and work station assignment which is treated as a two-dimensional multi-type bin packing problem. Because of NP-hard nature of this problem, an efficient tree-phase heuristic algorithm has been developed. Souilah, Mecheri and Bennesroune (1996) have also investigated intra-cell layout design problem and have proposed a solution based on simulated annealing (Laarhoven and Aarts, 1987) and taboo search (Glover, et al., 1993). However, in spite of the good computational properties, the previously

developed algorithms deal with the simplified models of robotic cells and can not be directly applied in industrial robotic CAD systems, while the proposed approach relies on exact 3D models of the cells. Besides, though the paper is addressed to industrial robotics mainly, its results can be also applied to some problems that arise in service and personal robot control, in particular for motion planning of mobile robots, scheduling and geometric reasoning.

2. DECOMPOSITION OF THE DESIGN PROCESS

The general problem of the robotic cell design is usually split into a number of separate stages which ensure selection of an optimal set of robots, a set of tools, fixture and their optimal placement (Pashkevich et al., 1998). Besides, it is necessary to optimize robot motions and to develop technological programs in a specific robot control language.

As observed in practice, optimal tool selection (or design) is performed first. The desired set of tools should enable robots to serve all given welding dots such that no technological constraints are violated and the total working time is minimized. Using these results, clusterization of welding dots between tools (robots) is performed. The main constraint here is the accessibility of all given dots by welding guns, the typical objective is minimization of the total working cycle. Subsequently, industrial robots are selected and their optimal location is determined. The last step is creating programs in specific robot control language.

Although this paper concentrates on the first and the second steps of the general design process, the developed algorithms can be also applied for robots selection and their optimal placement. Besides, after insignificant modification, the algorithms can be used for the design of robotic cells for arc welding, laser and plasma cutting.

3. OPTIMAL SELECTION OF WELDING TOOLS

For the whole production line, there can be about a thousand of welding dots, and several dozens of different tools are required. No doubt, it is more attractive in practice to select the tools from the available set instead of designing new ones. This leads to the following *optimization problem*: to select a subset of the given set of tools provided that for each welding dot there exists a tool which can access it and the "price" of the subset is minimal (Fig. 1).

The mathematical interpretation of the problem is the following. Let *Tools* be the set of available tools, and *Dots* be the set of welding dots. Let also S be a binary incidence matrix such that S_{ij} is equal to 1 if the jth tool can serve the welding dot i and is equal to 0 otherwise. In addition, let us assume that an abstract price c_j of every spotgun j is specified. Then the problem is to find such a set of tools $Tools^0$ that:

$$\begin{aligned} &\forall d \in Dots \quad \exists t \in Tools^0, \quad s\|d,t\| = 1; \\ &Tools^0 \subset Tools; \quad N(Tools^0) \to min, \end{aligned} \tag{1}$$

where $N(.)$ is a set capacity.

Let us also denote jth element of *Tools* and ith element of *Dots* as $Tool_i$ and Dot_j respectively. Besides, for easy reference, let a partial solution be any subset of *Tools* such that it does not ensure accessibility of all welding dots. For the described problem, serving of welding dots by tools can be also interpreted as “set covering”. This combinatorial problem has been proven NP-hard (Papadimitriou and Steiglitz, 1982; Christofides, 1975, Quagliarella et. al., 1998) and has attracted attention of numerous researchers. They have examined the performance of approximation algorithms, heuristics and exhaustive search methods.

In this paper, a *two-phase optimization algorithm* is developed for solving this specific set covering problem: 1) Decrease the problem complexity by checking trivial cases and transforming the matrix S into a block form; 2) Use a branch and bound method based on minimax heuristics and a specially developed set of going-back rules. Algorithm details are presented below.

The phase 1 prepares the input data for the branch and bound method. On its step 1 the following operations are performed:
1) if there exists a tool such that no welding dot can be served by it, then the corresponding column is marked as “useless”;
2) if there exist several tools with identical matrix rows, then only "the cheapest" of them is left;
3) if there exists a row with all zeroes in it, than it is eliminated from the matrix (the corresponding welding dot is left for manual welding);
4) if there can be found two rows such that the first row “takes up” the second (i.e. all tools that can serve the first welding dot can also serve the second one) than only the first row is left.

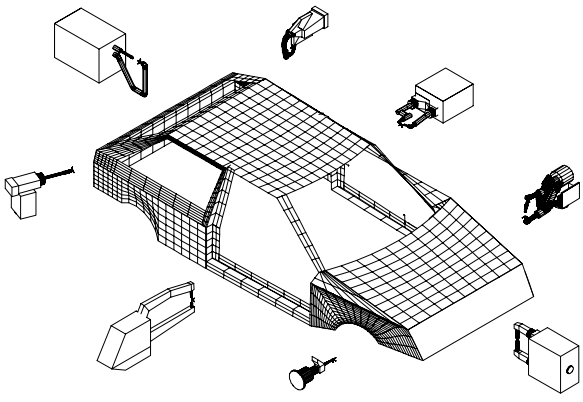

Fig. 1. Selection of spot welding guns.

The main objective of the step 2 is to restructure the data in order to increase the efficiency of the branch and bound method. Here a "greedy" approach is used. For each row, the number of nonzero elements is calculated and the rows are reorganised in the nondecreasing order. Subsequently, column blocks are created so that any column from block *i* has 1 in the *i*th row and does not have 1's in the rows 1, ..., *i*-1. The rows in the blocks are sorted in the nondecreasing order according to the number of welding dots which they can serve and their prices c_j. The complexity of the procedure is $O(n \log n)$.

The phase two is implemented as a recursive procedure and includes the following operations. For the current welding dot *k* (which has the minimum index from the dots which are not covered yet), a set of tools *ToolList* that can access the dot and are not used yet is determined. Afterward, these tools are sorted in the nondecreasing order according to the number of welding dots they can serve and their prices c_j . Than, variants of using tools $ToolList_1$, $ToolList_2$, ... are checked. This approach increases the probability of finding good solutions on the early algorithm steps. Additionally, two lower bound determination methods are used.

Additionally, two lower bound determination methods are used (Liao and Devadas, 1996). Let $\Psi(S)$ be the lower bound for the partial solution *S*. Then, if the following equation is true, there is no need to proceed with S:

$$\sum_{i=1}^{N(S)} c_{S_i} + \Psi(S) \; > \sum_{i=1}^{N(S^0)} c_{S_i^0}. \qquad (2)$$

Here S^o is the best tools grouping that has been already attained. The simple lower bound can be obtained as

$$\Psi(S) = U(S) \cdot c_{j_0} / N_{j_0}, \qquad (3)$$

where j_0 is the tool that can serve the maximum number N_{j_0} of welding dots that are not covered yet, and $U(S)$ is a number of uncovered dots.

The sophisticated method of lower bound determination is the following. Let $SubTools_i$ be a set of tools that are not used yet and can serve the dot $U(S)_i$. Then, if D_{iu} is the maximum number of welding dots that can be covered using tool sets $SubTools_1$, $SubTools_2$, ..., with the cost less than *u*, then the lower bound can be found as:

$$\Psi(S) = u,\ D\left(N\left(U(S)\right)+1, u\right) \ge N\left(U(S)\right),$$
$$D\left(N\left(U(S)\right)+1, v\right) < N\left(U(S)\right), \qquad (4)$$
$$v = 1, \ldots, u-1.$$

The matrix *D* is defined using dynamical programming (Christofides, 1975).

The proposed algorithm enables a user to generate a given number of the best tool sets and to chose a single solution in interactive mode using Pareto-opimisation technique. The latter is described in details in (Pashevich, 1996). It should be noted, that the developed algorithm is rather sensitive to the tool price assignments. It is obvious, that prices should be integer and not differ too much. For this reason, for the case study only three types of weights were used.

4. CLUSTERIZATION OF WELDING DOTS

After the tool set has been selected, it is necessary to assign the welding dots to the spotguns. This problem is referred as clusterisation of welding dots. Let the set of subsets $Cluster_j$ be a partition of the set *Dots* such that each cluster corresponds to a spotgun. Let the $Size_j$ be the geometrical bound of the cluster *j* and $Time_j$ be the time required for *j*th spotgun to serve all dots from its cluster. Than the combinatorial optimization problem can be stated in the following way:

$$\bigcup_j Cluster_j = Dots;$$
$$Cluster_j \cap Cluster_k = \varnothing,\ \forall j,k;\ j \ne k;$$
$$S\left(Dot_i, Tool_j\right) = 1\ \ \forall Dot_i \in Cluster_j, \qquad (5)$$
$$\max_j Size\left(Cluster_j\right) \to min;$$
$$\max_j Time\left(Cluster_j\right) \to min,$$

As seen, the formulated problem is an extension of the classical set partitioning problem, that is proven NP-hard. It has been investigated by a number of authors (Tu and Gonzalez, 1978; Papadimitriou and Steiglitz, 1982) who have proposed several effective heuristics and exhaustive search methods. But to date, there does not exist an effective optimization algorithm for clusterisation problem, a generalisation of the set partitioning.

In this paper, a three-phase algorithm is proposed for clusteriation of welding dots: 1) Build an initial solution using one of the proposed heuristics (Narrowing, Grouping or Dichotomy); 2) Improve the solution by applying local optimization or "soft computing" methods (Simulated Annealing or Taboo Search); 3) Use branch and bound method with the time limit to search for the optimum.

Before detailed algorithm description, let us define a optimization criterion. In this paper, two types of objective functions are used. It is assumed that while designing a cell, the user chooses the certain criterion interactively.

The first objective estimates the time that is required both for welding and tool movement from one dot to another:

$$T_j = N\left(Cluster_j\right) \cdot \Delta t + k_T k_M\, \rho_j / V, \qquad (6)$$

where Δt is the welding time per one dot, ρ_j is an estimation of the tool path length, *V* is an average

velocity of the tool and k_T, k_M are correction factors that takes account of technological requirements.

To compute ρ_j, the two-change heuristic for the NP-hard merchant problem (Papadimitriou and Steiglitz, 1982) is used. Its basic idea is that if the condition

$$\Re(i_1,i_2)+\Re(j_1,j_2)>\Re(i_1,j_1)+\Re(i_2,j_2). \quad (7)$$

is true, the replacement of the path segments (i_1 i_2) and (j_1 j_2) by (i_1 j_1) and (i_2 j_2) improves the solution. The complexity of the procedure is $O(N(Dots)^2)$.

The second objective estimates the geometrical size of the cluster. It is proposed to use one of the following definitions of the cluster size: (i) the maximum distance between two dots within the cluster, (ii) the sum of the distances from each dot in the cluster.

The clusterisation algorithm includes three phases. The first of them incorporates several heuristics (Narrowing, Grouping and Dichotomy) to build a good initial solution (it gives several options for a CAD user).

The first heuristic, Narrowing, reduces the set of possible solutions using "greedy" approach. At the beginning, the cluster of each tool contains all dots that can be served by it. On the subsequent steps, the algorithms is looking for dots which can be excluded from the cluster with maximum objective improvement. (A dot can be expelled from a cluster if and only if there exists another cluster that contains this dot). The routine is stopped when each welding dot is assigned to a single tool (i.e. when there is no dot that can be deleted). The complexity of the procedure is $O(\mathrm{N}(Dots)^2 \cdot \mathrm{N}(Toos)^2 \cdot CC)$, where CC is the complexity of the criterion computing.

The second heuristic, Grouping, implements the idea of image processing where a cluster is formed from the nearest dots (Tu and Gonzalez, 1978). But for welding applications, it is common that some neighbouring dots can not be accessed by the same tool. It is taken into account by this heuristic. First of all, the set *Dots* is split into $N(Tools)$ clusters. Then, the problem of accessibility is solved by exchanging dots between clusters and moving a dot from one cluster to another. Using this results, a set of dots that can not be served by their current tools is formed, and these dots are assigned to the allowable clusters to minimise the criterion. The computational complexity of this procedure is $O(\mathrm{N}(Dots)^2 + \mathrm{N}(Toos) \times \mathrm{N}(Dots) \cdot CC)$.

The third heuristic, Dichotomy, relies on the aggregation-decomposition approach (Papadimitriou and Steiglitz, 1982). Here, the tool set is represented as two abstract technological tools such that each of them can serve only dots that can be accessed by the tools it includes. Subsequently, the welding dots are distributed among them and the described procedure is applied to each of the abstract tools. It is continued until all abstract tools are reduced to a single element, i.e. when all the welding dots are assigned to the specific single spotguns. The complexity of the algorithms $O(2^{N(Dots)} \cdot CC)$ is very high, so it incorporates the first heuristic in combination with local optimization and exhaustive search at each step.

The phase 2 improves the initial solution using three successive procedures. First, the local optimization is applied. The dots are moved from one cluster to another and exchanged between clusters as long as criterion improvement is achieved. Then the simulated annealing runs (Laarhoven and Aarts, 1987). This approach allows to seek for the global optimum and overcome partially the primary disadvantage of the local optimization (Souilah et al., 1996). The last improvement procedure is Taboo Search (Glover et al., 1993). It is similar to the simulated annealing but, in addition, it saves the information about a fix number of last iterations to avoid cycling.

The phase 3 uses the branch and bound method to search for the global optimum. It is implemented as a recursive procedure that generates all possible clusterisation alternatives (Cheushev et. al., 1998). If the current partial solution (i.e. a solution that does not include all technological tools and welding dots) is worse than the best found by this moment, then the backtrack is done. Correctness of this operation follows from the monotonicity of the criterion.

It should be noted that the phase 3 is rather sensitive to the initial solution. For this reason, so much attention has been paid to the phases 1 and 2 which deal with generation of the initial solution and its improvement.

5. OPTIMIZATION OF ROBOT LOCATION

5.1 Problem description

Modern commercial software simulators (such as RobCAD, IGRIP, CimStation and others) "give a true engineering representation" which enables users to verify workcell layout, providing reduced risk of unexpected collisions and eliminating costly mistakes. However, the cell component placement is usually performed in interactive mode. Only limited number of CAD packages which have been originated from university research labs (SMAR, ROBOMAX, etc.) incorporate advanced design algorithms that allow automatic robot placement and collision-free path planning.

To eliminate interactive check-and-change loops from usual layout design process (Fig. 2), several approaches have been proposed. In particular, Lueth (1992) has developed automatic layout planning

routines based on fast obstacle transformations into the three-dimensional Cartesian configuration space. Several authors have tried to deal with this problem using conventional FMS layout planning algorithms. However, these algorithms deal with the simplified models of robotic cells and can not be directly applied in industrial robotic CAD systems.

A number of researchers focused on the workcell layout design via optimization of various kinematic criteria. The most popular of them is the manipulability measure of Yoshikava which has been used by Nelson, Pederson and Donath (1987). Another approach has been implemented by Pamanes (1989) whose objective was „to keep joint variables away from their limits as far as possible".

As a rule, the optimal placement of a robotic manipulator is based on minimisation (or maximisation) of a single objective. One of the first attempts of multiobjective robot placement was done by Chiu (1988) who combined four criteria in a scalar performance index. Later, Zeghloul et al (1997) also used multi-criteria approach. But their technique is also based on minimisation of a scalar objective that is computed as the difference between the mean and the standard deviation of the normalised objective components.

In contrast to them, our approach relies on Pareto-optimization that allows to generate a set of optimal solutions instead of a single one. For CAD systems, this feature can be treated as an advantage because the designer should possesses some flexibility of choosing the „best" solution from a set of the „good" ones taking into account some additional (unmodelled) constraints. In this paper, our approach is being developed using genetic algorithms.

5.2 Problem statement

Let us define the robot task (Fig. 3) by a set of positions and orientations (locations) of the end-effector referred to the world coordinate system:

$$^{\circ}Task = \left\{ ^{\circ}Loc_i = \left(^{\circ}pos_i ,\ ^{\circ}ori_i \right);\ \ i = 1{:}n \right\}, \qquad (8)$$

where pos_i and ori_i are the three-dimensional vectors of position and orientation respectively, and the symbol "°" denotes the absolute („world") coordinate system. Let us also define the coordinate transformation function $\vartheta(Loc, \omega)$ which allows to modify the prescribed task locations by varying of certain „free" parameters $\omega \in \Omega$ without violation the technological requirements:

$$^{\circ}Task(\vartheta) = \left\{ \vartheta\left(^{\circ}Loc_i , \omega_i \right);\ \ \omega_i \in \Omega_i ,\ \ i = 1{:}n \right\}. \quad (9)$$

The physical meaning of the function $\vartheta(Loc, \omega)$ essentially depends on the technological process. For instance, for arc and spot welding it is allowed to modify the end-effector orientation while the TCP position can not be changed.

To perform the prescribed task, the robotic manipulator should be proper located in the world coodinate system. Let us describe its location by the coordinate transformation function $^{\circ}T_b\left(Loc, \lambda\right)$ which incorporates 6 parameters $\lambda = \left(x_b , y_b , z_b , \alpha_b , \beta_b , \gamma_b \right)$ that define the position and orientation of the robot „*Base*" coordinate system (see Fig. 3). These parameters are the design variables in the robot placement problem.

Using these notation and assuming that each location of the robot base can be evaluated by a number of performance indices J_1 (.), J_2(.), ... J_m(.), the problem of optimal robot placement can be stated as follows:

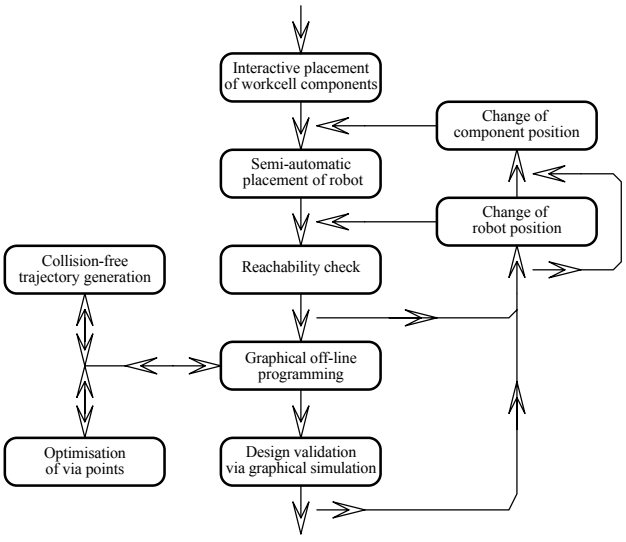

Fig. 2. Typical stages of the workcell layout design

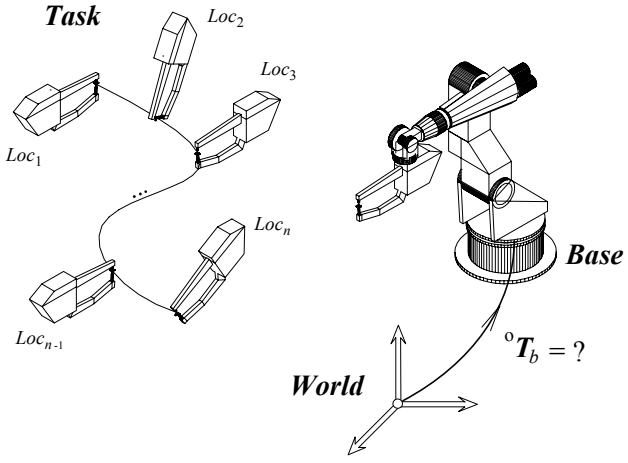

Fig. 3. Robot placement for a prescribed task

find the position and orientation of the robot base defined by the vector λ which minimises the objectives

$$J_j\left[\bigcup_i {}^{b}T_{o}\left(\vartheta\left({}^{o}Loc_i,\omega_i\right),\lambda\right)\right]\rightarrow \min_{\lambda,\Omega,C} \forall j=1:m \qquad (10)$$

where ${}^{b}T_{o}\left(Loc,\lambda\right)$ is the reverse coordinate transformation function; $\Omega=col\left(\omega_1,\omega_2,...,\omega_n\right)$; and $C=col\left(Conf_1,Conf_2,...,Conf_n\right)$ is the vector of the manipulator configuration indices. In order to complete the formulation, let us also define the design constraints and objectives.

5.3 Design constraits

In our formulation, the design vector $\boldsymbol{\lambda}$ is subject to the explicit and implicit constraints. The first type of constraints is directly applied to the components of $\boldsymbol{\lambda}$. Their upper and lower bound are defined taking into account the available workspace within the manufacturing line and, in addition, the preferable orientation of the robot base. The implicit constraints arise from several reasons. The most essential of them are the solvability of the inverse kinematics for all locations and avoiding robot/environment interferences during the motion. Besides, for specific applications, there exist some technological constraints.

The *inverse kinematics constraints* are caused by two main factors. The first of them is the limitation of the manipulator geometry (or workspace), and the second is the manipulator joint limits. Independent of the particular reason, the inverse kinematics constraints can be defined by the binary equations

$$InvKin\_Solv\left({}^{b}Loc_i,Conf\right)="Yes" \quad \forall i=1:n \qquad (11)$$

where ${}^{b}Loc_i$ is the end-effector location for the *i*-th task point relative to the robot base

$${}^{b}Loc_i={}^{b}T_{o}\left(\vartheta\left({}^{o}Loc_i,\omega_i\right),\lambda\right) \qquad (12)$$

and *Conf* is the configuration index. It should be noted, that for some applications (spot welding, for example) it is allowed to access task points with different configurations, while another applications (arc welding and laser cutting, for instance) require to keep the same configuration for some subsequent task points. The latter leads to the several versions of the inverse kinematics constraints.

The *collision constraints* allow to avoid interference between the robot and the workpiece, fixture, positioner or the workcell furniture. Collision detection technique depends on the graphical environment of the robotic CAD package. If a sophisticated graphical system is used, the collision detection is an incorporated tool. Otherwise, special application program are developed to detect interferences. Independent of the particular algorithms, the collision constraints can be also described by the binary equations

$$Collis\_Exst\left({}^{b}Loc_i,Conf\right)="No" \quad \forall i=1:n \qquad (13)$$

where the configuration index *Conf* must be exactly the same as for the inverse kinematics constraints.

The *technological constrains* reflect specific requirements of a particular application. For example, for welding robots, the deformation of the power supply cable is one of the most critical issues. In general case, the technological constrains are expressed as a set of algebraic inequalities but their validation in CAD system require user-supplied routines.

5.4 Design objectives

In industrial applications, the quality of robot placement is usually evaluated by several performance indices such as the working cycle time, deviation of the joint variables, distance to the joint limits, distance to the singular configurations, etc.

The *cycle time* T_Σ can be estimated for both a specified path defined by the node points and a global time-optimal path between given end points with undetermined sequence of intermediate points. The first approach is common for applications with continuos-path technological motions (arc welding, laser or water jet cutting) while the second approach is usual for operations where the motions do not incorporate technological actions (spot welding, drilling, etc.).

In our approach the travel time for each path segment is estimated using the manipulator kinematic model and the velocity/acceleration constraints imposed on each joint variable and on the Cartesian coordinates (in the case of the linear or circular interpolation). It corresponds to the commercial robot controllers where all path segments are generated using trapezoid (or triangular) velocity profiles. Under such assumptions, the travel time for a single segment is computed as

$$T=\max\left\{t_0,t_1,t_2,...t_N\right\}+\mu\cdot\max\left\{\tau_0,\tau_1,\tau_2,...t_N\right\} \qquad (14)$$

where $\mu\in[0,1]$ is the path-smoothing factor; N is the number of robot joints; τ_k and t_k are the duration of constant-velocity and constant-acceleration intervals respectively for each separate set of the velocity/acceleration constraints:

$$\begin{aligned} &if && \Delta q_k \geq \left(\dot{q}_k^{max}\right)^2 \Big/ \ddot{q}_k^{max} \\ &then && t_k=\dot{q}_k^{max}\Big/\ddot{q}_k^{max};\tau_k=\Delta q_k\Big/\dot{q}_k^{max}, \\ &else && t_k=\tau_k=\sqrt{\Delta q_k\Big/\ddot{q}_k^{max}}. \end{aligned} \qquad (15)$$

The similar expressions are also valid for the Cartesian constraints which correspond to the subscript "o".

Therefore, the total cycle time T_Σ is computed as a sum of travel time (14) for all segments. However, in the case when the end points of the global path are given but the sequence of intermediate points is not specified, it is necessary to solve the NP-hard travelling salesman problem. This difficulty is overcome by using heuristic algorithms which insure sub-optimal solution suitable for industrial applications (10).

The *deviation of joint variables* can be expressed via several performance indices. The simplest of them are the „*volume of motion*" that are defined as

$$Q_\Sigma^k(\lambda, C, \Omega) = \sum_i \left| q_k^i - q_k^{i-1} \right| \qquad k = 1{:}N \tag{16}$$

where k is the joint number , and i is the task point number. Another objectives of this type, the „*distance from the joint limits*", allow to keep the joint variables as far as possible from their upper and lower bounds

$$\delta Q_k(\lambda, C, \Omega) = \max_i \left\{ q_k^{max} - q_k^i;\ \ q_k^i - q_k^{min} \right\} \tag{17}$$

or to locate them as close as possible to the centre of permissible range

$$\delta \overline{Q}_k(\lambda, C, \Omega) = \max_i \left[q_k^i - \left(q_k^{max} + q_k^{min} \right) / 2 \right]. \tag{18}$$

Another group of indices, *dexterity*, quantify the manipulator ability to move the end-effector and to apply forces in the task points as easily as possible. The simplest of such measures evaluate *„distance from the singularity*" of the manipulator posture. The most popular index is the „*manipulability*"

$$w_i = \sqrt{\det\left(\boldsymbol{J}_i \cdot \boldsymbol{J}_i^T \right)} \tag{19}$$

of Yoshikawa which is computed using the manipulator Jacobians $\boldsymbol{J}_i$ in the task points. Some closely related measures are the condition number of Jacobian and the smallest singular value.

Similar performance indices are applied to quantify the „*dynamical mobility*" and „*lightness*" of a manipulator arm in the task points. In contrast to the dexterity measures that operates with the kinematic Jacobian only, the inertness measures consider the product $\boldsymbol{J}\cdot\boldsymbol{D}^{-1}$ of the Jacobian $\boldsymbol{J}$ and the inverse inertia matrix $\boldsymbol{D}$ of the manipulator. As follows from the equations of dynamics and kinematics, the product $\boldsymbol{J}\cdot\boldsymbol{D}^{-1}$ defines a linear mapping from the joint actuator space $\boldsymbol{\tau}$ to the Cartesian acceleration space $\boldsymbol{a}$, in the absence of external forces, friction and gravity for the steady state of the manipulator ($\dot{\boldsymbol{q}} = 0$).

The quality of the robot placement can be also evaluated by the shortest distance to the obstacles. Unlike conventional technique that utilise the joint-variable configuration space to deal with the obstacle avoidance (or the direct distance calculation in the Cartesian space), the proposed approach relies on the computing of the „nearness" to the border of admissible robot locations in the Cartesian space of robot base locations.

In this formulation, the total area of possible robot locations is represented as a 3D mesh $\lambda_j \in Mesh_{xyz}$ and each node λ_j is checked for a collision-free reachability of all task points. The set of admissible robot locations Λ_o is composed from only those nodes which ensure full task execution for certain configurations $\boldsymbol{C}$ and „free" parameters $\boldsymbol{\Omega}$. The performance measure is defined as the shortest distance from the current base location λ to the nearest node which is unsuitable to the robot placement:

$$\rho(\lambda, C, \Omega) = \min_j \left\| \lambda - \lambda_j \right\|, \quad \lambda_j \in Mesh_{xyz} \setminus \Lambda_o \tag{20}$$

It is obvious, that such objective can be also treated as an indirect measure of the „distance to the joint limits", because each node λ_j is checked for both possible collisions and the solvability of the inverse kinematics for all task points. It is an advantage for the CAD applications which allows to reduce the number of performance indices.

5.5 Optimization algorithm

A straightforward approach in multi-criteria optimal robot placement can be based on selecting the best solutions from the multi-dimensional grid in search space. But this technique is extremely time-consuming for real-life industrial applications. For example, evaluating of a single node for KUKA-760 robot and GAZEL lorry cabin takes about 0.5 s for processor Intel Pentium II (400MHz). Therefore, typical mesh with 400 mm width and 3000 nodes requires about 25 min to quantify all nodes and to select Pareto-optimal solutions (Statnikov et al., 1995). However, manufacturing engineers need higher precision with lower mesh width.

An intelligent approach to the problem, which is used here, enables designer to start optimization with a large-width mesh and to finish with Pareto-optimal solutions that are located in the nodes of a fine mesh. To avoid evaluating of each node of the fine mesh, a specific genetic algorithm (GA) is applied. Compared with traditional optimization methods, GAs are robust and tolerant to the form of the function to be optimized (Dill and Perkowski, 1997; Quagliarella et. al., 1998; Luba et. al., 2000). They have been successfully implemented to a number of multiobjective optimization problems (Fonseca and Fleming, 1995).

In order to solve the robot placement problem by a GA, it is necessary to represent the robot location $\boldsymbol{\lambda}$ as a *chromosome* which is evaluated by a set of criteria $J_1(\boldsymbol{\lambda})$, $J_2(\boldsymbol{\lambda})$, ... $J_m(\boldsymbol{\lambda})$. The *initial population* is generated by selecting Pareto-optimal solutions from multi-dimensional grid $Mesh_0(\boldsymbol{\lambda})$ with relatively large width. The next generations are snapped to another grid $Mesh_1(\boldsymbol{\lambda})$ with small width. They are created using two GA operators: cross-over and mutation.

The *cross-over* operator is applied as follows. Members of the current generations are pared at random. For the user defined number of randomly selected pairs, a cross-position is determined. This position is also computed at random as an intermediate point of the straight-line segment that joints two corresponding locations. Besides, the intermediate point is snapped to the fine mesh.

The *mutation* operator changes the components of population items by shifting them by small random values with user-defined range and probability. The number of chromosomes to be mutated is also limited and is treated as an algorithm parameter. Similar to the cross-over operator, the mutated locations are snapped to the fine mesh.

An elitist replacement approach is applied to *select* chromosomes for the next generation. The genetic algorithm uses criteria values to decide which population items should be used in the next evolution cycle. At each iteration, the population consists of the Pareto-optimal solutions only, and they are chosen from the union of "parents" and "children". In contrast to the classical GAs were selection is based on scalar fitness function, the proposed algorithm implements the survival-of-the-fittest concept in a different way. It emulates the "absorbing" of weak chromosomes by strong ones which overtake them by all components of multiple performance index.

For such selection operator, the size of population changes from generation to generation, and can both increase and decrease. It is even possible that a super-chromosome can appear at a certain step which dominates all other solutions that have been tested by the moment. In this case, the population is reduced down to a single chromosome which is used for reproduction via mutation. The search process continues until a predefined number of generations is reached (or a given number of evolution cycles).

6. COMPUTATIONAL RESULTS

To examine the robustness and efficiency of the developed optimization algorithms, a number of numerical examples has been generated and solved. Both algorithms were coded in C++, compiled for AutoCAD Development System and built in ROBOMAX system, software package for computer-aided design of welding robotic cells which is described in details in (Pashkevich et al., 1999) and is widely used in Russian automotive industry. Numerical experiments were carried out on an Intel Xeon Workstation. For the optimal tool selection algorithm, 10 problem sets has been tested (20 samples for each number of tools). It has been established, that within 60 seconds the number of tools has been decreased from 70 to 5-12.

For the dot clusterisation algorithm, three heuristics (Narrowing, Grouping and Dichotomy) have been compared. To perform it, 5 problem sets with 20 samples in each set has been generated and tested with 300 seconds time limit (N(Tools) = 5). It has been discovered, that for small number of dots Dichotomy is the best, but Grouping takes up for large dots sets. Besides, Grouping requires approximately 1 sec for all samples while Dichotomy is rather slow procedure. The third heuristic, Narrowing, is efficient for randomly generated dots only.

Another set of the simulation tests were based on realistic data and included from 20 to 100 working dots and from 2 to 5 robots. The average distance between the dots was about 0.85 m and they were located at from 2 to 10 straight-line welding seams. It has been established that the GA is slightly faster than the 3Ph, but the efficiency of the GA and the HGA are roughly the same. Depending on the welding dots distribution and the accessibility constraints, the GA can overcome the HGA and vice versa. For example, for the 5 clusters and 60 working dots the GA and the HGA yield the cluster processing time and size (19.0 s, 905 mm) and (20.5 s, 1105 mm) respectively. But for the for the 4 clusters and 40 working dots the corresponding results are (14.2 s, 617 mm) and (13.7 s, 439 mm) respectively.

The simulation results for two typical tests are shown on Figs. 4 and 5 where Test#1 corresponds to 3 robots and 50 working dots, and Test#2 corresponds to 4 robots and 30 dots. As follows from these results, the algorithms performance highly depends on both the workpiece geometry and the selection of the primary objective. Moreover, in certain cases the performance measures #1 and #2 can compete to each other, and the single-objective optimization can yield results which do not satisfy the designer.

The computational results for robot location problem are presented in Figs. 6 and 7. For demonstration purposes, the search space was limited by xy-plane and robot base was oriented strictly vertical. The quality of the robot placement was evaluated by two performance indices: work cycle time J_1 and distance to the joint limits or singularities J_2. As follows from

Fig.4, the admissible solutions form four regions in the search space where the objectives J_1 and J_2 compete to each other. Besides, functions $J_1(x,y)$ and $J_2(x,y)$ are neither differentiable nor continuous.

The initial population was generated by selecting 4 Pareto-optimal solutions from admissible nodes of the rough grid. During evolution, population items were mutated, cross-overed and snapped to the fine grid. After 90 generations (see Fig.7), the population grew up to 34 items which include all Pareto-optimal nodes of the fine grid. The same result have been also achieved by straightforward testing of the admissible nodes. Therefore, the simulation have proved the efficiency of the developed GA which allows to avoid exhaustive testing of all fine grid nodes.

7. CONCLUSION

The developed genetic algorithms ensure solution of the specific integer optimization problems, which arise in CAD of industrial robotic systems and take into account the essential technological features. In contrast to traditional set covering and set partitioning problems, that are common for logic synthesis (Coudert and Madre, 1995; Rudell, 1996), the developed algorithms deal will completely different objective functions which require NP-hard minimization per each iteration. For this reason, various known implicit techniques proved to be too slow for the problems of real-life size, and genetic programming looks like a reasonable solution.

The developed algorithms have been incorporated in commercial software package ROBOMAX that is widely used in Russian automotive industry. The efficiency of the proposed algorithms has been carefully investigated via computer simulation for real-life case studies.

This research is a part of a major project which is going on in cooperation with a number universities of Belarus, Russia, France and UK, and is aimed at the development of multiobjective optimization tools for robotic CAD systems. Further work will deal with GAs application to the efficient generation of robotic path for laser cutting application, as well as developing of web-based interface for ROBOMAX package.

ACKNOWLEDGEMENTS

The author gratefully acknowledges Vladimir Shmerko and Svetlana Yanushkevich (University of Szczecin, Poland), and prof. Marek Perkowski (Portland State University) for many constructive comments on the work leading to this paper. The financial support of the European Community through the INTAS project 96-0812 and Belarusian Fund of Fundamental Research through the project T97-030 are also gratefully acknowledged.

REFERENCES

Aho, A.V., J.E. Hopcroft and J.D. Ullman, (1974). *The Design and Analysis of Computer Algorithms*. Reading, MA: Addison-Wesley, 374 p.

Baker, B.S., E.G. Coffman and R.L. Rivest, (1980). Orthogonal packing in two dimensions, *SIAM Journal of Computing*, Vol. 9, no. 4, pp. 846-855.

Cheushev, V., V. Shmerko, D. Simovici and S. Yanushkevich, (1998). Functional entropy and decision trees, Proc. IEEE 28th Int. Symp. on Multiple-Valued Logic, pp. 357-362

Chiu, S. L. (1988). Task compitability of manipulator posters, *The Int. J. of Robotic Research*, Vol. 7 (5), 13-21.

Christofides, N. (1975) *Graph theory: an algorithmic approach*. Academic Press, New York, 415 p.

Cook, J.S. and B.T. Han (1994). Optimal robot selection and work station assignment for a CIM system, *IEEE Trans. on Robotics and Automatics*, vol. 10, no. 2, pp. 210-219.

Coudert, O. and Madre J.C., (1995). New Ideas for Solving Covering Problems. *Proceedings of 32nd* ACM/IEEE *Design Automation Conference*, Las Vegas, Nevada, pp. 641-646.

Fonseca, C. M. and P. J.Fleming, (1995). An overview of evolutionary algorithms in multiobjective optimization, *Evolutionary computing*, Vol. 3 (1), pp. 1-16.

Glover, E. Taillard and D. De Werra (1993), A user's guide to taboo search, *Annals of Operations research*, vol. 41, pp. 3-28.

Heragu, S.S. (1992). Recent models and techniques for the facility layout problem, *European Journal of Operational Research*, Vol. 57, no. 2, pp. 136-144.

Liao, S. and S. Devadas, (1996). Solving Covering Problems Using LPR-Based Lower Bounds, *MIT Technical Report*, December 1996

Luba, T., C. Moraga, S. Yanushkevich, M. Opoka and V. Shmerko, (2000). Evolutionary Multi-Level Network Synthesis in Given Design Style, *Proc. IEEE 30th Int. Symp. on Multiple-Valued Logic*.

Lueth, T C (1992). Automated planning of robots workcell layouts, P*roc. of the IEEE Int. Conf. on Robotics and Automation*, pp.1103-1108.

McKerrow, P. J. (1991). *Introduction to robotics*. Addison-Wesley, 797 p.

Nelson B, Pedersen K and Donath M (1987). Locating assembly tasks in a manipulator's workspace, *Proc. of the IEEE International Conf. on Robotics and Automation*, pp. 1367-1372.

Pamanes J A (1989). A criterion for optimal placement of robotic manipulators, *Proc. of the 6th IFAC Symp. on Information Conrol Problems in Manufacturing Technology*, pp. 183-187.

Papadimitriou, C.H. and K. Steiglitz (1982). *Combinatorial optimization: Algorithms and Complexity*. Englewood Cliffs, New Jersey, 435p.

Pashkevich A., M.Pashkevich and E.Antonov (1998). Combinatorial Algorithms for Computer-Aided Design of Industrial Robotic Cells. *Proceedings of the 5th IFAC Workshop on Algorithms and Architectures for Real-Time Control*, Cancun, Mexico.

Pashkevich, A. and M. Pashkevich (1998). Multi-objective optimization of robot location in a workcell using genetic algorithms. *Proceedings of the International Conference CONTROL'98*, Swansea, UK, September, pp. 132-140.

Pashkevich, A., E. Antonov and O.Chumakov (1999). Evolutionary Otimisation Tools in CAD of Industrial Robotic Cells, Int. Conference on Industrial Logistics ICIL'99, University of Southampton, UK , pp. 210-218.

Pashkevich, A., E.Levner and M.Pashkevich, (1999)**.** Network Optimization Techniques in CAD of Industrial Robotic Systems. *International Conference on Distributed Computer Communication Networks* (DCCN'99), Holon Institute of Technology Arts and Sciences, Israel, pp. 105-112.

Dill, K. and M. Perkowski, (1997). Minimization of Generalized Reed-Muller Forms with Genetic Operators. *Proc. Genetic Programming '97 Conf.,* Stanford Univ., CA.

Rudell, R. (1996). Tutorial: Design of a logic synthesis system. Proceedings of *33rd Design Automation Conference*, Las Vegas, NV, pp. 191-196.

Quagliarella, D. (Eds.) et. al. (1998), *Genetic Algorithm and Evolution Strategies in Engineering and Computer Science. Recent Advances and Industrial Applications*, John Wiley and Sons Ltd,

Souilah A, Mecheri Y and Bennesroune A, (1996). Intra-cell Layout Design: a Combinatorial Optimization Approach, V.Shmerko, J.Soldek, A.Dolgi, S.Yanushkevich (Eds.), *New Information Technologies in Education*. Technical University of Szczecin Academic Publishers, pp. 123-130.

Statnikov, R.B. and J. B. Matusov (1995). *Multi-criteria Optimization and Engineering.* New York, Chapman and Hall.

Stecke, K.E. (1983). Formulation and solution of nonlinear integer production planning problems for flexible manufacturing systems, *Management Science*, Vol. 29, no. 3, pp. 273-288.

Van Laarhoven and E.H. Aarts, (1987). *Simulated Annealing: Theory and Applications.* D. Reidel Publishing Co. Dordrecht, 253 p.

Zeghloul S, Blanchard B and Ayrault M, 1997, SMAR: A Robot Modelling and Simulating System, *Robotica*, Vol. 15, pp. 63-73.

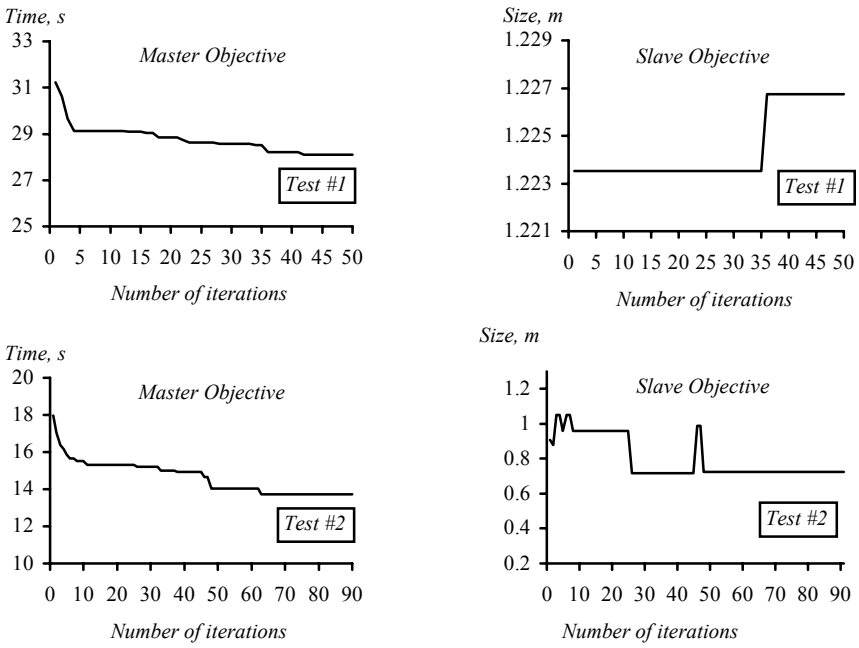

Fig. 4. Behaviour of the GA objectives during optimization of the cluster processing time.

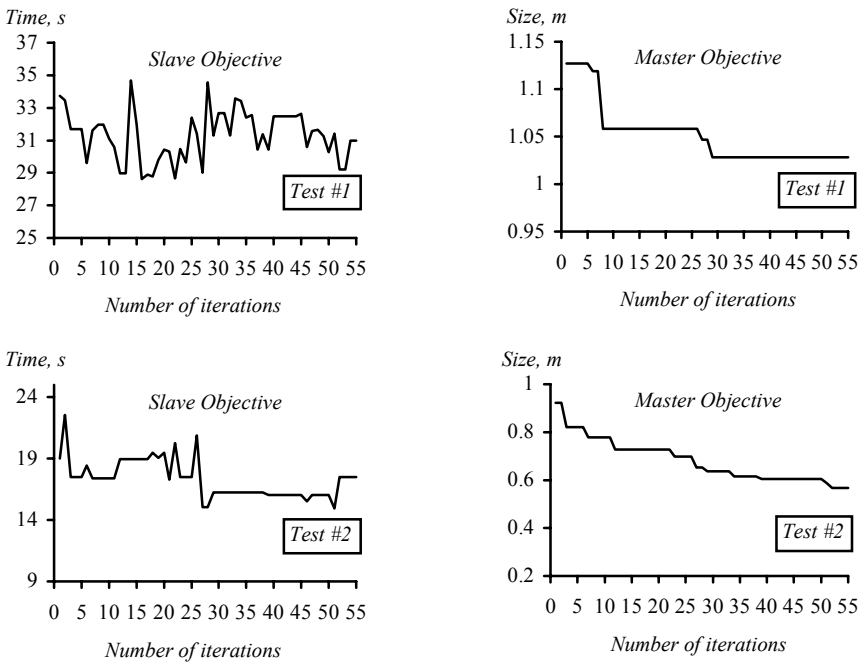

Fig. 6. Behaviour of the GA objectives during optimization of the cluster size.

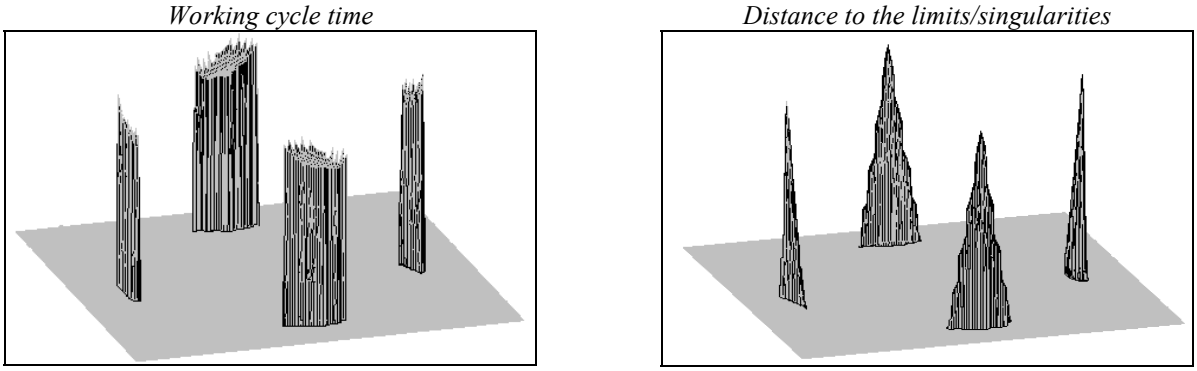

Fig. 6. Optimization criteria (axes X and Y correspond to design variables, i.e. robot base location in a workcell; axis Z corresponds to an according objective function).

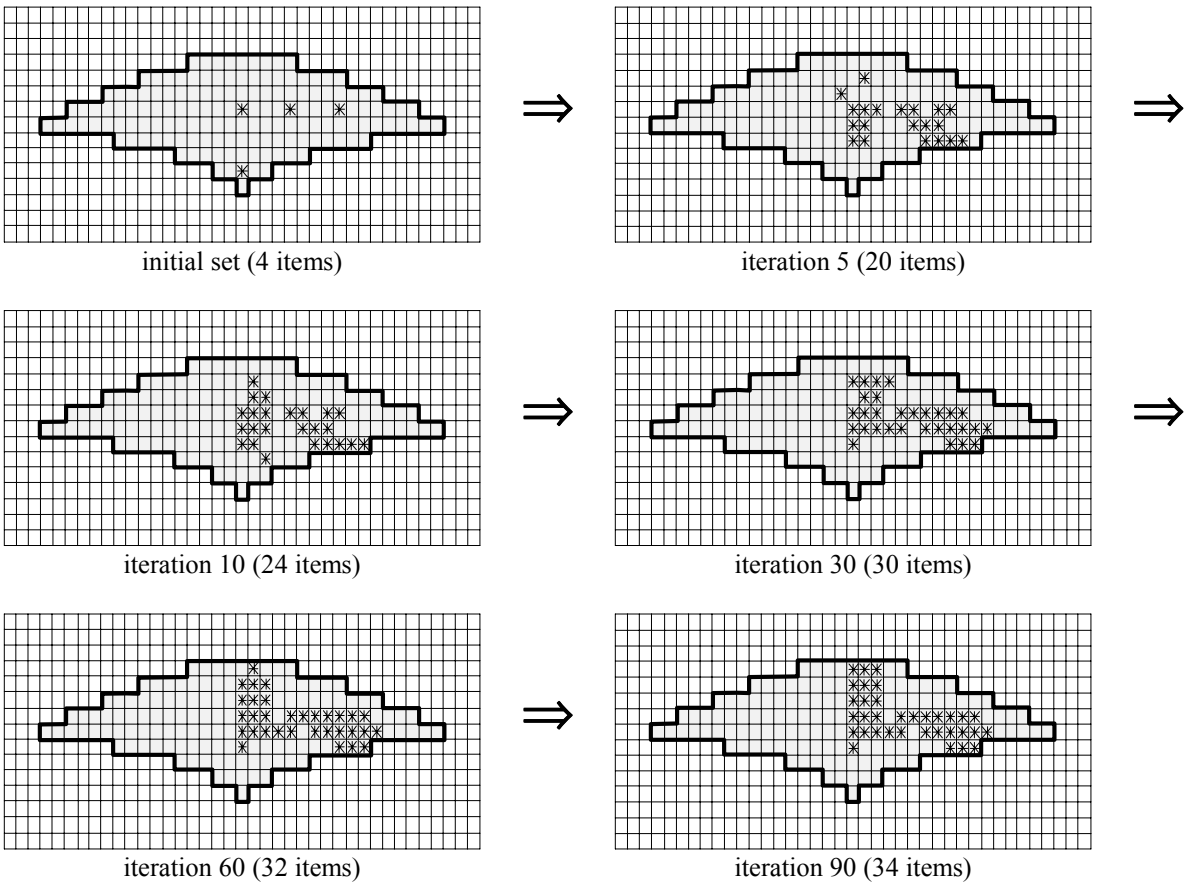

Fig. 7. Generation of Pareto-optimal solutions using the developed GA