Designing DNA Microarrays with Ant Colony Optimization

Ivković, Nikola; Golub, Marin; Jakobović, Domagoj

Designing DNA Microarrays with Ant Colony Optimization

Nikola Ivkovic[1*], Marin Golub[2], Domagoj Jakobovic[2]

[1] Faculty of Organization and Informatics, University of Zagreb, Pavlinska 2, HR 42000 Varaždin, Croatia
[2] Faculty of Electrical Engineering and Computing, University of Zagreb, Unska 3, HR 10000 Zagreb, Croatia

* Corresponding author. Tel.: +385 42 390872; email: Nikola.ivkovic@foi.hr
Manuscript submitted September 4, 2015; accepted September 25, 2015.
doi: ???

Abstract: DNA microarrays are manufactured by synthesizing probes on a solid surface with the help of light and a sequence of lithographic masks. Unintentional illumination can create defects on the microarray due to small dimensions and light properties, but a suitable arrangement of probes can reduce the probability of defects. The problem of designing DNA microarrays is computationally hard and there is no publicly available algorithm that can solve this problem exactly, in polynomial time. This study investigates the suitability of the ant colony optimization (ACO) metaheuristic for finding optimal or at least good microarray designs. This research is based on a MAX-MIN ant system variant that is enhanced with 2-opt local optimization and max-κ-best pheromone reinforcement strategy. Experiments were conducted on problem instances based on border length and conflict index models. The proposed algorithm found solutions that are better than the best solutions previously published for 10 out of 14 problem instances.

Key words: biochip design, swarm intelligence, max-κ-best, MAX-MIN ant system.

1. Introduction

A DNA microarray is a set of probes placed on solid material that can be used for genetic testing. The probes are fragments of DNA made from nucleobases: cytosine, guanine, adenine, and thymine. DNA microarrays, also known as biochips, have applications in genetics, molecular biology, medicine, and pharmacology for detecting genes, diagnosing diseases, inventing drugs, toxicological analyses etc. [1].

In the process of designing DNA microarrays arises a hard combinatorial problem of arranging probes in a way that minimizes the chance of defects and reduces the complexity of masks used in production.

For many optimization problems, efficient exact algorithms are not available. The question whether NP-hard problems can be solved in polynomial time remains unresolved in spite of serious research efforts. Over the past few decades, researchers have developed versatile metaheuristic methods that can often find good solutions for such problems, although they cannot guarantee finding optimal solutions.

In this paper, the possibility of using the ant colony optimization (ACO) metaheuristic for the microarray layout problem is investigated. Ant colony optimization (ACO) is a general metaheuristic inspired by the behavior of biological ants. It is a constructive type of a stochastic optimization algorithm whose construction procedure is guided by pheromone trails, and possibly by heuristic information that is specific to a particular type of the optimization problem. The pheromone trails are altered by ants to reflect the experience that the ants have gathered in previous attempts to find optimal solutions of the problem.

Since its introduction, ACO algorithms were found to be well-suited for many challenging computational problems: combinatorial optimization, dynamic and stochastic optimization, multiobjective optimization, constraint satisfaction, and continuous optimization [2], [3].

The first published ACO algorithm, ant system (AS), proved to be a promising concept, but it could not compete with state-of-the-art algorithms. In attempts to improve algorithmic performance, different variants of ACO algorithms were published in the literature. Elitist ant system (EAS) allowed the best solution found from the beginning of the algorithm to additionally reinforce pheromone trails in order to improve convergence toward more promising areas in the search space [4]. Ant colony system (ACS) uses a rather greedy pseudorandom proportional rule in the solution construction procedure; there are local and global pheromone evaporations, and only one ("the best") solution reinforces the pheromone trails [5]. Rank-based ant system (AS_{rank}) enhances AS by using more elitism in the pheromone reinforcement procedure than EAS. Only a subset of the best solutions found in the current iteration and the best so far solution are used for the pheromone reinforcement procedure [6]. Approximate nondeterministic tree search (ANTS) introduces a unique rule for solution construction and does not perform explicit pheromone evaporation [7]. MAX-MIN ant system uses explicit pheromone bounds and pheromone reinitialization in the case the algorithmic stagnation is detected [8]. Best-worst ant system (BWAS) is similar to MMAS, but in addition, it incorporates mutation of pheromone trails (a concept from evolutionary algorithms) and negative feedback of the worst solution [9]. Three bound ant system (TBAS) uses occasional pheromone contractions instead of regular pheromone evaporations [10]. Among many versions of ACO, MAX-MIN ant system is the most popular owing to good results for many optimization problems; followed by ant colony system. TBAS is a new ACO algorithm experimentally evaluated only on a few types of optimization problems, but it showed very promising results (better than MMAS).

For the purpose of this research, MMAS with 2-opt local optimization and extended pheromone reinforcement strategies were used. Conducted experiments showed that MMAS is well-suited for the microarray layout problem, and for most problem instances the new best solutions were obtained.

2. Microarray Design Problem

One way of producing a DNA microarray is to synthesize probes on a solid surface made of glass or plastic. At the beginning the microarray has empty spots, on which probes are built by adding nucleobase after nucleobase until all probes are completed. Although one probe in not actually one DNA molecule, but rather many copies of the same single-stranded DNA molecule on a particular spot, it is logically simpler to think about the probe as one single-stranded DNA molecule. The probes can be produced by using photolithography similar to the process of producing integrated circuits in microelectronics. Photolithographic masks are used to select spots on which particular nucleobases will be inserted. Using one mask (this is one step of production) only one type of nucleobases: cytosine (C), guanine (G), adenine (A) or thymine (T) can be inserted on unmasked spots. In all other spots on the microarray, which are masked, nucleobases are not inserted. These steps are repeated by using right masks in the right order until all probes are completed.

For example, Fig. 1 shows six masks for producing very simple microarray with 9 spots, organized in 3x3 matrix, where every single-stranded DNA is made from three nucleobases. The probe in the right upper corner (AGT) is produced by inserting A in step 1 (using mask M1), G in step 2, and T in step 5. The insertion of particular nucleobase is done by using light which goes through unmasked spots. Because of very small dimensions and nature of light, the nearby area around selected spots might be unintentionally illuminated, and some nucleobases might be added on wrong probes. It is more likely that this defects will happen around the borders between unmasked and masked spots. By carefully arranging and grouping probes it is possible to minimize defects in production. The goodness of layout might be evaluated by using two evaluation models: border length and conflict index [11].

The total border length for microarray layout is defined with (1) as a sum of border lengths $B_1,\ldots, B_T$, of all masks $M_1,\ldots, M_T$. For one mask M_t border length can be calculating by using (2), where $border(x, y)$ is equal to 1 if x and y are adjacent side by side or up and down, but not diagonally, and is equal to 0 otherwise. In the case of the microarray layout from Fig. 1, the border lengths for particular masks are: $B_1 = 9$, $B_2 = 7$, $B_3 = 7$, $B_4 = 6$, $B_5 = 8$, $B_6 = 3$, giving the total border length $B = 34$.

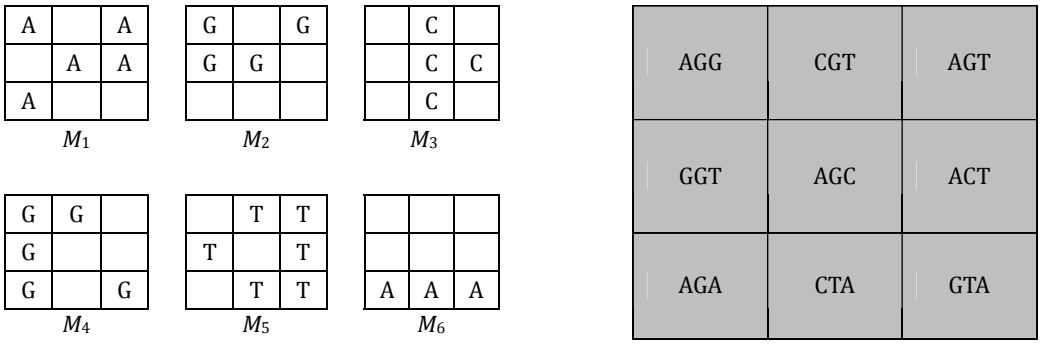

Fig. 1. The sequence of masks M_1, M_2, M_3, …, M_6 used for producing the microarray on the right.

Conflict index is a more general model which takes into account additional considerations about defects in microarray production. It is defined for spot u over all masks M_1 to M_T by (3), where $masked(u, t)$ is equal to 1 if spot u is masked by M_t, and equal to 0 otherwise. Euclidian distance between spots x and y is denoted as $d(x, y)$. Weighting function $\omega(x, t)$ assigns higher values to spots closer to the center of the microarray and lower values to sport closer to the edges [11]. The total conflict index C is a sum of conflict indexes for all spots of the microarray defined by (4).

The goal of the optimization problem is to find microarray layout that minimize total border length or total conflict index, depending on the chosen model. The both models can be coded in quadratic assignment problem.

$$B = \sum_{t=1}^{T} B_t \tag{1}$$

$$B_t = \sum_{x \in X_t, y \in Y_t} border(x, y) \tag{2}$$

$$C(u) = \sum_{t=1}^{T} \left(masked(u,t) \cdot \omega(z,t) \cdot \sum_{v \in N(u)} \frac{1 - masked(v,t)}{\left(d(u,v)\right)^2} \right) \tag{3}$$

$$C = \sum_{u} C(u) \tag{4}$$

3. Ant Colony Optimization for Microarray Design Problem

The ant colony optimization is an iterative constructive metaheuristic. In every iteration of the algorithm, m ants independently construct m different solutions. The solution is constructed by adding solution components to a partial solution until the complete solution is constructed. The choice of selecting particular component is affected by pheromone and heuristic values. After the end of each iteration, the pheromone values are updated based on the quality of constructed solutions.

On the high level of abstraction, MAX-MIN ant system metaheuristic can be described with pseudocode presented in Fig. 2. Precise implementation of particular procedures also depends on type of the problem and design decisions made by a researcher.

```
Initialize()
Loop(UntilStoppingCriteriaAreNotSatisfied){
        AntConstructSolutions();
        LocalOptimization();    //optional
        PheromoneEvaporation();
        PheromoneReinforcement();
}
```

Fig. 2. The pseudocode of MAX-MIN ant system for microarray design problem

In procedure Initialize() algorithmic parameters are set, memory structures are prepared, pheromone and heuristic values are initialized. Both pheromone and heuristic values are usually organized in matrices, with each element linked to one particular solution component. For MMAS, which uses two pheromone bounds in the pheromone update procedure, the upper pheromone bound is set according to (5), where ρ is evaporation rate parameter and $f(s^*)$ is the fitness of the optimal solution. In practice the optimal solution is not known in advance, but estimation of the optimal solution obtained by a simple method can be used instead.

The lower pheromone bound is set to $\tau_{\min} = \vartheta \cdot \tau_{\max}$. An appropriate value for parameter $\vartheta \in \langle 0, 1 \rangle$ can be calculated by choosing probability p_{SH} and using approximate expression given by (6). Alternativle, more precise formulas maight be used [12], [13]. In expression (6), |s| is the number of components in complete solution s, and n_{avg} is the average number of components that are considered in one step of solution construction procedure. The initial pheromone trail value is set equal to upper bound $\tau_{\max}$. When local optimization is applied ϑ is usually set to $0.5 \cdot |s|$.

$$\tau_{\max} = \left[\rho \cdot f(s^*)\right]^{-1} \tag{5}$$

$$\vartheta = \left(1 - \sqrt[|s|]{p_{SH}}\right) \cdot \left[\left(n_{avg} - 1\right) \cdot \sqrt[|s|]{p_{SH}}\right]^{-1} \tag{6}$$

After the initialization is complited, inside the loop, the ant constract solutions, update pheromone trails and optionaly some demone actions like local optimization is performed, until stoping criteria is met (e.g. predefined number of iterations or maximum alowed time).

In AntConstructSolutions() procedure, m ants construct solutions independently. A solution is constructed by adding a component by a component into partial solution s^P until the final solution s is constructed. At the beginning of construction, the partial solution s^P is empty. The C^P set contains all solution components that can be considered for adding to a partial solution s^P. In the case of assigning probes to spots on a microarray, a solution component is one assignment of a particular probe to a particular spot. For microarray with n spots and n probes, there are n^2 possible components.

In one step of AntConstructSolutions() procedure, one component is selected using random proportional rule defined by (7). The probability of adding the component c_l into partial solution s^P depends on the pheromone trail value $\tau_{c(l)}$ associated with that component, heuristic information for that component $\eta_{c(l)}$, parameters α and β, and values associated with all other components that are currently in the set C^P.

$$p\left(c_l \mid s^P\right) = \frac{\tau_{c(i)}^{\alpha} \cdot \eta_{c(l)}^{\beta}}{\sum_{c \in C^P} \left(\tau_c^{\alpha} \cdot \eta_c^{\beta}\right)} \tag{7}$$

When choosing solution component, i.e. probe-spot assignment, first a free spot (or alternatively an unassigned probe) is chosen randomly, and then the assignment is based on random-proportional rule (7).

Performing local optimization is generally optional. By changing the part of solutions, e.g. by removing and

adding some solution components, LocalOptimization() procedure is used to improve solutions constructed in AntConstructSolutions() procedure.

In PheromoneEvaporation() procedure all pheromone values are evaporated by multiplying with $(1 - \rho)$ according to expression (8). In the case that some pheromone value τ_c becomes lesser than τ_{min}, its value is set equal to τ_{min}.

$$\tau_c = (1-\rho)\cdot\tau_c, \forall c \in C \tag{8}$$

In PheromoneReinforcement() procedure it is necessary to choose a solution, according to some strategy, whose components will be rewarded with additional pheromone values. In the case of κ-best strategy the best solution found in last κ iterations is chosen. When max-κ-best strategy is used, then the best solution may be used at most κ iterations [13], [14]. Regardless of selected strategy, for each component c of chosen solution s^{best} the value of corresponding pheromone trail is increased according to expression (9). In the case that some pheromone value τ_c becomes larger than τ_{max}, its value is set equal to τ_{max}.

$$\tau_c = \tau_c + \left[f\left(s^{best}\right)\right]^{-1}, \forall c \in s^{best} \tag{9}$$

4. Experimental Settings and Results

Test instances of microarray layout problem, based on both border length and conflict index models, are publicly available on web page http://www.rahmannlab.de/research/microarray-design along with the best solutions that are known. The shorter names for the problem instances are used in this papers, but the original longer name can be uniquely identified by problem type and problem size. All instances are square-shaped so every solution has eight equivalent representations. There are four possible rotations and also for each rotation it is possible to make a mirror reflection of the solution.

Our implementation of MMAS with 2-opt local optimization and extended reinforcement strategy is done in C++ language. Experiments were conducted on Isabella cluster (isabella.srce.hr), running Linux on machines with various hardware. Local optimization was applied on all solutions constructed in AntConstructSolutions() procedure in the current iteration. For each solution, 2-exchange moves with first improvement were enforced until the solution become 2-optimal. In order to speed up local optimization, data structures with "don't look bits" were used. Also, look up table with precomputed values of expression (7) were used to speedup solution constructions in AntConstructSolutions() procedure. Algorithmic parameters were set according to recommendations from literature and our own experience [13], [15]. Parameters were set to $\alpha = 1$, $\beta = 0$, and number of ants $m = 20$. Few different values were tried for parameters ρ and κ (for max-κ-best strategy) in order to tune the algorithm for better performance.

For smaller problem instances (size n ≤ 81) parameter ρ was tested with values 0.01, 0.05, 0.2 and parameter κ with values 8, 32, ∞. For larger instances $\rho = 0.01$ and max-8-best were omitted.

The maximum number of iterations was set to 10000, except for instances mlpbl121, mlpci121, and mlpbl144 for which it was set to 20000. For each parameter setting experiment was repeated 40 times.

Results of our experiments based on medians are showed in Table 1 and Table 2. Solution quality is expressed as a relative deviation of a median of solutions from the new best solutions published. It was calculated by (10) where *med* is the median value of solutions and *nbs* is the new best solutions for a particular instance (for many instances the best solutions were obtained by experiments published in this paper).

$$med_{rel} = \frac{med - nbs}{nbs}\cdot 100\% \tag{10}$$

The backgrounds of cells in Table 1 are colored according to their values for particular problem instance. The best results are colored with yellow, the worst with red and all the other with shades between yellow and red. For example, in the case of mlpbl64 the best result (med_{rel} = 0.80%) obtained by max-∞–best and ρ = 0.01 is colored yellow and the worst result (med_{rel} = 2.32%) obtained by max-8–best and ρ = 0.01 is colored red.

By the color distribution on Fig. 1, it can be observed that the smallest tested parameter values ρ and κ resulted with poorer algorithmic performance, but the best parameter settings depended on problem instance.

In the case conflict index instances, the best performance was observed with max-32-best and ρ = 0.2 for the most instances. For mlpci36 the median of solutions is equal to the new best solution published (*nbs* = 168611971).

Table 1. Median solutions of MMAS with 2-opt for border length instances

instance	max-8-best			max-32-best			max-∞-best		
	ρ = 0.01	ρ = 0.05	ρ = 0.2	ρ = 0.01	ρ = 0.05	ρ = 0.2	ρ = 0.01	ρ = 0.05	ρ = 0.2
mlpbl36	1.21%	0.49%	**0.12%**	0.49%	0.49%	0.24%	0.61%	0.73%	0.61%
mlpbl49	2.29%	2.20%	1.14%	2.11%	**0.70%**	0.79%	1.23%	1.23%	1.14%
mlpbl64	2.32%	2.25%	1.79%	2.19%	1.53%	1.13%	**0.80%**	0.86%	0.83%
mlpbl81	2.51%	2.41%	2.15%	2.46%	1.89%	1.57%	**1.00%**	**1.00%**	**1.00%**
mlpbl100					2.25%	1.87%		1.27%	**1.23%**
mlpbl121					3.13%	2.82%		2.27%	**2.23%**
mlpbl144					3.24%	2.96%		2.56%	**2.40%**

Table 2. Median solutions of MMAS with 2-opt for conflict index instances

instance	max-8-best			max-32-best			max-∞-best		
	ρ = 0.01	ρ = 0.05	ρ = 0.2	ρ = 0.01	ρ = 0.05	ρ = 0.2	ρ = 0.01	ρ = 0.05	ρ = 0.2
mlpci36	**0.00%**	**0.00%**	**0.00%**	**0.00%**	**0.00%**	**0.00%**	**0.00%**	**0.00%**	**0.00%**
mlpci49	0.01%	**0.00%**	**0.00%**	0.01%	0.01%	**0.00%**	0.16%	0.16%	0.16%
mlpci64	0.22%	0.22%	0.18%	0.17%	0.13%	**0.13%**	0.20%	0.18%	0.19%
mlpci81	0.30%	0.31%	0.29%	0.29%	0.25%	**0.23%**	0.28%	0.25%	0.29%
mlpci100					0.55%	0.53%		0.51%	**0.50%**
mlpci121					0.38%	0.39%		**0.29%**	0.36%
mlpci144					0.29%	**0.25%**		0.28%	0.32%

The best results and algorithms previously used to solve instances of microarray layout problem are taken from web page http://www.rahmannlab.de/research/microarray-design and are presented along with our results in Table 3 and Table 4. The previously used algorithms are greedy randomized adaptive search with path relinking (GRASP-PR) [16], heuristic algorithms RTL-1 [17] and RTL-2 [18], and hybrid of genetic algorithm with tabu search (GATS) [19]. The best solutions in some cases were obtained with parameter settings for which algorithmic performance (in terms of median) was not the best. Parameters ρ and κ from max-κ-best strategy for which MMAS obtained the best solutions are listed in the rightmost column in Table 3 and Table 4.

For instance mlpbl36, MMAS with 2-opt obtained solution equal to the best solution previously published. MMAS with 2-opt local optimization obtained solutions that are better than the best solutions that were published for three instances. In cases of the three largest instances, MMAS with 2-opt obtained better solutions than GRASP-PR, but worse than GATS.

In the case of conflict index based instances of microarray layout problem, MMAS with 2-opt local optimization obtained the new best solutions for all available test instances.

Table 3. The best solutions obtained for instances based on border length

instance name	instance size	best solution previously published	best solution found by algorithm				MMAS with 2-opt	
			RTL-1	RTL-2	GRASP-PR	GATS	best solution	parameters (ρ, κ)
mlpbl36	36	3296	-	3304	3352	**3296**	**3296**	(0.01, 32), (0.01, ∞), (0.05, 8), (0.05, 32), (0.2, 8), (0.2, 32)
mlpbl49	49	4564	4580	-	4660	4564	**4548**	(0.05, 32)
mlpbl64	64	6048	6080	-	6200	6048	**6032**	(0.05, ∞)
mlpbl81	81	7644	-	-	7900	7644	**7636**	(0.01, ∞)
mlpbl100	100	9432	-	-	9684	**9432**	9440	(0.2, ∞)
mlpbl121	121	11640	-	-	12032	**11640**	11812	(0.2, ∞)
mlpbl144	144	13832	-	-	14196	**13832**	14060	(0.2, ∞)

Table 4. The best solutions obtained for instances based on conflict index

instance name	instance size	best solution previously published	best solution found by algorithm		MMAS with 2-opt	
			GRASP-PR	GATS	best solution	parameters (ρ, κ)
mlpci36	36	169016907	169925219	169016907	**168611971**	all tested
mlpci49	49	237077377	238859844	237077377	**236355034**	all tested
mlpci64	64	326696412	327770071	326696412	**325671035**	(0.2, 32)
mlpci81	81	428682120	434317170	428682120	**427582150**	(0.05, ∞)
mlpci100	100	525401670	532573788	525401670	**523806646**	(0.05, ∞)
mlpci121	121	658317466	664137090	658317466	**657514941**	(0.05, ∞)
mlpci144	144	803379686	813127758	803379686	**802219898**	(0.05, ∞)

5. Conclusion

In this research, the MAX-MIN ant system was adopted for finding optimal designs of DNA microarrays. The basic ACO metaheuristic was enhanced with 2-opt local optimization to improve algorithmic performance. Limited parameter exploration revealed that the best performance for the instances based on border length is often achieved with rather greedy parameters, i.e. max-κ-best strategy and evaporation rate set to 0.2. In the case of the conflict index based instances, the best results are achieved with slightly less greedy parameter settings. For many problem instances, the proposed algorithm successfully obtained solutions that are better than the best solutions that were previously published.

References

[1] de Rinaldis, E., Lahm, A., Eds. (2007). *DNA microarrays: Current applications*. Taylor & Francis.

[2] Dorigo, M., Stützle, T. (2010). Ant colony optimization: Overview and recent advances. In M. Gendreau, & J.-Y. Potvin (Eds.), *Handbook of Metaheuristics* (pp. 227-263). Springer.

[3] Chandra Mohan, B., Baskaran, R. (2012). A survey: Ant colony optimization based recent research and implementation on several engineering domain. *Expert Systems with Applications*, 39(4):4618 - 4627.

[4] Dorigo, M., Maniezzo, V., Colorni, A. (1996). Ant system: optimization by a colony of cooperating agents. *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, 26:29-41.

[5] Dorigo, M., Gambardella, L. M. (1997). Ant colony system: a cooperative learning approach to the traveling salesman problem. *IEEE Trans. Evolutionary Computation*, 1:53-66.

[6] Bullnheimer, B., Hartl, R.F., Strauss, C. (1999). A new rank based version of the ant system: A Computational Study. *Central European Journal for Operations Research and Economics*, 7:25-38.

[7] Maniezzo, V. (1999). Exact and approximate nondeterministic tree-search procedures for the quadratic assignment problem. *INFORMS J. on Computing*, 11:358-369.

[8] Stützle, T., Hoos, H. H. (2000). MAX-MIN ant system. *Future Generation Comp. Syst.*, 16:889-914.
[9] Cordón, O., Fernández de Viana, I., Herrera, F. (2002). Analysis of the best-worst ant system and its variants on the QAP. In Third International Workshop, Brussel, Belgium.
[10] Ivkovic, N., Golub, M. (2014). A new ant colony optimization algorithm: Three bound ant system. In Swarm Intelligence: 9th International Conference, ANTS 2014, Brussels, Belgium.
[11] de Carvalho, S. A. Jr. (2007). Algorithms for improving the design and production of oligonucleotide microarrays, PhD Dissertation, University of Bielefeld.
[12] Ivkovic, N., Golub, M., Malekovic, M. (2011). A pheromone trails model for MAX-MIN ant system. In Artificial Evolution 2011 (Evolution Artificielle 2011), 10th Biennal International Conference on Artificial Evolution, Angers, France.
[13] Ivkovic, N. (2014). Modeling, analysis and improvement of ant colony optimization algorithms. PhD Dissertation. University of Zagreb.
[14] Ivkovic, N., Malekovic, M., Golub, M. (2011). Extended trail reinforcement strategies for ant colony optimization. In Swarm, Evolutionary, and Memetic Computing – Second International Conference, SEMCCO 2011, Part I, vol. 7076, Springer, 2011, Visakhapatnam, Andhra Pradesh, India.
[15] Dorigo, M., Stützle, T. (2004). *Ant colony optimization*. MIT Press, 2004.
[16] de Carvalho, S. A. Jr., Rahmann, S. (2006). Improving the layout of oligonucleotide microarrays: Pivot partitioning. In Algorithms in Bioinformatics: 6th International Workshop, WABI 2006, Zurich, Switzerland.
[17] de Carvalho, S. A. Jr., Rahmann, S. (2006). Microarray layout as a quadratic assignment problem. In German Conference on Bioinformatics GCB 2006, Tübingen, Germany.
[18] de Carvalho, S. A. Jr., Rahmann, S. (2006). Microarray layout and the quadratic assignment problem. In 14th Annual International Conference on Intelligent Systems for Molecular Biology (ISMB), Fortaleza, Brazil.
[19] Hannenhalli, S., Hubbell, E., Lipshutz, R., Pevzner, P. A. (2002). Combinatorial algorithms for design of DNA arrays, In Dr. Jörg Hoheisel *et al.* (Eds.) Advances in Biochemical Engineering/Biotechnology: Chip Technology, 77:1-19.

Nikola Ivkovic received the MS degree in Computer Engineering and the Ph.D. degree in Computer Science from the Faculty of Electrical Engineering and Computing, University of Zagreb. He is a member of the research and teaching staff at the Department of Information Technologies and Computing of the Faculty of Organization and Informatics, University of Zagreb. His research interests include computational intelligence and optimization, parallel programming, formal methods, operating systems, and computer networks.

Marin Golub received the BS degree (1992) in Electrical Engineering, MS degree (1996) and Ph.D. degree (2001) in Computer Science, all at the Faculty of Electrical Engineering and Computing, University of Zagreb. Currently he is working as an associate professor at the Department of Electronics, Microelectronics, Computer and Intelligent Systems, Faculty of Electrical Engineering and Computing, University of Zagreb. His interests include parallel algorithms, operating systems, evolutionary algorithms and computer system security.

Domagoj Jakobovic received the BS degree in December 1996 and the MS degree in December 2001, in Electrical Engineering. Since April 1997, he is a member of the research and teaching staff at the Department of Electronics, Microelectronics, Computer and Intelligent Systems of the Faculty of Electrical Engineering and Computing, University of Zagreb. He received the PhD degree in December 2005 on the subject of generating scheduling heuristics with genetic programming.