Solutions to the Subset Sum and Partition Problems Using Kernel P Systems

Marian GHEORGHE, Florentin IPATE, Savas KONUR

Abstract - In this paper, we provide solutions to two NP-complete problems, the Subset Sum and the Partition, using a new class of membrane systems called *kernel P systems*. The complexities of the solutions are also investigated.

Key words and phrases : membrane computing, kernel P systems, NP-complete problems, subset sum problem, partition problem, formal verification.

1 Introduction

Membrane systems were introduced in [12][1] at the beginning of the millennium as a new computational model inspired by the structure and distribution of the compartments of living cells, as well as by the main bio-chemical interactions occurring within compartments and at the inter-cellular level. They were later also called P systems. A comprehensive overview of the main research developments in this area is provided in [13]. The key challenges of the membrane systems area and a discussion on some future research directions, can be found in a more recent survey paper [6]. Amongst the most investigated topics in membrane systems are those related to the study of complexity for various models and efficiency in providing solutions to complex problems, more often NP-complete problems.

Different (uniform) polynomial time solutions have been provided for the Subset Sum problem using families of P systems with active membranes [14], P systems with membrane creation [7], tissue P systems with cell division [2] and P systems with symport/antiport and membrane division [15].

Some (uniform) polynomial time solutions have also been produced for the Partition problem by using tissue P systems [1], P systems with cell division [8] and P systems with cell separation [16].

It is worth mentioning that all the classes of P systems providing solutions to such problems not only use the multiset operations occurring in membrane computing, but also rely on some encodings of the problems in

[1] [12] was circulated as a research report already from the Fall 1998.

complex data structures expressed as multisets that show similarities with DNA models. This confirms the statement in [11] regarding the roles of the membranes and DNA structures for life, but also illustrates their modelling capabilities in natural computing.

Kernel P (kP) systems have been introduced in [3] to unify the specification of different variants of P systems and reconsidered in [4]. An efficient solution to the 3-colouring problem has been provided in [5]. The kP systems have been also used to specify and analyse, through formal verification, various biology systems [10, 9].

In this paper we provide solutions to both the Subset Sum and Partition problems using kernel P systems. For both problems, the kernel P system solutions are provided. These solutions can work in maximal parallelism, asynchronous or sequential manner.

The structure of the paper is as follows. Section 2 provides the notation to be used in the paper. Section 3 introduces kernel P systems. Section 4 presents the kernel P system solutions for the Subset Sum and Partition problems. Finally, Section 5 draws the conclusions.

2 Preliminaries

An *alphabet*, A, is a non-empty finite set whose elements are called *symbols*. A *string* or *word* over A is a finite sequence of symbols from A. If u and v are strings over A, then uv is the string obtained by juxtaposition (or concatenation) of the two strings u and v. The number of symbols in a string u is called the *length* of u, denoted by $|u|$. The empty string (with length zero) is denoted by λ. The set of all strings over an alphabet A is denoted by A^* (A^+, when the empty string is removed from the set). Finite or infinite subsets of A^* are called languages over A. If $u \in A^*$ and $a \in A$, then $|u|_a$ denotes the number of occurrences of symbol a in string u.

A multiset m over a set A is a pair (A, f) where $f : A \to \mathbb{N}$ is a mapping. The support of the multiset m is defined as $supp(m) = \{x \in A \mid f(x) > 0\}$. A multiset, which in general might be infinite, is empty (respectively, finite) if its support is the empty set (respectively, a finite set). If $m = (A, f)$ is a finite multiset over A and $supp(m) = \{a_1, \ldots, a_k\}$, then it will be denoted as $m = \{a_1^{f(a_1)}, \ldots, a_k^{f(a_k)}\}$. It can also be represented by the string $a_1^{f(a_1)} \ldots a_k^{f(a_k)}$ over the alphabet $\{a_1, \ldots, a_k\}$.

3 Kernel P Systems

In this section we define a simplified version of kernel P systems. The full model is defined in [3, 4] and an application to solve the three colouring problem can be found in [5]. We first introduce some preliminary concepts.

Definition 3.1 *Let* $Rel = \{<, \leq, =, \neq, \geq, >\}$ *be a set of relational operators. Let* A *be a non-empty finite set. Then, a multiset over* A *with relational operators from* Rel *is an expression* $w = \theta_1 a_1^{n_1} \dots \theta_k a_k^{n_k}$*, where* $a_1^{n_1} \dots a_k^{n_k}$ *is a string over* A *(in which* a_i *and* a_j *are not necessarily distinct,* $1 \leq i < j \leq k$*), and* $\theta_j \in Rel$*, for each* $j, 1 \leq j \leq k$*.*

Remark 3.1 A guard g is a finite disjunction of expressions w introduced by Definition 3.1; $|g|$ denotes the number of w expressions occurring in g; $|g|$ is called the length of the guard g. A particular case of a guard g is the empty set. In this case, the guard is omitted.

Definition 3.2 *Given two non-empty finite sets* A *(alphabet),* L *(labels) and* $l \in L$*, the finite set of rules associated with* l *is denoted by* R_l*. A rule from* R_l *has one of the following two forms:*

(a) $[x]_l \to [y_1]_{l_1} \dots [y_h]_{l_h}$ $\{g\}$*, where* $x \in A^*$ *and* $y_j \in A^*$*,* $l_j \in L$*, for all* j*,* $1 \leq j \leq h$*, and* g *is a guard (*h-membrane division rule*). The length of the rule is defined as* $|x| + |y_1| + \dots + |y_h|$ *(*$|g|$ *can also be considered).*

(b) $x \to y$ $\{g\}$*, where* $x \in A^+$*,* y *is a string over* $A \times L$*,* $y = (a_1, t_1) \dots (a_h, t_h)$*, with* $a_j \in A$ *and* $t_j \in L$*,* $1 \leq j \leq h$*, and* g *is a guard (*rewriting and communication rule*). The length of the rule is defined as* $|x| + |y|$ *(the complexity of the guard* g*, denoted by* $|g|$*, can also be considered for this rule).*

Definition 3.3 *Given two non-empty finite sets* A *and* L*, a compartment is a tuple* (l, w_0, R_l)*, where* $l \in L$*,* w_0 *is a multiset over* A *and* R_l *is a finite set of rules associated with the compartment having the label* l*.*

Definition 3.4 *Let* C *be a compartment, represented by a tuple* (l, w_0, R_l) *as in Definition 3.3, and* r *be a rule from* R_l *with the guard* g*. The guard* g *is considered true at a given moment when the current multiset of* C *is* z *if and only if the following happens:*

(a) If $g = \theta_1 a_1^{n_1} \dots \theta_k a_k^{n_k}$*, then for every* j*,* $1 \leq j \leq k$*,* $|z|_{a_j} \theta_j n_j$ *holds,* $a_j \in A$*,* $\theta_j \in Rel$*.*

(b) If g *is a finite non-empty disjunction of multisets over* A *with relational operators from* Rel*, denoted by* $g = w_1 \mid \dots \mid w_p$*, then there exists* j*,* $1 \leq j \leq p$*, such that* w_j *is true, according to (a).*

(c) If g *is an empty set, then it is always evaluated to true.*

If one needs to refer to the environment then in L one should have a corresponding label for it. The environment does not contain any rules.

Definition 3.5 *A simple kernel P (skP for short) system of degree $n \geq 1$ is a tuple*

$$sk\Pi = (A, L, IO, C_1, ..., C_n, \mu, i_0),$$

where

- *A and L are non-empty finite sets of* objects *and* labels*, respectively;*
- *IO is a finite alphabet, $IO \subseteq A$, of the* environment objects*;*
- *$C_1, \ldots, C_n$ are compartments, as in Definiton 3.3;*
- *$\mu = (V, E)$ is an undirected graph, where $V \subseteq L$ are vertices and E the edges, and*
- *$i_0 \in L$ is the label of the* output compartment*.*

An skP system (see Definition 3.5) consists of a set of n compartments, $C_1, \ldots, C_n$, interconnected by edges from E, of the undirected graph μ. Each compartment is identified by a label from L associated with a vertex of the graph μ, with an initial multiset over A, and a finite set of rules. The compartment receiving the result of a computation, i_0, will always be the environment. An h-membrane division rule, $[x]_l \rightarrow [y_1]_{l_1} \ldots [y_h]_{l_h}$ $\{g\}$, associated with a compartment $C = (l, w_0, R_l)$ is applicable at a given instant to the current multiset z if the guard g is evaluated true with respect to z and x is in the multiset z. When such a rule is applied, the following actions take place: (i) the compartment labelled l is replaced by the h compartments labelled $l_1, \ldots, l_h$ and x is replaced by the multiset y_j in compartment l_j; (ii) the content of l (except x) is copied into each of these compartments after using all the applicable rewriting and communication rules; and (iii) all the links of l are inherited by each of the newly created compartments.

A rewriting and communication rule, $x \rightarrow (a_1, t_1) \ldots (a_h, t_h)$ $\{g\}$, associated with a set of rules, R_l, of a compartment, $C = (l, w_0, R_l)$, is applicable at a given moment to the current multiset z if the guard g is evaluated true, x is contained in z and the target $t_j \in L$, $1 \leq j \leq h$, is either the label of the current compartment, l, or the label of an existing neighbour ($\{l, t_j\} \in E$). When such a rule is applied, the object a_j is sent to the compartment labelled t_j, for each j, $1 \leq j \leq h$. If a target, t_j, refers to a label that appears more than once, then one of the involved compartments is nondeterministically chosen. When t_j indicates the label of the environment, the corresponding object a_j is sent to the environment.

In an skP system, the rules are applied in accordance with various strategies:

(i) *maximally parallel* mode: a multiset of rules are selected in an arbitrary step such that no additional rule is applicable in that step (the

usual restriction that *at most one rule of type (a) – membrane division rule – per membrane can be used in each step* also applies to these systems);

(ii) *asynchronous* mode: an arbitrary number of rules of type (b) and at most one of type (a) are applied;

(iii) *sequential* mode: one rule per compartment is used in each step of the computation.

A more generic strategy can be also instantiated in each compartment, but this is not discussed in this paper (see [3, 4] for the details).

4 Modelling Results

In this section we illustrate the modelling capabilities of the skP systems on two well known NP-complete problems: the Subset Sum and Partition problems.

4.1 Modelling the Partition Problem with skP Systems

The partition problem is formulated as follows: Let V be a finite set and *weight* be a function on V with positive integer values (this is an additive function). It is requested to find, if it exists, a partition of V, denoted V_1, V_2, such that $weight(V_1) = weight(V_2)$. A solution to this problem is provided in [1] by using a recognizer tissue P system with cell division and symport/antiport rules. We make the following notations: let $V = \{v_1, \ldots, v_n\}$, be a finite set, with $weight(v_i) = k_i$, where k_i is a positive integer, $1 \leq i \leq n$.

We provide a solution to the partition problem by using a classical approach in membrane systems, also illustrated in [1], which consists of the following stages: *working space generation*, *verification* and *solution generation*. In the first stage, an exponential space is generated in linear time; this consists of all the compartments that might contain a solution to the problem. Next, in the verification stage, it is checked, in every compartment, whether a solution has been produced. Finally, if at least one solution is found, then a special symbol, *yes*, is sent to the environment, labelled 0; otherwise a *no* is sent to it.

We build the following skP system, which depends on n, for solving the partition problem (i.e., checking whether there is a partition, V_1, V_2, with $weight(V_1) = weight(V_2)$) and working in the *maximally parallel manner*:

$$sk\Pi_P(n) = (A, L, IO, \mu, C_1, C_2, 0),$$

where

- A is the alphabet;
- $L = \{0, 1, 2\}$;
- IO consists of *yes*, *no*; at the end, after $n + 3$ steps, one of the two possible answers will be sent out;
- $C_1 = (1, w_{1,0}, R_1)$, $C_2 = (2, w_{2,0}, R_2)$, where $w_{1,0} = S$, $w_{2,0} = A_1 code(n)$, with $code(n) = v_1^{k_1} \dots v_n^{k_n}$ being the code of the weights of the elements of V ;
- μ is given by the graph with edge $(1, 2)$;
- R_1 and R_2 are given below;
  - R_1 contains:
    $r_{1,1} : S \to (yes, 0) \{\geq T\}$,
    $r_{1,2} : S \to (no, 0) \{\geq F < T\}$;
    $r_{1,1}$ or $r_{1,2}$ sends into the environment the answer *yes* or *no*, respectively;
  - R_2 contains
    membrane division rules:
    $r_{2,i} : [A_i]_2 \to [B_i A_{i+1}]_2 [A_{i+1}]_2, 1 \leq i < n$,
    $r_{2,n} : [A_n]_2 \to [B_n X]_2 [X]_2$;
    these rules generate all the subsets of V in n steps (2^n subsets); each of them being a potential V_1 (V_2 is its complement);
    rewriting rules:
    $r_{2,i,j} : v_i v_j \to v \{= B_i \neq B_j = X \mid \neq B_i = B_j = X\}$,
    $1 \leq i < j \leq n$,
    $r_{2,n+1} : X \to Y$; and
    rewriting and communication rules:
    $r_{2,n+2} : Y \to (F, 1)\{\geq v_1 \mid \dots \mid \geq v_n\}$,
    $r_{2,n+3} : Y \to (T, 1)\{< v_1 \dots < v_n\}$.

One starts the generation stage with two compartments, C_1 and C_2. In the compartment C_1, initially containing a symbol S, the output of the problem, either *yes* or *no*, will be collected. The compartment C_2 has initially a codification of the set V with the weights of its elements – $code(n) = v_1^{k_1} \dots v_n^{k_n}$. C_2 will be divided, using $r_{2,1}$, into two compartments with the same label 2: one with $B_1 A_2$ and the other only with A_2. The presence of B_1 means that the symbol a_1 is in the associated multiset (its absence means that it is not in the associated multiset). Then, each of these compartments is divided by using $r_{2,2}$ and the division process ends by applying $r_{2,n}$. As already mentioned, after n steps, 2^n compartments C_2 are generated, all connected to the compartment C_1. Each compartment corresponds to one of the 2^n

multisets of V. The elements of the multiset are codified by the presence of the corresponding symbols B_i in the compartment. This means that the multiset $V_1 = \{a_{i_1}, \ldots, a_{i_p}\}$ corresponds to a compartment C_2 with $B_{i_1} \ldots B_{i_p}$.

The verification stage consists of two steps. In step $n+1$, in each compartment C_2 every occurrence of an element of the subset of $V_1 \subseteq V$ is paired up with an element of the complement as many times as the weights permit using $r_{2,i,j}$. In parallel to this, X is transformed into Y using $r_{2,n+1}$. Step $n+2$ consists of sending either T or F to compartment C_1 depending on whether all the elements of the subset and its complement are paired up or the weights of the subsets are different, respectively. In these cases, the rules $r_{2,n+3}$ or $r_{2,n+2}$ are used.

Finally, in step $n+3$, the solution is generated, by providing an answer into the environment, either *yes* or *no*, using one of two rules from C_1.

We make some notations with respect to the partitions generated by the above model. As it generates all possible partitions, we denote them by $V_{1,l}, V_{2,l}$, $1 \leq l \leq 2^n$. Let us denote for the partition $V_{1,l} = \{a_{i,1}, \ldots, a_{i,p_l}\}$, $V_{2,l} = \{a_{j,1}, \ldots, a_{j,q_l}\}$, $p_l + q_l = n$, where $weight(a_h) = k_h$, $1 \leq h \leq n$, the following value:

$$m_l = min\{\sum_{1 \leq h \leq p_l} k_{i_h}, \sum_{1 \leq h \leq q_l} k_{j_h}\} \text{ and } M = max\{m_h \mid 1 \leq h \leq 2^n\}.$$

Theorem 4.1 *An answer to the Partition problem of a set with n elements is provided by an skP system using; (i) maximal parallelism in $n+3$ steps and (ii) asynchronous or sequential behaviour in at most $n+3+M$ steps.*

Proof. The proof of the first part, (i), follows immediately from the above construction and the explanations provided.

For (ii) we make the following observations:

- The working space generation stage takes n steps as the rules $r_{2,i}$ are applied one per each compartment and no other rule is applicable in this stage.

- We replace the rule $r_{2,n+1}$ by
  $r'_{2,n+1} : X \to Y \{\prod_{1 \leq i,j \leq n; i \neq j} = B_i \neq B_j = X(\geq v_i \neq v_j \mid \neq v_i \geq v_j \mid \neq v_i \neq v_j)\}$.
  This rule replaces X by Y only when for all the pairs B_i corresponding to an element of V_1 and B_j corresponding to an element of the complement of V_1 ($= B_i \neq B_j$) all the values v_i and v_j have been paired up - either have one of them $\geq v_i \neq v_j \mid \neq v_i \geq v_j$ or none, $\neq v_i \neq v_j$.

- The verification stage consists of first applying the rules $r_{2,i,j}$ – an arbitrary number per step (asynchronous mode) or only one of them (sequential mode). Finally, when none of these rules is applicable, $r'_{2,n+1}$ is applied.
- The final stage consists of applying one of the rules r_{n+2}, r_{n+3}.

One can observe that the first and last stages take $n+2$ steps and one more step is the execution of the rule $r'_{2,n+1}$. The rules $r_{2,i,j}$ take in the compartment corresponding to the partition $V_{1,l}, V_{2,l}$, at most m_l steps. So, the overall maximum is M, according to the previous notation. This proves the result. □

4.2 Modelling the Subset Sum Problem with skP Systems

The subset sum problem can be easily derived from the partition problem. It is formulated as follows: Let V be a finite set and *weight* be a function on V with positive integer values (this is an additive function). It is requested to find, if it exists, a subset of V, denoted W, such that $weight(W) = k$. A solution to this problem is provided in [7] by using a recognizer P system with membrane creation. We use the following notations introduced above: $V = \{v_1, \ldots, v_n\}$ is a finite set, with $weight(v_i) = k_i$, where k_i is a positive integer, $1 \leq i \leq n$.

We build the following skP system (for the maximally parallel mode)

$$sk\Pi_S(n) = (A, L, IO, \mu, C'_1, C'_2, 0),$$

where A, L, IO and μ are as in $sk\Pi_P$ built for the partition problem; $C'_i = (i, w'_{i,0}, R'_i)$, $1 \leq i \leq 2$, with $w'_{1,0} = w_{1,0}$ and $w'_{2,0} = A_1 code'(n)$, where $code'(n) = v_1^{k_1} \ldots v_n^{k_n}$; $R'_1 = R_1$ and R'_2 consist of $r_{2,i}$, $1 \leq i \leq n+1$ from R_2, the rewriting rules replacing $r_{2,i,j}$, $1 \leq i < j \leq n$, are
$r'_{2,n+1+i} : v_i \to v \{= B_i = X\}, 1 \leq i \leq n$
and the rules replacing $r_{2,n+2}, r_{2,n+3}$ are
$r'_{2,2n+2} : Y \to (F, 1) \{\neq v^k\}$,
$r'_{2,2n+3} : Y \to (T, 1) \{= v^k\}$.

For the asynchronous and sequential modes the rule $r_{2,n+1}$ is replaced by $r'_{2,n+1} : X \to Y \{\prod_{1 \leq i \leq n} (= B_i \neq v_i = X \mid \neq B_i = X)\}$.

We make the following notations: if $V = \{a_1, \ldots, a_n\}$ and $weight(a_h) = k_h$, $1 \leq h \leq n$, is the weight function then $W_l = \{a_{i_1}, \ldots a_{i_{p_l}}\}$, $1 \leq l \leq 2^n$, describes a subset of V. Let us denote for each W_l, $1 \leq l \leq 2^n$

$$m'_l = \sum_{1 \leq h \leq p_l} k_{i_h} \text{ and } M' = max\{m'_h \mid 1 \leq h \leq 2^n\}.$$

One can now formulate the main result of this section.

Theorem 4.2 *A answer to the Subset Sum problem of a set with n elements is provided by an skP system using; (i) maximal parallelism in $n + 3$ steps and (ii) asynchronous or sequential behaviour in at most $n + 3 + M'$ steps.*

Proof. The proof of this theorem is very similar to the proof of Theorem 4.1 and we leave it as an exercise. □

The families of P systems solving these NP-complete problems are polynomially uniform by a Turing machine as the other solutions to these problems provided in the current literature.

5 Conclusions

In this paper, we have illustrated the modelling power of kernel P systems on two well known NP-complete problems: the Subset Sum and the Partition problems. For each of these two problems, the kP systems solutions that work in maximal parallelism, asynchronous and sequential manner are provided.

In our future work we aim to show how formal verification methods, embedded in the tools developed for this model, are utilised for checking various properties of the systems investigated.

Acknowledgments

The authors would like to thank the anonymous reviewers for their comments that helped improving the paper. MG's & FI's work is partially supported by CNCS-UEFISCDI (PN-II-ID-PCE-2011-3-0688) and MG's & SK's work by EPSRC (EP/I031812/1).

References

[1] Díaz-Pernil, D., Gutiérrez-Naranjo, M. A., Pérez-Jiménez, M. J., Riscos-Núñez, A., A Linear Time Solution to the Partition Problem in a Cellular Tissue-Like Model, *J. Comput. Theor. Nanosci.*, **7** (2010), 884 – 889.

[2] Díaz-Pernil, D., Gutiérrez-Naranjo, M. A., Pérez-Jiménez, M. J., Riscos-Núñez, A., A Linear Time Solution for the Subset Sum Problem with Tissue P Systems with Cell Division, *LNCS, Springer,* **4527** (2007), 170 – 179.

[3] Gheorghe, M., Ipate, F., Dragomir, C., A Kernel P System, *Proceedings of the 10th Brainstorming Week on Membrane Computing*, Fénix Editora (2012), 153 – 170.

[4] Gheorghe, M., Ipate, F., Dragomir, C., Mierlă, L., Valencia-Cabrera, L., García-Quismondo, M., Pérez-Jiménez, M. J., Kernel P Systems – Version 1, *Proceedings of the 11th Brainstorming Week on Membrane Computing*, Fénix Editora (2013), 97 – 124.

[5] Gheorghe, M., Ipate, F., Lefticaru, R., Pérez-Jiménez, M. J., Ţurcanu, A., Valencia-Cabrera, L., García-Quismondo, Mierlă, L., 3-Col Problem Modelling Using Simple Kernel P Systems, *Int. J. Comput. Math.*, **90** (2013), 816 – 830.

[6] Gheorghe, M., Păun, Gh., Pérez-Jiménez, M. J., Rozenberg, G., Research Frontiers of Membrane Computing: Open Problems and Research Topics, *Int. J. Found. Comput. Sci.*, **24** (2013), 547 – 624.

[7] Gutiérrez-Naranjo, M. A., Pérez-Jiménez, M. J., Romero-Campero, F. J., A Linear Solution of Subset Sum Problem by Using Membrane Creation, *IWINAC 2005, LNCS, Springer,* **3561** (2005), 258 – 267.

[8] Jiang, Y., Chen, Z., Uniform Solution to Partition Problem Using P Systems with Membrane Division, *Bio-Inspired Computing – Theories and Applications, BIC-TA 2014, CCIS, Springer,* **472** (2014), 204 – 210.

[9] Konur, S., Gheorghe, M., Dragomir, Ipate, F., Krasnogor., N. Conventional Verification for Unconventional Computing: A Genetic XOR Gate Example. *Fundamenta Informaticae*, **134** (2014), 97–110.

[10] Konur, S., Gheorghe, M., Dragomir, C., Mierla, L., Ipate, F., Krasnogor., N. Qualitative and Quantitative Analysis of Systems and Synthetic Biology Constructs Using P Systems. *ACS Synthetic Biology*, **4** (2014), 83 – 92.

[11] Marcus, S., Membranes Versus DNA, *Fundamenta Informaticae*, **49** (2002), 223–227.

[12] Păun, Gh., Computing with Membranes, *J. Comput. Syst. Sci.*, **61** (2000), 108 – 143.

[13] Păun, Gh., Rozenberg, G., Salomaa, A. (eds.), *The Oxford Handbook of Membrane Computing*, Oxford University Press, Oxford, 2010.

[14] Pérez-Jiménez, M. J., Riscos-Núñez, Solving the Subset-Sum Problem by Active Membranes, *New Generat. Comput.*, **23** (2005), 367 – 384.

[15] Song, B., Pérez-Jiménez, M. J., Pan, L., Efficient Solutions to Hard Computational Problems by P Systems with Symport/Antiport Rules and Membrane Division, *BioSystems* (submitted).

[16] Zhang, X., Wang, S., Niu, Y., Pan, L., Tissue P Systems with Cell Separation: Attacking the Partition Problem, *Science China Information Sciences*, **54** (2011), 293 – 304.

Marian GHEORGHE
Faculty of Engineering and Informatics, The University of Bradford
Bradford, BD7 1DP, UK
E-mail: `M.Gheorghe@bradford.ac.uk`

Florentin IPATE
Faculty of Mathematics and Computer Science, The University of Bucharest
Str. Academiei 14, Bucharest 010014, Romania
Email: `Florentin.Ipate@ifsoft.ro`

Savas KONUR
Department of Computer Science, The University of Sheffield
211 Portobello Street, Sheffield, S1 4DP, UK
Email: `S.Konur@sheffield.ac.uk`