International Journal of Knowledge-based and Intelligent Engineering Systems 18 (2014) 247–253
DOI 10.3233/KES-140302
IOS Press

A probabilistic search algorithm for finding suboptimal branchings in mutually exclusive hypothesis graph

M.V. Davydov
L'viv Polytechnic National University, 12 Bandery Street, L'viv, Ukraine
E-mail: maks.davydov@gmail.com

Abstract. The concept of mutually exclusive hypothesis graph (MEHG) is introduced and NP-completeness of several problems on MEHG is proved. A probabilistic search algorithm is proposed for finding suboptimal branchings in such graphs and its performance is evaluated for graphs with uniform and exponential weight distribution.

Keywords: Mutually exclusive hypothesis graph, Edmonds'algorithm, suboptimal branching

1. Introduction

A necessity to check multiple linked hypotheses arises in many computer applications that deal with ambiguity. Markov random fields [1] and dynamic programming are well known methods for ambiguity elimination over graphs that represent dependency between hypotheses. These methods can be effectively used for object tracking [2,3], object detection [4], and context-free grammar parsing [5]. However some types of problems that deal with tree constraints cannot be formulated in terms of Markov random fields, and cannot be effectively solved using dynamic programming. Moreover as shown below, problems of finding optimal spanning tree or optimal branching over a set of linked hypotheses are NP-complete. Finding solution to these problems is crucial, for example, in optical recognition of mathematical formulas [6] and the disambiguation of words meaning in non-projective languages using dependency grammars [7]. Although these problems were studied for years, no algorithms are known that solve them efficiently in case of massive ambiguity. The known solutions use time consuming exhaustive search.

In order to illustrate one of the applications of finding optimal branching, consider the problem of optical recognition of mathematical formula

$$\sqrt{x^i}$$

After connected-component labelling of this formula, five components are obtained. These components and their possible meanings are listed in Table 1.

A human can easily recognize this formula and devise its parsing tree as depicted in Fig. 1. However, a computer program needs to consider possible meanings of all symbols and devise a structure of the tree from a set of possible relations between these symbols. More importantly, the relations between symbols can form a graph with simple cycles, which means it is not a tree. The dynamic programming cannot be used in such cases.

Formally, there is a set of observations (words, symbols, etc.). Each observation may have several hypotheses about its meaning. The problem is in constructing such a rooted tree that contains one hypothesis for every observation, which maximizes the total likelihood of the proper relation between concepts that are in parent-child relationship in the tree.

The concept of mutually exclusive hypotheses graphs is introduced in this paper to target a set of problems where a tree model is deduced based on the strength of pairwise connections between possible parts of the model.

Table 1
Connected components of formula $\sqrt{x^i}$ and their possible meanings

#	Component	Possible meaning
1	√	Start of the square root operator
2	–	Ending of the square root
		Line over some symbol i.e. $\overline{x}$
		Line under some symbol i.e. $\underline{x}$
		Division operator i.e. $\frac{a}{b}$
		Part of equation sign $=$
3	.	Part of a character that contains a dot such as “i” or “j”
		Dot over some symbol such as $\dot{x}$
		Part of several dots ($\ldots$, $\cdot^{\cdot^{\cdot}}$, $^{\cdot}\cdot{}_{\cdot}$, $\ddot{x}$, $\dddot{x}$)
		End of sentence
4	ı	Part of character i
		Small character l
5	$\boldsymbol{x}$	Symbol x

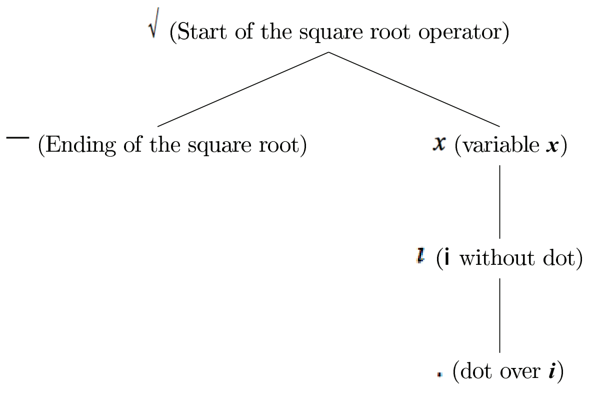
√ (Start of the square root operator)
— (Ending of the square root)
x (variable x)
ı (i without dot)
. (dot over i)

Fig. 1. The parsing tree for formula $\sqrt{x^i}$.

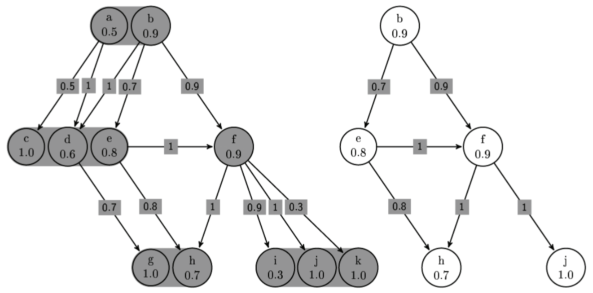

Fig. 2. Directed WMEHG (on the left) and one of its concretization (on the right). Hypothesis groups are $H_1 = \{a, b\}$, $H_2 = \{c, d, e\}$, $H_3 = \{f\}$, $H_4 = \{g, h\}$, and $H_5 = \{i, j, k\}$.

2. Problem formulation

A mutually exclusive hypotheses graph (MEHG) $G_H = (G, H)$ is a graph $G = (V, E)$ without self-loops, where V and E are finite sets of vertices and edges respectively, augmented with groups of hypotheses $H = \{H_1, H_2, \ldots, H_N\}$ that represent a partition of vertex set V. Each vertex is a hypothesis that belongs to a single group $\forall v \in V\, \exists H_i \in H : v \in H_i$, and groups of hypotheses do not intersect $H_i \cap H_j = \emptyset, i \neq j$. The dependency between hypotheses is represented by edges of graph G. Vertices that belong to the same hypotheses group can not be adjacent because they represent mutually exclusive hypotheses.

In case where dependency between hypotheses is not symmetrical, a directed mutually exclusive hypotheses graph (DMEHG) is used and set E represents arcs instead of edges.

Let *concretization* of mutually exclusive hypotheses graph G_H be its subgraph $G_C = (V_C, E_C)$ induced by set of vertices $V_C \subset V$ that contains exactly $|H|$ vertices – one from each hypothesis group. The set of all possible concretizations of G_H is denoted as $C(G_H)$.

A weighted mutually exclusive hypothesis graph (WMEHG) is a mutually exclusive hypothesis graph G_H with vertex and edge weighting functions $w_V : V \to \mathbb{R}$ and $w_E : E \to \mathbb{R}$. Figure 2 illustrates an example of directed WMEHG and one of its concretizations.

The interpretation of weights highly depends on the application. In practice, weights are often selected as values of logarithm of Bayesian probability

$$w_V(v) = \ln P(v \textit{ represents true hypothesis}),\quad v \in V,$$

$$w_E(e) = \ln P(e \textit{ links concepts that are in parent-child relationship}), e \in E.$$

In this case the resultant tree weight is a logarithm of its Bayesian probability that ensures maximum probability for trees with maximum weight.

The common optimization problem on MEHG is the problem of finding a concretization of MEHG that satisfies certain criteria. The four problems for MEHG and WMEHG, that are the main subject of this paper, are defined below.

Problem 1. Given an undirected MEHG G_H find any of its concretizations $G_C \in C(G_H)$ that is connected.

Problem 2. Given a directed MEHG G_H find its concretization that contains branching. A branching is a rooted directed tree that covers all vertices of the graph.

Problem 3. Given an undirected WMEHG G_H find its concretization that contains a spanning tree with a maximum of total weight of vertices and edges

$$G_{MST} \in \underset{G_C \in C(G_H)}{Argmax} \sum_{v \in V(G_C)} w_V(v) + \sum_{e \in MST(G_C)} w_E(e),$$

where $V(G_C)$ denotes a set of all vertices of graph G_C and $MST(G_C)$ denotes one of the maximum spanning trees of G_C (i.e. spanning tree with the largest sum of edge weights).

Problem 4. Given a directed WMEHG G_H find its concretization that contains branching of the maximum weight

$$G_{MB} \in \underset{G_C \in C(G_H)}{Argmax} \sum_{v \in V(G_C)} w_V(v) + \sum_{e \in MB(G_C)} w_E(e),$$

where $MB(G_C)$ denotes one of the maximum branchings on G_C, i.e. rooted tree that contains all vertices and has the maximum sum of vertex and edge weights.

The NP-completeness of the defined problems is proved in Lemma 1.

Lemma 1. All of the Problems 1–4 are NP-complete.

Proof The boolean 3-satisfiability problem (3-SAT) is known to be NP-complete [8]. Let's show that it can be reduced in polynomial time to Problem 1. First create hypothesis groups of two literals x_i and $\neg x_i$ for each boolean variable x_i and a hypothesis group of n vertices for each n-variable clause. Then connect all literal vertices in such a way that they form a complete subgraph, and connect all literals to clauses where they are used. An example of a MEHG for expression $(x_1 \vee x_2 \vee x_3) \wedge (\neg x_1 \vee x_2)$ is depicted in Fig. 3. The logical formula can be satisfied iff the connected con-

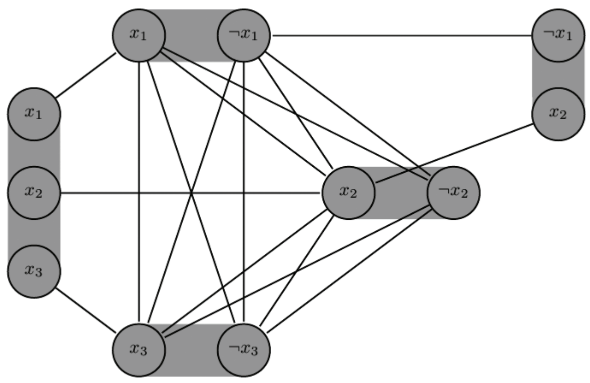

Fig. 3. MEHG obtained for boolean expression $(x_1 \vee x_2 \vee x_3) \wedge (\neg x_1 \vee x_2)$.

cretization of MEHG can be found. In order to prove this, assign values to boolean variables in such a way, that the literals of selected vertices become true. The connectedness of the graph implies that each clause contains a positive literal. This means that boolean formula is satisfied. On the other hand, when the formula can be satisfied then the connected concretization can be obtained by selecting positive literals in MEHG.

This concludes the proof that the 3-SAT problem is reducible to Problem 1 in polynomial time, which in turn proves that Problem 1 is NP-complete.

Problem 2 is NP-complete, because the Hamiltonian path problem in directed graphs can be reduced to it. Hamiltonian path is a path that contains each vertex of the graph exactly once. The Hamiltonian path problem is a problem of determining whether Hamiltonian paths exist in a given graph and this problem is known to be NP-complete [8].

The reduction from the Hamiltonian path problem to Problem 2 can be achieved by substituting the graph that is to be tested for existence of Hamiltonian paths with directed MEHG. In order to do this, substitute each graph vertex $v_i \in V$ that has P outgoing edges $(v_i, u_1), (v_i, u_2), \ldots, (v_i, u_P)$, with a hypotheses group that contains P hypotheses $v_{i1}, v_{i2}, \ldots, v_{iP}$ in case of $P > 0$ or one hypothesis v_{i1} otherwise. Every edge (v_i, u_j), $j = \overline{1, P}$ should be substituted with $Q(u_j)$ edges (v_{ij}, u_{jk}) in MEHG, $k = \overline{1, Q(u_j)}$, where $Q(u_j)$ is the number of hypotheses $u_{j1}, u_{j2}, \ldots, u_{jQ(u_j)}$ that replace vertex u_j of the original graph (i.e. $Q(u_j)$ is the number of outgoing edges of vertex u_j or 1 if it has no outgoing edges). It is obvious that $P < |V|$ and $Q(u_j) < |V|$ because the number of edges outgoing from any vertex of the original graph is less than the total number of its vertices. An example of the directed graph transformation into

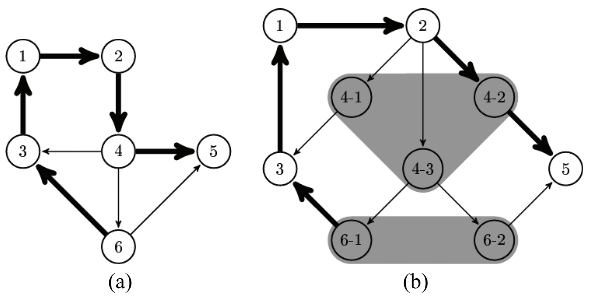

Fig. 4. The reduction of the problem of finding Hamiltonian path in a directed graph (a) to the problem of finding concretization that contains branching in MEHG (b).

MEHG is illustrated in Fig. 4. Please note that vertices 4 and 6 that have two or more outgoing edges are substituted by several vertices in MEGH, and the number of hypothesis groups in MEHG is equal to the total number of vertices in the original graph.

Now all edges outgoing from a single vertex of this MEGH have their endpoints in the same hypothesis group. This means that each vertex of any of its concretization can have only one outgoing edge, so if some concretization of MEHG has branching that contains all vertices then this branching is the Hamiltonian path. On the contrary, the Hamiltonian path in the original graph can be used to obtain concretization of MEHG that contains branching, which covers all vertices. This can be done by selecting hypotheses that correspond to the outgoing edges in Hamiltonian path.

The number of vertices in the resultant MEHG is bounded by $|V|^2$ and the number of edges by $|V| \cdot |E|$, where $|V|$ is the total number of vertices and $|E|$ is the total number of edges in the original graph. Thus the problem of finding Hamiltonian path in directed graphs can be reduced in polynomial time to the problem of finding concretization of MEHG that contains branching.

Problems 1 and 2 are partial cases of 3 and 4. Specifically, the following holds when $w_V(v) = 0$ and $w_E(e) = 1$:

1. The problem of concretization existence that contains a spanning tree with a weight of at least $|V| - 1$ (Problem 3) is equivalent to the problem of connected concretization existence of MEHG (Problem 1).
2. The problem of finding concretization that contains branching weighted at least $|V| - 1$ (Problem 4) is equivalent to the problem of verifying existence of concretization that contains branching that covers all vertices (Problem 2).

This proofs that Problems 3 and 4 are also NP-complete.

3. A stochastic search algorithm for finding sub-optimal branchings in directed WMEHG

Now that the NP-completeness of the problem of finding optimal branching in directed WMEHG is proved, a probabilistic algorithm for finding sub-optimal branchings is introduced and its performance is evaluated.

The proposed probabilistic algorithm uses two optimization approaches. The first finds the optimal branching B_C in some concretization $G_C \in C(G_H)$, of weighted MEHG G_H using Gabow et al. implementation of Edmonds' algorithm, known to run in $O(|V|\log|V|+|E|)$ time [9]. The second optimization approach maximizes the weight of branching B_C by changing the selection of vertices from each hypothesis group.

This algorithm is described below. For notation convenience branching B_C is represented as a mapping $B_C : V_C \to 2^{V_C}$ that maps each vertex to a set of its child vertices in the branching tree. Leaf vertices are mapped to an empty set.

Algorithm $\mathit{FixedTreeAscent}(G_H, B_C)$.

Input. Directed WMEHG G_H, branching B_C: $V_C \to 2^{V_C}$ over some concretization $G_C \in C(G_H)$.

Output. A concretization of G_H that has branching of highest possible weight with the same tree structure as B_C, and the weight of the maximum found branching.

Step 1. Convert branching B_C into tree B_H over H, which is defined as a function $B_H : H \to 2^H$:

$$B_H(h(v_i)) = \bigcup_{v \in B_C(v_i)} \{h(v)\}, v_i \in V_C,$$

where $h : V \to H$ is a mapping from any vertex from G_H to its hypothesis group.

Step 2. Use depth-first post-order traversal of all hypothesis groups in tree B_H to calculate the maximum possible weight $M(v)$ of branching over vertices in descendant hypothesis groups. $M(v)$ gets calculated for each vertex $v \in V$:

$$M(v) = \begin{cases} -\infty, \mathit{if} \exists H_T \in B_H(h(v)) \forall v_i \\ \in H_T : M(v) = -\infty \vee (v, v_i) \notin E \\ w_V(v) + \sum_{H_T \in B_H(h(v))} \max\limits_{v_i \in H_T} \\ (M(v_i) + w_E(v, v_i)), \mathit{otherwise} \end{cases}$$

Summation is done by all child hypothesis groups of group $h(v)$ in tree B_H. The depth-first post-order traversal guarantees that $M(v)$ gets calculated for all child groups prior to calculating it for vertex v.

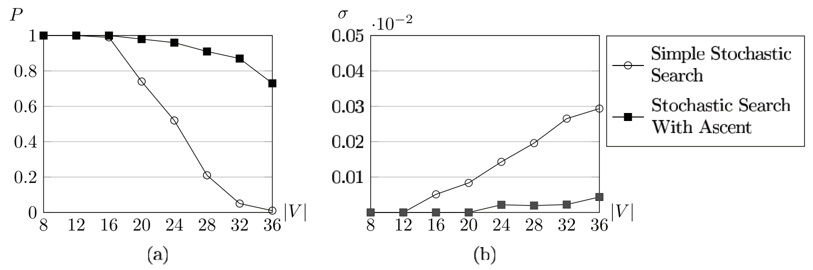

Fig. 5. The mean probability P of finding the optimal solution (a) and the relative root-mean-square deviation σ from the optimal solution (b) for the developed *StochasticSearchWithAscent* algorithm and simple stochastic search.

Step 3. Determine the root of the found maximum branching using formula

$$v_R = \underset{v \in H_R}{argmax} M(v),$$

where H_R is the root in tree B_H. In the worst case v_R will be the root of B_C and $M(v_R)$ will be the weight of branching B_C. If there are several candidates for the maximum, one of them that is not root of B_C is chosen randomly.

Step 4. Obtain new concretization $\tilde{V_C} = D(v_R)$ by calculating recursive formula

$$D(v) = \{v\} \cup \bigcup_{H_T \in B_H(h(v))} D\left(v_{max}(v, H_T)\right),$$

$$v_{max}(v, H_T) = \underset{v_i \in H_T, M(v_i) \neq -\infty, (v, v_i) \in E}{argmax} \left(M(v) + w_E(v, v_i)\right)$$

that selects hypotheses that maximize the weight of the branching. It is a back propagation step of the dynamic programming.

Step 5. Return concretization $\tilde{G_C} = (\tilde{V_C}, E \cap (\tilde{V_C} \times \tilde{V_C}))$ and the weight of the found maximum branching $M(v_R)$.

The computational complexity of the $FixedTree$ $Ascent$ algorithm is not more that $O(|V|^2)$. This is because step 2 can be calculated in time

$$O\left(\sum_{H_{TR} \in H} \sum_{H_T \in B_H(H_{TR})} |H_{TR}||H_T|\right) < O\left(\left(\sum_{i=1}^{N} |H_i|\right)^2\right) = O(|V|^2),$$

and all other steps can be calculated in time less than $O(|V|^2)$.

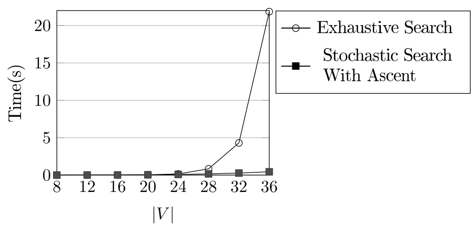

Fig. 6. The execution time of the developed algorithm in comparison to the exhaustive search among all possible concretization.

The final stochastic search algorithm for finding sub-optimal branchings in WMEHG G_H is defined as follows.

Algorithm $StochasticSearchWithAscent$ (G_H, K).

Input. Directed WMEGH G_H, the number of random trials K.

Output. Concretization of $G_{\max} \in C(G_H)$, maximum branching of which is the highest in the processed search space.

Initialization. Initialize a set of processed concretizations $G_{PR} := \emptyset$, the weight of the found maximum branching $w_{\max} := 0$, and concretization that contains the found maximum branching $G_{\max} := \emptyset$.

Step 1. Randomly select concretization $G_C \in C(G_H) \setminus G_{PR}$ and add it to set G_{PR}.

Step 2. Search for maximum branching B_C in G_C using Edmonds' algorithm.

Step 3. If B_C is not found then continue to step 10.

Step 4. If $w(B_C) > w_{\max}$ then make assignments $w_{\max} := w(B_C)$, $G_{\max} := G_C$, where $w(B_C)$ denotes the weight of branching B_C.

Step 5. $(\tilde{G_C}, w) = FixedTreeAscent(G_H, B_C)$ – find assignment of hypotheses, that maximizes branching B_C.

Step 6. If $w > w_{\max}$ then make assignments $w_{\max} := w$, $G_{\max} := \tilde{G_C}$.

Step 7. If $\tilde{G_C} \in G_{PR}$ then continue to step 10.

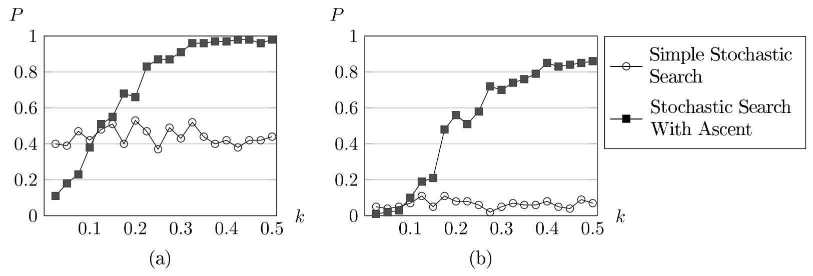

Fig. 7. The mean probability P of finding the optimal solution as a function of the directed graph density $k = \frac{E}{|V|\cdot|V|}$ for randomly generated graphs with $|V| = 24$ (a) and $|V| = 32$ (b).

Step 8. Set $G_{PR} := G_{PR} \cup \{\tilde{G_C}\}$. If $|G_{PR}| \geqslant K$ then return $G_{\max}$.

Step 9. Set $G_C := \tilde{G_C}$ and proceed to step 2.

Step 10. If $|G_{PR}| \geqslant K$ then return $G_{\max}$; otherwise proceed to step 1.

The computational complexity of $StochasticSearch WithAscent$ algorithm is $O(K|V|^2)$, where K is the number of trials and $|V|$ is the number of vertices in graph G_H. This is because the number of cycles is limited by K and the complexity of each cycle is the sum of the complexity of Edmonds' algorithm and the complexity of $FixedTreeAscent$ algorithm, that is $O(|V|\log|V| + |E|) + O(|V|^2)$.

4. Experimental results

While it is possible to devise a directed WMEHG where the proposed algorithm most likely would not find any concretization that has branching, the algorithm performance was tested for graphs with uniform and exponential distribution of arc weights in connection to various values of the directed graph density $k = |E|/|V|^2$.

The performance of the $StochasticSearchWith Ascent$ algorithm was compared to the $Stochastic SearchSimple$ algorithm that simply lacks tree ascent steps 5–9 of the $StochasticSearchWithAscent$ algorithm. In order to compensate the lack of these steps, the maximum number of trials in the $StochasticSearch Simple$ algorithm was increased to make the processing time equal for both algorithms. The number of vertices in each hypothesis group was set to 4 and experiments where conducted for $|V| = 8, 12, \ldots 36$. The number of trials was limited by $\min(4^{|H|}, |V|^2)$ in the $StochasticSearchWithAscent$ algorithm and by $\min(4^{|H|}, 4|V|^2)$ in the $StochasticSearchSimple$ algorithm that ensures the almost identical mean execution time for both algorithms.

The result of experiments for $k = 0.4$ is depicted in Fig. 5 for uniform weight distribution. In case of an exponential weight distribution, the relative root-mean-square deviation from the optimal solution is increased twice for both $StochasticSearchWith Ascent$ and $StochasticSearchSimple$ algorithms. The time reduction is significant for $|V| \geqslant 28$ as shown in Fig. 6 where execution time of the $StochasticSearch WithAscent$ algorithm is compared to the execution time of the exhaustive search.

The mean probability of finding the optimal solution as a function of directed graph density $k = |E|/|V|^2$ for $|V| = 24$ and $|V| = 32$ is depicted in Fig. 7. These results show that the $Stochastic SearchWithAscent$ algorithm yields better performance than the $StochasticSearchSimple$ algorithm for $k \geqslant 0.15$, and if k is close to zero algorithm fails to find any optimal solution.

5. Discussion

The developed heuristic algorithm represents an alternative to the exhaustive search for optimal branching that guarantees reduced computational time. Application of this algorithm is recommended in computer vision and linguistic applications that would greatly benefit from the reduced processing time. The developed algorithm proved to be stable in case of uniform and exponential weight distributions. The performance of the $StochasticSearchWithAscent$ algorithm is better than the performance of the $StochasticSearch Simple$ algorithms for directed graph density $0.15 \leqslant k \leqslant 1.0$. Further research of the developed approach should concentrate on the problem of finding solutions under the condition of $k < 0.15$.

References

[1] J. Moussouris, Gibbs and Markov random systems with constraints, *Journal of Statistical Physics* **10**(1) (1974), 11–33.

[2] A. Torabi and G.-A. Bilodeau, A Multiple Hypothesis tracking method with fragmentation handling, *Computer and Robot Vision, CRV'09 Canadian Conference on* (25–27 May 2009), 8–15.

[3] M. Antenreiter and P. Auer, A reasoning system to track movements of totally occluded objects, *Workshop Proceedings at ECCV 2006 (electronic, url:eprints.pascal-network.org/archive/00002379/01/icvw06.pdf)*, Graz, Austria.

[4] D. Zhang and S. Chang, Learning to detect scene text using a higher-order MRF with belief propagation, *Computer Vision and Pattern Recognition Workshop, Conference on* (2004), 101–108.

[5] A. Maletti and G. Satta, Parsing algorithms based on tree automata, in: *Proceedings of the 11th International Conference on Parsing Technologies (IWPT'09)*, Association for Computational Linguistics, Stroudsburg, (2009), 1–12.

[6] M. Suzuki, T. Kanahori, N. Ohtake and K. Yamaguchi, An integrated OCR software for mathematical documents and its output with accessibility, in: *Springer Berlin Heidelberg: Computers Helping People with Special Needs*, Lecture Notes in Computer Science, (2004), 648–655.

[7] R. McDonald, F. Pereira, K. Ribarov and J. Haji, Non-projective dependency parsing using spanning tree algorithms. in: *HLT'05 Proceedings of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing* (2005), 523–530.

[8] M.R. Garey and D.S. Johnson, Computers and intractability: A guide to the theory of NP-completeness, W.H. Freeman, (1979), 259–260.

[9] H.N. Gabow, Z. Galil, T. Spencer and R.E. Tarjan, Efficient algorithms for finding minimum spanning trees in undirected and directed graphs, in: *Combinatorica* **6**(2) (1986), 109–122.