Intelligent Tutorial Planning Based on Extended Knowledge Structure Graph*

Zhuohua Duan[1,3], Yunfei Jiang[2], and Zixing Cai[3]

[1] Department of Computer, School of Information Engineering, Shaoguan University, 512003, Shaoguan, Guangdong, China
duanzhuohua@163.com
[2] Institution of Software, SUN YAT-SEN University, 510000, Guangzhou, Guangdong, China
[3] College of Information Science and Engineering, Central South University, 410083, Changsha, Hunan, China

Abstract. Intelligent tutorial planning (ITP) is an important component of intelligent tutorial system (ITS). Models of domain knowledge, models of tutorial methods and models of learner are three key elements of ITS. In this paper, the concept of extended knowledge structure graph (EKSG) is presented. An EKSG integrates models of domain knowledge, models of tutorial methods and models of learner organically. Based on the EKSG, algorithms JUDGE and TPLAN are put forward to resolve ITP problems. The algorithm JUDGE calculates the optimal solution graph when there is a solution, and the algorithm TPLAN calculates optimal tutorial plan based on the solution graph. Both algorithms are proved to be correct, the efficiency of them is also discussed.

1 Introduction

Intelligent tutoring system (ITS) typically consists of a set of correlative coursewares, each courseware fulfills teaching task of one or more knowledge points. Planning is a problem-solving method which produces an action sequence (i.e. plan) to achieve the objective [1]. Intelligent tutorial planning (ITP) tailors learning plan for a given student according to domain knowledge structure, teaching methods and cognitive level of the student. It combines a set of coursewares organically, provides individual learn path for learner, and is a key module of ITS.

Earlier researches on planning concentrate on physical actions of robots[1]. Peachey et al. firstly use planning techniques to construct intelligent tutorial system [2]. Henceforth, tutorial planning has received extensive attentions in the field of computer-aided instruction (CAI). Based on blackboard architecture, Murray proposes the concept of dynamic instructional planning which has the ability to generate, monitor, and revise instructional plans during the course

* This work is partially supported by the National Natural Science Foundation of China Grant #60234030 and Master Excellent Course Project of China and CSU.

Z. Pan et al. (Eds.): Edutainment 2006, LNCS 3942, pp. 7–16, 2006.

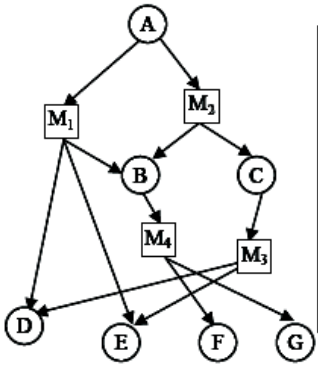
of instruction [3]. Katz et al. point out that there are two criteria for effective automatic curriculum planning, i.e. cognitive and motivational: (1) selection of tasks that are at the appropriate difficulty level for the student; i.e., challenging, but not frustrating and (2) enough variety in the chosen tasks to sustain the student's interest [4]. In [5], the authors present and investigate the application of multi-level planning techniques, together with meta and micro level knowledge architecture model, in building the domain knowledge, planning and navigating the curriculum of an intelligent tutoring system. Elorriaga points out that the ability to adapt the instruction to the student is based on two issues: the learner model and the instructional planning [6]. Martens presents a web- and case-based planning agents in intelligent tutoring [7]. Li et al. propose a layered tutorial planning scheme, including global tutorial planning, tutorial scheme scheduling, and local tutorial planning within scheme [8].

Researchers realized that it is critical for ITS to combine domain knowledge, student model, teaching strategy and their relationship in a systematical way [9,11]. Literature [9] points out that a courseware contains three key components, namely, domain knowledge model, student model and teaching strategy model. JIANG firstly combines the three key components to solve the problem of tutorial planning [10]. He proposes the concept of knowledge structure graph (KSG), and presents an ITP framework and algorithms based on KSG. In a KSG, the relationship between different knowledge points is represented with an AND/OR graph with weighted k-arcs. A k-arc in KSG represents several teaching methods for a knowledge node implicitly. The main drawback of KSG is that teaching methods are difficult to maintain in KSG and it is difficult to represent the relationship between student model and teaching methods. To handle this problem, this paper proposes the concept of extended knowledge structure graph (EKSG). In an EKSG, teaching methods and their relationships with domain knowledge and students are represented explicitly. It will be showed in Section 2 that the three key components of ITS, namely, domain knowledge model, teaching strategy model and learner model, are combined organically in EKSG. Based on EKSG, the intelligent tutorial planning algorithms are proposed. The correctness of the algorithms is proven, and the efficiency of the algorithms is analyzed.

2 Extended Knowledge Structure Graph

A good knowledge representation model for ITS must be able to represent the relationships of domain knowledge, teaching strategies and student models. In this paper, the concept of EKSG is presented to serve as knowledge representation model for ITS for the task of tutorial planning.

Definition 1. A directed acyclic graph (DAG) is called as an extended knowledge structure graph (EKSG), if and only if it satisfies the following conditions:



Students / methods	L_1	L_2
M_1	3	100
M_2	100	2
M_3	2	2
M_4	2	2

Fig. 1. An extended knowledge structure graph

- It contains two types of nodes: knowledge nodes and method nodes;
- The next nodes of a knowledge node are method nodes, which represent several optional methods for learning the knowledge node. The next nodes of a method node are knowledge nodes, which represent all the requisite knowledge needed to apply this method;
- Several weights are assigned to each method node. Each weight represents the cost needed for a certain type of learner to use this method.

EKSG have two important features. Firstly, the teaching methods are represented with nodes. So the information of method nodes is easy to be managed. Secondly, domain knowledge, teaching strategies, and learner models are integrated into an EKSG systematically.

An example of EKSG is shown in Fig. 1. Knowledge nodes are denoted with circles, and teaching method nodes are denoted with rectangles. There are 6 knowledge nodes, namely, 'A', 'B', 'C', 'D', 'E', 'F' and 'G', and 4 teaching method nodes, namely, 'M_1', 'M_2', 'M_3' and 'M_4'. 'L_1' and 'L_2' denote two kinds of learners. There are two methods for learning knowledge 'A", namely, 'M_1' and 'M_2'. When 'M_1' is applied, the learner must have mastered the knowledge 'B', 'D', and 'E'.

The costs of teaching methods using by different learners are showed in a two-dimensional table, as shown in Fig. 1. For example, the cost for student 'L_1' to learn the knowledge 'A' with method 'M_1' is 3.

For the sake of clarity, let bs(M) denote the set of next nodes of 'M', g(M) denote the previous node of 'M', c(L,M) denote the cost for learner 'L' using method 'M'. For instance, in Fig. 1, bs(M_1)={B,D,E}, g(M_1)=A, c(L_1,M_1)=3.

3 Tutorial Planning Problems and Their Representation

A tutorial planning problem (TPP) must provide the following information: an EKSG, a learning goal (G), a set of basic knowledge (BS) and a learner model

(L). Usually, the EKSG has been constructed by domain experts, and all the learners share a global EKSG.

Definition 2. A tutorial planning problem (TPP) defined as a triple (G, BS, L), where G denotes the learning goal, BS denotes a set of knowledge which is mastered by the learner L, and L denotes the learner who submits the planning problem.

Definition 3. A tutorial planning (TP) of a TPP (G, BS, L), denoted by TP(G,BS,L), is a method sequence satisfying

1) if G∈BS, then TP=( );
2) if G∉BS, then TP=(M_1,M_2,...,M_N), where g(M_N)=G, bs(M_i) $\subseteq BS \cup (\bigcup_{1\le j<i} g(M_j))$ (1≤i≤N).

Definition 4. The cost of a tutorial planning TP(G,BS,L), denoted by c(TP(G,BS,L)) (or c(TP)), is defined as c(TP)= c(TP(G,BS,L))=$\sum_{i=1}^{N} c(L, M_i)$, where TP(G,BS,L)= (M_1,M_2,...,M_N).

Definition 5. TP*(G, BS, L) is the optimal tutorial planning if c(TP*) <=c(TP) for every other tutorial planning TP(G,BS,L).

For example, let TPP1=(A, {D, E, F, G}, L_1) to be a tutorial planning problem, and the EKSG is as shown in Fig 1.

It can be shown that there are two tutorial planning for TPP1, i.e. TP1(TPP1)=(M_4, M_1) and TP2(TPP1)=(M_3, M_4, M_2). A tutorial planning can be represented with a solution graph, which is a subgraph of the EKSG.

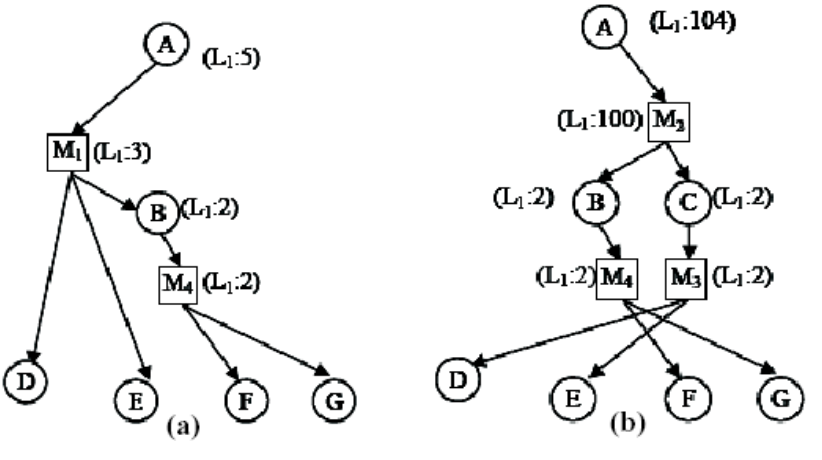


Fig. 2. Two solution graphs

For example, the solution graphs of TP1 and TP2 are shown in Fig. 2(a) and (b) respectively.

Planning TP1(TPP1) consists of two methods: M_4 and M_1, this means that the learner uses M_4 firstly and then uses M_1 can achieve the goal. The cost of learning knowledge B is 2, and the cost of learning knowledge A is 5.

Fig. 2 shows that the tutorial planning (M_4, M_1) is better than (M_3, M_4, M_2) for learner L_1.

4 Intelligent Tutorial Planning Algorithm

For a given planning problem, tutorial planning does not always exist due to incompleteness of EKSG and/or insufficiency of the learner's basic knowledge. Under this condition, the learner cannot achieve the goal G on the basis of BS and EKSG. For example, tutorial problem (A, {D, E, F}, L_1) and (A, {E, F, G}, L_2) have no solution. For this reason, for a given planning problem, the planning process consists of two steps:

1. Step JUDGE, which judges whether or not the problem has a solution, and output the optimal solution graph if a solution exists.
2. Step TPLAN, which computes optimal planning according to the optimal solution graph produced by JUDGE.

For convenience, let cl(BS,L) denote the set of knowledge nodes which can be obtain by L on the basis of BS and clg(G,BS,L) denote the set {K|K∈cl(BS,L), and K is not ancestor of G}. For example, cl({D, E, F, G}, L_1) = {D, E, F, G, B, C, A}, cl({F, G}, L_1) = {F, G, B}, clg(A, {D, E, F, G}, L_1)= {D, E, F, G, B, C, A}, clg(B, {D, E, F, G}, L_1) = {D, E, F, G,B, C}.

4.1 Algorithm JUDGE

Algorithm JUDGE judges whether or not the planning problem (G, BS, L) has a solution and computes the optimal solution graph SUB_EKSG if there exists a solution and computes the cost of each node. The algorithm is described as following.

```
Algorithm 1 SUB_EKSG=JUDGE((G, BS, L), EKSG)
  {Input: A tutorial planning problem (G, BS, L);
  An extended knowledge structure graph EKSG
  Output: A weighted optimal solution SUB_EKSG with root
  G and leafs BS if there exists a solution. If there
  has no solution, SUB_EKSG is set as ∅.}
  begin
    1. Let Base=BS, SUB_EKSG=BS;
    2. If (G∈ BS) {SUB_EKSG={G}; return SUB_EKSG;}
    3. Mark all the knowledge in Base 'unhandled',
       and set count=0;
```

4. If (there has a 'unhandled' knowledge node in Base) {let K_n be the first 'unhandled' node in Base; goto step 5} else {if count=0 goto step 10 else goto step 3; }
5. Mark K_n as 'handled', let $M_{Kn}=\{M \mid K_n \in bs(M)$ and $M \notin SUB\_EKSG\}$. If $M_{Kn}=\varnothing$ goto 4 else mark all the method nodes in M_{Kn} as 'unhandled';
6. While (there has a 'unhandled' method node in M_{Kn}) {let M be the first 'unhandled' node in M_{Kn}; if ($bs(M) \subseteq$ Base-{G}) then mark M as 'handled' else delete M from M_{Kn} ; }
7. If $M_{Kn}=\varnothing$ goto step 4;
8. For every method node M in M_{Kn}, let $K_M = g(M)$, if ($K_M \notin$ Base)
   {let count=count+1, add K_M into Base, add M and K_M into SUB_EKSG, mark K_M as 'unhandled',
   set $c(K_M)=c(M, L)+\sum_{K \in bs(M)} c(K)$ }
   if $K_M \in$ Base and $c(M, L)+\sum_{K \in bs(M)} c(K) < c(K_M)$
   {delete the original method node of K_M in SUB_EKSG, add M into SUB_EKSG,
   let $c(K_M)=c(M, L)+\sum_{K \in bs(M)} c(K)$ }
9. Goto step 4;
10. If (G $\in$ Base) return SUB_EKSG else return $\varnothing$;

end.

Several comments about JUDGE: It computes clg(G, BS, L) from step 3 to 9. The counter 'count' is used to determine whether or not new knowledge nodes have been added to clg(G, BS, L). If count is zero in a loop, then clg(G, BS, L) will not increase. In step 8, only the optimal method node has been preserved. Hence, the algorithm always produces the optimal solution graph if it exists.

JUDGE constructs the optimal solution graph from bottom to top, i.e. from leaf to root. For example, the process of solving the planning problem TPP1=(A, {D, E, F, G}, L_1) is illustrated following.

The first loop extends node 'D' (see Fig. 3(a)). There are two methods which take the knowledge 'D' as basic knowledge, namely, 'M_1' and 'M_3'. The basic knowledge nodes of M_1 is {D, E, B}, but 'B' is not in the set *Base*, so 'M_1'can't be applied according to the current basic knowledge. The basic knowledge nodes of 'M_3' is {D, E}, 'D' and 'E' are all in the set *Base*, so 'M_3' can be applied according to current basic knowledge. Hence, the knowledge 'C', which is the previous node of 'M_3', is added to *Base*. 'M_3' and 'C' are added to SUB_EKSG. The SUB_EKSG after extending the node 'D' is shown in Fig. 3(a).

The 2^{nd} loop extends node 'E', and no new nodes have been added into *Base* and SUB_EKSG, for 'M_3' is already in SUB_EKSG, and 'M_1' can't be applied for the same reason as expressed above.

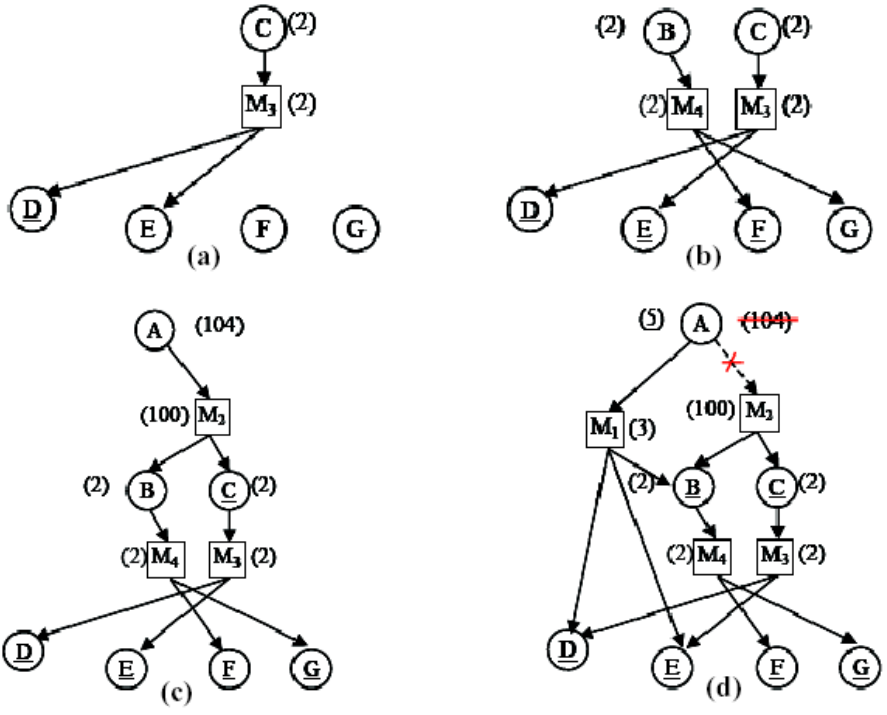


Fig. 3. Generating the solution graph. (a) Extend knowledge node D; (b) Extend knowledge node E; (c) Extend knowledge node F; (d) Extend knowledge node B.

The 3^{rd} loop extends node 'F', and the node 'B' is added into *Base*. 'B' and 'M_4' are added into SUB_EKSG. The SUB_EKSG after extending the node 'F' is shown in Fig. 3(b).

The 4^{th} loop extends node 'G', and the SUB_EKSG is not changed.

The 5^{th} loop extends node 'C', and the SUB_EKSG is shown in Fig. 3(c). It shows that the cost of learning 'A' with this planning is 104. Although we got a solution, we are not sure whether or not it is the optimal one, for there has some 'unhandled' node, such as 'B'.

So, the 6^{th} loop extends node 'B', and the SUB_EKSG is shown in Fig. 3(d). This loop shows that 'M_1' is a better method for learning 'A'. For this path, the cost of learning 'A' is only 5, much less than the previous cost, 104.

Now, all nodes except the goal ('A') in *Base* are extended. So the loop is over, and we get the optimal solution graph.

4.2 Algorithm TPLAN

When JUDGE produces a nonempty optimal solution graph SUB_EKSG, TPLAN creates the optimal tutorial planning. It employs a reverse-reasoning

approach, searching the optimal solution graph from root G to leaf BS. The algorithm is shown as follows.

```
Algorithm 2 PLAN=TPLAN ((G, BS, L), SUB_EKSG)
  {Input: A tutorial planning problem (G, BS, L);
  A optimal solution SUB_EKSG=JUDGE(G, BS, L);}
  Output: The optimal tutorial planning PLAN
  begin
  1. Let PLAN=(), OPEN=( G );
  2. If  (G∈ BS) return;
  3. while  (OPEN≠ ∅ )
  4.  {Let K be head node of OPEN;
  5.   delete K from OPEN;
  6.   if (K∉ BS)
  7.    { Let M be next node of K in SUB_EKSG;
  8.      Insert M into head of PLAN;
  9.      Insert all next nodes of M into OPEN;
  10.     Sort nodes on OPEN list by descending cost;
  11.     }
  12.  }
  end.
```

Several comments about TPLAN: PLAN is the optimal planning, OPEN is a list of nodes which are needed to be extended sorted by descending cost. TPLAN computes the tutorial planning conversely. Namely, it searches from the goal to the bases. It extends the most expensive node (i.e. the goal) firstly.

5 Correctness and Effectiveness of the Algorithms

This section proves that the algorithm JUDGE and TPLAN are correct and analyzes their efficiency.

Lemma 1. Algorithm JUDGE will terminate, and satisfies the condition of Base= clg(G,BS,L) when it terminates.

Proof. The main loop of JUDGE is from step 3 to step 9, this loop extends the set *Base* iteratively, and terminates when the set *Base* does not increase. Because EKSG is finite, JUDGE will terminate certainly. The proof of 'Base= clg(G,BS,L) when JUDGE terminates' consists of two steps: 1) clg(G, BS, L)⊆ Base; 2)*Base*⊆clg(G,BS,L):

1) If exist a knowledge node K ∈ clg(G,BS,L) , and K∉*Base* when JUDGE terminates. However, knowledge node K can be learned based on Base according to the definition of clg(G, BS, L). Hence, the main loop will not terminate. This contradicts the hypothesis. So, clg(G, BS, L)⊆Base when the loop terminates.
2) If exist a knowledge node K∈*Base*, and K∉ clg(G, BS, L). However, knowledge node K can be learned based on Base according to the function of the loop. So, K ∈ clg(G,BS,L) according to the definition of clg(G,BS,L). This contradicts the hypothesis. So, Base⊆clg(G,BS,L).

Theorem 2. JUDGE produces the optimal solution graph when there have solutions, and produces ∅ when there has no solution.

Proof. Obviously, G∈*Base* denotes that there has a solution. Otherwise, there has no solution. According to the step 10, JUDGE produces ∅ when there has no solution. The proof of 'SUB_EKSG is the optimal solution graph when G ∈ *Base*' is shown as following.

According to the step 8, when there exist several methods available to a knowledge node, JUDGE always deletes the more expensive one and adopts the cheaper one. Hence, JUDGE uses the optimal methods for every knowledge node. This assures that the cost of the goal knowledge node is also the cheapest.

Theorem 3. Algorithm TPLAN gives the optimal planning PLAN according the optimal solution graph SUB_EKSG.

Theorem 4. The time complexity of algorithm TPLAN is O(|clg(G,BS,L)|).

Theorem 3 and 4 are obviously satisfied. We omit their proofs.

Theorem 5. The time complexity of algorithm JUDGE is O(|clg(G,BS,L)-BS|*|clg(G,BS,L)|*n), where n= max { $|M_{Kn}|$ } (K_n is a knowledge node, and K_n ∈SUB_EKSG, the definition of M_{Kn} is shown in the step 5 of the algorithm JUDGE).

Proof. Algorithm has 3 levels of loop. The main loop is from step 3 to step 9. If *Base* is not extended completely, the knowledge number of *Base* increases at least 1 at every loop. So the running times of main loop d <= |clg(G,BS,L)-BS|. The second level of loop is from step 4 to step 9, the running times is |*Base*|, it is obvious that |*Base*| <= |clg(G,BS,L)|. The inner loop is from step 6 to step 9, the running times is $|M_{Kn}|$. So the overall time is O(|clg(G,BS,L)-BS|*|clg(G,BS,L)|*n).

6 Conclusions

This paper proposes the concept of extended knowledge structure graph (EKSG). EKSG integrates domain knowledge, tutorial methods, and learner's models rigorously. Based on EKSG, the planner can take into account the student, the teaching materials, and the tutorial methods simultaneously. Additionally, the tutorial methods are explicitly expressed in EKSG, so the planner can maintain the tutorial methods efficiently and conveniently.

According to EKSG, two algorithms, namely JUDGE and TPLAN are proposed. JUDGE determines whether or not a planning problem has a solution, and produces the optimal solution graph when there has a solution. TPLAN produces the optimal planning according to the optimal solution graph. The correctness and the efficiency of both algorithms are analyzed.

References

1. Nilsson, N. J.: Principle of Artificial Intelligence. Los Altos, CA: Morgan Kaufmann, 1980.
2. Peachey, D. R., McCalla, G. I.: Using planning techniques in intelligent tutoring systems. International Journal on the Man-Machine Studies, 1986(24), 77-98.
3. Murray, W. R.: Control for intelligent tutoring systems: a blackboard-based dynamic instructional planner. AI Communications, 1989, 2(2), 41-57.
4. Katz, S., Lesgold, A., Eggan, G., Gordin, M., Greenberg, L.: Self-adjusting curriculum planning in Sherlock II. Computer Assisted Learning. 4th International Conference, ICCAL '92 Proceedings, 1992, 343-355.
5. Liu, Y. W., Lai, W., Keong, W. K.: Multi-level navigation for curriculum planning in intelligent tutoring systems. 1997 IEEE International Conference on Intelligent Processing Systems (Cat. No.97TH8335), 1997(2), 1151-1154.
6. Elorriaga, J. A. Instructional planning in intelligent evolutive tutoring systems from a case-based reasoning approach. AI Communications, 1999, 12(4), 259-260.
7. Martens, A. and Uhrmacher, A. M. Adaptive tutoring processes and mental plans. Intelligent Tutoring Systems. 6th International Conference, ITS 2002. Proceedings, 2002, 71-80.
8. Li, Y. C., Zhang, X. Z.: Instruction pattern scheduling in Multi-instruction-pattern ITS. Journal of Southwest China Normal University (Natural Science), 2003, 28(6), 877-881.
9. Duan, Z. H.: A model of intelligence computer assisted instruction course ware. Journal of Shaoguan University (Natural Science), 2000(4), 37-41.
10. Jiang Y. F.: Intelligent tutorial planning based on knowledge structure graph. Computer research & development, 1998, 35(9), 787-792.
11. Rambally, G.: A theory for adaptive personalization in web-based courses. Proceedings of the International Conference on Software Engineering Research and Practice, SERP'04, 2004, p 42-48