Carl v. Ossietzky
UNIVERSITÄT
OLDENBURG

BERICHTE

AUS DEM FACHBEREICH INFORMATIK

Herausgeber: Die Professorinnen und Professoren des Fachbereichs Informatik

Flowerfree Finding of Maximum Matchings

Günther Stiege

Bericht

Nummer 2/12 - June 2012
ISSN 0946-2910

Carl v. Ossietzky
UNIVERSITÄT
OLDENBURG

BERICHTE

aus dem

FACHBEREICH INFORMATIK

Herausgeber: Die Professorinnen und Professoren
des Fachbereichs Informatik

Flowerfree Finding of Maximum Matchings

Günther Stiege

Abteilung für Betriebssysteme und Verteilte Systeme

Bericht

BERICHT Nummer 2/12 - June 2012
ISSN 0946-2910

Author's address:

Prof. em. Dr. Günther Stiege
Universität Oldenburg
Graphen und Netzwerke
D-26111 Oldenburg, Germany
www-bvs.informatik.uni-oldenburg.de

This report is available from `www-bvs.informatik.uni-oldenburg.de`

Bibliographische Angaben:
Stiege, Günther:
Flowerfree Finding of Maximum Matchings / Stiege, Günther – Oldenburg: Universität Oldenburg, 2012
(Berichte aus dem Department für Informatik Nr. 2/12)

ISSN 0946-2910

Flowerfree Finding of Maximum Matchings

Günther Stiege
Universität Oldenburg
(June 2012)

Abstract. Actual algorithms for finding maximum matchings are based on Edmonds' techniques of shrinking and expanding circuits of odd length (blossoms and flowers). Here a new approach based on a modified alternating depth-first search is presented. It uses coupled recursion.

Kurzfassung. Derzeitige Algorithmen zum Finden maximaler Korrespondenzen basieren auf Edmonds' Techniken zum Schrumpfen und Expandieren von Kreisen ungerader Länge (Blüten und Blumen). Hier wird eine neue Lösung mit einer modifizierten alternierenden Tiefensuche vorgestellt. Sie benutzt gekoppelte Rekursion.

Categories and Subject Descriptors: E.1 [**Data Structures**] (Graphs and Networks).

General Terms: Algorithms

Additional Keywords and Phrases: maximum matching

Contents

1 Introduction

The problem of finding a maximum matching in general graphs is important, interesting, and much discussed. It is central to discrete mathematics and algorithmic graph theory. In 1965 Edmonds [Edmo1965] published an algorithm to find such matchings. Although several improvements on the efficency have been made since, Edmonds' main ideas still constitute the core of every algorithm for finding maximum matchings. To overcome the difficulties of odd length paths, Edmonds shrunk these paths to single entities (blossoms) which afterwards were expanded again (flowers). All actual algorithms for finding maximum matchings use the technique of shrinking and expanding.

Here a new solution is proposed. The main idea is: If vertex u can be reached from vertex v in an even number of steps and a first attempt yields an odd number of steps than there must be another path from v to u with an even number of steps. The problem is to find it with sufficient efficiency. The solution lies in allowing to traverse each line twice, once in each direction.

This report is organized as follows:
Section 2 summarizes the prerequisites. These are at the one hand results on augmenting paths. On the other hand, some implementation details of storing and traversing graphs are described.

Section 3 presents a motivating example.

In section 4 the algorithm is presented. It is proved that it does what it is supposed to do. Some remarks on its complexity conclude the section.

2 Prerequisites

2.1 Augmenting Paths

We assume simple graphs, i. e. undirected graphs without loops and without multiple edges. A path in a graph is a sequence of vertices where each vertex is joined to the next by an edge. The correponding sequence of edges defines the path as well. A path is open if the first vertex differs from the last. A path is simple if no vertex occurs more than once. A matching is a set of mutually non-adjacent edges. A matching is saturated if no addition of an edge yields a greater matching. A matching is maximum if no other matching has a greater cardinality. Not every saturated matching is maximum. We will speak of *matching edges* and *non-matching edges*.

The central tool for finding maximum matchings are augmenting paths. A path is an *alternating path* if it is open and simple and if a matching edge is always followed by a non-matching edge and vice-versa. An *augmenting path* is an alternating path whose first edge and last edge are non-matching edges. As easily can be seen, changing matching edges to non-matching edges and vice-versa along the path yields a new matching with one more edge. The existence of an augmentig path is sufficient to make a matching larger. A famous theorem due to Berge [Berg1957a] and known as Berge's lemma states that the existence of an augmenting path is also a necessary condition for a matching not to be maximum.

A vertex is *exposed* with respect to a matching if none of its edges is a matching edge. The vertex is *covered* with respect to the matching if it is incident with a matching edge. The first vertex as well as the last vertex of an augmenting path are exposed vertices. If an exposed vertex is such, that no augmenting path starting in it can be found, then no posterior augmenting path will pass through that vertex. The vertex is classified as *frustrated* and need not be considered in the sequel.

2.2 Data Structures

We assume the graph represented by an incidence list structure. There are a list of vertices and a list of edges. Vertices as well as edges have identifying names. The graph structure is given by an incidence list adjoint to each vertex. For the matching algorithm described in the following section it is convinient to store some information in the vertex records and in the edge records. We have three fields (`vtattr`, `vttype` and `vtactive`) for each vertex and one field (`edtype`) for each edge.

`vttype`	E	exposed
	C	covered
	F	frustrated
`vtattr`	U	unvisited
	E	even visit only
	O	odd visit only
	EO	even visit as well as odd visit
`vtactive`	FALSE	vertex not active
	TRUE	vertex active
`edtype`	FALSE	non-matching line
	TRUE	matching line

Table 1: *Data fields for algorithm* `FFMAXMAT`

3 A Motivating Example

Figure 1 shows a graph in which we want to find an augmenting path. We have found

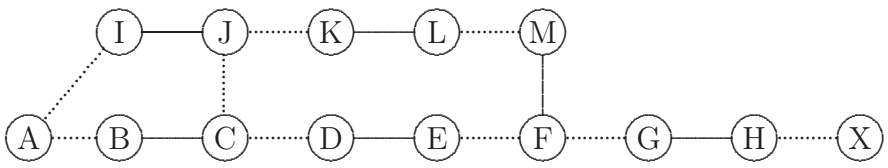

Figure 1: *Graph* `matchgraph`

already a matching in the graph. Solid lines indicate matching edges. Non-matching

edges are characterized by dotted lines. There are two exposed vertices in the graph, namely A and X. We want to find an augmenting path from A to X. To this end, we start an alternating depth-first search in A. There are several different runs of such a search. In our example, the choice of the first edge traversed is critical. In table 2 we see what

Depth-first search starting with edge AC								
AB AI	BC	CJ CD	JI DE	IA EF	FM	ML	LK	KJ
Depth-first search starting with edge AI								
AI	IJ	JC JK	CB KL	BA LM	MF	FE FG	ED GH	DC HX!

Table 2: *Different runs of an alternating depth-first search*

happens. In the upper part the dfs starts with edge AB, in the lower part it starts with edge AI.
Upper part. The first branch stops with edge IA since vertex A is still active. The second branch stops with edge KJ because vertex J had been visited before. The third and last branch stops immediately with edge AI since vertex I has also been visited previously. No augmenting path has been found.
Lower part. The first branch stops with line BA since vertex A is active. The second branch stops with edge DC because vertex C has been visited before. Finally, the last branch reaches vertex X along edge HX. An augmenting path has been detected and the procedure stops.

What now? The solution lies in differentiating the two cases where a branch is stopped. The first case occurs when the next vertex to be visited by the depth-first search is still active. This occurs along a "backward arc" and closes a circuit. No circuit can ever be part of an augmenting path and therefore the depth-first search always has to return. However, as can be seen in table 2 the depth-first search also returns at vertices which are not active anymore. They have been visited before and are visited again along so called "forward arcs". These do not close circuits and there is no reason which forces stopping the search. A closer look at this case reveals that there are two kinds of visits to a vertex: *even visits* when the vertex is reached after an even number of steps and *odd visits* when the vertex is reached after an odd number of steps.

We now modify the alternating depth-first search in the following way. We allow an even visit to a vertex if it is as yet unvisited or if it had only an odd visit. We allow an odd visit in the corresponding situation. We do not allow multiple visits of the same kind because nothing new happens after such a visit. To illustrate the modified dfs we use again the graph of figure 1. Table 3 shows a run of the modified alternating depth-first search. An entry like BC followed by an entry e means that line BC is traversed and an even visit to C is made. In the table again we see two depth-first search trees. The first

Depth-first search with visit marking e and o																
AB	o	BC	e	CJ	o	JI	e	IA								
				CD	o	DE	e	EF	o	FM	e	ML	o	LK	e	KJ
AI	o	IJ	e	JC	o	CB	e	BA								
				JK	o	KL	e	LM	o	MF	e	FE	o	ED	e	DC
												FG	o	GH	e	HX!

Table 3: *Run of a modified alternating depth-first search*

starts with edge `AB` and the second with edge `AI`. The first tree has two branches. The first branch ends with edge `IA` since `A` is active. The second branch ends with edge `KJ` for `J` though actually inactive had a previous odd visit. The tree starting with edge `AI` is traversed next. The first branch traverses vertices which all have been visited already but with opposite even/odd characteristics and stops with active vertex `A`. The second branch passing again through a number of vertices with opposite even/odd marking eventually stops at vertex `C` who had previously visits of both kinds. Finally, the last branch passes through some vertices already visited and three unvisited vertices until reaching vertex `X`. An augmenting path has been found!.

4 Algorithm FFMAXMAT

4.1 Description of the Algorithm

In the following algorithm `FFMAXMAT`[1] is presented. The algorithm uses the modified alternating depth-first search approach. It consists of a main program `FFMAXMAT` and two recursively coupled routines, `evenvisit` and `oddvisit`. The programs are shown in tables 4, 5 and 6. They are formulated in a C-like pseudocode.
At the beginning all vertices are exposed and all edges are non-matching.
`FFMAXMAT:` This is the main loop. In it the list of all vertices is processed. For each vertex the indicators `vtattr` and `vtactive` of all vertices are reset (statement 3). Then it is checked whether the actual vertex is exposed and whether there is at least one other exposed vertex (statement 4). If so, then an alternating depth-first search is started in that vertex. This is done by calling the routine `evenvisit` (statement 5). In a coupled recursion routines `evenvisit` and `oddvisit` perform the modified alternating depth-first search and return eventually with an augmenting path or with the notification that there is none.

`evenvisit:` It expects a vertex and an edge as input parameters. The vertex is the vertex visited. The line, called *entryline*, is the line along which the vertex is entered. If the vertex is active (statement 2) or there has been a previous even visit to it (statement 3), the routine returns. Otherwise, all edges incident with the vertex are checked by calling `oddvisit` (statement 7). If this call was successful, `entryline` is changed to non-

[1] `FFMAXMAT` stands for "flowerfree maximum matching".

```
    FFMAXMATCH()

1   VERTEX   vtx;
2   forall (vertices v)
3      { reset vtattr; reset vtaktive;
4        if  ( v is exposed && there are more than one exposed vertices left)
5          { vtx = evenvisit(v, NULL);
6            if (vtx != NULL) mark v and vtx as covered;
7            else mark v as frustrated;
8          }
9      }
```

Table 4: *Main loop of algorithm FFMAXMAT*

matching line and the end vertex of the augmenting path is returned. If none of the calls were successful, the vertex is set inactive and NULL is returned.

`oddvisit:` Input parameters are a vertex v and an edge `entryline`. At first it is tested whether v is inactive and an odd visit is allowed, If so, the corresponding fields are set. Then it is checked if v is exposed. If it is, we have found an augmenting path. `entryline` becomes a matching line and v is returned as end point of the augmenting path. If v is not exposed it must be covered, for it cannot be frustrated. In this case there must exist a matching line incident with v and different from `entryline`. The routine `findmed` determines this edge. Along this edge `evenvisit` is called and the alternating depth-first search continued. Depending on the value returned, we know that an augmenting path has been found and its end vertex returned or that no augmenting path exists.

4.2 Correctness and Efficiency of the Algorithm

4.2.1 Correctness

Theorem 4.1 *There is an augmenting path starting in exposed vertex v if and only if algorithm `FFMAXMAT` finds one.*

Proof: It is clear, that the algorithm will not find an augmenting path if there is none. Assume

$$v = v_0, v_1, v_2, \ldots v_i, v_{i+1}, \ldots, v_n = u$$

is an augmenting path from exposed vertex v to exposed vertex u. This path determines an even/odd characteristic of all its vertices. We shall say that algorithm `FFMAXMAT` visits vertex v_i of the path *correctly* if the vertex is visited with its path-specific even/odd characteristic. The algorithm visits vertex v_0 correctly. If it has visited vertex v_i correctly and the vertex is of odd visit then either $v_i = v_n$ or the next visit is even and leads to v_{i+1}. If v_i is of even visit, the all its covered neighbors are checked. If no augmenting

```
VERTEX    *evenvisit(VERTEX *v, EDGE *entryline)

1   VERTEX    *vtend, *vtx;
2   if  (v->vtactive)  return NULL;
3   if  (v->vtattr == E || v->vtattr == EO) return NULL; // Previous even visit
4   v is set active;
5   update v->vtattr to E or EO;
6   forall (edges ed incident with v and different from entryline)
7       { vtend = oddvisit(otherend(ed, v), ed);
8         if  (vtend != NULL) // Augmenting path found
9           {if (entryline != NULL)  change entryline to non-matching edge;
10           return vtend;
11          }
12      }
13  set v not active;
14  return NULL;
```

Table 5: *Routine* `evenvisit`

path is found beforehand v_{i+1} will be the next vertex visited correctly. Hence either an augmenting path is found beforehand or eventually v_n is found correctly and thus the initial augmenting path is found. □

4.2.2 Efficiency

In each dfs run every edge is processed at most twice. So, the worst case complexity of the depth first search is $O(m)$ where m is the number of edges. This is still true if changing edges to a new matching is added. In the worst case the dfs has to be run n times where n is the number of vertices. As a result we have $O(nm)$ as the worst case complexity of the new algorithm.

4.3 Concluding Remarks

General Graphs. The preceding definitions and proofs are formulated for undirected graphs without loops and multiple lines. However, as can easily be seen, everything works as well if general graphs are used. The modified alternating depth-first search has to be an a-depth-first search.

Initial Matching. As with other matching finding algorithms, it will be useful not to start `FFMAXMAT` with an empty matching. We should start with a matching as large as possible, for instance with a saturated matching found by a greedy algorithm. Then, the lines of that matching have to be marked as such in the beginning. In addition, their incidence points have to be marked as covered.

```
VERTEX    *oddvisit(VERTEX *v, EDGE *entryline)

1   VERTEX    *vtend, *vtx;
2   EDGE    *med;
3   if  (v->vtactive) return NULL;
4   if  (v->vtattr == O || v->vtattr == EO) return NULL; // Previous odd visit
5   set v active;
6   update v->vtattr to O or EO;
7   if  (v is exposed) // augmenting path found
8     {change entryline to matching edge;
9      return v;
10    }
11  else
12    {med = findmed(v);
13     vtx = evenvisit(otherside(med, v), med);
14     if  (vtx != NULL) // augmenting path found
15      { change entryline to matching edge;
16        return vtx;
17      }
18     set v not active;
19     return NULL;
20    }
```

Table 6: *Routine* `oddvisit`

References

Pages where cited are in parentheses.

[Berg1957a] Berge, Claude. Two theorems in graph theory. *Proceedings of the National Academy of Sciences (USA)*, 43:842–844, 1957. *(2)*

[Edmo1965] Edmonds, Jack. Paths, Trees and flowers. *Canadian Journal of Mathematics*, 17:449–467, 1965. *(2)*

Technical Reports

Fakultät II, Department für Informatik, Universität Oldenburg,
Postfach 2503, 26111 Oldenburg, Germany

1/87 A. Viereck: *Klassifikationen, Konzepte und Modelle für den Mensch-Rechner-Dialog* (Dissertation)

2/87 A. Schwill: *Forbidden subgraphs and reduction systems: A comparison*

3/87 J. Kämper: *Non-uniform proof systems: A new framework to describe non-uniform and probabilistic complexity classes*

1/88 K. Ambos-Spies, H. Fleischhack, H. Huwig: *Diagonalizing over deterministic polynomial time*

2/88 A. Schwill: *Shortest edge-disjoint paths in geodetically connected graphs*

3/88 V. Claus, U. Lichtblau (Hrsg.): *1. Tagung zur Küsten-Informatik*

1/89 U. van der Valk: *Einige Entscheidbarkeits- und Unentscheidbarkeitsresultate für Klassen von S/T-Netzen unter Maximum Firing Strategie und unter Prioritätenstrategien*

2/89 J. Kämper: *Strukturelle Untersuchungen im Umfeld der Komplexitätsklassen P und NP unter besonderer Berücksichtigung nichtuniformer, probabilistischer und disjunktiv selbstreduzierender Algorithmen* (Dissertation)

3/89 J. Kämper: *Nondeterministic oracle Turing machines with maximal computation paths*

1/90 A. Schwill: *Shortest edge-disjoint paths in graphs* (Dissertation)

2/90 K.R. Apt, E.-R. Olderog: *Using transformations to verify parallel programs*

3/90 U. Lichtblau: *Flußgraphgrammatiken* (Dissertation)

4/90 K.R. Apt, E.-R. Olderog: *Introduction to program verification*

5/90 H. Jasper: *Datenbankunterstützung für Prolog-Programmierumgebungen* (Dissertation)

1/91 F. Korf: *Net-based efficient simulation of AADL specifications*

2/91 S.V. Krishnan, C. Pandu Rangan, A. Schwill, S. Seshadri: *Two disjoint paths in chordal graphs*

3/91 H. Eirund: *Modellierung und Manipulation multimedialer Dokumente* (Dissertation)

4/91 G. Schreiber: *Ein funktionaler Äquivalenzbegriff für den hierarchischen Entwurf von Netzen*

1/92 A. Viereck (Hrsg.): *Ergebnisse der 11. Arbeitstagung, Mensch-Maschine Kommunikation*

2/92 P. Gorny, U. Daldrup, H. Schwab: *Zwischenbilanz: Menschengerechte Gestaltung von Software*

3/92 E.-R. Olderog, St. Rössig, J. Sander, M. Schenke: *ProCoS at Oldenburg: The Interface between Specification Language and occam-like Programming*
Language

4/92 F. Korf: *Synthesis of VHDL Test Environments form Temporal Logic Specifications*

5/92 W. Kowalk: *Konstruktorentechnik: Neue Methoden zur Mengenrechnung,*
Logikrechnung und Intervallrechnung

1/93 Ch. Dietz, G. Schreiber: *Eine Termdarstellung für S/T-Netze*

2/93 J. Sauer: *Wissensbasiertes Lösen von Ablaufplanungsproblemen durch explizite Heuristiken*

3/93 M. Sonnenschein, U. Lichtblau (Hrsg.): *6. Kolloquium der Arbeitsgruppe Informatik-Systeme*

4/93 H. Fleischhack, U. Lichtblau, M. Sonnenschein, R. Wieting: *Generische Definition hierarchischer zeitbeschrifteter höherer Petrinetze*

5/93 F. Köster, L. Twele, R. Wieting, W. Ziegler: *Fallbeispiele zur Modellierung mit THOR-Netzen*

1/94 R. Götze: *Dialogmodellierung für multimediale Benutzerschnittstellen*

2/94 B. Müller: *PPO – Eine objektorientierte Prolog-Erweiterung zur Entwicklung wissensbasierter Anwendungssysteme*

3/94 W. Damm/A. Mikschl: *Projekt Entwurf und Implementierung eines Multi-threaded RIS C-Prozessors*

4/94 S. Rössig: *A Transformational Approach to the Design of Communicating Systems* (Dissertation)

5/94 G. Schreiber: *Funktionale Äquivalenz von Petri-Netzen* (Dissertation)

1/95 A. Gronewold, H. Fleischhack: *Language Preserving Reductions of Safe Petri-Nets*

2/95 H. Reineke: *Struktur und Verhalten von verteilten endlichen Automaten* (Dissertation)

3/95 H. Behrends: *Beschreibung ereignisgesteuerter Aktivitäten in datenbankgestützten Informationssystemen* (Dissertation)

4/95 U. M. Levens: *Computerunterstütztes Modellieren von Musikstücken mit Petri-Netzen: Das Mailänder Konzept*

1/96 M. Burke: *FDDI und ATM in multimedialen Anwendungsumgebungen* (Dissertation)

2/96 I. Pitschke: *Interaktive Rekonstruktion geometrischer Modelle aus digi-*

1/98 S. Kleuker: *Inkrementelle Entwicklung von verifizierten Spezifikationen für verteilte Systeme* (Dissertation)

2/98 J. Bohn: *Mechanical Support and Validation of a Design Calculus for Communicating Systems by a Logic-Based Proof System* (Dissertation)

3/98 L. Köhler: *Fuzzy Geometrie und Anwendungen in der medizinischen Bildverarbeitung* (Dissertation)

4/98 J. Helbig: *Linking Visual Formalisms: A Compositional Proof System for*
Statecharts Based on Symbolic Timing Diagrams (Dissertation)

5/98 G. Stiege: *Edge Partitions in Undirected Graphs*

6/98 A. Gerns: *Entwicklung und Bewertung von Objektmigrationsstrategien für*
verteilte Umgebungen

7/98 M. Stadler: *Abstrakte Rechnernetzmodelle als Grundlage einer umfassenen Automatisierung des Netzmanagements – Konzepte und Sprachen zu ihrer Umsetzung* (Dissertation)

8/98 M. S. Steiner: *Lastverteilung in heterogenen Systemen*

9/98 Clemens Otte: *Fuzzy-Protopyp-Klassifikatoren und deren Anwendung zur automatischen Merkmalsselektion*

1/99 Juliane Vorndamme: *Die Auswirkungen rechtlicher Verpflichtungen auf die Softwareentwicklung*

2/99 Eike Best, Kerstin M. Richter: *Relational Semantics Revisited*

3/99 Jung Sun Lie: *Einsatz von Objektmigrationsstrategien zur Leistungssteigerung in verteilten Systemen*

4/99 Fachbereich Informatik: *Zwei-Jahresbericht des Fachbereichs Informatik (1.10.1996 – 30.9.1998)*

5/99 Ingo Stierand, Olaf Maibaum, Björn Briel und Günther Stiege: *Cassandra – Generierung, Analyse und Simulation von eingebetteten Multiprozessor-Echtzeitsystemen*

6/99 Gunnar Wittich: *Ein problemorientierte Ansatz zum Nachweis von Realzeiteigenschaften eingebetteter Systeme*

7/99 Annegret Habel, Jürgen Müller, Detlef Plump: *Double-Pushout Graph Transformation Revisited*

8/99 Ingo Stierand: *Eine Konfigurationssprache zur Erstellung von Ambrosia/MP-Systemen*

9/99 Igor V. Tarasyuk: *Equivalences for Concurrent and Distributed Systems*

10/99 Eike Best, Alexander Lavrov: *Generalised Composition Operations for High-Level Petri-Nets*

11/99 Alexander Lavrov: *Enhancing Mixed Nonlinear Optimisation: A Hybrid Approach*

12/99 Alexander Lavrov: *Hybrid Techniques in Discrete-Event System Modelling and Control: Some Examples*

13/99 Eike Best, Raymond Devillers, Maciej Koutny: *Recursion and Petri Nets*

14/99 Eike Best, Raymond Devillers, Maciej Koutny: *The Box Algebra = Petri Nets + Process Expressions*

15/99 Eike Best, Harro Wimmel: *Reducing k-safe Petri Nets to Pomset-Equivalent 1-safe Petri Nets*

16/99 Udo Brockmeyer: *Verifikation von STATEMATE Designs* (Dissertation)

1/00 Henning Dierks: *Specification and Verification of Polling Real-Time Systems* (Dissertation)

2/00 Clemens Fischer: *Combination and Implementation of Process and Data: From CSP-OZ to Java* (Dissertation)

3/00 Cheryl Kleuker: *Constraint Diagrams* (Dissertation)

4/00 Thomas Thielke: *Linear-algebraische Methoden zur Beschreibung, Verfeinerung und Analyse gefärbter Petrinetze* (Dissertation)

1/01 Günther Stiege: *Higher Decompositions in Undirected Graphs*

2/01 Ute Vogel: *Zwei-Jahres-Bericht*

3/01 Joseph Tapken: *Model-Checking in Duration Calculus Specifications* (Dissertation)

4/01 Björn Briel: *Analyse eingebetteter Systeme mittels verteilter Simulation* (Dissertation)

5/01 Günther Stiege: *Standard Decomposition and Periodicity of Digraphs*

6/01 Ingo Stierand: *Ambrosia/MP – Ein Echtzeitbetriebssystem für eingebettete Mehrprozessorsysteme*

1/02 Giorgio Busatto, Annegret Habel: *Improving the Quality of Hypertexts Using Graph Transformation*

2/02 Giorgio Busatto: *Modeling Hyperweb Dynamics through Hierarchical Graph Transformation*

3/02 Giorgio Busatto: *An Abstract Model of Hierarchical Graphs and Hierarchical Graph Transformation* (Dissertation)

4/02 Laila Kabous: *An Object Oriented Design Methotology for Hard Real-Time Systems: The OOHARTS Approach* (Dissertation)

1/03 Ute Vogel: *2-Jahres-Bericht*

2/03 Olaf Maibaum: *Bestimmung symbolischer Laufzeiten in eingebetteten Echtzeitsystemen* (Dissertation)

3/03 Günther Stiege, Ingo Stierand: *Connectedness-Based Hierarchical Decomposition of Undirected Graphs* (Bericht)

4/03 Willi Hasselbring, Susanne Petersen: *Standards f"ur medizinische Kommunikation und Dokumentation* (Bericht)

5/03 Andreas Möller: *Eine virtuelle Maschine f"ur Graphprogramme* (Bericht)

6/03 Tom Bienmüller: *Reducing Complexity for the Verification of Statemate Designs* (Bericht)

7/03 Sandra Steinert: *Graph Programs for Graph Algorithms* (Bericht)

8/03 Jochen Klose: *Live Sequence Charts: A Graphical Formalism for the Specification of Communication Behaviour* (Dissertation)

1/04 Jens Oehlerking: *Transformation of Edmonds' Maximum Matching Algorithm into a Graph Program* (Bericht)

2/04 Sergej Alekseev: *Dienste Intelligenter Netze / Graphentheoretische Methoden in der Kontrollflu"sanalyse* (Bericht)

3/04 Giorgio Busatto: *GraJ: A System for Executing Graph Programs in Java* (Bericht)

1/05 Sergej Alekseev, Johannes Wust: *Graph Theoretical Methods in the Control Flow Analysis of Object Oriented Real Time Software (Bericht)*

2/05 Ute Vogel: *2-Jahres-Bericht* (Bericht)

3/05 Igor Tarasyuk: *Discrete time stochastic Petri box calculus* (Bericht)

1/06 Henning Dierks: *Time, Abstraction and Heuristics* (Habilitation)

2/06 Li Sek Su: *Full-Output Siphons and Deadlock-Freeness for Free Choice Petri Nets* (Bericht)

3/06 Timo Warns: *Solving Concensus Using Structural Failure Models* (Bericht)

4/06 Sergej Alekseev: *Ablaufanalyse objektorientierter Echtzeitanwendungen mit graphentheoretischen Methoden* (Dissertation)

5/06 Li Sek Su: *Some Considerations on the Foundation of NP-Completeness Theory* (Bericht)

6/06 Li Sek Su: *Semitraps and Deadlock-Freeness for Reduced Asymmetric Choice Nets* (Bericht)

7/06 Li Sek Su: *Algorithms of computing the Deadlock Markings Sets for Petri Nets* (Bericht)

8/06 Annegret Habel, Karl-Heinz Pennemann, Arend Rensink: *Weakest Preconditions for High-Level Programs (Long Version)* (Bericht)

9/06 Jochen Hoenicke: *Combination of Processes, Data, and Time* (Dissertation)

10/06 Steffen Becker, Marco Boscovic, Abhishek Dhama, Simon Giesecke, Jens Happe, Wilhelm Hasselbring, Heiko Koziolek, Henrik Lipskoch, Roland Meyer, Margarethe Muhle, Alexandra Paul, Jan Ploski, Matthias Rohr, Mani Swaminathan, Timo Warns, Daniel Winteler: *Truthworthy Software Systems: A Discussion of Basic Concepts and Teminology* (Bericht)

11/06 Christian Zuckschwerdt: *Ein System zur Transformation von Konsistenz-und Anwendungsbedingungen* (Bericht)

01/07 Andreas Schäfer: *Specification and Verification of Mobile Real-Time Systems* (Dissertation)

02/07 Günther Stiege: *General Graphs* (Bericht)

03/07 Wolfgang Kowalk: *Integralrechnung* (Bericht)

04/07 Karl Azab, Karl-Heinz Pennemann: *Type Checking C++ Template Instantiation by Graph Programs* (Bericht)

01/08 Roland Meyer: *On depth and breath in the Pi-Calculus* (Bericht)

02/08 Ingo Brückner: *Slicing Integrated Formal Specifications for Verification* (Dissertation)

03/08 Ute Vogel: *2-Jahres-Bericht 2004 – 2006* (Bericht)

04/08 Günther Stiege: *Summierbare Familien* (Bericht)

05/08 Igor V. Tarasyuk: *Investigating equivalence realations in dtsPBC* (Bericht)

01/09 Elke Wilkeit: *2-Jahres-Bericht, 01.10.2006 – 3.09.2008* (Bericht)

02/09 Roland Meyer: *Structural Stationarity in the pi-Calculus* (Dissertation)

03/09 InformatikerInnen des Moduls Soft Skills: *E-Book Soft Skills* (Bericht)

04/09 Eike Best: *Separability in Persistent Petri Nets* (Bericht)

01/10 Igor V. Tarasyuk: *Equivalence relations for behaviour-perserving reduction and modular performance evaluation in dtsPBC* (Bericht)

02/10 Roman Dubtsov: *Time Trannsition Systems with Independence and Marked Sott Domains: an Adjunction* (Bericht)

01/11 Elena S. Oshevskaya: *Matching Equivalences on Higher Dimensional Automata Models* (Bericht)

02/11 Elke Wilkeit: *2-Jahres-Bericht, 01.10.2008 – 30.09.2010* (Bericht)

03/11 Johannes Faber: *Verification Architectures for Complex Real-Time Systems* (Dissertation)

04/11 Igor V. Tarasyuk: *Eqivalences for modular performance analysis in dtsPBC* (Bericht)

01/12 Irina Virbitskaite, Nataliya Gribovskaja, Eike Best: *Some Evidence on the Consistency of Categorical Semantics for Time Interleaving Behaviours* (Bericht)

1/12 Günther Stiege: *Playing with Knuth's words.dat* (Bericht)

02/12 Günther Stiege: *Flowerfree Finding of Maximum Matchings* (Bericht)

03/12 Günther Stiege: *f-Euler Tours in General Graphs* (Bericht)